Создание эффективных веб-страниц

6-е издание

HTML и XHTML

Подробное руководство

Символ®

O'REILLY®

Чак Муссиано и Билл Кеннеди

HTML & XHTML

The Definitive Guide

Sixth Edition

Chuck Musciano and Bill Kennedy

O'REILLY®

HTML и XHTML

Подробное руководство

Шестое издание

Чак Муссиано и Билл Кеннеди

Санкт-Петербург — Москва
2008

Чак Муссиано и Билл Кеннеди

HTML и XHTML. Подробное руководство, 6-е издание

Перевод С. Иноземцева

Главный редактор	*А. Галунов*
Зав. редакцией	*Н. Макарова*
Выпускающий редактор	*В. Овчинников*
Научный редактор	*О. Цилюрик*
Редактор	*Е. Орлова*
Корректоры	*О. Макарова, С. Минин*
Верстка	*Д. Орлова*

Муссиано Ч., Кеннеди Б.

HTML и XHTML. Подробное руководство, 6-е издание. – Пер. с англ. – СПб: Символ-Плюс, 2008. – 752 с., ил.

ISBN-10: 5-93286-104-5

ISBN-13: 978-5-93286-104-2

Шестое издание «HTML и XHTML» – самая полная и современная книга по языкам HTML и XHTML, разъясняющая работу и взаимодействие каждого их элемента. Она удачно сочетает в себе лучшие качества понятного учебного пособия, адресованного начинающим, и всеобъемлющего справочника, который всегда под рукой даже у опытных веб-программистов. Этот труд, ставший классическим, содержит все от базового описания синтаксиса и семантики до практических советов, поможет вам найти свой неповторимый стиль и в совершенстве овладеть языком веб-дизайна.

Описаны стандарты HTML 4.01, XHTML 1.0 и CSS2, приведен обзор еще не вступивших в силу стандартов XHTML 2 и CSS3. Уделено внимание и новейшим инициативам разработчиков XHTML (XFroms, XFrames и модуляризации), а также основам XML. Рассмотрены: управление внешним видом документа с помощью таблиц стилей; работа с HTML-кодом, сгенерированным автоматически; работа с фреймами, интерактивными формами, динамическими документами; интеграция HTML-кода с мультимедийными данными, сценариями JavaScript и Java-апплетами.

ISBN-10: 5-93286-104-5
ISBN-13: 978-5-93286-104-2
ISBN 0-596-52732-2 (англ)

Издательство «Символ-Плюс». 199034, Санкт-Петербург, 16 линия, 7,
тел. (812) 324-5353, www.symbol.ru. Лицензия ЛП N 000054 от 25.12.98.
Налоговая льгота – общероссийский классификатор продукции
ОК 005-93, том 2; 953000 – книги и брошюры.

Подписано в печать 21.02.2008. Формат 70×100 1/16 . Печать офсетная.
Объем 47 печ. л. Тираж 2000 экз. Заказ N
Отпечатано с готовых диапозитивов в ГУП «Типография «Наука»
199034, Санкт-Петербург, 9 линия, 12.

Эту книгу мы посвящаем нашим женам
Синди, Кортни и Коул
и детям Жанне, Еве и Этан.

Если бы не их любовь и терпение,
то у нас не было бы ни сил ни времени,
чтобы написать ее.

Оглавление

Предисловие

Изучение языка разметки гипертекста (Hypertext Markup Language, HTML) и расширяемого языка разметки гипертекста (Extensible Hypertext Markup Language, XHTML) подобно изучению любого другого языка – естественного или компьютерного. Многие учащиеся первым делом углубляются в примеры. Изучение чужих работ – это естественный, легкий и приятный способ обучения. Поэтому мы советуем всякому, кто хочет изучить HTML и XHTML, подключиться к Сети и с помощью какого-либо броузера своими глазами увидеть, что выглядит хорошо, что эффективно работает и что может оказаться полезным. Изучайте чужое творчество и размышляйте над открывающимися возможностями. Именно подражая работам друг друга, большинство современных веб-мастеров выучило язык.

Подражание, однако, полезно только до известного предела. Образцы могут оказаться и удачными, и не очень. На пути подражания можно познакомиться с языком только в «первом приближении». Чтобы овладеть языком в полной мере, нужно узнать, как использовать его во множестве различных ситуаций. Всему можно научиться на примерах, если жить достаточно долго.

Компьютерные языки отличаются от естественных строгостью синтаксических конструкций, хотя языки разметки намного более снисходительны к пишущему, чем языки программирования. Достаточно одной ошибки в записи – и ничего не будет работать. Кроме того, существует проблема «стандартов». Академические комитеты и промышленные эксперты определяют грамматику и правила употребления компьютерных языков, подобных HTML. Проблема состоит в том, что броузерные технологии, используемые вами и вашей аудиторией, не всегда придерживаются стандартов. Некоторым, например тем, что реализованы в броузерах получающих все более широкое распространение мобильных устройствах, это просто не под силу, а разработчики других создают собственные версии языка, пренебрегая стандартами.

Сами стандарты также меняются. HTML преобразовался в XHTML – HTML с использованием расширяемого языка разметки (Extensible Markup Language, XML). HTML и XHTML настолько похожи, что мы часто говорим о них как об одном языке. Есть между ними и коренные различия, о которых рассказывается далее в этом предисловии.

Полное, отражающее последние нововведения руководство по языку, которое охватывает синтаксис, семантику и детальное описание изменений и поможет отличить хороший текст от плохого, – вот надежный путь усвоения HTML и XHTML.

На пути к овладению языком нужно сделать еще один шаг. Чтобы стать настоящим мастером, нужно выработать свой собственный стиль. Это подразумевает умение выдать не просто работающий код, но код, который будет работать эффективно. «Порядок слов» имеет большое значение, так же как и структурирование материала в пределах документа, нескольких документов или собраний документов.

Наша цель при написании данной книги – помочь овладеть HTML и XHTML, дать вам исчерпывающие сведения о синтаксисе, семантике и правилах хорошего стиля. Мы избираем естественный подход к изучению – через примеры (хорошие, разумеется). Мы детально изучим все элементы современных версий этих языков (HTML 4.01 и XHTML 1.0), так же как и все расширения, поддерживаемые популярными броузерами. Мы объясним, как каждый из элементов работает и как он взаимодействует с другими элементами.

И со всем почтением к Странку и Уайту[1] на протяжении книги мы будем давать советы, касающиеся стиля и организации, чтобы помочь научиться эффективно использовать HTML и XHTML для выполнения различных задач – от создания простой онлайновой документации до сложных маркетинговых и торговых презентаций. Мы покажем вам, что работает, а что нет, расскажем, что существенно для тех, кто смотрит на ваши странички, и предупредим о возможных ошибках.

Короче говоря, эта книга – полное руководство по созданию документов с использованием HTML и XHTML, начиная с основ синтаксиса и семантики и заканчивая разнообразными рекомендациями в отношении стиля, призванное помочь в создании красивых, информативных и доступных документов, которые вы с гордостью сможете представить своим читателям.

Для кого эта книга

Мы написали эту книгу для всех тех, кто интересуется изучением и использованием языков, применяемых в Интернете, от самых обычных пользователей до профессиональных веб-дизайнеров. Мы не предполагаем у читателей никакого опыта в HTML и XHTML. На самом деле мы даже не ожидаем, что вы бывали во Всемирной паутине, хотя удивились бы, если бы вы до сих пор ни разу не попробовали познакомиться с ней. Нет необходимости иметь подключение к Интернету, чтобы чи-

[1] William Strunk Jr. и Elwyn Brooks White – авторы знаменитой книги «The Elements of Style», ставшей «классикой жанра», которую специалисты по стилю называют не иначе как «Strunk & White». – *Примеч. науч. ред.*

тать эту книгу; но если вы не подключены, эта книга будет для вас как путеводитель для домоседа.

Единственное, что вам нужно, – это компьютер, редактор для создания простых ASCII[1]-текстовых файлов и броузеры. Примеры, приведенные в этой книге, были протестированы на последних версиях броузеров Internet Explorer, Mozilla Firefox, Netscape Navigator и Opera от Opera Software ASA. Поскольку HTML- и XHTML-документы хранятся в доступном для всех текстовом формате и поскольку эти языки аппаратно-независимы[2], мы не станем делать никаких предположений относительно того, каким компьютером вы пользуетесь. Броузеры, однако, различаются в зависимости от платформы и операционной системы, так что ваши HTML- и XHTML-документы могут выглядеть совершенно по-разному на разных компьютерах и при использовании различных версий броузеров. Мы объясним, как разные броузеры относятся к определенным элементам языка, уделяя особое внимание их различиям.

Если вы новичок в HTML, WWW или вообще в гипертекстовых документах, то вам следует начать с чтения главы 1. В этой главе мы рассказываем, как соединились все технологии WWW при создании сети взаимосвязанных документов.

Если вы уже знакомы с Интернетом, но не знаете ни HTML, ни XHTML, начните с главы 2. В ней дан краткий обзор наиболее важных особенностей языка, и она может служить схемой, демонстрирующей наши подходы к языку в остальной части этой книги.

Последующие главы касаются конкретных аспектов HTML и XHTML. Изложение в них ведется сверху вниз. Читайте их по порядку, чтобы познакомиться с языком в целом, или перескакивайте от раздела к разделу, чтобы найти то, что вас особенно интересует.

---

[1] ASCII (American Standard Code for Information Interchange) – американский стандартный код для обмена информацией. В данном случае имеется в виду текстовый файл, содержащий только символы в коде ASCII (в отличие, например, от документа Microsoft Word, содержащего кроме текста служебную информацию). Для пользователей Windows одним из таких редакторов (хотя и не самым удобным) является обычный «Блокнот». – *Примеч. науч. ред.*

[2] Под аппаратной независимостью подразумевается тот факт, что документы HTML и XHTML не требуют никакой доработки кода при их переносе на различные типы вычислительной техники (от интеллектуальных мобильных телефонов до суперкомпьютеров) под управлением практически любых операционных систем. Однако при переносе сохраняется только логическая структура и содержание документа, а визуальное представление (размещение и внешний вид элементов) может изменяться. – *Примеч. науч. ред.*

Как устроена эта книга

На протяжении книги мы используем моноширинные шрифты для выделения собственных элементов стандартов HTML/XHTML, тегов и атрибутов. В тегах мы всегда используем строчные буквы.[1] Мы употребляем *курсив*, когда даем определение новых терминов, а также для элементов, необходимых при создании ваших собственных документов, таких как атрибуты тегов и определяемые пользователем строки.

Мы обсуждаем элементы языка на протяжении всей книги, но для каждого из подробно (может быть, даже слишком дотошно) изучаемых элементов можно найти краткое описание в справочном окне, которое выглядит примерно, как приведенное ниже.

<title>

Функция:	Определяет заголовок документа
Атрибуты:	`dirlang`
Закрывающий тег:	`</title>`; присутствует обязательно
Содержит:	*plain_text*
Может содержаться в:	*head_content*

Первая строка в окне содержит название элемента, за которым следует краткое описание его функции. Затем перечислены атрибуты, если они имеются, которые можно или нужно указать в качестве составной части элемента.

Значок  идентифицирует теги и атрибуты, не представленные в стандартах HTML 4.01 и XHTML 1.0, а также те, которые по-разному интерпретируются разными броузерами.

Краткое описание содержит также закрывающий HTML-тег элемента (если он требуется) вместе с указанием, может или нет закрывающий тег быть безопасно опущен при обычном использовании HTML. Для нескольких тегов, которые в HTML не имеют закрывающего тега, но имеют его в XHTML, язык позволяет обозначить это завершение с помощью символа «прямой слэш» (/) в конце тега, например `<br />`. В этих случаях тег может также содержать атрибуты, обозначенные многоточием, например `<br.../>`.

В разделе «Содержит» перечисляются правила грамматики HTML, определяющие элементы, которые могут быть заключены в описываемом теге. Соответственно раздел «Содержится в» перечисляет те пра-

[1] HTML безразличен к регистру в том, что касается имен тегов и атрибутов, но в XHTML строчные и заглавные буквы различаются. Также в некоторых элементах HTML, таких как имена файлов, регистр букв имеет значение, так что будьте осторожны.

вила, которые могут содержать этот тег. Данные правила определены в приложении А.

Наконец, HTML и XHTML в значительной степени пересекаются. В зависимости от контекста вы будете по-разному использовать элементы, но многие из них имеют общие атрибуты. Там, где это возможно, мы помещаем в тексте перекрестные ссылки, отсылающие вас к обсуждению родственных тем в других местах этой книги. Эти ссылки, как та, что находится в конце этого абзаца, являются грубой бумажной моделью гипертекстовой документации, которые могли бы превратиться в настоящие гипертекстовые ссылки, если бы эта книга была представлена в электронном виде. [синтаксис тегов, 3.3.1]

Советуем пользоваться этими ссылками при всякой возможности. Мы будем часто говорить об атрибутах вкратце, полагая, что вы последуете за ссылкой в поисках более подробных сведений. В других случаях ссылка приведет вас к описанию альтернативных вариантов использования обсуждаемого элемента, к советам по его применению или к описанию его стилистических особенностей.

Версии и семантика

Последний стандарт HTML – это версия 4.01, но большая часть изменений и нововведений языка присутствует и в версии 4.0. Поэтому в данной книге мы в основном упоминаем стандарт HTML как HTML 4, подразумевая версию 4.0 и более поздние. Мы точно указываем номер версии, только когда это существенно.

Стандарт XHTML в настоящее время сделал только первый шаг – версия 1.0. Консорциум W3C (World Wide Web Consortium) выпустил рабочий проект второй версии (XHTML 2.0), но этот стандарт еще только устанавливается. В основном XHTML 1.0 совпадает с HTML 4.01. Мы подробно опишем их различия в главе 16. На протяжении этой книги мы будем специально оговаривать случаи, когда XHTML рассматривает какой-либо аспект или элемент языка иначе, чем HTML.

В стандартах HTML и XHTML четко различаются «типы элементов» документа и «теги», размечающие эти элементы. Например, стандарт говорит об элементе типа «абзац», который вовсе не то же, что тег . Элемент типа «абзац» состоит из принятого имени типа элемента в открывающем теге ( ), содержимого элемента (в промежутке между тегами разметки) и тега завершения абзаца ( ). Тег – это открывающий тег для элемента типа «абзац», и его содержимое (атрибуты) воздействует на содержимое элемента типа «абзац».

Хотя это важные различия, на практике именно теги разметки применяются авторами в их документах с целью повлиять на содержимое элемента, если это необходимо. Соответственно и мы в этой книге нивелируем понятия «тип элемента» и «тег», говоря чаще всего о тегах

и их содержимом, не обязательно употребляя при этом термин «тип элемента», более подходящий в данном случае. Простите нам эту вольность, но мы допускаем ее для большей доходчивости изложения.

HTML и XHTML

HTML – не латинский язык, но он достиг преклонного возраста уже к версии 4.01. Консорциум W3C не планирует разработку следующей версии, и об этом объявлено официально. Напротив, HTML получил обоснование и внешнюю форму, став частью расширяемого языка разметки (Extensible Markup Language, XML). Его новое имя – XHTML (Extensible Hypertext Markup Language, расширяемый язык разметки гипертекста).

Появление XHTML – лишь еще одна глава в непростой истории HTML и Интернета, где путаница – это скорее норма, чем исключение. Хуже всего было, когда в период «броузерных войн» между Netscape и Microsoft старейшины W3C, ответственные за разработку и принятие приемлемых стандартов языка, потеряли контроль над языком. Стандарт HTML+ оказался не слишком удачным, а обсуждение HTML 3.0 настолько затянулось, что W3C просто положила весь проект на полку. Что бы ни утверждали в своей литературе предприимчивые деятели рынка, HTML 3.0 так и не появился. Вместо этого в конце 1996 года производители броузеров убедили W3C выпустить стандарт HTML версии 3.2, единственной целью которого было «узаконить» (включить в стандарт) большую часть расширений HTML для ведущего броузера (Netscape).

Доминирование Netscape в качестве ведущего броузера и лидера вебтехнологий практически сошло на нет в конце XX века. К этому времени корпорация Microsoft эффективно встроила Internet Explorer в операционную систему Windows не только в качестве очередного приложения, но и как главную функциональную возможность графического пользовательского интерфейса. Кроме того, броузер Internet Explorer обладал функциональностью (нестандартной на тот момент), которая была принципиально ориентирована на растущий бизнес и маркетинг в Интернете.

К удовольствию тех из нас, кто ценит и твердо поддерживает стандарты, W3C вернул себе ведущую роль, выпустив стандарт HTML 4.0, который в декабре 1999 года сменился на HTML 4.01. Этот стандарт, вобравший в себя многие новшества, предложенные броузерами Netscape и Internet Explorer, более прост и понятен, чем все предыдущие. Он устанавливает четкие модели отображения документов, обеспечивающие переносимость кода между броузерами и платформами, хорошо интегрирован с дружественным стандартом каскадных таблиц стилей (Cascading Style Sheets, CSS). Стандарт CSS предназначен для задания параметров оформления содержимого HTML-документов и дает сред-

ства для альтернативных (невизуальных) клиентских программ, а также для поддержки различных универсальных языков.

В процессе разработки стандарта W3C осознал, что HTML никогда не сможет удовлетворить потребности сетевого сообщества в многообразии способов распространения, обработки и отображения документов. HTML предлагает лишь ограниченный набор примитивов для конструирования документов и решительно не способен справиться с таким нетрадиционным материалом, как химические формулы, музыкальная нотация или математические выражения. К тому же он не способен удовлетворительно поддерживать альтернативные средства отображения, такие как карманные компьютеры или интеллектуальные мобильные телефоны.

Отвечая на эти запросы, W3C разработал стандарт расширяемого языка разметки XML. В стандарте XML описываются принципы построения собственных языков разметки.[1] Причем для реализации таких языков уже не требуется никаких дополнительных действий со стороны W3C. XML-совместимые языки несут информацию, которая в дальнейшем может быть проанализирована, обработана, отображена (и что там еще делают с информацией) посредством разнообразных технологий, появившихся в результате революции в средствах цифровой связи, перевернувшей мир десять лет назад с появлением Интернета. XHTML – это HTML, переработанный с тем, чтобы отвечать стандарту XML. Это язык, на котором основано будущее Сети.

Почему бы просто не отказаться от HTML в пользу XHTML? По многим соображениям. Первое и главное – стандарт XHTML не взял Всемирную паутину штурмом. В создание основанной на HTML документации и в обучение языку вложено уже столько сил и средств, что вряд ли это может произойти в скором времени. Кроме того, XHTML – это переработанный с учетом XML язык HTML 4.01. Выучите HTML 4 – и вы готовы к будущему.[2]

Функциональные возможности, объявленные нежелательными

Одной из непопулярных мер, которую вынуждены предпринимать разработчики стандартов, является отказ от элементов, уже получивших широкое распространение, в пользу элементов, которые разработчики

---

[1] XML, в отличие от HTML, не определяет разметку для какого-либо конкретного применения (в данном случае для формирования WWW-документов), а определяет формальные правила разметки для *любых* целей: в этом и его сила, и его слабость. XHTML, конкретизирующий XML применительно к сфере WWW-документов, и восполняет избыточную общность XML. – *Примеч. науч. ред.*

[2] Мы погрузимся в глубины XML и XHTML в главе 15 «XML» и главе 16 «XHTML».

считают более совершенными. Авторы стандартов HTML и XHTML реализуют эту меру, объявляя «нежелательными»[1] те функциональные возможности, которые не вписываются в генеральную схему развития языка.

Например, тег предписывает броузеру выводить обозначенный им текст в центре окна. В то же время стандарт CSS предоставляет свои средства для отцентровки текста. Консорциум W3C сделал выбор в пользу поддержки CSS и не рекомендует веб-авторам использовать тег , объявив его нежелательным. Возможно, в одной из следующих версий стандарта тег и другие нежелательные элементы и атрибуты будут удалены из языка.

На протяжении всей книги мы специально отмечаем нежелательные HTML-теги и другие элементы, а также постоянно напоминаем об их «нежелательности». Должны ли вы перестать пользоваться ими? И да, и нет.

Да, потому что существует более предпочтительный и, вероятно, лучший способ добиться того же эффекта. Применяя его, вы обеспечиваете долгую жизнь своих документов во Всемирной паутине. Кроме того, инструменты, которыми вы будете подготавливать HTML/XHTML-документы, будут, скорее всего, придерживаться рекомендуемых стандартов. У вас просто не будет выбора, если вы сознательно не откажетесь от этих инструментов. В любом случае, если вы не создаете все документы вручную, вы должны понимать, как работает предпочтительный способ, хотя бы для того, чтобы уметь найти его в сгенерированном коде и исправить.

И все-таки, какими бы убедительными ни были аргументы в пользу отказа от нежелательных элементов и атрибутов, они все еще являются стандартными. Они хорошо поддерживаются большинством броузеров, и их исчезновение в ближайшем будущем остается под вопросом. А поскольку стандарт HTML изменяться не будет, навешивание ярлыков «нежелательности» вводит в заблуждение.

Так что вам не следует излишне беспокоиться о нежелательных функциональных возможностях HTML, и уж тем более нет повода для паники. Просто мы рекомендуем вам сделать шаг вперед по направлению к новым стандартам.

---

[1] Дословно deprecated – нежелательный, устаревший, неподдерживаемый; практика постепенного «перевода» терминов и понятий в категорию deprecated сложилась и в других языковых технологиях, например (раньше, чем в других) в стандартах C/C++ компилятора GCC – «открытого» проекта GNU, и оказалась в высшей степени плодотворной. Состоит она в постепенном (от версии к версии) выведении из обращения терминов и понятий, не нарушающем цельности языка и не порождающем несовместимости версий. – *Примеч. науч. ред.*

Подробное руководство

Парадокс заключается в том, что стандарт HTML 4.01 не является полным и окончательным. Значительная часть возможностей, включенных в последний стандарт HTML, широко используется и поддерживается популярными броузерами. С другой стороны, многие части стандарта игнорируются. Уверяем вас, что вы можете окончательно запутаться.

Мы уже разложили все по полочкам, так что вам не придется потеть, выясняя, что в каком-то броузере работает, а что нет. Таким образом, эта книга является полным и окончательным руководством по HTML и XHTML. Мы сообщаем подробности обо всех элементах стандартов HTML 4.01 и XHTML 1.0, а также о множестве интересных и полезных расширений языка. Мы также включили в книгу подробное обсуждение стандарта CSS, поскольку он неразрывно связан с разработкой веб-страниц.

Кроме того, есть технологии, тесно связанные с HTML, но не являющиеся его частью. Так, например, мы лишь вскользь упоминаем о CGI[1] и Java. CGI- и Java-программы действуют в тесной связи с HTML-документами и работают под управлением броузеров или параллельно с ними, но все же не являются частью языка как такового, так что мы в них не углубляемся. Кроме того, эти темы столь обширны, что заслуживают отдельных книг, таких как David Flanagan «JavaScript: The Definitive Guide», Scott Guelich, Shishir Gundavaram, Gunther Birzneiks «CGI Programming with Perl»[2], Eric Meyer «Cascading Style Sheets: The Definitive Guide» и Pat Niemeyer и Jonathan Knudsen «Learning Java», выпущенных издательством O'Reilly.

Эта книга – полное руководство по HTML и XHTML, рассказывающее, что они собой представляют и как ими следует пользоваться, включая все расширения, о которых нам известно. Некоторые расширения нигде не документированы, даже в многочисленных руководствах, встречающихся в Интернете. Если мы все-таки пропустили что-нибудь, дайте нам знать, и мы заполним пробел в следующем издании.

Использование фрагментов кода

Эта книга предназначена для оказания вам помощи в работе. Вообще говоря, вы можете использовать код, приведенный в этой книге, в своих программах и документах. Вы не обязаны получать наше разрешение

---

[1] CGI (Common Gateway Interface) – общий шлюзовой интерфейс. В данном случае подразумеваются программы и сценарии, использующие принципы CGI. – *Примеч. науч. ред.*

[2] С. Гулич, Ш. Гундаварам, Г. Бирзнекс «CGI программирование на Perl», издательство «Символ-Плюс», 2001.

за исключением тех случаев, когда воспроизводите значительные фрагменты кода. Например, для написания программы, в состав которой входят отрывки кода, приведенные в этой книге, разрешение не требуется. Продажа или распространение CD-ROM с примерами из книг O'Reilly *требует* разрешения. Когда вы отвечаете на чей-то вопрос, цитируя эту книгу и фрагменты кода из нее, вы не нуждаетесь в разрешении. Если вы включаете значительный объем кода из этой книги в документацию к своему продукту, вы *должны* получить разрешение.

Мы не требуем обязательных ссылок на книгу при цитировании, но будем благодарны, если вы их приведете. Ссылка обычно содержит название книги, автора, издательство и ISBN. Например, «HTML & XHTML: The Definitive Guide, Sixth Edition, by Chuck Musciano and Bill Kennedy. Copyright 2007 O'Reilly Media, Inc., 978-0-596-52732-7».

Safari® Enabled

Если на обложке технической книги есть пиктограмма «Safari® Enabled», это означает, что книга доступна в Сети через O'Reilly Network Safari Bookshelf.

Safari предлагает намного лучшее решение, чем электронные книги. Это виртуальная библиотека, позволяющая без труда находить тысячи лучших технических книг, вырезать и вставлять примеры кода, загружать главы и находить быстрые ответы, когда требуется наиболее верная и свежая информация. Она свободно доступна по адресу *http://safari.oreilly.com*.

Обратная связь

Вопросы и пожелания относительно этой книги направляйте по адресу:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international/local)
707-829-0104 (fax)

На сайте O'Reilly поддерживается веб-страница, содержащая списки подтвержденных и неподтвержденных ошибок, примеры и дополнительную информацию:

http://www.oreilly.com/catalog/html6

Комментарии и вопросы по сути материала направляйте электронной почтой по адресу:

bookquestions@oreilly.com

Дополнительные сведения о книгах, конференциях, информационных центрах и сети информационных сайтов O'Reilly Network можно найти на сайте:

http://www.oreilly.com

Благодарности

Мы бы не написали и не могли бы написать ни эту книгу, ни любое из предыдущих изданий без великодушной помощи многих людей. Наши жены, Джоанна и Синди, наши дети, Эва, Этан, Кортни и Коул (они появились *раньше*, чем мы начали писать эту книгу) обеспечили первую линию поддержки. Множество друзей, соседей и коллег помогали нам, делясь идеями, тестируя броузеры и предоставляя свое оборудование для исследования HTML. Вы знаете, что мы говорим о вас, и мы вас всех благодарим.

Также мы благодарим наших технических редакторов Чата Классмана (Chat Clussman), Патрика Крекелберга (Patrick Krekelberg), Сэма Маршалла (Sam Marshall) и Шломи Фиш (Shlomi Fish) за тщательное изучение нашей работы. Мы приняли большинство ваших остроумных замечаний. Мы благодарим наших редакторов Майка Лукидеса (Mike Loukides) и Татьяну Апанди (Tatiana Apandi) за терпение. И особая благодарность Татьяне за доведение этого, шестого, издания до логического завершения.

1

- Интернет
- Основные понятия Интернета
- HTML и XHTML – что они собой представляют
- HTML и XHTML – чем они не являются
- Стандарты и расширения
- Инструменты веб-дизайнера

HTML, XHTML и World Wide Web

Родившись в эксперименте военных и проведя детство в песочнице для ученых и чудаков, в настоящее время всемирная сеть компьютерных сетей (известная также как *Интернет*) превратилась в невероятно разнообразное и экономически самостоятельное сообщество пользователей и поставщиков электронной информации. И в зале заседаний, и в обычной квартире можно столкнуться с пользователями Интернета практически любой национальности и любых убеждений, от серьезных до праздных особ, от бизнес-корпораций до некоммерческих организаций и от возродившихся христиан-евангелистов до дельцов порноиндустрии.

Во многом именно Сеть – открытое всем пространство взаимодействующих гипертекстовых серверов и клиентов в Интернете – ответственно за стремительный рост популярности сети. Любой может стать полноправным членом этого сообщества, внеся свой вклад в виде созданных HTML- и XHTML-документов, представленных затем взору веб-серферов[1] всего мира.

Давайте взберемся на генеалогическое древо Интернета, чтобы основательнее рассмотреть его великолепие, не только удовлетворяя свое любопытство, но и стараясь лучше понять, с кем и с чем мы имеем дело, когда оказываемся в сети.

1.1. Интернет

Хотя распространенные представления часто бывают беспорядочными и запутывающими, понятие «Интернет» на деле, пожалуй, простое. Это всемирное сообщество компьютерных сетей – сеть сетей, – в кото-

[1] Веб-серфер – это тот, кто занимается веб-серфингом. Другими словами, это тот, кто «лазает» по веб-серверам. – *Примеч. науч. ред.*

ром обмен информацией происходит по общим правилам, зафиксированным в сетевых и программных протоколах.

Сети в компьютерном мире не новость. Что делает Интернет уникальным, так это всемирное собрание цифровых телекоммуникационных связей, применяющих общий набор компьютерных сетевых технологий, протоколов и приложений. Неважно, работаете ли вы в Microsoft Windows XP, Linux, Mac OS или даже в древней ОС Windows 3.1, – все компьютеры будут говорить на одном сетевом языке и применять функционально идентичные программы, так что можно обмениваться информацией – даже изображениями и звуками – с контрагентами, которые могут находиться как за соседней дверью, так и на краю света.

Обычные и в наше время хорошо известные программы, которые используются для общения и совместной работы через Интернет, нашли путь в закрытые и полузакрытые сети. Эти так называемые сети *интранет* и *экстранет* применяют то же программное обеспечение, приложения и сетевые протоколы, что и Интернет. Но, в отличие от него, сети типа интранет – это закрытые сети, открывающие доступ только членам организации. Подобным образом сети типа экстранет используют Интернет для оказания услуг зарегистрированным пользователям, ограничивая доступ посторонним.

В Интернете нет подобных ограничений. Всякий, у кого есть компьютер, необходимое программное обеспечение и соединение, может «войти в сеть» и начать обмениваться сообщениями, звуками и картинками с другими посетителями сети, где бы они ни находились, ночью или днем, при этом никакого членства не требуется. Именно это и делает Интернет таким запутанным.

Подобно восточному базару Интернет плохо организован, путеводителей по его содержимому мало, и может потребоваться много времени и технической опытности, чтобы выудить из него весь потенциал. И вот почему...

1.1.1. В начале

Интернет появился в конце шестидесятых как результат экспериментов по созданию жизнестойких компьютерных сетей. Целью было создать такую сеть, которая перенесла бы потерю нескольких машин, но сохранила бы за уцелевшими возможность общаться между собой. Финансирование поступало от Министерства обороны США, которое было заинтересовано в разработке информационных сетей, способных выстоять после ядерной атаки.

Получившаяся сеть стала удивительным достижением техники, но оставалась ограниченной в размерах и области применения. По большей части, лишь оборонные предприятия и академические институты могли получить доступ к тому, что тогда называлось ARPAnet (Advanced Research Projects Agency Network, или сеть агентства перспективных исследовательских разработок).

С появлением скоростных модемов, пригодных для цифровой связи по обычным телефонным линиям, некоторые люди и организации, не имевшие прямого доступа к главным каналам передачи информации, стали подключаться к сети и использовать выгоды передовой глобальной коммуникации. Тем не менее бастион Интернета был окончательно взят лишь примерно в 1993 году.

К стремительному взлету популярности Интернета привели несколько «судьбоносных» событий. Во-первых, в начале 90-х годов корпорации и частные лица, стремившиеся воспользоваться удобством и мощью глобальных цифровых коммуникаций, добились, наконец, того, что крупнейшие сети, связанные с Интернетом, который финансировало главным образом правительством США, открыли свои системы для практически свободного применения. (Зависимость маршрутизации потоков от содержания не предусматривалась при проектировании сети, из чего следовало, что коммерческие сообщения передавались через компьютеры университетов, для которых в то время коммерция была запрещена.)

Верные академическим традициям свободного обмена и совместного пользования, многие из создателей интернет-сообщества продолжали предоставлять значительное количество электронных документов и программ для бесплатного употребления. Глобальная связь, изобилие бесплатных программ и другой информации – кто мог устоять?

Честно говоря, целину Интернета в то время распахивать было нелегко. Установление соединения и применение разнообразного программного обеспечения, если даже «железо» позволяло, ставило непреодолимый технический барьер перед большинством людей. При этом наиболее доступной информацией в сети был «сухой паек» текстовых файлов, посвященных научным материям, а не конфетки в цветных бумажках, которые привлекают посетителей онлайновых сервисов, таких как America Online. Интернет был еще слишком неорганизованным и, если не считать правительства и ученых, мало кто обладал достаточным знанием и мотивом, чтобы изучать таинственное программное обеспечение, или избытком времени для перекапывания гор документов в поисках нужного материала.

1.1.2. HTML и Сеть

Для запуска ракеты Интернета понадобилась еще одна искра. Почти одновременно с открытием Интернета для бизнеса некие физики из CERN[1] (Европейская организация по ядерным исследованиям) объявили о создании нового языка и системы распространения, разработанных ими для формирования и передачи по сети электронных документов, включающих в себя мультимедиа-элементы. Так родился *язык*

[1] CERN – Conseil Europeen pour la Recherche Nucleaire (*фр.*). – *Примеч. науч. ред.*

разметки гипертекста (HyperText Markup Language, HTML), программа броузер и Всемирная паутина. Авторам теперь уже не нужно было распространять свои работы, разбитые на части и содержащие по отдельности звуки, изображения и текст. HTML объединил все эти элементы. Более того, веб-системы ввели в обращение *гипертекстовые ссылки*, благодаря которым документ автоматически связывался с другими документами, находящимися где угодно в сети. Это повлекло за собой меньшие затраты времени на поиски и возросшую эффективность пребывания в Интернете.

Подъем начался, когда группа преподавателей и способных студентов из Национального центра прикладного использования суперкомпьютеров (National Center for Supercomputing Applications, NCSA) университета Иллинойса в Урбана-Чемпейн написала броузер, названный Mosaic. Хотя он был первоначально разработан для просмотра HTML-документов, Mosaic имел встроенные средства доступа к значительно более богатым ресурсам Интернета, таким как FTP-архивы[1] программ и Gopher-собрания[2] документов.

С появлением версий, основанных на удобном в употреблении графическом интерфейсе, знакомом большинству владельцев компьютеров, Mosaic получил немедленный успех. Он, как и большинство программных продуктов для Интернета, распространялся в сети бесплатно. Миллионы пользователей получили свои копии Mosaic и ринулись на поиски «крутых страничек».

1.1.3. Золотая жила

За время своего существования Сеть породила совершенно новое средство для глобального обмена информацией и коммерческой деятельности. Так, например, с того момента, как специалисты по маркетингу догадались, что они могут дешево производить и распространять яркие, вызывающие восторг рекламу и каталоги продукции среди миллионов веб-серверов по всему миру, «мелькание башмаков из синей замши»[3] было не остановить. Даже основные разработчики Mosaic и соответствующих веб-серверов почувствовали запах денег. Они покинули NCSA и создали Netscape Communications с целью производить

---

[1] FTP (File Transfer Protocol) – протокол передачи файлов. Позволяет производить обмен файлами между сервером и клиентским компьютером. – *Примеч. науч. ред.*

[2] Gopher – интерактивный сервис для поиска и работы с документами в Интернете. В настоящее время практически не применяется. – *Примеч. науч. ред.*

[3] Имеется в виду победное шествие в середине 50-х годов хита рок-н-ролла под названием «Blue Suede Shoes» Карла Перкинса (Carl Perkins), не сходившего с верхних строчек хит-парадов более 5 месяцев. – *Примеч. науч. ред.*

коммерческие веб-броузеры и программное обеспечение серверов. Это продолжалось, пока не проснулся спящий великан, корпорация Microsoft. Впрочем, это совсем другая история...

Участие деловых людей и открывшиеся маркетинговые перспективы способствовали оживлению Интернета и обеспечили топливом его феноменальный взлет. Коммерция в Интернете превратилась в серьезный бизнес, оборот которого к 2005 году превысил 150 миллиардов долларов. Традиционные предприятия, обслуживающие клиентов в офисах, либо открыли коммерческие веб-сайты, либо оказались на грани исчезновения.

Есть мнение, особенно популярное среди старожилов Интернета, что бизнес и маркетинговая деятельность замусорили это пространство. Во многих отношениях Сеть превратилась в огромный супермаркет и раздражающий носитель рекламы. Хотите – верьте, хотите – нет, но в стародавние времена пользователи Интернета придерживались общепринятых, хотя и неписаных правил «сетикета»[1], который запрещает такие фокусы, как рассылку «спама»[2] по электронной почте.

Однако могущество HTML и распространение сведений по компьютерным сетям выходят далеко за рамки маркетинга и денежных выгод, поскольку выигрывает серьезная информационная деятельность. Публикации, включающие изображения и другие формы носителей, подобные исполняемым программам[3], достигают своей аудитории в мгновение ока, тогда как прежде требовались месяцы для печати и почтовой рассылки. Образование сделало огромный скачок, когда студенты получили доступ к величайшим библиотекам мира. И в часы досуга интерактивные возможности HTML могут освежить мозговые извилины, отупевшие от телевизионных передач.

1.2. Основные понятия Интернета

Каждый компьютер, подключенный к Интернету (даже старый разбитый Apple II), имеет уникальный адрес – число, формат которого задается *протоколом Интернета* (Internet Protocol, IP) – стандартом, определяющим, как сообщения перемещаются по сети от машины к ма-

---

[1] Сетикет – сокр. от сетевой этикет, аналогично netiquette – сокр. от network etiquette (*англ.*). – *Примеч. науч. ред.*

[2] Термин «спам» произошел от названия скетча английской комик-группы Monty Python Flying Circus, в котором посетителям ресторана приходилось слушать хор викингов, воспевающий мясные консервы (SPAM – «spiced ham»). Причем в меню не было ни одного блюда без содержимого этих консервов (по материалам сайта *www.antispam.ru*). – *Примеч. науч. ред.*

[3] Но с такой же легкостью началось и массовое распространение «программ», представляющих из себя компьютерные вирусы: «троянских коней», «сетевых червей» и т. п. – *Примеч. науч. ред.*

шине. *IP-адрес* состоит из четырех[1] разделенных точками чисел, не превосходящих 256 каждое, например 192.12.248.73 или 131.58.97.254.

В то время как компьютеры оперируют только числами, люди предпочитают слова. Вследствие этого у всякого компьютера в Интернете есть также имя, присвоенное ему владельцем. В сети находятся сотни миллионов, если не миллиарды, машин, поэтому было бы очень трудно управиться с таким количеством уникальных имен, не говоря уже о том, чтобы помнить путь к каждому из них. Заметим, однако, что Интернет – это сеть сетей. Он разделен на группы, именуемые *доменами*, которые, в свою очередь, подразделяются на один или несколько *поддоменов*.[2] Так что, хотя вы можете выбрать совсем неоригинальное имя для своего компьютера, оно превратится в уникальное при добавлении к нему, подобно фамилии, всех доменных имен в виде разделенных точками суффиксов, что дает *полностью квалифицированное* доменное имя.[3]

Конструкция имен проще, чем кажется. Например, полностью квалифицированное доменное имя *www.oreilly.com* читается как имя машины «www», которая принадлежит домену с названием «oreilly», а он, в свою очередь, является частью коммерческой (com) ветви Интернета. Среди других ветвей Интернета упомянем институты образования (edu), некоммерческие организации (org), правительство США (gov) и интернет-провайдеров (net). Компьютеры и сети за пределами США могут иметь двухбуквенные сокращения в конце их имен: (ca) для Канады, (jp) для Японии и (uk) для Великобритании.[4]

Специальные компьютеры, известные как *серверы имен* (точнее, серверы системы доменных имен, Domain name system server, DNS server), хранят таблицы имен машин с ассоциированными с ними IP-адресами и переводят их друг в друга для человеческих и компьютерных

---

[1] Это так называемый протокол IPv4 – с 4-х байтными сетевыми адресами; в связи с практически полным исчерпанием свободных Ipv4 адресов в последние годы все шире применяется протокол Ipv6 с 6-ти байтными адресами. – *Примеч. науч. ред.*

[2] Обычно говорят об уровне домена (или зоне): самый глобальный домен имеет первый уровень, вложенный в него – второй и т. д. Применительно к именам доменов можно сказать, что адрес *www.books.ru*, принадлежащий домену третьего уровня с именем «www», включен в домен второго уровня «books», которые являются, в свою очередь, поддоменами домена первого уровня «ru». – *Примеч. науч. ред.*

[3] Отметим (это не с очевидностью всеми воспринимается), что полностью определенное доменное имя – это не что иное, как синоним IP-адреса, более легкий для человеческого восприятия: любое обращение по имени может быть с таким же успехом осуществлено и непосредственно по IP-адресу, при этом оно не будет ничем отличаться. – *Примеч. науч. ред.*

[4] Для России это (ru) или (su), для Украины – (ua), Беларуси – (by) и т. д. Полный список можно найти, например, на *http://www.webclub.ru/materials/domaintop/index.html*. – *Примеч. науч. ред.*

нужд. Доменные имена нужно регистрировать и платить за них одной из многочисленных коммерческих организаций-регистраторов.[1] После регистрации владелец доменного имени сообщает его и соответствующий адрес другим серверам доменных имен по всему миру.

1.2.1. Клиенты, серверы и броузеры

Интернет связывает компьютеры двух видов: *серверы*, хранящие документы и обслуживающие внешние запросы, и *клиенты*, которые получают документы и отображают их для людей. О том, что происходит на сервере, говорят как о находящемся «*со стороны сервера*», тогда как о действиях на машине клиента говорят как о производимых «*со стороны клиента*».

Чтобы принять и отобразить HTML-документы, мы запускаем программы, называемые *броузерами*, на своем компьютере. Броузер в качестве клиента обращается к специальным *веб-серверам* в Интернете для получения доступа к документам.

Сегодня доступны самые разные броузеры. Например, Internet Explorer поставляется с операционной системой корпорации Microsoft, а большинство других броузеров можно бесплатно загрузить из Интернета. Большинство типов броузеров работает на клиентских компьютерах, имеющих графические дисплеи с огромным количеством цветов и высоким разрешением. У современных броузеров, с позволения сказать, под капотом, находятся практически одинаковые механизмы визуализации HTML, и различаются эти броузеры только внешними, хотя и довольно полезными характеристиками. Например, когда вы устанавливаете Netscape Navigator версии 8, вы решаете, будете ли вы пользоваться механизмом визуализации NCSA Mosaic, отдельные детали которого присутствуют в Internet Explorer от Microsoft, или предпочтете механизм от Mozilla, который можно найти в другом популярном броузере, Firefox.

Эта ситуация сильно отличается от той, что была в начале XXI века, когда Internet Explorer яростно конкурировал с Netscape Navigator, используя в качестве оружия уникальные расширения языка HTML. Internet Explorer победил.[2] Многие его расширения даже стали стандартными для HTML, а другие, например расширенные средства макетирования Netscape, исчезли, и в этой книге упоминание о них можно найти только в приложениях.

---

[1] В России регистрацией и распределением доменных имен и IP-адресов ведает РосНИИРОС (RIPN) – Российский НИИ развития общественных сетей (*www.ripn.net*). – *Примеч. науч. ред.*

[2] Это спорное утверждение (или скоропалительное на время написания книги): после некоторого «перевеса» IE наступил период массового перехода пользователей на новые версии броузеров от Mozilla: FireFox, SeaMonkey, и эта миграция продолжается, так как все версии IE «страдают» очень большой несовместимостью с общепризнанными стандартами (например с DOM – Document Object Model). – *Примеч. науч. ред.*

1.2.2. Информационные потоки

Деятельность в сети начинается на стороне клиента, когда пользователь запускает свой броузер. Тот приступает к работе с загрузки *домашней страницы*, хранящейся либо на локальном компьютере, либо в какой-нибудь сети: в Интернете, в корпоративном интранете или в муниципальном экстранете. Выходя в сеть, броузер клиента сначала обращается к серверу системы доменных имен (DNS), чтобы преобразовать имя сервера, на котором находится домашняя страница, например *www.oreilly.com*, в IP-адрес, перед тем как послать запрос к другому серверу через Интернет. Этот запрос (так же как и возвращаемый ответ) имеет формат, предписанный стандартом *протокола передачи гипертекста* (Hypertext Transfer Protocol, HTTP).

Большую часть времени сервер проводит, слушая сеть, ожидая адресованные ему (т. е. содержащие в теле его уникальный адрес) запросы на документы. Получив такой запрос, сервер выясняет, имеет ли право обращающаяся сторона на получение документа, и, если это так, проверяет наличие запрашиваемого объекта. Если документ найден, сервер посылает его броузеру. Сервер обычно регистрирует запрос, включающий в себя IP-адрес компьютера клиента, запрошенный документ и момент времени[1], в который произошла данная операция. Кроме того, сервер может приложить к документу специальные информационные элементы, называемые *cookies*, которые содержат дополнительную информацию о броузере и его владельце.

Далее документ прибывает на броузер. Если это рядовой текстовый файл, то чаще всего он отображается самым обычным способом. Каталоги также обрабатываются как текстовые файлы, хотя большинство графических броузеров воспроизводят пиктограммы папок, которые пользователь может выбирать, чтобы просмотреть содержимое подкаталога.

С сервера можно получать файлы самых разных типов. Если броузер не пользуется *вспомогательными программами* (helper program, иногда называемая wizard – «мастер») и не укомплектован *плагинами* (plug-in sofware, встраиваемый модуль) или *апплетами*[2] (applets, минипрограммы), отображающими картинки или видеофайлы либо проигрывающими аудиофайлы, он обычно сохраняет файл на локальном диске для использования в будущем.

Однако чаще всего броузер получает особые документы, выглядящие как простые текстовые файлы, но содержащие кроме текста специальные обозначения разметки, называемые тегами. Броузер обрабатывает

---

[1] Поскольку Интернет – глобальная сеть мирового масштаба, чтобы избежать путаницы, серверы, как правило, регистрируют время относительно Всемирного (Гринвичского среднего) времени – GMT. – *Примеч. науч. ред.*

[2] Обычно так называют Java-приложения, встраиваемые в HTML-страницы. – *Примеч. науч. ред.*

эти HTML- или XHTML-документы, форматирует текст в соответствии с тегами и, если необходимо, загружает дополнительные файлы, например картинки.

Пользователь читает документ, выбирает гиперссылку на другой источник, и процесс начинается сначала.

1.2.3. В оффлайне

Следует еще раз отметить, что броузеры и HTTP-серверы необязательно подключать к Интернету, чтобы они работали. В действительности не требуется никакого подсоединения к Интернету или к какой-либо сети, чтобы писать HTML/XHTML документы и использовать броузер. Он позволяет загружать и отображать хранящиеся на локальных машинах документы и их аксессуары. Многие фирмы пользуются этим, распространяя каталоги и инструкции к товарам, например на CD-ROM, гораздо более дешевых и интересных в смысле интерактивности, чем бумажные документы. Различные приложения с графическим интерфейсом описывают свои функциональные возможности в виде электронных справочников в формате HTML/XHTML.

Такая изоляция полезна: она дает возможность завершить, в смысле редактирования, комплект материалов, предназначенных для последующего распространения. Добросовестные авторы работают локально, когда пишут и тестируют свои документы, прежде чем открыть их для общего употребления, избавляя таким образом читателей от мучений с поврежденными картинками и ссылками, ведущими в никуда.[1]

Организации также могут быть соединены с Интернетом и при этом поддерживать свои частные веб-сайты и подборки документов, предназначенные для распространения среди клиентов, входящих в их интранет. И в самом деле частные веб-сайты становятся быстро развивающейся технологией безбумажного документооборота, о которой так много говорили в последние годы. С применением HTML и XHTML электронные архивы предприятий могут поддерживать базы данных персонала, укомплектованные фотографиями служащих, подборку технической документации, включая копии чертежей, различные справочники и т. д., – все это в готовом к употреблению и легко доступном виде.

1.2.4. Стандартизация HTML

Как многие другие популярные технологии, HTML начался с неформального описания, которым пользовалось лишь небольшое число людей. Когда же количество применяющих этот язык возросло, стало яс-

[1] Мы настоятельно рекомендуем проводить тщательное тестирование HTML-документов и после того, как они «выложены» в сеть, что необходимо для исправления различных ошибок, связанных с неправильным указанием ссылок.

но, что требуется более строгий подход к определению и сопровождению языка с тем, чтобы сделать создание и совместное использование написанных на нем документов более легким и удобным.

1.2.4.1. World Wide Web Consortium

World Wide Web Consortium (W3C) был сформирован с уставной целью определить стандарты для HTML и позднее для XHTML. Его члены ответственны за создание чернового варианта стандарта, организацию его обсуждения и за внесение изменений в первоначальный вариант, учитывая замечания и предложения, поступившие от интернет-сообщества, чтобы удовлетворить пожелания большинства.

Помимо HTML и XHTML в сферу ответственности W3C входит стандартизация любых технологий, относящихся к Сети. Он занимается HTTP, каскадными таблицами стилей и расширяемым языком разметки (Extensible Markup Language, XML) так же, как и связанными с ними стандартами адресации в сети. Кроме того, W3C рассматривает проекты стандартов для расширений существующих веб-технологий.

Если вы хотите следить за развитием HTML, XML, XHTML, CSS и других впечатляющих разработок в области веб-технологий, обращайтесь по адресу *http:// www.w3c.org*.

Обсуждению веб-технологий посвящены также несколько групп новостей в иерархии *comp.infosystems.www*. Среди них *comp.infosystems.www.authoring.html* и *comp.infosystems.www.authoring.images*.

1.2.4.2. Инженерная рабочая группа Интернета

Еще более широкие задачи у Инженерной рабочей группы Интернета (Internet Engineerig Task Force, IETF), ответственной за управление всеми аспектами интернет-технологий. «Всемирная паутина» – это лишь небольшая часть того, что находится под надзором IETF.

IETF определяет все, что касается технологии Интернета, в официальных документах, именуемых Requests for Comment (RFC, запрос на комментарии). Обладающий индивидуальным номером для удобства ссылок, каждый из RFC посвящен отдельной технологии Интернета от синтаксиса доменных имен и распределения IP-адресов до формата писем электронной почты.

Узнать больше об IETF и проследить за прогрессом различных RFC[1], обсуждаемых и пересматриваемых, можно, посетив домашнюю страницу IETF *http://www.ietf.org*.

---

[1] В опубликованный RFC не вносятся изменения. При необходимости подготавливается новый документ с очередным порядковым номером, при этом в предшествующий RFC вставляется ссылка на новую редакцию. – *Примеч. науч. ред.*

1.3. HTML и XHTML – что они собой представляют

HTML и XHTML определяют синтаксис и правила употребления специальных встроенных в язык инструкций – тегов, которые не воспроизводятся броузером, но указывают ему, как надо отобразить содержимое документа: изображения, текст и другие вспомогательные виды информации. Кроме того, эти языки позволяют сделать документ интерактивным с помощью гипертекстовых ссылок, которые связывают его с документами на вашем или любом другом компьютере, а также с другими ресурсами Интернета.

Вы, определенно, уже слышали о HTML и XHTML, но известно ли вам, что это только два из множества языков разметки? По правде, HTML среди них – белая ворона. HTML основан на SGML, стандартном обобщенном языке разметки (Standard Generalized Markup Language). Сильные мира сего создали SGML, рассчитывая, что он станет единственным метаязыком разметки, порождающим все ее элементы. Все – от иероглифов до HTML – может быть определено при помощи SGML, что и отменяет потребность в каких-либо других средствах.

Проблема SGML состоит в том, что он так широк и всеобъемлющ, что простые смертные не способны им пользоваться. Эффективная работа с ним требует чрезвычайно дорогого и сложного инструментария, недоступного для большинства обычных людей, желающих быстро «состряпать» веб-документ. В результате HTML придерживается некоторых, но не всех стандартов SGML[1], игнорируя множество особенно эзотерических его черт, так что HTML легко и удобно пользоваться.

Осознав, что SGML чересчур громоздок и не слишком подходит для описания популярного HTML, и что обработка различных сетевых документов требует создания новых HTML-подобных языков, W3C определил расширяемый язык разметки (XML). Подобно SGML, XML является самостоятельным метаязыком, использующим избранные элементы SGML для описания новых языков разметки. В нем отсутствуют многие возможности SGML, не применимые к таким языкам, как HTML, а другие элементы SGML упрощены, чтобы их было легче понимать и употреблять.

Однако HTML версии 4.01 не соответствует стандарту XML. Поэтому W3C предложил XHTML, переработав HTML в соответствии с требованиями XML. Язык XHTML пытается поддерживать все возможности HTML 4.01 вплоть до самых незначительных, применяя более жесткие правила XML. В основном он справляется с этой задачей, хотя и обладает достаточным количеством отличий, чтобы затруднить жизнь добросовестному приверженцу прежних стандартов.

---

[1] Определение типа документа (Document Type Definition, DTD) для HTML, приведенное в приложении D «HTML 4.01 DTD», использует подмножество SGML для определения типов в стандарте HTML 4.01.

1.4. HTML и XHTML – чем они не являются

При всем богатстве свойств (использование мультимедиа, применение новейших технологий и структурирование материала на странице), определивших роль HTML/XHTML-документов в Интернете, эти языки остаются ограниченными в своих возможностях, о чем необходимо помнить. Они не являются средствами обработки текста, настольными издательскими системами или даже языками программирования. Их основная задача – определить структуру документов и семейств документов так, чтобы они могли быть легко и быстро доставлены пользователю по сетям и отображены на самых разных дисплеях. Можно сказать, эти языки умеют все, но только в своей области применения.

1.4.1. Форма или содержание?

HTML и его потомок XHTML предоставляют много различных средств управления внешним видом документа, но созданы они для того, чтобы структурировать документы, а вовсе не для форматирования их перед отображением. Разумеется, внешний вид важен, поскольку он может как ухудшать, так и улучшать восприятие информации и ее использование. И поэтому так важен сопровождающий стандарт CSS.

Однако в HTML и XHTML содержание имеет первостепенную важность. Внешний вид имеет второстепенное значение, особенно потому, что он менее предсказуем в силу различия возможностей броузеров в изображении графики и форматировании текста. Фактически HTML и XHTML содержат множество средств структурирования документа, не подразумевающих конкретного способа отображения. Заголовки разделов, структурированные списки, абзацы, линейки, заголовки и встроенные картинки – все они определяются стандартами языка без оглядки на то, как они могут быть представлены броузером. Рассмотрим, к примеру, броузер для слепых, в котором графика на страницах сопровождается аудиоописаниями и для навигации используются альтернативные средства. В стандартах HTML/XHTML все определяется фразой: содержание важнее внешнего вида.

Если вы считаете HTML и XHTML инструментами для оформления документа, то будете горько разочарованы. В этих языках просто отсутствуют средства для создания чего-либо подобного тому, что вы могли бы сделать с применением FrameMaker или Microsoft Word. Попытки использовать элементы структурирования для достижения специальных эффектов форматирования потерпят неудачу, так как желаемый эффект вряд ли будет проявляться во всех броузерах. Короче говоря, не тратьте времени, пытаясь заставить HTML и XHTML делать то, для чего они не предназначены.

Вместо этого используйте языки сообразно их назначению – укажите структуру документа так, чтобы броузер обработал его подходящим образом. HTML и XHTML необычайно богаты тегами, позволяющими

подчеркнуть смысл содержимого в материале, – это то, чего не хватает текстовым редакторам и программам макетирования страниц. Создавайте документы, используя теги, и вы станете счастливее, документы будут смотреться лучше, а читатели выиграют неизмеримо.

1.5. Стандарты и расширения

Базовый синтаксис и семантика HTML определены в стандарте HTML, последней версией которого на сегодня является 4.01. Язык HTML быстро развивался в течение десятилетия. Было время, когда новая версия появлялась раньше, чем читатели успевали изучить предыдущее издание этой книги. Сейчас эволюционирование HTML прекратилось. С точки зрения группы W3C полномочия перешли к языку XHTML. Остается подождать, пока производители броузеров реализуют стандарты.

Текущей версией XHTML является 1.0. По счастью, версия 1.0 языка XHTML в основном совпадает с версией HTML 4.01. Имеются кое-какие отличия, на которых мы остановимся в главе 16. Популярные броузеры продолжают поддерживать HTML-документы, так что нет причин в панике бежать к XHTML. Тем не менее, имеет смысл начать неспешное движение в этом направлении. Новая версия XHTML, 2.0, уже рассматривается группой W3C, а производители броузеров медленно, но верно исключают нестандартные функциональные особенности HTML из своих продуктов.

Очевидно, разработчики броузеров придерживаются стандартов и общепринятых соглашений, когда реализуют в своих продуктах форматирование и вывод HTML- и XHTML-документов. Что касается авторов, они ориентируются на стандарты, чтобы создавать эффективные и корректные документы, правильно представляемые броузерами.

Однако в стандартах не все прописано явно, и у производителей есть некоторая свобода действий в отношении того, как их броузеры будут выводить тот или иной элемент. Ситуацию усложняет тот факт, что коммерческие соображения заставляют разработчиков добавлять в броузеры нестандартные расширения под предлогом улучшения языка.

Боитесь запутаться? Не волнуйтесь: в этой книге мы подробно исследуем синтаксис, семантику и идиомы языков HTML 4.01 и XHTML 1.0, а также многие важные расширения, поддерживаемые последними версиями наиболее популярных броузеров.

1.5.1. Нестандартные расширения

Не нужно быть очень наблюдательным, чтобы заметить, что многие люди подчеркивают свои особенности, стремясь привлечь внимание окружающих. То же происходит и с броузерами. Яркие и эффектные возможности нового продукта дают его производителю преимущество на рынке, стандартизованном во всех других отношениях. Для авто-

ров страниц это может обернуться кошмаром. Множество людей захотят, чтобы вы использовали новейший и самый потрясающий трюк или какое-то, возможно, даже полезное, расширение HTML/XHTML. Но оно не входит в стандарт, и не все броузеры его поддерживают. По счастью, иногда популярные броузеры позволяют достигать одного и того же результата различными средствами.

1.5.2. Расширения – за и против

Всякий продавец программного обеспечения строго придерживается технологических стандартов. Неловко иметь несовместимые программные продукты: конкуренты при всякой возможности напомнят покупателям о том, что вам «не удалось» соблюсти правила, как бы вычурен или бесполезен ни был тот элемент стандарта, который вы обошли. В то же время продавцы стремятся к тому, чтобы их продукт отличался от продукции конкурентов в лучшую сторону. Расширения стандарта HTML, осуществленные Netscape и Microsoft, превосходно демонстрируют, к чему ведет давление рынка.

Многие авторы страниц чувствуют себя в безопасности, используя расширения, поддерживаемые броузерами этих производителей, поскольку вместе они повелевают рынком. Плохо это или хорошо, но расширения HTML из наиболее популярных броузеров становятся частью «уличной версии» языка, точно так же, как английский сленг прокрадывается в лексикон большинства французов, несмотря на все старания их Академии.

К счастью, на HTML версии 4.0 производители броузеров и W3C сошлись. Действительно, с появлением таблиц все изменилось.[1] Многие расширения HTML, впервые возникшие в Netscape Navigator и Microsoft Internet Explorer, сейчас являются частью стандартов HTML 4 и XHTML 1, хотя и не все другие элементы стандарта поддерживаются популярными броузерами.

1.5.3. Избегать расширений

В целом, мы настоятельно советуем не применять расширения без убедительных и веских причин. Используя расширения, особенно в ключевых местах документа, вы рискуете потерять существенную часть аудитории. Справедливости ради отметим, что большинство броузеров остерегается расширений.

Безусловно, коварно запрещать использование расширений и вместе с тем давать полное описание их употребления. В согласии с общей философией Интернета мы предпочитаем вручить вам веревку и пистолет, полагая, что вам хватит ума, чтобы не повеситься и не прострелить себе ногу.

---

[1] В HTML 4.0 были добавлены новые возможности форматирования таблиц, а также таблицы стилей. – *Примеч. науч. ред.*

И все же совет остается: применяйте расширения только там, где это необходимо или дает огромные преимущества. Делая это, помните, что таким образом вы отталкиваете от себя часть аудитории. В подобном случае вы можете даже поразмышлять, не следует ли представить отдельную, основанную на стандарте версию вашего документа, которая устраивала бы пользователей других броузеров.

1.5.4. Расширения через модули

Версия 1.1 языка XHTML предусматривает стандартный механизм его расширения: модули XML. Язык XHTML фактически сам состоит из модулей.

Модули XHTML разбивают язык HTML на дискретные типы документов, каждый из которых определяет характеристики и функции, являющиеся частью языка. Существуют отдельные модули для XHTML-форм, текста, сценариев, таблиц и т. д., то есть всех элементов XHTML 1.0, не вызывающих разногласий.

Достоинством модулей является расширяемость. В дополнение к средствам разметки, реализованным в XHTML-модулях, включенных в стандарт, новый язык позволяет вам «подмешивать» другие XML-модули в ваши документы, расширяя их функциональные возможности стандартным образом. Например, группа W3C определила модуль MathML, предоставляющий элементы разметки для математических уравнений, которые вы сможете использовать для XHTML-форматирования вашей следующей статьи по высшей математике.

За пределами языка XHTML 1.1 модули являются экспериментальным решением и не вполне поддерживаются популярными броузерами. Поэтому мы не рекомендуем вам применять XHTML-модули прямо сейчас. На данный момент эта тема лежит за пределами нашей книги, и, если вас интересуют подробности, посетите сайт W3C.

1.6. Инструменты веб-дизайнера

Хотя вы можете использовать самый что ни есть простой текстовый редактор для создания HTML- и XHTML-документов, у большинства авторов существует чуть более совершенный инструментарий, чем просто обработчик слов. Чтобы тестировать и совершенствовать свои работы, нужно, как минимум, иметь под рукой броузер. Помимо упомянутых основных средств, есть и специализированные программы для подготовки и редактирования HTML-документов, а также программы для проектирования и подготовки мультимедийных аксессуаров.

1.6.1. Основные орудия

По меньшей мере вам необходим текстовый редактор, броузер для проверки результатов работы и, в идеале, подключение к Интернету.

1.6.1.1. Текстовый процессор или WYSIWYG–редактор?

Некоторые авторы используют возможности своих специализированных программ-редакторов HTML/XHTML. Другие применяют инструменты WYSIWYG[1], которые поставляются вместе с броузерами или последними версиями популярных текстовых процессоров. Третьи, как авторы книги, предпочитают выполнять свои работы в обычных текстовых редакторах, а затем расставлять в них теги разметки и атрибуты. Есть и те, кто пишет теги сразу.

Мы считаем, что ступенчатый подход – написать, затем разметить – это лучший способ. Более того, определив и написав содержимое документа, гораздо легче сделать еще один проход, чтобы основательно и результативно расставить HTML/XHTML-теги, размечая текст. Если действовать по-другому, разметка может отвлечь от содержания. Отметим также, что без специальной «дрессировки» (если они ей вообще поддаются) словари и программы, проверяющие правописание, будут, как правило, спотыкаться на тегах разметки и их разнообразных параметрах. Вам придется полжизни провести, нажимая кнопку «Пропустить» на совершенно правильно написанных тегах при проверке правописания.

От того, когда и как вы вставляете в свой документ теги разметки, зависит, какие средства вам необходимы. На наш взгляд, стоит использовать хороший текстовый редактор, который подходит для этих целей лучше, чем простейшие текстовые или встроенные в броузеры специализированные редакторы разметки. В частности, легко заметить, что планировщик[2], словарь и проверка орфографии помогают лучше всего, когда вы пишете, не отвлекаясь на внешний вид, который проявится впоследствии. Поздние версии упомянутых редакторов могут перекодировать документы в HTML, но не ждите чудес. За исключением случаев разработки шаблонов вам, вероятно, придется достаточно потрудиться, доводя до ума эти документы. (А также «ужимать» их, когда вы увидите, как велик сгенерированный HTML-код). И пройдет еще немало времени, прежде чем в популярные редакторы будут включены средства разработки XHTML-документов.

Еще одно предостережение относительно средств автоматического построения документов: обычно они изменяют содержимое или вносят в него дополнения (например, заменяя, относительные ссылки абсолютными) и компонуют ваш документ способом, вызывающим раздражение. В особенности неприятно то, что они редко предоставляют возможность сделать так, как хотите вы, а не разработчики данного продукта.

---

[1] WYSIWYG (what-you-see-is-what-you-get – что видишь на экране, то и получишь в итоге) – средства для интерактивного визуального редактирования в режиме «полного соответствия». – *Примеч. науч. ред.*

[2] Планировщик – это средство для формирования оглавления. – *Примеч. науч. ред.*

Так что не надейтесь на автоматический компоновщик – за ним, как правило, приходится многое переделывать. Овладевайте старыми добрыми HTML/XHTML, тем самым вы обеспечите себе возможность в любой момент доделать свой документ по собственному вкусу.

1.6.1.2. Броузеры

Вы, очевидно, должны увидеть и протестировать вновь созданный документ, прежде чем опубликовать его. Серьезным авторам, особенно тем, кто намерен выходить за пределы HTML/XHTML-стандартов, мы рекомендуем иметь несколько броузеров, возможно, с их разными версиями, работающими на различных компьютерах, чтобы быть уверенными, что великолепная презентация на одних не превращается в кошмар на других.

Самые популярные в настоящее время броузеры и, следовательно, самые необходимые – это Microsoft Internet Explorer, Mozilla Firefox, Safari (от Apple), Opera и Netscape Navigator, хотя последний быстро сходит со сцены. Большинство версий работает на распространенных компьютерных платформах, таких как разнообразные ОС от корпорации Microsoft, а также Linux, Mac OS, и т. д. Версии броузеров часто расходятся в том, какие элементы HTML и XHTML они поддерживают. В нашей книге мы всячески стараемся подчеркнуть эти различия. Тем не менее, мы советуем вам загружать с сайтов производителей не только последние версии броузеров, но и предыдущие, чтобы тщательнее протестировать свою работу на совместимость. Это особенно важно, если учесть, что несколько миллионов из более чем миллиарда пользователей Интернета по-прежнему работают на допотопной версии Internet Explorer 5!

1.6.2. Расширенный инструментарий

Если вы серьезно займетесь разработкой документов, то скоро обнаружите, что существуют всевозможные приспособления, которые могут облегчить вашу жизнь. Список продуктов, распространяемых бесплатно, условно-бесплатно (с ограничением срока или возможностей бесплатного использования) и на коммерческой основе, растет изо дня в день, так что здесь его приводить бесполезно. Это действительно еще один повод посещать различные группы новостей и сайты, которые поддерживают в сети обновляемые списки HTML- и XHTML-ресурсов. Если вы всерьез решили заняться написанием документов на HTML и XHTML, то непременно посетите эти сайты. Вы станете обращаться к ним регулярно, чтобы быть в курсе новостей, касающихся языка, инструментария и тенденций их развития.

- На чем писать
- Первый HTML-документ
- Встраиваемые теги
- Костяк HTML
- Плоть HTML- и XHTML-документов
- Текст
- Гиперссылки
- Изображения – это особая статья
- Списки, строка поиска, формы
- Таблицы
- Фреймы
- Таблицы стилей и JavaScript
- Вперед!

2

Быстрый старт

Мы не изучали часами какие-либо справочники, перед тем как написать наш первый HTML-документ. Вы, вероятно, тоже не станете. HTML легко читать и понимать, да и писать на нем не трудно. А ваш первый опус, созданный в HTML, будет почти готовым XHTML-документом. Так что приступим, не тратя времени на изучение множества трудных правил.

Чтобы помочь вам быстро и с удовольствием сделать первый шаг, мы выполнили эту главу в качестве краткой сводки по элементам HTML и его потомка, XHTML. Разумеется, было опущено много деталей и ряд уловок, которые следует знать. Читайте следующие главы для полного овладения HTML и XHTML.

Даже если эти языки вам не в новинку, мы советуем проработать эту главу, прежде чем переходить к оставшейся части книги. Она позволит не только усвоить основы HTML/XHTML и их жаргон, но и сделает дальнейшую работу эффективной, ободрив вас успехами в создании таких привлекательных документов за столь короткое время.

2.1. На чем писать

Для создания HTML- или XHTML-документов используйте любой редактор, способный сохранить на диске текстовый файл. Даже если документы имеют сложную планировку и включают в себя изображения, они тем не менее остаются простыми текстовыми файлами. Продвинутый WYSIWYG-редактор или конвертер, входящий в комплект вашего любимого текстового процессора, тоже годится, хотя они могут не поддерживать все возможности, которые будут обсуждаться в этой книге. Вам все равно придется внести последние штрихи в выданный ими

текст; к тому же не стоит надеяться, что компоновка страницы будет такой же, как при использовании специализированного приложения.

Вам также понадобится установить на компьютере популярный броузер, который хотя и не нужен для создания документов, но необходим, чтобы видеть результаты работы. Причина проста: исходный текст, составленный с помощью текстового редактора, выглядит совсем непохоже на то, что покажет броузер, хотя это один и тот же документ. Убедитесь, что ваши читатели действительно увидят то, что вы хотели им показать, просмотрев документ до них с помощью броузера. Кстати, популярные броузеры можно скачать бесплатно. На сегодняшний день мы можем рекомендовать Microsoft Internet Explorer, Mozilla Firefox, Apple Safari, Netscape Navigator и Opera.

Заметьте также, что нет необходимости подключаться к Интернету, чтобы писать или просматривать HTML- или XHTML-документы, – их можно сохранять на жестком диске или на дискете. Можно даже путешествовать по локальным документам, применяя гиперссылки, при этом не имея никакой связи с Интернетом или какой-либо другой сетью. На наш взгляд, следует работать локально, проектируя и тщательно тестируя документы, прежде чем делиться ими с окружающими.

Тем не менее мы настоятельно рекомендуем *обязательно иметь* выход в Интернет, если вы всерьез настроены создавать собственные документы. Можно загружать и просматривать чьи-то интересные страницы и выяснять, как на них применяются интересные возможности языка – хорошо или плохо. К тому же учиться на примерах нескучно. (С другой стороны, использование чужих находок часто проблематично, если даже не противозаконно.) Подключение к Интернету существенно, если в своей работе вы обращаетесь к ссылкам на другие документы в глобальной сети.

2.2. Первый HTML–документ

Похоже, что всякий когда-либо написанный учебник по языкам программирования начинается с простого примера: как отобразить сообщение «Hello, World!». Что же, в этой книге никакого «Hello, World!» вы не увидите. В конце концов, перед вами руководство по стилю, предназначенное для нового тысячелетия. Вместо этого мы пошлем приветствие World Wide Web:

```
<html>
<head>
<title>My first HTML document</title>
</head>
<body>
<h2>My first HTML document</h2>
Hello, <i>World Wide Web!</i>
 <!-- Обойдемся без "Hello, World" -->
<p>
```

```
                Greetings from<br>
<a href="http://www.ora.com">O'Reilly</a>
<p>
Composed with care by:
<cite>(insert your name here)</cite>
<br>&copy;2000 and beyond
</body>
</html>
```

Пойдем дальше. Наберите исходный текст примера на новой странице редактора и сохраните его на локальном диске как *myfirst.html*. Убедитесь, что он записан в простом текстовом формате. Другие форматы, такие как *.doc*, присущий Microsoft Word, добавляют к тексту специальные (непечатаемые) символы, которые могут запутать броузер и нарушить вывод HTML-документа.

Сохранив на диске файл *myfirst.html* (или *myfirst.htm*, если вы пользуетесь архаичным стандартом[1] образования имен файлов, поддерживаемым DOS или Windows 3.11), запустите броузер, обратитесь к его меню и с помощью команды File (Файл) найдите и откройте документ. Ваш экран должен выглядеть примерно так (рис. 2.1). Хотя некоторые элементы интерфейса, например меню и панели инструментов, могут выглядеть по-разному в разных броузерах, содержимое окна будет таким же.

***Рис. 2.1.** Очень простой HTML-документ*

[1] Здесь подразумевается устаревшее так называемое «правило 8.3», по которому обозначение файла должно состоять из двух частей: имени, содержащем не более восьми символов, и расширения, включающего точку и следующие за ней от 1 до 3 символов. – *Примеч. науч. ред.*

2.3. Встраиваемые теги

Вы, наверное, уже заметили, и возможно с удивлением, что броузер отображает меньше половины исходного текста примера. Ближайшее рассмотрение источника показывает: пропало все, что заключено в скобки из символов «меньше» (<) и «больше» (>). [синтаксис тегов, 3.3.1]

HTML и XHTML – это встраиваемые языки: их директивы, или *теги*, вставляются в тот самый документ, который авторы и читатели загружают в броузер для просмотра. Броузер использует информацию, находящуюся внутри этих тегов, чтобы решить, как отобразить (или иначе обработать) следующее за тегом содержимое документа.

В частности, тег , стоящий в примере за словом «Hello», предписывает броузеру отобразить следующий за ним текст курсивом.[1] [теги физической разметки, 4.5]

Первое слово в теге – это его официальное имя, которое обычно описывает присущие ему функции. Любые добавочные слова в теге являются специальными *атрибутами*, иногда с присвоенным им значением, стоящим после знака равенства (=), которые определяют или модифицируют действие тега.

2.3.1. Открывающий и закрывающий теги

Большая часть тегов определяет и оказывает влияние на ограниченную область документа. Эта область начинается там, где тег со своими атрибутами появляется впервые (его называют иногда *открывающим тегом*), и продолжается до соответствующего *закрывающего тега*, в котором его имени предшествует прямой слэш (/). Например, тегу «начало курсива» соответствует закрывающий тег .

В закрывающем теге не бывает атрибутов. В HTML большинство тегов, хотя и не все, имеют закрывающий тег. Но чтобы сделать жизнь пишущих на HTML легче, синтаксический анализатор броузера часто по контексту может однозначно определить место, где должен стоять закрывающий тег, поэтому в ряде случаев его присутствие в исходном HTML-тексте не обязательно. В последующих главах мы скажем, какие из закрывающих тегов можно опускать, а какие – нет. В нашем простом примере нет закрывающего тега </p>, местоположение которого обычно легко определяется автоматически, и, следовательно, он

[1] Отображение текста курсивом – это очень простой пример, поддерживаемый большинством броузеров, за исключением разве что текстовых, таких как Lynx. Как правило, броузер пытается сделать то, что ему сказано, но его настройки, как будет показано в следующих главах, могут отличаться в зависимости от компьютера, операционной системы и желания пользователя. Поэтому набор допустимых или отобранных шрифтов может быть непредсказуем. Имейте в виду, что не все могут или хотят отображать ваш HTML-документ в точности так, как вы видите его на своем экране.

так часто не включается в исходный текст, что один опытный HTML-разработчик даже не знал, что этот тег существует. Каково?

Стандарт XHTML гораздо более строг и настаивает на том, чтобы все теги имели соответствующие закрывающие. [закрывающие теги, 16.3.2] [пустые элементы, 16.3.3]

2.4. Костяк HTML

Обратите внимание на то, что в простом примере исходного текста HTML-документ начинается и заканчивается тегами `<html>` и соответственно `</html>`. Они сообщают броузеру, что весь документ написан на HTML.[1] Стандарты HTML и XHTML требуют присутствия тега `<html>`, но большинство броузеров способны определить и адекватно отобразить HTML-разметку текстового документа, в котором не хватает этого самого внешнего тега. [<html>, 3.6.1]

Как и в примере, все HTML- и XHTML-документы (за исключением специальных фреймовых документов) содержат две главные структуры: *заголовок* (head) и *тело* (body), заключенные в тексте каждый в соответствующую пару открывающего и закрывающего тегов. Информация о документе помещается в заголовок, а содержание, которое должно быть отображено в окне броузера, – в тело. За исключением редких случаев большую часть времени занимает работа над содержимым тела документа. [тег <html>, 3.6.1] [тег <body>, 3.8.1]

Есть несколько различных тегов заголовков, которые можно использовать, чтобы задать стиль отображения данного документа, подходящий для какой-либо подборки или большой схемы, размещенной в сети. Кроме того, существуют нестандартные теги заголовка, позволяющие вносить в документ элементы анимации.

Однако в большинстве случаев важнейшим заголовочным элементом является название документа. Стандарты требуют, чтобы каждый HTML- или XHTML-документ имел название, несмотря на то, что популярные в настоящее время броузеры не настаивают на этом правиле. Выбирайте осмысленное имя, которое тотчас скажет читателю, о чем документ. Заключайте это название, как в примере, в теги `<title>` и `</title>` в заголовке документа. Популярные броузеры обычно отображают название в верхней части окна документа. [тег <title>, 3.7.2]

2.5. Плоть HTML– и XHTML–документов

Помимо тегов `<html>`, `<head>`, `<body>` и `<title>` стандарты HTML и XHTML включают еще несколько необходимых структурных элементов. Вы мо-

[1] XHTML-документы тоже начинаются с тега `<html>`, но с дополнительной информацией, чтобы можно было отличить их от обычных HTML-документов. Подробное описание см. в главе 16 «XHTML».

жете свободно вставить в содержимое документа несметное количество разного материала. (Веб-серферы, в числе которых состоят и авторы книги, всласть попользовались этой свободой.) Наверное, удивительно, но существует всего три главных типа содержимого HTML/XHTML-документов – теги (о которых уже шла речь), комментарии и текст.

2.5.1. Комментарии

Черновой документ со всеми вставленными в него тегами может быстро стать таким же неудобочитаемым, как исходные коды программ. Мы настоятельно рекомендуем использовать комментарии, чтобы самим себе помочь разобраться в создаваемых материалах.

Хотя комментарии и являются частью документа, ничего из их содержимого, находящегося между специальным открывающим тегом `<!--` и закрывающим `-->`, не отображается броузером. Посмотрите на комментарии в исходном тексте примера и убедитесь, взглянув на рис. 2.1, что броузер их не отобразил. Всякий тем не менее может скачать исходный текст вашего документа и прочитать комментарии, поэтому следите за тем, что вы пишете.

2.5.2. Текст

Если это не тег и не комментарий, значит, это текст. Основной объем содержимого HTML/XHTML-документа (то, что читатели видят на экране) занимает текст. Специальные теги задают структуру текста – заголовки, списки, таблицы. Остальные теги указывают броузеру, как содержимое документа должно быть отформатировано и выведено на экран.

2.5.3. Мультимедийные элементы

А что же с изображениями и другими мультимедийными элементами, которые мы видим и слышим при воспроизведении страницы нашим броузером? Разве они не являются частью HTML-документа? Нет. Данные, содержащие цифровые изображения, видео, аудио и прочие мультимедийные элементы, которые может показывать и проигрывать броузер, хранятся отдельно от основного HTML/XHTML-документа. В HTML-документ включаются ссылки на них при помощи специальных тегов. Броузер использует эти ссылки, чтобы загрузить и интегрировать документы других типов с вашим текстом.

Мы не включили никаких мультимедийных ссылок в предыдущий пример, потому что они являются отдельными нетекстовыми документами, которые вы не могли бы просто набрать в своем текстовом редакторе. В главе еще пойдет разговор и будет показано, как можно включить изображения и другие мультимедийные элементы в текст. В последующих главах этот вопрос обсуждается весьма детально.

2.6. Текст

Относящихся к тексту тегов разметки в стандартах HTML/XHTML-языков больше всего. Дело в том, что изначально язык HTML появился как средство обогащения структуры и организации текста.

HTML вышел из академической среды. Для первых разработчиков языка было важно, да и сейчас остается важным, чтобы их, главным образом, научные материалы (как правило, текстовые) быстро просматривались и читались, не теряя при этом возможность распространяться в Интернете на разнообразные компьютерные платформы. (Unicode-текст – это единственный универсальный формат[1] в глобальном масштабе Интернета.) Возможность включения мультимедиа – только довесок к HTML и XHTML, хотя и немаловажный.

Макет страницы тоже вторичен по отношению к структуре. Люди просматривают документ визуально и оценивают построение и отношения между частями текста, основываясь на его внешнем виде, тогда как машины могут только читать закодированную разметку. Поскольку теги, содержащиеся в тексте, соотнесены со смыслом документов, они хорошо подходят для компьютерного поиска, а также для автоматизированной обработки содержимого. Все это относится к средствам, очень важным для исследователей. Не так существенно то, *как* сказано, сколько то, *что* сказано.

Таким образом, ни HTML, ни XHTML не являются языками, описывающими расположение текста на странице. Фактически при существующем разнообразии настраиваемых пользователями броузеров и не меньшем разнообразии платформ для отображения электронных документов все, к чему можно стремиться, – это *советовать*, но не предписывать, как должен выглядеть на экране представляемый материал. Невозможно заставить броузер показать документ каким-либо определенным образом. Вы повредитесь в уме, если будете настаивать на противном.

2.6.1. Внешний вид текста

В частности, нельзя предсказать, какой шрифт и какого абсолютного размера – 8 или 40 пунктов, Helvetica, Geneva, Subway или еще какой-то – будет применять конкретный пользователь. Постойте, но новейшие броузеры теперь поддерживают стандарт каскадных таблиц стилей и другие средства настольно-издательского типа, позволяющие

[1] Unicode – это, точнее, не формат, а универсальный стандарт кодирования символов, не зависящий от языка и кодировки; а вот формат представления Unicode может быть различный, например: UTF-16, UTF-8. В наши дни большие надежды возлагаются на применение формата Unicode. Многие современные операционные системы, программные пакеты, в частности броузеры, работают с кодировкой Unicode. – *Примеч. науч. ред.*

управлять компоновкой и внешним видом документов! Да, но пользователь может перенастроить характеристики своего броузера, отвечающие за отображение, и своевольно разрушить ваши старательно разработанные планы. Некоторые старые броузеры не поддерживают новых средств компоновки, а другие работают только с текстом без каких-либо красивых шрифтов. Что же делать? Сосредоточиться на содержании. Крутые странички – это пена на поверхности волн. Глубокое содержание – вот что заставит людей возвращаться раз за разом к вашему документу.

Тем не менее стиль облегчает чтение, поэтому там, где получается, его следует применять, если только это не вредит представлению содержания. Можно задать для своего текста обычные атрибуты стилей при помощи тегов *физических стилей*, таких как тег курсива `<i>` в примере. Более значимо и точно отражают идеологию языка теги *логических* (*контекстно-зависимых*) *стилей*, которые придают *смысловую нагрузку* различным фрагментам текста. Кроме того, можно изменять характеристики отображения текста, такие как стиль и размер шрифта, цвет и тому подобное, с помощью каскадных таблиц стилей.

Современные графические броузеры распознают теги физических и логических стилей и изменяют внешний вид относящихся к ним фрагментов текста, выражая визуально их значение и структуру. Но нельзя точно предсказать, как эти изменения будут выглядеть.

Стандарт HTML 4 (и тем более стандарт XHTML 1.0) подчеркивает, что в будущем броузеры не будут так тесно привязаны к визуальному отображению. Текстовое содержимое можно слушать или распознавать на ощупь, а не только читать глазами. Контекстная разметка в этих случаях подходит, очевидно, лучше, чем физические стили.

2.6.1.1. Логические стили

Теги логических стилей указывают броузеру, что некий фрагмент HTML/XHTML-текста имеет особое значение или употребляется особенным образом. Тег `<cite>` в примере означает, что заключенный в нем текст является какой-то ссылкой[1] – в нашем случае на автора документа. Броузеры обычно, хотя и не всегда, отображают такие ссылки курсивом, в отличие от основного текста. [теги логической разметки, 4.4]

Ясно читателю или нет, что данный текст является ссылкой, но в один прекрасный день кто-нибудь может написать программу, прочесывающую различные документы в поисках тегов `<cite>` и составляющую по заключенному в них тексту специальный список ссылок. Схожие программы уже обшаривают Интернет, составляя списки обнаруживаемой информации, как это делает поисковый сервер Google.

---

[1] Тегом `<cite>` можно обозначить как ссылку, так и цитату. – *Примеч. науч. ред.*

К числу самых часто употребляемых логических стилей в настоящее время относится тег логического выделения, обозначаемый . Ну а если вы настроены особенно решительно, то можете употребить тег . Другие контекстно-зависимые стили включают в себя <code> – для обозначения фрагментов программного кода, <kbd> – для обозначения текста, вводимого пользователем с клавиатуры, <samp> – для выделения примеров, <dfn> – для определений, <var> – для имен переменных в программном коде. У всех этих тегов есть соответствующие закрывающие теги.

2.6.1.2. Физические стили

Даже самые простые редакторы поддерживают несколько традиционных стилей текста, таких как курсив и полужирный шрифт. Не являясь инструментами обработки текста в традиционном смысле, HTML и XHTML предоставляют теги, явно предписывающие броузеру отобразить (если он сумеет) символ, слово или фразу особенным физическим стилем.

Хотя по вышеизложенным соображениям вам следует пользоваться соответствующими логическими стилями, иногда форма важнее содержания. Тогда применяйте тег <i>, чтобы выделить текст курсивом, не приписывая ему какого-либо определенного смысла, тег <b> – для выделения текста полужирным шрифтом, а тег <tt> применяйте, чтобы броузер, если сможет, отобразил текст моноширинным шрифтом. [теги физической разметки, 4.5]

Легко попасть в ловушку использования физических стилей, когда надо было бы применять логические. Стоит выбрать «контекстно-зависимую» манеру изложения, поскольку, как уже было сказано, она несет в себе смысл вместе со стилем, приспосабливая тем самым документ к автоматизации и сопровождению.

2.6.1.3. Специальные символы

Не все символы, отображаемые броузером, можно набрать с клавиатуры. К тому же, некоторые из них применяются как управляющие, например угловые скобки, в которые заключаются теги. Если их специально не выделить, то примененные в основном тексте, например знак «меньше» (<) в математическом неравенстве, они запутают броузер и превратят документ в электронную макулатуру. HTML и XHTML предоставляют способ *символьной замены* для включения с помощью ссылок любого Unicode-символа в произвольное место текста.

Так же как знак «авторского права» (copyright) в нашем примере, код символа начинается с амперсанда (&), за которым идет его условное имя, закрывающееся точкой с запятой. Альтернативный способ записи состоит в употреблении позиционного номера символа в Unicode-кодировке, который ставится вместо имени после знака «решетка» (#). Выводя документ, броузер показывает соответствующий символ, если он присутствует в шрифте, выбранном пользователем. [коды символов, 3.5.2]

По понятным причинам чаще всего кодируются знаки «больше» (`>`), «меньше» (`<`) и амперсанд (`&`). Загляните в приложение F «Коды символов», чтобы узнать, какой из них соответствует записи `¦`. Вы будете приятно удивлены !

2.6.2. Элементы структуры

Из приведенного примера это не видно, но обычный символ возврата каретки[1], используемый для разделения абзацев в исходном документе, не имеет смысла в HTML или XHTML, за исключением специальных случаев. Документ, даже набранный в одну строку, все равно выглядел бы в окне броузера так, как на рис. 2.1.[2]

Вы бы скоро обнаружили, даже не прочитав об этом здесь, что за некоторым исключением броузер игнорирует начальные и конечные пробелы, а иногда и лишние в середине. (Если внимательно рассмотреть исходный текст примера, то видно, что строка «Greetings from» сдвинута вправо с помощью вставленных перед ней пробелов, что не заметно на рис. 2.1.)

2.6.2.1. Раздел, абзац и конец строки

Броузер принимает текст, находящийся в теле документа, и «размещает» его на экране компьютера, не обращая внимания на обычные символы возврата каретки и новой строки, расположенные в исходном тексте. Броузер заполняет каждую строку в своем окне, насколько это возможно: начинает от левого края и, только поместив справа последнее вошедшее слово, переходит на следующую строку. Если изменить размер окна, то текст перетечет, захватывая новое отведенное ему пространство, демонстрируя тем самым гибкость, присущую HTML.

Конечно, читатели возмутились бы, если текст тянулся бы и тянулся одной сплошной цепочкой. Чтобы этого не случилось, HTML и XHTML предоставляют как явные, так и неявные методы управления структурой документа. Самый элементарный и популярный способ состоит в использовании тегов раздела (`<div>`), абзаца (`<p>`) и конца строки (`<br>`). Все они прерывают поток текста, возобновляя его с новой стро-

---

[1] В Unix-системах символ возврата каретки (CR) используется в ASCII-файлах для обозначения конца строки. В DOS и Windows для этих же целей применяется комбинация из символов «перевод строки» (LF) и «возврат каретки» (CR). То, что сказано про символ CR, в полной мере относится и к комбинации LF + CR. – *Примеч. науч. ред.*

[2] Чтобы облегчить чтение HTML/XHTML-документов, мы используем разбиение на строки и организацию отступов так, как делается в языках программирования. Это не обязательно, и нет никаких формальных правил организации исходных HTML/XHTML-файлов. Тем не менее мы настоятельно рекомендуем выработать стиль оформления кода и придерживаться его, чтобы вам и окружающим было легко разбираться в исходных текстах.

ки. Разница в том, что теги и выделяют как блок фрагмент документа и соответственно текста, содержимое которого можно специальным образом выровнять в окне броузера, выделить стилем или изменить с помощью других средств, применимых к блоку текста.

Когда теги и
 не сопровождаются специальными атрибутами выравнивания, они просто обрывают текущую строку и помещают следующие символы на новую строку. У тега вертикальный отступ имеет больший размер, чем у тегов и
. [тег , 4.1.1] [тег , 4.1.2] [тег
, 4.6.1]

Естественно, стандарт HTML определяет закрывающие теги для тегов абзаца и раздела, в то время как «конец строки» им не обладает.[1] Не многие авторы используют в своих документах закрывающий тег абзаца. Броузер обычно может разобраться сам, где один абзац заканчивается и начинается другой.[2] Вручите себе орден, если вы знали, что существует тег .

2.6.2.2. Заголовки

Помимо деления текста на разделы и абзацы, можно формировать документ, применяя озаглавленные разделы. Так же как и на всех страницах этой книги, заголовки не только разбивают текст и именуют его различные разделы, но и несут некий визуальный смысл. При этом заголовки могут оказаться полезными при автоматической обработке документа.

Существует шесть тегов заголовков, от <h1> до <h6>, с соответствующими закрывающими тегами. Обычно броузер отображает их содержимое шрифтами от очень большого до соответственно очень маленького размера, как правило применяя и полужирное написание. Текст в теге <h4> имеет обычно тот же размер, что и основной документ. [теги заголовков, 4.2.1]

Теги заголовков также прерывают поток текста, занимая целые выделенные из него строки, даже если перед заголовком или после него нет явно выписанных тегов абзаца или конца строки.

2.6.2.3. Горизонтальные линейки

Помимо заголовков HTML и XHTML предоставляют горизонтальные линейки, помогающие отделить друг от друга разделы документа.

---

[1] В XHTML начало и конец тега
 заключается в одни и те же скобки:
. Броузеры склонны ко всепрощению и часто игнорируют все посторонние вещи, как прямой слэш (/) в данном случае, так что стоит привыкнуть оставлять такой признак конца.

[2] Завершающий тег абзаца теперь используют чаще, так как популярные броузеры поддерживают атрибуты выравнивания, применяемые к абзацу.

Когда броузер встречает тег <hr>, он прерывает поток текста и в следующей строке рисует через все окно горизонтальную прямую черту. Поток текста возобновляется сразу под линейкой.[1] [тег <hr>, 5.1.1]

2.6.2.4. Преформатированный текст

Бывает, что хочется видеть блок текста на экране таким, как он был написан, скажем с отступами и вертикально выровненными буквами, цифрами, и чтобы вид его сохранялся прежним, даже если окно броузера изменяется в размере. На этот случай есть тег <pre>. Весь текст вплоть до закрывающего тега </pre> представлен на экране точно так, как вы его набрали, включая возвраты каретки, концы строк и начальные, конечные и промежуточные пробелы. Весьма полезный для таблиц и форм, <pre>-текст выглядит тем не менее довольно скучно, так как популярные броузеры отображают его моноширинными шрифтами. [тег <pre>, 4.6.5]

2.7. Гиперссылки

Если сравнить текст с костяком и плотью HTML или XHTML-документа, то гипертекст будет его сердцем. Гипертекст дает возможность отыскать и отобразить любой документ из произвольной подборки (размещенной где угодно в сети), просто щелкнув кнопкой мыши по связующим словам или фразе (*гиперссылке*), расположенной на текущей странице. Используйте эти интерактивные гиперссылки, чтобы помочь читателям путешествовать в поисках нужной информации по вашим или чужим собраниям документов, которые иначе оставались бы просто ворохом отдельных файлов разных форматов, включая мультимедийные, HTML-, XHTML-, прочие XML- и, наконец, обычные текстовые элементы. Гиперссылки, образно говоря, буквально сводят богатства знаний всего Интернета к кончику указателя мыши.

Чтобы вставить ссылку на какой-либо документ, лежащий в собственной коллекции или на сервере в Тимбукту, нужно знать только уникальный адрес документа и способ, как «бросить» *якорь* в вашу разработку.

2.7.1. URL

Как бы ни было трудно в это поверить, но каждый из миллиардов документов и ресурсов Интернета имеет уникальный адрес, именуемый *универсальным указателем ресурса* (Uniform Resource Locator, URL). URL состоит из имени документа, которому предшествует иерархия имен каталогов, содержащих файл (*путь* к документу), *доменного имени* сервера, где он находится, и названия способа (*протокола*), поддерживающего связь и обмен данными между сервером и броузером:

---

[1] Подобно тегу
, в XHTML тег горизонтальной линейки формально записывается <hr/>.

протокол://доменное_имя_сервера /путь_к_документу

Вот несколько примеров URL:

- *http://www.kumquat.com/docs/catalog/price_list.html*
- *price_list.html*
- *../figs/my_photo.png*
- *ftp://ftp.netcom.com/pub/*

Первый пример – это *абсолютный*, или полный, URL. Он содержит все части формата URL: протокол, имя сервера и путь к документу. Хотя абсолютные URL однозначно определены и точны, их употребление может прибавить вам головной боли, если вы перемещаете документы в другой каталог или на новый сервер. К счастью, броузеры позволяют работать с *относительными* URL и автоматически заполняют всякий пропущенный фрагмент соответствующей частью *базового* URL текущего документа. Второй пример дает нам простейший из возможных относительный адрес. Рассматривая его, броузер будет предполагать, что *price_list.html* находится на том же сервере, в том же каталоге, что исходный документ, и применяет тот же сетевой протокол (*http*). Аналогичным образом третий пример является относительным URL, который отсылает броузер наверх и в каталог */figs*, где хранится файл с картинкой.

Хотя внешний вид обманчив, но последний пример URL для FTP является полным. Он указывает прямо на содержимое каталога */pub*. Более того, указание протокола *ftp* в этом примере является обращением к совсем иной программе на сервере, чем та, что работает с протоколом *http* в других примерах.

2.7.2. Якоря

Тег якоря `<a>` – это средство HTML/XHTML для определения как исходной точки, так и пункта назначения гиперссылки.[1] Чаще всего тег `<a>` используется с атрибутом `href` для создания исходной точки ссылки. Значение атрибута `href` – это URL пункта назначения.

Содержимое тега `<a>`, определяющего источник ссылки, – это слова и/или изображение между ним и его закрывающим тегом `</a>`. Они представляют собой особым образом активизированную броузером часть документа, с помощью которой пользователь выбирает гиперссылки. Это *якорное* содержимое обычно отличается по внешнему виду от своего окружения (подчеркнутый или выделенный цветом текст, изобра-

[1] Принятые здесь обозначения несколько неудачны: тег якоря должен был бы отмечать только конечный пункт ссылки, но не то место, с которого она «прыгает». Мы «бросаем якорь», но не прыгаем с него. Не стоит даже упоминать о страшно запутанной терминологии, которую W3C использует для обозначения различных частей гиперссылки. Заметим только, что кое-кто находит положение дел здесь чертовски неуклюжим.

жения в рамке особого цвета), кроме того, экранный образ указателя мыши меняется, когда он попадает на якорь. Содержимое тега <a>, таким образом, должно быть текстом или изображением (пиктограммам здесь самое место), которое явно или намеком говорит пользователю, куда приведет его гиперссылка. [тег <a>, 6.3.1]

Так, в следующем примере броузер специальным образом выводит текст «Кумкват[1]-архив» и изменяет изображение попавшего на него указателя мыши:

```
Дополнительную информацию о кумкватах можно узнать, посетив
<a href="http://www.kumquat.com/archive.html">
Кумкват-архив</a>
```

Если пользователь щелкнет кнопкой мыши по этой ссылке, броузер автоматически запросит с сервера *www.kumquat.com* веб-страницу (с помощью протокола *http:*) под названием *archive.html* и затем отобразит ее на экране.

2.7.3. Имена гиперссылок и навигация

Возможность указывать на документ, находящийся на другом конце света, – это не только круто, но и полезно для ваших собственных веб-страниц. Еще одна важная задача гиперссылок – помогать читателям перемещаться по вашим подборкам в поисках необходимой информации. Естественным образом возникает понятие о домашней странице и сопровождающих материалах.

Ни один из документов не должен тянуться бесконечно. Во-первых, это неудобно для пользователя. Ценность вашей работы, как бы насыщена она ни была, страдает, если нужна вечность, чтобы загрузить документ, а затем, с трудом получив, без конца прокручивать его вверх и вниз, чтобы найти нужный раздел.

Лучше проектируйте свою работу в виде собрания нескольких компактных, сжатых страниц, подобных главам этой книги, каждая из которых посвящена определенной теме так, чтобы ее легко мог выбрать и просмотреть пользователь. Затем применяйте гиперссылки для организации вашей коллекции.

К примеру, используйте домашнюю страницу – ведущий документ подборки – как оглавление с краткими описаниями и соответствующими гиперссылками на остальные части коллекции.

Чтобы особым образом поименовать все разделы документа, вы можете применять тег <a> с атрибутами name или id (последний годится практически для всех тегов). Идентификаторы разделов будут слу-

[1] Кумкват (кинкан, японский апельсин) – разновидность миниатюрных цитрусовых. Кроме того, так же называется направление в современной музыке и дизайне. – *Примеч. науч. ред.*

жить целями внутренних ссылок в коллекции и помогут пользователю ориентироваться в одном документе или перепрыгивать к определенному разделу другого. Ссылайтесь на поименованные разделы, приписывая знак «решетка» (#) и имя раздела после URL документа.

Чтобы сослаться на определенную тему в архиве, скажем, «Рецепты тушеных кумкватов» в кумкват-архиве нашего примера, пометим сначала имя раздела с помощью `id`:

```
... предыдущее содержимое...
<h3 id="Stews">Рецепты тушеных кумкватов</h3>
```

Затем в этом же или другом документе приготовим исходную гиперссылку, которая указывает прямо на данные рецепты, так как включает название раздела в виде суффикса к URL, отделенного от него знаком «решетка»:

```
Дополнительную информацию о кумкватах можно узнать, посетив
<a href="http://www.kumquat.com/archive.html">
  Кумкват-архив</a>,
и, возможно, попробовав наши
<a href="http://www.kumquat.com/archive.html#Stews">
  Рецепты тушеных кумкватов</a>.
```

Последняя гиперссылка, выбранная пользователем, побудит броузер загрузить документ *archive.html* и начать его отображение с раздела «Stews».[1]

2.7.4. Что не под силу якорю

Употребление гиперссылок не ограничено только связью с HTML- и XHTML-документами. Якоря позволяют указывать на практически любые материалы, доступные в глобальной сети, включая другие сервисы Интернета.

Однако «позволить» и «сделать доступным» – это не одно и то же. Броузеры могут работать с различными сервисами, подобными FTP и Gopher, так что удается загружать не только HTML-документы. Но пока они не вполне справляются и не слишком ладят с мультимедийными данными.

Сейчас есть несколько стандартов для многих типов и форматов мультимедийных данных. Компьютерные системы, подключенные к сети, чрезвычайно различаются своими возможностями воспроизведения аудио- и видеоформатов. За исключением некоторых графических изображений, стандарт HTML/XHTML не предлагает никаких специаль-

[1] Здесь подразумевается, что запрошенный документ в окне броузера будет прокручен таким образом, что начало раздела, отмеченное именованным тегом, будет располагаться (если это возможно) вверху видимой части окна. – *Примеч. науч. ред.*

ных средств для представления мультимедийных документов помимо возможности ссылаться на них. Броузер, получивший мультимедийный документ, должен запустить *вспомогательное приложение*, загрузить и исполнить *апплет* или иметь дополнительно устанавливаемый *плагин* (встраиваемый модуль), позволяющий декодировать и показать такой материал прямо внутри демонстрационного окна.

Хотя сегодня HTML и большая часть броузеров избегают подобной неразберихи, просто обходя ее стороной, это не значит, что невозможно или запрещено использовать мультимедийные данные в своих документах. Помните только об имеющихся ограничениях.

2.8. Изображения – это особая статья

Файлы изображений – это мультимедийные элементы, на которые можно ссылаться с помощью якорей в документе для их загрузки и отображения посредством броузера. В отличие от других мультимедийных данных, стандарты HTML и XHTML имеют явные средства, обеспечивающие вывод картинок «в строку» с текстом, при этом на картинках можно создавать сложные карты гиперссылок. Это связано с тем, что существует корпоративный стандарт, касающийся применения только определенных форматов файлов изображений, а именно форматов GIF, PNG и JPEG, и графические броузеры имеют встроенные декодеры, которые включают изображения этих типов в документы.[1]

2.8.1. Встроенные изображения

HTML/XHTML-тег для встроенных изображений – это `<img>`; его обязательным атрибутом `src` является URL файла, который нужно вставить в документ. [тег <img>, 5.2.6]

Броузер загружает изображения и вставляет их в поток текста, представляя как бы специальными, возможно очень большими, символами. Обычно броузер выравнивает изображения и текущую строку по нижней границе. Такое расположение можно изменить с помощью специального атрибута `align` тега `<img>`, значение которого вы устанавливаете равным `top`, `middle` или `bottom`, чтобы изображение позиционировалось по верхней границе, середине или нижней границе примыкающего текста. Проверьте, какое выравнивание вам больше нравится (рис. 2.2–2.4).

---

[1] Некоторые броузеры поддерживают вывод в виде внедренного объекта не только GIF и JPEG, но и других мультимедийных файлов. Internet Explorer, например, поддерживает тег, который проигрывает аудио в фоновом режиме. Кроме того, стандарты HTML 4 и XHTML предоставляют способ отображения других мультимедийных данных, встраивая их в текст документа с помощью общего для них тега.

Конечно, широкая картинка может занять всю строку и таким образом прервать поток текста. Впрочем, вы можете разместить изображение отдельно, включив в документ предшествующие и последующие теги раздела, абзаца или конца строки.

Опытные HTML-авторы не только используют рисунки как иллюстрации, но также добавляют маленькие встроенные символы или глифы (glyphs), чтобы помочь читателю при просмотре и подчеркнуть разделы документа, имеющие особое значение. Ветераны HTML[1] также применяют изображения в качестве маркеров элементов списка и более четких разделителей, чем стандартные горизонтальные линейки. Изображения также могут быть включены в гиперссылку, так что пользователи смогут выбирать вставленную в строку маленькую пиктограмму, чтобы загрузить полноэкранное изображение. Нет числа способам применения встроенных изображений.

Мы также должны упомянуть атрибут `<alt>`. Укажите для него какое-нибудь текстовое значение, чтобы описать изображение тем, кто отключил показ картинок, или чтобы броузеры, способные озвучивать текст, могли прочитать его незрячему пользователю.

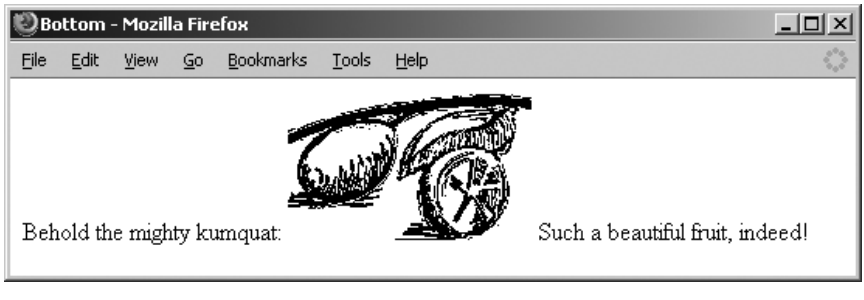

Рис. 2.2. *Встроенное изображение, выровненное по нижнему краю (bottom) текста (по умолчанию)*

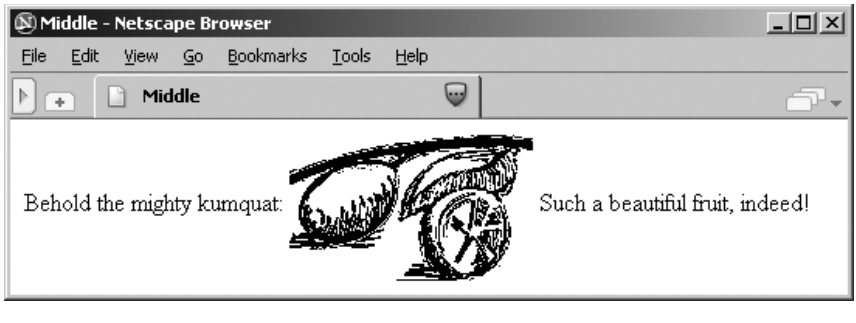

Рис. 2.3. *Встроенное изображение, выровненное по центру (middle) текста*

[1] XHTML слишком молод, чтобы называть кого-либо ветераном или опытным XHTML-автором.

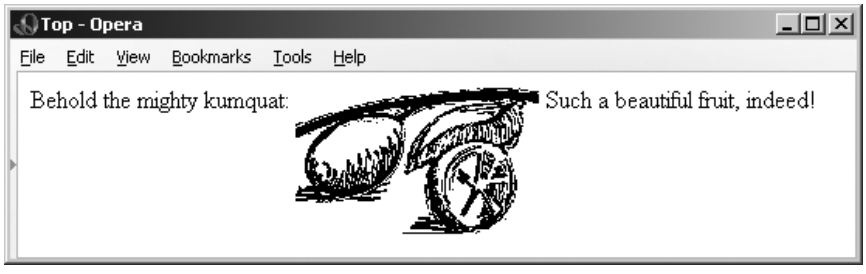

Рис. 2.4. *Встроенное изображение, выровненное по верхнему краю (top) текста*

2.8.2. Карты

Карты[1] – это изображения в теге якоря со специальными атрибутами. Они могут содержать несколько гиперссылок.

Один из способов создания карт состоит в добавлении атрибута `ismap` к тегу `<img>`, помещенному внутри тега `<a>`. Когда пользователь щелкает мышью где-то на рисунке, графический броузер посылает координаты положения мыши серверу, который также указан в якоре. Специальная программа на сервере преобразует эти координаты в некоторое действие, такое как загрузка нового документа. [что происходит на сервере, 6.5.1.1]

Хорошим примером употребления карты может быть выбор отеля по маршруту путешествия. Пользователь щелкает на карте местности, которую он намеревается посетить, и сервер карты возвращает название, адрес и номер телефона местной гостиницы.

Такие мощные и внешне привлекательные карты *со стороны сервера* подразумевают, что автор имеет доступ к программе, обрабатывающей координаты на карте. Тем временем, многие авторы не имеют даже доступа к серверу, не говоря уже о программах на нем. Лучшее решение – воспользоваться картой *со стороны клиента*, или *клиентской*.

Чтобы не зависеть от веб-сервера, можно применять атрибут `usemap` для тега `<img>` вместе с тегами `<map>` и `<area>`, которые позволяют автору вставить всю информацию, нужную броузеру для обработки карты, в тот же документ, где находится и ссылка на ее изображение. Независимость от сервера и меньший объем обмена по сети сделали клиентские карты популярными среди авторов документов и системных администраторов. [клиентские карты, 6.5.2]

---

[1] Более точно, навигационные карты – это разделение объекта (не только изображения) на замкнутые области, с каждой из которых может быть связано некоторое действие: переход по гиперссылке, запуск сценария и т. п. Назначенное действие выполняется, когда соответствующая область активизируется пользователем. – *Примеч. науч. ред.*

2.9. Списки, строка поиска, формы

Полагаете, мы исчерпали все текстовые элементы? Заголовки, абзацы и концы строк – это всего лишь рудиментарные составляющие для организации текста в документе. Кроме них HTML и XHTML предлагают несколько продвинутых текстовых структур, включая три вида списков, строку поиска и формы. Строки поиска и формы находятся за пределами задач форматирования; они служат взаимодействию с читателями документа. Формы дают возможность пользователю вводить текст, ставить флажки в окошках, нажимать на переключатели, делая свой выбор, и затем возвращать информацию серверу. Специальная программа, получив сообщение от формы, обрабатывает его и соответственно отвечает, либо заполняя бланк заказа, либо подбирая данные, интересующие пользователя.[1]

Синтаксис этих специальных средств языка и их разнообразные атрибуты, по-видимому, чрезмерно сложны для быстрого старта. Так что здесь мы их только упоминаем и предлагаем детально с ними познакомиться в следующих главах.

2.9.1. Неупорядоченные, упорядоченные списки и списки определений

Эти три типа списков – как раз те, с которыми мы более всего знакомы. Неупорядоченный список, в котором порядок элементов не имеет значения, такой как перечень бакалейных лавок или прачечных, заключается в теги `<ul>` и `</ul>`. Каждый элемент списка, чаще всего слово или короткая фраза, помечается тегом `<li>` (list item, пункт списка) и – в случае XHTML – закрывающим тегом `</li>`. Обычно список отображается с отступом от левого края. Как правило, броузер снабжает его элементы маркерами. [тег <ul>, 7.1.1] [тег <li>, 7.3]

Упорядоченные списки, заключенные между тегами `<ol>` и `</ol>`, совпадают по формату с неупорядоченными, включая теги `<li>` (и `</li>`, когда речь идет о XHTML), помечающими элементы списка. Однако порядок составляющих списка здесь важен, например последовательность сборки оборудования. Броузер отображает каждый элемент списка, ставя перед ним его номер, и располагает их в возрастающем порядке. [тег <ol>, 7.2.1]

Список определений несколько сложнее предыдущих двух. В списке определений, заключаемом в теги `<dl>` и `</dl>`, каждый элемент состоит из двух частей: имени или заголовка в теге `<dt>` и значения или оп-

[1] Разработка, установка и использование серверных сценариев и программ, которые необходимы для обслуживания форм, находятся за границами нашего повествования. Мы даем некоторые вводные советы в подходящих главах, но за подробной информацией обращайтесь к документации и к администратору сервера.

ределения в теге <dd> (XHTML требует соответствующего закрывающего тега). При отображении броузер обычно помещает имя элемента на отдельной строке без отступа, а определение, которое может состоять из нескольких абзацев – с отступом под именем. [тег <dl>, 7.5.1]

Различные виды списков содержат практически все, что может входить в тело документа. Так, к примеру, вы можете создать упорядоченный список, включающий коллекцию семейных фотографий в оцифрованном виде, или поместить их в список определений, дополнив аннотацией. Стандарты языков разметки позволяют даже помещать списки внутри списков (вложенные списки), открывая интереснейшие способы различных комбинаций.

2.9.2. Строка поиска

Простейшее интерактивное средство, появившееся в ранних версиях HTML и все еще доступное сегодня, хотя и не рекомендуемое стандартами, – это *строка поиска, в основе которой лежит тег* <isindex>. Броузер автоматически предоставляет пользователю какой-либо способ ввести одно или несколько ключевых слов в поле ввода и послать их соответствующему обрабатывающему приложению на сервере.[1]

Ясно, что строки поиска крайне ограничены – один и только один элемент пользовательского ввода на документ. К счастью, HTML и XHTML обеспечивают лучшее, более мощное средство для сбора информации, вводимой пользователем, а именно *формы*. [тег <isindex> (нежелателен), 6.6.1] [тег <form>, 9.2]

Вы создаете один или несколько специальных разделов формы в документе при помощи тегов <form> и </form>, внутри которых можно разместить как жестко определенные, так и настраиваемые окна ввода, допускающие однострочный или многострочный ввод. Туда можно вставить переключатели выбора между двумя и большим числом вариантов, кнопки специального назначения для приведения содержимого формы в начальное состояние и для отправки его на сервер. Пользователи заполняют форму по своему усмотрению, возможно прочитав перед этим остальные части документа, и нажимают специальную кнопку «отправить», которая приказывает броузеру послать данные формы серверу. Специальная программа со стороны сервера обрабатывает данные и соответствующим образом отвечает броузеру, например запрашивая дополнительную информацию или модифицируя документы, отсылаемые затем пользователю, и т. д. [тег <form>, 9.2]

HTML/XHTML-формы предоставляют все средства, которые встречаются у автоматической ее разновидности, включая добавление меток

[1] Этот тег использовался немногими авторами. Тег <isindex> был признан «нежелательным» в стандарте HTML 4.0; отправлен, как говорится, в прошлое, но еще не умер.

областей ввода, встраивание текста команд, установку значения полей по умолчанию и т. п., за исключением автоматической проверки ввода, такой как проверка количества цифр в почтовом индексе или в номере телефона. Эту функцию должна исполнять программа на стороне сервера или JavaScript-сценарий на стороне клиента.

2.10. Таблицы

Как и следовало ожидать от языка, возникшего в научном мире, переполненном цифрами, HTML и его потомок XHTML поддерживают набор тегов для таблиц данных, которые не только выравнивают числа, но и могут специальным образом форматировать текст.

Восемь тегов обеспечивают формирование таблицы, включая собственно тег `<table>` и тег `<caption>`, нужный для задания описания таблицы. Специальные атрибуты позволяют изменять размер и внешний вид таблицы. Вы создаете ее построчно, располагая между тегом строки таблицы (`<tr>`) и его закрывающим тегом (`</tr>`) теги данных (`<td>`) или теги заголовка (`<th>`) и их содержимое для каждой ячейки таблицы (а в случае XHTML еще и закрывающие теги). Заголовки и данные могут содержать практически все что угодно: текст, изображения, формы и даже другие таблицы. Поэтому разумно использовать таблицы для сложного форматирования текста, располагая его, скажем, в нескольких колонках, или для создания боковых заголовков (рис. 2.5). За дополнительной информацией обратитесь к главе 10.

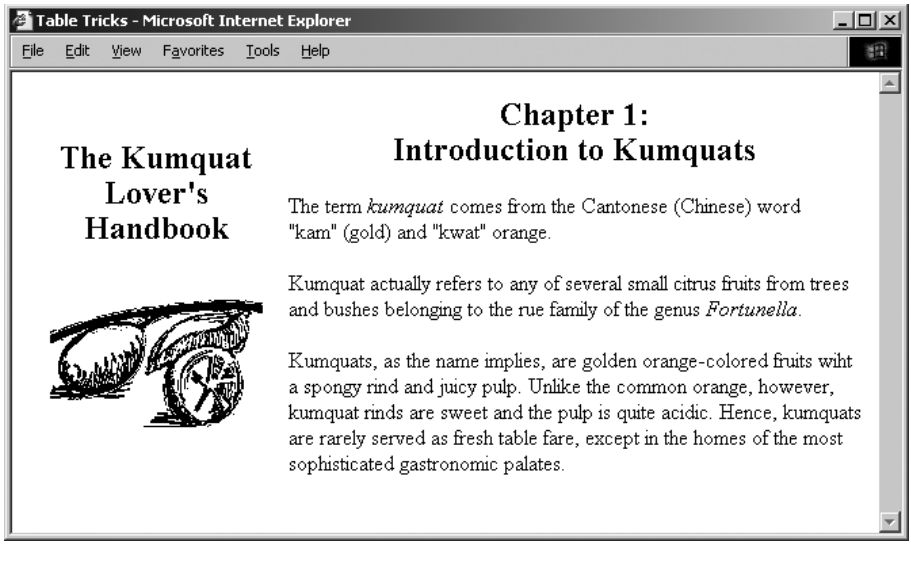

***Рис. 2.5.** Таблицы тоже позволяют совершать чудеса форматирования*

2.11. Фреймы

Всякий, кто открывал на своем экране одновременно несколько окон приложений, немедленно оценит преимущества, которые дают фреймы.[1] Фреймы позволяют разделить окно броузера на несколько областей отображения, в каждой из которых содержится свой документ.

Вот пример отображения фрейма (рис. 2.6). Он показывает, как окно документа может быть разбито на независимые окна, разделенные простыми линейками и полосами прокрутки. А вот чего не видно на рисунке, так это того, что в каждом фрейме может отображаться отдельный документ, который вовсе не обязан иметь HTML/XHTML-происхождение. Фрейм представляет любое значимое содержимое, которое умеет воспроизводить броузер, в том числе мультимедийное. Если содержимое фрейма включает в себя гиперссылку и пользователь ее выбирает, то новый документ или даже другой документ с фреймами может быть отображен либо во фрейме, с которого осуществлялся

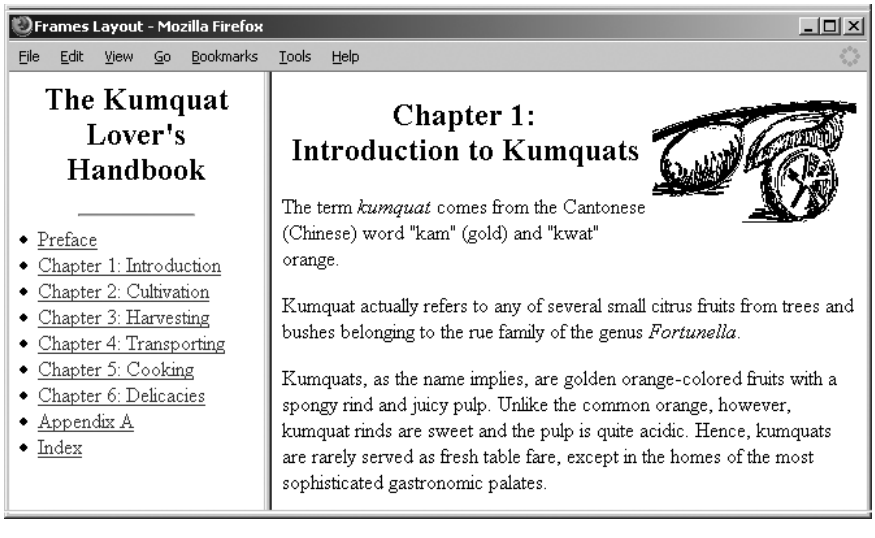

***Рис. 2.6.** Окно броузера разбито на фреймы, в каждом из которых отображается свой документ*

[1] У фреймов есть не только достоинства. Они затрудняют навигацию и установку закладок на загруженные в них документы, содержимое фрейма, не поместившегося в окно броузера, нельзя просмотреть, а если документ не поместился во фрейм, появляются линейки прокрутки. Некоторые броузеры до сих пор не поддерживают фреймы. Поэтому отношение к фреймам предвзятое. Разработчик, решивший использовать фреймы, должен тщательно продумать навигацию и их размещение на странице. Самая распространенная ошибка новичков в веб-строительстве – необоснованное применение фреймов. – *Примеч. науч. ред.*

переход по ссылке, либо в другом фрейме, либо он будет загружен в новое или то же самое окно броузера.

Фреймы определяются в специальном документе, в котором на месте тега <body> находятся один или несколько тегов <frameset>; они и сообщают броузеру, как разделить его окно на несколько отдельных фреймов. Особые теги <frame> содержатся в теге <frameset> и указывают на документы, входящие во фреймы. [тег <frameset>, 11.3.1]

Документ с фреймами управляет окном в целом, хотя отдельные элементы, представленные в своем окне показа, действуют до известной степени независимо. Можно, тем не менее, распорядиться, чтобы документ, находящийся в одном из фреймов, загрузил и отобразил новое содержимое в смежном фрейме. Например, в окне на рис. 2.6 выбор гиперссылки Chapter (глава) во фрейме Table of Contents (содержание) заставит броузер загрузить эту главу и вывести ее в правом фрейме. При этом пользователю всегда доступно оглавление набора документов, который он просматривает в окне с фреймами. За дополнительной информацией обращайтесь к главе 11.

2.12. Таблицы стилей и JavaScript

Кроме того, броузеры включают поддержку двух мощных нововведений HTML – таблиц стилей и JavaScript. Подобно их близким родственницам в настольных издательских системах, таблицы стилей в HTML/XHTML позволяют управлять внешним видом документов – стилями и размером шрифтов, цветом, фоном, выравниванием и т. д. Что более существенно, таблицы стилей дают возможность устанавливать характеристики отображения для документа в целом или даже для полного собрания документов.

JavaScript – это язык программирования с функциями и командами, которые позволяют управлять поведением броузера по отношению к пользователю. Эта книга посвящена не JavaScript, но мы довольно детально обсудим в следующих главах этот язык и покажем, как вставлять программы, написанные на JavaScript, в документы для достижения очень мощных и привлекательных эффектов.

W3C – организация, устанавливающая стандарты, – рекомендует использовать для проектирования HTML/XHTML-документов модель каскадных таблиц стилей (CSS). Все современные броузеры с графическим интерфейсом поддерживают CSS и JavaScript.[1] «Древний» Netscape 4

[1] Строго говоря, язык JavaScript в полной мере поддерживает только Netscape Navigator. Internet Explorer поддерживает язык Jscript, который во многом сходен с JavaScript, но все же имеет некоторые существенные различия. Тот факт, что интерпретатор Jscript, встроенный в Internet Explorer, воспринимает фрагменты кода JavaScript как код своего родного языка, вносит еще больше путаницы в теорию и практику веб-строительства. – *Примеч. науч. ред.*

работает также с моделью таблиц стилей языка JavaScript (JavaScript-based Style Sheets, JSS), которая описана в главе 12, но мы не рекомендуем ее использовать. Нет, мы скажем иначе: «Не тратьте свое время на JSS». CSS – это всеми одобренный и всеми поддерживаемый способ управления тем, как может (но не обязательно будет) отображаться ваш документ броузером пользователя.

Чтобы проиллюстрировать CSS, покажем способ представления заголовков самого верхнего уровня (`h1`) красным цветом:

```
<html>
<head>
<title>Пример использования CSS</title>
<!-- Помещаем описание свойств CSS внутрь комментария, чтобы старые броузеры,
которые не поддерживают CSS, не "споткнулись" или не вывели на экран
непонятное им содержимое тега style. -->
  <style type="text/css">
    <!--
    h1 {color: red}
    -->
  </style>
</head>
<body>
<h1>Если ваш броузер поддерживает CSS, то этот текст будет красным</h1>
Что-нибудь между.
<h1> И этот тоже красный!</h1>
</body>
</html>
```

Вы, разумеется, не сможете увидеть красного цвета в этой черно-белой книге, так что мы не станем показывать результат на рисунке. Поверьте нам или наберите и загрузите этот пример в броузер – заключенный в теги `<h1>` текст будет красным на цветном дисплее.

JavaScript – это объектно-ориентированный язык. Он рассматривает документ и отображающий его броузер как совокупность частей (объектов), обладающих определенными свойствами, которые вы можете изменять или вычислять. Это очень мощное средство, но вряд ли оно посильно большинству. Скорее многие из нас попытаются ухватить несколько крутых JavaScript-программ, распространенных в сети, чтобы украсить ими свои документы. Мы расскажем вам, как это сделать, в главе 12.

2.13. Вперед!

Конечно, данная глава – только верхушка айсберга. Если вы дочитали до этого места, наверное, интерес подстегивает вас и к дальнейшим исследованиям. Теперь вы уже имеете базовое представление об основных чертах и пределах возможностей HTML и XHTML. Переходите к следующим главам, чтобы расширить свои знания и узнать больше о каждой из возможностей языка.

- Внешность обманчива
- Структура HTML-документа
- Теги и атрибуты
- Корректные документы и XHTML
- Содержимое документа
- Элементы HTML/XHTML-документа
- Заголовок документа
- Тело документа
- Редакторская разметка
- Тег <bdo>

3

Анатомия HTML-документа

HTML- и XHTML-документы в основном очень просты, и их написание не смутит даже самого робкого пользователя компьютера. Документ, хотя и составленный вначале с помощью мощного редактора WYSIWYG, в конечном счете будет храниться, распространяться и читаться броузером как простой текстовый файл.[1] Вот почему пользователь, вооруженный даже простейшим текстовым редактором, может строить веб-страницы. (Настоящие веб-мастера часто вызывают восхищение новичков, создавая отменные страницы с использованием примитивных текстовых редакторов на дешевом ноутбуке где-нибудь в автобусе или в ванной комнате.) Тем не менее авторам надо иметь под рукой последние (а также предыдущие) версии нескольких популярных броузеров, чтобы просматривать материал в каждом из них. Не забывайте, что броузеры различаются тем, как они отображают страницу. Не все броузеры поддерживают полный стандарт языка, а некоторые из них имеют собственные расширения.

3.1. Внешность обманчива

Документы, представленные броузером и текстовым редактором, никогда не выглядят одинаково. Взгляните на любой исходный документ в Интернете. Вы увидите, что большая часть символов возврата

[1] Не вдаваясь в тонкости, можно сказать, что и текст, и теги разметки являются ASCII-символами. Формально, если вы не определили другого, текст и теги состоят из восьмибитовых символов, определенных стандартом ISO-8859-1 для набора латинских символов. Некоторые стандарты поддерживают другую кодировку и включают, кроме латинских, арабские буквы или символы кириллицы. Обратитесь к приложению F «Коды символов».

каретки, табуляции и начальных пробелов, важных при чтении текста, игнорируется броузером. В исходном материале есть много включений, по большей части это теги разметки со своими параметрами и атрибутами, которые влияют на отображение разных частей документа, но сами при этом не отображаются.

Начинающие авторы часто противятся необходимости разработать определенный стиль оформления для своих начальных текстов, не связанный прямо со стилями веб-страниц. Макет исходного документа должен подчеркивать сходство HTML-языка разметки с языками программирования, а не ориентироваться на способ, каким будет отображаться содержимое. Исходный текст должен быть разборчивым не только для вас, но и для других читателей.

Опытные создатели документов обычно используют, хотя и в несколько смягченном виде, определенный стиль оформления записи программных кодов. Мы в этой книге действуем так же, и суть этого стиля станет ясной из сравнения примеров исходного текста и его отображения в окне броузера.

Наши принципы форматирования просты, но пригодны для написания документов, которые легко читать и поддерживать:

- За исключением структурных тегов документа, таких как `<html>`, `<head>`, `<frameset>` и `<body>`, элементы, применяемые для структурирования содержимого документа, помещаются на отдельной строке с отступами, отражающими глубину его вложенности в документе. Такие элементы включают списки, формы, таблицы и им подобные теги.
- Элементы, используемые для управления внешним видом или стилем текста, помещаются в текущей строке. Это касается основных стилей шрифта (`<b>`, полужирный шрифт) и гиперссылок (`<a>`, якорь).
- По мере возможности следует избегать переносов URL на следующую строку.
- Пустыми строками в начале и в конце выделяются особые разделы в тексте, например абзацы и таблицы.

Задача систематической организации отступов в исходном тексте бывает тривиальной, но может оказаться и утомительной. Некоторые текстовые редакторы, такие как Emacs, управляют отступами автоматически. Другие совсем о них не заботятся, и подобная задача целиком ложится на ваши плечи. Если текстовый редактор создает вам проблемы, то подумайте о компромиссном решении, например, набирайте структурные теги с отступами, а основной текст – без них, чтобы было легче вносить изменения.

Не имеет значения, какие решения были приняты относительно стиля оформления исходных текстов, – важно им следовать. Вы будете очень рады, что так поступали, когда вернетесь к документу, написанному

три месяца назад, чтобы найти тот классный трюк, который был проделан с... Стоп, а собственно, с чем же?

3.2. Структура HTML–документа

HTML- и XHTML-документы включают текст, определяющий их содержание, и теги, задающие структуру и внешний вид. Структура HTML-документа проста; она состоит из внешнего тега <html>, заключающего в себя документ:[1]

```
<html>
<head>
<title>Простейший HTML-документ</title>
</head>
<body>
Здесь приведен <i>прост</i>ейший пример,
структуры HTML-документа.
</body>
</html>
```

У всякого документа есть *заголовок* и *тело*, ограниченные тегами <head> и <body>. В заголовке указывается заголовок документа и другие параметры, которые могут быть использованы броузером при его отображении. В тело помещается действительное содержание документа. Оно включает в себя отображаемый текст и управляющие маркеры (теги), которые советуют броузеру, как следует его представлять. Теги также ссылаются на файлы спецэффектов, включая графику и звук, и отмечают особые области документа (*гиперссылки* и *якоря*), связывающие ваш документ с другими документами.

3.3. Теги и атрибуты

Теги – элементы разметки в HTML и XHTML – по большей части легко понимать и употреблять, так как они составлены из обычных английских слов, сокращений и обозначений. Например, теги и предлагают броузеру начать и закончить отображение содержащегося между ними текста курсивом (italic). Поэтому слог simp (в слове simple –

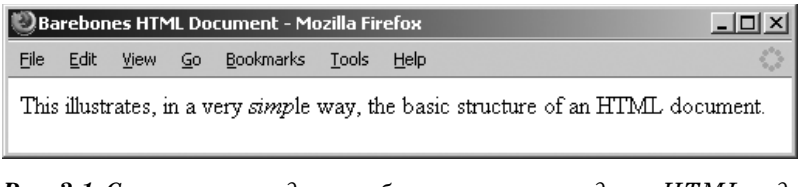

Рис. 3.1. *Сравните вывод в окне броузера с его исходным HTML-кодом, приведенным ранее*

[1] Структура XHTML-документа несколько сложнее. Ее мы рассмотрим в главе 16.

«прост») в вышеприведенном примере будет представлен в окне броузера *курсивом*.

Стандарты HTML и XHTML с их различными расширениями определяют, как и где следует расставлять теги в созданном материале. Рассмотрим внимательнее тот синтаксический сироп, которым склеены все документы.

3.3.1. Синтаксис тегов

Каждый тег состоит из *имени* тега, за которым может следовать список *атрибутов*, при этом все перечисленное заключено между открывающей и закрывающей угловыми скобками (< и >). Простейший тег – это не что иное, как имя в скобках, например `<head>` и `<i>`. Более сложные теги содержат один или несколько *атрибутов*, определяющих или изменяющих его действие.

В соответствии со стандартом HTML имя и атрибуты тега нечувствительны к регистру. Нет никакой разницы, напишете ли вы `<head>`, `<Head>`, `<HEAD>` или даже `<HeaD>`, – все они эквивалентны. В XHTML регистр имеет значение – все имена и атрибуты тегов действующего стандарта набираются строчными буквами; всегда пишите `<head>` и никогда – `<HEAD>`.

И для HTML, и для XHTML значения, приписываемые конкретному атрибуту, могут быть чувствительны к регистру; это зависит от вашего броузера и от сервера. В частности, ссылки на расположение и имя файла (URL) чувствительны к регистру. [ссылки на документы: URL, 6.2]

Атрибуты тега, если они имеются, располагаются вслед за именем тега, разделенные одним или несколькими символами табуляции, пробела или возврата каретки. Порядок, в котором записываются атрибуты тега, безразличен.

Присваиваемое атрибуту значение помещается за именем атрибута через знак равенства (=). Вы можете вставлять пробелы вокруг знака равенства, так что `width=6`, `width = 6` и `width =6` означают одно и то же. Для разборчивости, однако, мы предпочитаем не добавлять пробелы. Так легче выхватывать глазом пару атрибут-значение из набора пар в длинном теге.

В HTML, если значением атрибута служит отдельное слово или число (без пробелов), вы можете просто написать их после знака равенства. Все другие значения должны быть заключены в одинарные или двойные кавычки, особенно те из них, что содержат набор слов, разделенных пробелами.[1] В XHTML все значения атрибутов должны быть заключены в кавычки. Длина значения ограничена 1024 символами.

[1] Или слово, содержащее кириллические символы, поэтому в наших условиях возьмите за правило заключать в кавычки все символьные значения (не числа), тем более что эта привычка сослужит добрую службу при наборе XML или XHTML. – *Примеч. науч. ред.*

Большинство броузеров терпимо относятся к переносам тегов. Тем не менее, если возможно, избегайте расползания тега на несколько строк. Это правило сослужит добрую службу при чтении и уменьшит риск ошибок в HTML-документах.

3.3.2. Примеры тегов

Вот несколько тегов с атрибутами:

```
<a href="http://www.oreilly.com/catalog.html">
<ul compact>
<ul compact="compact">
<input type=text name=filename size=24 maxlength=80>
<link title="Полное оглавление">
```

Первый пример – это тег `<a>`, определяющий гиперссылку на каталог продукции издательства O'Reilly. У него один атрибут `href`, которому присвоено значение адреса каталога в киберпространстве, – его URL.

Второй пример показывает тег HTML, форматирующий текст в виде неупорядоченного списка. Его единственный атрибут, `compact`, ограничивающий интервалы между элементами списка, не требует значения.

Третья строка демонстрирует, как второй пример должен быть написан в XHTML. Отметьте, что атрибут `compact` имеет значение, хотя и избыточное, и оно заключено в двойные кавычки.

Четвертая строка представляет тег HTML с несколькими атрибутами, каждому из которых присваивается значение, не требующее кавычек. Разумеется, в XHTML каждое значение должно быть заключено в двойные кавычки.

Последний пример показывает верное употребление кавычек, когда значение атрибута состоит из нескольких слов.[1]

Единственное, что не очевидно из этих примеров, так это чувствительность большей части значений атрибутов к регистру, хотя сами HTML-атрибуты безразличны к нему (`href` работает в HTML так же, как `HREF` или `HreF`). Значение `filename` атрибута `name` в теге `<input>` – это не то же самое, что значение `Filename`.

3.3.3. Открывающие и закрывающие теги

Мы упоминали уже, что большинство тегов имеют начало и конец и действуют на часть содержимого документа, заключенную между ними. Этот размещенный в теге сегмент может быть большим или малень-

[1] В оригинале значение тега записано как «Table of Contents», поэтому для сохранения логики примера оно было заменено на «Полное оглавление»; но если бы мы оставили точный эквивалент: «Оглавление», то и здесь требовались бы кавычки, но уже согласно другому требованию – помещать в кавычки символы, не принадлежащие латинскому набору. – *Примеч. науч. ред.*

ким, от одного символа, слога или слова, такого как отображаемый курсивом слог в слове «прост» в нашем простейшем примере, до тега <html>, заключающего в себе весь документ. Любой начальный тег – это имя тега и его атрибуты, если последние имеются. Закрывающий тег состоит только из имени тега, которому предшествует наклонная черта (/). Закрывающий тег не имеет атрибутов.

3.3.4. Правильная и неправильная вложенность тегов

Тег может быть вложен в область действия другого тега, чем достигается сложение их действий на общий сегмент документа. К примеру, фрагмент следующего текста одновременно отображается полужирным шрифтом и является частью якоря, определяемого тегом <a>:

```
<body>
Этот текст тела документа со
<a href="another_doc.html">ссылкой, часть которой
<b>будет полужирной.</b></a>
</body>
```

В соответствии со стандартами HTML и XHTML следует завершать вложенные теги, начиная с самого внутреннего и проделав весь путь в обратном направлении – по принципу «первым вошел, последним вышел». В нашем примере это означает, что мы завершаем тег полужирного шрифта () перед завершением тега гиперссылки (), поскольку начинали в противоположном порядке – сначала тег <a>, затем тег . Следовать этому стандарту – хорошее начало, хотя большинство броузеров на нем даже не настаивает. Вы, скорее всего, не пострадаете, нарушив это правило вложенности при работе с одним, а иногда даже со всеми современными броузерами. Но, в конце концов, какая-нибудь новая версия перестанет допускать это уклонение от стандарта, и вам придется выправлять исходный HTML-документ. Кроме того, имейте в виду, что стандарт XHTML явным образом запрещает употребление неправильно вложенных тегов.

3.3.5. Без закрывающих тегов

Согласно стандарту HTML у некоторых тегов нет закрывающих тегов. Более того, они просто запрещены к использованию, хотя большинство броузеров снисходительно игнорируют ошибочный закрывающий тег. Например, тег
 вызывает переход на новую строку, не оказывая никакого действия на следующую за ним часть документа, и, следовательно, не нуждается в закрывающем теге.

HTML-теги, не имеющие соответствующих закрывающих:

<area>	<base>	<basefont>
 	<col>	<frame>
<hr>	<img>	<input>

<isindex> <link> <meta>

<param>

XHTML всегда требует закрывающий тег. [пустые элементы, 16.3.3]

3.3.6. Отсутствующие теги

Часто можно видеть документы, в которых авторы, видимо, забыли вставить закрывающий тег в явном противоречии со стандартами HTML и XHTML. Иногда бывает, что потерян тег <body>. Но броузер не протестует, и документ отображается прекрасно. Как же так? Стандарт HTML позволяет опускать некоторые теги или их завершения для удобства и ясности. Авторы стандарта HTML не стремились сделать его нудным.

К примеру, тег , определяющий начало абзаца, имеет соответствующий закрывающий тег , который, правда, употребляется редко. Фактически, многие HTML-авторы не знают даже о его существовании. [тег , 4.1.2]

В целом, стандарт HTML позволяет опустить открывающий или закрывающий тег всякий раз, когда его наличие может быть недвусмысленно выведено из контекста. Многие броузеры делают верные догадки, сталкиваясь с потерянными тегами, а авторы думают, что опустили тег в соответствии со стандартом.

Мы рекомендуем практически всегда добавлять закрывающий тег. Это облегчит вашу жизнь при переходе на XHTML и понравится как броузеру, так и всякому, кому придется модифицировать ваш документ в будущем.

3.3.7. Игнорируемые и лишние теги

HTML-броузеры порой игнорируют теги. Это обычно случается с лишними тегами, действие которых отменяется или поглощается ими самими. Наилучший пример дает последовательность тегов , идущих один за другим без текста в промежутке между ними. В отличие от текстового редактора, большинство броузеров перейдет на новую строку только один раз.[1] Все остальные теги лишние, поэтому они обычно игнорируются броузерами.

Так же большинство HTML-броузеров пропускают любой непонятный им тег (или неправильно определенный автором). Броузеры обычно идут дальше и придают документу какой-то смысл, как бы плохо он ни был написан. Это не только тактика борьбы с ошибками, но и важный стратегический ход в сторону расширяемости. Вообразите только, на-

[1] Точнее, начнет новый абзац, а переход на новую строку будет лишь следствием этого действия: новый абзац должен начинаться с новой строки. – *Примеч. науч. ред.*

сколько труднее было бы добавлять новые возможности языка, если бы все старые броузеры спотыкались на них.

За чем нужно внимательно следить при использовании нестандартных тегов, не поддерживаемых большинством броузеров, так это за заключенным в них содержимым, если оно имеется. Броузеры, признающие новый тег, могут обрабатывать его содержимое иначе, чем те, которые его не поддерживают. Например, старые броузеры, многие из которых до сих пор используются, не поддерживают стилей. Послушные долгу, они игнорируют тег <style> и выводят ограничиваемое им содержимое на экран, не только аннулируя действие этого тега, но и нарушая внешний вид документа. [таблицы стилей на уровне документа, 8.1.2]

3.4. Корректные документы и XHTML

XHTML – это благонравный родственник HTML. HTML-документы, способные выиграть конкурсы красоты в качестве образцово полных и правильно составленных документов, которые написаны в соответствии с этой книгой, вплоть до снабжения каждого тега <p> закрывающим тегом, могут быть отвергнуты экспертами по XML как неполноценные файлы.

В соответствии с XML язык XHTML настаивает, чтобы документы были «корректными» (well-formed). Помимо всего это означает, что все теги, даже такие, как
 и <hr>, для которых стандарт HTML запрещает использование закрывающих тегов, должны иметь закрывающие теги. В XHTML завершение помещается внутрь открывающего тега, например
.[1] [пустые элементы, 16.3.3]

Это также означает, что имена тегов и атрибутов чувствительны к регистру и в соответствии с действующим стандартом XHTML необходимо записывать строчными буквами. Следовательно, приемлемо только написание <head>, но *не* <HEAD> или <HeAd>, в отличие от стандарта HTML. [зависимость от регистра, 16.3.4]

Корректные XHTML-документы, как и соответствующие стандарту HTML-документы, требуют правильной вложенности тегов. И никаких возражений! [корректно вложенные элементы, 16.3.1]

В свою защиту стандарт XML и его порождение XHTML отдают предпочтение расширяемости. То есть тег <p> может означать в HTML начало абзаца, тогда как другая разновидность языка определит содержимое тега <P> отвечающим результатам предвыборных опросов, вывод которых оформлен весьма специфическим образом, возможно в виде

[1] Из всего предыдущего текста это не очевидно: согласно правилам XML любой тег, не обрамляющий «тело» (т.е. некоторый фрагмент текста), указывающийся, как правило, только для задания некоторых его атрибутов, может закрываться конструкцией />. – *Примеч. науч. ред.*

таблицы с белыми, синими и красными полосками в сопровождении патриотической музыки.[1]

Подробнее об этом читайте в главах 15 и 16, в которых мы детально обсуждаем стандарты XML и XHTML (и условия соответствия им).

3.5. Содержимое документа

Практически все остальное, что помещается в HTML- или XHTML-документ и не является тегом, по определению является содержимым и по преимуществу текстом. Подобно тегам, содержимое кодируется особым множеством символов, по умолчанию соответствующим набору латинских символов стандарта ISO-8859-1. Это множество является расширением стандартного ASCII, в который добавлены необходимые символы для поддержки восточноевропейских языков.[2] Если клавиатура не позволяет прямо вводить нужные символы, можно использовать их коды, чтобы делать необходимые вставки.

3.5.1. Приказывать или советовать

Возможно, самое трудное для запоминания правило разметки HTML- и XHTML-материалов состоит в том, что все теги, вставляемые для форматирования текста и управления внешним видом документа, могут только советовать броузеру, но не могут жестко руководить способом его отображения.

Броузер фактически может проигнорировать все ваши теги и проделать с содержимым документа все, что ему заблагорассудится. Еще хуже то, что пользователи (всех национальностей и соответственно языков!) настраивают свои броузеры, произвольно изменяя характеристики отображения текста.

Привыкайте к этому отсутствию жесткого контроля за внешним видом документа. Лучший способ управлять им – это сосредоточиться на содержании документа, а не на конечном представлении. Если вы будете слишком беспокоиться по поводу интервалов, выравнивания, раз-

---

[1] Формальный «педантизм» XML (XHTML), описываемый автором, имеет следствием достоинство, с лихвой перекрывающее возникающую громоздкость: любой XML-текст может быть подвержен формальной машинной проверке на синтаксическую правильность структуры документа. – *Примеч. науч. ред.*

[2] Было бы слишком грустно, если бы на способ кодирования накладывались столь жесткие ограничения, но обычно англоязычные авторы, не испытывающие таких проблем, не озабочены их детальным освещением. HTML и XML/XHTML легко позволяют указать использование любой кодировки из многих десятков вплоть до Unicode UTF-8 (ISO 10646), решающей все проблемы локализации. В HTML это может делаться, например, с помощью атрибутов тега `<meta>` секции `<head>`; в XML кодирование может указываться в заголовке `<?XML encoding=``...`` ?>`. – *Примеч. науч. ред.*

рывов текста и позиционирования символов, то закончите язвой желудка, ибо выйдете за пределы сферы влияния HTML. Если вы сосредоточитесь на предоставлении информации пользователю в привлекательном виде, используя теги только для того, чтобы советовать, как было бы лучше подать материал, тогда применение HTML и XHTML будет эффективным и ваши документы будут удовлетворительно выводиться большинством броузеров.

3.5.2. Коды символов

Наряду с обычным текстом, HTML и XHTML дают возможность отображать специальные текстовые символы, которые нельзя обыкновенным способом включить в исходный документ, или те, что имеют особое значение в языке. Хорошим примером является символ «знак меньше» (<). В HTML он обычно означает начало тега, так что если вы вставите его просто как часть текста, броузер запутается и, вероятно, интерпретирует документ неправильно.

И в HTML, и в XHTML знак «амперсанд» (&) предупреждает броузер об использовании специальной последовательности символов, именуемой *кодом символа* (*символьной заменой*). К примеру, команда `<` вставляет пресловутый знак «меньше» в выводимый текст, и броузер уже не будет ошибочно считать его началом тега. Подобным образом `>` вставляет символ «больше», а `&` – знак «амперсанда». Между амперсандом, именем символа и необходимой в конце точкой с запятой не может быть пробелов. (Точка с запятой – это не специальный символ; нет необходимости в использовании кода для его отображения.) [как обращаться со специальными символами, 16.3.7]

Также можно подставить вместо названия символа, следующего за амперсандом, знак решетки (#) и десятичное число, соответствующее этому знаку в кодовой таблице. Так, последовательность `<` соответствует тому же самому, что и `<`, и представляет знак «меньше». Действительно, можно заменить все обычные символы в HTML-документе их кодами с амперсандами, скажем вставляя `A` вместо заглавной «A» или `a` вместо ее строчного аналога, но это будет глупо. Полный список всех символов, их названий и численных эквивалентов можно найти в приложении F.

Имейте в виду, что не все специальные символы могут быть отображены любым броузером. Некоторые броузеры просто игнорируют многие из них. Символы бывают недоступны и вследствие особенностей платформы. Не забудьте протестировать свой материал на ряде броузеров, прежде чем решить использовать какой-нибудь особо редкий символ в документе.

3.5.3. Комментарии

Комментарии – это еще один вид текстового содержимого, появляющегося в исходных документах HTML, но не отображаемого броузе-

ром. Комментарии находятся между специальными (<!-- и -->) элементами разметки. Броузеры игнорируют текст между этими последовательностями символов, выделяющими примечания автора.

Вот образец комментариев:

```
<!-- Это комментарий -->
<!-- Это комментарий,
занимающий несколько строк
и кончающийся в данной строке -->
```

Необходимо вставлять пробелы после начального <!-- и перед закрывающим --> элементом, но в остальном вы вольны поместить в качестве примечаний практически любой текст.

Главное исключение состоит в том, что стандарт HTML запрещает вложенные комментарии.[1]

Internet Explorer позволяет также размещать комментарии внутри специального тега <comment>. Все, находящееся между тегами <comment> и </comment>, Internet Explorer игнорирует. Другие броузеры покажут эти записи пользователю. Чтобы избежать такого нежелательного поведения, не следует применять тег <comment>. Вместо него комментарии лучше обозначать последовательностями <!-- и -->.

Помимо естественного употребления комментариев в исходных материалах, многие веб-серверы используют их для реализации в документах программных возможностей, специфических для данного сервера. Такие серверы просматривают документ в поисках специальных последовательностей символов в стандартных HTML- и XHTML-комментариях и затем совершают какие-то действия, основываясь на обнаруженных директивах. Эти действия могут быть такими простыми, как включение текста из другого файла (известное как *серверное включение*, server-side include, SSI), или такими сложными, как исполнение на сервере команд для динамического генерирования содержимого документа.

3.6. Элементы HTML/XHTML-документа

Всякий HTML-документ должен соответствовать HTML SGML DTD – формальному определению типа документа (DTD), описывающему стандарт HTML. Набор правил DTD определяет теги и синтаксис, используемые для создания HTML-разработок. Чтобы сообщить броузеру, какому DTD соответствует ваш материал, можно поместить специальную SGML-директиву (Standard Generalized Markup Language, стандартный обобщенный язык разметки) в первой строке документа:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
```

[1] Netscape допускает вложенные комментарии, но хитрым способом, а как другие броузеры к ним отнесутся, предсказать невозможно.

Это таинственное объявление означает, что ваш документ намерен соответствовать окончательному DTD HTML 4.01, определенному World Wide Web Consortium (W3C). Другие версии DTD определяют более ограниченные модификации стандарта HTML, и не все броузеры поддерживают полный набор версий HTML DTD. Фактически, обозначив какой-нибудь другой тип, вы можете заставить броузер неправильно истолковать документ при его отображении. Также неясно, какой тип документа указывать, когда вы пользуетесь нестандартными, хотя и популярными расширениями или, скажем, тегами устаревшего стандарта HTML 3.0, для которого DTD так и не был выпущен.

Сейчас все больше разработчиков ставит в начале своего HTML-документа название его SGML-типа. Вследствие имеющей место путаницы версий и стандартов, уж если вы решили указать DOCTYPE в своем HTML-документе, делайте это внимательно, чтобы документ отображался корректно.

Тем, кто пишет на XHTML, мы настоятельно рекомендуем указывать тип документа в XHTML-разработке в соответствии со стандартом XML. Читайте главы 15 и 16, чтобы получить более полную информацию о DTD и стандартах XML и XHTML.

3.6.1. Тег <html>

Как мы уже знаем, теги `<html>` и `</html>` служат для обозначения начала и конца документа. Поскольку типичный броузер может по заключенному содержимому легко сообразить, что он имеет дело с HTML- или XHTML-разработкой, то включать этот тег в исходный документ необязательно.

Тем не менее разумно включать этот тег в документы, чтобы другие программы, особенно обработчики текста (менее умные, чем броузеры), могли распознать в нем HTML-разработку. По крайней мере, присутствие открывающего и закрывающего тегов гарантирует, что начальная или конечная часть документа не будут случайно удалены. Кроме того, XHTML требует обязательного наличия тегов `<html>` и `</html>`.

Между тегами `<html>` и `</html>` содержатся заголовок и тело документа. Внутри заголовка вы найдете теги, которые идентифицируют документ и определяют его место в коллекции документов. В теле документа на-

<html>

Функция:	Показывает границы всего HTML- или XHTML-документа
Атрибуты:	`dir` [I], `lang`, `version`
Закрывающий тег:	`</html>`; в HTML может быть опущен
Содержит:	*head_tag*, *body_tag*, *фреймы*

ходится собственно его содержание вместе с тегами, определяющими макет и внешний вид текста. Как и следовало ожидать, заголовок документа содержится между тегами <head> и </head>, а тело – между тегами <body> и </body>, которые подробно описываются далее в этой главе.

Самая обычная форма тега <html> проста:

```
<html>
заголовок документа и его тело
</html>
```

3.6.1.1. Атрибут dir

Атрибут dir определяет, в каком направлении броузер должен выводить текст в элементе, к которому применяется этот атрибут. Включенный в тег <html>, он устанавливает, как текст будет представлен во всем документе. Примененный к другому тегу, он назначает направление текста только для содержимого данного тега.

По умолчанию значение этого атрибута – ltr (left-to-right). Следовательно, текст представляется пользователю слева направо. Применяйте другое значение, rtl, чтобы отображать текст справа налево для таких языков, как арабский и иврит. Разумеется, результат будет зависеть от содержимого и от того, поддерживает ли броузер стандарт HTML 4 или XHTML. Netscape и Internet Explorer версии 4 и ниже игнорируют атрибут dir. Совместимый со стандартом HTML 4, Internet Explorer версий 5 и 6 просто выравнивает по правому краю текст, находящийся под действием (dir=rtl), хотя, взглянув на рис. 3.2, вы заметите, что броузер переместил пунктуацию (точку) на другой конец предложения. Netscape 6 выравнивал по правому краю все, даже точки в конце предложений, но в версиях 7 и 8 это уже не происходит (еще один признак окончания броузерных войн):

```
<html dir=rtl>
<head>
<title>Display Directions</title>
</head>
<body>
This is how IE 6 renders right-to-left directed text.
</body>
</html>
```

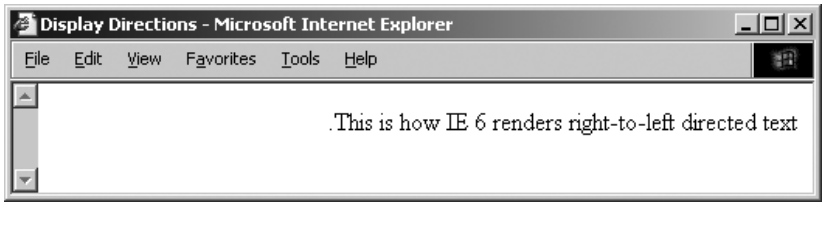

Рис. 3.2. *Все современные броузеры просто выравнивают по правому краю текст с атрибутом dir и кроме того они нарушают пунктуацию*

3.6.1.2. Атрибут lang

Включенный в тег <html>, атрибут lang определяет язык, который, главным образом, употребляется в документе. Использованный с другим тегом, атрибут lang назначает язык содержимого тега. В идеале броузер в конечном счете применяет lang, чтобы лучше отображать текст.

Вводимые значения атрибута lang соответствуют двухбуквенным кодам языков, определенным в стандарте ISO-639. Вы также можете обозначить диалект, поставив за кодом языка тире и код диалекта. Например, «en» – это ISO-код для английского языка; «en-US» – полный код для американского диалекта английского языка. Другие основные коды включают: «fr» (французский), «de» (немецкий), «it» (итальянский), «ul» (голландский), «el» (греческий), «es» (испанский), «pt» (португальский), «ar» (арабский), «he» (иврит), «ru» (русский), «zh» (китайский), «hi» (хинди).

3.6.1.3. Атрибут version

Атрибут version определяет версию стандарта HTML, использованного при составлении документа. Его значение для HTML версии 4.01 выглядит в точности так:

```
version="-//W3C//DTD HTML 4.01//EN"
```

В целом, от указания версии в теге <html> толку мало, и этот атрибут признан нежелательным в стандарте HTML 4. Серьезным авторам следует вместо него использовать SGML-тег <!DOCTYPE>, например:[1]

```
<!DOCTYPE HTML PUBLIC "-//W3C/DTD HTML 4.01//EN"
     "http://www.w3c.org/TR/ html4/strict.dtd">
```

3.7. Заголовок документа

Заголовок документа описывает в целом его различные свойства, включая название, нахождение в сети и взаимосвязь с другими документами. Большая часть данных, содержащихся в заголовке, никогда не отображается броузером.

3.7.1. Тег <head>

Тег <head> служит для того, чтобы заключать в себя другие теги заголовка. Ставьте его в начале документа, сразу за тегом <html> и перед тегом <body> или <frameset>. И тег <head> и его закрывающий тег </head>

[1] В данном примере используется строгое определение типа документа (Strict DTD). Кроме того, в HTML 4.01 заданы DTD для документов, использующих фреймы (Frameset DTD), и документов, которые могут содержать нежелательные и устаревшие элементы (Traditional DTD). Более подробную информацию смотрите в приложении D. – *Примеч. науч. ред.*

<head>	
Функция:	Определяет заголовок документа
Атрибуты:	dir, lang, profile
Закрывающий тег:	</head>; редко опускается в HTML
Содержит:	*head_content* (содержимое заголовка)
Может содержаться в:	*html_tag* (тег <*html*>)

могут быть недвусмысленно «предсказаны» броузером, и поэтому ничто не мешает их опустить. Однако мы советуем включать их в документ, поскольку они облегчают чтение и автоматическую обработку документа.

Тег <head> может содержать много других тегов, которые помогают определить содержимое документа и управлять им. К ним относятся: <base>, <isindex>, <link>, <meta>, <nextid>, <object>, <script>, <style> и <title>. Все они могут появляться в любой очередности.

3.7.1.1. Атрибуты dir и lang

Как было сказано в разделах, посвященных атрибутам тега <html>, атрибуты dir и lang помогают адаптировать HTML и XHTML для международной аудитории. [атрибут dir, 3.6.1.1] [атрибут lang, 3.6.1.2]

3.7.1.2. Атрибут profile

Часто заголовок документа содержит множество тегов <meta>, используемых для сообщения броузеру добавочной информации о документе. В будущем авторы смогут применять для более подробного описания своих материалов заранее определенные профили стандартных метаданных документа. Атрибут profile предоставляет URL профиля, связанного с данным документом.

Формат профиля и то, как он будет использоваться броузером, еще не определены. Этот атрибут пока лишь отмечает направление будущих разработок.

3.7.2. Тег <title>

Тег <title> делает в точности то, что вы ожидали: слова, помещенные в нем, определяют название документа. (Предмет обсуждения в большой степени сам себя объясняет проще, чем можно было бы подумать.) Название используется броузером специальным образом и чаще всего размещается в заголовке окна или в строке состояния. Кроме того, название обычно становится именем по умолчанию для ссылок на документ, когда он добавляется в коллекцию избранных ссылок или в пользовательский список предпочтений.

<title>	
Функция:	Определяет название документа
Атрибуты:	`dir`, `lang`
Закрывающий тег:	`</title>`; присутствует обязательно
Содержит:	*plain_text* (простой текст)
Может содержаться в:	*head_content* (содержимое заголовка)

Тег `<title>` является единственным обязательным объектом в заголовке документа. Поскольку `<head>` и даже `<html>` бывают опущены, тег `<title>` может оказаться первой строкой в действительном[1] (valid) HTML-документе. Более того, большинство броузеров даже дадут вашему документу какое-либо родовое имя, если в нем отсутствует тег `<title>`, назвав его, например, по имени файла, в котором он содержится. Следовательно, вовсе не обязательно указывать наименования, но это уже слишком, даже для нашего низменного вкуса. Никакой уважающий себя автор не выпустит документ без тега `<title>` и без названия.

Когда указываете тег `<title>`, не забывайте закрывать его тегом `</title>`. В противном случае текст названия будет выведен в теле документа, даже если он предшествует тегу `<body>`.

Броузеры никак не форматируют текст названия документа и игнорируют в теге `<title>` все, кроме текста. В частности, они пропустят все изображения и любые ссылки.

А вот еще более жалкий пример действительного HTML-документа, в который включены только теги заголовка и названия. Посмотрите, как Netscape отображает его (рис. 3.3).

```
<html>
<head>
<title>HTML & XHTML: The Definitive Guide</title>
</head>
</html>
```

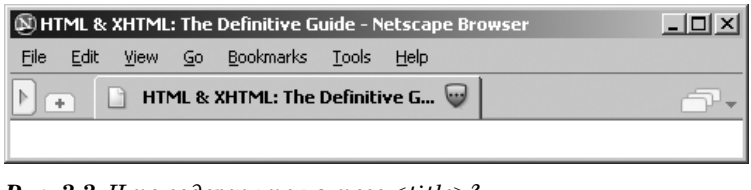

Рис. 3.3. *Что содержится в теге <title>?*

[1] Согласно спецификации XML, документ считается действительным, если он имеет DTD (Document Type Definition) и удовлетворяет ему. – *Примеч. науч. ред.*

3.7.2.1. Что следует помещать в теге <title>?

Выбор правильного названия категорически важен для определения характера документа и гарантии того, что он сможет эффективно функционировать в Интернете.

Учтите, что пользователи могут получать доступ к каждому документу в собрании в любом порядке и независимо от всех остальных. Следовательно, название должно определять не только место вашего документа среди других, но и говорить о его собственных достоинствах.

Названия, включающие в себя ссылки на последовательный номер документа, обычно не годятся. Простые названия – «Глава 2», «Часть 6» – не помогают пользователю догадаться, о чем пойдет речь. Более описательная манера изложения, например «Глава 2: Мастерство бального танца» или «Часть 6: Юность и зрелость Черчилля», дает сведения о положении документа в некотором собрании и о его особенном содержании, приглашающем читателя ознакомиться с ним.

Ничего не дает ссылка на самое себя. Наименование «Моя домашняя страница» полностью лишено содержания; тем же отличаются «Страница отзывов» или «Популярные ссылки». Вам нужно название, несущее сведения о содержании и назначении документа, чтобы пользователи могли, основываясь только на нем, решить, посетить вашу страницу или нет. «Домашняя страница любителей кумкватов» что-то описывает и, вероятно, привлечет любителей этого горького фрукта так же, как «Книга отзывов любителей кумкватов» или «Популярные среди любителей кумкватов ссылки».

Люди проводят много времени, создавая документы для Сети, часто лишь затем, чтобы блеклым непривлекательным названием перечеркнуть свой труд. В связи с возрастающей популярностью программ, автоматически собирающих ссылки для пользователей, единственной фразой, описывающей содержание ваших страниц в огромной базе данных, могут оказаться их названия, выбранные вами. Нельзя переборщить, повторяя снова и снова: позаботьтесь о том, чтобы выбрать содержательное, полезное, описывающее отличия от других название для каждого из ваших документов.

3.7.2.2. Атрибуты dir и lang

Атрибуты `dir` и `lang` помогают адаптировать HTML и XHTML для международной аудитории. [атрибут dir, 3.6.1.1] [атрибут lang, 3.6.1.2]

3.7.3. Теги, относящиеся к заголовку

Другие теги, которые вы можете включать в тег `<head>`, относятся к отдельным аспектам создания документа, управления им, связывания его с другими документами, автоматизированной обработки и разметки. Вот почему мы только упомянем их здесь, а подробное описание отложим до других, более подходящих разделов и глав этой книги.

Вот, вкратце, специальные теги заголовка:

`<base>` и `<link>`

Они определяют базовый адрес (который будет использоваться для относительных ссылок) текущего документа и ссылки на другие документы (например, другие версии данного документа или внешние таблицы стилей). [элемент заголовка <base>, 6.7.1] [элемент заголовка <link>, 6.7.2]

`<isindex>`

Нежелательный в HTML 4 тег `<isindex>` раньше применяли для создания автоматизированных форм поиска, позволявших искать в базах данных нужную информацию, используя текущий документ как средство осуществления запросов. [тег <isindex> (нежелателен), 6.6.1]

`<nextid>`

Не поддерживается в HTML 4 и XHTML. Тег `<nextid>` облегчает создание уникальных меток при использовании автоматической обработки. [элемент заголовка <nextid> (архаизм), 6.8.2]

`<meta>`

Содержит дополнительные данные о документе, которые нельзя поместить в другие теги заголовка. [элемент заголовка <meta>, 6.8.1]

`<object>`

Определяет методы, которыми броузер может обрабатывать нестандартные объекты. [тег <object>, 12.2.1]

`<script>`

Определяет один или несколько сценариев, которые могут быть вызваны элементами документа. [тег <script>, 12.3.1]

`<style>`

Позволяет определять свойства каскадных таблиц стилей (CSS) для управления характеристиками отображения документа в целом. [таблицы стилей на уровне документа, 8.1.2]

3.8. Тело документа

Тело документа – это основа основ; именно здесь помещается содержание документа. Теги `<body>` и `</body>` определяют его границы.

3.8.1. Тег <body>

В HTML 4 и XHTML тег `<body>` располагает множеством атрибутов, управляющих цветом и фоном документа. Различные броузеры имеют расширения этого тега для достижения еще большего контроля над внешним видом отображения.

Все, что находится между открывающим тегом `<body>` и закрывающим тегом `</body>`, называется *содержимым тела*. Простейший документ может состоять лишь из последовательности абзацев, заключенной

<body>	
Функция:	Определяет тело документа
Атрибуты:	alink, background, bgcolor, bgproperties, class, dir, id, lang, leftmargin, link, onBlur, onClick, onDblClick, onFocus, onKeyDown, onKeyPress, onKeyUp, onLoad, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, onUnload, style, text, title, topmargin, vlink
Закрывающий тег:	</body>; в HTML может опускаться
Содержит:	*body_content* (содержимое тела)
Может содержаться в:	*html_tag* (тег <*html*>)

в тег <body>. Более сложные документы будут содержать основательно отформатированный текст, рисунки, таблицы и множество специальных эффектов.

Поскольку положение тегов <body> и </body> легко прогнозируется броузером, они могут быть безболезненно опущены в HTML-документе, но не в XHTML-документе. Однако так же, как теги <html> и <head>, теги <body> и </body> облегчают чтение и удобны для поддержки исходного документа, поэтому мы рекомендуем их применять.

Различные атрибуты тега <body> могут быть разделены на три группы: те, что позволяют до известной степени управлять внешним видом документа, те, что связывают функции, встроенные или определенные автором в сценариях, с самим документом, и те, что помечают и именуют тело для последующих на него ссылок. Мы обратимся к атрибутам внешнего вида alink, background, bgcolor, bgproperties, leftmargin, link, text, topmargin и vlink в главе 5, к атрибутам class и style, связанным с каскадными таблицами стилей, – в главе 8, к таблицам стилей JavaScript и к атрибутам программирования (атрибуты «по событию») – в главе 12. Языковые атрибуты (dir и lang) были рассмотрены выше в этой главе, а атрибуты идентификации (id и title) будут разобраны в главе 4. [атрибут dir, 3.6.1.1] [атрибут lang, 3.6.1.2] [атрибут id, 4.1.1.4] [атрибут title, 4.1.1.5]

3.8.2. Фреймы

Стандарты HTML и XHTML определяют специальный тип документов, в которых тег <body> заменен на тег <frameset>. Эти так называемые *фреймовые* документы делят окно вывода на несколько независимых окон, в каждом из которых отображается свой источник. Мы подробно опишем это нововведение в главе 11.

3.9. Редакторская разметка

HTML 4.0 вводит два новых тега, которые могут помочь группам авторов сотрудничать при разработке документов и поддерживать кон-

троль за их версиями и процессом редактирования. Теги вставки <ins> и удаления <del> позволяют помечать части тела документа как новые или добавленные либо выделять старый материал, который подлежит замене. Специальные атрибуты позволяют отмечать время внесения изменений (datetime) и ссылаться на документы, которые могут объяснять причину модификаций (cite).

3.9.1. Теги <ins> и <del>

Теги <ins> и <del> позволяют авторам выделять те части содержимого тела документа, которые они намереваются в него вставить или удалить из текущей редакции. HTML 4/XHTML-совместимые броузеры отображают содержимое тегов <ins> или <del> специальным образом, чтобы читатели могли быстро обнаруживать изменения в документе.

<ins> и <del>

Функция:	Определяют вставленное и удаленное содержимое документа (см. рис. 3.4)
Атрибуты:	cite, class, datetime, dir, id, lang, onClick, onDblClick, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, style, title
Закрывающий тег:	</ins> и </del>; присутствуют обязательно
Содержит:	*body_content* (содержимое тела)
Может содержаться в:	*body_content* (содержимое тела)

Netscape 4 так же, как Internet Explorer 4 и их более ранние версии, игнорируют эти теги, а все современные популярные броузеры, наоборот, поддерживают.

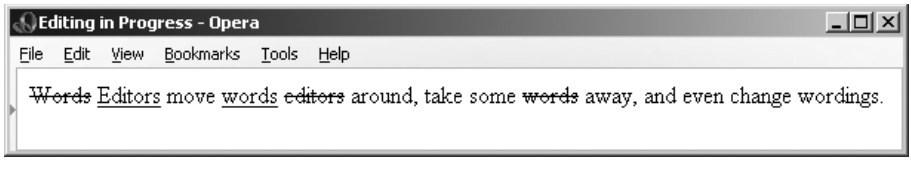

***Рис. 3.4.** Теги <ins> и <del> в действии*

3.9.1.1. Атрибут cite

Атрибут cite позволяет ссылаться на документ, на основании которого сделаны вставка или удаление. Его значением должен быть URL, указывающий на какой-то другой источник, поясняющий вставленный/удаленный текст. Как с атрибутом cite будет обходиться броузер, это вопрос будущего.

3.9.1.2. Атрибут datetime

Хотя причины внесенных изменений важны, знать, когда они были проведены, еще важнее. Атрибут `datetime` для тегов `<ins>` и `<del>` принимает единственное значение – закодированное особым образом указание на дату и время правки.

Полный формат значения атрибута `datetime` – это `YYYY-MM-DDThh:mm:ssTZD`. Вот его компоненты:

- `YYYY` – год из четырех цифр, например 1998 или 2010.
- `MM` – месяц, от 01 для января до 12 для декабря.
- `DD` – день месяца, от 01 до 31.
- `T` – обязательный символ, обозначающий начало сегмента, отражающего время суток.
- `hh` – час в 24-часовом формате, от 00 до 23. (Далее следует двоеточие, если нужно указать минуты.)
- `mm` – минуты, от 00 до 59. (Далее следует двоеточие, если нужно указать секунды.)
- `SS` – секунды, от 00 до 59.
- `TZD` – обозначение часового пояса. Оно может принимать одно из трех значений: `Z`, обозначающее время по Гринвичу[1], или смещение относительно Гринвича, выраженное в часах, минутах и секундах с указанием знака смещения.

Например,

```
2007-02-22T14:26Z
```

расшифровывается как 22 февраля 2007 года в 14:26 по Гринвичу. Чтобы обозначить восточное стандартное время (EST), код для той же даты и того же часа будет:

```
2007-02-22T19:26-05:00
```

Заметьте, что местное время может меняться в зависимости от места редактирования документа, тогда как универсальное время остается одним и тем же.

3.9.1.3. Атрибуты class, dir, event, id, lang, style, title и events

Существует несколько практически универсальных атрибутов, которые могут использоваться со многими тегами HTML и XHTML. Эти атрибуты дают вам общий способ идентифицировать (`title`) и снабжать метками (`id`) содержимое тегов для последующих ссылок или автоматической обработки; изменять характеристики отображения (`class`, `style`); указывать применяемый язык (`lang`) и направление, в котором

[1] Время по Гринвичу (всемирное среднее время, UTC) также известно как Зулусское (Zulu) время, отсюда обозначение «Z».

должен отображаться текст (dir). На внешние события[1], которые могут происходить в выделенном тегами фрагменте текста или где-то еще, вы можете отреагировать с использованием атрибута «по событию» и некоторого программного кода. [атрибут dir, 3.6.1.1] [атрибут lang, 3.6.1.2] [атрибут id, 4.1.1.4] [встроенные стили: атрибут style, 8.1.1] [стилевые классы, 8.3] [JavaScript-обработчики событий, 12.3.3]

3.9.2. Использование редакторской разметки

Удобство тегов <ins> и <del> совершенно очевидно для всех, кто когда-либо использовал шаблон документа или заполнял бланки, а также для тех, кто сотрудничал с кем-либо при подготовке материала.

К примеру, адвокатская контора обычно располагает собранием шаблонных общедоступных юридических документов, особым образом подготавливаемых для каждого клиента. Заполнение производят обычно секретари фирмы, а результат просматривает юрист. Для выделения тех мест документа, где были внесены изменения, – чтобы сделать их заметными при просмотре – используется тег <ins> (для обозначения добавленного секретарем текста) и тег <del> (выделяющий замененный текст). По желанию используйте атрибуты cite и datetime для указания, когда и зачем были сделаны изменения.

К примеру, секретарь мог заполнить шаблон документа, вписать в него название фирмы и имя представителя, обозначив время и источник оформления.

```
Представитель первой стороны,
<ins datetime=2002-06-22T08:30Z
    cite="http://www.mull+dull.com/tom_muller.html">

Томас Мюллер, совладелец компании "Мюллер и Дюллер"
</ins>
<del>[вставьте представителя компании]</del>
```

Теги редакторской разметки могут применяться также для отслеживания процесса последовательных изменений, вносимых автором в течение некоторого промежутка времени. При условии правильного использования атрибутов cite и datetime может оказаться возможным восстановление версии документа, относящейся к заданному моменту времени.

3.10. Тег <bdo>

Как уже ранее отмечалось, авторы стандарта HTML 4 предприняли согласованные действия, чтобы включить в него типовые способы пред-

[1] Броузер отслеживает все действия пользователя в своем окне (нажатия клавиш, перемещение мыши, загрузку и выгрузку документов и т. п.), при этом на каждое действие генерируется соответствующее событие (event). Сообщение о событии можно перехватить и как-либо отреагировать с помощью программы-обработчика события. – *Примеч. науч. ред.*

<bdo>	
Функция:	Аннулирует предписания о направлении вывода содержимого
Атрибуты:	`class`, `dir`, `id`, `lang`, `style`, `title`
Закрывающий тег:	`</bdo>`; присутствует обязательно
Содержит:	*текст*
Может содержаться в:	*body_content* (содержимое тела)

полагаемых действий броузеров при обработке и отображении множества различных естественных языков и диалектов. В соответствии с этим стандарт HTML 4 и его потомок XHTML содержат универсальные атрибуты `dir` и `lang`, позволяющие прямо сообщить броузеру, что весь документ или часть его, заключенная в некоторый тег, написаны на определенном языке. Эти относящиеся к языку атрибуты могут влиять на отдельные характеристики отображения. К примеру, атрибут `dir` предлагает броузеру писать на дисплее либо слева направо, как в большинстве западных языков, либо справа налево, как во многих азиатских языках. [атрибут dir, 3.6.1.1] [атрибут lang, 3.6.1.2]

Различные стандарты кодирования и отображения языков (Unicode и ISO) могут добавить вам проблемы. В частности, не исключено, что содержимое какого-то документа, такого как файл в MIME-кодировке, уже хорошо отформатировано, а ваш документ может ошибочно приказать броузеру отменить эту кодировку. В связи с этим в стандарты HTML 4 и XHTML включен тег `<bdo>`. С его помощью вы отменяете текущие и унаследованные спецификации атрибута `dir`. А с помощью присущего и обязательного для этого тега собственного атрибута `dir` вы окончательно определяете направление, в котором содержимое тега должно быть отображено.

Вот пример (рис. 3.5), как Internet Explorer отображает следующий HTML-фрагмент, содержащий перенаправление текста с помощью тега `<bdo>`:

```
<bdo dir=rtl>This would be readable if in Chinese, perhaps.</bdo>
Back to the Western way of reading and writing.
```

Надо признаться, эффекты тега `<bdo>` несколько заумны, кроме того, случай его применить встречается нечасто.

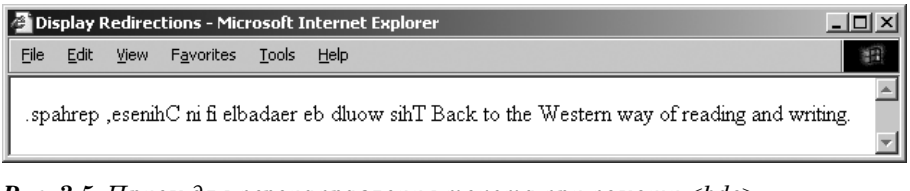

Рис. 3.5. *Прием для перенаправления текста при помощи <bdo>*

4

Текст

- Разделы и абзацы
- Заголовки
- Управление внешним видом текста
- Теги логической разметки
- Теги физической разметки
- Точные интервалы и макет
- Блоки цитат
- Адреса
- Кодирование специальных символов
- Использование расширенной HTML-модели шрифтов (нежелательно)

В любой успешной публикации, будь то даже глубокомысленный трактат, текст должен быть организован привлекательным и эффектным образом. Превращение записей в притягательный документ – это как раз то, что дается HTML и XHTML лучше всего. Данные языки предоставляют множество средств для упорядочения текста и внятного изложения его основного смысла. Кроме того, они помогают так структурировать документ, что каждое слово легко находит путь к читателю.

При разработке документов всегда помните (мы повторяем это и будем повторять), что теги разметки, особенно в отношении текста, только советуют – они не в силах приказывать, – как броузеру следует вывести документ. Вывод варьируется от броузера к броузеру. Не усердствуйте чрезмерно в попытках добиться единственного, на ваш взгляд, правильного внешнего вида. Таким стараниям почти всегда помешает броузер.

4.1. Разделы и абзацы

Подобно большинству текстовых процессоров, броузер располагает текст на экране во всю ширину окна, переходя на следующую строку по достижении его края. Сделайте окно шире, и текст автоматически растечется, заполняя окно удлинившимися строками. Сожмите окно, и текст выдавится обратно.

Но, в отличие от большинства процессоров, HTML и XHTML явным образом применяют теги раздела `<div>`, абзаца `<p>` и конца строки `<br>` для управления выравниванием и потоком текста. Возврат каретки[1], хотя

[1] Введенный непосредственно в потоке ввода текста. – *Примеч. науч. ред.*

и очень полезный для удобства чтения, обычно игнорируется броузером, и авторы должны применять тег
, чтобы явно указать конец строки. Тег <p>, который также вызывает переход на следующую строку, несет в себе дополнительный смысл помимо возврата каретки.

Тег несколько отличается от тегов <p> и
. Включенный впервые в стандарт HTML 3.2, он был задуман, чтобы служить простым средством организации текста и разбивать документ на отдельные куски. В силу смысловой неопределенности этого тега он оставался непопулярным. Но последние нововведения (атрибуты выравнивания и стилей, а также атрибут id для организации ссылок и автоматической обработки) позволяют теперь яснее помечать отдельные фрагменты документа, придавая им особый характер, равно как и управлять их внешним видом. Эти возможности придали тегу новый смысл и стимулировали его использование.

Присваивая атрибутам id и class имена в разных секциях документа, разграниченных тегами (так поступают и с другими тегами, например с <p>), вы не только помечаете их для последующего обращения к ним при помощи гиперссылок или для автоматической обработки и поддержки (в частности, составления списка библиографических данных по разделам), но можете также определить явно различающиеся стили для этих частей документа. К примеру, можно ввести класс разделов, содержащих аннотацию к документу (скажем,), другой класс – для основного текста, третий – для заключения и четвертый – для библиографии ( ).

Затем каждому классу может быть присвоен собственный способ отображения как на уровне документа, так и с помощью внешней присоединенной таблицы стилей: аннотация выводится с отступом и курсивом (скажем, div.abstract {left-margin: +0.5in; font-style: italic}); основной текст – выровненным по левому краю прямым шрифтом; заключение – так же, как аннотация; библиография – с применением автоматической нумерации и подходящим образом отформатированная.

Мы дадим детальное описание таблиц стилей, классов и их приложений в главе 8.

4.1.1. Тег

Как это определено в стандартах HTML 4.01 и XHTML 1.0 и 1.1, тег разбивает документ на отдельные различающиеся части. Он может использоваться как чисто организационное средство, без какого-либо ассоциированного с ним форматирования, но становится более эффективным, если вы добавляете в него атрибуты id и class, помечающие раздел. Тег может комбинироваться также с атрибутом align для управления выравниванием всего раздела документа при отображении, а также с множеством связанных с программированием атрибутов «по событию» для взаимодействия с пользователем.

Функция:	Определяет блок текста
Атрибуты:	align, class, dir, id, lang, nowrap, onClick, onDblClick, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, style, title
Закрывающий тег:	; обычно опускается в HTML
Содержит:	*body_content* (содержимое тела)
Может содержаться в:	*блоке*

4.1.1.1. Атрибут align

Атрибут `align` с тегом `<div>` помещает содержимое тега либо в левой части окна (`left`, по умолчанию), либо посередине (`center`), либо справа (`right`). Кроме того, ему можно присвоить значение `justify`, чтобы выровнять текст одновременно по левому и правому краю. Тег `<div>` допускает вложения, и выравнивание во вложенном разделе для его содержимого имеет больший приоритет, чем аналогичная операция во внешнем. При этом другие вложенные теги выравнивания, такие как `<center>`, выравнивание абзаца (`<p>` в разделе 4.1.2) или особенным образом выстроенные строки и ячейки таблиц, отменяют результаты действия тега `<div>`. Подобно атрибутам `align` других тегов, он признан нежелательным стандартами HTML и XHTML, уступая место управлению макетированием вывода с помощью таблиц стилей.

4.1.1.2. Атрибут nowrap

Поддерживаемый броузерами Internet Explorer и Opera, но не Firefox или Netscape Navigator и не одобренный текущими стандартами атрибут `nowrap` подавляет автоматическое разделение текста на строки в пределах раздела. Переход на новую строку при этом происходит только там, где в исходном тексте стоит возврат каретки.

Хотя применение атрибута `nowrap` не выглядит осмысленным для больших разделов документа, которые бы в итоге сливались на странице, он может чуть облегчить жизнь при создании фрагментов с большим количеством переходов на новую строку, например, в стихах или адресах. Вам не придется явно вставлять все теги `<br>` в поток текста, заключенный в тег `<div nowrap>`. С другой стороны, большое количество пользователей, броузеры которых проигнорируют атрибут `nowrap`, увидит ваш текст слитым в сплошной поток. Если вы ориентируетесь только на Internet Explorer или Opera, используйте при необходимости `nowrap`, но мы никак не можем рекомендовать этот атрибут для всеобщего употребления.

4.1.1.3. Атрибуты dir и lang

Атрибут `dir` советует броузеру, в каком направлении следует выводить текст, а атрибут `lang` позволяет определить употребляемый в разделе язык. [атрибут dir, 3.6.1.1] [атрибут lang, 3.6.1.2]

4.1.1.4. Атрибут id

Применяйте атрибут `id` для обеспечения разделов документа метками, позволяющими ссылаться на них из гиперссылок, таблиц стилей, апплетов, сценариев и прочих программ автоматической обработки. Вообще говоря, приемлемым значением атрибута `id` является заключенная в двойные кавычки строка, которая уникальным образом именует раздел и в дальнейшем может быть использована для недвусмысленной ссылки на него. Более конкретно, это значение должно начинаться с буквы и может содержать буквы, цифры, дефисы, двоеточия, символы подчеркивания и точки, но ни в коем случае не пробелы. Хотя мы представляем этот атрибут в контексте тега `<div>`, он может употребляться практически с любым тегом.

Применив значение атрибута `id` как идентификатор элемента, можно затем добавлять его к URL, адресуясь таким образом к помеченной части определенного документа. Вы вправе маркировать именами большие фрагменты текста (посредством тега `<div>`, например) и маленькие его отрывки (применяя `id` в таких тегах, как `<i>` или `<span>`). В частности, если пометить аннотацию технического доклада, использовав `<div id="abstract">`, то ее URL может быть записан так: `report.html#abstract`. В случае такого применения значение атрибута `id` должно быть уникальным для данного документа среди значений других `id` и всех значений атрибута `name`, определенных в тегах `<a>`. [гиперссылки внутри одного документа, 6.3.3]

Употребленное в качестве селектора в таблице стилей значение атрибута `id` становится именем правила, которое таким образом ассоциируется с текущим тегом. Так получается второй набор определяемых стилевых правил, подобный тому, который вы вправе создать для классов стилей. Тег может использовать оба атрибута – и `class` и `id` – для наложения двух различных стилей на содержимое одного тега. В этом употреблении имя, ассоциированное с атрибутом `id`, должно быть уникальным среди всех других имен стилей в текущем документе. Более полное описание классов стилей и имен стилей расположено в главе 8.

4.1.1.5. Атрибут title

Используйте необязательный атрибут `title` и заключенную в кавычки строку – значение атрибута для привязывания к разделу описывающей его фразы. Подобно `id`, атрибут `title` может применяться почти со всеми тегами и ведет себя с ними одинаково.

Для значения атрибута `title` не предусмотрено никакого специального использования, однако современные броузеры отображают название элемента всякий раз, когда на нем задерживается мышь, в данном случае – в любой точке текстовой области, определяемой тегом `<div>`. Таким способом, если правильно употреблять атрибут `title`, можно организовать всплывающие подсказки для разных элементов документа.

4.1.1.6. Атрибуты class и style

Используйте атрибут `style` с тегом `<div>`, чтобы определить встроенный стиль отображения содержимого тега. Атрибут `class` позволяет применять к содержимому тега стиль, заранее присвоенный данному классу разделов. Значение атрибута `class` – это имя стиля, определенного в таблице стилей, находящейся либо на уровне документа, либо вне его. Кроме того, разделы, для которых установлена принадлежность классу, хорошо подходят для компьютерной обработки документов, такой как, например, извлечение всех разделов, принадлежащих классу «biblio», при автоматической сборке полной библиографии. [встроенные стили: атрибут style, 8.1.1] [стилевые классы, 8.3]

4.1.1.7. Атрибуты событий

Многие относящиеся к поведению пользователя события как внутри раздела, так и вне его, в частности щелчок или двойной щелчок мышью в области отображения раздела, распознаются броузером, если он отвечает текущим стандартам HTML или XHTML (а все популярные броузеры этим качеством обладают). С помощью соответствующих атрибутов «по событию», иначе называемых «on»-атрибутами[1], и их значений можно реагировать на эти события, отображая окно диалога с пользователем или активизируя какое-нибудь мультимедийное событие. [обработчики событий JavaScript, 12.3.3]

4.1.2. Тег <p>

Тег `<p>` сигнализирует о начале нового абзаца. Это не очень хорошо известно даже искушенным веб-мастерам, поскольку противоречит нашим интуитивным ожиданиям, основанным на опыте. Большинство текстовых редакторов, с которыми мы знакомы, используют только один специальный символ (обычно это возврат каретки) для обозначения *конца* абзаца. В HTML и XHTML каждый абзац должен начинаться с тега `<p>` и завершаться соответствующим тегом `</p>`. И в то время как последовательность символов новой строки или возврата каретки

[1] Имена таких атрибутов обслуживания начинаются с `on`, после чего следует название события, по возникновению которого данный атрибут «сработает». Например, за попадание указателя мыши в область отображения данного тега отвечает атрибут `onmouseover`, а за двойной щелчок мыши в этой же области – `ondblclick`. – *Примеч. науч. ред.*

Функция:	Определяет абзац текста
Атрибуты:	align, class, dir, id, lang, onDblClick, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, style, title
Закрывающий тег:	; часто опускается в HTML
Содержит:	*текст*
Может содержаться в:	*блоке*

в документе, отображаемом текстовым редактором, появляющаяся при многократном нажатии на клавишу <Enter>, порождает по одному пустому абзацу на каждый такой символ, броузеры обычно игнорируют все[1] теги абзаца и символы новой строки, кроме первого.

На практике в HTML можно опустить открывающий тег в начале первого абзаца и тег в конце всех абзацев – они могут быть логически выведены из других тегов, встречающихся в документе, следовательно, их не обязательно указывать.[2] К примеру:

```
<body>
This is the first paragraph, at the very beginning of the
body of this document.
<p>
The tag above signals the start of this second paragraph. When rendered
by a browser, it will begin slightly below the end of the first paragraph,
with a bit of extra white space between the two paragraphs.
<p>
This is the last paragraph in the example.
</body>
```

Отметьте, что мы не включили в пример тег начала абзаца ( ) для первого параграфа и не вставили ни одного тега завершения абзаца. Они могут быть недвусмысленно вычислены броузером и не являются, следовательно, необходимыми.

Вообще говоря, несложно заметить, что люди, создающие документы, склонны опускать все вычисляемые теги везде, где это возможно, тогда как автоматические генераторы документов склонны вставлять их. Это оттого, быть может, что разработчики программного обеспечения не хотят рисковать тем, что конкуренты станут бранить их продукт за нарушение стандарта HTML, даже если речь идет о мелочном следовании букве закона. Вперед, не робейте – выбрасывайте эти в первом абзаце и не беспокойтесь о закрывающих до тех пор, конечно, пока

[1] Непосредственно следующие друг за другом. – *Примеч. науч. ред.*

[2] XHTML, с другой стороны, требует явного указания открывающего и закрывающего тегов.

от этого не пострадает структура и ясность вашего документа. Но не забывайте, что XHTML не допускает подобной распущенности.

4.1.2.1. Отображение абзаца

Встретив тег нового абзаца <p>, броузер обычно вставляет в документ перед его началом пустую строку, несколько увеличенную по высоте против обычной. Затем броузер собирает в этот абзац все слова и, если они присутствуют, встроенные изображения, опускает все пробелы перед ними и в их конце (но не пробелы между словами, конечно) и символы возврата каретки, находящиеся в исходном тексте. Далее программа заполняет получившейся последовательностью слов и изображений окно броузера, автоматически генерируя переходы на новую строку по достижении края окна. Сравним, как броузер располагает текст в строках и абзацах (рис. 4.1), с тем, как выглядит предыдущий пример, напечатанный на странице.

Броузер умеет автоматически переносить длинные слова со строки на строку и «полностью выравнивать» абзац, так что текст будет равномерно растянут между левым и правым краем окна.

В результате при составлении документов не нужно заботиться ни о длине, ни о разбиении строки. Броузер примет любую последовательность слов и изображений и предъявит вам хорошо отформатированный абзац.

Если вы хотите сами управлять длиной строки и переходами с одной строки на другую, подумайте об использовании блоков текста, форматированных с помощью тега <pre>. Если необходимо принудительно перейти на новую строку, употребляйте тег
. [тег <pre>, 4.6.5] [тег
, 4.6.1]

4.1.2.2. Атрибут align

Большинство броузеров автоматически выравнивают текст по левому краю. Чтобы изменить такое положение дел, можно использовать пре-

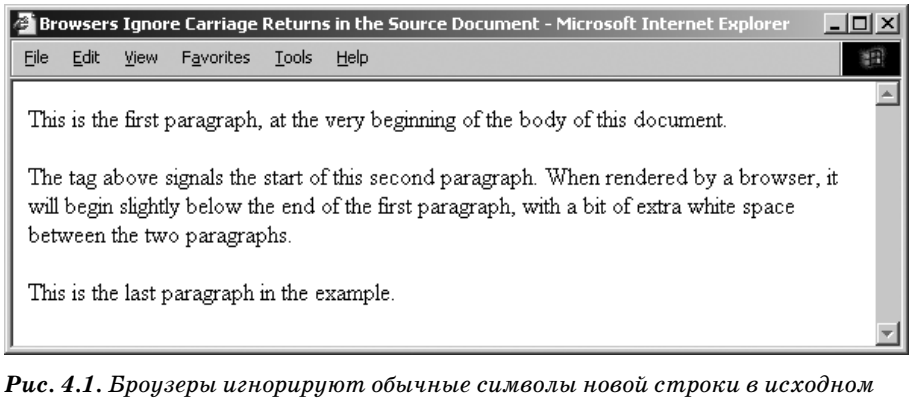

Рис. 4.1. *Броузеры игнорируют обычные символы новой строки в исходном HTML/XHTML-документе*

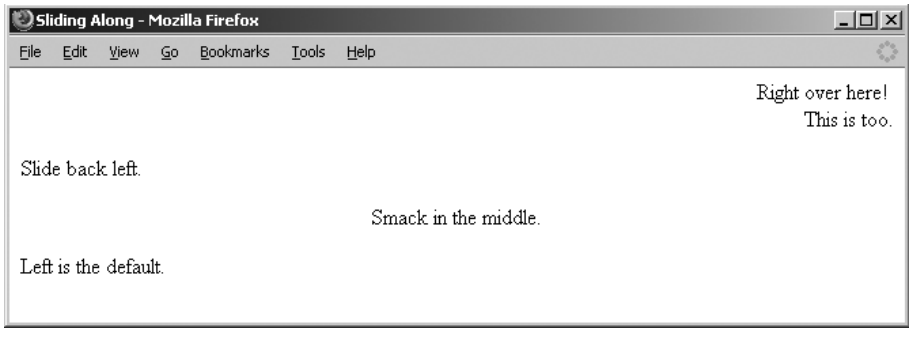

Рис. 4.2. *Действие атрибута align на выравнивание абзаца*

доставляемый HTML 4 и XHTML атрибут `align` тега `<p>` и указать в качестве его значения один из четырех видов выравнивания – `left`, `right`, `center` или `justify`.

Посмотрите (рис. 4.2), как отображается в окне броузера текст, выровненный по-разному:

```
<p align=right>
Right over here!
<br>
This is too.
<p align=left>
Slide back left.
<p align=center>
Smack in the middle.
</p>
Left is the default.
```

Обратите внимание на то, что выравнивание сохраняется до появления либо следующего тега `<p>`, либо текущего закрывающего тега `</p>`. Мы намеренно опустили последний тег `<p>`, чтобы продемонстрировать действие закрывающего тега `</p>` на выравнивание абзаца. Другие элементы, расположенные в теле документа, также могут прекращать действие выравнивания, принятого для текущего абзаца, и заставлять последующие абзацы вернуться к принятому по умолчанию выравниванию по левому краю. К таким элементам относятся формы, заголовки, таблицы и большинство других тегов, применяющихся в теле документа.

Заметим также, что атрибут `align` является нежелательным в HTML 4 и XHTML. Поэтому лучше задавать выравнивание с помощью таблиц стилей.

4.1.2.3. Атрибуты dir и lang

Атрибут `dir` советует броузеру, в каком направлении следует выводить текст абзаца, а атрибут `lang` позволяет определить язык, который будет употребляться в абзаце. Атрибуты `dir` и `lang` поддерживаются по-

пулярными броузерами, хотя для конкретных языков их поведение никак не определено. [атрибут dir, 3.6.1.1] [атрибут lang, 3.6.1.2]

4.1.2.4. Атрибуты class, id, style и title

Применяйте атрибут `id` для обеспечения абзацев документа метками, позволяющими ссылаться на них в гиперссылках, таблицах стилей, апплетах и других автоматических процессах. [атрибут id, 4.1.1.4]

Используйте необязательный атрибут `title` и заключенную в кавычки строку – значение атрибута – для привязывания к абзацу описывающей его фразы. [атрибут title, 4.1.1.5]

Применяйте атрибут `style` с тегом `<p>` для задания стиля отображения, встроенного в тег. Атрибут `class` позволяет отнести абзац к заранее определенному классу тегов `<p>`, заданному в таблице стилей, находящейся либо на уровне документа, либо вне его. В дополнение к этому абзацы, для которых установлена принадлежность классу, хорошо подходят для компьютерной обработки документов, такой как извлечение всех абзацев, принадлежащих классу «citation», при автоматической подборке полного списка цитат. [встроенные стили: атрибут style, 8.1.1] [стилевые классы, 8.3]

4.1.2.5. Атрибуты событий

Подобно ситуации с разделами, многие действия пользователя, такие как одинарный или двойной щелчок мышью в области отображения абзаца, распознаются броузером, если он отвечает текущим стандартам HTML или XHTML. С помощью соответствующих on-атрибутов и их значений можно реагировать на эти события, отображая окно диалога с пользователем или активизируя какое-нибудь мультимедийное событие. [обработчики событий JavaScript, 12.3.3]

4.1.2.6. Допустимое содержимое абзаца

Абзац может содержать любой элемент, допустимый в потоке текста, включая слова и знаки препинания, гиперссылки (`<a>`), изображения (`<img>`), теги новой строки (`<br>`), модификаторы шрифта (`<b>`, `<i>`, `<tt>`, `<strike>`, `<big>`, `<small>`, `<sup>`, `<sub>` и `<font>`), а также компоненты логической разметки[1] (`<acronym>`, `<cite>`, `<code>`, `<dfn>`, `<em>`, `<kbd>`, `<samp>`, `<strong>` и `<var>`). Если в абзаце появляется элемент, отличный от перечисленных, это влечет за собой окончание абзаца, и броузер полагает, что тег `</p>` был опущен.

4.1.2.7. Допустимое использование абзаца

Вы можете определить абзац только внутри *блока* наряду с другими абзацами, списками, формами и форматированным текстом. В целом

[1] Иногда применяется термин «структурная разметка». – *Примеч. науч. ред.*

это означает, что абзацы могут появляться там, где уместен поток текста – в теле документа, в элементе списка и т. д. Формально абзацы не могут встречаться в заголовках, якорях и других элементах, содержимое которых строго определено как чисто текстовое. На практике большинство броузеров пренебрегают этим ограничением и допускают форматирование содержимого элементов в виде абзаца.

4.2. Заголовки

Пользователи тратят немало времени на чтение с экрана. Длинный поток текста, не прерываемый заголовками, подзаголовками и другими названиями фрагментов, скользит перед глазами, затуманивая голову, не говоря уже о невозможности при беглом просмотре обнаружить интересующую тему.

Сплошной текст обязательно нужно разбивать на несколько небольших разделов, снабжая их заголовками (как это делается в книге, которую вы читаете). Существует шесть уровней HTML/XHTML-заголовков, которыми можно пользоваться для превращения сплошного потока текста в легко читаемый и удобный в поддержке документ. Как говорится в главе 5 и в главе 8, есть множество графических и основанных на стиле отображения текста хитростей, помогающих разбить документ на части и сделать его более понятным и удобным для чтения.

4.2.1. Теги заголовков

Шесть заголовочных тегов, записываемых `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>` и `<h6>`, представляют иерархию заголовков от высшего (`<h1>`) до нижнего (`<h6>`) уровня, которая может встречаться в вашем документе.

Текст, заключенный внутри заголовка, обычно отображается специальным образом в зависимости от технологии, доступной броузеру. Броузер может обозначить заголовок, поместив его в центре, выделив полужирным шрифтом и/или шрифтом большего размера, курсивом, подчеркнув его или отобразив отличным от основного цветом. А кроме того, пользователи в пику самым нудным авторам часто меняют способ, которым воспроизводятся различные заголовки.

<h1>, <h2>, <h3>, <h4>, <h5>, <h6>

Функция:	Определяют заголовок одного из шести уровней
Атрибуты:	`align`, `class`, `dir`, `id`, `lang`, `onClick`, `onDblClick`, `onKeyDown`, `onKeyPress`, `onKeyUp`, `onMouseDown`, `onMouseMove`, `onMouseOut`, `onMouseOver`, `onMouseUp`, `style`, `title`
Закрывающий тег:	`</h1>`, `</h2>`, `</h3>`, `</h4>`, `</h5>`, `</h6>`; присутствует обязательно
Содержит:	*текст*
Может содержаться в:	*body_content* (содержимое тела)

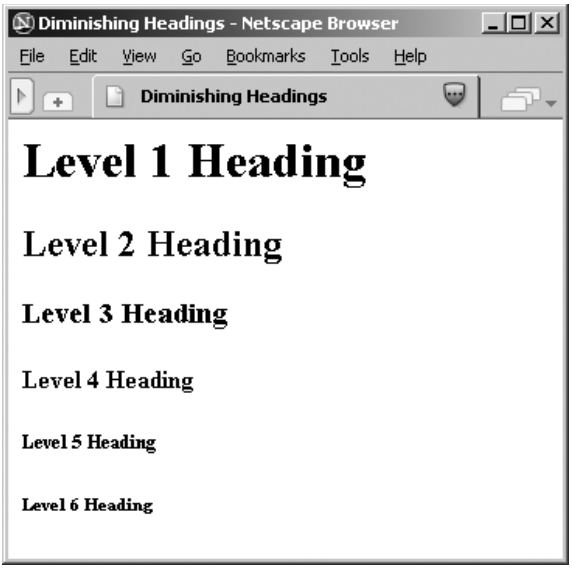

Рис. 4.3. *Броузеры обычно используют уменьшающиеся размеры шрифта для отображения иерархии заголовков*

Традиционно авторы применяют заголовок `<h1>` для названия документа, `<h2>` – для названий разделов и т. д., часто так же, как мы привыкли оформлять работы с использованием заголовков, подзаголовков и заголовков более низких уровней.

Наконец, не забывайте включать соответствующие закрывающие теги. Броузер не вставит его за вас автоматически, и пропуск закрывающего тега заголовка может повлечь за собой катастрофические последствия для вашего документа.

К счастью, на практике большинство броузеров применяют для отображения иерархии заголовков последовательно уменьшающийся размер шрифта, так что текст в `<h1>` будет очень крупным, а текст в `<h6>` – очень мелким (рис. 4.3).

4.2.1.1. Атрибут align

Выравнивание по умолчанию для большинства броузеров соответствует значению `left` атрибута `align`. Так же, как с тегами `<div>` и `<p>`, вы можете определять выравнивание, присваивая атрибуту `align` одно из следующих значений: `left`, `center`, `right`, `justify`. Посмотрите (рис. 4.4), как отображаются выравнивания, заданные в следующем исходном тексте:

```
<h1 align=right>Right over here!</h1>
<h2 align=left>Slide back left.</h2>
<h3 align=center>Smack in the middle.</h3>
```

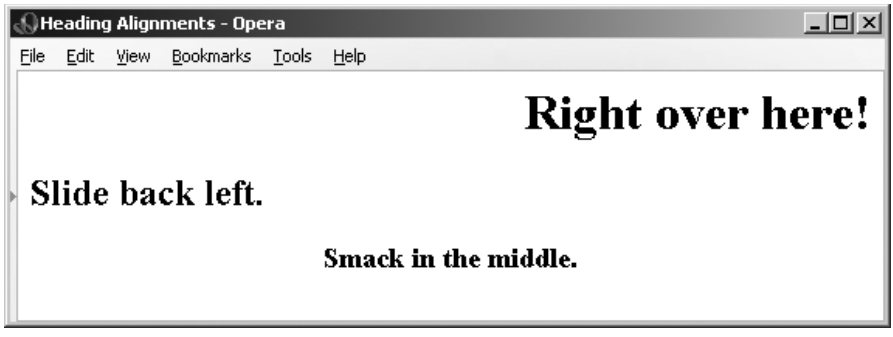

Рис. 4.4. *Выравнивание заголовков в действии*

Значение `justify` атрибута `align` пока не поддерживается ни одним броузером и, видимо, уже не будет. Атрибут `align` является нежелательным в HTML 4 и XHTML. Поэтому лучше задавать выравнивание при помощи таблиц стилей.

4.2.1.2. Атрибуты dir и lang

Атрибут `dir` советует броузеру, в каком направлении следует выводить текст заголовка, а атрибут `lang` позволяет определить язык, который будет употребляться в заголовке. [атрибут dir, 3.6.1.1] [атрибут lang, 3.6.1.2]

4.2.1.3. Атрибуты class, id, style и title

Применяйте атрибут `id` для обеспечения заголовков вашего документа метками, позволяющими ссылаться на них в гиперссылках, таблицах стилей и из внешних программ автоматизированного поиска и обработки. [атрибут id, 4.1.1.4]

Используйте необязательный атрибут `title` и заключенную в кавычки строку – значение атрибута – для привязывания к заголовку описывающей его фразы. [атрибут title, 4.1.1.5]

Применяйте атрибут `style` с тегом заголовка для встроенного в тег определения стиля содержимого. Атрибут `class` позволяет присвоить заголовку имя, относящее его к заранее определенному классу, заданному в таблице стилей, находящейся либо на уровне документа, либо вне его. [встроенные стили: атрибут style, 8.1.1] [стилевые классы, 8.3]

4.2.1.4. Атрибуты событий

Любые относящиеся к поведению пользователя события как внутри заголовка, так и вне его распознаются броузером, если он отвечает текущим стандартам HTML или XHTML. С помощью соответствующих on-атрибутов и их значений вы можете реагировать на эти события, отображая окно диалога с пользователем или активизируя какое-нибудь мультимедийное событие. [обработчики событий JavaScript, 12.3.3]

4.2.2. Как использовать заголовки

Разумно повторить название вашего документа в первом теге заголовка, поскольку оно не отображается в основном окне броузера, если было определено в теге <head>.

Следующий HTML-текст может служить хорошим примером повторения названия вашего документа в его заголовке и в теле:

```
<html>
<head>
<title>Выращивание кумкватов в Северной Америке</title>
</head>
<body>
<h3> Выращивание кумкватов в Северной Америке</h3>
<p>
Возможно, одним из самых привлекательных фруктов является...
```

В типичном случае броузер помещает текст <title> в самом верху выведенной страницы. Тег может использоваться для создания закладок и включения в списки любимых страниц, отображаясь только где-то на краю окна броузера, заголовок третьего уровня в этом примере всегда будет появляться в самом начале документа. Он будет служить видимым названием документа, как бы броузер ни обрабатывал содержимое тега <title>. И, в отличие от текста в теге <title>, название в заголовке окажется в начале первой страницы документа при его распечатке, потому что является частью основного текста. [тег <title>, 3.7.2]

В нашем примере мы решили использовать заголовок третьего уровня <h3>, который отображается обычно чуть более крупным шрифтом, чем основной текст. Заголовки первого и второго уровня еще крупнее и смотрятся непропорционально. Следует выбрать уровни заголовков, представляющиеся вам уместными и привлекательными, и систематически применять их в своих документах. Слишком большие заголовки разрушат композицию окна; слишком мелкие визуально сольются с текстом.

Установив заголовок высшего уровня в документе, применяйте для добавочных заголовков теги того же или более низкого уровня, обеспечивая структурированность документа и его читабельность. Если вы выбрали третий уровень заголовка для названия документа, разбейте текст на части с применением заголовков четвертого уровня. Если есть необходимость в дальнейшем разбиении текста на подразделы, подумайте о том, не использовать ли для названия документа заголовок второго уровня, заголовок третьего уровня – для выделения разделов, а четвертого – для подразделов.

4.2.3. Использование заголовков нижних уровней

Большинство графических броузеров для представления заголовков <h1>, <h2> и <h3> применяют более крупный шрифт, для заголовка

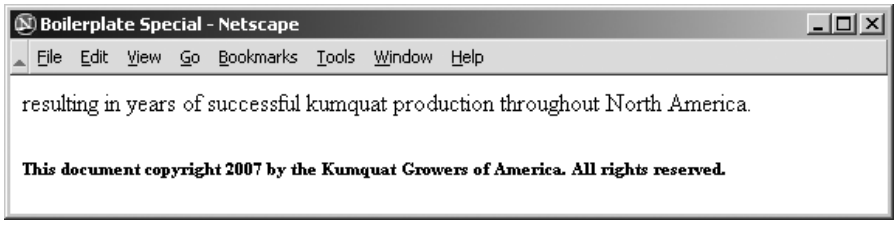

Рис. 4.5. *HTML/XHTML-авторы обычно используют заголовки шестого уровня для вспомогательного текста*

<h4> – такой же, как для отображения основного текста, и для заголовков <h5> и <h6> – более мелкий. Авторы обычно используют последние два размера для вывода стандартного текста, такого как уведомление об авторских правах или отказ от ответственности. Хотя рекомендуется применять стилевые правила, некоторые авторы употребляют мелкий текст заголовков в оглавлении и в навигационной панели, расположенной на главной странице сайта. Посмотрите, как типичный броузер выводит информацию о защите авторских прав в следующем XHTML-примере (рис. 4.5):

```
resulting in years of successful kumquat production
throughout North America.
</p>
<h6>This document copyright 2007 by the Kumquat Growers of
America. All rights reserved. </h6>
</body>
</html>
```

4.2.4. Допустимое содержимое заголовков

Заголовок может содержать любой элемент, допустимый в потоке текста, включая слова и знаки препинания, гиперссылки (<a>), изображения (<img>), теги новой строки (
), украшения шрифта (<b>, <i>, <tt>, <strike>, <big>, <small>, <sup>, <sub> и <font>), а также компоненты логической разметки (<acronym>, <cite>, <code>, <dfn>, , <kbd>, <samp>, и <var>). На практике, однако, изменения шрифта и стиля в заголовке могут не оказывать действия, поскольку сам заголовок предписывает броузеру изменение шрифта.

Раньше было принято ругать использование тегов заголовков для управления размером шрифта во всем документе. Формально абзацы, списки и другие «блочные» элементы не допускаются в заголовках и могут быть ошибочно истолкованы броузером как обозначения их конца. На практике большинство броузеров применяют стиль заголовка ко всем содержащимся в нем абзацам. Мы не одобряем такую практику, поскольку она не только нарушает стандарты HTML и XHTML, но как правило, безобразно выглядит. Представьте себе газету, напечатанную одними заголовками!

Описание больших фрагментов текста в качестве заголовков противоречит назначению этого тега. Если вам действительно нужно изменить вид шрифта или его размер в документе, пользуйтесь стандартными каскадными определениями стилей. См. главу 8.

Кроме того, мы настоятельно рекомендуем тщательно тестировать свои страницы более чем на одном броузере и при нескольких различных разрешениях. Можно ожидать, что ваш <h6>-текст будет читаться при разрешении 480×320, но пропадет при разрешении 800×600.

4.2.5. Допустимое использование заголовков

Формально стандарты HTML и XHTML допускают вхождение заголовков исключительно в *содержимое тела* документа. На практике большинство броузеров признают заголовки почти везде, форматируя текст при выводе в соответствии с установками текущего элемента. Во всех случаях появление заголовка означает конец абзаца или другого текстового элемента, так что вы не можете использовать теги заголовков, чтобы изменить высоту нескольких символов в строке. Используйте каскадные определения стилей для достижения таких локальных эффектов. [встроенные стили: атрибут style, 8.1.1]

4.2.6. Вставка изображений в заголовки

В заголовок можно вставить одно или несколько изображений – от маленьких маркеров или кнопок до полноразмерных картинок. Комбинирование продуманной системы заголовков с соответствующими пиктограммами не только внешне привлекательно, но и может быть эффективным средством помощи пользователям в их работе с коллекцией документов. [тег <img>, 5.2.6]

Вставить изображение в заголовок легко. К примеру, следующий текст помещает пиктограмму «информация» в заголовок «For more information» («Дальнейшие справки»), как видно на рис. 4.6:

```
<h2>
<img src="info.gif">
For More Information</h2>
```

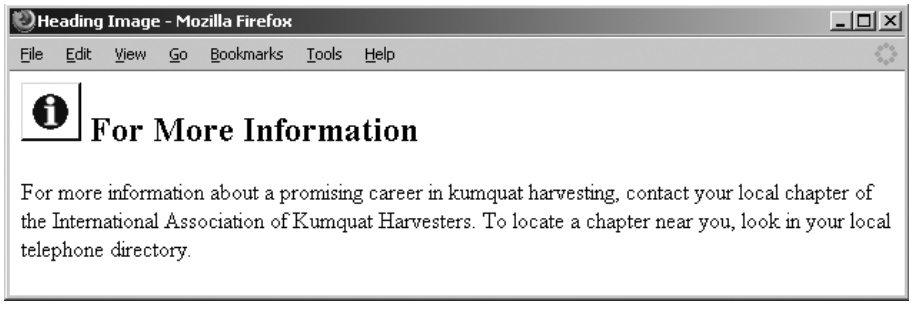

Рис. 4.6. *Изображение, вставленное в заголовок*

Вообще говоря, изображения в заголовках выглядят лучше всего, когда их помещают в начало строки, выравнивая по нижнему краю текста или посередине.

4.3. Управление внешним видом текста

Ряд тегов изменяют внешний вид текста и придают ему некий скрытый смысл. В целом они могут быть разделены на две группы – теги физической и логической разметки.

Кроме того, стандарт каскадных таблиц стилей, созданный W3C, теперь хорошо поддерживается популярными броузерами. Он дает авторам другой, более мощный способ управления макетом и внешним видом документов. В этой главе будет рассмотрено управление стилями с помощью тегов. Подробное описание каскадных таблиц стилей можно найти в главе 8.

4.3.1. Логическая разметка

Теги логических стилей (логическая разметка) информируют броузер о том, что заключенный в них текст имеет специальное значение, контекст или употребление. Броузер вследствие этого форматирует текст в соответствии с этими значением, контекстом или употреблением. Обратите внимание на важное отличие. Теги логических стилей передают смысл, а не форматирование. Следовательно, они важны для автоматизированных процессов. Программам все равно, как выглядит документ, – по крайней мере, пока.

Так как разметка в данном случае определяется на основе семантики, броузер может выбрать стиль отображения, приемлемый для пользователя. Поскольку у разных клиентов такие стили отличаются, использование логической разметки увеличивает вероятность того, что документ будет выглядеть осмысленно оформленным в глазах более широкой аудитории. Это особенно важно, если броузер ориентирован на слепых или ограниченных в других отношениях читателей, чей выбор способа отображения может радикально отличаться от стандартного текста или быть крайне ограниченным.

Действующие стандарты HTML и XHTML не определяют формат для каждого из тегов логической разметки, требуя только, чтобы их отображение отличалось по виду от основного текста в документе. Стандарты не настаивают даже на том, чтобы логические стили были различно отражены на экране. На практике вы обнаружите, что многие из этих тегов связаны очевидным родством с типографскими обозначениями, обладая схожими значениями и обликом и отображаясь одними и теми же стилями и шрифтами большинством броузеров.

4.3.2. Физическая разметка

Говоря о логической разметке, мы часто употребляем слово «смысл». Это связано с тем, что смысловое значение, которое передает тег, важнее способа, каким броузер отображает заключенное в нем содержимое. Бывают, однако, ситуации, когда хочется, чтобы текст выглядел неким определенным образом, скажем был выведен курсивом или полужирным шрифтом, по каким-либо юридическим или другим соображениям. В этих случаях используйте теги физических стилей.

Тогда как другие системы обработки текста склонны к явному управлению стилями и внешним видом, при использовании HTML и XHTML вам следует избегать явных, физических тегов, применяя их лишь в редких случаях. Снабжайте (если это возможно) броузер информацией о контексте. Применяйте логическую разметку. Если даже современные броузеры не делают с документами ничего лучшего, чем представление их содержимого курсивом или полужирным шрифтом, то броузеры будущего и программы, автоматически создающие документы, смогут использовать структурную разметку более творчески.

4.4. Теги логической разметки

Работа с логической разметкой требует дисциплины, так как гораздо легче думать только о внешнем виде документа, а не о том, что он при этом будет значить. Как только вы начинаете использовать логическую разметку, документы становятся более последовательными и подходящими для автоматического поиска и извлечения информации.

Теги логической разметки

Функция:	Изменяют внешний вид текста на основании его смысла, контекста или употребления
Атрибуты:	`class`, `dir`, `id`, `lang`, `onClick`, `onDblClick`, `onKeyDown`, `onKeyPress`, `onKeyUp`, `onMouseDown`, `onMouseMove`, `onMouseOut`, `onMouseOver`, `onMouseUp`, `style`, `title`
Закрывающие теги:	Присутствуют обязательно
Содержит:	*текст*
Могут содержаться в:	*тексте*

4.4.1. Тег <abbr>

Впервые появившись в HTML 4.0, тег `<abbr>` («аббревиатура») указывает, что заключенный в нем текст является сокращенной формой более длинного слова или фразы. Броузер может использовать эту информацию для выбора способа отображения заключенного в нем содержимого. Заметьте, мы сказали: «может». Не все популярные бро-

узеры реагируют сейчас на текст, заключенный в тег <abbr>, и мы не можем предсказать, как они будут его обрабатывать в будущем.

4.4.2. Тег <acronym>

Тег <acronym> означает, что заключенный в нем текст является акронимом – сокращением, составленным из первых букв каждого слова какого-то названия или фразы, например HTML или IBM. Как и в случае с тегом <abbr>, не все броузеры соответствующим образом отобразят текст в теге логической разметки <acronym>.

4.4.3. Тег <cite>

Тег <cite>, как правило, означает, что заключенный в нем текст является библиографической ссылкой, в частности названием книги или журнала. Традиционно текст ссылки отображается курсивом. Вот как Internet Explorer показывает следующий исходный текст (рис. 4.7):

```
While kumquats are not mentioned in Melville's
<cite>Moby Dick</cite>, it is nonetheless apparent
that the mighty cetacean represents the bitter
"kumquat-ness" within every man. Indeed, when Ahab
spears the beast, its flesh is tough, much like the noble fruit.
```

Используйте тег <cite> для выделения ссылок на другие документы, особенно представленные в традиционной форме, например книги, журналы, журнальные статьи. Если существует доступная в сети версия работы, на которую вы ссылаетесь, то следует также заключить ссылку в тег <a> и сделать ее гиперссылкой на онлайновую версию.

Кроме того, тег <cite> обладает скрытой возможностью: он позволяет автоматически выделять библиографию из документов. Легко вообразить броузер, который автоматически создает таблицу ссылок, отображая их в сносках или в виде отдельного материала. Семантика тега <cite> идет гораздо дальше простого изменения внешнего вида охватываемого текста, она позволяет броузеру представить содержимое тега множеством полезных способов.

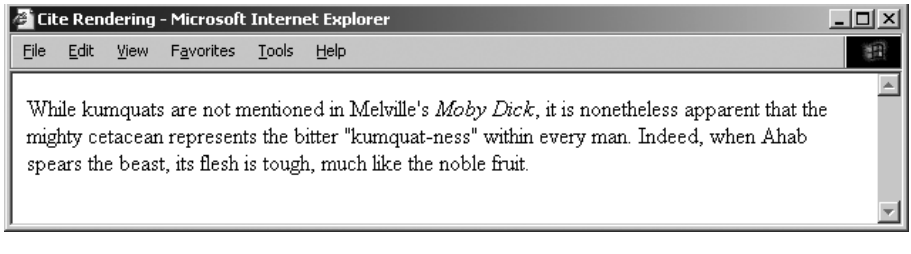

Рис. 4.7. *Internet Explorer выводит <cite> курсивом*

4.4.4. Тег <code>

Программисты привыкли к особому стилю представления своих исходных кодов. Тег <code> создан для них. Он возвращает заключенный в нем текст моноширинным шрифтом (как в пишущей машинке или телетайпе), таким как Courier, знакомым большинству программистов и читателей книжных серий издательства O'Reilly, включающих и эту книгу.

Следующий отрывок текста в теге <code> отображается моноширинным шрифтом в результате работы Firefox, как показано на рис. 4.8 (эффект, скажем прямо, не потрясает):

```
The array reference <code>a[i]</code> is identical to
the pointer reference <code>*(a+i)</code>.
```

Тег <code> следует применять только для текста, который представляет исходный код программы или другое содержимое, предназначенное для машинного чтения. Обычно, когда тег <code> просто делает текст моноширинным, подразумевается, что это исходный код программы, и будущие броузеры сумеют добавить новые эффекты при его отображении. К примеру, броузер программиста может искать сегменты в тегах <code> и производить какое-то дополнительное форматирование текста, типа специального выделения отступами циклов и условных предложений. Если единственный эффект, которого вы добиваетесь, состоит в выводе текста моноширинным шрифтом, используйте тег <tt>. Если же вы хотите представить код программы в виде четко отформатированного моноширинного текста, пользуйтесь тегом <pre>. [тег <tt>, 4.5.10] [тег <pre>, 4.6.5]

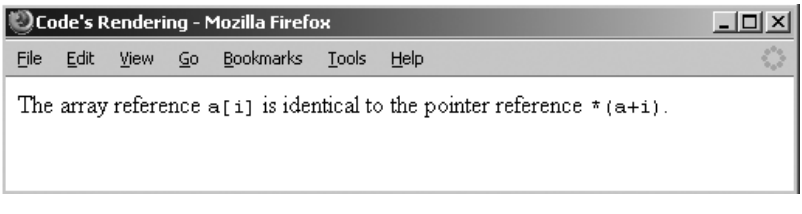

***Рис. 4.8.** Используйте <code> для представления «исходников» программ*

4.4.5. Тег <dfn>

Употребляйте тег <dfn> для выделения впервые появившихся в тексте специальных терминов. Популярные броузеры обычно выводят текст тега <dfn> курсивом. Впоследствии <dfn> поможет вам при составлении индекса или словаря.

К примеру, используйте тег <dfn>, чтобы познакомить читателя с новым термином:

```
Согласно результатам многолетних наблюдений, <dfn>спектроскопия кожуры</dfn>
может оказаться весьма полезной. Сравнение с ее помощью относительных уровней
содержания насыщенных гидрокарбонатов в плодах, снятых с соседних деревьев,
выявило, что спектроскопия кожуры оказалась эффективной в 87% случаев
для предупреждения водянки ствола у четырехлетних деревьев.
```

Заметьте, что мы выделили только первое вхождение «спектроскопии кожуры» в тексте примера. Чувство стиля советует нам не засорять текст излишней пестротой. Подобно другим тегам логической и физической разметки, здесь справедливо правило: чем меньше, тем лучше.[1] Как это принято, особенно в технической документации, выделяйте новые термины, когда они впервые вводятся, чтобы помочь читателю лучше понять обсуждаемый вопрос, но воздержитесь от выделения этого термина в дальнейшем.

4.4.6. Тег

Тег сообщает броузеру, что его содержимое должно быть отображено с визуальным выделением. Почти для всех броузеров это означает, что текст будет выведен курсивом. К примеру, популярные броузеры выделят курсивом слова «всегда» и «никогда» в следующем исходном HTML/XHTML-тексте:

```
Те, кто выращивают кумкваты, должны <em>всегда</em> называть их не иначе как
"благородный фрукт", но <em>никогда</em> – просто "фрукт."
```

Расстановка логических ударений в тексте – совсем непростое занятие. Слишком мало – и важная фраза может потеряться. Слишком много – и пропадет эффект, к которому вы стремитесь. Как любую приправу, тег лучше употреблять скупо.

Хотя он и отображается всегда курсивом, тег намекает на многое, и броузеры завтрашнего дня, возможно, научатся воспроизводить его с разными спецэффектами. Тег <i> прямо требует выделить текст курсивом; применяйте его, если вам нужен именно курсив. В качестве альтернативы вы можете включить в документ каскадные определения стилей. [тег <i>, 4.5.4]

Кроме постановки логического ударения подумайте о возможности использования при введении нового термина или постоянного стиля, ссылающегося на определенный круг объектов или понятий. К примеру, один из стилей в книгах O'Reilly предназначен для обозначения имен файлов и устройств. Тег можно применять (в отличие от простого курсива) для акцентирования внимания на терминах.

---

[1] Если вам нужны доказательства того, что «меньше» значит «лучше» (это касается применения логической и физической разметки), попробуйте почитать школьный учебник, в котором кто-то желтым карандашом подчеркнул слова и фразы, показавшиеся ему важными.

4.4.7. Тег <kbd>

Если говорить о специальных стилях, предназначенных для технических понятий, то следует упомянуть тег <kbd>. Как вы уже, наверное, догадались, он используется для обозначения текста, который набирается на клавиатуре. Заключенный в этом теге текст обычно отображается моноширинным шрифтом.

Тег <kbd> чаще всего используется в компьютерной документации и руководствах так, как в следующем примере:

```
Введите <kbd>quit</kbd> для завершения программы или введите<kbd>menu</kbd>
для возвращения к главному меню.
```

4.4.8. Тег <samp>

Тег <samp> означает последовательность символов, которая не должна иметь для пользователя никакого другого смысла, кроме буквального. Этот тег чаще всего применяется, когда последовательность символов вырвана из обычного контекста. К примеру, следующий исходный текст:

```
The <samp>ae</samp> character sequence may be converted
to the &aelig; ligature if desired.
```

выводится Netscape (рис. 4.9).

***Рис. 4.9.** Выделение образца с помощью тега <samp>*

Принятое в HTML указание на лигатуру «ae» – это æ, и большинством броузеров она правильно отображается символом "?". За дополнительной информацией обратитесь к приложению F.

Тег <samp> не очень часто используется. Его следует употреблять в тех немногих случаях, когда нужно особо выделить короткую последовательность символов, выхваченную из обычного контекста.

4.4.9. Тег

Подобно тегу , тег используется для более выраженного логического подчеркивания текста. Броузеры обычно отображают теги и различным образом, выводя, как правило, текст в полужирным шрифтом (в отличие от курсива), так что пользователи могут их различать. К примеру, в следующем тексте под-

черкнутое с помощью «never» выводится в броузере Opera курсивом, тогда как выделенное тегом слово «forbidden» отображается жирными буквами (рис. 4.10):

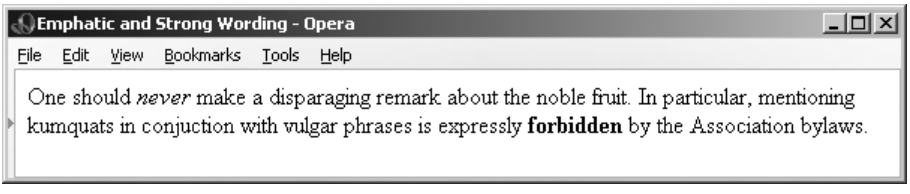

***Рис. 4.10.** «Сильный» и подчеркнутый текст отображаются по-разному*

```
One should <em>never</em> make a disparaging remark
about the noble fruit. In particular, mentioning
kumquats in conjunction with vulgar phrases is
expressly <strong>forbidden</strong> by
the Association bylaws.
```

Если здравый смысл подсказывает, что тег должен употребляться скупо, то тег должен появляться в тексте еще реже. Текст в похож на возглас. Текст в – это вопль. Подобно тому как ярко звучит хорошо подобранный эпитет в устах обычно молчаливого человека, употребленный по особенному случаю, будет замечен и оценен.

4.4.10. Тег <var>

Тег <var> – это тоже штучка из компьютерной документации. Он обозначает или имя переменной, или данные, вводимые пользователем. Этот тег часто используется в соединении с тегами <pre> и <code> для отображения отдельных частей программного кода и тому подобного. Текст в теге <var> обычно выводится курсивом, как показано на рис. 4.11, который представляет вывод для следующего примера:

```
The user should type
<pre>
  cp <var>source-file</var>  <var>dest-file</var>
```

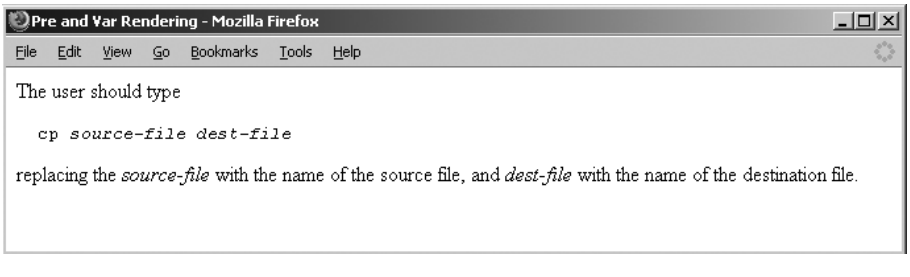

***Рис. 4.11.** Тег <var> обычно появляется в преформатированном (<pre>) компьютерном коде*

```
</pre>
replacing the <var>source-file</var> with the name of
the source file, and <var>dest-file</var> with the name
of the destination file.
```

Подобно другим тегам, связанным с программированием и компьютерной документацией, тег `<var>` не только облегчает пользователям просмотр и усвоение документа, но позволяет автоматизированным системам употреблять подходящим образом размеченный текст для извлечения из него информации. Скажем еще раз: чем больше разнообразной семантической информации вы сообщите броузеру, тем лучше он сможет представить ее пользователю.

4.4.11. Атрибуты class, id, style и title

Хотя каждый тег логической разметки имеет свой стиль при выводе, вы можете отменить эти установки и присвоить тегам новый образ. Этот новый внешний вид может быть применен к тегам логических стилей при помощи атрибута `style` или атрибута `class`. [встроенные стили: атрибут style, 8.1.1] [стилевые классы, 8.3]

Вы можете также присвоить каждому тегу логической разметки свой уникальный идентификатор, задаваемый атрибутом `id`, как и менее строгое название (`title`), используя соответствующие атрибуты и их значения в виде заключенных в кавычки строк. [атрибут id, 4.1.1.4] [атрибут title, 4.1.1.5]

4.4.12. Атрибуты dir и lang

Атрибут `dir` советует броузеру, в каком направлении следует выводить текст в теге логической разметки, а атрибут `lang` позволяет определить язык, который употребляется в теге. [атрибут dir, 3.6.1.1] [атрибут lang, 3.6.1.2]

4.4.13. Атрибуты событий

Как внутри тега логической разметки, так и вне его происходят события, и с помощью соответствующих on-атрибутов и их значений вы можете реагировать на эти события, отображая окно диалога с пользователем или активизируя какое-нибудь мультимедийное событие. [обработчики событий JavaScript, 12.3.3]

4.4.14. Сводка тегов логических стилей

Различные графические броузеры выводят содержимое тегов логических стилей похожим образом. Чисто текстовые броузеры, такие как Lynx, отображают все теги одинаково. В табл. 4.1 представлены физические стили, используемые различными броузерами для отображения тегов логической разметки.

Таблица 4.1. Теги логических стилей

Тег	Netscape Navigator	Internet Explorer	Mozilla Firefox	Opera	Lynx
<abbr>	– (не поддерживается)	–	–	–	–
<acronym>	–	–	–	–	–
<cite>	*курсив*	*курсив*	*курсив*	*курсив*	моноширинный
<code>	моноширинный	моноширинный	моноширинный	моноширинный	моноширинный
<dfn>	*курсив*	*курсив*	*курсив*	*курсив*	–
	курсив	*курсив*	*курсив*	*курсив*	моноширинный
<kbd>	моноширинный	моноширинный	моноширинный	моноширинный	моноширинный
<samp>	моноширинный	моноширинный	моноширинный	моноширинный	моноширинный
	полужирный	**полужирный**	**полужирный**	**полужирный**	моноширинный
<var>	*курсив*	*курсив*	*курсив*	*курсив*	моноширинный

4.4.15. Допустимое содержимое

Любой тег логической разметки может содержать любой элемент, допустимый в *тексте*, включая обычный текст, якоря, изображения и перевод на новую строку. Кроме того, в их содержимое могут вкладываться другие теги логической и физической разметки.

4.4.16. Допустимое употребление

Любой тег логической разметки может быть использован в любом месте, где применяются элементы, относящиеся к *тексту*. На практике это означает, что вы вправе употреблять , <code> и им подобные теги везде, за исключением сегментов, выделенных тегами <title>, <listing>, <xmp>. Применение символических тегов в заголовках допустимо, но их влияние может быть перекрыто действием самого заголовка.

4.4.17. Комбинирование логических стилей

Иногда бывает, что вы комбинируете два или большее число логических стилей, чтобы создать интересный и, кажется, даже полезный гибрид. Так, например, логически подчеркнутая ссылка может быть получена следующим образом:

```
<cite><em>Moby Dick</em></cite>
```

На практике, многоуважаемый доктор Франкенштейн, броузер обычно проигнорирует вашего монстра. Как нетрудно убедиться, набрав и протестировав приведенный пример, Моби Дик будет цитатой без всякого подчеркивания.

Стандарты HTML и XHTML не требуют от броузера поддержки всех возможных комбинаций логических стилей и не определяют, как он должен обращаться с такими комбинациями. Может быть, когда-нибудь... Пока же лучше выбрать один тег.

4.5. Теги физической разметки

Есть девять физических стилей текста, предлагаемых действующими стандартами HTML и XHTML. Это полужирный, курсив, моноширинный, подчеркнутый, зачеркнутый, увеличенный, уменьшенный, верхний и нижний индексы.

Теги физической разметки

Функция:	Определяют физические стили текста
Атрибуты:	`class`, `dir`, `id`, `lang`, `onClick`, `onDblClick`, `onKeyDown`, `onKeyPress`, `onKeyUp`, `onMouseDown`, `onMouseMove`, `onMouseOut`, `onMouseOver`, `onMouseUp`, `style`, `title`
Закрывающие теги:	Присутствуют обязательно
Содержит:	*текст*
Могут содержаться в:	*тексте*

К нашему облегчению, Internet Explorer перестал поддерживать десятый физический стиль, «мерцающий» текст. Надеемся, остальные последуют его примеру. Все теги физической разметки требуют закрывающего тега.

При детальном обсуждении каждого тега физической разметки не упускайте из виду, что они несут в себе определения стиля только для содержащегося в них текста. Для управления отображением текста в масштабах документа используйте таблицы стилей (глава 8).

4.5.1. Тег <b>

Тег `<b>` – это физический эквивалент тега логической разметки `<strong>`, но без дополнительных смыслов, связанных с последним. Он явным образом требует отображения полужирным шрифтом символа или текста, заключенного между ним и его закрывающим тегом `</b>`. Если полужирный шрифт оказывается недоступным, броузер может использовать другое его представление, такое как подчеркивание или инверсная печать «белым по черному».

4.5.2. Тег <big>

Тег `<big>` позволяет легко увеличить размер текста. Нет ничего проще: броузер возвращает фрагмент, заключенный между тегом `<big>` и соответствующим ему закрывающим тегом `</big>`, используя шрифт, раз-

мер которого имеет номер на единицу больше, чем у окружающего текста. На текст наибольшего размера тег <big> действия не оказывает. [<font>, 4.10.3]

Более того, можно вкладывать теги <big>, чтобы еще сильнее увеличить текст. Каждый <big> делает шрифт крупнее на один размер вплоть до седьмого, как это определяется моделью шрифта.

4.5.3. Тег <blink> (нежелателен)

Текст, содержащийся между тегом <blink> и его закрывающим тегом </blink>, мерцает. Firefox, например, просто раз за разом меняет местами цвет шрифта и цвет фона для заключенного в теге текста. Ни HTML, ни XHTML не включают в себя <blink>. Первоначально он поддерживался в качестве расширения только в версиях Netscape Navigator до шестой. В версии 6 от него отказались, но потом он был восстановлен в версии 7 и последующих. Opera and Firefox тоже поддерживают его, и только Internet Explorer игнорирует. Игнорируйте и вы.

На бумажных страницах книги мы не можем показать этот анимационный эффект, который, впрочем, легко вообразить и лучше никогда не реализовывать. Мерцающий текст сначала привлекает внимание читателя, но затем быстро надоедает, не обещая ничего, кроме бесконечного раздражения. Забудьте о нем.

4.5.4. Тег

Тег похож на тег логической разметки . Он и его необходимый закрывающий тег предлагают броузеру отобразить заключенный между ними текст курсивом или наклонным шрифтом. Если этот шрифт недоступен, может быть применено выделение яркостью, обращение цвета или подчеркивание.

4.5.5. Тег <s> (нежелателен)

Тег <s> – это сокращенная форма тега <strike>, поддерживаемая всеми современными броузерами, несмотря даже на то, что она объявлена устаревшей в HTML 4 и XHTML. Не пользуйтесь тегом <s>. Рано или поздно он уйдет.

4.5.6. Тег <small>

Тег <small> действует точно так же, как противоположный ему тег <big> (см. [тег <big>, 4.5.2]), за исключением того, что он уменьшает текст, а не увеличивает его. Если окружающий текст уже имеет наименьший допускаемый моделью шрифта размер, тег <small> не оказывает никакого действия.

Как и в случае тега <big>, можно применять несколько вложенных тегов <small> для кратного уменьшения текста. Каждый следующий тег

<small> уменьшает размер текста вплоть до достижения предельно малого размера.

4.5.7. Тег <strike> (нежелателен)

Большинство броузеров выводят текст, заключенный между тегом <strike> и его закрывающим тегом </strike>, с добавлением горизонтальной линии, проходящей сквозь буквы (зачеркивают текст). Подразумевается, что это редакционная пометка, предлагающая читателю игнорировать вычеркнутый текст, как напоминание о временах, предшествовавших изобретению коррекционной ленты. Редко где вы встретите этот тег сегодня: он признан нежелательным в HTML 4 и XHTML, как раз на расстоянии одного шага до полного исключения из стандарта.

4.5.8. Тег <sub>

Текст, содержащийся между тегом _{и его закрывающим тегом}, отображается тем же типом и обычно немного меньшим размером шрифта, чем у окружающего потока. При этом он расположен ниже уровня строки на половину высоты текущего символа. Тег <sub> и ему противоположный <sup> применяются в математических выражениях, научных обозначениях и в химических формулах.[1]

4.5.9. Тег <sup>

Теги ^{и его закрывающий} создают верхний индекс из содержащегося между ними текста. Он отображается тем же типом и обычно немного меньшим размером шрифта, чем у окружающего потока. При этом он расположен выше уровня строки на половину высоты текущего символа. Этот тег удобен при добавлении сносок и показателей степени. Комбинируя его с тегом <a>, можно создать красивую сноску, снабженную гиперссылкой:

```
Личинка кумкватового долгоносика<a href="footnotes.html#note74"
><sup><small>74</small></sup></a> - это
```

В этом примере подразумевается, что все ссылки хранятся в файле *footnotes.html*, подходящим образом разграниченные и пронумерованные.

4.5.10. Тег <tt>

Подобно тегу <code> и <kbd>, тег <tt> предлагает броузеру отобразить моноширинным шрифтом текст, заключенный между ним и его обязательным закрывающим тегом </tt>. Для тех броузеров, что уже используют моноширинный шрифт, этот тег может не оказать никакого выделяющего действия на представление текста.

---

[1] Символы нижнего и верхнего индекса соответственно. – *Примеч. науч. ред.*

4.5.11. Тег <u> (нежелателен)

Этот тег говорит броузеру, чтобы тот подчеркнул текст, содержащийся между `<u>` и `</u>`. Подчеркивание состоит в том, что под текстом, включая пробелы между словами и знаки препинания, проводится сплошная черта. Этот тег признан нежелательным в HTML 4 и XHTML, но популярные броузеры поддерживают его. Такой же эффект может быть достигнут с применением таблиц стилей, описанных в главе 8.

4.5.12. Атрибуты dir и lang

Атрибут `dir` советует броузеру, в каком направлении следует выводить текст в теге физического стиля, а атрибут `lang` позволяет определить язык, который будет в нем употребляться. [атрибут dir, 3.6.1.1] [атрибут lang, 3.6.1.2]

4.5.13. Атрибуты class, id, style и title

Хотя каждый тег физической разметки имеет свой стиль при выводе, эти установки можно отменить и присвоить тегам новый образ. Этот новый внешний вид может быть применен к тегам физических стилей при помощи атрибута `style` или атрибута `class`. [встроенные стили: атрибут style, 8.1.1] [стилевые классы, 8.3]

Можно также присвоить каждому тегу физической разметки свой уникальный идентификатор (`id`), как и менее строгое название (`title`), используя соответствующие атрибуты и их значения в виде заключенных в кавычки строк. [атрибут id, 4.1.1.4] [атрибут title, 4.1.1.5]

4.5.14. Атрибуты событий

Как и в случае тегов логической разметки, инициированные пользователем события с мышью и клавиатурой могут происходить как внутри тега физической разметки, так и вокруг него. Большинство подобных событий распознаются броузерами, отвечающими действующим стандартам, и с помощью соответствующих on-атрибутов и их значений вы можете реагировать на эти события, отображая окно диалога с пользователем или активизируя какое-нибудь мультимедийное событие. [обработчики событий JavaScript, 12.3.3]

4.5.15. Сводка тегов физических стилей

Различные графические броузеры выводят содержимое тегов физической разметки похожим образом. В табл. 4.2 приведены стили отображения, которые используются броузерами для соответствующих тегов. Определения в таблицах стилей могут переустановить начальные стили отображения.

Таблица 4.2. Теги физических стилей

Тег	Значение	Стиль отображения
<b>	Содержимое выводится полужирным шрифтом	**bold**
<big>	Увеличенный размер шрифта	bigger text
<blink> (архаизм)	Поочередно меняет цвета фона и текста	blinking text
<i>	Содержимое выводится курсивом	*italic*
<small>	Уменьшенный размер шрифта	smaller text
<s>, <strike> (нежелателен)	Зачеркивает текст	strike
<sub>	Текст нижнего индекса	subscript
<sup>	Текст верхнего индекса	superscript
<tt>	Стиль телетайпа (консоль)	monospaced
<u> (нежелателен)	Подчеркивает текст	underlined

Посмотрите, как отображает Firefox следующий пример исходного HTML-текста (рис. 4.12):

```
Explicitly <b>boldfaced</b>, <i>italicized</i>, or
<tt>teletype-style</tt> text should be used
<big><big>sparingly</big></big>.
Otherwise, drink <strike>lots</strike> 1x10<sup>6</sup>
drops of H<sub><small><small>2</small></small></sub>0.
```

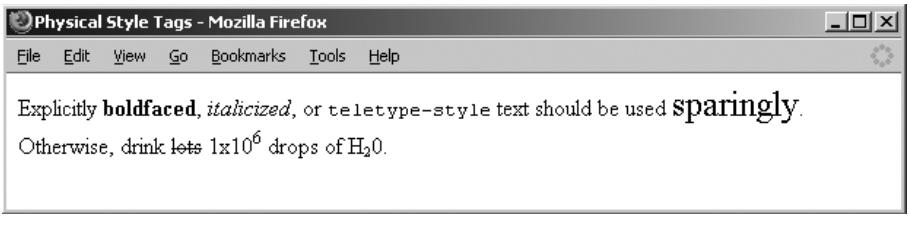

Рис. 4.12. *Пользуйтесь тегами физической разметки осторожно*

4.5.16. Допустимое содержимое

Любой тег физической разметки может содержать любой элемент, допустимый в *тексте*, включая обычный текст, якоря, изображения и перевод на новую строку. Кроме того, вы вправе комбинировать теги логической и физической разметки.

4.5.17. Допустимое употребление

Любой тег физической разметки может быть использован в любом месте, где применяются элементы, относящиеся к *тексту*. На практике

это означает, что они употребляются везде, кроме сегментов, выделенных тегами <title>, <listing>, <xmp>. Применение тегов физических стилей в заголовках допустимо, но их влияние может быть перекрыто действием самого заголовка.

4.5.18. Комбинирование физических стилей

Ваши эксперименты с комбинированием физических стилей будут, вероятно, более удачными, чем опыты с логическими стилями. Например, следующий текст все популярные броузеры отобразят жирным курсивом:

```
<b><i>Затем она взорвалась!</i></b>
```

На практике другие броузеры, возможно, не обратят внимания на такого рода вложенные теги. Стандарты требуют, чтобы броузер «старался» поддерживать всевозможные комбинации стилей, но не определяет, как именно следует эти комбинации обрабатывать.

Хотя большинство броузеров делает удачные попытки следовать требованию стандарта, не рассчитывайте все же, что все комбинации стилей окажутся вам доступны.

4.6. Точные интервалы и макет

Не будем пока рассматривать каскадные таблицы стилей, так как изначальная идея HTML состояла в том, чтобы организовать содержание документа без определения его формата, иными словами, следовало прорисовать структуру и смысл документа, а не задавать в точности, как он будет представлен пользователю. Пусть броузер сам переносит текст со строки на строку, определяет расстояние между символами и между строками – это его дело. Содержание документа, богатство его информации, а вовсе не внешний вид – вот что имеет значение. Когда же декорация определяет смысл, как в коммерческих презентациях, обратитесь к таблицам стилей для управления планировкой и форматами (см. главу 8).

4.6.1. Тег

Тег
 прерывает нормальный процесс заполнения строк и расстановки переносов текста абзацев в HTML или XHTML-документе. В HTML у него нет закрывающего тега, он просто отмечает в потоке текста точку[1], с которой должна начинаться новая строка. Большинство броузеров просто перестают добавлять слова и изображения в текущую строку, спускаются на следующую, отступают до левого края и возобновляют вывод.

---

[1] В XHTML помещайте признак завершения внутрь открывающего тега:
. Обратитесь к главе 16 «XHTML».

Функция:	Вставляет в поток текста разрыв строки
Атрибуты:	`class`, `clear`, `id`, `style`, `title`
Закрывающий тег:	Отсутствует в HTML; `</br>` или `<br ... />` в XHTML
Содержит:	Ничего
Может содержаться в:	*тексте*

Такой эффект полезен при форматировании традиционного текста с фиксированными разрывами строк, например адреса, текста песни или стихотворения. Посмотрите для примера, как разбивается на строки текст песни, отображаемой GUI-броузером:

```
<h3>
Heartbreak Hotel</h3>
<p>
Ever since my baby left me<br>
I've found a new place to dwell.<br>
It's down at the end of lonely street<br>
Called <cite>Heartbreak Hotel</cite>.
</p>
```

Результаты показаны на рис. 4.13.

Рис. 4.13. *Пишите стихотворения построчно (
)*

Отметьте еще и то, что тег `<br>` просто заставляет текст начинаться с новой строки, тогда как появление тега `<p>` влечет за собой отображение дополнительного вертикального промежутка, который разделяет смежные абзацы. [тег <p>, 4.1.2]

4.6.1.1. Атрибут clear

Обычно тег `<br>` предписывает броузеру немедленно остановить поток текста и возобновить его от левого края следующей строки (или от пра-

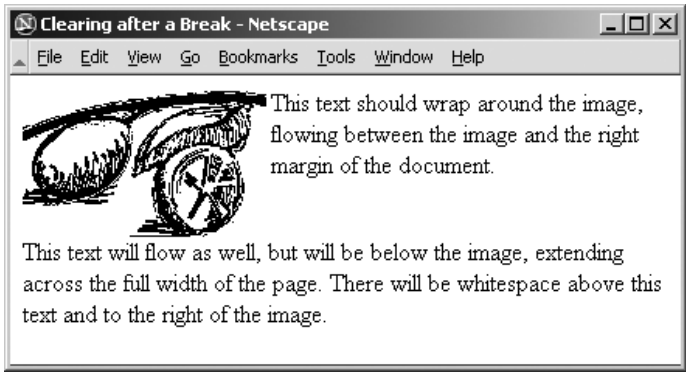

Рис. 4.14. *Вывод текста возобновляется под рисунком*

вого края выровненного влево рисунка или таблицы). Но иногда хотелось бы продолжить поток текста ниже рисунка или таблицы, блокирующих левое или правое поля.

HTML и XHTML предоставляют возможность добиться этого при помощи атрибута `clear` тега `<br>`. Данный атрибут принимает одно из трех значений: `left`, `right`, `all`. Каждое из значений относится к соответствующему краю или к обеим сторонам окна (`all`). Там, где указанный в атрибуте край или оба края свободны от таблиц и рисунков, броузер возобновляет отображение текста.

Вот как выглядит эффект атрибута `clear` при отображении следующего HTML-документа (рис. 4.14):

```
<img src="kumquat.gif" align=left>
This text should wrap around the image, flowing between the
image and the right margin of the document.
<br clear=left>
This text will flow as well, but will be below the image,
extending across the full width of the page.
There will  be white space above this text and
to the right of the image.
```

Встроенные изображения ведут себя в точности так же: они располагаются в строке с текстом, но лишь в ней одной. Последующий текст выводится ниже изображения, если только оно не выровнено специально путем присвоения атрибуту `align` тега `<img>` значений `left` или `right` (аналогично для тега `<table>`). Следовательно, атрибут `clear` тега `<br>` имеет смысл только с выровненными влево или вправо изображениями или таблицами. [тег <img>, 5.2.6] [атрибут align (нежелателен), 10.2.1.1]

Следующий фрагмент XHTML-кода показывает, как используются тег `<br>` и его атрибут `clear` в сочетании с атрибутами тега `<img>` для помещения подписей прямо над рисунком, справа по центру от рисунка и под рисунком, который выровнен по левому краю окна броузера:

```
Paragraph tags separate leading and following
text flow from the captions.
<p>
I'm the caption on top of the image.
<br />
<img src="kumquat.gif" align="absmiddle" />
This one's centered on the right.
<br clear="left" />
This caption should be directly below the image.
</p>
<p />
```

Рис. 4.15 иллюстрирует действие этого примера.

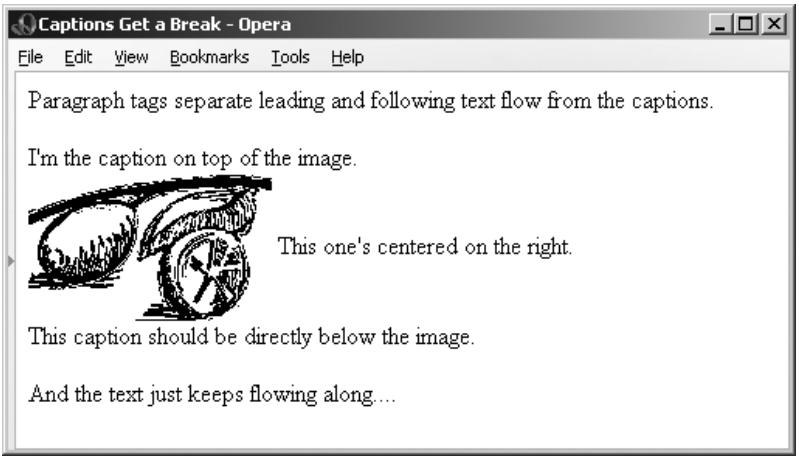

***Рис. 4.15.** Подписи, размещенные над рисунком, справа по центру от рисунка и под рисунком*

Можно также вставить тег `<br clear=all>` сразу после тега `<img>` или после таблицы, находящихся в самом конце раздела. Сделав так, вы получаете гарантию, что текст следующего раздела не всплывет из-под рисунка, приводя читателей в недоумение. [тег <img>, 5.2.6]

4.6.1.2. Атрибуты class, id, style и title

Можно ассоциировать с тегом `<br>` дополнительные правила отображения его содержимого, используя таблицы стилей. Новые правила применяются к тегу `<br>`, например, при помощи атрибута `style` или атрибута `class`. [встроенные стили: атрибут style, 8.1.1] [стилевые классы, 8.3]

Вы можете также присвоить каждому тегу `<br>` свой уникальный `id`, как и менее строгое название (`title`), используя соответствующие атрибуты и их значения в виде заключенных в кавычки строк. [атрибут id, 4.1.1.4] [атрибут title, 4.1.1.5]

4.6.2. Тег <nobr> (расширение)

Иногда хочется, чтобы некая фраза была представлена в окне пользовательского броузера в точности одной строкой текста, даже если при этом она не поместится в окне целиком. Хороший пример – это компьютерные команды. Как правило, вы набираете команды, пусть и состоящие из многих слов, в одну строку. Поскольку невозможно знать, сколько слов помещается в строку окна пользовательского броузера, не исключено, что последовательность, составляющая компьютерную команду, окажется разбита на две или большее число строк. Синтаксис команды и так сложен, нет необходимости запутывать читателя еще сильнее, сворачивая строку в несколько раз.

В стандартах HTML и XHTML существует способ предохранить текст от вторжения броузера. Вы можете заключить текст в теги <pre> и отформатировать его вручную. Это приемлемый способ, и его поймут почти все броузеры. Однако содержимое тега <pre> отображается шрифтом, который отличается от основного, и выставленные вручную переходы на новую строку внутри этого тега не всегда обрабатываются корректно. [тег <pre>, 4.6.5]

<nobr> 

Функция:	Создает область, в которой текст не разрывается
Атрибуты:	Нет
Закрывающий тег:	</nobr>; всегда присутствует
Содержит:	*текст*
Может содержаться в:	*блоке*

Современные броузеры предлагают тег <nobr> как альтернативу <pre>, пригодную в случае, когда нужно гарантировать сохранение целостности некой строки текста, не влияя при этом на ее начертание.[1] Действие его состоит в том, что броузер обращается с текстом, заключенным в тег <nobr>, как с одним большим словом. Содержимое тега отображается текущим шрифтом и можно перейти внутри тега к другому стилю.

Вот <nobr> в действии в примере со строкой компьютерной команды:

```
When prompted by the computer, enter
<nobr>
<tt>find . -name \*.html -exec rm \\;</tt>.
</nobr>
<br>
<nobr>After a few moments, the load on your server will begin
to diminish and will eventually drop to zero.</nobr>
```

[1] Имейте в виду, что <nobr> и его коллега <wbr> принадлежат к расширениям языка и не являются частью стандарта HTML.

Заметьте, глядя на пример исходного текста и его отображение (рис. 4.16), что мы вставили в первый тег <nobr> тег <tt>, тем самым требуя, чтобы содержимое передавалось моноширинным шрифтом. Если заключенный в тег <nobr> текст не помещается в частично заполненную строку, поддерживающий это расширение броузер вставит перед ним переход на новую строку, как видно на рисунке. Второй сегмент <nobr> в нашем примере демонстрирует, что текст может выйти за правую границу окна, если сегмент слишком длинный и не помещается на одной строчке. Среди популярных броузеров только Netscape по неизвестной причине не предоставляет горизонтальную полосу прокрутки, позволяющую прочитать скрытую часть текста. [тег <tt>, 4.5.10]

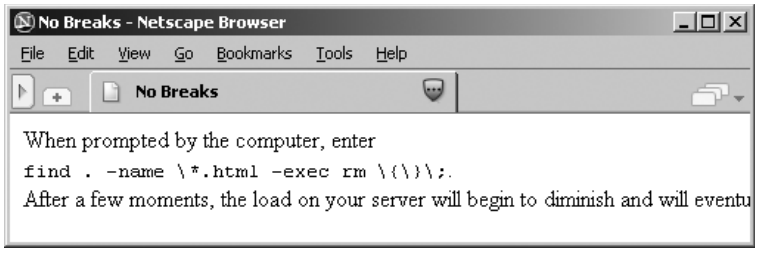

***Рис. 4.16.** Тег расширения <nobr> подавляет разбиение текста на строки; по неизвестной причине Netscape не предоставляет горизонтальную полосу прокрутки, позволяющую прочитать скрытую часть текста*

Тег <nobr> не отменяет и не останавливает нормальный процесс заполнения строк, осуществляемый броузером. Броузер точно так же собирает и вставляет изображения и – поверите или нет – осуществляет вынужденные переходы на новую строку, предписанные, например, тегами
 или <p>. Единственный эффект применения тега <nobr> состоит в подавлении автоматического перехода на новую строку по достижении текущей строкой правого края окна.

Вы можете решить, что этот тег необходим для подавления перехода на новую строку внутри только состоящих из нескольких слов фраз, но не нужен для последовательностей символов, не содержащих пробелов, поскольку они и без того могут выходить за пределы окна броузера. Современные броузеры не переносят слова по слогам со строки на строку, но в один прекрасный день они научатся это делать. Кажется разумным защитить последовательность символов, разрывать которую нежелательно, с помощью тега <nobr>.

4.6.3. Тег <wbr> (расширение)

Тег <wbr> – это последнее слово в расширениях HTML, предлагаемых броузером Internet Explorer (и только им) для достижения настоящего изящества в макетировании текста. Используемый вместе с тегом <nobr>, тег <wbr> отмечает те места в защищенном от разрывов тексте, в которых броузеру разрешается прервать строку. В отличие от тега
,

появление которого всегда, даже внутри тега <nobr>, влечет за собой переход на новую строку, тег <wbr> действует, находясь только в сегменте текста, уже заключенном в теге <nobr>, и вызывает переход на новую строку лишь в том случае, когда текущая строка уже вышла за пределы окна броузера.

<wbr>

Функция:	Определяет места допустимых разрывов текста
Атрибуты:	Нет
Закрывающий тег:	Отсутствует в HTML; </wbr> или <wbr ... /> в XHTML
Содержит:	Ничего
Может содержаться в:	*тексте*

Если <wbr> показался вам слишком заумным, не спешите сердиться. Может наступить момент, когда вы захотите застраховаться от разрывов в некоторых местах документа, однако пожалеете читателя и не сочтете разумным заставлять его надевать рюкзак и отправляться в долгий путь по полосе прокрутки для того только, чтобы узнать, чем кончилось дело в вашем рассказе. Вставляя тег <wbr> в подходящих местах «неразрывной» последовательности слов, вы позволите броузеру деликатно разделить текст на строки разумного размера:

```
<nobr>
This is a very long sequence of text that is
forced to be on a single line, even if doing so causes
<wbr>
the browser to extend the document window beyond the
size of the viewing pane and the poor user must scroll right
<wbr>
to read the entire line.
</nobr>
```

Нетрудно заметить, что при отображении этого фрагмента (рис. 4.17) оба вхождения тега <wbr> оказали свое действие. Растянув окно броузе-

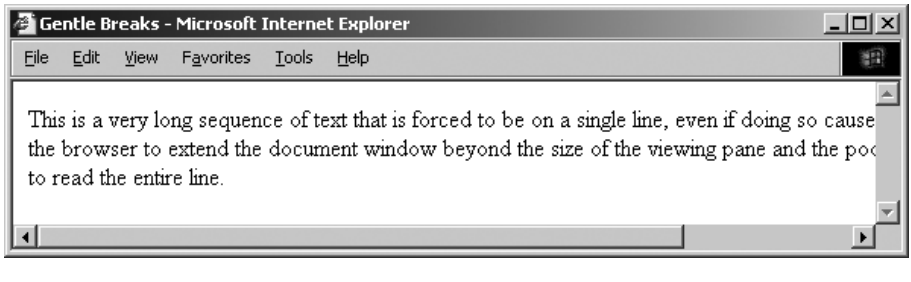

Рис. 4.17. *Деликатное разбиение на строки в окне Internet Explorer с помощью тега <wbr>*

ра по горизонтали или уменьшив шрифт, вы, может быть, сумеете уместить весь предшествующий первому <wbr> текст в окне броузера. В этом случае только второй <wbr> сработает – весь текст, находящийся до него, продолжит строку и скроется за краем окна.

4.6.4. Хороший тон при разбиении на строки

В отличие от большинства своих собратьев, упоминаемые в этой книге популярные броузеры, и это делает им честь, не рассматривают теги в качестве хорошего повода перейти на новую строку. Задумайтесь над неприятными последствиями для вашего документа, если бы, отображая нижеследующий пример, броузер поместил бы запятую, смежную с «du», или точку, смежную с «df», на отдельной строке.

```
Убедитесь, что вы набрали <tt>du</tt>, а не <tt>df</tt>.
```

4.6.5. Тег <pre>

Тег <pre> и его обязательный закрывающий тег </pre> выделяют часть исходного текста, которую броузер должен отобразить «как есть», теми же символами и с тем же разбиением на строки. Обычное расположение текста и заполнение абзацев отключаются, и все «избыточные» пробелы в начале и в конце заключенного в теги фрагмента учитываются. Весь текст между тегами <pre> и </pre> броузеры отображают моноширинным шрифтом.

<pre>

Функция:	Выводит блок текста без какого-либо форматирования
Атрибуты:	class, dir, id, lang, onClick, onDblClick, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, style, title, width
Закрывающий тег:	</pre>; присутствует обязательно
Содержит:	*pre_content* (преформатированное содержимое)
Может содержаться в:	*блоке*

Авторы часто используют тег «преформатированного» вывода <pre>, когда должна быть сохранена целостность строк и столбцов символов, например при выводе числовых таблиц, которые должны быть правильно разбиты. С помощью <pre> можно также создать пустой сегмент – последовательность пустых строк, для того чтобы скрыть от читателя часть документа на время его загрузки в броузер.

Символы табуляции в <pre>-блоке действуют, как обычно, с шагом в восемь символов. Мы тем не менее советуем воздерживаться от их употребления, поскольку их обработка разными броузерами может отличаться. Для гарантированного выравнивания по горизонтали текста внутри тега <pre> лучше применять пробелы.

Тег <pre> чаще всего используется для представления исходных кодов программ, как в нижеследующем примере:

```
<p>
The processing program is:
<pre>
main(int argc, char **argv)
{
   FILE *f;
   int i;

   if (argc != 2)
   fprintf(stderr, "usage: %s %lt;file&gt;\n",
      argv[0]);
   <a href="http:process.c">process</a>(argv[1]);
   exit(0);
}
</pre>
```

Результат показан на рис. 4.18.

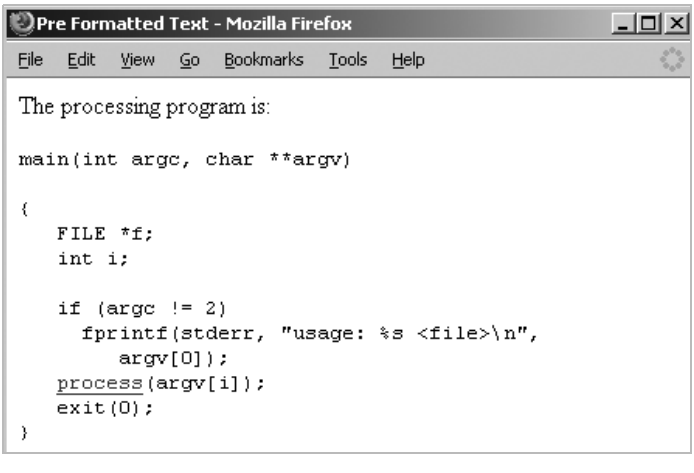

Рис. 4.18. *Используйте <pre> для сохранения целостности строк и столбцов*

4.6.5.1. Допустимое содержимое

Текст внутри тега <pre> может содержать теги физической и логической разметки так же, как и якоря, изображения и горизонтальные линейки. По возможности броузер учтет модификации стилей с тем ограничением, что весь текст в <pre>-блоке будет отображен моноширинным шрифтом.

Теги, вызывающие переход на новую строку (например, заголовки, <p>, <address>), не следует употреблять внутри <pre>-блока. Хотя некоторые броузеры проинтерпретируют тег <p> как простой переход на новую строку, нельзя ожидать, что так будут действовать все броузеры.

Поскольку внутри <pre>-блока допустимо применение тегов, вам придется пользоваться кодировкой для некоторых символов: < для <, > для > и & для амперсанда (&).

В <pre>-блок теги вставляются точно так же, как в любое другое место в HTML/XHTML-документе. Рассмотрите в качестве примера гиперссылку для функции process() в предыдущем примере. С помощью тега <a> она указывает на исходный файл *process.c*.

4.6.5.2. Атрибут width

У тега <pre> есть необязательный атрибут width, указывающий число символов, которые следует поместить в одной строке <pre>-блока. Броузер может использовать это значение, подбирая шрифт и его размер таким образом, чтобы строка указанной ширины целиком поместилась в окне. Это не значит, что броузер станет автоматически переносить текст, если его ширина превышает значение, равное значению параметра width. Напротив, строки, которые окажутся длиннее указанной ширины, просто выйдут за пределы окна.

Атрибут width – это только пожелание, сообщаемое броузеру пользователя. Броузер может суметь, а может и не суметь подобрать нужный шрифт.

4.6.5.3. Атрибуты dir и lang

Атрибут dir советует броузеру, в каком направлении следует выводить текст <pre>-блока, а атрибут lang позволяет определить язык, который употребляется в теге. [атрибут dir, 3.6.1.1] [атрибут lang, 3.6.1.2]

4.6.5.4. Атрибуты class, id, style и title

Хотя содержимое <pre>-блока выводится броузерами с применением определенного стиля, эти установки можно отменить и добавить специальные эффекты, такие как фоновая картинка, назначив для тега новый стиль. Этот новый стиль может быть применен к тегам <pre> при помощи атрибута style или атрибута class. [встроенные стили: атрибут style, 8.1.1] [стилевые классы, 8.3]

Также можно присвоить каждому тегу <pre> свой уникальный id, как и менее строгое название (title), используя соответствующие атрибуты и их значения в виде заключенных в кавычки строк. [атрибут id, 4.1.1.4] [атрибут title, 4.1.1.5]

4.6.5.5. Атрибуты событий

Как и в случае большинства других заключенных в теги сегментов содержимого, инициированные пользователем события с мышью и клавиатурой могут происходить как внутри тега <pre>, так и вне его. Множество таких событий распознаются современными броузерами, и с помощью соответствующих on-атрибутов и их значений можно реагиро-

вать на эти события, отображая окно диалога с пользователем или активизируя какое-нибудь мультимедийное событие. [обработчики событий JavaScript, 12.3.3]

4.6.6. Тег (нежелателен)

Тег – один из тех, чье действие очевидно, – его содержимое, включая текст, графику, таблицы и т. д., выравнивается по центру горизонтально в окне броузера. Применительно к тексту это означает, что каждая строка центрируется после того, как текст выведен и разбит на строки. Выравнивание по центру сохраняет свое действие от появления тега до возникновения закрывающего тег .

** ** 

Функция:	Центрирует фрагмент текста
Атрибуты:	align, class, dir, id, lang, onClick, onDblClick, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, style, title
Закрывающий тег:	; присутствует обязательно
Содержит:	*body_content* (содержимое тела)
Может содержаться в:	*блоке*

Этот обычный, строка за строкой, хотя и примитивный способ центрирования текста следует применять с умом. Броузер не будет пытаться сбалансировать абзац или другие объекты, относящиеся к блокам, такие как элементы списка. Пусть центрированный текст будет коротким и приятным. Названия – это хорошие кандидаты для центрирования, а центрированный список, напротив, обычно трудно просматривать. HTML-авторы употребляют для центрирования картинок или таблиц (для внедренных картинок и таблиц нет опции выравнивания по центру, хотя этот эффект может быть достигнут с применением таблиц стилей).

Так как пользователи любят варьировать ширину окна, разрешение дисплея и т. д., вы, быть может, захотите работать с тегами расширений <nobr> и <wbr> (см. разд. 4.6.2 и 4.6.3), чтобы сохранить центрированный текст неискаженным и приятным глазу. К примеру:

```
<center>
<nobr>
Copyright 2000 by QuatCo Enterprises.<wbr>
All rights reserved.
</nobr>
</center>
```

Тег <nobr> в приведенном примере гарантирует, что текст останется на одной строке, а тег <wbr> указывает броузеру, где следует разорвать строку, если она выйдет за пределы окна.

Тег годится и для создания заголовков, четко отделяющих разделы один от другого, хотя того же эффекта можно достигнуть при помощи явного присвоения значения атрибуту align=center в соответствующем теге заголовка. Нетрудно центрировать текст, используя align=center в тегах или <p>. В целом, тег может быть заменен на эквивалентный или нечто подобное, и его употребление не рекомендуется.

Действительно, подобно тегу <font> и другим тегам стандарта HTML 3.2, впавшим в немилость с появлением таблиц стилей, тег признан нежелательным стандартами HTML 4 и XHTML и подлежит замене на его CSS-эквивалент. Тем не менее он используется чуть ли не во всех HTML-документах, и популярные броузеры наверняка будут поддерживать его во множестве будущих версий. Все же не забывайте, что в конце концов он исчезнет.

4.6.6.1. Атрибуты dir и lang

Атрибут dir советует броузеру, в каком направлении следует выводить текст в сегменте , а атрибут lang позволяет определить язык, который будет употребляться в этом теге. [атрибут dir, 3.6.1.1] [атрибут lang, 3.6.1.2]

4.6.6.2. Атрибуты class, id, style и title

Используйте атрибут style для встроенного в тег определения стиля содержимого тега . Атрибут class позволяет применять к содержимому тега заранее определенный для данного класса стиль. [встроенные стили: атрибут style, 8.1.1] [стилевые классы, 8.3]

Также можно присвоить каждому тегу свой уникальный id, как и менее строгое название (title), используя соответствующие атрибуты и их значения в виде заключенных в кавычки строк. [атрибут id, 4.1.1.4] [атрибут title, 4.1.1.5]

4.6.6.3. Атрибуты событий

Как и в случае большинства других заключенных в теги сегментов содержимого, инициированные пользователем события с мышью и клавиатурой могут происходить как внутри тега , так и вне его. Множество таких событий распознаются современными броузерами, и с помощью соответствующих on-атрибутов и их значений вы можете реагировать на эти события, отображая окно диалога с пользователем или активизируя какое-нибудь мультимедийное событие. [обработчики событий JavaScript, 12.3.3]

4.6.7. Тег <listing> (архаизм)

Тег <listing> – это устаревший тег, он явным образом исключен из стандарта HTML 4, а значит, вам не следует его употреблять. Мы вклю-

чили его сюда из исторических соображений, поскольку он поддерживается некоторыми броузерами и действует почти так же, как тег <pre width=132>.

<listing>

Функция:	Выводит блок текста без какого-либо форматирования
Атрибуты:	class, style
Закрывающий тег:	</listing>; присутствует обязательно
Содержит:	*literal_text* (гладкий текст)[1]
Может содержаться в:	*блоке*

Единственное различие между <pre> и <listing> состоит в том, что внутри тега <listing> не допускается никакой разметки, следовательно, нет необходимости заменять символы "<", ">" и "&" их кодировкой, как это приходится делать в <pre>-блоке.

Поскольку тег <listing> совпадает по действию с тегом <pre width=132>, и, возможно, не будет поддерживаться в последующих версиях популярных броузеров, мы рекомендуем отказаться от его использования.

4.6.8. Тег <xmp> (архаизм)

Подобно тегу <listing>, тег <xmp> устарел, и его не следует употреблять, несмотря на то, что популярные броузеры его поддерживают. Мы включили его сюда в основном по историческим соображениям.

<xmp>

Функция:	Выводит блок текста без какого-либо форматирования
Атрибуты:	class, style
Закрывающий тег:	</xmp>; присутствует обязательно
Содержит:	*literal_text* (гладкий текст)
Может содержаться в:	*блоке*

Тег <xmp> форматирует текст так же, как тег <pre> с шириной, определенной в 80 символов. В отличие от тега <pre>, внутри <xmp> нет необходимости заменять символы "<", ">" и "&" их кодами. Имя <xmp> – это сокращение слова «example» (пример). Разработчики языка предполагали, что тег будет использоваться для изображения текста подобно тому, как он выглядел на дисплеях шириной в 80 символов. Поскольку

[1] Не содержит никаких управляющих символов. – *Примеч. науч. ред.*

восьмидесятисимвольные дисплеи ушли той же дорогой, что и зеленые экраны текстовых терминалов или телетайпы, а действие тега <xmp>, по существу, совпадает с действием тега <pre width=80>, не употребляйте его. Он может исчезнуть в последующих версиях HTML.

4.6.9. Тег <plaintext> (архаизм)

Выбросьте тег <plaintext> из вашей коллекции HTML-приемов, он устарел так же, как <xmp> и <listing>. Включение его в эту книгу – дань истории. Прежде авторы использовали <plaintext>, приказывая броузеру обращаться со всем последующим текстом так, как он написан, причем за тегом <plaintext> разметка уже не допускалась. У <plaintext> нет закрывающего тега (поскольку нет разметки вообще), но сам он пришел к своему завершению. Забудьте о нем.

<plaintext>	
Функция:	Выводит блок текста без какого-либо форматирования
Атрибуты:	Нет
Закрывающий тег:	Нет
Содержит:	*literal_text* (гладкий текст)
Может содержаться в:	*блоке*

4.7. Блоки цитат

Обычный элемент в традиционных документах – это блочная цитата, длинная выдержка из другого документа. Короткие цитаты обычно выделяются кавычками, тогда как блочные состоят из нескольких абзацев, представленных, как правило, со специальным отступом и иногда с применением наклонного шрифта. Эти признаки цитат поддаются управлению при помощи определений стилей и классов (обратитесь к главе 8).

4.7.1. Тег <blockquote>

Заключенный между <blockquote> и </blockquote> текст обычно выделяется из основного отступами от левого и правого края окна, а иногда и применением наклонного шрифта. Реальное отображение меняется, конечно, от броузера к броузеру.

Стандарты HTML и XHTML допускают внутри тега <blockquote> любую разметку, хотя некоторые физические и логические стили могут конфликтовать с принятым для отображения блочной цитаты шрифтом. Экспериментируя с тегом, легко обнаружить эти неувязки.

<blockquote>	
Функция:	Определяет блочную цитату
Атрибуты:	`cite`, `class`, `dir`, `id`, `lang`, `onClick`, `onDblClick`, `onKeyDown`, `onKeyPress`, `onKeyUp`, `onMouseDown`, `onMouseMove`, `onMouseOut`, `onMouseOver`, `onMouseUp`, `style`, `title`
Закрывающий тег:	`</blockquote>`; присутствует обязательно
Содержит:	*body_content* (содержимое тела)
Может содержаться в:	*блоке*

Тег `<blockquote>` часто используется для выделения длинных заимствований из других документов. К примеру, популярные броузеры выведут следующую цитату в тексте в виде блока с отступом:

```
We acted incorrectly in arbitrarily changing the Kumquat
Festival date. Quoting from the Kumquat Growers' Bylaws:
<blockquote>
  The date of the Kumquat Festival may only be changed by
  a two-thirds vote of the General Membership, provided
  that a <strong>60 percent quorum</strong> of the Membership
  is present.
</blockquote>
(Emphasis mine) Since such a quorum was not present, the
vote is invalid.
```

Рис. 4.19 демонстрирует результат.

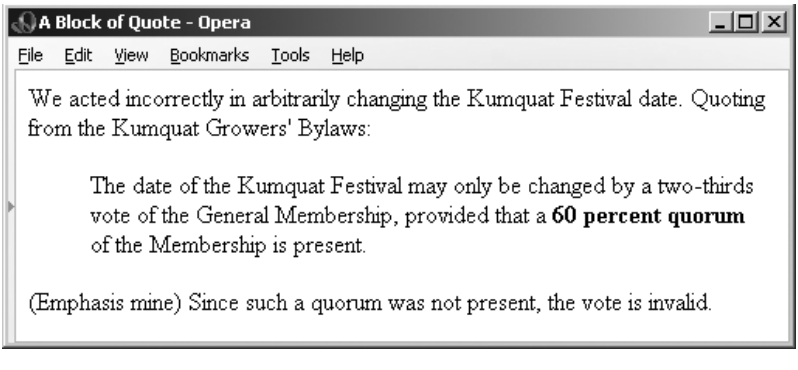

***Рис. 4.19.** Длинная цитата занимает свое место*

4.7.1.1. Атрибут cite

Атрибут `cite` позволяет указать источник цитаты. Значение атрибута должно быть URL, заключенным в кавычки и указывающим на внешний документ, из которого сделана выписка.

Например, попробуйте сослаться на определенный раздел в «Кодексе чести Кумкватоводов» из нашего примера. Наверное, когда-нибудь вы сможете просто щелкнуть мышью на цитате, и броузер покажет вам документ, на который ссылается атрибут `cite`. Пока же для достижения такого эффекта вам придется вставить в документ гиперссылку явным образом. Детальное описание – в главе 6.

```
<blockquote cite="http://www.kumquat.com/growers/bylaws#s23.4">
```

4.7.1.2. Атрибуты dir и lang

Атрибут `dir` советует броузеру, в каком направлении следует выводить текст в сегменте `<blockquote>`, а атрибут `lang` позволяет определить язык, который будет употребляться в этом теге. [атрибут dir, 3.6.1.1] [атрибут lang, 3.6.1.2]

4.7.1.3. Атрибуты class, id, style и title

Используйте атрибут `style` для встроенного в тег определения стиля содержимого тега `<blockquote>`. Атрибут `class` позволяет применять к содержимому заранее определенный для данного класса стиль. [встроенные стили: атрибут style, 8.1.1] [стилевые классы, 8.3]

Можно также присвоить каждому тегу `<blockquote>` свой уникальный `id`, как и менее строгое название `title`, используя соответствующие атрибуты и их значения в виде заключенных в кавычки строк. [атрибут id, 4.1.1.4] [атрибут title, 4.1.1.5]

4.7.1.4. Атрибуты событий

Как и в случае большинства других заключенных в теги сегментов содержимого, инициированные пользователем события с мышью и клавиатурой могут происходить как внутри тега `<blockquote>`, так и вне его. Множество таких событий распознаются современными броузерами, и с помощью соответствующих on-атрибутов и их значений вы можете реагировать на эти события, отображая окно диалога с пользователем или активизируя какое-нибудь мультимедийное событие. [обработчики событий JavaScript, 12.3.3]

4.7.2. Тег <q>

Введенный стандартом HTML 4.0 тег `<q>` по смыслу совпадает со своим товарищем `<blockquote>`. Различие состоит в том, как они употребляются и отображаются. Используйте тег `<q>` для коротких цитат, находящихся, возможно, в одной строке с окружающим их обычным текстом. Стандарты HTML и XHTML требуют, чтобы заключенный в теге `<q>` текст начинался и заканчивался двойными кавычками. Все популярные броузеры, кроме Internet Explorer, поддерживают тег `<q>` и ставят двойные кавычки в начале и конце обозначенного текста. В результате вы получите два набора двойных кавычек, если в угоду

броузеру Internet Explorer поставите собственные. И все-таки мы рекомендуем вам пользоваться тегом <q>, и не только потому, что любим стандарты. Просто мы смотрим на них чуть шире, чем с утилитарной точки зрения вывода документов, извлечения данных и т. д.

Применяйте, напротив, тег <blockquote> для более длинных сегментов, которые броузер выделит из окружающего текста (обычно блоком текста с отступами), как на рис. 4.20.

<q>

Функция:	Определяет короткую цитату
Атрибуты:	cite, class, dir, id, lang, onClick, onDblClick, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, style, title
Закрывающий тег:	</q>; присутствует обязательно
Содержит:	*body_content* (содержимое тела)
Может содержаться в:	*тексте*

4.7.2.1. Атрибут cite

Атрибут cite работает с тегом <q> точно так же, как с тегом <blockquote>, – он позволяет указать источник цитаты. Значение атрибута должно быть URL, заключенным в кавычки и указывающим на онлайновый документ, из которого сделана выписка.

4.7.2.2. Атрибуты dir и lang

Атрибут dir советует броузеру, в каком направлении следует выводить текст в сегменте <q>, а атрибут lang позволяет определить язык, который будет употребляться в этом теге. [атрибут dir, 3.6.1.1] [атрибут lang, 3.6.1.2]

4.7.2.3. Атрибуты class, id, style и title

Используйте атрибут style для встроенного в тег определения стиля содержимого тега <q>. Атрибут class позволяет применять к содержимому тега заранее определенный для данного класса стиль. [встроенные стили: атрибут style, 8.1.1] [стилевые классы, 8.3]

Можно также присвоить каждому тегу <q> свой уникальный id, как и менее строгое название title, используя соответствующие атрибуты и их значения в виде заключенных в кавычки строк. [атрибут id, 4.1.1.4] [атрибут title, 4.1.1.5]

4.7.2.4. Атрибуты событий

Как и в случае большинства других заключенных в теги сегментов содержимого, инициированные пользователем события с мышью и клавиатурой могут происходить как внутри тега <q>, так и вне его. Множе-

ство таких событий распознаются современными броузерами, и с помощью соответствующих on-атрибутов и их значений вы можете реагировать на эти события, отображая окно диалога с пользователем или активизируя какое-нибудь мультимедийное событие. [обработчики событий JavaScript, 12.3.3]

4.8. Адреса

Адрес – это очень распространенный элемент текстового документа, поэтому существует специальный тег, отделяющий адреса от остального текста. Хотя и верится с трудом (при форматировании адресов с помощью тега достигается несколько специальных эффектов), но данный тег может служить примером ориентации на содержание, а не на внешний вид, в согласии со смыслом и назначением HTML- и XHTML-разметки.

Определяя текст, составляющий адрес, авторы оставляют броузеру выбор способа его отображения, так же как и других методов его обработки, которые пойдут на пользу читателю документа. Выделение адресов делает их также доступными для автоматической обработки и извлечения. К примеру, раздел веб-сайта может содержать таблицу или отдельный документ, в которых отображаются собранные броузером адреса. Или же автоматизированным способом можно извлекать адреса из некоторого собрания документов для последующего их включения в базу данных адресов.

4.8.1. Тег <address>

Тег `<address>` и его обязательное завершение `</address>` говорят броузеру, что заключенный между ними текст является контактным адресом обычной или электронной почты. Впрочем, текст может содержать и другую контактную информацию. Броузер отображает этот текст, выделяя адрес из окружающего текста, или использует его другим способом. Вы также можете управлять параметрами отображения адреса с применением атрибутов `class` и `style` этого тега (подробная информация в главе 8).

<address>

Функция:	Определяет адрес
Атрибуты:	`class`, `dir`, `id`, `lang`, `onClick`, `onDblClick`, `onKeyDown`, `onKeyPress`, `onKeyUp`, `onMouseDown`, `onMouseMove`, `onMouseOut`, `onMouseOver`, `onMouseUp`, `style`, `title`
Закрывающий тег:	`</address>`; присутствует обязательно
Содержит:	*body_content* (содержимое тела)
Может содержаться в:	*address_content* (содержимое адреса)

Текст внутри тега <address> может содержать любые элементы тела документа, за исключением еще одного тега <address>.

Изменения стиля допускаются, но могут конфликтовать со стилем, выбранным броузером для отображения адресов.

Мы полагаем, что большинство документов, если не все, должны содержать адреса авторов, помещенные в каком-нибудь удобном для пользователей месте, например в конце документа. На худой конец в качестве адреса может быть указан адрес электронной почты автора или вебмастера, снабженный ссылкой на его домашнюю страницу.

Домашний адрес и телефон не обязательны и редко включаются по конфиденциальным соображениям.

К примеру, адрес вебмастера, отвечающего за собрание коммерческих документов, представленных в сети, часто появляется в исходных текстах в виде, подобном представленному ниже, и включает mailto: URL с протоколом, позволяющим броузеру вызвать программу электронной почты.

```
<address>
  <a href="mailto:webmaster@oreilly.com">Webmaster</a><br>
  O'Reilly<br>
  Cambridge, Massachusetts<br>
</address>
```

На рис. 4.20 показан результат. Во всех популярных броузерах адрес по умолчанию выводится курсивом.

Каким бы ни был ваш документ, неповторимым и кратким или длинным и полным, убедитесь, что в нем содержится адрес. Если документ заслуживает того, чтобы его написали и опубликовали в Сети, он достоин и откликов, и вопросов со стороны читателей. Анонимным документам в сети не доверяют.

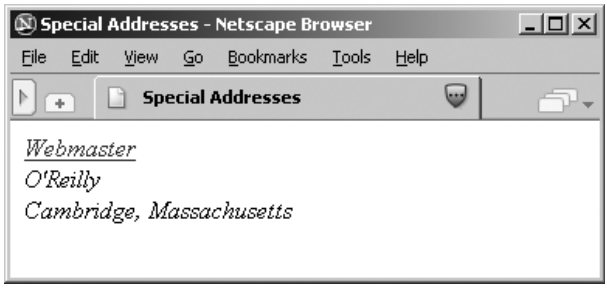

***Рис. 4.20.** Тег <address> в действии*

4.8.1.1. Атрибуты dir и lang

Атрибут dir советует броузеру, в каком направлении следует выводить текст в сегменте <address>, а атрибут lang позволяет определить язык,

который будет употребляться в этом теге. [атрибут dir, 3.6.1.1] [атрибут lang, 3.6.1.2]

4.8.1.2. Атрибуты class, id, style и title

Используйте атрибут `style` для встроенного в тег определения стиля содержимого тега `<address>`. Атрибут `class` позволяет применять к содержимому тега заранее определенный для данного класса стиль. [встроенные стили: атрибут style, 8.1.1] [стилевые классы, 8.3]

Вы можете также присвоить каждому тегу `<address>` свой уникальный `id`, как и менее строгое название `title`, используя соответствующие атрибуты и их значения в виде заключенных в кавычки строк. [атрибут id, 4.1.1.4] [атрибут title, 4.1.1.5]

4.8.1.3. Атрибуты событий

Как и в случае большинства других заключенных в теги сегментов содержимого, инициированные пользователем события с мышью и клавиатурой могут происходить как внутри тега `<address>`, так и вне его. Множество таких событий распознаются современными броузерами, и с помощью соответствующих on-атрибутов и их значений вы можете реагировать на эти события, отображая окно диалога с пользователем или активизируя какое-нибудь мультимедийное событие. [обработчики событий JavaScript, 12.3.3]

4.9. Кодирование специальных символов

Содержащиеся в документе символы, не являющиеся при этом частью тегов, обычно возвращаются броузером «как есть». Некоторые символы, однако, имеют специальное значение и прямо не выводятся, другие же невозможно набрать со стандартной клавиатуры. Для включения специальных символов в документ требуется их кодирование: изображение либо с помощью специальных названий, либо с помощью числовой кодировки.

4.9.1. Специальные символы

Как ясно из предшествующих примеров и обсуждений, три символа в исходном тексте имеют особое значение: знак «меньше» (<), знак «больше» (>) и амперсанд (&). Эти символы разграничивают теги, а амперсанд является признаком начала символьной ссылки. Оставленные одиноко болтаться в тексте или употребленные несообразно с синтаксисом тегов, они приведут броузер в недоумение. Следовательно, приходится отказаться от их прямого и буквального употребления.[1]

[1] Единственное исключение составляют теги `<listing>` и `<xmp>`, в которых данные символы могут появляться в собственном виде, но это вряд ли важно, поскольку сами теги устарели.

Придется также пользоваться специальной кодировкой для включения двойной кавычки в строку, которая сама заключается в кавычки согласно синтаксическим правилам; или применяя в тексте специальный символ, отсутствующий на клавиатуре, но все же входящий в множество символов Latin-1 стандарта ISO, поддерживаемое большинством броузеров.

4.9.2. Как вставляются специальные символы

Чтобы вставить в документ специальный символ, заключите между амперсандом и точкой с запятой (без пробелов) либо его стандартное название, либо предваренный знаком решетки (#) номер, соответствующий ему в множестве символов стандарта Latin-1.[1] Уф! Это длинное объяснение того, что легко сделать, как видно из следующих примеров. В первом примере отражено, как включить знак «больше» в образец программного кода, используя кодировку символа (символьную ссылку) с применением его стандартного названия. Второй пример показывает, как закодировать символ с применением его порядкового номера в множестве символов Latin-1.

```
if a &gt; b, then t = 0
if a &#62; b, then t = 0
```

Оба примера исходного текста будут отображены так:

```
if a > b, then t = 0
```

Полный набор названий и порядковых номеров символов содержится в приложении F. Вы могли бы записать весь документ, используя закодированные символы, но это было бы глупо.

4.10. Использование расширенной HTML-модели шрифтов (нежелательно)

В предыдущих изданиях этой книги мы с радостью констатировали, что в версии HTML 3.2 представлена модель работы со шрифтами, расширяющая возможности вывода текста. Когда в HTML 4 эти специальные шрифтовые теги были объявлены нежелательными, мы тем не менее оставили их описание на том же видном месте в этой главе, потому что они все-таки были частью стандарта HTML 3.2, пользовались

[1] Популярное множество символов ASCII – это подмножество полного набора символов Latin-1. Составленное почтенной Международной организацией стандартизации (International Organization for Standartization, ISO) множество Latin-1 представляет собой список всех букв, цифр, знаков препинания и т. д., обычно используемых авторами, пишущими на западных языках. Символы этого множества снабжены порядковыми номерами и специальными названиями. В приложении F содержится полный список символов множества Latin-1 и их кодировка.

популярностью у авторов HTML-документов и имели хорошую поддержку со стороны всех популярных броузеров. В этом издании книги мы уже не можем так поступить.

Как и многие морально устаревшие теги и атрибуты HTML, теги расширенной обработки шрифтов из HTML 3.2 безвозвратно канули в прошлое. Самый популярный в мире броузер, Internet Explorer, поддерживает все эти теги; другие броузеры поддерживают одни теги, но игнорируют другие. Мы не исключили из книги описание тегов расширенной модели шрифтов, но перенесли его в конец главы, как бы огородив ее предупредительными красными флажками.

W3C хочет, чтобы для прямого управления стилями шрифтов, цветом и размером текстовых символов HTML-авторы пользовались таблицами стилей, а не локальными тегами. Вот почему средства расширенной модели шрифтов попали в немилость. Одним словом, вам теперь тоже следует сторониться этих тегов.

4.10.1. Расширенная модель размеров шрифтов

Для обозначения размеров расширенная модель шрифтов в HTML 3.2 вместо указания абсолютной величины в пунктах использует относительный метод. Размеры шрифтов меняются от 1 (самый мелкий) до 7 (самый крупный), причем по умолчанию используется шрифт размера 3 (*базовый шрифт*).

Почти невозможно надежно установить действительный размер шрифта, соответствующий виртуальному. Большинство броузеров позволяют читателю изменять физический размер шрифта, в то же время размер, используемый по умолчанию, варьируется от броузера к броузеру. Однако все же полезно знать, что каждый виртуальный размер на 20% больше предыдущего и на 20% меньше последующего. Таким образом, шрифт размера 4 на 20% больше, а размера 5 – на 40% больше базового шрифта, тогда как шрифт размера 2 – на 20% меньше, а размера 1 – на 40% меньше базового шрифта, размер которого равен 3.

4.10.2. Тег <basefont> (нежелателен)

Тег `<basefont>` позволяет определить основной размер шрифта, который будет использоваться броузером при выводе обычного текста документа. Мы не рекомендуем употребление этого тега, так как он признан нежелательным стандартами HTML 4 и XHTML и более не поддерживается броузерами, за исключением Internet Explorer.

Только один атрибут тега `<basefont>` признают в настоящее время все броузеры. Это атрибут `size`, значение которого определяет основной размер шрифта в документе. Данное значение может быть определено абсолютным образом, когда атрибуту присваивается число от 1 до 7, или относительным образом, когда перед числом ставится знак «плюс» или «минус». В последнем случае основной шрифт соответст-

<basefont>	
Функция:	Определяет базовый размер шрифта, относительно которого отсчитываются изменения величин шрифтов
Атрибуты:	color, face, id, name, size
Закрывающий тег:	</basefont>; часто опускается в HTML
Содержит:	Ничего
Может содержаться в:	*блоке*, *head_content* (содержимое заголовка)

венно увеличивается или уменьшается на эту относительную величину. Размер основного шрифта по умолчанию равен 3.

Internet Explorer поддерживает еще два атрибута для тега <basefont> — color и name. HTML 4 определяет также атрибут face в качестве синонима name. Эти атрибуты управляют цветом и начертанием (гарнитурой) шрифта в документе и употребляются совершенно аналогично атрибутам color и face тега <font>, описанного в следующем разделе.

HTML 4 определяет также атрибут id тега <basefont>, позволяющий присваивать тегу имя для обращения в последующем к его содержимому. [атрибут id, 4.1.1.4]

Если авторы используют тег <basefont>, то обычно помещают его в заголовок HTML-документа, устанавливая для него размер основного шрифта. Тем не менее тег с новым атрибутом size может встречаться в документе несколько раз и почти где угодно. При каждом своем появлении тег <basefont> обнаруживает свое действие немедленно и сохраняет его на протяжении всего последующего текста.

В вопиющем противоречии со стандартами HTML и SGML Internet Explorer *не* интерпретирует закрывающий тег </basefont> как прекращающий действие последнего тега <basefont>, а восстанавливает размер основного шрифта, который тот имеет по умолчанию. То есть написать </basefont> это то же самое, что написать <basefont size=3>.

Следующий пример исходного текста и рис. 4.21 показывают, как Internet Explorer реагирует на теги <basefont> и </basefont>:

```
Unless the base font size was reset above,
Inernet Explorer renders this part in font size 3.
<basefont size=7>
This text should be rather large (size 7).
<basefont size=6> Oh,
<basefont size=4> no!
<basefont size=2> I'm
<basefont size=1> shrinking!
</basefont><br>
Ahhhh, back to normal.
```

Мы рекомендуем использовать <basefont size=3>, а не </basefont>.

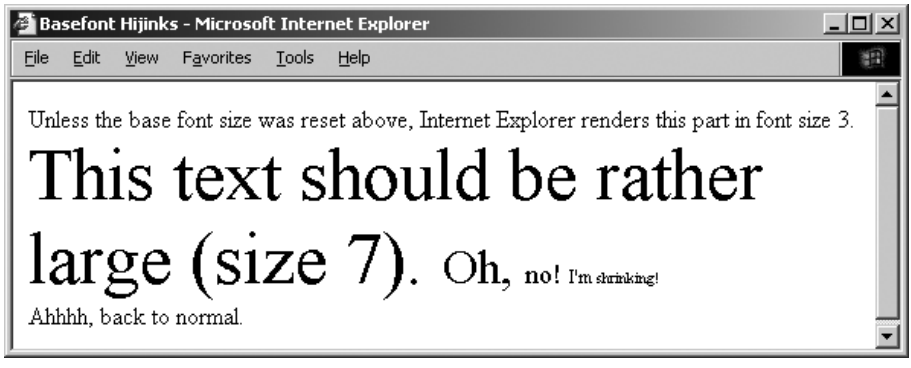

Рис. 4.21. *Играем с <basefont>*

4.10.3. Тег <font> (нежелателен)

Тег `<font>` позволяет изменять размер, стиль и цвет текста. Мы не рекомендуем его употреблять, так как он признан нежелательным стандартами HTML 4 и XHTML, хотя все популярные броузеры по-прежнему поддерживают его. Решив все же пренебречь нашим советом, используйте его так же, как какой-нибудь тег физической или логической разметки для управления внешним видом небольших фрагментов текста.

<font>

Функция:	Устанавливает размер шрифта, цвет текста
Атрибуты:	`class`, `color`, `dir`, `face`, `id`, `lang`, `size`, `style`, `title`
Закрывающий тег:	`</font>`; присутствует обязательно
Содержит:	*текст*
Может содержаться в:	*тексте*

Для управления цветом текста во всем документе применяйте атрибуты тега `<body>`, описанные в разделе 5.3.1.

4.10.3.1. Атрибут size

Значением атрибута `size` должен быть либо один из описанных выше виртуальных размеров (1–7), принимаемый для заключенного в теге текста в качестве абсолютного, либо величина со знаком «плюс» или «минус», рассматриваемая как относительный размер, который броузер прибавляет или вычитает из основного размера шрифта (тег `<basefont>`, раздел 4.10.2). Броузер автоматически ограничивает вычисленное число 1 или 7, когда оно выходит за пределы допустимого промежутка.

Применяйте абсолютные значения размера, если хотите, чтобы текст выводился либо очень большим, либо очень маленьким, или нужно

получить на экране абзац, отображенный шрифтом определенного размера.

К примеру, использование самого крупного шрифта для первой буквы абзаца напоминает стиль старинной рукописи (рис. 4.22):

```
<p>
<font size=7>C</font>all me Ishmael.
```

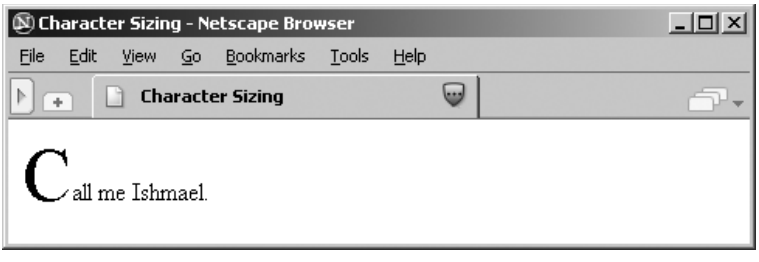

Рис. 4.22. *Создание буквицы в предложении при помощи атрибута size тега <font>*

Абсолютный размер годится также и для вставки чего-нибудь восхитительно неразборчивого – по требованию шаблона либо закона – в самый «подвал» документа (рис. 4.23):

```
<p> <font size=1> All rights reserved. Unauthorized redistribution of this
document is prohibited. Opinions expressed herein are those of the authors,
not the Internet Service Provider.
```

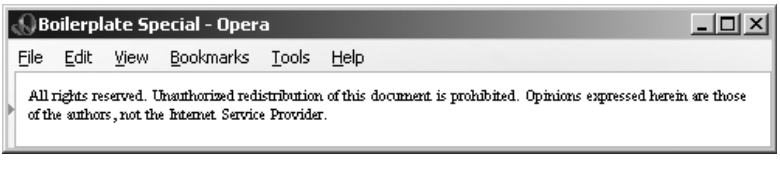

Рис. 4.23. *Используйте мельчайший шрифт для вывода информации об авторских правах*

Исключая крайние позиции, употребляйте относительный размер шрифта, чтобы вывести фрагмент, отличающийся по размеру от окружающего текста, для смыслового подчеркивания слова или фразы (рис. 4.24):

```
<p> Make sure you <font size=+2>always</font> sign and date the form!
```

Если результат перевода указанного относительного размера в абсолютные цифры превышает 7, броузер применяет шрифт размера 7. Точно так же размеры меньше 1 приводятся к 1.

Отметьте, что действия спецификаций `size=+1` и `size=-1` совпадают с действием тегов `<big>` и `<small>` соответственно. Однако вложенные

Рис. 4.24. *Для украшения текста используйте относительные размеры шрифта*

относительные изменения размера с применением тега <font> не имеют кумулятивного эффекта, свойственного тегам <big> и <small>. Каждый тег <font> отсчитывает смещение относительно базового, а не текущего размера шрифта. Например (рис. 4.25):

```
<p>
The ghost moaned, "oo<font size=+1>oo<font size=+2>oo<font
size=+3>oo</font>oo</font>oo</font>oo."
```

Сравните этот пример с действием тегов <big> и <small>, увеличивающих или уменьшающих размер текста на один уровень при каждом новом вложении. [<big>, 4.5.2]

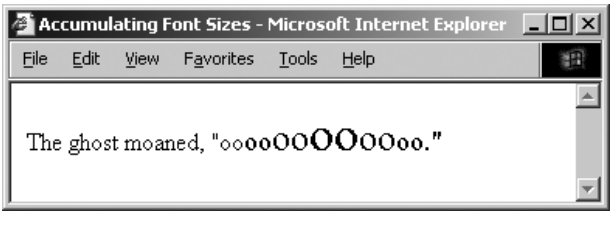

Рис. 4.25. *Относительные размеры шрифта складываются*

4.10.3.2. Атрибут color

Все еще пользующийся поддержкой популярных броузеров атрибут color, применяемый с тегом <font>, устанавливает цвет заключенного в теге текста. Значение атрибута может быть введено одним из двух способов: как определение красного, зеленого и синего (RGB) компонента цвета или как стандартное название цвета. Кавычки рекомендуются, но не являются обязательными.

В цветовой модели RGB[1] обозначение цвета представляет собой шестизначное шестнадцатеричное число, следующее непосредственно за зна-

[1] Цвета, полученные с помощью RGB, могут существенно отличаться на различных платформах (например, под Windows9x и Mac). В будущем, впрочем, если удастся внедрить поддержку броузерами цветовой модели sRGB, которая предполагает использование цветовых профилей ICC, идентичность выводимых цветов заданным намного улучшится. Способ обозначения цветов в sRGB останется таким же, как и в RGB. – *Примеч. науч. ред.*

ком решетки (#). Первые две цифры обозначают интенсивность красного компонента от 00 (нет красного) до FF (ярко-красный). Подобным образом две следующие цифры соответствуют зеленому компоненту и две последние – синему. Черный цвет – это отсутствие цвета (#000000), белый – все цвета (#FFFFFF).

Например, чтобы выкрасить текст в основной желтый цвет, можно написать:

```
Вот восходит <font color="#FFFF00">солнце</font>!
```

Чтобы установить цвет заключенного в тег шрифта альтернативным способом, следует использовать одно из множества стандартных названий. В приложении G приведен список основных цветов. К примеру, вы могли сделать желтым текст в предыдущем фрагменте, написав:

```
Вот восходит <font color=yellow>солнце</font>!
```

4.10.3.3. Атрибут face

В своих предыдущих версиях броузеры Internet Explorer и Netscape Navigator позволяли также изменять стиль шрифта в текстовых отрывках при помощи атрибута `face` для тега `<font>`.[1] Хотя этот атрибут до сих пор поддерживается большинством броузеров, мы настоятельно рекомендуем вам управлять шрифтами с помощью стилей. Броузеры по-разному интерпретируют атрибут `face`, и отсутствующие глифы в шрифте могут спровоцировать непредсказуемые эффекты. Заключенное в кавычки значение атрибута `face` – это одно или несколько названий шрифтов, разделенных запятыми.

Начертание шрифта, которое в итоге выбирает броузер, зависит от состава доступных шрифтов в системе пользователя. Броузер разбирает список названий, предложенных авторами, перебирая одно за другим, пока не обнаружит какого-нибудь, поддерживаемого системой. Если не найдется ни одного из перечня, текст будет отображен шрифтом, который пользователь указал в настройках броузера в качестве предпочтительного. В частности:

```
Это шрифт, установленный по умолчанию. Но,
<font face="Braggadocio, Machine, Zapf Dingbats">
одному Богу известно</font>
каким именно он будет?
```

Если у пользователя броузера есть Braggadocio или Machine, или ни один из трех шрифтов списка не установлен на его системе, то он смо-

[1] Для борцов за чистоту нравов в HTML, когда-то могучих, контролировавших каждый шаг своего броузера, все это только источник ночных кошмаров. Форма важнее функциональности, внешний вид важнее содержания – что дальше? Встроенные рекламные видеоролики, от которых невозможно избавиться?

жет прочитать фразу «одному Богу известно», отображенную либо соответствующими, либо используемым по умолчанию шрифтом. В противном случае этот текст будет искажен, поскольку шрифт Zapf Dingbats содержит символы, а не буквы. Но, может быть, вы и хотели, чтобы эта фраза была закодирована таинственными знаками.

4.10.3.4. Атрибуты dir и lang

Атрибут `dir` советует броузеру, в каком направлении следует выводить текст в теге, а атрибут `lang` позволяет определить язык, который будет в нем употребляться. [атрибут dir, 3.6.1.1] [атрибут lang, 3.6.1.2]

4.10.3.5. Атрибуты class, id, style и title

С тегом `<font>` можно ассоциировать дополнительные правила отображения его содержимого при помощи таблиц стилей. Эти новые правила могут быть применены к тегу `<font>` при помощи атрибута `style` или атрибута `class`. [встроенные стили: атрибут style, 8.1.1] [стилевые классы, 8.3]

Вы можете также присвоить каждому тегу `<font>` свой уникальный `id`, как и менее строгое название (`title`), используя соответствующие атрибуты и их значения в виде заключенных в кавычки строк. [атрибут id, 4.1.1.4] [атрибут title, 4.1.1.5]

- Горизонтальные линейки
- Изображения в документе
- Цвета документа и фоновые изображения
- Звуковой фон
- Анимация текста
- Другое мультимедийное содержимое

5

Линейки, изображения и мультимедийные элементы

Хотя основой большинства документов является текст, приправа из уместно вставленных горизонтальных линеек, изображений и мультимедийных элементов сделает материал более нарядным и привлекательным. Такие средства нельзя воспринимать как дешевую бижутерию, пригодную лишь для украшения документа. Мультимедийные элементы вносят в HTML и XHTML новую жизнь, добавляют в информационное пространство еще одно измерение, позволяя представлять сведения, которые невозможно перенести, например, на бумагу. В этой главе мы детально опишем, как вставить специальные мультимедийные элементы в документ, когда их употребление уместно и как ими не заиграться.

Вы, может быть, захотите также забежать вперед и заглянуть в главу 12. Там мы описываем теги универсальных контейнеров (`<object>` из стандартов HTML 4 и XHTML и `<embed>`, поддерживаемый популярными броузерами), которые позволяют вставить в документ файлы любого типа и с любым содержимым, включая мультимедийные элементы.

5.1. Горизонтальные линейки

Горизонтальная линейка служит средством визуального отделения разделов документа друг от друга. С ее помощью вы элегантно и доходчиво даете читателю понять, что один раздел закончился и начался другой. Горизонтальные линейки эффектно выделяют небольшие куски текста, отграничивают верхние и нижние колонтитулы документа, увеличивают в нем ударную силу заголовков.

5.1.1. Тег <hr>

Тег `<hr>` предлагает броузеру вставить горизонтальную линейку, пересекающую окно броузера от края до края. В языке HTML у него нет за-

крывающего тега. В XHTML-документе требуется либо косая, закрывающая тег перед знаком ">" после атрибутов (<hr .../>), либо закрывающий тег сразу после открывающего (<hr></hr>).

Подобно тегу
, <hr> заставляет выравнивание абзаца вернуться к принятому по умолчанию (по левому краю). Броузер размещает линейку непосредственно под текущей строкой, а поток текста возобновляется сразу под линейкой. [тег
, 4.6.1]

<hr>

Функция:	Разрывает поток текста и вставляет горизонтальную линейку
Атрибуты:	align, class, color, dir, id, lang, noshade, onClick, onDblClick, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, size, style, title, width
Закрывающий тег:	Отсутствует в HTML; </hr> или <hr ... /> в XHTML
Содержит:	Ничего
Может содержаться в:	*body_content* (содержимое тела)

Способ отображения горизонтальной линейки относится к компетенции броузера. Обычно она протягивается через все окно. Графические броузеры могут изображать ее выпуклой или углубленной, броузеры, выводящие только символы, могут обозначить ее строкой тире или подчеркиваний.

Над линейкой и под ней нет добавочных интервалов. Если вы хотите отделить ее от окружающего текста, то должны поместить линейку в новый абзац, за которым последует другой абзац, содержащий текст. Обратите внимание на дополнительные интервалы вокруг линейки в следующем примере и на рис. 5.1:

```
This text is directly above the rule.
<hr>
And this text is immediately below.
<p>
Whereas this text will have space before the rule.
<p>
<hr>
<p>
And this text has space after the rule.
```

Тег абзаца за тегом линейки необходим, если вы хотите, чтобы содержимое документа, расположенное под линейкой, было выровнено отличным образом от принятого по умолчанию выравнивания по левому краю.

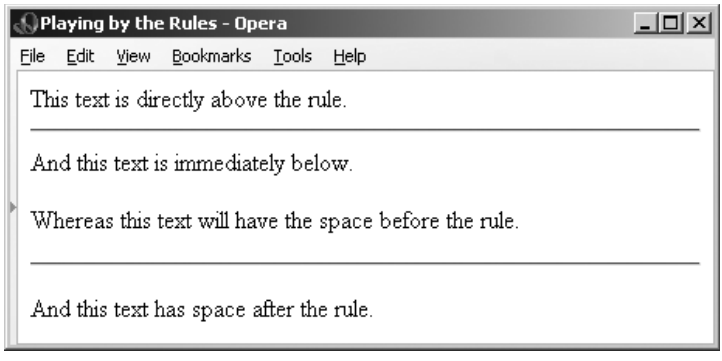

Рис. 5.1. *Теги абзацев обеспечивают текст дополнительным жизненным пространством*

5.1.1.1. Атрибут size

Обычно броузеры изображают горизонтальные линейки в два или три пиксела[1] толщиной, имитируя трехмерный эффект прорези или глубокой царапины на странице. Вы можете сделать линейку толще, используя атрибут `size`. Его значение определяет толщину в пикселах. На рис. 5.2 видно действие этого атрибута, вызванное следующим исходным текстом:

```
<p>
This is conventional document text,
followed by the standard 2-pixel tall rule line.
<hr>
The next three rule lines are 12, 36, and 72 pixels tall, respectively.
<hr size=12>
<hr size=36>
<hr size=72>
```

Атрибут `size` признан нежелательным в HTML 4 и XHTML, поскольку его действие может быть достигнуто при подходящем использовании таблиц стилей.

5.1.1.2. Атрибут noshade

Этот атрибут для тех, кому не нравится трехмерное изображение и кто предпочитает плоскую, или двумерную, линейку. Чтобы убрать эф-

[1] Пиксел – это одна из множества крошечных точек, составляющих экран дисплея вашего компьютера. Хотя размеры дисплеев различаются, есть простое правило, говорящее, что один пиксел равен одному пункту (на мониторе с разрешением 75 точек на дюйм). Пункт – это единица измерения, принятая в полиграфии и равная приблизительно 1/72 дюйма (в дюйме 72,27 пункта, если быть точными). Типичные шрифты, используемые броузерами, имеют 12 пунктов в высоту, а это значит, что на одном дюйме в высоту помещается шесть строк текста.

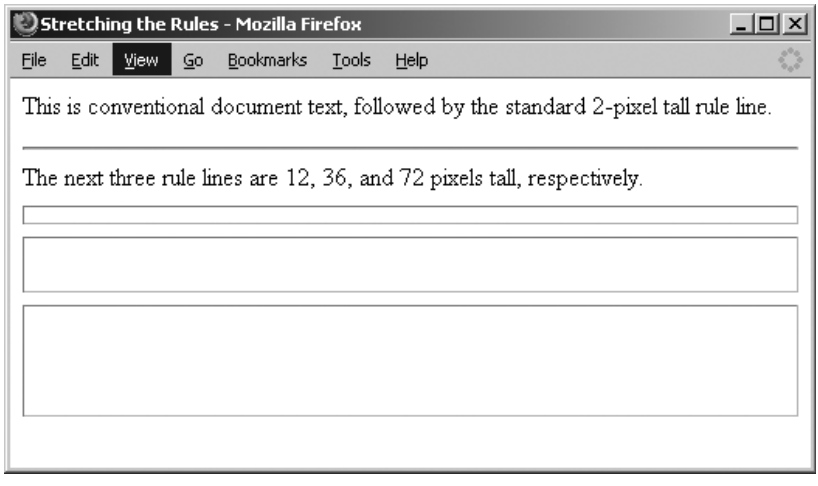

Рис. 5.2. *Популярные броузеры позволяют варьировать толщину горизонтальной линейки*

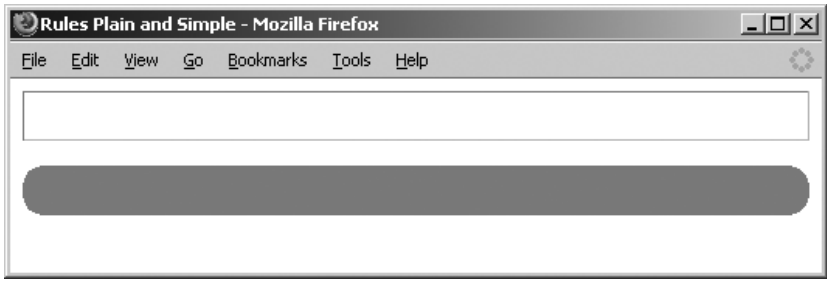

Рис. 5.3. *Обычная трехмерная линейка и ее двумерный вариант noshade*

фект трехмерности, просто добавьте в тег `<hr>` атрибут `noshade`. HTML не требует для него никакого значения. В XHTML используйте `noshade= "noshade"`. Посмотрите (рис. 5.3), чем отличаются «нормальная» трехмерная и плоская `noshade`-линейка. (Как видно из исходного HTML-фрагмента, мы сильно увеличили толщину линеек, чтобы сделать различие более явным.)

```
<hr size=32>
<p>
<hr size=32 noshade>
```

Интересно, что в броузере Internet Explorer линейка `noshade` имела тупые концы, а не скругленные, как в других броузерах (см. рис. 5.3). Как бы то ни было, атрибут `noshade` признан нежелательным в HTML 4 и XHTML, поскольку его действие может быть достигнуто при подходящем использовании таблиц стилей.

5.1.1.3. Атрибут width

По умолчанию линейка проводится во всю ширину окна броузера. Ее можно укорачивать и удлинять с помощью атрибута `width`, проводя линейку шириной в заданное число пикселов или распространяя ее на определенную долю ширины окружающего текста. Большинство броузеров автоматически выравнивают укороченные линейки по центру. Другое выравнивание обеспечивается атрибутом `align` (раздел 4.1.1.1).

Приведем несколько примеров горизонтальных линеек с установленной шириной (рис. 5.4).

```
The following rules are 40 and 320 pixels wide
no matter the actual width of the browser window:
<hr width=40>
<hr width=320>
Whereas these next two rules will always extend across
10 and 75 percent of the window, regardless of its width:
<hr width="10%">
<hr width="75%">
```

Заметьте, что относительное (указанное в процентах) значение атрибута `width` заключено в кавычки, тогда как абсолютное (целое) число пикселов – нет. В действительности кавычки не обязательны в стандарте HTML (хотя и требуются в XHTML). Но поскольку значок процента обычно подразумевает, что за ним следует закодированный символ, пропуск кавычек при записи относительного размера ширины может запутать броузер и испортить вывод части вашего документа.

Вообще говоря, не очень разумно обозначать ширину линейки определенным числом пикселов. Окна броузеров различаются по ширине, и то, что было коротенькой чертой на одном из них, может оказаться неприятно большой на другом. Поэтому мы рекомендуем указывать ширину линейки в процентах от ширины окна. Тогда, если величина окна изменяется, линейка сохраняет свой относительный размер.

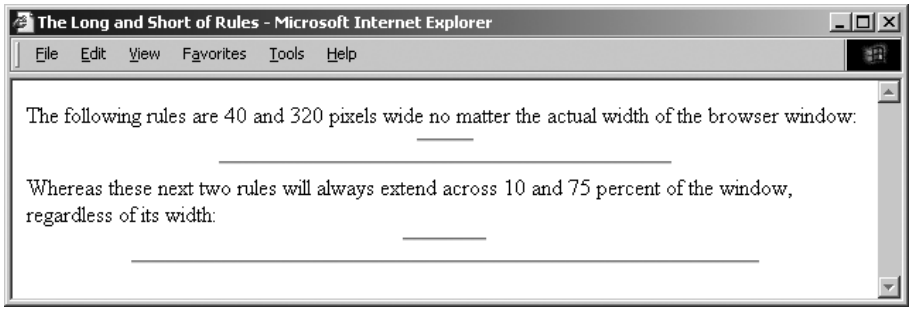

Рис. 5.4. *Линейки длинные и короткие, с абсолютным и относительным указанием ширины*

Атрибут `width` признан нежелательным в HTML 4 и XHTML, поскольку его действие может быть достигнуто при подходящем использовании таблиц стилей.

5.1.1.4. Атрибут align

Атрибут `align` для горизонтальной линейки может принимать одно из трех значений: `left`, `center`, `right`. Линейки, ширина которых меньше ширины окна, будут прижаты к соответствующему краю. По умолчанию линейка центрируется.

Меняющееся выравнивание годится для красивого оформления заголовков. Например, в приведенном ниже тексте линейка шириной в 35% окна перемещается в центр из положения «справа», а затем – налево (рис. 5.5):

```
<hr width="35%" align=right>
<h3>Fruit Packing Advice</h3>
...
<hr width="35%" align=center>
<h3>Shipping Kumquats</h3>
...
<hr width="35%" align=left>
<h3>Juice Processing</h3>
...
```

Атрибут `align` признан нежелательным в HTML 4 и XHTML, поскольку результат его действия может быть достигнут при использовании таблиц стилей.

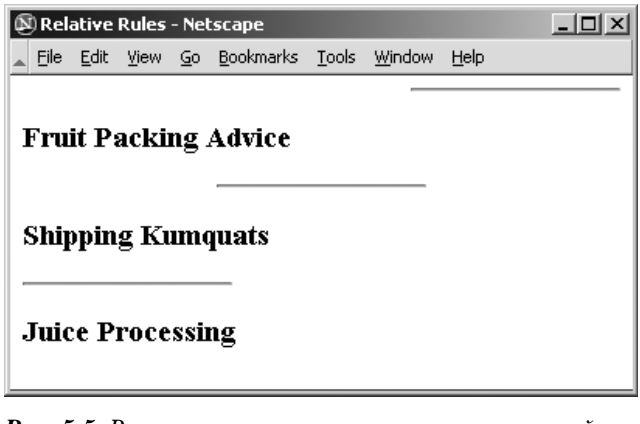

Рис. 5.5. *Различное выравнивание горизонтальной линейки позволяет изящно оформить заголовки*

5.1.1.5. Атрибут color

Поддерживаемый только Internet Explorer и Netscape Navigator версий 7 и 8, но не другими популярными броузерами (например, Opera)

атрибут color позволяет устанавливать цвет линейки. Значение данного атрибута – это или название, или шестнадцатеричный триплет, обозначающий определенный цвет. Полный список названий цветов и их числовых обозначений содержится в приложении G.

По умолчанию линейка отображается тем же цветом, что и фон документа, с более темной или светлой окантовкой. Определяя другой цвет линейки с помощью таблиц стилей или атрибута color, вы теряете эффект трехмерности.

5.1.1.6. Комбинирование атрибутов линейки

Можно комбинировать атрибуты линейки, при этом их порядок не имеет значения. Например, для создания больших прямоугольников скомбинируйте атрибуты size и width:

```
<hr size=32 width="50%" align=center>
```

В действительности некоторые атрибуты необходимо комбинировать – align и width, например. В одиночку атрибут align не произведет никакого действия, поскольку по умолчанию линейка занимает всю ширину окна.

На рис. 5.6 показан результат соединения атрибутов size и width.

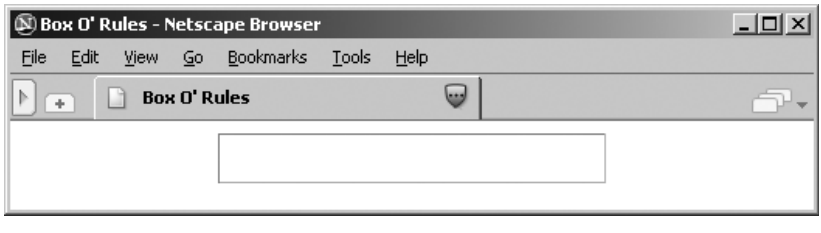

Рис. 5.6. *Комбинирование атрибутов для достижения спецэффектов*

5.1.1.7. Атрибуты class, dir, event, id, lang, style и title

Многие теги содержимого документа обладают рядом почти универсальных атрибутов. Эти атрибуты дают стандартный способ пометить (id) или назвать (title) содержимое тега для дальнейших ссылок или автоматической обработки, изменить характеристики отображения содержимого тега (class, style), указать используемый язык (lang) и связанное с ним направление отображения текста (dir). Разумеется, неизвестно, как будут действовать на горизонтальную линейку атрибуты dir и lang. Тем не менее они являются стандартными атрибутами тега. [атрибут dir, 3.6.1.1] [атрибут lang, 3.6.1.2] [встроенные стили: атрибут style, 8.1.1] [стилевые классы, 8.3] [атрибут id, 4.1.1.4] [атрибут title, 4.1.1.5]

И конечно, существуют все инициированные пользователем события, которые могут происходить как внутри, так и вокруг горизонтальной линейки, которые броузер воспринимает и на которые он может реаги-

ровать при помощи on-атрибутов и некоторого программирования. [обработчики событий JavaScript, 12.3.3]

5.1.2. Использование линеек для разбиения документа

Горизонтальные линейки помогают читателю зрительно ориентироваться в документе. Чтобы эффективно использовать тег <hr> для отделения друг от друга частей документа, прикиньте вначале, сколько в нем будет уровней заголовков и какова приблизительно длина его разделов. Затем решите, какие из заголовков следует отделять линейками.

Горизонтальная линейка может также отмечать завершение вводной части документа, проводя границу, например, между оглавлением и основным текстом. Кроме того, используйте их для отделения основного текста от следующих за ним индексов, библиографических списков, списков иллюстраций.

Опытные авторы применяют горизонтальную линейку и для обозначения начала и окончания формы. Это особенно полезно в случае длинных форм, которые пользователям приходится прокручивать при заполнении не раз и не два. Систематически отмечая ее начало и конец с помощью линеек, вы поможете читателю не выходить за пределы формы, что уменьшит риск непреднамеренного пропуска в заполнении полей.

5.1.3. Горизонтальные линейки в начале и конце документа

Принцип единого оформления семейства документов состоит в придании каждому из них общих стилистических и художественных особенностей. Это касается и входящих в каждый документ вводных и закрывающих частей. Обычно вводная часть (шапка) содержит средства навигации, которые помогают читателю легко перемещаться к внутренним разделам и другим объектам семейства, тогда как в завершающей части (подвале) располагается информация об авторах и о документе вместе с механизмом обратной связи, таким как ссылка на адрес электронной почты веб-мастера.

Рис. 5.7 демонстрирует действие приведенного ниже кода.

Чтобы гарантировать, что шапка и подвал документа не сольются с основным текстом, попробуйте вставить линейки сразу под шапкой и перед подвалом. Вот пример:

```
<body>
Kumquat Growers Handbook - Growing Season Guidelines
<hr>
<h1 align=center>Growing Season Guidelines</h1>
Growing season for the noble fruit varies throughout
North America, as shown in the following map:
<p>
<img src="pics/growing-season.gif">
```

```
<p>
<hr>
<i>Provided as a public service by the
<a href="feedback.html">Kumquat Lovers of America</a></i>
```

Единообразно отделяя шапку и подвал при помощи линеек, вы помогаете пользователю распознать основное содержимое документа и сосредоточиться на нем.

Рис. 5.7. *Четкое выделение шапки и подвала документа при помощи горизонтальных линеек*

5.2. Изображения в документе

Одной из наиболее подкупающих черт HTML и XHTML является предоставляемая ими возможность включать в текстовый документ изображения либо в виде встроенных компонентов (внедренных изображений), либо отдельных документов, которые могут быть загружены с помощью соответствующей гиперссылки, либо в виде фона документа. Изображения – статические и анимированные пиктограммы, картинки, иллюстрации, рисунки – могут при умеренном их употреблении сделать документ более привлекательным, зазывающим и профессиональным, так же как и более информативным и легким для изучения. Помимо этого существуют способы включения изображения

в качестве карты гиперссылок. Однако чрезмерное количество изображений загромождает документ, делает его запутанным и труднодоступным, а заодно увеличивает время, которое необходимо пользователю для загрузки и просмотра ваших страниц.

5.2.1. Форматы изображений

Ни HTML, ни XHTML не дают официальных предписаний для формата изображений. Однако популярные броузеры особенно хорошо приспособлены для работы с определенными форматами, в частности GIF, PNG и JPEG (объяснения в следующих разделах). Большинство других мультимедийных форматов требуют, чтобы пользователь приобрел, установил на своем компьютере и научился успешно применять специальные вспомогательные программы. Поэтому неудивительно, что GIF, PNG и JPEG *de facto являются стандартами для изображений в сети*.

Оба эти формата получили широкое распространение еще до появления сети, поэтому существует множество программ, которые помогут вам создать графический файл в одном из них. Тем не менее у каждого формата есть свои достоинства и недостатки, так же как и особенности, эксплуатируемые некоторыми броузерами для достижения специальных эффектов при отображении.

5.2.1.1. Формат GIF

Формат обмена графикой (Graphics Interchange Format, GIF) был изначально разработан для передачи изображений между пользователями онлайновой службы CompuServe. У этого формата обнаружилось несколько черт, сделавших его популярным при употреблении в HTML- и XHTML-документах. Его кодировка является платформно-независимой, так что, используя подходящую программу-декодер (встроенную в большинство броузеров), вы можете, например, без особых хлопот загрузить и просматривать на персональном компьютере с Windows изображение, созданное и превращенное в GIF-файл на Macintosh. Вторая главная черта GIF – это применяемая при создании файла особая технология сжатия, которая существенно сокращает его размер, что позволяет передавать изображение по сети быстрее. Сжатие в GIF производится «без потерь» – никакие данные исходного образца не удаляются и не изменяются, так что в распакованном виде изображение точно совпадет с оригиналом. Кроме того, GIF-изображения легко анимировать.

Хотя все файлы с GIF-изображениями неизменно обозначаются именами с расширением *.gif* (или *.GIF*), существует две версии GIF – оригинальная GIF87 и расширенная GIF89a, которая поддерживает несколько новых возможностей, распространенных у сетевых авторов (раздел 5.2.1.2), включая прозрачный фон, чересстрочное хранение и анимацию. Популярные броузеры в настоящее время поддерживают обе

версии GIF, использующие одинаковую схему кодировки, которая представляет собой карту значений цвета пиксела. Сами коды берутся из встроенной в файл восьмибитной таблицы (палитры), что дает не более 256 различных цветов в изображении. В большинстве случаев употребляется меньше цветов: для этого применяются специальные средства, упрощающие слишком богатую палитру графики. При упрощении GIF-изображений создается палитра меньшего размера и увеличивается избыточность данных в попиксельном представлении, что влечет за собой лучшее сжатие файла и, следовательно, более быструю его загрузку.

Тем не менее ограниченный размер палитры делает GIF-формат непригодным для фотореалистических картинок (см. раздел 5.2.1.3). GIF идеально подходит для создания пиктограмм, цветных картинок с ограниченным набором цветов и рисунков.

Поскольку в большинство графических броузеров встроена поддержка данного формата, такое кодирование изображений является самым распространенным в сети. Оно подходит как для внедренных в документ изображений, так и для внешних, подключенных по гиперссылке. При любых сомнениях используйте GIF.[1] Это сработает почти всегда.

5.2.1.2. Всплывание, прозрачность, анимация

Изображения в формате GIF можно использовать для достижения трех особых эффектов: всплывания (другое название этого явления – чересстрочность), прозрачности и анимации. Всплывающее изображение на глазах материализуется на экране дисплея, тогда как обычное растекается по нему сверху вниз. Обычное изображение в GIF-кодировке представляет собой последовательность пикселов, хранящихся строка за строкой сверху вниз. Тогда как обычное GIF-изображение выводится на экран, словно опускающийся занавес, его всплывающая разновидность открывается, как жалюзи. Дело в том, что при всплывании воспроизводятся не все подряд, а только каждая четвертая строки (сначала первые, кратные четырем, строки, на втором этапе – третьи, и т. д.) оригинального GIF-изображения. Читатель просматривает образ целиком – сверху донизу, хотя и в нечетком виде – за четверть времени, необходимого для того, чтобы загрузить и вывести на экран весь объект. Полученное изображение, прорисованное на четверть, обычно достаточно ясно, чтобы пользователь с медленным сетевым соединением мог определиться, стоит ли ему ждать, пока загрузится остаток файла.

Не все графические броузеры, даже если они умеют отображать чересстрочные GIF-изображения, способны достигать при этом эффекта постепенной материализации картинки. На тех броузерах, которые это

[1] Мы не можем удержаться от искушения заметить, что разборчивые авторы выбирают GIF.

производят, эффект детализации может быть подавлен пользователем, указавшим в настройках броузера, что следует ждать, пока изображение полностью загрузится и будет расшифровано, и только тогда выводить его на экран. С другой стороны, старые броузеры всегда загружают и декодируют изображение перед тем, как начать его воспроизводить, и, следовательно, не поддерживают средств чересстрочного вывода.

Другой популярный эффект, достижимый с помощью GIF-изображений (точнее, изображений в формате GIF89a), – это возможность сделать некоторую их часть прозрачной, так что подкладка, находящаяся внизу (обычно фон окна броузера), будет сквозь нее проступать. Прозрачное GIF-изображение выделяет один из цветов в своей палитре, назначая его для окраски фона. Броузер просто игнорирует всякий пиксел, употребляющий этот цвет, позволяя таким образом фону окна проступать сквозь рисунок. Тщательно подрезав края изображения и выбрав сплошной монотонный фон, можно достичь впечатления, что картинка вписывается в страницу без скачков и видимых швов или плывет над ней.

Прозрачные GIF-изображения очень подходят для любой графики, которую хотелось бы видеть слитой воедино с документом, а не лежащей на его поверхности в виде прямоугольного блока. Очень популярны прозрачные логотипы, а также прозрачные пиктограммы и графические метки, – как, впрочем, и любая графика, которая должна иметь естественные очертания. Кроме того, можно вставлять сквозные изображения в строку с традиционным текстом для его украшения.

Недостатком прозрачных изображений может быть их ужасный вид, когда они окружены прямоугольной рамкой, как, например, бывает при включении их в якорь гиперссылки (тег <a>), если рамка не удалена явным предписанием. Поток текста также будет «обтекать» прямоугольные границы рисунка, а не его видимые очертания. Результат может выглядеть как «очень одинокое изображение», висящее в пустоте.

Третья замечательная «примочка», реализуемая с помощью изображений в формате GIF89a, – это простая мультипликация. Используя специальную утилиту для GIF-анимации, можно приготовить отдельный GIF89a-файл, содержащий последовательность картинок. Броузер отображает картинки, хранящиеся в файле, одну за другой, создавая эффект, подобный возникающему при быстром перелистывании анимационного буклета, какой у нас был или который мы даже рисовали сами в детстве. Специальные управляющие сегменты между отдельными картинками в GIF-файле позволяют устанавливать число прокруток всей последовательности, продолжительность паузы перед появлением очередного образа, указывать, нужно ли стирать предыдущую картинку (обнажая фон) перед появлением следующей, и тому подобное. Комбинируя эти средства управления с теми, что применяются к обычным GIF-изображениям, включая индивидуальные палит-

ры, прозрачность и эффект всплывания, можно создавать очень привлекательные и сложные мультики.[1]

Сила простой GIF-анимации состоит в том, что для достижения требуемого эффекта не нужно специально программировать HTML-документ. Есть, впрочем, одно важное обстоятельство, ограничивающее использование этого средства, делающее его пригодным только для маленьких, размером с пиктограмму, форматов или для узких полосок в окне броузера. Размер файлов GIF-анимации очень быстро растет, даже если вы тщательно избегаете повторения статического содержимого картинок в последовательных кадрах. И если в документе присутствует несколько анимашек, затраты времени для их загрузки могут – и обычно будут – раздражать пользователя. Если в чем и нужно соблюдать умеренность, так это в употреблении GIF-анимации.

Никакой из GIF-эффектов – ни всплывание, ни прозрачность, ни анимация – не возникнет сам собой. Вам понадобится специальное программное обеспечение для изготовления GIF-файла. В наши дни многие программы, работающие с изображениями, сохраняют последние в формате GIF, и большинство из них позволяют получить прозрачные и чересстрочные файлы. Существует также уйма бесплатных программ, специализирующихся на этих задачах, так же как и на создании анимационных файлов. Загляните в ваши любимые архивы программ в Интернете, и вы найдете там средства для создания и преобразования GIF-графики. В главе 17 есть раздел, посвященный созданию прозрачных изображений.

5.2.1.3. Формат JPEG

Joint Photographic Experts Group (JPEG) – это организация по стандартизации, разработавшая то, что теперь называется JPEG-форматом кодировки изображений. Подобно GIF-форматам, JPEG-изображения являются платформно-независимыми и специально сжатыми для скоростной передачи посредством цифровых коммуникационных технологий. В отличие от GIF, формат JPEG поддерживает десятки тысяч цветов для отображения более детальных, фотореалистических изображений. При этом JPEG использует специальные алгоритмы, обеспечивающие значительно больший коэффициент сжатия. Нередко удается, например, 200-килобайтное GIF-изображение сократить до 30 Кбайт, применяя JPEG. Достигая такого впечатляющего сжатия, JPEG теряет часть содержащихся в изображении данных. Впрочем, можно управлять уровнем потерь при сжатии, применяя специальные JPEG-средства, так что, хотя восстановленное изображение, скорее

[1] Songline Studios опубликовал целую книгу, посвященную GIF-анимации, «GIF Animation Studio» («Руководство по GIF-анимации»), написанную Ричардом Команом (Richard Koman).

всего, и не совпадает с оригиналом в точности, оно будет настолько на него похоже, что большинство людей не заметит разницы.

Хотя для фотографий JPEG подходит великолепно, он не особенно хорош для иллюстраций. Алгоритмы, используемые при сжатии и восстановлении изображений, оставляют заметные следы на больших пространствах, заполненных одним цветом. Следовательно, если вы хотите отобразить рисунок, формат GIF может оказаться предпочтительнее.

Формат JPEG, обычно обозначаемый расширением *.jpg* (или *.JPG*), распознается почти всеми современными графическими броузерами. Изредка встречаются старые броузеры, которые не могут сами воспроизводить JPEG-изображения.

5.2.1.4. Формат PNG

Технология PNG (Portable Network Graphics, переносимая сетевая графика) была создана в качестве замены GIF, но не из-за того, что GIF плохо справлялся со своими обязанностями. В действительности GIF был и остается самым востребованным форматом в Интернете. Многие пользователи были возмущены, когда в 1993 году, после того как GIF приобрел популярность и широкое распространение, компания Unisys решила зарегистрировать свои авторские права и собирать дань за использование технологии сжатия GIF. Эта акция противоречила философии свободного обмена информацией, которой придерживалось тогдашнее сообщество пользователей Интернета, состоявшее, в основном, из научных работников. Она подтолкнула неформальную группу, возглавляемую Томасом Бутеллом (Thomas Boutell), к разработке альтернативы, а именно PNG.

Преимущества технологии PNG над GIF и JPEG (помимо того, что она предоставляет формат, пользователи которого не рискуют оказаться ответчиками в суде) заключаются в более широком выборе цветовых форматов (24-битовый RGB, шкала оттенков серого и 8-битовая GIF-подобная палитра) и в меньших потерях при сжатии. Уникальными и весьма привлекательными чертами технологии PNG являются альфа-каналы, позволяющие указывать огромное количество слоев прозрачности (более 65 тыс. против одного слоя в GIF) и способные симулировать трехмерные изображения; гамма-коррекция, управляющая яркостью при переносе с одной платформы на другую и обеспечивающая более четкую графику; а также двухмерное чередование строк, обеспечивающее более точное постепенное проявление картинки.

PNG не поддерживает анимацию. Возможно, этот факт заставит кого-то усомниться в целесообразности перехода на PNG, но мы все-таки советуем вам попробовать эту технологию, особенно ради многоцветных и высококачественных изображений.

5.2.2. Когда следует использовать изображения

Одна картинка может стоить тысячи слов, но не забывайте, что никто не слушает болтунов. Во-первых, и это главное, смотрите на изображения в вашем документе как на средства представления визуальной информации, а не как на приманку для неопытных веб-серферов. Изображения должны помогать тексту легче доходить до читателей, а читателям – легче ориентироваться в документе. Используйте изображения, чтобы пояснить, проиллюстрировать материал и обеспечить его примерами. Сопровождающие текст фотографии, графики, диаграммы, карты и рисунки – это естественные и приемлемые кандидаты. Фотографии товаров – это неотъемлемая часть каталогов и путеводителей по товарам в интернет-магазинах. Пиктограммы в якорях гиперссылок, снабженные анимацией, могут служить эффективным навигационным инструментом, облегчающим пользователю доступ к внутренним и внешним ресурсам. Если изображение не выполняет ни одной из этих достойных функций, выбросьте его вон из своего документа!

Одно из важнейших обстоятельств, которые необходимо учитывать при добавлении изображений в документ, – это привносимая ими добавочная задержка при получении документа из сети, особенно при модемном соединении. Тогда как обычный текстовый документ может иметь объем не более 10–15 Кбайт, любому изображению понадобятся сотни килобайт. Кроме того, общее время получения документа состоит не только из суммы времени, потраченного на каждую из его частей, но включает в себя все дополнительные задержки, происходящие при передаче данных по сетям.

В зависимости от скорости соединения (обычно измеряемой в битах или байтах в секунду), а также от уровня загрузки (перегрузки) сети, влияющего на задержки при передаче данных, один документ, содержащий 100-килобайтное изображение, может потребовать менее секунды при кабельном соединении в ранние утренние часы, когда все спят, до более чем десяти минут при подключении через сотовый телефон в полдень. Получили картинку?

Разумеется, можно сказать, что рисунки и прочие мультимедийные «излишества» подталкивают интернет-провайдеров стремиться ко все более быстрому, лучшему, надежному обеспечению доставки сетевых материалов. В ближайшем будущем модемные соединения последуют за почтовыми каретами, уступив место кабельным модемам и ADSL.[1]

Но цены падают, число потребителей растет – вот и источник задержек. И не будем забывать о броузерах в сотовых телефонах, а также о том, что в странах третьего мира связь ненадежная и медленная. Кроме то-

[1] ADSL (Asymmetric Digital Subscriber Line) – асимметричная цифровая абонентская линия. Технология, позволяющая вести прием/передачу цифрового сигнала в диапазоне от 1 до 8 Мбит/с. – *Примеч. науч. ред.*

го, когда вы пытаетесь получить доступ к перегруженному серверу, не имеет значения, с какой скоростью могли бы передаваться данные.

5.2.3. Когда следует использовать текст

Текст еще не вышел из моды. Для некоторых пользователей это единственная составляющая вашего документа, которая им доступна. Мы настаиваем, что в большинстве случаев документы должны быть пригодны для употребления теми пользователями, которые не могут просматривать изображения или отключили их автоматическую загрузку, чтобы ускорить воспроизведение страниц. Хотя велико искушение вставлять изображения всюду, где только возможно, случается, что чисто текстовый документ бывает гораздо полезней.

Документы, преобразуемые в сетевые из других форматов, редко включают изображения. Обзорные материалы и другое серьезное содержимое часто вполне годится для употребления в чисто текстовой форме.

Следует создавать чисто текстовые документы, когда скорость доступа критически важна. Если известно, что ваши страницы будут нужны сразу множеству пользователей, то следует позаботиться о них, избегая размещения изображений в документах. В особых, самых критических случаях вы можете создать главную (начальную) страницу, которая позволит читателям выбирать между двумя исполнениями вашего собрания документов, одно из которых содержит рисунки, тогда как другое от них освобождено. (Популярные броузеры обладают специальными пиктограммами, отмечающими место, где могли бы находиться (или вот-вот появятся) загружаемые изображения, которые могут замусорить и испортить макет документа, превратив его в кашу.)

Текст – даже с изображениями, но без всяких пустых украшений или бессмысленной графики – подходит лучше всего, если вы хотите, чтобы документ было удобно обрабатывать поисковым службам, во множестве работающим в сети. Изображения почти всегда игнорируются этими поисковыми средствами. Если главное содержимое страниц составляют рисунки, очень мало информации об этих документах попадет в онлайновые рубрикаторы (сборники ссылок на разнообразные ресурсы сети, отсортированные по темам).

5.2.4. Ускорение загрузки изображений

Помимо главного совета быть сдержанным и разборчивым при использовании изображений, приведем еще несколько рекомендаций, как снизить издержки и задержки, возникающие при загрузке графики.

Будьте проще

Полноэкранная 24-битовая цветная графика, даже уменьшенная в размере с применением сжатия, осуществляемого при преобразовании в один из стандартных форматов (GIF, PNG или JPEG), все же остается пожирателем сетевых ресурсов. Приобретайте и исполь-

зуйте разнообразные средства управления изображениями, позволяющие оптимизировать изображение под конкретное применение, особенно если вы планируете специальные эффекты вроде анимации GIF или трехмерного эффекта PNG. Упрощайте свои рисунки. Не используйте панорамных снимков. Избегайте картинок с обширным и пустым фоном так же, как вычурных рамочек и других занимающих много места элементов. Избегайте также сглаживания (смешивания двух цветов соседних пикселов для получения третьего цвета). Эта техника препятствует сжимаемости ваших изображений. Предпочитайте большие области, покрытые одним цветом, которые легко поддаются сжатию.

Используйте изображения повторно

Это особенно справедливо для пиктограмм и GIF-анимации. Большинство броузеров кэширует компоненты поступающих к ним документов, сохраняя их локальные копии именно для того, чтобы быстро, не обращаясь к сети, восстанавливать данные. Для уменьшения размеров файлов GIF-анимации попробуйте подготовить ее так, чтобы каждое последующее изображение обновляло только ту часть картинки, которая должна измениться, а не весь кадр (это ускорит и смену кадров).

Разделяйте на части большие документы

Это общее правило касается и изображений. Множество небольших сегментов, связанных между собой гиперссылками (конечно же!), снабженное оглавлением, воспринимается пользователями лучше, чем небольшое число документов большего размера. Как правило, люди скорее предпочтут «перелистывать» страницы, чем сидеть и ждать, пока загрузится большой документ. (Это сродни синдрому перескакивания с одного телевизионного канала на другой.) Есть хорошее правило: не создавайте документы размером больше 50 Кбайт, тогда даже самое медленное соединение не будет травмировать ваших читателей.

Изолируйте изображения, размер которых нельзя уменьшить

Вставляйте в документ гиперссылки на большие изображения, возможно в виде их уменьшенных копий, давая таким образом читателю право решить, хочет ли он, а если хочет, то когда, потратить время на его полную загрузку. И поскольку загружаемое изображение не замешано в гущу других элементов документа, как это случается со встроенными, читателю гораздо легче пометить его и сохранить локально для последующего изучения. (Детальное обсуждение загрузки внешних изображений содержится в разделе 5.6.2.)

Указывайте размеры изображений

Наконец, еще один способ улучшить производительность состоит во включении информации о высоте и ширине изображения в его тег. Доставив эту информацию броузеру, вы избавите его от необхо-

димости совершать дополнительные действия по определению пространства, которое следует отвести под картинку, что позволит ему быстрее вывести страницу. Этот подход имеет и обратную сторону, которую мы исследуем в разделе 5.2.6.12.

5.2.5. JPEG, PNG или GIF?

Вы можете выбрать только один формат изображения для своих HTML-документов, если ваш источник изображений или программное обеспечение подталкивают к этому. Все три формата практически универсально поддерживаются современными броузерами, так что со стороны пользователей это не создаст проблем.

Тем не менее мы рекомендуем приобрести средства для создания и преобразования изображений во всех трех форматах, чтобы использовать преимущества каждого из них. Например, применяйте способность GIF и PNG создавать прозрачные объекты при изготовлении пиктограмм. Напротив, используйте глубокое сжатие JPEG, пусть и за счет целостности, для быстрой загрузки больших полноцветных изображений.

5.2.6. Тег <img>

Тег `<img>` позволяет вставить в текстовый поток документа графическое изображение, на которое он ссылается. Ни до, ни после тега `<img>` не требуется обязательного присутствия конца абзаца или перехода на новую строку, так что изображение может быть помещено буквально «в линию» с текстом и другим содержимым документа.

Сам формат изображения стандартами HTML и XHTML не определяется, хотя популярные графические броузеры поддерживают GIF-, PNG- и JPEG-расширения. Стандарты также не задают и не ограничивают их размеры. Изображения могут быть сколь угодно многоцветными, но то, как эти цвета будут воспроизводиться, сильно зависит от броузера.

Функция:	Вставляет в документ изображение
Атрибуты:	`align`, `alt`, `border`, `class`, `controls` [!], `dir`, `dynsrc` [!], `height`, `hspace`, `id`, `ismap`, `lang`, `longdesc` [!], `loop` [!], `lowsrc` [!], `name` [!], `onAbort`, `onClick`, `onDblClick`, `onError`, `onKeyDown`, `onKeyPress`, `onKeyUp`, `onLoad`, `onMouseDown`, `onMouseMove`, `onMouseOut`, `onMouseOver`, `onMouseUp`, `src`, `start` [!], `style`, `title`, `usemap`, `vspace`, `width`
Закрывающий тег:	Отсутствует в HTML; `</img>` или `<img ... />` в XHTML
Содержит:	Ничего
Может содержаться в:	*тексте*

В целом, представление изображений во многом определяется броузером. Неграфические броузеры их могут просто игнорировать. Броузеры, работающие в окружении, налагающем некоторые ограничения, способны, в свою очередь, изменять размеры или уровень сложности изображений. И пользователи, особенно те из них, кто располагает медленным соединением с сетью, могут отключать их загрузку. Соответственно вам следует убедиться, что документы сохраняют свой смысл и остаются полезными, даже если изображения полностью из них удалены.

В языке HTML у тега `<img>` нет закрывающего тега. В XHTML-документе требуется либо косая, закрывающая тег перед знаком `">"` (например, `<img src="kumquat.gif" />`), либо закрывающий тег `</img>` сразу после открывающего и его атрибутов.

5.2.6.1. Атрибут src

Атрибут `src` обязателен для тега `<img>` (если только не используется атрибут `dynsrc` для видео, которое поддерживает Internet Explorer, см. раздел 5.2.7.1). Значением этого атрибута является URL содержащего изображение файла, абсолютный или указанный относительно ссылающегося на него документа. Чтобы не загромождать хранилище документов, авторы, как правило, собирают такие файлы в отдельной папке, именуемой часто «pics» или «images». [ссылки на документы: URL, 6.2]

К примеру, следующий HTML-фрагмент помещает изображение знаменитой фабрики по расфасовыванию кумкватов в повествовательный текст (рис. 5.8):

```
Here we are, on day 17 of the tour, in
the kumquat packing plant:
<p>
<img src="pics/packing_plant.gif">
<p>
What an exciting moment, to see the boxes of fruit piled high to
```

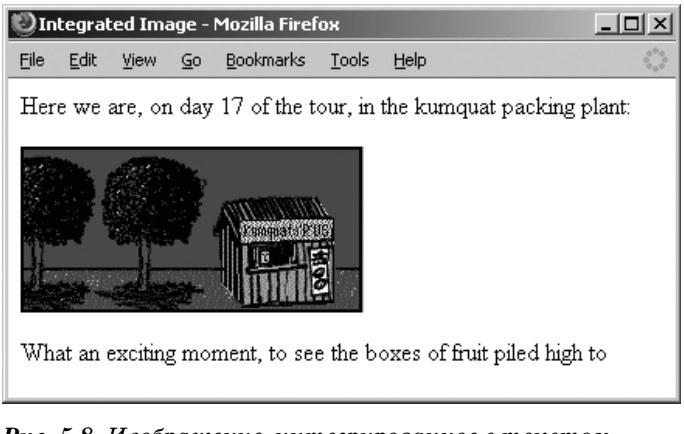

***Рис. 5.8.** Изображение, интегрированное с текстом*

В этом примере теги абзаца (<p>), окружающие тег <img>, заставляют броузер вывести изображение отдельно, на некотором расстоянии от предшествующего и последующего текста. Текст способен также обтекать изображение, как это описано в разделе 4.1.1.1.

5.2.6.2. Атрибут lowsrc

Заботясь о пользователях, особенно о тех, кто располагает медленным подключением к сети, ранние версии Netscape поддерживали атрибут lowsrc, сотоварища src для тега <img>, в качестве средства для ускорения вывода документа. Значение атрибута lowsrc, как и src, – это URL файла изображения, которое броузер Netscape старой версии загружает и воспроизводит, когда впервые встречает тег <img>. Когда этот документ уже полностью загружен и доступен для чтения, Netscape получает изображение, определяемое в атрибуте src.

Ни один броузер, кроме Netscape версии 4 и более ранних, не поддерживает атрибут lowsrc. Начиная с версии 6, Netscape просто использует размеры изображения, указанного в атрибуте lowsrc, для временного выделения места на экране под картинку, пока выводится документ. В старых версиях Netscape размеры lowsrc были нужны для масштабирования окончательного изображения, и эту особенность можно было эксплуатировать для достижения специальных эффектов. Но это уже в прошлом. Мы рекомендуем вам не применять это расширение Netscape и определять пространство на экране явно, с помощью атрибутов height и width, описанных далее в данной главе.

5.2.6.3. Атрибуты alt и longdesc

Атрибут alt определяет альтернативный текст, который броузер может показать, если вывод изображения невозможен или заблокирован пользователем. Популярные броузеры, между прочим, пользующиеся высокой оценкой у инвалидов по зрению, позволяют нам выводить наряду с изображением текст, задаваемый атрибутом alt. Поэтому, хотя данный атрибут и не является обязательным, мы настоятельно рекомендуем применять его к большинству изображений в вашем документе. Тогда, если изображение оказалось недоступным, читатель все же получит некоторое представление о том, что именно пропало. А пользователям с нарушениями зрения атрибут alt часто предоставляет единственную возможность воспринять выведенное изображение.

Кроме того, Internet Explorer выводит текст, помещенный в атрибуте alt, в виде всплывающей подсказки, когда на него попадает курсор мыши. Поэтому можно вставить короткое замечание, которое будет всплывать, как только пользователь укажет курсором на маленькую встроенную в текст пиктограмму (рис. 5.9).

Значение атрибута alt – это строка текста длиной до 1024 символов, с учетом пробелов и знаков пунктуации. Строка должна быть заключена в кавычки. Альтернативный текст может содержать коды специаль-

Рис. 5.9. *Internet Explorer выводит альтернативный текст во всплывающей подсказке*

ных символов, но не должен содержать никакой разметки, в частности, никаких тегов стиля.

Графические броузеры обычно не выводят атрибут `alt`, если изображение доступно и настройки разрешают его загрузку. В противном случае броузеры вставляют текст атрибута как метку сразу вслед за пиктограммой, замещающей изображение. Таким образом, хорошо подобранные метки в `alt` оказывают дополнительную поддержку пользователям графических броузеров, которые отключили автоматическую загрузку изображений вследствие низкой скорости их соединения с сетью.

Неграфические, чисто текстовые броузеры, такие как Lynx, например, помещают альтернативное сообщение прямо в содержимое потока, подобно любому другому текстовому элементу. Так что, используемый эффективно, `alt` может иногда быть заменой отсутствующих изображений. (Пользователи чисто текстовых броузеров будут благодарны за то, что им хотя бы здесь не напоминают, что они второсортные граждане сети.) Например, рассмотрите употребление звездочки в качестве альтернативы графическому маркеру:

```
<h3><img src="pics/fancy_bullet.gif alt"="*">Предисловие</h3>
```

Графический броузер отобразит пиктограмму маркера, тогда как неграфический поставит на ее место звездочку. Подобным образом используйте альтернативный текст для замещения графических маркеров элементов списка. К примеру, следующий код:

```
<ul>
  <li> Рецепты блюд из кумкватов <img src="pics/new.gif"
       alt="(New!)">
  <li> Сроки сбора урожая
</ul>
```

выводит изображение *new.gif* в графических броузерах и сообщение «(New!)» – в текстовых.

Атрибут `alt` может содержать даже более сложный текст (рис. 5.10):

```
Here we are, on day 17 of the tour, in the kumquat
```

```
packing plant:
<p>
<img src="pics/packing_plant.gif"
  alt="[Image of our tour group outside the main packing
  plant]">
<p>
What an exciting moment, to see the boxes of fruit moving
```

```
Here we are, on day 17 of the tour, in front of the kumquat packing
plant:

[Image of our tour group outside the main packing plant]

What an exciting moment, to see the boxes of fruit moving

Commands: Use arrow keys to move, '?' for help, 'q' to quit, '<-' to go back
```

Рис. 5.10. *Чисто текстовые броузеры, такие как Lynx, вместо изображения выводят альтернативный текст*

Согласно спецификации HTML 4.01, атрибут `alt` необходим во всех тегах `<img>`. Чтобы ваши документы строго соответствовали стандарту, сопровождайте все изображения пустыми атрибутами `alt` (т. е. пишите `alt=""`).

Атрибут `longdesc` подобен атрибуту `alt`, но позволяет более длинные описания. Значение `longdesc` – это URL документа, содержащего описание изображения. Если оно длиннее 1024 символов, используйте `longdesc` для ссылки на него. Ни HTML, ни XHTML не определяют, из чего должно состоять описание, не знает этого и любой броузер, поддерживающий в настоящее время `longdesc`. Ничто не поможет, когда придется решать, как составлять длинное описание.

5.2.6.4. Атрибут align

Для изображений стандарты не определяют принятого по умолчанию выравнивания по отношению к тексту и другим изображениям в той же строке – не всегда можно предсказать, как эти компоненты будут выглядеть.[1] В HTML-документах изображения обычно расположены «в линию» только с одной строкой текста. Как принято в печатных средствах массовой информации текст их «обтекает», так что с изображением могут соседствовать несколько строк, а вовсе не одна.

---

[1] Большинство популярных графических броузеров вставляют изображения, выравнивая их основание с опорной линией текста. К такому же эффекту приводит значение `bottom` данного атрибута. Но дизайнерам документов следует иметь в виду, что это выравнивание зависит от броузера, поэтому лучше всегда явно определять положение изображений.

К счастью, создатели документов способны отчасти управлять выравниванием изображений по отношению к окружающему тексту с помощью атрибута `align` тега `<img>`. Стандарты HTML и XHTML определяют пять значений для атрибута `align`: `left`, `right`, `top`, `middle`, `bottom`. Значения `left` и `right` заставляют текст обтекать изображение, смещенное к соответствующему краю. Оставшиеся три выравнивают изображение в вертикальном направлении по отношению к окружающему тексту.

Все популярные броузеры, включая Opera, Firefox, Netscape и Internet Explorer, соглашаются с тем, что атрибут `align=bottom` является умолчанием вертикального выравнивания, и одинаково располагают картинки выше самого верхнего символа в строке текста, см. рис. 15.11.

В то же время броузеры расходятся во мнении, как располагать относительно текста изображения с атрибутом `align=middle`. На рис. 5.11 показано, что Netscape и Opera помещают изображение, ориентируясь на среднюю линию текста. Что касается броузеров Internet Explorer

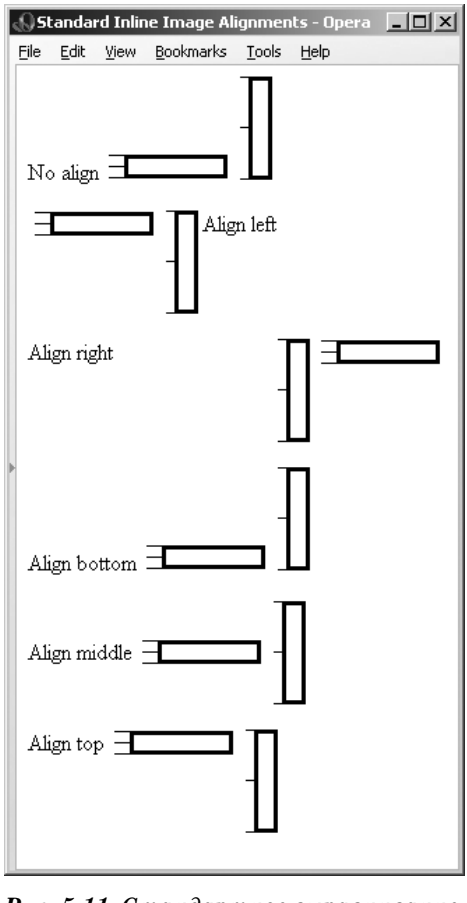

Рис. 5.11. *Стандартное выравнивание внедренных изображений*

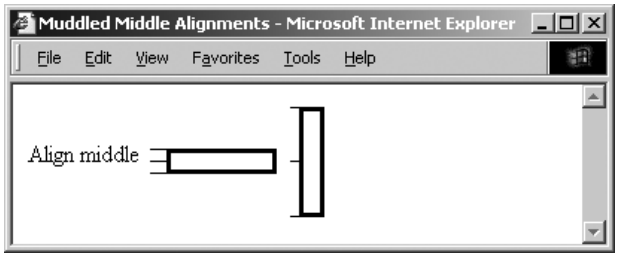

Рис. 5.12. *Internet Explorer и Firefox совмещают среднюю линию изображения со средней линией самого высокого элемента, а не средней линией текста*

и Firefox, они располагают изображение по средней линии самого высокого элемента, который не обязательно является текстовым (рис. 5.12).

Кроме того, броузеры в той или иной степени поддерживают пять дополнительных признаков вертикального выравнивания изображений: `texttop`, `center`, `absmiddle`, `baseline` и `absbottom` (кто не догадался об их смысле, поднимите руку):

`texttop`

Выражение `align=texttop` приказывает броузеру выровнять верхний край изображения с верхним краем самого высокого элемента текста в текущей строке. Иначе действует значение `top`, которое выравнивает верхний край изображения с верхним краем самого высокого элемента, текстового компонента или изображения в текущей строке. Если строка не содержит других изображений, выступающих над текстом, `texttop` и `top` имеют одинаковый эффект. Opera не поддерживает `texttop`, но другие популярные броузеры интерпретируют его строго в соответствии с этим описанием.

`center`

Впервые появившись в броузере Internet Explorer и Netscape, значение `center` интерпретируется в броузерах Internet Explorer, Netscape и Firefox точно так же, как и значение `middle`. Только не будем забывать, что значение `middle` каждый из них понимает по-своему. Opera игнорирует выражение `align=center`.

`absmiddle`

Если вы присвоите атрибуту `align` тега `<img>` значение `absmiddle`, броузер выставит абсолютную середину изображения против абсолютной середины текущей строки. Не так действуют значения `middle` и `center`, которые выравнивают середину изображения с базовой линией текущей строки текста. В то время как Netscape и Opera не проводят разницы между `absmiddle` и `middle`, Firefox и Internet Explorer используют эти значения для избирательного выравнивания изображений по средним линиям. Иными словами, Firefox и Internet Explorer воспринимают признак `absmiddle` так же, как Netscape воспринимает `middle`.

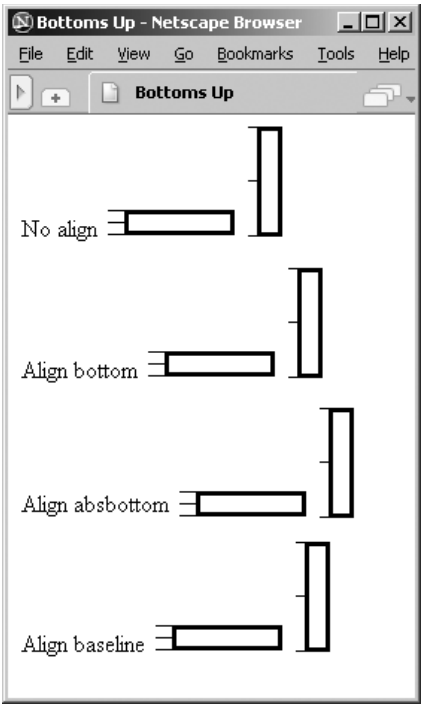

Рис. 5.13. *При выравнивании изображений с атрибутом alingn=absbottom броузеры принимают во внимание нижние выносные элементы*

`bottom` *и* `baseline` **(принимаются по умолчанию)**

Значения `bottom` и `baseline` имеют тот же эффект, как если бы вы не включили атрибут выравнивания совсем. Броузер выравнивает нижний край изображения с базовой линией текста. Не следует путать это с действием значения `absbottom`, которое учитывает и те части букв, что спускаются *ниже базовой линии*.[1] (Если вы все еще держите руку поднятой, можете ее опустить.)

`absbottom`

Выражение `align=absbottom` предлагает броузеру выровнять нижний край изображения с истинным нижним краем текста в текущей строке. Истинный нижний край – это самая нижняя точка текста с учетом всех нижних выносных элементов (например, нижняя часть символа «y»), даже если таковых в строке нет. Базовая линия проходит по нижнему краю «v» в символе «y». Opera, носитель стандарта, игнорирует значение `absbottom`, а другие популярные броузеры обращаются с ним согласно описанию (рис. 5.13).

[1] *Нижний выносной элемент* (*descender*) иначе называется подстрочным элементом литеры. – *Примеч. науч. ред.*

Используйте `top` и `middle` для совмещения пиктограмм, маркеров списков и других специальных изображений с близлежащим текстом. В других случаях `align=bottom` (выбирается по умолчанию) достигает лучшего внешнего вида. Выравнивая одно или несколько изображений в строке, выбирайте то значение атрибута, которое позволит добиться наилучшего внешнего вида вашего документа.

5.2.6.5. Обтекание изображений текстом

Значения `left` и `right` атрибута `alignment` предлагают броузеру поместить изображение соответственно у левого или правого края текстового потока. Затем броузер выводит оставшееся содержимое документа в свободное пространство, смежное с изображением. В результате содержимое документа, следующее за изображением, обтекает его.

На рис. 5.14 текст обтекает изображение, выровненное влево.

```
<img src="pics/kumquat.gif" align=left>
The kumquat is the smallest of the citrus fruits, similar in appearance to a
tiny orange. The similarity ends with its appearance, however. While oranges
are generally sweet, kumquats are extremely bitter. Theirs is an acquired
taste, to be sure.
```

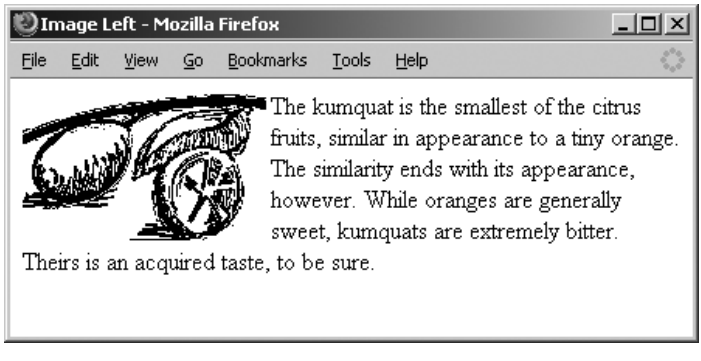

***Рис. 5.14.** Текст обтекает выровненное влево изображение*

Можно поместить изображения одновременно по обоим краям текста (рис. 5.15), и он будет выводиться между ними сверху вниз:

```
<img src="pics/kumquat.gif" align=left>
<img src="pics/tree.gif" align=right>
The kumquat is the smallest of the citrus fruits, similar in appearance to a
tiny orange. The similarity ends with its appearance, however. While oranges
are generally sweet, kumquats are extremely bitter. Theirs is an acquired
taste, to be sure.
```

Когда текст обтекает изображение, левый (или правый) край страницы временно переопределяется и становится краем изображения, противоположным тому, которым оно примыкает к краю страницы. Последующие изображения с тем же выравниванием позиционируются

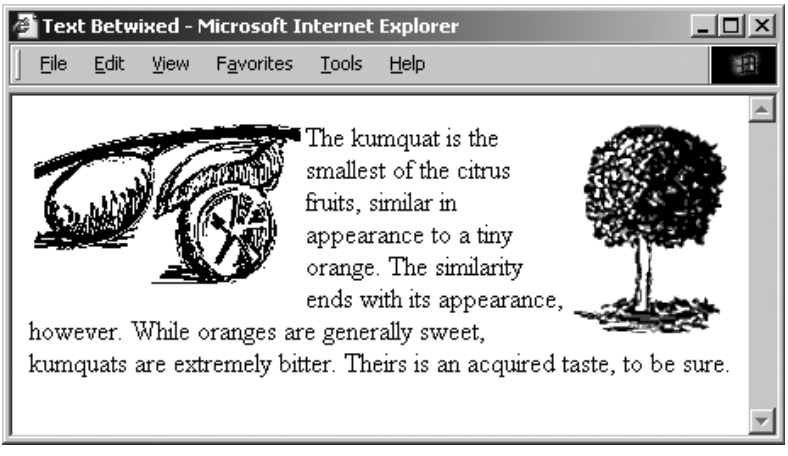

Рис. 5.15. *Текст выводится между изображениями, выровненными влево и вправо*

относительно предыдущего. Следующий исходный текст достигает этого потрясающего эффекта:

```
<img src="pics/marcia.gif" align=left>
Marcia!
<br>
<img src="pics/jan.gif" align=left>
Jan!
<br>
<img src="pics/cindy.gif" align=left>
Cindy!
```

Результат показан на рис. 5.16.

Когда текст добегает до низа изображения, положение полей восстанавливается, перемещаясь обычно к краям окна броузера.

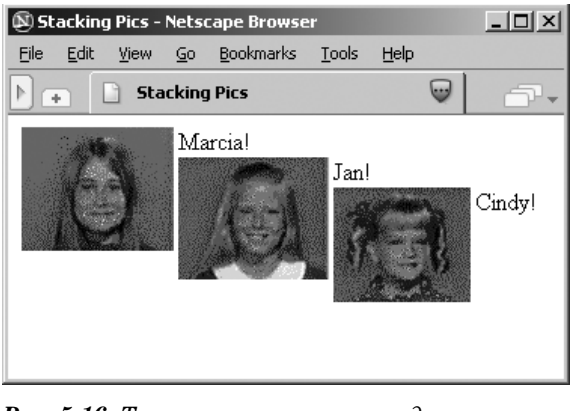

Рис. 5.16. *Три очень симпатичные девочки из старого сериала*

5.2.6.6. Центрирование изображений

Заметили ли вы, что невозможно горизонтально центрировать изображение в окне броузера с помощью атрибута `align`? Значения `middle` и `absmiddle` вертикально центрируют его по отношению к текущей строке, но положение изображения по горизонтали определяется предшествующим содержимым текстового потока и размером окна броузера.

Центрировать изображение по горизонтали можно, только изолировав его от окружающего содержимого с помощью тегов абзаца, раздела или новой строки. Затем следует использовать либо тег `<center>`, либо атрибут `align=center` для тегов абзаца или раздела. К примеру:

```
Kumquats are tasty treats
<br>
<center>
<img src="pics/kumquat.gif">
</center>
that everyone should strive to eat!
```

Используйте тег абзаца с атрибутом `align=center`, если хотите, чтобы над изображением и под ним оставалось свободное место.

```
Kumquats are tasty treats
<p align=center>
<img src="pics/kumquat.gif">
</p>
that everyone should strive to eat!
```

5.2.6.7. Атрибут align и тег нежелательны

В HTML 4 и XHTML атрибут `align` признан нежелательным для всех тегов, включая `<img>`. Предпочтительнее использовать таблицы стилей. Тег `<center>` тоже объявлен нежелательным. Тем не менее атрибут `align` и тег `<center>` очень популярны среди HTML-авторов и хорошо поддерживаются известными броузерами. Так что, хотя мы и ожидаем, что когда-нибудь `align` и `<center>` исчезнут, это случится не скоро. Только потом не говорите, что вас не предупреждали.

Что делать, если вы решили не использовать `align` и `<center>`? Некоторые авторы и многие WYSIWYG-редакторы выравнивают содержимое с помощью таблиц HTML/XHTML. Это первый путь, правда, тернистый (см. главу 10). Консорциум W3C хочет, чтобы вы пользовались стилями. Например, применяйте стиль `margin-left` для смещения изображения от левого края экрана. Вы узнаете о каскадных таблицах стилей (CSS) гораздо больше, если прочитаете главу 8.

5.2.6.8. Атрибут border

Броузеры обычно заключают изображения, которые в то же время являются гиперссылками (расположены в теге `<a>`), в цветную рамку толщиной в два пиксела, давая этим читателю знать, что такое изобра-

жение можно выбрать и посетить ассоциированный с ним документ. Используйте атрибут `border` и его значение, определяющее толщину рамки в пикселах, чтобы убрать ее (`border=0`) или сделать толще. Учтите, что этот атрибут также нежелателен в HTML и XHTML (подобный эффект достигается с помощью таблиц стилей), но по-прежнему поддерживается популярными броузерами.

Рис. 5.17 показывает толстые и тонкие рамки изображений, описанные в следующем XHTML-примере и воспроизведенные Internet Explorer:

```
<a href="test.html">
  <img src="pics/kumquat.gif" border="1" />
</a>
<a href="test.html">
  <img src="pics/kumquat.gif" border="2" />
</a>
<a href="test.html">
  <img src="pics/kumquat.gif" border="4" />
</a>
<a href="test.html">
  <img src="pics/kumquat.gif" border="8" />
</a>
```

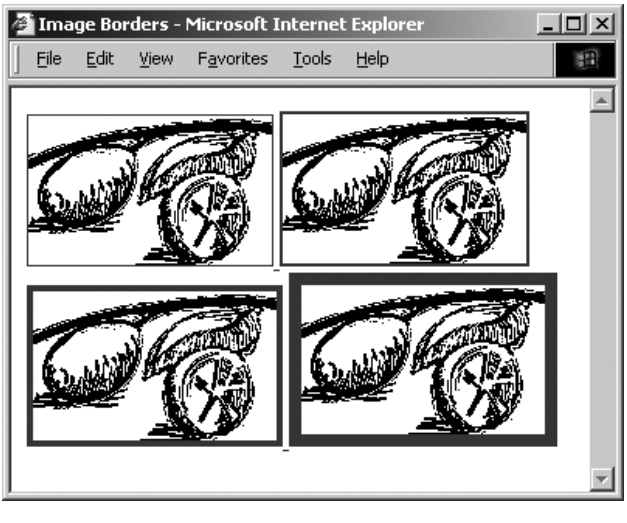

Рис. 5.17. *Толстые и тонкие рамки*

5.2.6.9. Удаление рамки вокруг изображения

Можно полностью удалить рамку вокруг изображения, служащего гиперссылкой, указав в теге `<img>` атрибут `border=0`. Отсутствие рамки, особенно у карт, часто улучшает внешний вид страницы. Изображения, суть которых служить кнопками для перехода на другие страницы, тоже выглядят приятнее без рамок.

Будьте, однако, осторожны – удалив рамку, вы можете ухудшить качество страницы. Убирая рамку, вы удаляете главный визуальный признак гиперссылки, затрудняя читателю ее поиск на странице. Броузер изменит курсор мыши, когда он попадет на гиперссылку, но нельзя заранее предполагать, что его туда поставят, так же как нельзя заставить пользователя проверить на наличие гиперссылок все необрамленные картинки страницы.

Мы настоятельно рекомендуем каким-либо еще образом дать читателю понять, что на лишенном рамки изображении есть смысл щелкнуть мышью. Даже добавление простых текстовых инструкций может очень сильно помочь вашим посетителям.

5.2.6.10. Атрибуты height и width

Видели ли вы когда-нибудь, как содержимое документа при загрузке начинает неожиданно сдвигаться, занимая новое положение? Это происходит из-за того, что броузер каждый раз перестраивает макет страницы, чтобы принять и расположить на нем любое новое изображение. Броузер определяет величину объекта и необходимый для его размещения на странице прямоугольный участок, когда получает файл изображения и извлекает из него спецификацию его размеров. Затем броузер подправляет макет воспроизводимой страницы так, чтобы в нем можно было разместить новое изображение.[1] Это не самый эффективный способ вывода, поскольку броузеру приходится последовательно проверять все файлы изображений и вычислять необходимое пространство на экране, прежде чем воспроизводить соседнее и последующее содержимое документа. Все это может значительно увеличить время на вывод документа и помешать пользователю при чтении.

Более эффективный способ определения размеров изображения состоит в использовании авторами атрибутов `height` и `width` тега `<img>`. В этом случае броузер может зарезервировать место до начала фактической загрузки объекта, что ускоряет вывод документа и устраняет сдвиги содержимого. Оба атрибута принимают целые значения, обозначающие размеры изображения в пикселах. Порядок, в котором они появляются в теге `<img>`, значения не имеет.

5.2.6.11. Изменение размера изображений и «заливка»

Скрытые возможности, содержащиеся в атрибутах `height` и `width`, связаны с тем, что никто не обязан указывать реальные размеры изображения. Значения атрибутов могут быть и больше и меньше действительных размеров объекта. Броузер автоматически масштабирует изображение, чтобы поместить его в выделенное пространство. Это дает вам грубый и неприличный способ создания крошечных версий боль-

[1] Напомним, что изображения – это отдельные файлы, которые загружаются индивидуально и в дополнение к исходному документу.

ших изображений и увеличения очень маленьких картинок. Будьте, однако, осторожны – броузеру все равно придется загрузить файл целиком, независимо от того, каким будет размер, отведенный на экране, и вы исказите изображение, если не соблюдете пропорции между его изначальной высотой и шириной.

Другой прием с атрибутами `height` и `width` позволяет легко «заливать» области на странице и увеличивает скорость исполнения документа. Допустим, вы хотите вставить в макет цветную полосу.[1] Чем создавать изображение полного размера, сделайте лучше картинку высотой и шириной в один пиксел и нужного цвета. Затем используйте атрибуты `height` и `width`, чтобы растянуть ее до требуемой величины:

```
<img src="pics/one-pixel.gif" width=640 height=20>
```

Изображения меньшего размера загружаются быстрее, чем полномасштабные, а атрибуты `width` и `height` (в окне броузера Firefox, например) создают из крошечной точки, полученной броузером, цветную полосу, о которой вы мечтали (рис. 5.18).

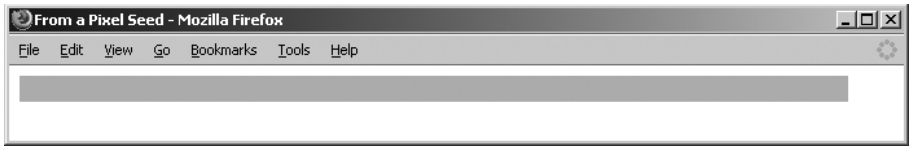

***Рис. 5.18.** Эта полоса была сделана из однопиксельного изображения*

Еще одна последняя хитрость с атрибутом `width` состоит в присвоении ему значения в процентах, а не в абсолютном числе пикселов. Это заставляет броузер масштабировать изображение в процентном отношении к ширине окна документа. Так, чтобы создать цветную полосу высотой в 20 пикселов и шириной во все окно, можно написать:

```
<img src="pics/one-pixel.gif" width="100%" height=20>
```

С изменением размеров окна будут меняться и размеры изображения.

Если вы укажете значение `width` в процентах и пропустите `height`, броузер будет сохранять пропорции изображения при его увеличении и сокращении. Это означает, что высота будет всегда находиться в правильном отношении к ширине, и изображение будет выведено без искажений.

5.2.6.12. Проблемы с width и height

Хотя атрибуты `height` и `width` для тега `<img>` могут улучшить процесс исполнения документа и позволяют совершать с их помощью ловкие проделки, у их применения есть и обратная сторона.

---

[1] Это позволяет рисовать цветные горизонтальные линейки в Netscape 3 и более ранних версий, не поддерживавших атрибут `color` для тега `<hr>`.

Броузер выделяет на экране прямоугольник, размеры которого описаны в теге <img>, даже если пользователь отключил автоматическую загрузку изображений. Ну и с чем тогда остается читатель, как не со страницей, заполненной полупустыми рамками, содержащими бессмысленные пиктограммы. Такая страница выглядит недоделанной и практически бесполезна. Если же размеры не указаны, броузер просто вставляет замещающие изображения пиктограммы в окружающий текст, так что на экране что-то можно хотя бы прочитать.

У нас нет другого решения этой проблемы, кроме как еще раз посоветовать использовать атрибут alt с каким-то описательным текстом, чтобы читатели хотя бы могли знать, чего они лишились. Мы все же рекомендуем включать атрибуты размера, поскольку в целом одобряем методы, улучшающие производительность сетевых процедур.

5.2.6.13. Атрибуты hspace и vspace

Графические броузеры обычно не оставляют большого расстояния между изображениями и окружающим текстом. И если вы не создали прозрачную границу изображения, увеличивающую расстояние между ними, то стандартного буфера толщиной в два пиксела будет не хватать большинству дизайнеров. Вставьте изображение в гиперссылку, и специальная цветная рамка испортит прозрачный буфер, старательно вами созданный, а заодно обратит дополнительное внимание читателя на то, в какой близости от картинки расположен смежный текст.

Атрибуты hspace и vspace могут обеспечить ваши картинки жизненным пространством. С помощью hspace вы указываете число пикселов дополнительного расстояния, которое нужно оставить справа и слева от изображения. Значение vspace – это число пикселов сверху и снизу:

```
<img src="pics/kumquat.gif" align=left>
The kumquat is the smallest of the citrus fruits, similar
in appearance to a tiny orange. The similarity ends with its
appearance, however. While oranges are generally sweet,
kumquats are extremely bitter. Theirs is an acquired taste,
to be sure. Most folks, at first taste, wonder how you could
ever eat another, let alone enjoy it!
<p>
<img src="pics/kumquat.gif" align=left hspace=10 vspace=10>
The kumquat is the smallest of the citrus fruits, similar
in appearance to a tiny orange. The similarity ends with its
appearance, however. While oranges are generally sweet,
kumquats are extremely bitter. Theirs is an acquired taste,
to be sure. Most folks, at first taste, wonder how you could
ever eat another, let alone enjoy it!
```

Рис. 5.19 показывает разницу между двумя изображениями, окруженными текстом.

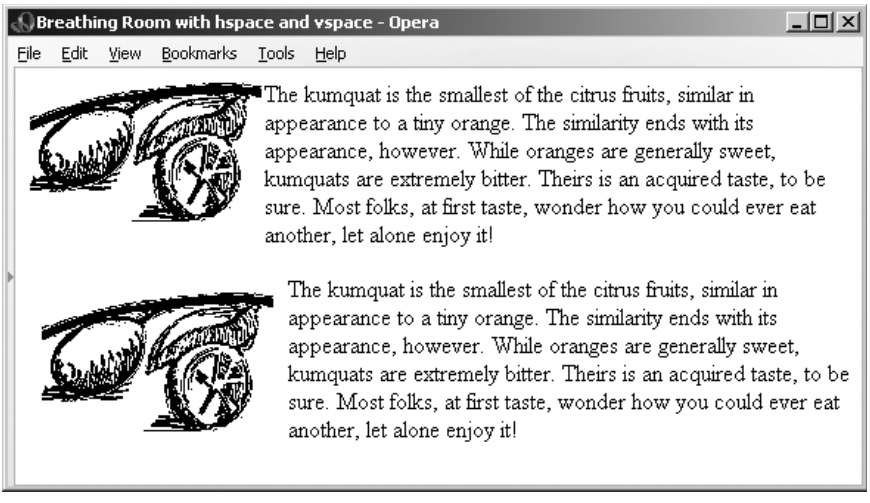

Рис. 5.19. *Улучшайте взаимное расположение текста и изображений спомощью vspace и hspace*

Нет сомнений, что вы согласитесь: дополнительное пространство вокруг изображения облегчает чтение текста и делает страницу в целом более привлекательной.

5.2.6.14. Атрибуты ismap и usemap

Атрибуты `ismap` и `usemap` для тега `<img>` сообщают броузеру, что изображение представляет собой «навигационную карту», содержащую одну или несколько гиперссылок, которые пользователь может выбирать мышью. Атрибут `ismap` указывает на *серверную навигационную карту* и может определяться только в гиперссылочном теге `<a>`. [тег <a>, 6.3.1]

Например (обратите внимание на избыточный атрибут и его значение, а также на завершающую косую в теге `<img>`; это недвусмысленные признаки XHTML):

```
<a href="/cgi-bin/images/map2">
  <img src="pics/map2.gif" ismap="ismap" />
</a>
```

Броузер автоматически посылает серверу координаты мыши относительно левого верхнего угла изображения, когда пользователь щелкает где-то на изображении `ismap`. Специальная программа на сервере (*/cgi-bin/images/map2* в нашем примере) может затем употребить эти координаты при создании ответа.

Атрибут `ismap` – это деталь *серверного* механизма, поскольку он предполагает обработку пользовательского ввода на сервере. Атрибут `usemap` обеспечивает механизм *клиентской навигационной карты*, который эффективно избегает обработки координат мыши со стороны сер-

вера и сопряженных проблем и задержек, связанных с передачей данных по сети. Используя специальные теги <map> и <area>, HTML-авторы указывают на карте области (которые по сути являются гиперссылками) и присваивают соответствующие этим областям URL. Значением атрибута usemap служит URL, указывающий на специальный <map>-раздел. Броузер на компьютере пользователя преобразует координаты щелчка на изображении в некоторое действие, например, загрузку и отображение другого документа. [тег <map>, 6.5.3] [тег <area>, 6.5.4]

К примеру, следующий исходный текст выделяет на карте *map2.gif* размером 100×100 пикселов четыре области, щелчок мышью на каждой из которых вызывает свой документ. Отметьте, что мы законно включили в тег <img> атрибут ismap, который дает пользователям с броузерами, не поддерживающими usemap-механизм, возможность применять альтернативную схему обработки карты с помощью программы на сервере:

```
<a href="/cgi-bin/images/map2">
  <img src="pics/map2.gif" ismap usemap="#map2">
</a>
...
<map name="map2">
  <area coords="0,0,49,49" href="link1.html">
  <area coords="50,0,99,49" href="link2.html">
  <area coords="0,50,49,99" href="link3.html">
  <area coords="50,50,99,99" href="link4.html">
</map>
```

Географические карты дают превосходные примеры употребления атрибутов ismap и usemap. Просматривая веб-страницы компании, ведущей бизнес в национальном масштабе, пользователь может щелкнуть мышью на изображении своего города на карте, чтобы получить адреса и телефоны ближайших представительств компании.

Преимущество обработки карты со стороны клиента по схеме usemap состоит в том, что она не требует сервера или серверного программного обеспечения, и таким образом, в отличие от механизма ismap, может использоваться в отрыве от сети с применением только локальных файлов или CD-ROM.[1]

Прочтите, пожалуйста, раздел 6.5, в котором более полно обсуждаются якоря и ссылки, включая карты.

5.2.6.15. Атрибуты class, dir, event, id, lang, style и title

Несколько почти универсальных атрибутов дают общий способ пометить (id) или назвать (title) содержимое тега для дальнейших ссылок

[1] Другая сторона медали: usemap, как и все технологии на стороне клиента, может произвести, как правило, только грубую детализацию, в то время как серверный механизм ismap – сколь угодно дифференцированную; таким образом, это два взаимно дополняющих механизма, а не альтернативы. – *Примеч. науч. ред.*

или автоматической обработки, изменить характеристики отображения содержимого тега (class, style), указать используемый язык (lang) и связанное с ним направление отображения текста (dir). И, конечно, на все совершаемые пользователем действия, которые связаны с содержимым тега, броузер может соответствующим образом отреагировать с помощью on-атрибутов. [встроенные стили: атрибут style, 8.1.1] [стилевые классы, 8.3]

Среди множества этих HTML 4- и XHTML-атрибутов id важнее всех. Он позволяет пометить изображение для доступа к нему в будущем со стороны программы или операций броузера (глава 12). [атрибут id, 4.1.1.4]

Остальные атрибуты спорны в контексте тега <img>. Естественно, существует возможность влиять на вывод изображения при помощи таблиц стилей, не вредно включить в тег и атрибут title (хотя атрибут alt лучше). И трудно себе вообразить, какое действие на изображение могли бы оказывать атрибуты языка (lang) и направления вывода (dir). [атрибут dir, 3.6.1.1] [атрибут lang, 3.6.1.2] [атрибут title, 4.1.1.5]

5.2.6.16. Атрибуты name, onAbort, onError, onLoad и другие атрибуты событий

В настоящее время все популярные броузеры последовали примеру Netscape и поддерживают четыре атрибута для <img>, позволяющих использовать JavaScript для манипулирования изображениями. Первый – это атрибут name. Ставший избыточным с появлением стандартного атрибута id[1], атрибут name позволяет поименовать изображение, чтобы иметь возможность ссылаться на него из JavaScript-программы. К примеру:

```
<img src="pics/kumquat.gif" name="kumquat">
```

позволит вам упомянуть изображение кумквата в функции JavaScript как «kumquat», чтобы его изменить или удалить. Вы не сможете воздействовать на конкретное изображение с помощью JavaScript, если ему не присвоено имя или с ним не связано определенное значение id.

Три других атрибута позволяют обеспечить обработку специальных событий с помощью JavaScript. Значением атрибута служит заключенный в кавычки отрывок программного кода. Он может состоять из одного или нескольких выражений JavaScript, разделенных точкой с запятой.

Популярные броузеры вызывают обработчик события onAbort, если читатель останавливает загрузку изображения, нажав, как это обычно бывает, на кнопку «Stop». Вы можете, например, применять onAbort сообщение, предупреждающее пользователя, когда он останавливает загрузку существенного изображения, такого как, скажем, карты со ссылками (раздел 6.5):

---

[1] HTML версии 4.01 включил атрибут name в стандарт, хотя в настоящее время только Netscape поддерживает его употребление с тегом <img>.

```
<img src="pics/kumquat.gif" usemap="#map1"
onAbort="window.alert('Caution: This image contains important hyperlinks.
Please load the entire image.')">
```

Атрибут `onError` вызывается, когда во время загрузки происходит какая-либо ошибка, отличающаяся от потери изображения или упомянутого отказа пользователя. Предположительно, программа должна попытаться исправить ошибку или загрузить на выделенное место другое изображение.

Современные броузеры исполняют код JavaScript, связанный с атрибутом `onLoad` тега `<img>` сразу после того, как броузер успешно загрузил и вывел изображение.

В разделе 12.3.3 содержится дополнительная информация о JavaScript и обработке событий.

5.2.6.17. Комбинирование атрибутов в теге <img>

Можно комбинировать стандартные и относящиеся к расширениям атрибуты в любых случаях, когда это имеет смысл. Порядок, в котором перечисляются атрибуты, не важен. Остерегайтесь только избыточных атрибутов, так как иначе вы не сможете предвидеть результат.

5.2.7. Видеорасширения

Internet Explorer поддерживает специальные видеорасширения атрибута тега `<img>`, которые позволяют вам встраивать видеоролики в HTML-документы. Это `controls`, `dynsrc`, `loop`, и `start`. Они не являются частью HTML 4 и вряд ли станут стандартными в XHTML. Пользователям приходится явно включать эту функциональность броузера Internet Explorer флажком «Воспроизводить видео на веб-страницах» на вкладке «Дополнительно».

Эквивалентное поведение других популярных броузеров достигается за счет программных расширений, известных как *плагины* (*plug-ins*). Плагины доставляют читателям лишние хлопоты: каждый пользователь должен раздобыть и установить подходящую программу, прежде чем сможет просматривать встроенное в страницу видео.[1] Расширения тега `<img>`, которые принял Internet Explorer, напротив, делают видео собственной частью броузера. [вложенное содержимое, 12.2]

---

[1] Плагины, как правило, выложены для свободного скачивания на тех же URL, где представлен и сам броузер, поэтому найти их нетрудно и выбрать есть из чего. Перечень свободно распространяемых плагинов постоянно расширяется по мере появления новых форматов видеоизображений. Напротив, «многофункциональность», принятая в Internet Explorer, ограничивает вкусы пользователя только тем, что ему предоставили разработчики броузера; такой подход скорее ограничивает выбор пользователя, чем расширяет функциональность. – *Примеч. науч. ред.*

5.2.7.1. Атрибут dynsrc

Используйте атрибут `dynsrc`, расширяющий тег `<img>`, для воспроизведения видеоролика в одном из популярных форматов (AVI, MPG или MPEG, MOV, WMV и др.) в окне Internet Explorer. Обязательное значение этого атрибута представляет собой заключенный в кавычки URL файла с видео. Следующий пример представляет тег и его атрибут для воспроизведения AVI-видео, содержащегося в файле *intro.avi*:

```
<img dynsrc="movies/intro.avi">
```

Internet Explorer выделяет прямоугольную область в окне документа и показывает в нем видео в сопровождении аудио, если оно входит в клип и если ваш компьютер может его воспроизводить. Internet Explorer обращается с видео, на которое ссылается `dynsrc`, как с обычными изображениями, – помещает в строку с текущим содержимым документа в соответствии с размерами рамки видео. И подобно обычным изображениям, видео, на которое ссылается `dynsrc`, запускается сразу же после того, как оно загружено с сервера. Вы можете изменить это поведение так же, как и управлять некоторыми другими характеристиками воспроизведения видео с помощью других атрибутов, описанных ниже.

Поскольку другие броузеры в настоящее время не поддерживают специальных атрибутов для управления видео, которые ввел Internet Explorer, они могут быть приведены в замешательство тегом `<img>`, который не содержит обязательного в других случаях атрибута `src` и URL изображения. Мы рекомендуем включать атрибут `src` и значимый URL во всякий тег `<img>`, включая и те, что предназначены для ссылок на видео для пользователей Internet Explorer. Другие броузеры на месте видео отобразят картинку, Internet Explorer поступит наоборот, покажет видео, но не покажет картинку. Заметьте, что порядок атрибутов значения не имеет. В следующем примере:

```
<img dynsrc="movies/intro.avi" src="pics/mvstill.gif">
```

Internet Explorer загрузит и воспроизведет AVI-клип *intro.avi*, а другие графические броузеры загрузят и покажут изображение *mvstill.gif*.

5.2.7.2. Атрибут controls

Обычно Internet Explorer проигрывает видео в окне только один раз, не предоставляя пользователю никаких видимых средств управления. В старых версиях броузера Internet Explorer пользователь мог перезапустить, остановить и продолжить клип, щелкая мышью на области вывода, но эти возможности уже не поддерживаются, начиная с версии 5. Кроме того, если видеоклип содержал звуковую дорожку, Internet Explorer старых версий добавлял регулятор громкости. Пример:

```
<img dynsrc="movies/intro.avi" controls src="pics/mvstill.gif">
```

5.2.7.3. Атрибут loop

Internet Explorer проигрывает клип от начала и до конца один раз сразу после загрузки. Атрибут `loop` для тега `<img>` позволяет заставить клип исполняться повторно целое число раз, равное значению атрибута, или бесконечно, если установлено значение `infinite`. Пользователь может разорвать цикл воспроизведения, нажав кнопку «Стоп» (если имеются элементы управления) или перейдя на другой документ.

Клип *intro.avi*, запущенный в следующем примере, будет исполнен с начала и до конца, затем снова начнется с начала и будет проигран до конца еще девять раз:

```
<img dynsrc="movies/intro.avi" loop=10 src="pics/mvstill.gif">
```

Тогда как следующий HTML-код заставит клип воспроизводиться без конца:

```
<img dynsrc="movies/intro.avi" loop=infinite src="pics/mvstill.gif">
```

Зацикленные клипы не всегда раздражают. Некоторые специальные анимационные эффекты, к примеру, представляют собой последовательность повторяющихся кадров. Вместо того чтобы связывать повторяющиеся фрагменты в один большой видеоклип, занимающий много времени при загрузке, просто зациклите один компактный фрагмент.

5.2.7.4. Атрибут start

Обычно Internet Explorer начинает исполнение клипа сразу после его загрузки. Такое положение дел можно изменить при помощи атрибута `start` в теге `<img>`. Присвоив ему значение `mouseover`, можно отложить воспроизведение до того момента, как мышь пользователя попадет в область вывода клипа. Другое допустимое значение атрибута `fileopen` соответствует принятому по умолчанию исполнению сразу после загрузки. Это свойство упоминается здесь, потому что оба значения могут быть присвоены атрибуту совместно, чтобы клип был исполнен один раз при загрузке, а затем проигрывался всякий раз, когда пользователь попадает мышью в поле вывода. При комбинировании значений атрибута `start` они разделяются запятой без пробелов между ними и заключаются в кавычки.

Например, уже надоевшее видео *intro.avi* будет исполнено первый раз сразу после того, как пользователь Internet Explorer загрузит HTML-документ, содержащий ссылку на него, а затем будет проигрываться всякий раз, когда мышь попадет в область вывода:

```
<img dynsrc="movies/intro.avi" start="fileopen,mouseover"
     src="pics/mvstill.gif">
```

5.2.7.5. Комбинирование атрибутов <img>, относящихся к видео

Обращайтесь со встроенным видео для Internet Explorer так же, как с изображениями, комбинируя и подбирая различные атрибуты, относящиеся не только к видео, но и к стандартным или принадлежащим расширениям тега `<img>`. К примеру, можно выровнять видео (или заменяющее его изображение, если документ воспроизводится другим броузером) по правому краю окна броузера:

```
<img dynsrc="movies/intro.avi" src="pics/mvstill.gif" align=right>
```

Комбинировать атрибуты для достижения спецэффектов – это верное дело. Мы также рекомендуем комбинировать атрибуты, чтобы предоставить пользователю возможность управления.

Как установлено в разделе 5.2.7.4, не трудно, манипулируя атрибутами, отключить воспроизведение видео до появления мыши в области вывода. Видео волшебным образом оживает и крутится без конца:

```
<img dynsrc="movies/magic.avi" start=mouseover
     loop=infinite src="pics/magic.gif">
```

5.3. Цвета документа и фоновые изображения

Стандарты HTML 4 и XHTML предоставляют множество атрибутов для тега `<body>`, позволяющих устанавливать цвета текста, ссылок, других элементов и, кроме того, определять изображение, которое будет применяться для создания фона документа. Все популярные броузеры расширяют множество этих атрибутов, чтобы включить поля страницы и улучшить управление фоновым изображением. И разумеется, новейшие технологии таблиц стилей, встроенные в современные броузеры, позволяют манипулировать всеми этими параметрами отображения.

5.3.1. Дополнения и расширения тега <body>

Атрибуты, управляющие фоном и полями документа, а также цветом текста, используются с тегом `<body>`. [тег <body>, 3.8.1]

5.3.1.1. Атрибут bgcolor

Стандартный (но уже объявленный нежелательным) способ изменить принятый по умолчанию цвет фона состоит в использовании атрибута `bgcolor` для тега `<body>`. Как и для атрибута `color` тега `<font>`, обязательное значение `bgcolor` может быть выражено одним из двух способов – с помощью указания красного, зеленого и синего (RGB) компонентов выбранного цвета или его стандартного наименования. Приложение G содержит RGB-кодировки цветов вместе с таблицей наименований, которые следует применять в качестве значений атрибута `bgcolor`.

Установить цвет фона легко. Чтобы получить ярко-красный фон, используя RGB-кодировку, попробуйте написать:

```
<body bgcolor="#FF0000">
```

А более нежный фон, персикового цвета, так:

```
<body bgcolor="peach">
```

5.3.1.2. Атрибут background

Если однотонный фон вас не устраивает, то с помощью атрибута `background` тега `<body>` можно поместить на его место изображение.

Обязательным значением атрибута `background` является URL изображения. Броузер автоматически повторяет (в виде мозаики) картинку по вертикали и по горизонтали, заполняя все окно.

Обычно следует выбирать маленькое не слишком яркое изображение, чтобы создать интересный, но не навязчивый тип фона. Кроме того, маленькая простая картинка загружается из сети гораздо быстрее, чем сложное полноэкранное изображение.

Рис. 5.20 и 5.21 показывают, как расширенный броузер использует картинку деревянной панели, создавая покрытие стенки, служащей фоном документа:

```
<body>
<img src="pics/wood_panel.gif">
```

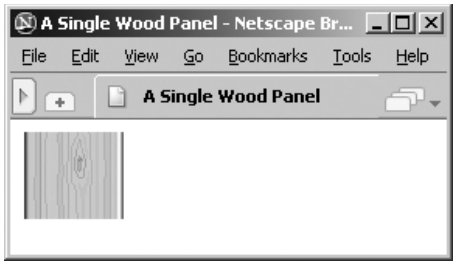

***Рис. 5.20.** Одна деревянная панель...*

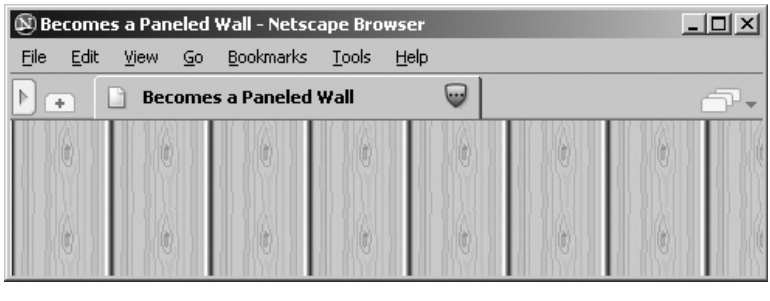

***Рис. 5.21.** ...превращается в стенку*

Сравните:

```
<body background="pics/wood_panel.gif">
```

Фоновые изображения разных размеров и пропорций создают интересные горизонтальные и вертикальные эффекты на странице. К примеру, высокая узкая картинка может выделять заголовок документа:

```
<body background="pics/vertical_fountain.gif">
<h3>Kumquat Lore</h3>
For centuries, many myths and legends have arisen around the kumquat.
...
```

Если *vertical_fountain.gif* – это узкое, вытянутое вверх изображение, которое постепенно светлеет к своему основанию и длина которого превосходит длину тела документа, то в результате получится что-то, похожее на рис. 5.22.

Не трудно организовать подобный эффект в горизонтальном направлении, использовав изображение, которое гораздо больше в ширину, чем в высоту (рис. 5.23).

Атрибут `background` нежелателен в HTML 4 и XHTML, поскольку аналогичные эффекты могут достигаться с применением таблиц стилей.

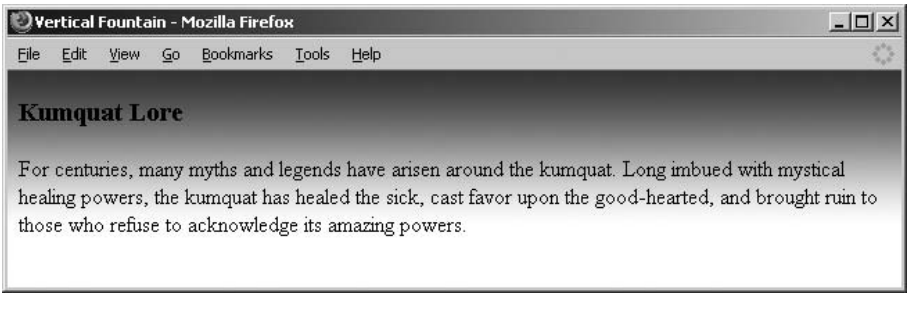

Рис. 5.22. *Высокий узкий фон*

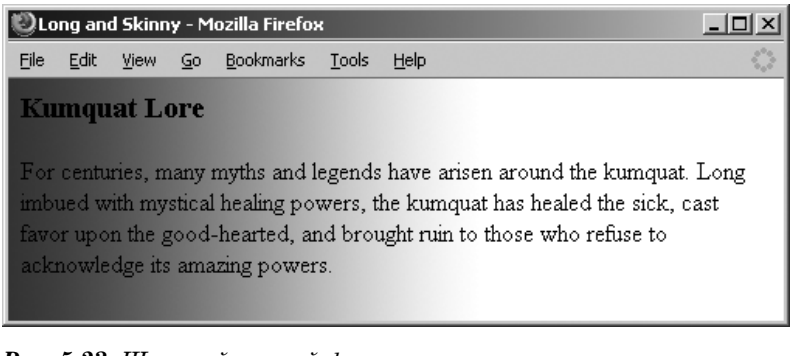

Рис. 5.23. *Широкий низкий фон*

5.3.1.3. Атрибут bgproperties

Популярные броузеры больше не поддерживают атрибут-расширение `bgproperties` для тега `<body>`. Он работает только вместе с атрибутом `background`. Атрибут `bgproperties` имеет лишь одно значение, `fixed`. Оно «замораживает» фоновое изображение, которое перестает прокручиваться вместе с другим содержимым окна. И, следовательно, фоновое изображение *H2Omark.gif* может служить для документа гербовой бумагой с водяными знаками:

```
<body background="pics/H2Omark.gif" bgproperties="fixed">
```

5.3.1.4. Атрибут text

Изменив цвет фона или установив фоновое изображение, следует, вероятно, поменять цвет текста, чтобы документ можно было прочесть. Атрибут `text` для тега `<body>`, включенный в стандарты HTML и XHTML, делает именно это – определяет цвет текста во всем документе, кроме якорей гиперссылок.

Присваивайте атрибуту `text` цветовое значение, как это делалось при определении цвета фона (употребление `bgcolor` в разделе 5.3.1.1), используя триплет RGB или его название в соответствии с приложением G. Например, чтобы создать документ с синим текстом на бледно-желтом фоне, напишите:

```
<body bgcolor="#777700" text="blue">
```

Разумеется, лучше выбрать цвет текста, контрастирующий с фоном.

Атрибут `text` нежелателен в стандартах HTML 4 и XHTML, поскольку подобного эффекта можно достигнуть с помощью таблиц стилей.

5.3.1.5. Атрибуты link, vlink и alink

Атрибуты `link`, `vlink` и `alink` тега `<body>` управляют цветом гиперссылок (текст внутри тега `<a>`) в документе. Все они принимают значения, определяющие цвет в виде RGB-триплета или его названия, так же как атрибуты `text` и `bgcolor`.

Атрибут `link` определяет цвет всех гиперссылок, по которым пользователь еще не «ходил». Атрибут `vlink` устанавливает цвет посещенных ссылок, а атрибут `alink` назначает цвет активной гиперссылки, т. е. той, что выбрана пользователем и находится под курсором мыши при ее нажатой кнопке.

Как и в случае текста, следует позаботиться о том, чтобы окраска ссылок позволяла прочесть их на выбранном фоне. Кроме того, цвета ссылок в различных состояниях должны отличаться и от тональности основного текста и друг от друга.

Эти атрибуты нежелательны в стандартах HTML 4 и XHTML, поскольку подобного эффекта можно достигнуть с помощью таблиц стилей.

5.3.1.6. Атрибут leftmargin

Свойственный только Internet Explorer атрибут `leftmargin` тега `<body>` позволяет отодвинуть левый край текста от левого края окна броузера, создав подобие поля на бумажной странице. Другие броузеры игнорируют этот атрибут и обычно выравнивают текст по левому краю окна броузера. Значение атрибута `leftmargin` представляет собой целое число пикселов, составляющее ширину левого поля. По умолчанию принимается значение, равное 10. Поле заполняется фоновым цветом или изображением.

Исполняя следующий пример исходного текста, Internet Explorer выводит текст, выровненный по левому полю, имеющему ширину отступа 50 пикселов (рис. 5.24).

```
<body leftmargin=50>
Modern browsers let you indent the<br>
&lt;--left margin<br>
away from the left edge of the window.
</body>
```

***Рис. 5.24.** Атрибут leftmargin для вывода содержимого с отступом*

5.3.1.7. Атрибут topmargin

Подобен `leftmargin` атрибут `topmargin`, но в настоящее время он поддерживается не только в Internet Explorer, но и во всех популярных броузерах. Его можно включить в тег `<body>`, чтобы установить ширину верхнего поля в документе. Пространство поля заполняется фоновыми цветом или изображением.

К примеру, Opera выводит следующий текст по меньшей мере на 50 пикселов ниже верхнего края окна броузера (рис. 5.25).

```
<body topmargin=50>
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

Modern browsers give your documents
a little extra headroom with topmargin.
</body>
```

Содержимое тела начинает отображаться ниже на целое число пикселов, которое указывается в качестве значения атрибута `topmargin`. По умолчанию принимается значение 0.

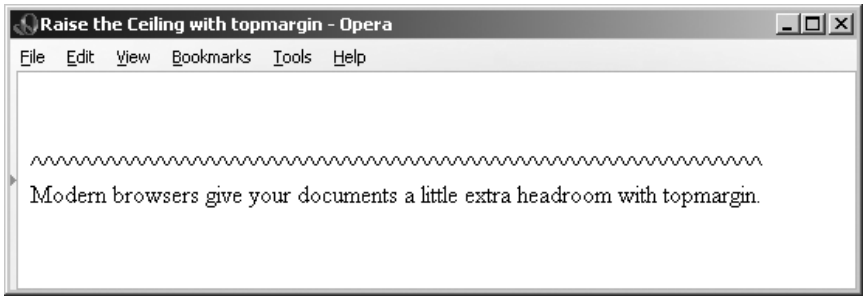

Рис. 5.25. *Атрибут topmargin смещает содержимое тела вниз*

5.3.1.8. Атрибуты style и class

Также можно устанавливать различные относящиеся к стилю свойства тела при помощи CSS. Но хотя для оформления всего содержимого тела можно включить атрибут `style` в тег `<body>`, мы рекомендуем устанавливать стили для всего тела на уровне документа (используя тег `<style>` в его заголовке) или на уровне собрания документов (применяя импортируемую таблицу стилей). Используйте атрибут `class` и его значение для применения характеристики заранее определенного класса стилей к содержимому тега `<body>`. (Поскольку в одном документе может быть только одно тело, зачем еще нужно название класса?) Мы обсуждаем использование атрибута `style` и определение классов в главе 8.

5.3.1.9. Комбинирование атрибутов тега <body> и их взаимодействие

Хотя атрибуты `background` и `bgcolor` могут появиться в одном теге `<body>`, фоновое изображение скроет выбранный цвет фона, если только оно не содержит областей прозрачности существенного размера, которые уже были описаны в этой главе. Но даже если изображение полностью скрывает цвет фона, не отступайте и включите в тег атрибут `bgcolor` с подходящим значением фонового цвета. Пользователи могут отключить загрузку изображений (в том числе и фонового), и тогда ваша страница предстанет перед ними голой и непривлекательной. Более того, если в теге отсутствует атрибут `bgcolor`, а фоновое изображение почему-либо не загрузится, броузер хладнокровно проигнорирует атрибуты, устанавливающие цвет текста и ссылок, назначив вместо них принятые по умолчанию или заданные пользователем.

5.3.2. Расширенное предостережение

Подобно тому как первые пользователи Макинтошей испытывали соблазн создавать документы, по богатству примененных шрифтов не уступающие письмам с требованиями выкупа (ход их мысли: «Здесь стоит 40 шрифтов, и я вставлю их все!»), многие HTML-авторы не могут

отказаться от добавления текстурированного фона в каждую построенную ими страницу («У меня есть 13 паркетных плашек и 22 сорта плитки, и они все будут здесь!»).

В действительности, за редчайшими исключениями, текстурированный фон не добавляет никакой информации. Ценность документа заключается, в конечном счете, в его тексте и образности выражения, а не в изысканных завитках цвета сырной плесени на фоновой картинке. Неважно, как все «оригинально выглядит», – ваши читатели ничего от этого не выиграют, разве только текст будет не разобрать.

Мы советуем не использовать «украшений», кроме как в сравнительно легкомысленных опусах или в случаях, когда такие расширения действительно увеличивают ценность документа, как это бывает с маркетинговыми и рекламными страницами.

5.3.2.1. Проблемы с фоновыми изображениями

Здесь перечислены некоторые неприятности, связанные с фоновыми изображениями.

- Время, необходимое на загрузку документа, увеличивается за счет загрузки фонового изображения, до завершения которой вывод документа не происходит.
- Фоновое изображение занимает место в локальном кэше броузера, порой выталкивая изображения, содержащие действительно полезную информацию. Это может увеличить время загрузки других документов, даже не включающих фоновой живописи.
- Цвета изображения оказываются иногда недоступными для дисплея пользователя, заставляя броузер «расщеплять» исходную палитру. В результате большие области, представленные в оригинале одним цветом, будут заполнены чередующимися слоями различной, хотя и близкой по цвету окраски. Документ, вероятно, будет труднее читать.
- Поскольку броузеру приходится постоянно перерисовывать фоновое изображение (а это совсем не то же самое, что заполнить область одним цветом), прокручивание документа может занимать значительно больше времени.
- Даже если все хорошо выглядит на дисплее, напечатанный поверх изображения текст всегда трудно, а бывает и невозможно прочитать.
- Шрифты на разных машинах могут сильно различаться. Те, что неплохо сочетаются с выбранным фоном на вашем броузере, порой нельзя будет прочитать на другом компьютере.

5.3.2.2. Проблемы с цветами текста, фона и гиперссылок

Играя с цветами фона, вы также столкнетесь с множеством проблем.

- Выбранный цвет тешит ваш глаз, а пользователям кажется ужасным. Зачем же их раздражать, отменяя ту окраску фона, которую они сами установили для применения по умолчанию?
- В то время как вы, скажем, принадлежите к школе дизайнеров, твердо придерживающейся принципа «светлый текст на темном фоне», многим людям нравится также стиль «темный текст на светлом фоне», сохраняющий популярность последние три тысячи лет.
- Вместо того чтобы плыть против течения, допустите на минуту, что пользователи уже настроили свои броузеры на устраивающую их цветовую схему.
- Некоторые читатели – дальтоники, не различающие цветов. То, что будет прелестным красочным сочетанием для вас, может оказаться совершенно неразборчивым для них. Одну комбинацию следует особенно избегать: зеленый цвет для неиспользованных ссылок и красный – для тех, что уже были посещены. Миллионы людей не различают красный и зеленый.
- Превосходный оттенок, разработанный вами, иногда оказывается недоступным для дисплея пользователя, и броузеру придется заменить его на близкий. Для дисплеев с очень небольшим набором цветов (таких, как несколько миллионов находящихся в употреблении 16-цветных VGA-мониторов) сходная тональность текста и фона может обернуться одной и той же окраской!
- По тем же причинам активные, неиспользованные и использованные гиперссылки на дисплее с ограниченным набором цветов могут оказаться неразличимыми.
- Изменяя цвета текста, особенно в случае использованных и неиспользованных гиперссылок, вы можете совершенно запутать читателя. Достаточно сказать, что это эффективно стимулирует пользователей к экспериментам со страницей. Они будут щелкать на ссылках там и здесь, разучивая вашу цветовую схему.
- Большинство дизайнеров веб-страниц не получили специального образования в области когнитивной психологии[1], изящных искусств или промышленного дизайна, но чувствуют себя уверенно, подбирая цвета для документа. Если вам нужно принять цветовые решения, обратитесь к профессионалу.

5.3.2.3. И еще раз

Невозможно отрицать тот факт, что «украшения» послужили созданию абсолютно сногсшибательных HTML- и XHTML-документов. Их

[1] Когнитивная психология – направление в психологии, исследующее человеческий и искусственный интеллект, распределение внимания, память, воображение, восприятие и распознавание образов, речь, мышление, психологию развития. – *Примеч. науч. ред.*

интересно исследовать, с ними весело играть. И чтобы не заканчивать на унылой ноте предостережений, мы говорим вам: изобретайте, играйте, но играйте осторожно.

5.4. Звуковой фон

Существует еще один мультимедийный элемент, как правило доступный веб-серферам: это аудио. Большинство броузеров обращаются с аудио как с отдельными документами, которые исполняются специальными вспомогательными приложениями, апплетами или плагинами. Internet Explorer и Opera, кроме того, имеют встроенный звуковой декодер и поддерживают специальный тег (<bgsound>), позволяющий интегрировать аудиофайл с документом в качестве фонового звукового сопровождения страницы. [апплеты и объекты, 12.1] [вложенное содержимое, 12.2]

Весьма похвально, что разработчики Internet Explorer и Opera предоставили механизм, который аккуратно интегрировал аудио с HTML 4- и XHTML-документами. Возможности, открывающиеся с применением аудио, очень заманчивы. Но в то же время мы предостерегаем авторов, напоминая им, что, во-первых, специальные теги и атрибуты для аудио не работают с другими броузерами, и, во-вторых, неизвестно, поддержат ли в конечном счете этот метод большинство броузеров.

5.4.1. Тег <bgsound>

Используйте тег <bgsound> для фонового исполнения звуковой дорожки. Этот тег поддерживают только Internet Explorer и Opera. Все другие броузеры его игнорируют. Он загружает и запускает аудиофайл при загрузке содержащего его документа. Фоновое звуковое сопровождение будет возобновляться при обновлении отображаемого документа.

<bgsound> 

Функция:	Проигрывает фоновое звуковое сопровождение
Атрибуты:	loop, src
Закрывающий тег:	Нет в HTML
Содержит:	Ничего
Может содержаться в:	*body_content* (содержимое тела)

5.4.1.1. Атрибут src

Атрибут src является необходимым в теге <bgsound>. Его значением является URL звукового файла. К примеру, когда пользователь Internet Explorer или Opera впервые загружает документ, содержащий этот тег:

```
<bgsound src="audio/welcome.wav">
```

он услышит однократное исполнение аудиофайла *welcome.wav* – возможно, приветствия – звуковой системой компьютера.

В настоящее время `<bgsound>` поддерживает несколько форматов звуковых файлов: `wav` – родной формат для PC; `au` – родной формат для рабочих станций под Unix; MIDI – формат универсальной схемы кодирования музыки (табл. 5.1).

Таблица 5.1. Мультимедийные форматы и соответствующие расширения имен файлов

Формат	Тип	Расширение	Оригинальная платформа
GIF (Graphics Interchange Format)	Изображение	*gif*	Любая
JPEG (Joint Photographic Experts Group)	Изображение	*jpg, jpeg, jpe*	Любая
XMB (X Bit Map)	Изображение	*xbm*	Unix
TIFF (Tagged Image File Format)	Изображение	*tif, tiff*	Любая
PICT	Изображение	*pic, pict*	Apple
Rasterfile	Изображение	*ras*	Sun
PNG (Portable Network Graphics)	Изображение	*png*	Любая
MPG (Moving Pictures Expert Group)	Видео	*mpg, mpeg*	Любая
AVI (Audio Video Interleave)	Видео	*avi*	Microsoft
QuickTime	Видео	*qt, mov*	Apple
WMV (Windows Media Video)	Видео	*wmv*	Microsoft
Shockwave	Видео	*dvr*	Macromedia
Real Video	Видео	*ra, rm, ram*	Real Networks
DivX	Видео	*div, divx, tix, mp4*	DivX
AU	Аудио	*au, snd*	Sun
WAV (Waveform Audio)	Аудио	*wav*	Microsoft
AIFF (Audio Interchange File Format)	Аудио	*aif, aiff*	Apple
MIDI (Musical Instrument Digital Interface)	Аудио	*midi, mid*	Любая
PostScript	Документ	*ps, eps, ai*	Любая
Acrobat	Документ	pdf	Любая

5.4.1.2. Атрибут loop

Как и в случае встроенного видео, атрибут `loop` для тега `<bgsound>` позволяет повторять воспроизведение фонового саундтрека определен-

ное число раз (или до бесконечности), если только пользователь не переходит на другую страницу или не закрывает свой броузер.

Значение атрибута `loop` – это целое число раз исполнения аудио или `infinite`, которое заставляет звуковую дорожку (саундтрек) повторяться без конца.

К примеру:

```
<bgsound src="audio/tadum.wav" loop=10>
повторяет "ta-dum" десять раз, тогда как:
<bgsound src="audio/noise.wav" loop=infinite>
```

безостановочно проигрывает саундтрек «noise».

5.4.2. Альтернативная поддержка аудио

Существуют другие способы вставить аудио в документы. Эти способы задействуют более общие механизмы, поддерживающие вложение и других мультимедийных элементов. Самая обычная альтернатива тегу `<bgsound>` – это тег `<embed>`, изначально введенный Netscape и вытесненный тегом `<object>` в стандартах HTML и XHTML. За подробностями обратитесь к главе 12.

В конечном счете, вы должны обрабатывать весь фоновый аудиоматериал, в том числе речевые документы, с помощью различных аудиорасширений, определенных в стандарте CSS. Хотя мы и обсуждаем в главе 8 расширения, относящиеся к синтезу речи, они еще не поддерживаются ни в одном броузере. Когда такая поддержка станет широко доступной, все эти первые аудиорасширения постигнет участь тегов `<blink>` и `<isindex>`, которые служили специфическим целям, а затем уступили место обобщенным и более мощным функциональным возможностям.

5.5. Анимация текста

Заботясь, видимо, о производителях рекламы, Internet Explorer ввел в HTML анимированный текст, который теперь поддерживается всеми броузерами. Анимация простая – текст бежит по экрану в горизонтальном направлении, – но пригодная для движущихся баннеров и других элементов, с помощью которых не трудно оживить документ, остававшийся бы без них статичным. С другой стороны, анимированный текст, как и мерцающий, может легко стать надоедливым, начать раздражать читателей. Если собираетесь использовать его, будьте осторожней.

5.5.1. Тег <marquee>

Тег `<marquee>` определяет текст, который будет отображаться в бегущей строке окна. Тег `<marquee>` не является стандартным. Этого достаточно, чтобы мы не рекомендовали вам его употреблять.

Текст, заключенный между <marquee> и обязательным </marquee>, бежит в горизонтальном направлении через окно броузера. Атрибуты тега управляют областью отображения, внешним видом, выравниванием по отношению к окружающему тексту и скоростью бегущей строки.

Тег <marquee> и его атрибуты игнорируются другими броузерами, но его содержимое не опускается. Оно выводится как статический текст без учета выравнивания и прочих свойств, определенных атрибутами данного тега.

<marquee>

Функция:	Создает бегущую строку
Атрибуты:	align, behavior, bgcolor, class, controls, direction, height, hspace, loop, scrollamount, scrolldelay, style, vspace, width
Закрывающий тег:	</marquee>; присутствует обязательно
Содержит:	*plain_text* (гладкий текст)
Может содержаться в:	*body_content* (содержимое тела)

5.5.1.1. Атрибут align

Популярные броузеры размещают текст из тега <marquee> так же, как если бы это было вложенное изображение. Следовательно, можно выровнять бегущую строку по отношению к окружающему тексту.

Атрибут align принимает значения top, middle и bottom, устанавливающие, что определенная точка строки будет выровнена с соответствующей точкой в окружающем тексте. Таким образом:

```
<marquee align=top>
```

выравнивает верхний край области вывода текущей строки с верхним краем облегающего текста. Смотрите также атрибуты height, width, hspace, vspace (далее в этой главе), управляющие размерами бегущей строки.

5.5.1.2. Атрибуты behaviour, direction, loop

Вместе эти атрибуты управляют стилем, направлением и продолжительностью отображения бегущей строки.

Атрибут behaviour принимает три значения:

scroll (*принимается по умолчанию*)

Значение scroll заставляет бегущую строку вести себя подобно огромной бегущей строке на Times Square. Первоначально область ее размещения пуста. Текст начинает бежать от одного края (атрибут direction определяет от которого), добегает до противоположного и затем исчезает. В конце концов, область текущей строки снова пуста.

`slide`

Это значение заставляет строку появляться, как в предыдущем случае, заполняя отведенное для нее изначально пустое пространство и двигаясь в направлении, определенном атрибутом `direction`, но затем по достижении противоположного края области вывода останавливаться и застывать на экране.

`alternate`

Это значение порождает следующий эффект: текст бегущей строки сразу же полностью выводится на одном крае области вывода, затем течет до противоположного края отведенного пространства, где «отражается» и бежит назад к начальной точке своего движения.

Если значение атрибута `behaviour` явно не указано, то по умолчанию он принимает значение `scroll`.

Атрибут `direction` устанавливает направление движения строки на экране. Допустимые значения – это `left` (принимается по умолчанию) или `right`. Заметьте, что начальный пункт движения находится на краю, противоположном значению `direction`. Значение `left` определяет, что текст начинает движение справа и перетекает налево. Помните также, что текст, бегущий слева направо, противоречит интуиции людей, читающих слева направо.

Атрибут `loop` определяет число «прогонов» строки. Если ему присвоено целое значение, строка пробежит по экрану указанное число раз. Если атрибуту `loop` назначено значение `infinite`, строка будет пробегать раз за разом, пока пользователь не перейдет к другому документу.

Соберем несколько описанных атрибутов вместе:

```
<marquee align=center loop=infinite>
  Кумкваты без начинки
  ..........      Очень вкусно!
</marquee>
```

Сообщение из примера появляется у правого края (по умолчанию), пробегает, двигаясь налево, весь свой путь и пропадает из окна, затем появляется снова и снова, пока пользователь не перейдет к другому документу. Обратите внимание на промежуточные точки и пробелы в тексте примера. Вы не можете присоединить одну бегущую строку к другой.

Отметим также, что строка стиля `slide` неприятно дергается при повторениях и ее следует запускать лишь однажды, тогда как другим значениям `behaviour` повторение показано.

5.5.1.3. Атрибут bgcolor

Атрибут `bgcolor` позволяет изменить цвет фона в области вывода бегущей строки. Его значением могут быть либо RGB-код цвета, либо его стандартное наименование. Смотрите приложение G, где подробно обсуждаются оба способа спецификации цвета.

Чтобы создать бегущую строку на желтом фоне, следует написать:

```
<marquee bgcolor=yellow>
```

5.5.1.4. Атрибуты height и width

Атрибуты `height` и `width` определяют размеры области вывода для бегущей строки. Если значения этих атрибутов не указаны, область вывода простирается слева направо через все окно и имеет высоту, как раз достаточную, чтобы вместить текст бегущей строки.

Оба атрибута принимают либо целые значения, являющиеся абсолютными размерами в пикселах, либо процентные значения, определяющие размеры в отношении к высоте и ширине окна броузера.

Например, чтобы создать бегущую строку высотой в 50 пикселов и протянувшуюся на одну треть ширины окна, напишите:

```
<marquee height=50 width="33%">
```

Хотя в целом разумно позаботиться о достаточно большой величине значения атрибута `height`, чтобы текст бегущей строки не обрезался сверху, авторы нередко используют значение атрибута `width`, меньшее длины текста. В этом случае текст пробегает отведенное ему небольшое пространство, создавая эффект «электронного табло», знакомый большинству людей.

5.5.1.5. Атрибуты hspace и vspace

Атрибуты `hspace` и `vspace` позволяют оставлять некоторое расстояние между бегущей строкой и окружающим текстом, что делает ее более заметной.

Оба атрибута принимают целые значения, определяющие нужные расстояния в пикселах. Атрибут `hspace` оставляет промежутки над бегущей строкой и под ней. Атрибут `vspace` вводит их слева и справа от бегущей строки. Чтобы оставить промежутки в 10 пикселов со всех сторон бегущей строки, напишите:

```
<marquee vspace=10 hspace=10>
```

5.5.1.6. Атрибуты scrollamount и scrolldelay

Эти атрибуты управляют скоростью и гладкостью движения бегущей строки.

Значение атрибута `scrollamount` – это число пикселов, на которое смещается строка за одни шаг. Меньшие значения сглаживают течение текста, но и замедляют его; большие значения дают более быстрое движение, возможно, с заметными рывками.

Атрибут `scrolldelay` позволяет установить число миллисекунд между последовательными шагами. Чем меньше это число, тем быстрее бежит строка.

Можно применять небольшие значения `scrolldelay`, чтобы скомпенсировать замедление строки, вызванное маленьким значением `scrollamount`, обеспечивающим гладкость хода. Например:

```
<marquee scrollamount=1 scrolldelay=1>
```

сдвигает текст на один пиксел за один шаг, но делает это так быстро, как только возможно. В этом случае скорость движения строки ограничивается только свойствами пользовательского компьютера.

5.6. Другое мультимедийное содержимое

У сети нет никаких предрассудков относительно типов содержимого, которым могут обмениваться серверы и броузеры. В этом разделе мы рассмотрим различные способы ссылаться на изображения, аудио, видео и документы других форматов.

5.6.1. Встраивать или ссылаться?

Изображения в настоящее время занимают особое место среди различных мультимедийных данных, которые можно включать в HTML- или XHTML-документы и отображать «в строку» с другим содержимым средствами почти любого броузера. Однако, как уже обсуждалось в этой главе, иногда можно ссылаться на изображение внешним образом. Чаще всего речь идет о крупных объектах, в которых детали имеют существенное значение, но не являются сиюминутно необходимой частью содержимого документа. На другие мультимедийные элементы, включая аудио и видео, также можно ссылаться как на отдельные документы, внешние по отношению к рассматриваемому.

Обычно для ссылок на внешние мультимедийные элементы используются якоря (`<a>`). Броузер загружает и представляет мультимедийные объекты (возможно, с помощью внешних приложений или плагинов) так же, как и другие элементы, ссылку на которые выбрал пользователь. Процесс просмотра внешних элементов состоит из двух этапов: сначала загружается документ, ссылающийся на мультимедийный объект, а затем и сам объект, если была употреблена соответствующая ссылка. [тег <a>, 6.3.1]

В случае изображения вы можете решать, как представить его пользователю: встроенным в документ и непосредственно доступным при употреблении тега `<img>` или путем ссылки на него с помощью тега `<a>`. Если изображения невелики и критически важны для документа, вам следует представить их как встроенные. Если они велики или имеют второстепенное значение, обеспечьте доступ к ним через ссылку, применяя якорь (`<a>`).

Если вы решили употребить тег `<a>`, будет нелишним обозначить размер изображения, на которое ссылается документ, и, возможно, дать его маленький эскиз. Пользователь тогда сможет обоснованно решить, стоит ли времени и денег получение изображения в полном варианте.

5.6.2. Ссылки на аудио, видео и изображения

Вы ссылаетесь на любой внешний документ, независимо от его типа или формата, при помощи традиционной гиперссылки (якоря <a>):

```
<a href="sounds/anthem.au">Гимн Кумкватоводов</a> воодушевляющий на новые
трудовые подвиги тысячи кумкватоводов всего мира.
```

Сервер отправляет броузеру запрошенный мультимедийный объект так же, как и любой другой документ, едва только пользователь выбирает соответствующую гиперссылку. Если броузер обнаруживает, что полученный документ не написан на HTML или XHTML, а имеет какой-то другой формат, он автоматически вызывает подходящее средство отображения, чтобы показать на экране или каким-то иным образом донести до пользователя содержимое полученного документа.

Вы можете сконфигурировать свой броузер специальными вспомогательными приложениями, которые будут по-разному обращаться с документами разных форматов. Аудиофайлы, например, будут переданы средству обработки аудиофайлов, а видеофайлы – отправлены на видеопроигрыватель. Если броузер не сконфигурирован для обработки какого-либо формата документов, он сообщит об этом и предложит сохранить документ на диске. Позже вы сможете изучить документ с помощью необходимого средства просмотра.

Броузеры опознают форматы и соответственно обращаются с мультимедийными файлами, пользуясь одной из двух подсказок – либо MIME-типом (Multipurpose Internet Mail Extension), который указывает сервер, либо суффиксом в имени файла. Броузеры предпочитают MIME как более полное описание формата файла и его содержимого, но могут догадаться о содержимом (о его типе и формате) по расширению: *.gif* или *.jpg* для GIF- и JPEG-носителей соответственно или *.au* для специальных аудиофайлов.

Поскольку не все броузеры смотрят на MIME-тип и не все они верно сконфигурированы со вспомогательными приложениями, следует всегда использовать правильное расширение файла в названиях мультимедийных объектов (см. табл. 5.1).

5.6.3. Стиль ссылок

Создание эффективных ссылок на внешние мультимедийные документы критически важно. Пользователю нужно знать, что за объект скрывается под ссылкой и, возможно, приложение какого вида следует запустить для его обработки. Кроме того, большинство мультимедийных файлов имеют весьма солидный размер, так что обычная вежливость подсказывает нам, что нужно предупредить посетителя, во что ему обойдется требуемая загрузка.

Будет полезно включить в состав якоря или окружающего текста маленький эскиз большого изображения или всем знакомую пиктограмму, обозначающую формат объекта, на который нацелена гиперссылка.

5.6.4. Вложение документов других типов

По сети можно доставить электронный документ практически любого типа, а не только графику, аудио- или видеофайлы. Чтобы отобразить их, броузеру клиента понадобятся правильно сконфигурированные вспомогательные приложения. Современные броузеры также поддерживают вспомогательные плагины и, как это описано в главе 12, допускают расширения для выполнения специальных функций, включая встроенное отображение мультимедийных объектов.

Рассмотрим для примера компанию, обширная маркетинговая документация которой подготовлена при помощи какого-либо популярного средства макетирования, например Adobe Acrobat, FrameMaker, QuarkXPress или PageMaker, и хранится в соответствующем формате. Хотя конвертирование всех этих материалов в HTML или XHTML может оказаться слишком дорогостоящим, существует способ обеспечить распространение таких документов в сети.

Решение проблемы состоит в том, чтобы подготовить небольшое количество HTML- или XHTML-документов, которые содержат ссылки на эти файлы в альтернативных форматах и вызывают подходящий воспроизводящий эти документы апплет. Или же следует убедиться, что броузеры пользователей располагают нужными плагинами или сконфигурированы для вызова пригодных вспомогательных приложений. Adobe's Acrobat Reader, например, является очень популярным плагином. Если документ подготовлен в формате Acrobat (*.pdf*) и выбрана ссылка на Acrobat-документ, плагин запускается и соответственно отображает материал – зачастую прямо в окне броузера.

- Основы гипертекста
- Ссылки на документы: URL
- Создание гиперссылок
- Эффективное применение гиперссылок
- Изображения, реагирующие на мышь
- Создание поисковых документов
- Отношения
- Поддержка автоматической обработки и создания документов

6

Гиперссылки и Сети

До сих пор мы обращались с HTML- и XHTML-документами как с изолированными объектами, сосредоточившись на элементах языка, которые используются при их структурировании и форматировании. Однако истинная сила данных языков разметки заключается в способности объединять собрания документов в полные библиотеки и связывать эти библиотеки с другими собраниями, разбросанными по всему миру. Подобно тому как пользователь может в заметной степени управлять видом документа на своем экране, так и гиперссылки позволяют ему управлять очередностью появления документов. Буквы «HT» в аббревиатурах HTML и XHTML означают «гипертекст» («hypertext»), и именно из нитей гипертекста сплетена сеть.

6.1. Основы гипертекста

Фундаментальное свойство гипертекста состоит в том, что он позволяет связывать документы гиперссылками. Можно указать на определенное место внутри текущего документа, внутри какого-то другого, хранящегося в локальном собрании, или внутри документа, находящегося где-либо в Интернете. Документы превращаются в точки пересечения нитей информационной паутины. (Понимаете теперь, откуда взялась Всемирная паутина?) Документ, на который нацелена гиперссылка, обычно имеет какое-то отношение к источнику ссылки и обогащает его содержание; элемент-носитель гиперссылки в исходном документе должен доносить до читателя эту связь.

Гиперссылки применяются для достижения всевозможных эффектов. Их можно использовать в оглавлениях и в списках обсуждаемых вопросов. Щелчком мыши на экране или нажатием клавиши на клавиа-

туре читатели выбирают интересующую их тему и автоматически перескакивают к тому месту, где она обсуждается, в открытом ли уже документе или в каком-то ином, расположенном в совершенно другом собрании где-то на краю земли.

Кроме того, гиперссылки указывают читателю источники дополнительной информации по упомянутой теме, например «За дополнительными сведениями обращайтесь к „Кумкватам на параде“». Авторы используют гиперссылки, избегая повторения информации. В частности, мы рекомендуем подписывать все свои документы. Вместо того чтобы включать полную контактную информацию в каждый документ, свяжите гиперссылкой свое имя с одним-единственным местом, содержащим ваш адрес, номер телефона и т. д.

Гиперссылки, или, как их обычно называют, *якоря*, выделяются тегом `<a>` и бывают двух видов. Первый тип, детально описанный ниже, создает в документе область, которая инициализирует переход по гиперссылке, если она была предварительно активизирована и выбрана (обычно при помощи мыши) пользователем. Броузер автоматически загружает и отображает другой фрагмент старого либо полностью новый документ или совершает какое-то действие, связанное с Интернетом, например запускает программу, работающую с электронной почтой, или загружает из сети некий файл. Якорь другого типа создает метку, выделяющую в документе место, на которое можно указать с помощью гиперссылки.[1]

Кроме того, гиперссылки реагируют на некоторые инициируемые с помощью мыши события, обрабатывая которые с помощью JavaScript, можно совершать эффектные фокусы.

6.2. Ссылки на документы: URL

Каждый документ в Сети имеет уникальный адрес. (Вообразите-ка себе тот хаос, который бы непременно возник, если бы этого не было.) Адресом документа называют URL – *универсальный указатель ресурсов* (*Uniform Resource Locator*).

Несколько HTML/XHTML-тегов имеют атрибуты, принимающие URL в качестве значений. Среди них – гиперссылки, встроенные изображения, формы. Все они используют один и тот же синтаксис URL для определения местоположения сетевого ресурса независимо от его типа и содержимого. Вот почему в его название входит слово *uniform* (унифицированный).

---

[1] Якоря обоих типов создаются при помощи одного и того же тега. Поэтому, видимо, их и называют одинаково. На наш взгляд, их следует различать и говорить о том типе, что создает в документе специальную область, ассоциированную с адресом гиперссылки, как о «ссылке», а другой тип, отмечающий место, на которое гиперссылка может быть нацелена, так и называть якорем.

Употребляемые для представления почти любых ресурсов Интернета, URL бывают нескольких типов. Тем не менее все URL имеют общий синтаксис:

```
scheme:scheme_specific_part
```

Здесь *scheme* (схема) описывает тип объекта, на который указывает URL, а *scheme_specific_part* – это специфичная для каждой схемы часть. Необходимо отметить, что *scheme* всегда отделяется от *scheme_specific_part* двоеточием без пробелов между ними.

6.2.1. Как записывается URL

Записывайте URL, применяя отображаемые элементы набора символов US-ASCII. Вот пример простого URL:

```
http://www.blah-blah.com[1]
```

Если нужно включить в URL символ, не являющийся элементом набора US-ASCII, то придется использовать его кодировку с помощью специальных обозначений. Код символа заменяет нужный символ тремя: знаком процента и двумя шестнадцатеричными цифрами, значение которых соответствует номеру символа в наборе ASCII.[2]

Все это проще, чем кажется. Один из самых обычных специальных символов, пробел (пользователям Macintosh следует отметить это для себя особо), имеет своим номером в ASCII шестнадцатеричное[3] число 20. Нельзя вставить пробел в URL (то есть можно, но ничего хорошего из этого не выйдет). Замените его в URL на `%20`:

```
http://www.kumquat.com/new%20pricing.html
```

---

[1] Здесь *http:* – то, что автор называет «схемой»; иногда об этой составляющей говорят как о «протоколе». А *//www.blah-blah.com* – специфическая часть, адрес ресурса, в данном конкретном случае – адрес сервера. Таким образом, следующие друг за другом символы *://* (практически повсеместный случай в веб-адресах) не имеют какого-либо сакрального смысла: это компоненты двух разных составляющих URL, что не всегда сразу очевидно. – *Примеч. науч. ред.*

[2] Вот как выглядит URL, в записи которого использованы русские литеры: *http://www.trezvo-koms.org/modules.php?name=Downloads&d_op=viewdownloaddetails&lid=27*&ttitle=%D0%E5%EB%E8%E3%E8%FF_%E4%E5%ED%E5%E3_-_%C0%E2%F2%EE%F0:%C4%EC%E8%F2%F0%E8%E9_%CD%E5%E2%E5%E4%E8%EC%EE%E2 – все это, естественно, записано в одну строку... Такая запись: а) примерно в 3 раза длиннее эквивалента, записанного латинскими литерами, б) абсолютно нечитаема. Именно поэтому в записи URL стараются всячески избегать кириллических литер, заменяя их латинскими эквивалентами. – *Примеч. науч. ред.*

[3] Шестнадцатеричная нумерация основана на 16 символах: цифрах от 0 до 9 и буквах от A до F, шестнадцатеричные значения которых соответствуют десятичным от 0 до 15. При этом строчные буквы не отличаются от заглавных: «a» (шестнадцатеричное 10) имеет тот же смысл, что и «A».

Данный URL позволяет получить документ *new pricing.html* с сервера *www.kumquat.com*.

6.2.1.1. Зарезервированные и небезопасные символы

В дополнение к неотображаемым символам вам придется также использовать коды, вставляя в URL зарезервированные и небезопасные символы.

Зарезервированными называются символы, имеющие в самом URL специальный смысл. Наклонная черта, к примеру, в URL служит для отделения друг от друга элементов пути. Если нужно включить в поле URL символ наклонной черты, который не должен служить разделителем элементов, следует закодировать его последовательностью `%2F`:

```
http://www.calculator.com/compute?3%2f4
```

Этот URL ссылается на ресурс, именуемый *compute*, на сервере *www.calculator.com* и передает ему строку `3/4`, предваренную вопросительным знаком (`?`). Предположительно, ресурс представляет собой программу со стороны сервера, производящую над переданным значением некоторые арифметические действия и возвращающую результат.

Небезопасными являются символы, которые, хотя и не имеют специального значения в URL, могут иметь особый смысл в контексте, включающем указатель ресурса. К примеру, URL, являющийся значением атрибута некоего тега, выделяется двойными кавычками (`" "`). Если бы вам случилось включить в URL двойные кавычки как таковые, то, вероятно, броузер оказался бы в замешательстве. Вместо этого, во избежание возможных конфликтов, следует закодировать символ двойной кавычки последовательностью `%22`.

В табл. 6.1 приведены зарезервированные и небезопасные символы, которые всегда необходимо кодировать.

Таблица 6.1. Зарезервированные и небезопасные символы и их кодировки в URL

Символ	Описание	Употребление	Код
;	Точка с запятой	Зарезервирован	%3B
/	Наклонная черта	Зарезервирован	%2F
?	Вопросительный знак	Зарезервирован	%3F
:	Двоеточие	Зарезервирован	%3A
@	Символ «at»	Зарезервирован	%40
=	Знак равенства	Зарезервирован	%3D
&	Амперсанд	Зарезервирован	%26
<	Знак «меньше»	Небезопасный	%3C
>	Знак «больше»	Небезопасный	%3E
"	Двойная кавычка	Небезопасный	%22

Таблица 6.1 (продолжение)

Символ	Описание	Употребление	Код
#	Решетка	Небезопасный	%23
%	Процент	Небезопасный	%25
{	Левая фигурная скобка	Небезопасный	%7B
}	Правая фигурная скобка	Небезопасный	%7D
|	Вертикальная черта	Небезопасный	%7C
\	Обратный слэш	Небезопасный	%5C
^	Знак вставки	Небезопасный	%5E
~	Тильда	Небезопасный	%7E
[	Левая квадратная скобка	Небезопасный	%5B
]	Правая квадратная скобка	Небезопасный	%5D
`	Обратная одиночная кавычка	Небезопасный	%60

В целом, применять коды символов следует всегда, когда есть сомнение, допустимо ли их помещать в URL. Мы советуем кодировать все, что не является цифрой, буквой или одним из символов: $ - _ . + ! * ' ( ).

Кодирование символа не является ошибкой, если только он не имеет специального значения в URL. В частности, закодировав слэши в URL, вы заставите броузер думать, что это обычные символы, а не разделители элементов пути, что приведет к разрушению URL. Аналогичным образом кодирование амперсанда, используемого в качестве разделителя параметров в URL-адресе, аннулирует это его предназначение. Записывайте амперсанды в форме &, и вы достигнете своей цели.

6.2.2. Абсолютные и относительные URL

Можно записывать URL-адреса в одном из двух видов – *абсолютном* или *относительном*. Абсолютный URL – это полный адрес ресурса, содержащий все, что нужно системе, чтобы найти документ и его сервер в сети. Абсолютный URL содержит как минимум схему и все обязательные элементы части адреса, именуемой *scheme_specific_part*. Он может также содержать любые необязательные составляющие этой части.

При помощи относительного URL вы указываете сокращенный адрес документа, который, автоматически комбинируясь с «базовым адресом», становится полным адресом документа. В относительном URL любой его компонент может быть опущен. Броузер автоматически заполняет все пропущенные элементы относительного адреса, используя соответствующие компоненты базового URL. Этот базовый URL обычно является URL документа, содержащего относительный URL, но может быть и другим, определенным в документе при помощи тега <base>, что и обсуждается далее в этой главе. [элемент заголовка <base>, 6.7.1]

6.2.2.1. Относительные схемы и серверы

Обычно в относительном URL схема и имя сервера опущены. Поскольку множество документов, на которые приходится ссылаться, расположены на том же сервере, кажется разумным поступить именно так. К примеру, предположим, что базовый документ был получен с сервера *www.kumquat.com*, тогда относительный URL:

```
another-doc.html
```

будет эквивалентен абсолютному URL:

```
http://www.kumquat.com/another-doc.html
```

Нетрудно понять, как базовый и относительный URL из примера комбинируются при формировании абсолютного URL (табл. 6.2).

Таблица 6.2. Формирование абсолютного URL

	Протокол	**Сервер**	**Каталог**	**Файл**
Базовый URL	http	*www.kumquat.com*	/	
Относительный URL	↓	↓	↓	*another-doc.html*
↓	↓	↓	↓	↓
Абсолютный URL	http	*www.kumquat.com*	/	*another-doc.html*

6.2.2.2. Относительный каталог документа

Другая распространенная форма относительного URL не содержит ведущего слэша и одного или нескольких названий каталогов в начале пути к документу. Каталог базового URL автоматически подставляется на место опущенных компонентов. Это самое распространенное сокращение, поскольку большинство авторов размещают свои собрания документов и подкаталоги со вспомогательными ресурсами в каталоге с тем же путем, что и у домашней страницы. У вас, например, может существовать подкаталог *special*, содержащий FTP-файлы, на которые ссылается документ. Допустим, абсолютный URL этого документа таков:

```
http://www.kumquat.com/planting/guide.html
```

Относительный URL файла *README.txt* в подкаталоге *special* такой:

```
ftp:special/README.txt
```

В действительности вы запрашиваете:

```
ftp://www.kumquat.com/planting/special/README.txt
```

Эта процедура отображена в табл. 6.3.

Таблица 6.3. Формирование абсолютного FTP URL

	Протокол	Сервер	Каталог	Файл
Базовый URL	http	*www.kumqu at.com*	*/planting*	*guide.html*
Относительный URL	ftp	↓	*special*	*README.txt*
↓	↓	↓	↓	↓
Абсолютный URL	ftp	*www.kumqu at.com*	*/planting/special*	*README.txt*

6.2.2.3. Использование относительных URL

Относительные URL не просто облегчают набор ссылок на клавиатуре. Поскольку они связаны с текущим сервером и каталогом, можно переместить весь набор документов в другой каталог или даже на другой сервер без необходимости менять хотя бы одну ссылку. Вообразите только те трудности, которые пришлось бы преодолевать, если бы нужно было заглянуть в каждый исходный документ и исправить URL для всех гиперссылок при любом перемещении коллекции.[1] От слова «гиперссылки» вас бы выворачивало наизнанку! Используйте относительные URL везде, где это возможно.

6.2.3. URL типа http

URL типа http до сих пор является самым распространенным. Он используется для получения доступа к документам, расположенным на веб-серверах, и имеет два формата:

```
http://сервер:порт/путь#фрагмент
http://сервер:порт/путь?поиск
```

Некоторые из частей могут быть опущены. Самый обычный URL:

```
http://сервер/путь
```

указывает на уникальный сервер, путь к документу и имя документа.[2]

---

[1] Более того, вы можете (и часто так и делаете, не задумываясь об этом) загрузить всю коллекцию связанных с базовым URL документов в файловую локальную систему – теперь при обращениях к локальной копии документа все относительные ссылки получат протокольную часть file:, что дает возможность связно просматривать локальную иерархию документов. – *Примеч. науч. ред.*

[2] Имя документа как раз в этом примере отсутствует – это обычная практика, т. к. администратор веб-ресурса может определить имя документа по умолчанию, поиск которого начнется при отсутствии явно указанного имени, например: *index.html* или *home.html*. – *Примеч. науч. ред.*

6.2.3.1. Http-сервер

Сервер – это компьютерная система, хранящая и выдающая по внешнему запросу веб-ресурс и обладающая уникальным в Интернете IP-адресом. Мы подозреваем, что в своих URL вы будете чаще пользоваться не IP-адресами, состоящими из особой последовательности цифр, а соответствующими этим адресам именами серверов, которые легче запоминаются.[1]

Имя состоит из нескольких частей, включая собственное имя сервера и последовательность доменных имен, отделенных друг от друга точками. Типичное имя в Интернете выглядит как *www.oreilly.com* или *hoohoo.ncsa.uiuc.edu*.[2]

Стало уже традицией, что веб-мастера именуют свои серверы *www* для создания простого и ясного идентификатора в сети. К примеру, O'Reilly Media имеет сервер с именем *www*, который вместе с доменным именем издательства становится легко запоминаемым веб-сайтом *www.oreilly.com*. Подобным образом MobileRobots располагает сервером с именем *www.mobilerobots.com*. Являясь некоммерческой организацией, консорциум World Wide Web Consortium имеет сервер с другим доменным суффиксом: *www.w3c.org*. Традиционный способ формирования имен дает очевидные выгоды, которыми вам тоже следует воспользоваться, если придется создавать веб-сервер для своей организации.

Вы также можете указать адрес сервера, применяя его IP-адрес. Он представляет собой последовательность четырех разделенных точками чисел, от 0 до 255 каждое. Реальный IP-адрес выглядит примерно так: 137.237.1.87 или 192.249.1.33.

Было бы скучно теперь рассказывать, что обозначает каждое число или как, зная доменное имя, вычислять IP-адрес, тем более, что вам

---

[1] Каждый подключенный к Интернету компьютер имеет уникальный, разумеется, числовой (IP) адрес, поскольку компьютеры работают только с числами. Люди предпочитают имена, поэтому в Интернете действуют специальные серверы и программы системы доменных имен (Domain Name System, DNS), которые автоматически преобразуют интернет-имена в IP-адреса. Некоммерческое агентство InterNIC регистрирует имена в основном по принципу «первым пришел, первым обслужен» и распространяет их по DNS-серверам во всем мире.

[2] Трехбуквенный суффикс в доменном имени обозначает тип организации или компании, действующей в Интернете. К примеру, *.com* – это коммерческое предприятие, *.edu* – научное или образовательное учреждение, *.gov* имеет отношение к правительству. Помимо вышеуказанных, за пределами США суффиксы присваиваются в основном по географическому признаку. Обычно это двухбуквенные сокращения названий стран, например «jp» – для Японии и «de» – для Германии. В настоящее время широкое распространение получили трехбуквенные национальные суффиксы вместо традиционных двухбуквенных.

редко, если вообще когда-нибудь, придется использовать их в URL. Лучше здесь поместить гиперссылку: возьмите какое-нибудь хорошее руководство в области интернет-технологий, чтобы получить строгое изложение IP-адресации, такое как «The Whole Internet User's Guide and Catalog» (Общее руководство и каталог для пользователей Интернета) Эда Крола (Ed Krol) (O'Reilly).

6.2.3.2. Http-порт

Порт – это номер коммуникационного порта сервера, через который подключается броузер клиента. Это понятие из области сетевых протоколов: серверы исполняют много различных функций – помимо предоставления веб-документов и других веб-ресурсов броузерам клиентов они отправляют и получают электронную почту, осуществляют передачу документов с помощью FTP, работают в локальной сети и т. д. Хотя весь поток информации, связанной с этой деятельностью, может передаваться по одному кабелю, он обычно подразделяется на программно управляемые «порты»[1], каждый из которых пропускает через себя сообщения, соответствующие его специализации. Это похоже на абонентские ящики в местном отделении связи.

По умолчанию для веб-серверов используется порт 80. Для защищенной передачи данных – Secure HTTP (SHTTP) или Secure Socket Layer (SSL) – употребляется порт 443. В настоящее время большинство веб-серверов применяют порт 80. Вы должны включить номер порта вместе с непосредственно предшествующим ему двоеточием в URL, если указанный в нем сервер не использует восьмидесятый порт для коммуникаций в сети.

Когда сеть находилась в младенчестве, ее пионеры осуществляли свои «необузданные» (Wild Wild Web, сильно одичавшая паутина) соединения по самым разным номерам портов. Из технических представлений и соображений безопасности необходимы полномочия системного администратора, чтобы установить сервер на восьмидесятый порт. Не обладая таким уровнем ответственности, веб-мастера прошлого выбирали другие, более доступные номера портов.

В наши дни, когда веб-серверы приобрели черты цивилизованности и находятся под заботливой опекой ответственных администраторов, тот факт, что документы обслуживаются на порту с номером, отличным от 80 или 443, может служить основанием для беспокойства. Вероятнее всего, такой нетипичный сервер ведет без ведома настоящего администратора какой-то умный пользователь.[2]

---

[1] Порт TCP/IP (а именно над этим протоколом работает HTTP) – это чисто логическая абстракция. – *Примеч. науч. ред.*

[2] Изменение (администрирование) параметров TCP/IP портов с номерами ниже 1024 требует полномочий суперпользователя (root) в операционной системе. – *Примеч. науч. ред.*

6.2.3.3. Http–путь

Путь – это записанная в стиле Unix[1] иерархическая последовательность, указывающая местоположение документа на сервере. Путь состоит из одного или нескольких имен, разделенных наклонной чертой. Все такие имена, кроме последнего, обозначают каталоги, предшествующие документу. Последнее имя – это обычно имя самого документа, хотя типичный веб-сервер в качестве умолчания принимает имя файла *index.html*.

Стало традицией для более легкой идентификации давать HTML-документам имена, завершающиеся суффиксом *.html* (вообще-то они являются просто ASCII-файлами, помните?). Хотя последние версии Windows допускают длинные суффиксы, разработчики со стажем часто по-прежнему пишут трехбуквенный суффикс (расширение файла) *.htm* в именах HTML-документов.

Несмотря на то что имя сервера в URL нечувствительно к регистру, к пути это может не относиться. Поскольку большинство веб-серверов работают под Linux, а имена файлов в этой операционной системе чувствительны к регистру, путь к документу будет обладать тем же свойством. Веб-серверы, работающие под Windows, не различают строчных и прописных букв, следовательно, и путь их не спутает. Тем не менее, поскольку невозможно знать, какая операционная система установлена на сервере, с которым вы соединяетесь, всегда предполагайте, что сервер такой чувствительностью обладает, и позаботьтесь при наборе URL о правильности регистров.

Возникло несколько соглашений, касающихся путей. Если последним элементом пути является каталог, а не один документ, сервер обычно возвращает или список его файлов, или HTML-документ, содержащий данный список. Если вы обращаетесь к каталогу, следует завершать путь наклонной чертой, но на практике большинство серверов исполнят запрос, даже если этот символ будет опущен.

Если имя каталога – это просто одна наклонная черта или совсем ничего, сервер сам решит, что отправить броузеру. Как правило, это будет так называемая *домашняя страница*, хранящаяся в корневом каталоге под именем *index.html*. Всякий хорошо спроектированный веб-сервер должен иметь привлекательную, хорошо оформленную домашнюю (главную) страницу. Это удобный способ предоставить пользователям доступ к вашей коллекции, поскольку им не нужно помнить точное название документа, достаточно знать имя сервера. Вот почему, к примеру, вы можете набрать в диалоговом окне «Open» Netscape *http://www.oreilly.com* и получить домашнюю страницу издательства O'Reilly.

---

[1] «В стиле UNIX» означает, что элементы пути (имена каталогов) разделяются прямым слэшем: /. В Windows для записи путей используется разделитель обратного слэша: \. – *Примеч. науч. ред.*

Еще один момент: если первый компонент пути начинается с тильды (~), это означает, что остальная часть пути начинается с персонального каталога в домашнем каталоге указанного пользователя на сервере. К примеру, URL *http://www.kumquat.com/~chuck/* запрашивает страницу верхнего уровня из собрания документов, принадлежащего человеку с именем Chuck.

Разные серверы по-разному ищут документы в домашнем каталоге пользователя. Многие будут искать их в каталоге под названием *public_html*. Серверы под Unix любят название *index.html* для домашних страниц. Когда все это не срабатывает, серверы выдают содержание каталога или HTML-документ из домашнего каталога, заданный по умолчанию.[1]

6.2.3.4. Http–фрагмент

Фрагмент – это идентификатор, указывающий на определенный раздел документа. В записи URL он следует за именем сервера и путем и отделяется от них символом решетки (#). Идентификатор фрагмента сообщает броузеру, что он должен начать отображение указанного в URL документа с места, обозначенного именем фрагмента. Как это детально описано далее в главе, вставить название фрагмента в документ можно при помощи либо универсального атрибута `id`, либо атрибута `name` тега `<a>`. В следующем примере броузер загружает файл по имени *kumquat_locations.html* с сервера *www.kumquat.com*, а затем выводит документ, начиная с раздела, названного Northeast:

```
http://www.kumquat.com/kumquat_locations.html#Northeast
```

Как и путь, имя фрагмента может представлять собой любую последовательность символов, если вы внимательно обращаетесь с пробелами и другими специальными символами.

Название фрагмента и предшествующий ему символ решетки не обязательны. Пропускайте их, когда ссылаетесь на документ, в котором фрагменты не определены.

Формально говоря, фрагмент в URL применим только к HTML- или XHTML-документам. Если URL указывает на документ другого типа, название фрагмента может быть неправильно понято броузером.

Фрагменты полезны в случае длинных разработок. Снабжая идентификаторами ключевые разделы, вы облегчаете пользователю непосредственный доступ к ним, позволяя избегать скучного прокручивания документа в поисках интересующего их материала.

---

[1] Имя файла, выдаваемого по умолчанию, обычно указывается в настройках сервера. Как правило, если файл с таким именем (или именами – их может быть несколько) отсутствует, то, если разрешен просмотр содержимого данного каталога, посетителю будет выдан список содержимого указанной папки, в противном случае выдается сообщение об ошибке. Вероятно, авторы встречали серверы, настроенные столь странно, что в подобных случаях возвращают посетителя на главную страницу. – *Примеч. науч. ред.*

Мы рекомендуем сопровождать каждый заголовок раздела эквивалентным названием фрагмента. Систематически следуя этому правилу, вы дадите читателю возможность перескакивать к любому разделу в любом из ваших документов. Кроме того, фрагменты облегчают создание оглавлений для ваших коллекций.

6.2.3.5. Параметр поиска в http

Поисковый компонент в http URL с предшествующим ему вопросительным знаком не является обязательным. Он указывает, что путь ведет к производящему поиск или другому исполняемому на сервере ресурсу. Содержимое поискового компонента передается серверу и применяется в качестве параметра поиска или аргумента запускаемого сценария или программы.

Способ кодировки параметров в поисковом компоненте зависит от сервера и ресурса, на который ссылается URL. Параметры для ресурсов, допускающих поиск, рассматриваются далее в этой главе при обсуждении поисковых документов. Параметры для исполняемых ресурсов рассматриваются в главе 9.

Хотя до сих пор описание http URL устанавливало, что в URL может входить либо поисковый компонент, либо идентификатор фрагмента, некоторые броузеры позволяют использовать их вместе.[1] Если вам так хочется, поставьте поисковый компонент после названия фрагмента, предлагая броузеру начать отображение результатов поиска с обозначенного места. Netscape, например, поддерживает такое употребление URL.

Мы, однако, не рекомендуем такой вид URL. Первое и самое главное – это работает не со всеми броузерами. Не менее важное соображение: использование фрагмента подразумевает вашу уверенность, что в найденном в результате поиска документе есть раздел с указанным именем. Для больших коллекций это маловероятно. Лучше опустить фрагмент, показывая результаты поиска с начала документа и избегая тем самым возможного замешательства читателей.

6.2.3.6. Образцы http URL

Вот несколько примеров http URL:

```
http://www.oreilly.com/catalog.html
http://www.oreilly.com/
http://www.kumquat.com:8080/
```

[1] Более того, согласно RFC1630 (Request for Comments), в котором описывается синтаксис уникальных идентификаторов ресурсов (Uniform Resource Identifier) в целом и URL и URN в частности, предполагается совместное использование фрагментов и поисковых компонентов в URL. Дело в том, что идентификатор фрагмента не передается серверу, а применяется броузером после того, как получен ответ с серверной стороны. – *Примеч. науч. ред.*

```
http://www.kumquat.com/planting/guide.html#soil_prep
http://www.kumquat.com/find_a_quat?state=Florida
```

Первый пример – это явная ссылка на обыкновенный HTML-документ, именуемый *catalog.html*, хранящийся в корневом каталоге сервера *www.oreilly.com*.

Во втором фигурирует ссылка на домашнюю страницу самого высокого уровня на этом же сервере. Эта страница может оказаться (или не оказаться) документом *catalog.html*.

В третьем примере также предполагается найти домашнюю страницу в корневом каталоге сервера *www.kumquat.com*, причем сетевое соединение должно устанавливаться по нестандартному порту 8080.

В четвертом запрос на URL исходит из веб-документа под названием *guide.html* из каталога *planting* на сервере *www.kumquat.com*. Получив документ, броузер должен начать его отображение с фрагмента *soil_prep*.

В последнем примере вызывается исполняемый ресурс под названием *find_a_quat* с параметром, именуемым *state*, которому присвоено значение *Florida*. Предполагается, что этот ресурс генерирует ответ об успехах разведения кумкватов во Флориде в формате HTML или XHTML, и этот ответ будет отображен броузером.

6.2.4. URL типа file

Файловый URL, пожалуй второй по частоте употребления, еще не получил заслуженного признания среди веб-пользователей и особенно веб-авторов. Он указывает на файл, хранящийся на компьютере, без регламентирования протокола, применяемого для его загрузки. В таком виде он имеет ограниченное употребление в сетевом окружении. Файловый URL полезен, например, тем, что позволяет вам загружать и выводить на экран локальный документ и крайне удобен для ссылок на личную коллекцию HTML/XHTML-документов, в частности тех, что находятся в процессе разработки и еще не готовы для общего употребления, и для хранящихся на CD-ROM. Он имеет следующий формат:

```
file://сервер/путь
```

6.2.4.1. File-сервер

File-*сервер*, как и вышеописанный http-сервер, должен быть доменным именем или IP-адресом компьютера, содержащего скачиваемый файл. Однако, в отличие от http-сервера, для работы которого требуется сеть с протоколом Transmission Control Protocol/Internet Protocol (TCP/IP), файловый сервер может быть неквалифицированным, но уникальным именем компьютера в личной сети, или устройством хранения информации на том же компьютере (например, CD-ROM), или отображением другого компьютера в сети. Никаких предположений относительно того, как броузер соединяется с этой машиной, не дела-

ется. По-видимому, чтобы получить файл, броузер может установить какое-то соединение, скажем через Network File System[1] или FTP.

Если вы опустите имя сервера, добавив дополнительную косую в URL, или если вы зададите специальное имя *localhost*, броузер прочитает файл с компьтера, на котором он работает. В этом случае броузер добирается до файла, применяя обычные средства операционной системы. Как правило, file URL употребляется именно так. Расположив семейство документов на дискете или CD-ROM, а затем ссылаясь на них с помощью указателя *file:///*, вы создаете самостоятельное, пригодное для распространения собрание документов, не требующее для его использования никакого подключения к сети.

6.2.4.2. Путь в URL типа file

Это путь к запрашиваемому файлу на указанном сервере. Синтаксис пути может различаться в зависимости от операционной системы на сервере. Не забудьте закодировать в нем все потенциально опасные символы.

6.2.4.3. Образцы file URL

Форма file URL записывается просто:

```
file://localhost/home/chuck/document.html
file:///home/chuck/document.html
file://marketing.kumquat.com/monthly_sales.html
file://D:/monthly_sales.html
```

Первый URL запрашивает */home/chuck/document.html* с локального компьютера пользователя, с текущего устройства хранения данных (на компьютере с ОС Windows это, скорее всего, *C:*). Второй совпадает с первым, только в нем опущено имя сервера *localhost*. По умолчанию назначается имя локального диска.[2]

Третий пример использует любой доступный протокол для получения файла *monthly_sales.html* с сервера *marketing.kumquat.com*, а четвертый читает тот же файл с диска *D:* при помощи стандартных средств операционной системы.

6.2.5. URL для mailto

URL типа mailto очень распространен в HTML/XHTML-документах. Он заставляет броузер отправить почтовое сообщение указанному адресату. Он имеет формат:

---

[1] Network File System (NFS) – сетевая файловая система – набор протоколов, позволяющих компьютерам с различными операционными системами обмениваться файлами в локальной сети. – *Примеч. науч. ред.*

[2] Это пример для Unix-систем. В Windows-системах, если документ расположен на диске C, то соответствующий file URL будет: *file://c:\home\chuck\document.html*. – *Примеч. науч. ред.*

```
mailto:адрес
```

Здесь *адрес* – это любой допустимый адрес электронной почты, обычно имеющий форму:

```
user@server
```

Следовательно, типичный mailto URL может выглядеть так:

```
mailto:chuckandbill@kumquats.com
```

Вы можете указать в mailto URL несколько получателей с адресами, отделяемыми запятыми. К примеру, следующий URL определяет трех получателей:

```
mailto:chuck@kumquats.com,bill@kumquats.com,booktech@ora.com
```

Перед запятыми и после них в URL не должно быть пробелов.

6.2.5.1. Заполнение полей заголовка в письме

Популярные броузеры открывают почтовый клиент, когда пользователь выбирает гиперссылку с URL для mailto. Это может быть программа электронной почты, установленная в системе по умолчанию, или широко распространенное приложение, такое как Outlook Express, созданное для броузера Internet Explorer, или Communicator фирмы Netscape. Некоторые броузеры позволяют пользователям указывать почтовую программу, настраивая опции броузера.

Подобно тому, как вы добавляете в конец URL-адреса параметры http-поиска, разделенные вопросительными знаками, вы включаете в HTML-документ параметры электронной почты вместе с URL. В типичном случае дополнительные параметры определяют заголовочные поля сообщения, такие как тема или адреса получателей копий. Обработка этих дополнительных полей перекладывается на почтовую программу.

Рассмотрим несколько примеров:

```
mailto:chuckandbill@kumquats.com?subject=Loved your book!
mailto:chuck@kumquats.com?cc=booktech@oreilly.com
mailto:bill@kumquats.com?bcc=archive@myserver.com
```

Как вы, вероятно, догадались, первый URL устанавливает тему (subject) сообщения. Обратите внимание, что некоторые почтовые программы позволяют вставлять пробелы в значения параметров, а другие – нет. Особенно раздражает то, что пробелы невозможно заменять их шестнадцатеричным эквивалентом %20, потому что многие почтовые программы неспособны выполнить корректную подстановку. Безопаснее использовать пробелы, так как почтовые программы, которые их запрещают, просто отсекут от параметра все, начиная с пробела.

Второй URL помещает адрес booktech@oreilly.com в поле cc нашего сообщения. Аналогично определяется содержимое поля bcc в сообщении последнего примера. Кроме того, можно установить несколько полей в одном URL, разделяя присваивания амперсандами. В частности:

```
mailto:cmusciano@aol.com?subject=Loved your book!&cc=booktech@
oreilly.com&bcc=archive@myserver.com
```

устанавливает тему и адрес для копии. Не все почтовые программы распознают поля bcc и cc в URL mailto: некоторые их игнорируют, а другие присоединяют к предшествующему полю темы. Поэтому, формируя URL для mailto, располагайте дополнительные поля в таком порядке: вначале тема, а за ней – cc и bcc. И не ставьте себя в зависимость от того, будут ли адреса получателей копий включены в почтовое сообщение.

6.2.6. URL типа ftp

Указатель ресурса типа ftp (ftp URL) используется для получения документов с FTP-серверов[1] и имеет следующий формат:

```
ftp://пользователь:пароль@сервер:порт/путь;тип=код_типа
```

6.2.6.1. Пользователь и пароль в URL типа ftp

FTP (File Transfer Protocol) – это служба, требующая аутентификации. Это значит, что если вы хотите получить документы с сервера, вы, вообще говоря, должны располагать зарегистрированным именем пользователя и паролем. Однако большинство FTP-серверов поддерживают также ограниченный доступ к своему содержимому, известный как *анонимный FTP*, для всех желающих. В этом режиме всякий может указать имя пользователя «anonymous» или «guest» и получить доступ к ограниченной части документов сервера. Большинство FTP-серверов также поймут пропущенные имя и пароль как попытку получить анонимный доступ (которого могут не дать).

Если вы применяете ftp URL для сайта, требующего аутентификацию, то можете включить в URL компоненты *пользователь* и *пароль*, употребив двоеточие (:) и символ @. Если вы оставите имя пользователя вместе с символом @, но опустите пароль и предшествующее ему двоеточие, большинство броузеров предложат ввести пароль после соединения с FTP-сервером. Рекомендуется именно этот способ получения доступа к FTP-ресурсам, требующим аутентификации, поскольку он не позволяет другим видеть ваш пароль.

Мы советуем *никогда* не помещать ftp URL с именем пользователя и паролем ни в какой HTML/XHTML-документ. Причина проста – любой может получить ваши материалы, извлечь из URL аутентификационные данные, зарегистрироваться на FTP-сервере и начать хулиганить с лежащими на нем документами.

---

[1] FTP – это древний протокол Интернета, относящийся к непросвещенным временам около 1975 г. Он был разработан как простой способ передачи файлов с машины на машину и остается популярным и полезным до сих пор. Те, у кого нет возможности завести настоящий веб-сервер, выкладывают свои документы на FTP-сервер.

6.2.6.2. Сервер и порт в URL типа ftp

Сервер и *порт* в ftp добавляются к ftp URL по тем же правилам, по каким записываются сервер и порт в http URL. Компонент «сервер» должен быть значимым доменным именем или IP-адресом. Необязательный компонент «порт» определяет номер порта, по которому FTP-сервер ожидает запросы.

Если компонент «порт» и предшествующее ему двоеточие опущены, по умолчанию используется порт 21.

6.2.6.3. Путь и код типа передачи в ftp URL

Компонент *путь* в ftp URL представляет собой последовательность каталогов, разделенных символом наклонной черты, ведущей к запрашиваемому файлу. По умолчанию файл передается как двоичный. Это положение можно изменить, добавив к URL *код типа передачи*, перед которым должно стоять `;type=`.

Если установлен код типа `d`, предполагается, что путь – это название каталога. Броузер запросит у сервера листинг содержимого каталога и представит его пользователю. Если в качестве кода типа передачи указана другая буква, она используется как параметр FTP команды *type* для получения файла, на который указывает путь. Хотя некоторые FTP-серверы могут использовать другие коды, большинство из них поймут тип `i` как приказ начать передачу двоичного файла, а тип `a` – как указание на то, что передаваемый файл является потоком ASCII-текста.

6.2.6.4. Образцы ftp URL

Вот несколько образцов ftp URL:

```
ftp://www.kumquat.com/sales/pricing
ftp://bob@bobs-box.com/results;type=d
ftp://bob:secret@bobs-box.com/listing;type=a
```

Первый пример запрашивает файл под названием *pricing* из каталога *sales* на анонимном FTP-сервере на *www.kumquat.com*.[1] Второй регистрируется на FTP-сервере *bobs-box.com* в качестве пользователя *bob*, получая, вероятно, запрос на ввод пароля, до того как принять оглавление каталога *results* и представить его на обозрение. Последний пример регистрирует пользователя *bob* с паролем *secret* на *bobs-box.com* и получает файл *listing*, содержимое которого интерпретируется как ASCII-текст.

---

[1] Это не тот же самый сервер, к которому мы обращались в примере с http URL, хотя их адреса совпадают. Не забудьте, что http- и ftp-серверы, как правило, работают на разных портах (80 и 21 соответственно) и обычно предоставляют доступ к различным ресурсам (хотя эти серверы можно настроить так, чтобы они указывали на одни и те же каталоги и документы). – *Примеч. науч. ред.*

6.2.7. JavaScript URL

JavaScript URL – это в действительности псевдопротокол, не слишком часто упоминаемый при обсуждении URL. Ориентируясь на передовые броузеры, такие как Netscape, Opera, Firefox и Internet Explorer, вы можете указать в качестве цели гиперссылки JavaScript URL и использовать такую гиперссылку для запуска команд JavaScript. Хотя такие приемы работают, мы не рекомендуем вам прибегать к ним. Пользуйтесь атрибутом `onclick`, чтобы ассоциировать команды JavaScript с элементами ваших документов.

6.2.7.1. Аргументы JavaScript URL

То, что следует за псевдопротоколом, является не чем иным, как одним или несколькими выражениями и методами JavaScript, отделяемыми друг от друга точкой с запятой, среди которых могут быть также ссылки на функции JavaScript, включаемые в документ при помощи тега `<script>` (см. главу 12). К примеру:

```
javascript:window.alert('Привет, мир!')
javascript:doFlash('red', 'blue'); window.alert('Не нажимай сюда больше!')
```

являются правильными URL, которые можно использовать как значения соответствующих атрибутов в гиперссылках (см. разд. 6.3.1.2). Первый пример содержит один метод JavaScript, вызывающий диалоговое окно с простым сообщением «Привет, мир!», если пользователь не запретил выполнение сценариев JavaScript на своем броузере.

Следующий пример содержит два аргумента. Первый вызывает JavaScript-функцию `doFlash`, которую вы, видимо, поместили где-то в документе, заключив ее в тег `<script>`, и которая, скорее всего, расцвечивает фон документа вспышками красного и синего цветов. Второе выражение – это тот же метод, что и в предыдущем примере, но с чуть другим сообщением.

JavaScript URL может появиться в гиперссылке и без аргументов. В этом случае броузер откроет специальный редактор для JavaScript (если таковой установлен), в котором пользователь вправе набрать и протестировать различные выражения и методы.

6.2.8. URL для news

Хотя теперь он используется довольно редко, указатель ресурса типа news (news URL) запрашивает либо отдельное сообщение, либо целую группу новостей, принадлежащих системе телеконференций Usenet. Он имеет соответственно две формы:

```
news:группа_новостей
news:id_сообщения
```

Неприятное ограничение, связанное с news URL, состоит в том, что в нем нельзя указывать сервер *новостей*. Вместо этого пользователи

назначают применяемый ими сервер новостей в настройках броузера. В былые времена, не такие уж и давние, новости в Интернете распространялись почти универсальным образом – везде содержались одни и те же группы новостей и связанные с ними статьи, так что каждый сервер был не хуже любого другого. В наши дни для хранения всех поступающих за день новостей одному серверу не хватит места на диске, к тому же всюду осуществляется фильтрация сообщений или модерирование. Стало быть, не приходится даже надеяться, что все группы новостей и тем более все связанные с ними статьи будут представлены на сервере новостей пользователя.

Многие пользовательские броузеры могут не быть сконфигурированы надлежащим для чтения новостей образом. Мы советуем, за исключением редчайших случаев, избегать news URL в ваших документах.

6.2.8.1. Запрос на полную группу новостей

Существуют тысячи групп новостей, посвященных всем мыслимым и немыслимым темам. Каждая группа имеет уникальное имя, составленное из элементов иерархии, разделенных точками.

К примеру, группа новостей, содержащая объявления из мира World Wide Web, называется:

```
comp.infosys.www.announce
```

Чтобы получить к ней доступ, используйте следующий URL:

```
news:comp.infosys.www.announce
```

6.2.8.2. Запрос на отдельное сообщение

Каждое сообщение на сервере новостей обладает уникальным идентификатором (ID). Этот ID имеет форму:

```
уникальная_строка@сервер
```

Уникальная_строка представляет собой последовательность ASCII-символов; *сервер* – это обычно машина, от которой данное сообщение исходит. *Уникальная_строка* должна быть действительно уникальной среди всех посланий, исходящих от сервера. Примером URL для получения отдельного сообщения может быть:

```
news:12A7789B@news.kumquat.com
```

В целом уникальный идентификатор ID – это загадочная последовательность символов, не слишком-то и понятная человеку. Кроме того, продолжительность жизни сообщения на сервере измеряется обычно днями, по прошествии которых оно удаляется и его ID перестает что-либо значить. Резюмируем: URL отдельного сообщения трудно написать, он быстро утрачивает смысл, и его обычно не используют.

6.2.9. URL для nntp

Указатель ресурса типа nntp (nntp URL) идет дальше, чем news URL, и дает полный механизм для доступа к статьям системы телеконференций Usenet. Он имеет следующую форму:

```
nntp://сервер:порт/группа_новостей/статья
```

6.2.9.1. Сервер и порт в URL типа nntp

Сервер и *порт* в nntp определяются так же, как вышеописанные http-сервер и порт. Сервер должен быть доменным именем или IP-адресом nntp-сервера, порт – это номер порта, по которому сервер слушает сеть в ожидании запросов.

Если порт и предшествующее ему двоеточие опущены, по умолчанию используется порт 119.

6.2.9.2. Группа новостей и статья в URL типа nntp

Группа новостей – это название телеконференции, из которой должна быть извлечена статья, как это описано в разделе 6.2.8. *Статья* – это числовой идентификатор запрашиваемой статьи в группе новостей. Хотя номер статьи легче определить, чем идентификатор сообщения, к нему относятся описанные в разделе 6.2.8 ограничения, связанные со ссылками на отдельные сообщения при использовании news URL. В особенности справедливо то, что статьи недолго живут на nntp-серверах, и в итоге nntp URL быстро становятся неактуальными.

6.2.9.3. Образец nntp URL

Примером nntp URL может служить:

```
nntp://news.kumquat.com/alt.fan.kumquats/417
```

Этот URL запрашивает статью 417 из группы новостей *alt.fan.kumquats* на сервере *news.kumquat.com*. Имейте в виду, что будет обслужен запрос только от тех машин, которым позволено получать статьи с этого сервера. Обычно большинство nntp-серверов разрешают доступ только компьютерам, принадлежащим к той же локальной сети.

6.2.10. URL для telnet

Указатель ресурса типа telnet (telnet URL) открывает интерактивный сеанс с указанным сервером, разрешая пользователю зарегистрироваться на машине и применять ее ресурсы удаленно. Часто при соединении с машиной на стороне клиента автоматически запускается специальная программа для обслуживания пользователя; в других случаях он сам должен знать необходимые для работы с системой команды. Указатель telnet URL имеет следующую форму:

```
telnet://пользователь:пароль@сервер:порт/
```

6.2.10.1. Пользователь и пароль в URL типа telnet

Компоненты *пользователь* и *пароль* в telnet применяются в точности так же, как имя пользователя и пароль в вышеописанном ftp URL. В частности, здесь справедливы те же предостережения относительно защиты вашего пароля и недопустимости включения его в URL.

Аналогично ftp URL, если пароль в URL отсутствует, броузер должен предложить вам его ввести непосредственно перед тем, как начать общение с telnet-сервером.

Если вы опустите и имя, и пароль, сеанс telnet начнется без указания имени пользователя. В некоторых случаях будет проведено соединение со службой, доступной анонимному посетителю. В других броузер может предложить ввести имя пользователя и пароль в процессе установки соединения с telnet-сервером.

6.2.10.2. Сервер и порт в URL типа telnet

Сервер и порт в telnet определяются так же, как http-сервер и порт, рассмотренные ранее. Сервер должен быть доменным именем или IP-адресом telnet-сервера. Порт – это номер порта, по которому сервер ожидает запросы. Если порт и предшествующее ему двоеточие опущены, по умолчанию используется порт 23.

6.2.11. URL типа gopher

Gopher – это система передачи документов, получившая некоторую популярность в Интернете накануне появления Интернета, а теперь морально устаревшая. Однако некоторые gopher-серверы еще существуют, и с помощью gopher URL вы можете запрашивать gopher-документы. Указатель gopher URL имеет следующую форму:

```
gopher://сервер:порт/путь
```

6.2.11.1. Сервер и порт в URL типа gopher

Сервер и порт в gopher определяются так же, как http-сервер и порт, описанные ранее. Сервер должен быть доменным именем или IP-адресом gopher-сервера. Порт – это номер порта, по которому сервер ожидает запросов. Если порт и предшествующее ему двоеточие опущены, по умолчанию используется порт 70.

6.2.11.2. Путь в URL типа gopher

Путь может принимать одну из трех форм:

```
тип/селектор
тип/селектор%09поиск
тип/селектор%09поиск%09gopherplus
```

Здесь *тип* – это односимвольное обозначение типа gopher-ресурса. Если в gopher URL путь опущен, по умолчанию принимается тип 1.

Селектор соответствует пути к ресурсу, размещенному на Gopher-сервере. Он может быть опущен, в этом случае запрашивается индекс Gopher-сервера самого высокого уровня.

Если ресурс является в действительности поисковой машиной Gopher, компонент *поиск* передает строку, предназначенную для поиска. Ей должен предшествовать код символа табуляции (%09).

Если Gopher-сервер поддерживает ресурсы Gopher+, компонент *gopherplus* предоставляет информацию для нахождения такого ресурса. Точное содержимое этого компонента варьируется в зависимости от ресурсов на Gopher-сервере. Этому компоненту предшествует код символа табуляции (%09). Если вы хотите включить в URL *gopherplus*, опустив при этом поисковый компонент, то должны оставить в URL оба кода табуляции.

6.3. Создание гиперссылок

Используйте HTML/XHTML-тег <a> для включения гиперссылок на другие источники и для создания в документах идентификаторов фрагментов.

6.3.1. Тег <a>

Чаще всего тег <a> будет применяться с атрибутом <href> для создания гипертекстовых ссылок или *гиперссылок* на другие места в том же документе или на иные документы. В таких случаях документ, в котором помещена гиперссылка, – это источник ссылки, а значение атрибута href, URL, – это цель.[1]

<a>

Функция:	Определяет якорь в потоке текста
Атрибуты:	accesskey, charset, class, coords, dir, href, hreflang, id, lang, name, onBlur, onClick, onDblClick, onFocus, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, rel, rev, shape, style, tabindex, target, title, type
Закрывающий тег:	; всегда присутствует
Содержит:	*a_content* (содержимое якоря)
Может содержаться в:	*тексте*

[1] Вы можете встретить термины *голова* (*head*) и *хвост* (*tail*), обозначающие цель и источник гиперссылки. Эта терминология подразумевает, что документ, на который направлена ссылка (голова) имеет множество хвостов, концы которых находятся во множестве ссылающихся на него документов по всей сети. Мы находим, что такое терминологическое соглашение чревато недоразумениями, и на протяжении этой книги остаемся приверженными терминам «источник» и «цель».

Другой способ использования тега <a> состоит в применении атрибута name, чтобы отметить в документе цель гиперссылки или идентификатор фрагмента. Этот прием, хотя и является частью стандартов HTML 4 и XHTML, постепенно уступает место использованию атрибута id, с помощью которого можно пометить в качестве целей гиперссылок практически любой элемент, включая абзацы, разделы, формы и т. д.

Стандарты позволяют применять атрибуты href и name вместе в одном теге <a>, определяя в текущем документе ссылку на другой документ и идентификатор фрагмента. Мы рекомендуем так не делать, поскольку это нагружает один тег множеством функций, и некоторые броузеры, возможно, не сумеют в нем разобраться. Вместо этого используйте, если возникнет такая нужда, два тега <a>. Ваш исходный текст будет легче читать и модифицировать, и он будет лучше работать с большим числом броузеров.

6.3.1.1. Допустимое содержимое

Между тегом <a> и его обязательным закрывающим тегом можно помещать только обычный текст, встроенные элементы, переходы на новую строку и изображения. Броузер воспроизводит все эти элементы обычным образом, но добавляет при этом какой-нибудь специальный эффект, давая читателю понять, что он имеет дело с гиперссылкой на другой документ. В частности, популярные броузеры обычно подчеркивают и выделяют цветом текст, находящийся в теге <a>, а вокруг содержащихся в этом теге изображений рисуют цветную рамочку.

6.3.1.2. Атрибут href

Применяйте атрибут href для определения URL цели гиперссылки. Значение этого атрибута представляет собой любой допустимый URL документа, абсолютный или относительный, включающий идентификатор фрагмента или фрагмент кода JavaScript. Если пользователь выбирает содержимое тега <a>, броузер пытается получить и отобразить документ, указанный в атрибуте href, или исполнить список выражений, методов и функций JavaScript. [ссылки на документы: URL, 6.2]

Простой тег <a>, ссылающийся на другой документ, может выглядеть так:

```
The <a href="http:growing_season.html">growing
season</a> for kumquats in the Northeast.
```

что будет отображено броузером таким образом (рис. 6.1):

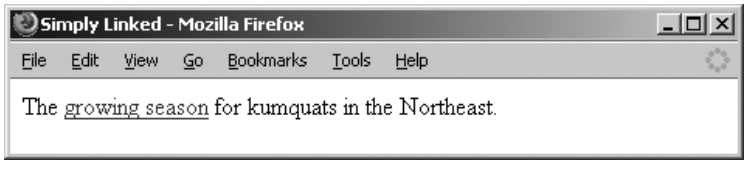

Рис. 6.1. *Гиперссылка на другой HTML-документ*

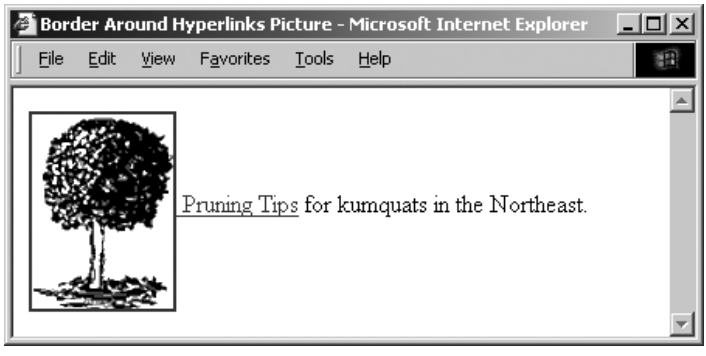

Рис. 6.2. *Internet Explorer рисует специальную рамочку вокруг изображения, содержащегося в якоре*

Отметьте, что фразу «growing season» броузер отображает специальным образом, давая читателю понять, что это ссылка на другой документ. Кроме того, пользователи обычно имеют возможность установить цвет, которым отображаются ссылки до и после употребления. Например, синяя изначально и лиловая после того, как она была хотя бы раз использована.

Более сложные якоря могут включать в себя изображения:

```
<ul>
  <li><a href="pruning_tips.html">
        <img src="pics/new.gif" align=center>
         New pruning tips!</a>
  <p>
  <li><a href="xhistory.html">
        <img src="pics/new2.gif" align=center>
         Kumquats throughout history</a>
</ul>
```

Большинство графических броузеров, таких как Internet Explorer (но не Opera), рисует вокруг изображений, являющихся частью якоря, специальную рамочку (рис. 6.2). Удалить эту рамку гиперссылки можно при помощи присвоения значения `0` атрибуту `border` тега `<img>`. [атрибут border, 5.2.6.8]

6.3.1.3. Атрибуты name и id

Используйте с тегом `<a>` атрибуты `name` и `id` для создания в документе идентификаторов фрагментов. Однажды сформированный, он становится потенциальной целью гиперссылки.

До появления HTML 4.0 единственным способом создания идентификатора фрагмента было применение атрибута `name` с тегом `<a>`. С возникновением в языке атрибута `id`, который используется практически со всяким тегом, любой HTML- или XHTML-элемент может служить

идентификатором фрагмента. Тег <a> сохраняет атрибут name по историческим причинам и поддерживает также id. Эти атрибуты могут заменять друг друга, так как id – это просто более современная версия атрибута name. Каждый из них может быть использован совместно с атрибутом href, позволяя при помощи одного тега <a> определить как гиперссылку, так и идентификатор фрагмента.

Можно думать об идентификаторе фрагмента как о принятом в HTML аналоге метки, на которую осуществляется переход при исполнении оператора goto во многих языках программирования. Атрибут name в теге <a> или id в допустимых тегах устанавливает в документе метку. Когда такая метка применяется в гиперссылке, это эквивалентно приказу броузеру перейти (goto) на нее в документе.

Значение атрибутов name и id – это любая строка символов, заключенная в кавычки. Данная строка должна начинаться с буквы, за которой могут следовать буквы, числа, дефисы, символы подчеркивания, двоеточия и точки. Она должна быть уникальной меткой, которую нельзя повторно использовать другим атрибутам name или id в этом документе, хотя в иных разработках можно применять снова.

Вот примеры использования name и id:

```
<h2><a name="Pruning"> Обрезка вашего кумкватового дерева</a></h2>
<h2 id="Pruning">Обрезка вашего кумкватового дерева</h2>
```

Заметьте, что якорь помещен в заголовок раздела предположительно большого документа. Мы рекомендуем так поступать со всеми главными разделами ваших работ, чтобы было удобнее на них ссылаться и в будущем их можно было бы автоматически обрабатывать, извлекая, скажем, информацию о затронутых в документе темах.

Следующая выбранная пользователем ссылка:

```
<a href="growing_guide.html#Pruning">
```

сразу перенесет его к разделу, который мы поименовали в предыдущем примере.

Содержимое тега <a> с атрибутами name и id отображается обычным способом, без каких-либо специальных эффектов.

Формально говоря, нет необходимости помещать что-либо из содержимого документа в тег <a> с атрибутом name, поскольку он отмечает лишь место в документе. Однако на практике некоторые броузеры игнорируют тег, не содержащий хотя бы фразы, слова или изображения. По этой причине разумно вставлять по меньшей мере один отображаемый элемент в тело всякого тега <a>.

6.3.1.4. Атрибуты событий

В современные броузеры встроено много обработчиков событий. Они ожидают выполнения каких-либо условий или наступления событий,

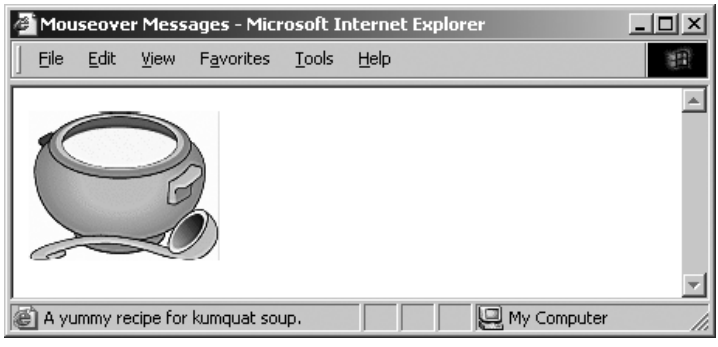

Рис. 6.3. *Используйте JavaScript для отображения сообщений в строке состояния броузера*

таких как щелчок мыши или окончание загрузки изображения. Вы можете включить необходимые обработчики в качестве атрибутов определенных тегов и исполнить на стороне клиента одну или несколько функций JavaScript по наступлении требуемых событий.

При помощи тега `<a>` можно ассоциировать коды JavaScript с множеством событий, относящихся к мыши и клавиатуре. Значением атрибута обработчика событий служит заключенная в кавычки последовательность разделяемых точками с запятой выражений, методов и вызовов функций JavaScript, которая исполняется, когда происходит соответствующее действие. [обработчики событий JavaScript, 12.3.3]

Популярное, хотя и простое, использование события `onMouseOver` в отношении гиперссылки – это вывод в строке состояния броузера расширенного описания документа, на который нацелена гиперссылка (рис. 6.3).

Без этого всякий раз, когда пользователь попадает указателем мыши на содержимое тега `<a>`, в строке состояния появляется таинственный URL:

```
<a href="http://www.ora.com/kumquats/homecooking/recipes.html#quat5"
onMouseOver="status='A yummy recipe for kumquat soup.'; return true;">
<img src="pics/bowl.gif" border=0>
</a>
```

Мы утверждаем, что содержимое тега само по себе должно давать представление о пункте назначения гиперссылки, но бывает, что на экране не хватает места и дополнительное объяснение не повредит[1], как, например, в случае, когда ссылки составляют оглавление.

Подробнее о JavaScript см. в главе 12.

---

[1] Замена стандартного вывода в строке состояния броузера (тем более вывод в нее при помощи сценария «бегущей строки» с каким-либо текстом) считается дурным тоном в оформлении веб-страниц. Старайтесь по возможности избегать этого, не говоря уже о том, что для «пространных» комментариев в строке состояния слишком мало места. – *Примеч. науч. ред.*

6.3.1.5. Атрибуты rel и rev

Необязательные атрибуты `rel` и `rev` тега `<a>` выражают формальные отношения между документами, являющимися источником и целью гиперссылки. Атрибут `rel` определяет отношение документа-источника к цели, а атрибут `rev` определяет отношение цели к источнику. Оба атрибута могут помещаться в одном теге `<a>`, а броузер может использовать их для придания особого вида содержимому якоря или для автоматического построения навигационного меню. Другие программы также могут применять эти атрибуты для создания специальных коллекций ссылок, оглавлений и списков.

Значениями атрибутов `rel` и `rev` являются списки отношений, разделителями в которых служат пробелы. Действительные названия отношений и их смысл в ваших руках, о них ничего не говорится в стандартах HTML и XHTML. К примеру, документ, являющийся элементом некой коллекционной последовательности, может содержать в своих ссылках такие значения атрибутов отношений:

```
<a href="part-14.html" rel=next rev=prev>
```

Отношение источника к цели соответствует переходу к следующему в последовательности документу. Обратное отношение соответствует переходу к предыдущему.

Эти отношения документов применяются также в теге `<link>` в заголовке документа (`<head>`). Тег `<link>` устанавливает отношения, не создавая гиперссылки. Тег `<a>` создает гиперссылку и характеризует ее атрибутами отношений. [элемент заголовка <link>, 6.7.2]

Часто используются такие соотношения между документами:

`next`

Ссылается на следующий документ в коллекции

`prev`

Ссылается на предыдущий документ в коллекции

`head`

Ссылается на документ самого верхнего уровня в коллекции

`toc`

Ссылается на оглавление коллекции

`parent`

Ссылается на документ более высокого уровня

`child`

Ссылается на документ более низкого уровня

`index`

Ссылается на индекс для этого документа

glossary

Ссылается на словарь для этого документа

Немногие броузеры пользуются возможностью изменять внешний вид гиперссылки с учетом этих атрибутов. Тем не менее применять такие атрибуты при создании гиперссылок – это хорошее правило, и мы рекомендуем не жалеть времени, вставляя их везде, где возможно.

6.3.1.6. Атрибуты class и style

Используйте атрибуты class и style с тегом <a> для управления стилем отображения содержимого тега и для форматирования его содержимого в соответствии с заранее определенным стилевым классом для тега <a>. [встроенные стили: атрибут style, 8.1.1] [стилевые классы, 8.3]

6.3.1.7. Атрибуты lang и dir

Подобно почти всем другим тегам, <a> имеет атрибуты lang и dir, обозначающие язык, используемый в его содержимом, и направление, в котором этот язык выводится. [атрибут dir, 3.6.1.1] [атрибут lang, 3.6.1.2]

6.3.1.8. Атрибут target

Атрибут target позволяет указать, куда отображать содержимое документа-цели гиперссылки. Этот атрибут используется обычно с фреймами или в многооконном режиме. Его значением служит имя фрейма или окна, в которые должно быть загружено содержимое документа, на который указывает ссылка. Если фрейм или окно с таким именем существуют, документ загружается в них. Если же нет, создается новое окно с указанным в атрибуте именем, и документ загружается в это окно. За дополнительной информацией, включающей список специальных имен для атрибута target, обратитесь к разделу 11.7.

6.3.1.9. Атрибут title

Атрибут title позволяет определять название документа, на который направлена гиперссылка. Значением атрибута является любая строка, заключенная в кавычки. Броузер может использовать ее при отображении гиперссылки, когда на нее перемещается мышь, например высвечивая название документа-цели (в виде всплывающей подсказки). Броузер может применять атрибут title и при добавлении ссылки в закладки или избранное.

Атрибут title особенно удобен при ссылках на ресурсы, не имеющие собственного названия, на изображения или отличные от HTML документы. К примеру, броузер может включить следующее название в состав представляемого материала, без которого изображение на странице осталось бы немым:

```
<a href="pics/kumquat.gif"
   title="Фотография Благородного Фрукта">
```

В идеале значение атрибута title должно совпадать с названием документа, на который указывает ссылка, но это необязательно.

6.3.1.10. Атрибуты charset, hreflang и type

В соответствии со стандартами HTML 4 и XHTML атрибут charset определяет кодировку символов, используемую в документе-цели. Значением этого атрибута должно быть название стандартного набора символов, например «euc-jp». По умолчанию принимается значение «ISO-8859-1».

Атрибут hreflang применяется только в сочетании с атрибутом href. Подобно атрибуту lang, он принимает значения из множества двухсимвольных кодов языков стандарта ISO (International Organization for Standardization). В отличие от lang, атрибут hreflang не относится к содержимому тега, а определяет язык, используемый в документе, указанном в атрибуте href. [атрибут lang, 3.6.1.2]

Атрибут type определяет тип содержимого документа, на который ссылается тег <a>. Его значением является MIME-тип кодировки. К примеру, вы вправе информировать броузер, что ссылаетесь на обычный ASCII-документ, написав:

```
<a href="readme.txt" type="text/plain">
```

Броузер может использовать данную информацию при отображении документа или даже представить ссылку различным образом в зависимости от типа содержания.

6.3.1.11. Атрибуты coords и shape

Эти атрибуты, определенные в стандартах HTML и XHTML для тега <a>, не поддерживаются современными популярными броузерами. Подобно одноименным атрибутам тега <area>, атрибуты coords и shape определяют область, на которую распространяется влияние тега <a>. Эти атрибуты следует использовать только в том случае, когда тег <a> является частью содержимого тега <map>, как это описано далее в главе. [тег <map>, 6.5.3] [атрибут coords, 6.5.4.2] [атрибут shape, 6.5.4.7]

6.3.1.12. Атрибуты accesskey и tabindex

Традиционно пользователи графических броузеров выбирают и запускают гиперссылку на выполнение, помещая указатель мыши в область, в которой броузер отобразил содержимое якоря, а затем щелкая кнопкой мыши. Менее известно, что существует способ выбрать гиперссылку из множества объектов в окне броузера, нажимая клавишу <Tab>, а затем активизируя ее клавишей <Enter>. С помощью атрибута tabindex вы можете переупорядочить последовательность, в которой броузер проходит элементы документа при нажатии клавиши <Tab>. Значением этого атрибута служит целое положительное число. Броузер начинает с объекта, у которого tabindex=1, и перемещается по объектам в порядке возрастания значения tabindex.

Используя атрибут accesskey, можно установить альтернативную горячую клавишу, с помощью которой активизируется определенная гиперссылка. Значением этого атрибута является одиночный символ, нажимаемый совместно с клавишей <Alt> (или <Meta>), в зависимости от платформы и броузера. В идеале данный символ должен быть представлен в содержимом тега <a>. В таком случае броузер мог бы отобразить символ особым образом, чтобы обозначить его специальное назначение – горячая клавиша.

В главе 9 приведено подробное описание этих атрибутов.

6.3.2. Гиперссылки на другие документы

Вы можете создать гиперссылку на другой документ при помощи тега <a> и его атрибута href, определяющего URL документа-цели. Содержимое тега <a> представляется пользователю неким отличительным образом, дающим понять, что оно относится к гиперссылке.

При создании гиперссылки на другой документ вам следует подумать об использовании атрибутов title, rel и rev тега <a>. Они способствуют документированию вашей гиперссылки и позволяют броузеру улучшить отображение содержимого якоря.

6.3.3. Гиперссылки внутри одного документа

Создание гиперссылки внутри одного и того же документа или на определенный фрагмент другого документа распадается на две части. Сначала надо создать фрагмент, который сможет служить целью гиперссылки, а затем – гиперссылку на фрагмент.

Для того чтобы снабдить фрагмент идентификатором, предназначен тег <a> с атрибутом name. Вот пример:

```
<h3><a name="Section_7">Раздел 7</a></h3>
```

В качестве альтернативы можно применять атрибут id, помещая цель гиперссылки прямо в определяющий тег, например в заголовок:[1]

```
<h3 id="Section_7">Раздел 7</h3>
```

Гиперссылка на фрагмент записывается при помощи тега <a> с атрибутом href, при этом значение атрибута – URL цели гиперссылки – заканчивается именем фрагмента, перед которым ставится знак решетки (#). Ссылка на идентификатор фрагмента из предыдущего примера может выглядеть так:

```
Смотрите <a href="index.html#Section_7">Раздел 7</a>
для дополнительной информации.
```

[1] Мы предпочитаем способ, задействующий id, хотя пока не все броузеры его поддерживают.

В настоящее время идентификаторы фрагментов употребляются чаще всего при создании оглавлений больших документов. Начните с разбиения документа на несколько логических разделов, используя подходящие заголовки и последовательное форматирование. В начале каждой части, обычно в заголовке, вставьте идентификатор фрагмента этого раздела. Завершите свою деятельность, поместив в начало документа список ссылок на данные идентификаторы.

Документ, выбранный нами в качестве образца, восхваляет жизнь и деяния могучего кумквата. Материал очень большой и сложный, включает в себя много интересных разделов и подразделов. Это документ, который надо читать и перечитывать. Чтобы поклонники кумкватов всего мира могли быстро находить интересующие их темы, мы вставили идентификаторы фрагментов в заголовки всех главных разделов и поместили их упорядоченный список – оглавление, снабженное гиперссылочным механизмом, – в начало каждого документа библиотеки «Поклонников кумквата». Ниже приведен пример с образцами идентификаторов фрагментов. Многоточие (...) означает, конечно, что пропущен кусок содержимого:

```
...
<h3>Table of Contents</h3>
<ol>
  <li><a href="#soil_prep">Подготовка Почвы</a>
  <li><a href="#dig_hole">Рытье ям</a>
  <li><a href="#planting">Посадка деревьев</a>
</ol>
...
<h3 id=soil_prep>Подготовка Почвы</h3>
...
<h3 id=dig_hole>Рытье ям</h3>
...
<h3 id=planting>Посадка деревьев</h3>
...
```

Поклонник кумквата может, таким образом, щелкнуть на понравившейся ему гиперссылке в оглавлении и прямо перескочить к интересующему его разделу, что позволяет избежать скучного пролистывания документа.

Обратите внимание на то, что в этом примере использованы относительные URL, что может оказаться очень удобным для сохранения ссылочной целостности, если вам когда-нибудь придется перемещать или переименовывать документ.

6.4. Эффективное применение гиперссылок

Документ становится гипертекстовым после вставки в него нескольких гиперссылок, как вода превращается в суп, когда в нее бросили немного овощей. С формальной точки зрения вы своего достигли, но результат получился, возможно, не очень вкусный.

Включение якорей в документ – это своего рода искусство, требующее хороших навыков письма, знания HTML или XHTML и архитектурного чутья, позволяющего соотнести структуру документа с «застройкой» сети. Эффективные гиперссылки плавно проникают в документ, без лишнего шума предоставляя читателю дополнительные возможности просмотра. Кричащие, плохо оформленные гиперссылки нарушают гладкое течение документа и обычно раздражают читателя.

Хотя существует столько же стилей введения гиперссылок, сколько авторов HTML-документов, мы описываем здесь несколько самых популярных приемов оснащения ими документов. Все они делают две вещи: дают читателю быстрый доступ к информации, имеющей отношение к обсуждаемой теме, и сообщают ему, в чем это отношение состоит.

6.4.1. Списки ссылок

Возможно, самым распространенным способом представления гиперссылок являются их упорядоченные и неупорядоченные списки, выполняющие функцию оглавления или списка ресурсов.

Существуют две школы создания списков. Приверженцы первой из них помещают в якорь весь элемент списка, другие выделяют якорем только отрывок, одну фразу из элемента. В первом случае добивайтесь, чтобы содержимое якоря было кратким и ясным, во втором – используйте более пространный стиль речи, в которую легко вставить ссылку.

Если ссылочный список становится чрезмерно длинным, подумайте о том, чтобы разбить его на несколько подсписков, сгруппированных по темам. Читатели тогда смогут просмотреть темы (выделенные, например, <h3>-заголовками), чтобы найти подходящий список, а уже затем искать в нем нужный документ.

Альтернативный стиль списков значительно информативнее, но и многословнее, так что постарайтесь не делать его чересчур громоздким:

```
<p>
Kumquat-related documents include:
<ul>
  <li>A concise guide to <a href="kumquat_farming.html">
      profitable kumquat farming</a>,
      including a variety of business plans, lists of fruit
      packing companies, and farming supply companies.
  <li>101 different ways to <a href="kumquat_uses">
      use a kumquat</a>, including stewed kumquats and kumquat pie!
  <li>The kumquat is a hardy tree, but even the greenest of
      thumbs can use a few <a href="news:alt.kumquat_growers">
      growing tips</a> to increase
      their yield.
  <li>The business of kumquats is an expanding one, as
      shown by this 10 year overview of the
      <a href="http://www.oreilly.com/kumquat_report/">
```

```
        kumquat industry</a>.
    </ul>
```

Иногда чтение исходных HTML-документов утомляет, а XHTML-документы – еще скучнее. Вообразите только, какой бы получился беспорядок, если бы мы использовали якоря с идентификаторами фрагментов для каждой подтемы в описаниях элементов списка! Тем не менее все это неплохо выглядит и удобно для целей навигации, когда отображается броузером, таким как Opera (рис. 6.4).

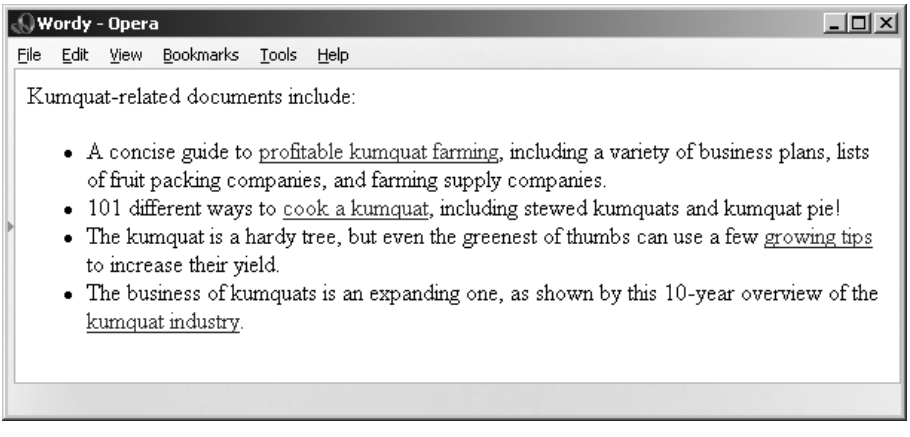

Рис. 6.4. *Многословный, но информативный список со ссылками*

Этот более информативный стиль представления ссылочных списков много делает для того, чтобы дать читателям полнее почувствовать вкус предлагаемых им блюд. Поскольку каждый элемент списка длиннее и требует более продолжительного чтения, вам следует использовать этот стиль пореже и жестко ограничивать количество ссылок в списке.

Применяйте списки с краткими элементами, когда вы представляете большое число ссылок хорошо информированной аудитории. Второй, более информативный, стиль лучше подходит для небольшого числа ссылок на темы, с которыми читатели, возможно, менее знакомы.

6.4.2. Встроенные ссылки

Если вы не собираете ссылки в списки, то, вероятно, разбрасываете их по всему документу. Так называемые *встроенные ссылки* больше соответствуют духу гипертекста, поскольку они позволяют читателю заметить место, где он в данный момент находится, посетить документ, содержащий более глубокое изложение или лучшее объяснение темы, а затем вернуться назад и продолжить чтение. Это вносит элемент индивидуального подхода в процесс изучения.

Большая ошибка, к которой склонны новички, состоит тем не менее в чрезмерном употреблении гиперссылок и обращении с ними, как

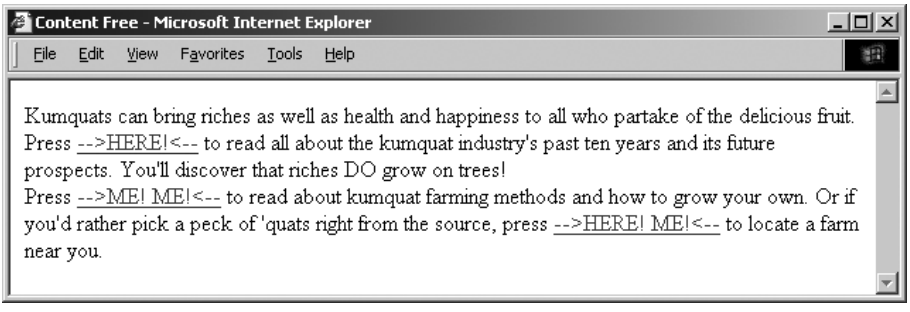

Рис. 6.5. *Гиперссылки не должны тянуть руку и вопить, как отличники: «Меня, меня, вызовите меня!»*

с кнопками аварийной связи, взывающими, чтобы их нажали. Вы, возможно, видели такую манеру исполнения гиперссылок. Взгляните все же (рис. 6.5) на документ, безусловно, одержимый паникой, так как он заполнен здесь и там словом «here» (ну никак не заставить себя привести исходный текст этой пародии).

Используемые в качестве гиперссылок фразы: «щелкни здесь» или «также вы можете» – бессодержательны и вызывают раздражение. Они вынуждают человека, который ищет на странице важную для него ссылку, прочитать весь окружающий текст, чтобы найти то, что ему действительно нужно.

Лучший, более изысканный стиль встроенных гиперссылок требует, чтобы каждая из них содержала существительное или фразу, составленную из существительного и глагола, связанных с рассматриваемой темой. Сравните, как представлены ссылки, относящиеся к выращиванию и промышленной обработке кумкватов (рис. 6.6), и как кричат другие: «Меня! Меня!» (рис. 6.5).

Быстрый просмотр (рис. 6.6) немедленно выделяет полезные ссылки на «kumquat farming methods» (методы выращивания кумкватов) и «kumquat industry past ten years» (промышленная обработка кумкватов в прошедшее десятилетие). Нет необходимости читать окружающий текст, чтобы понять, куда перенесет вас гиперссылка. Дейст-

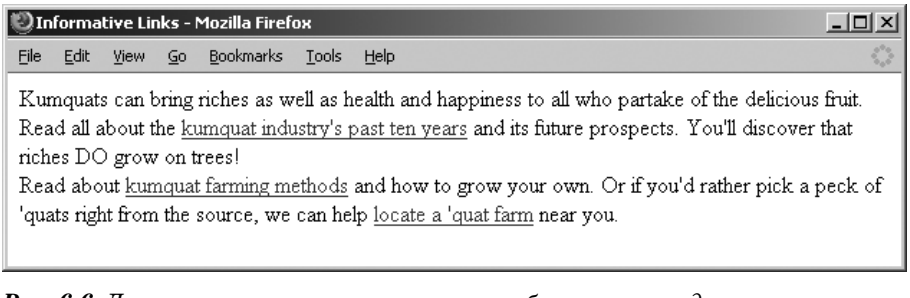

Рис. 6.6. *Деликатные встроенные ссылки работают гораздо лучше*

вительно, непосредственно окружающий ссылки текст работает в этом примере, а также в случае большинства встроенных ссылок, как синтаксический сироп, в который погружены сладкие кусочки вложенных гиперссылок.

Включение гиперссылок в основной текст документа требует больших усилий, чем составление ссылочных списков. Необходимо действительно понимать как содержание документа, в который гиперссылка вставлена, так и того, на который она указывает, уметь выразить существующее между ними отношение в нескольких словах, а затем разумно вставить эту ссылку в ключевом месте документа. Хорошо, если это ключевое место находится там, где, как вы ожидаете, читатель готов прерваться и задать вопрос или запросить дополнительную информацию. Самое трудное, особенно для создателей традиционной технической документации, состоит в том, что такая форма диалога автора и читателя особенно эффективна, когда читатель занимает позицию активного субъекта действия, а не пассивного поглотителя предложенной ему автором информации. Потраченные усилия окупятся, приводя к созданию более информативных и удобных для изучения документов. Помните, что вы пишете документ один раз, а читать его будут тысячи, если не миллионы раз. Не стесняйтесь угождать своим читателям.

6.4.3. Что нужно и чего нельзя делать с гиперссылками

Вот несколько советов по поводу гиперссылок:

Пусть содержимое ваших ссылок будет как можно короче

Длинные ссылки и огромные встроенные изображения, используемые для ссылок, разрушают визуальную целостность документа и могут служить источником недоразумений.

Никогда не располагайте две ссылки вплотную друг к другу

Большинство броузеров отобразят их так, что будет трудно сказать, где окончилась одна ссылка и началась другая. Разделяйте ссылки обычным текстом или размещайте их на разных строках.

Будьте последовательны

Используете встроенные ссылки? Пусть все ваши гиперссылки будут встроенными. Предпочитаете ссылочные списки? Применяйте либо длинную, либо короткую их форму. Не смешивайте стили в одном документе.

Попробуйте прочитать свой документ, из которого выброшен весь текст, кроме гиперссылок

Если некоторые ссылки выглядят бессмысленными, перепишите их так, чтобы они сами за себя говорили. (Многие люди просматривают документы, замечая в них только гиперссылки. Окружающий их текст становится тогда едва ли чем-то большим, чем серый фон для визуально более притягательных ссылок.)

6.4.4. Изображения и гиперссылки

Теперь стало модным использовать в якорях пиктограммы вместо слов. К примеру, вместо слова «следующий» можно вставить пиктограмму с маленьким указующим перстом. Ссылка на домашнюю страницу кажется незавершенной, если в ней нет картинки с маленьким домиком. Ссылки на поисковые системы обязаны в наши дни содержать изображение увеличительного стекла, вопросительного знака или бинокля. И еще вся эта мелькающая реклама с GIF-анимацией.

Берегитесь поддаться «синдрому Эвереста», симптомом которого служит стремление вставить изображение только потому, что умеете это делать. Если вы или ваши читатели не могут с первого взгляда сказать, в каком отношении к документу находится ссылка, вы с ней ошиблись. Скупее употребляйте для гиперссылок замысловатые картинки, делайте это не бессистемно и только для того, чтобы помочь читателям обнаружить важную информацию. Задумывайтесь также над тем, что увидят в ваших страницах читатели, разбросанные по всей планете (а может быть, и за ее пределами), не забывая, что картинки тоже по-разному воспринимаются представителями разных культур. (Слышали ли вы, что означают для японцев пальцы, сложенные так, чтобы показать американцу «ОК»?)

Создание цельной подборки пиктограмм для собрания документов – это задача пугающей трудности, к исполнению которой следует приступать, только заручившись поддержкой специалиста-дизайнера. Поверьте нам, тот тип ума, который продуцирует затейливые коды и хорошо пишет на XHTML, редко подходит для создания красивых и притягательных образов. Найдите хорошего дизайнера, и ваши странички и читатели выиграют неизмеримо.

6.5. Изображения, реагирующие на мышь

Обычно изображение, помещенное в якорь, становится просто частью содержимого якоря. Броузер может как-то по-особенному представить такое изображение, чаще всего с помощью специальной рамки, предупреждая читателя, что с ним связана гиперссылка, но пользователи щелкают мышью на картинке так же, как на текстовой гиперссылке.

Стандарты HTML и XHTML предоставляют возможность разместить несколько ссылок над одним изображением. Щелчок мышью в разных местах объекта вызывает переход броузера к различным документам. Такие изображения, известные как *карты*, открывают широкий простор для творчества.

Существуют два типа карт: *серверная карта* и *клиентская карта*. Первый тип используется, когда применяется атрибут `ismap` тега `<img>`, и требует доступа к серверу и программным приложениям на нем, отрабатывающим информацию, связанную с картой. Другой способ за-

пускается при включении в тег <img> атрибута usemap и применении тегов <area> и <map>.

Поскольку преобразование положения мыши на изображении в ссылку на другой документ происходит на машине пользователя, карты, работающие на стороне клиента, не требуют особого соединения с сервером и могут осуществляться даже без сети на локальном диске или с коллекцией документов на CD-ROM. Любой HTML/XHTML-код может реализовать клиентскую карту (usemap). [тег <map>,6.5.3] [тег <area>, 6.5.4] [тег <img>, 5.2.6]

6.5.1. Серверные карты

Изображение вставляется в якорь просто помещением тега <img> в тело тега <a>. В карту такое вложенное изображение превращается при добавлении в тег <img> атрибута ismap. Этот атрибут сообщает броузеру, что изображение представляет собой карту, содержащую более чем одну ссылку. (Броузер игнорирует атрибут ismap, если тег <img> не вложен в тег <a>.)

Когда пользователь щелкает мышью где-то на изображении, броузер передает координаты указателя мыши по URL, указанному в теге <a>, серверу документа. Сервер рассматривает координаты указателя мыши, чтобы решить, какой документ отправить броузеру в ответ.

При использовании атрибута ismap атрибут href соответствующего тега <a> должен содержать URL приложения на сервере или для некоторых HTTP-серверов ассоциированный с картой файл, содержащий информацию о координатах и ссылках. Если в href указан URL обычного документа, результатом может быть ошибка и нужный документ, вероятнее всего, не будет получен.

Координаты положения мыши представляют собой измеренные в пикселах расстояния, отсчитанные от левого верхнего угла изображения, принимаемого за (0, 0). Координаты дописываются в конец URL после вопросительного знака.

К примеру, если пользователь щелкнул мышью на 43 пиксела правее и на 15 пикселов ниже верхнего левого угла изображения, представляющего следующую гиперссылку:

```
<a href="/cgi-bin/imagemap/toolbar.map">
<img ismap src="pics/toolbar.gif">
</a>
```

броузер пошлет HTTP-серверу такие параметры поиска:

```
/cgi-bin/imagemap/toolbar.map?43,15
```

В приведенном примере *toolbar.map* – это специальный файл карты в каталоге *cgi-bin/imagemap*, содержащий связь между координатами и ссылками. Специальный процесс обработки карты применяет этот файл, чтобы узнать, какой гиперссылке соответствуют переданные

координаты (в нашем случае (43, 15)), и вернуть соответствующий документ.

6.5.1.1. Что происходит на сервере

При обработке карты, запущенной с использованием атрибута `ismap`, броузер должен только передать серверу URL с присоединенными к нему координатами мыши. Сервер преобразует эти координаты в определенный документ. Процесс такого преобразования различается от сервера к серверу и не определяется стандартами HTML или XHTML.

Вам необходимо проконсультироваться с администратором сервера и, возможно, даже прочитать документацию вашего сервера, чтобы узнать, как создать и запрограммировать серверную карту. Большинство серверов имеют программную утилиту, расположенную обычно в каталоге *cgi-bin/imagemap* и предназначенную для обработки карт. И большинство из них используют текстовые файлы, содержащие области карты и ассоциированные с этими областями ссылки. К этому файлу и обращается URL в соответствующем теге `<a>` за информацией, необходимой для обработки запроса, поступившего от карты.

Вот пример файла карты, описывающего чувствительные области изображения из нашего примера:

```
# Imagemap file=toolbar.map

default                 dflt.html
circ 100,30,50          link1.html
rect 180,120,290,500    link2.html
poly 80,80,90,72,160,90 link3.html
```

Каждая чувствительная область карты описана названием геометрической фигуры и определяющими ее координатами в пикселах. Это окружность, ее центр и радиус; прямоугольник с координатами верхнего левого и правого нижнего углов; многоугольник с координатами вершин. Все координаты откладываются от верхнего левого угла изображения (0, 0). Каждой области соответствует свой URL.

Обработчик карты обычно проверяет каждую фигуру в том порядке, в каком они представлены в файле карты, и возвращает броузеру документ, указанный в соответствующем фигуре URL, если координаты пользовательской мыши попадают в пределы этой фигуры. Это значит, что фигуры могут накладываться одна на другую, надо только не забывать об их порядке. При этом не обязательно, чтобы все изображение было покрыто областями чувствительности – если переданные серверу координаты не попали ни в одну из них, броузеру будет отправлен документ «по умолчанию».

Это только один пример того, как может обрабатываться карта и как при этом применяются вспомогательные файлы. Обратитесь к своему вебмастеру и документации по серверу, чтобы узнать, как реализовать карту со стороны сервера для документов на вашей системе.

6.5.2. Клиентские карты

Недостаток карт со стороны сервера заключается, очевидно, в том, что для них необходим сервер. Это означает, что нужен доступ к HTTP-серверу или его каталогу */cgi-bin*, а такими правами обычно обладают только владельцы и системные администраторы. Вдобавок карты, обрабатываемые со стороны сервера, ограничивают переносимость, так как не все программы, обслуживающие карты, работают одинаково.

Серверные карты для пользователя означают также дополнительные задержки, так как броузер должен сначала обратить на себя внимание сервера, чтобы тот затем обработал переданные ему координаты. И это так, даже если серверу не требуется предпринимать никаких действий, как бывает в случае, когда выбранная область изображения не имеет соответствующей гиперссылки и никуда не ведет.

С клиентскими картами не связано ни одно из этих затруднений. Создаваемые при помощи атрибута `usemap` тега `<img>` и определяемые специальными тегами расширений `<map>` и `<area>`, карты со стороны клиента позволяют авторам включать в их документы описания чувствительных областей изображения и ассоциированных с ними ссылок. Броузер на компьютере клиента преобразует координаты мыши в некоторое действие, например в загрузку и отображение другого документа. При этом ряд предназначенных для работы с JavaScript атрибутов позволяет осуществить разнообразные эффекты для карт со стороны пользователя. [обработка событий JavaScript, 12.3.3]

Для создания карты со стороны клиента включите в тег `<img>` атрибут `usemap`.[1] Его значением должен быть URL сегмента `<map>` в HTML- или XHTML-документе, содержащий координаты на карте и соответствующие URL гиперссылок. Часть URL, указывающая на документ, сообщает, где содержится описание карты. Идентификатор фрагмента в URL сообщает, какое из описаний должно быть использовано. Чаще всего описание карты содержится в том же документе, где и само изображение, и URL сводится к идентификатору фрагмента – значку решетки (#), за которым следует имя `<map>`-сегмента.

К примеру, следующий фрагмент исходного документа сообщает броузеру, что изображение *map.gif* – это карта со стороны клиента и что координаты чувствительных областей и соответствующее URL содержатся в `<map>`-фрагменте текущего документа:

```
<img src="pics/map.gif" usemap="#map">
```

[1] В соответствии со стандартом HTML 4 вы можете создать клиентскую карту, применив атрибут `usemap` с тегом `<object>` и тегом формы `<input>`. Детальные пояснения см. в главе 12.

6.5.3. Тег <map>

Чтобы клиентская карта заработала, необходимо где-то в документе поместить информацию, определяющую чувствительные области изображения и гиперссылки, которые должны быть загружены по щелчку мыши (или в результате иных действий пользователя)[1] в каждой из этих областей. Включите эту информацию в значения атрибутов традиционных тегов <a> или специальных <area>. Набор спецификаций <area> или тегов <a> заключается между тегом <map> и его закрывающим </map>. При этом сегмент <map> может быть размещен где угодно в пределах тела документа.

<map>	
Функция:	Содержит спецификации для клиентской карты
Атрибуты:	class, dir, id, lang, name, onClick, onDblClick, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, style, title
Закрывающий тег:	</map>; всегда присутствует
Содержит:	*map_content* (содержимое карты)
Может содержаться в:	*body_content* (содержимое тела)

Точнее говоря, тег <map> может содержать либо последовательность тегов <area>, либо традиционное HTML/XHTML-содержимое, включая теги <a>. Нельзя смешивать теги <area> с традиционным содержимым. Традиционное содержимое тега <map> может быть отображено броузером, теги <area> – нет. Если вас беспокоит совместимость со старыми броузерами, используйте <map>, включающий только теги <area>.

Если вы все же помещаете теги <a> в тег <map>, они должны содержать атрибуты coords и shape, определяющие области в объектах, на которые ссылается <map>.

6.5.3.1. Атрибут name

Значение атрибута name в теге <map> – это то имя, которое применяет атрибут usemap в тегах <img> и <object>, указывая на место, где расположено описание карты. Такое имя должно быть уникальным и не использоваться никаким другим <map> в этом же документе, но ссылаться на один <map>-сегмент могут несколько карт. [атрибуты ismap и usemap, 5.2.6.14]

6.5.3.2. Атрибуты class, dir, id, lang и style

Стилевые атрибуты class и style тега <map>, которые имеют отношение к представлению данных на экране, полезны только тогда, когда тег

[1] Клавиша <Tab> тоже позволяет перебирать гиперссылки в документе, включая клиентские карты. Выберите гиперссылку и нажмите клавишу <Enter>.

<map> определяет обычное содержимое. В этом случае они применяются ко всему содержимому. [встроенные стили: атрибут style, 8.1.1] [стилевые классы, 8.3]

Зато атрибуты id и title просты и понятны. Это стандартные атрибуты, позволяющие пометить тег для последующего указания на него с помощью гиперссылки или программы и соответственно озаглавить раздел для последующей ссылки на него. [атрибут id, 4.1.1.4] [атрибут title, 4.1.1.5]

6.5.3.3. Атрибуты событий

Разнообразные атрибуты событий позволяют вам указывать JavaScript-обработчики для событий, которые могут произойти в пределах карты. [обработчики событий JavaScript, 12.3.3]

6.5.4. Тег <area>

Внутреннее устройство карты описывается при помощи тегов <area> в <map>-сегменте. Данные теги определяют каждую чувствительную область карты и действие, которое должен произвести броузер, когда пользователь выбирает эту область, обрабатываемую со стороны клиента.

Определенная тегом <area> область работает как любая другая гиперссылка: когда указатель мыши попадает в нее, вид указателя меняется (как правило, принимая форму руки), и броузер, возможно, показывает в строке состояния под демонстрационным окном URL соответствующей гиперссылки.[1] Области изображения, которые не принадлежат ни одной из тех, что определены в тегах <area>, не являются чувствительными.

<area>

Функция:	Определяет координаты областей на карте и ассоциированные с ними гиперссылки
Атрибуты:	accesskey, alt, class, coords, dir, href, id, lang, nohref, notab, onBlur, onClick, onDblClick, onFocus, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseUp, shape, style, tabindex, taborder [!], target [!], title, type
Закрывающий тег:	Отсутствует в HTML; </area> или <area ... /> в XHTML
Содержит:	Ничего
Может содержаться в:	*map_content* (содержимое карты)

[1] Так будет до тех пор, пока вы не активизируете обработчик событий JavaScript, который осуществляет вывод в строку состояния. См. описание обработчика события onMouse в разделе 6.5.4.6.

6.5.4.1. Атрибут alt

Подобно своему тезке из тега `<img>`, атрибут `alt` тега `<area>` позволяет присоединить метку, но теперь не к изображению в целом, а к той его части, которая определяется тегом. Популярные броузеры отображают метку, когда курсор мыши попадает в соответствующую область, а нерафические броузеры могут использовать этот текст для создания взамен карты списка гиперссылок, состоящего из значений атрибута `alt`.

6.5.4.2. Атрибут coords

Обязательный для тега `<area>` атрибут `coords` определяет на клиентской карте координаты точек, входящих в чувствительную область. Число координат и их смысл зависят от формы области, которую задает атрибут `shape`, обсуждаемый ниже в этой главе. Можно определить области как прямоугольники, круги и многоугольники. Ниже приведены значения координат, свойственные каждой форме.

`circle` (круг) *или* `circ`

`coords` = «*x, y, r*», где *x* и *y* определяют положение центра круга (`0,0` – верхний левый угол изображения), а *r* – его радиус в пикселах.

`polygon` (многоугольник) *или* `poly`

`coords` = «*x1, y1, x2, y2, ...*», где каждая пара координат *x, y* задает вершину многоугольника, а `0,0` – левый верхний угол изображения. По меньшей мере три пары координат требуются, чтобы определить треугольник, многоугольники более высокого порядка определяются большим числом пар координат. Многоугольник является замкнутой фигурой, следовательно, нет необходимости повторять в конце списка координаты первой вершины.

`rectangle` (прямоугольник) *или* `rect`

`coords` = «*x1, y1, x2, y2*», где первая пара координат соответствует одному углу прямоугольника, а вторая – противоположному по диагонали, и `0,0` – левый верхний угол изображения. Заметьте, что прямоугольник – это просто сокращенная запись многоугольника с четырьмя вершинами.

К примеру, следующий XHTML-фрагмент определяет одну чувствительную область в правой нижней четверти изображения 100×100 и вторую круговую в самом его центре:

```
<map name="map1">
  <area shape="rect" coords="75,75,99,99" nohref="nohref" />
  <area shape="circ" coords="50,50,25" nohref="nohref" />
</map>
```

Если точка принадлежит двум областям, определенным тегами `<area>`, приоритет имеет первый по порядку тег. Броузеры игнорируют координаты, выходящие за пределы изображения.

6.5.4.3. Атрибут href

Подобно атрибуту `href` для тега `<a>`, атрибут `href` для тега `<area>` определяет URL гиперссылки, переход по которой происходит при щелчке на соответствующей чувствительной области. Значением атрибута `href` может служить любой допустимый URL, относительный или абсолютный, включая код JavaScript.

В частности, броузер загрузит и отобразит документ *link4.html*, если пользователь щелкнет в левой нижней четверти изображения размером 100×100 пикселов, как это предписано первым тегом `<area>` в следующем примере HTML:

```
<map name="map">
  <area coords="75,75,99,99" href="link4.html">
  <area coords="0,0,25,25" href="javascript:window.alert
  ('Ой, щекотно!')"
</map>
```

Второй тег `<area>` применяет JavaScript URL, и когда пользователь щелкнет на левом верхнем квадрате карты, он выполнит метод JavaScript, который выведет в диалоговое окно глупое сообщение.

6.5.4.4. Атрибут nohref

Атрибут `nohref` для тега `<area>` определяет чувствительную область изображения, для которой не производится никакого действия, даже если пользователь ее выбрал. В каждый тег `<area>` должен быть включен либо атрибут `href`, либо `nohref`.

6.5.4.5. Атрибуты notab, taborder и tabindex

Альтернативой работы с мышью является переход по горячим точкам документа, таким как включенные в карту гиперссылки, при помощи клавиши <Tab>. Выбранную гиперссылку пользователь активизирует, нажав клавишу <Enter>. По умолчанию броузер перешагивает с одного активного элемента на другой в том порядке, в каком они появляются в документе. При помощи родившегося в Internet Explorer атрибута `taborder`, теперь включенного в стандарт в виде атрибута `tabindex`, можно изменить этот порядок. Значение данного атрибута – целое число, указывающее позицию этого элемента в общей последовательности переходов для текущего документа.

Поддерживаемый только Internet Explorer и не являющийся частью стандартов HTML 4 и XHTML атрибут `notab` отмечает области, которые должны пропускаться, когда читатель нажимает на клавишу <Tab>, перемещая курсор по документу. В противном случае эта область была бы частью последовательности переходов. Атрибут полезен, разумеется, в комбинации с атрибутом `nohref`.

Атрибуты `notab` и `taborder` поддерживались Internet Explorer версии 4. Броузер Internet Explorer версии 5 и более поздних поддерживает так-

же tabindex, поэтому вам следует пользоваться стандартными атрибутами, а не расширениями.

6.5.4.6. Атрибуты событий

С клиентской картой работают те же обработчики событий JavaScript, что и с тегом гиперссылки <a>. Значением атрибута обработчика событий служит заключенная в кавычки последовательность разделяемых точками с запятой выражений, методов и вызовов функций JavaScript, которая исполняется, когда наступает соответствующее событие. [обработчики событий JavaScript, 12.3.3]

Популярное, хотя и простое, использование события onMouseOver — это вывод в строке состояния броузера пространного объяснения, когда пользователь попадает указателем мыши в соответствующую область на карте.

```
<area href="http://www.oreilly.com/kumquats/homecooking/recipes.html#quat5"
   onMouseOver="self.status='Рецепт супа из кумкватов.';return true">
```

Следует указать, что современные версии популярных броузеров автоматически отображают строку-значение атрибута alt, выполняя ту же функцию. Так что стоит вместо колдовства с JavaScript включать в тег атрибут alt с его значением. И в полном соответствии с текстовыми гиперссылками содержимое тега должно само объяснять ссылку. Однако изображения могут быть обманчиво ясными, поэтому мы настаиваем, что следует использовать все возможности, включая атрибут alt и обработчики событий, чтобы снабдить ваши карты текстовыми пояснениями.

6.5.4.7. Атрибут shape

Используйте атрибут shape для определения формы чувствительной области на карте — круга (circ или circle), многоугольника (poly или polygon), прямоугольника (rect или rectangle).

Значение атрибута shape влияет на то, как броузер интерпретирует значение атрибута coords. Если вы не включили в тег shape, принимается значение default. В соответствии со стандартом default означает, что область покрывает все изображение. На практике по умолчанию броузеры ожидают прямоугольник и рассчитывают получить четыре значения coords. И если вы не определили форму области и не включили в тег четырех координат, броузер игнорирует ее полностью.

В действительности только самые последние версии популярных броузеров распознают значение default атрибута shape, создавая «всеохватывающую» область, которая принимает все щелчки мышью, не попавшие ни в одну из определенных частей карты. Поскольку области в теге <map> срабатывают в том порядке, в котором они перечислены, вам следует поместить зону «по умолчанию» последней. В противном случае она покроет все области, следующие за ней.

Броузерам не хватает порядка в отношении названий форм областей. Netscape 4, например, не признает «rectangle», зато понимает, что «rect» имеет прямоугольную форму. Поэтому мы советуем использовать сокращенные имена.

6.5.4.8. Атрибут target

Атрибут `target` позволяет указать, куда отображать содержимое документа-цели гиперссылки. Этот атрибут используется обычно с фреймами или в многооконном режиме. Значением данного атрибута служит имя фрейма или окна, в которые должно быть загружено содержимое документа, на который указывает ссылка. Если фрейм или окно с таким именем существуют, документ загружается в них. Если же нет, создается новое окно под указанным в атрибуте именем, и документ загружается в него. За дополнительной информацией, включающей список специальных имен для атрибута `target`, обратитесь к разделу 11.7.

6.5.4.9. Атрибут title

Атрибут `title` позволяет определять название документа, на который направлена гиперссылка. Значением атрибута может служить любая строка, заключенная в кавычки. Броузер подчас использует ее при отображении гиперссылки, например высвечивая название документа-цели (в виде всплывающей подсказки), когда на ссылку попадает мышь. Броузер может применять атрибут `title` и при добавлении ссылки в закладки или избранное.

Атрибут `title` особенно удобен при ссылках на ресурсы, не имеющие собственного названия, на изображения или отличные от HTML документы. К примеру, броузер может включить следующее название в состав представляемого материала, без которого изображение на странице осталось бы немым:

```
<a href="pics/kumquat.gif"
   title=" Фотография Благородного Фрукта">
```

В идеале значение атрибута `title` должно совпадать с названием документа, на который указывает ссылка, но это необязательно.

6.5.4.10. Атрибуты class, dir, id, lang и style

Атрибуты `class` и `style` позволяют определить свойства отображения и имена классов для управления внешним видом чувствительной области, хотя их ценность для этого тега представляется ограниченной. Атрибут `id` служит для того, чтобы стандартным способом пометить содержимое тега для позднейших ссылок. [атрибут id, 4.1.1.4] [встроенные стили: атрибут style, 8.1.1] [стилевые классы, 8.3]

Атрибуты `lang` и `dir` обозначают язык, применяемый для этой области, и направление, в котором он выводится. Способ их использования с этим тегом также неясен. [атрибут dir, 3.6.1.1] [атрибут lang, 3.6.1.2]

6.5.5. Пример клиентской карты

Следующий фрагмент HTML-текста сводит вместе обсуждавшиеся выше в этом разделе различные компоненты карты, обрабатываемой со стороны клиента. Пример включает в себя тег `<img>` со ссылкой на содержащий картинку файл и с атрибутом `usemap`, значение которого указывает `<map>`-сегмент, где определены четыре (три явно и одна по умолчанию) чувствительные области и соответствующие гиперссылки:

```
<body>
...
<img src="pics/map.gif" usemap="#map1" border=0>
...
<map name="map1">
  <area shape=rect coords="0,20,40,100"
      href="k_juice.html"
      onMouseOver="self.status='How to prepare kumquat juice.'
         ;return true">
  <area shape=rect coords="50,50,80,100"
      href="k_soup.html"
      onMouseOver="self.status='A recipe for hearty kumquat soup.'
      ;return true">
  <area shape=rect coords="90,50,140,100"
      href="k_fruit.html"
      onMouseOver="self.status='Care and handling of the native kumquat.'
      ;return true">
  <area shape=default
      href="javascript:window.alert('Choose the cup or one of the bowls.')"
      onMouseOver="self.status='Select the cup or a bowl for more
information.'
      ;return true">
</map>
```

Результат представлен на рис. 6.7.

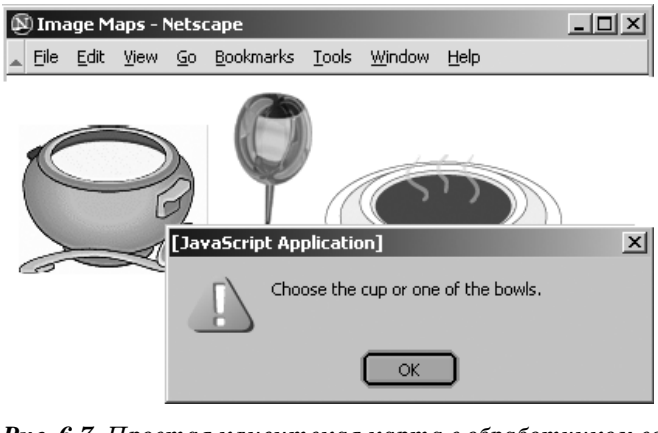

Рис. 6.7. *Простая клиентская карта с обработчиком событий на JavaScript*

6.5.6. Учет особенностей других броузеров

В отличие от его серверного (ismap) аналога, тег клиентской карты (<img usemap>) не обязан входить в тег <a>. Но может, а это позволяет изящно обслужить броузеры, которые не умеют обрабатывать карты со стороны клиента.

К примеру, если пользователь щелкнет на изображении *map.gif*, старый броузер Mosaic и ранние версии Netscape просто загрузят документ *main.html*. Современные броузеры, с другой стороны, разделят изображение на чувствительные области, определенные в ассоциированном <map>-сегменте, и исполнят гиперссылки на уникальную для каждой области часть документа *main.html*.

```
<a href="main.html">
  <img src="pics/map.gif" ismap usemap="#map1">
</a>
...
<map name="map1">
  <area coords="0,0,49,49" href="main.html#link1">
  <area coords="50,0,99,49" href="main.html#link2">
  <area coords="0,50,49,99" href="main.html#link3">
  <area coords="50,50,99,99" href="main.html#link4">
</map>
```

Чтобы сделать карту совместимой с предшествующими версиями броузеров, можно одновременно включить ее обработку как со стороны сервера, так и со стороны клиента. Те броузеры, которые это умеют, предпочтут более быстрое обслуживание со стороны клиента. Все другие проигнорируют атрибут usemap в теге <img> и воспользуются указанным в ссылке обработчиком со стороны сервера, чтобы удовлетворить выбор читателя традиционным способом. К примеру:

```
<a href="/cgi-bin/images/map.proc">
  <img src="pics/map2.gif" usemap="#map2" ismap>
</a>
...
<map name="map2">
  <area coords="0,0,49,49" href="link1.html">
  <area coords="50,0,99,49" href="link2.html">
  <area coords="0,50,49,99" href="link3.html">
  <area coords="50,50,99,99" href="link4.html">
</map>
```

6.5.7. Эффективное использование карт

Многие из самых визуально привлекательных страниц в сети содержат изображения, чувствительные к щелчкам мышью и нажатиям на клавиши. В географических картах, например, щелчок на определенной области или выбор ее с помощью клавиш <Tab> и <Enter> вызывает появление дополнительной информации о стране, городе или местных представительствах корпорации и возможностях контакта с ними.

Нам доводилось видеть манекен-карту, чувствительными областями которой были части одежды со ссылками, ведущими к разделам каталога, содержавшего детальное описание шляпы или башмака и ценник для заказа.

Визуальная природа карт вместе с потребностью в эффективном интерфейсе подразумевают, что вам необходимо всерьез подумать о привлечении художника, разработчика пользовательских интерфейсов и даже эксперта по психологии, чтобы оценить такое произведение искусства. По меньшей мере, предложите кому-нибудь протестировать плоды вашего творчества в надежде узнать, догадываются ли люди, какую область изображения следует выбрать, чтобы получить нужный документ. Убедитесь, что чувствительные области изображения, употребляя согласованную систему визуальных приемов, объясняют пользователю свое назначение. Рассмотрите возможность применения рамок, отбрасываемой тени или выделения цветом для индикации областей, которые могут быть кем-то выбраны.

Наконец, не забывайте, что решение использовать карту явным образом исключает из ряда читателей владельцев как чисто текстовых, так и тех броузеров, на которых ограничена возможность вывода изображений. Это касается и броузеров, владельцы которых имеют медленное модемное соединение с Интернетом. Для таких людей загрузка ваших красивых картинок просто разорительна. Чтобы учесть запросы растущего населения Интернета, убедитесь, что каждая страница с картой имеет чисто текстовый эквивалент, к которому легко перейти по гиперссылке с источника, содержащего изображения. Некоторые заботливые вебмастеры создают даже разные варианты страниц – для тех, кто предпочитает полные графические версии документов, и для тех, кого интересует только текст.

6.6. Создание поисковых документов

В HTML есть еще один гибкий способ связывания документов без применения тега `<a>`. Он заставляет сервер искать в базе данных документ, содержащий указанные пользователем слово или группу слов. HTML-документ, содержащий в себе такое средство связывания, называется *поисковым* документом.

6.6.1. Тег <isindex> (нежелателен)

До того как он был объявлен нежелательным стандартами HTML 4 и XHTML, тег `<isindex>` использовался для передачи серверу ключевых слов вместе с URL поисковой машины. Затем сервер, сопоставляя полученные слова с содержимым указанной базы данных, выбирал следующий документ. Современные авторы для передачи данных серверу и вспомогательным программам используют формы. Подробности см. в главе 9.

<isindex>

Функция:	Означает, что документ является поисковым
Атрибуты:	action, class, dir, id, lang, prompt, style, title
Закрывающий тег:	Отсутствует в HTML; </isindex> или <isindex ... /> в XHTML
Содержит:	Ничего
Может содержаться в:	*head_content* (содержимое заголовка)

Когда броузер наталкивается на тег <isindex>, он вставляет в документ стандартный поисковый интерфейс (как его отображает Internet Explorer, можно увидеть на рис. 6.8):

```
<html>
<head>
<title>Kumquat Advice Database</title>
<base href="cgi-bin/quat-query">
<isindex>
</head>
<body>
<h3>Kumquat Advice Database</h3>
<p>
Search this database to learn more about kumquats!
</body>
</html>
```

Пользователь вводит в предоставленное окошко разделенные пробелами ключевые слова. Когда он нажимает клавишу <Enter>, броузер автоматически дописывает список запроса в конец URL и передает информацию серверу для дальнейшей обработки.

Хотя стандарты HTML и XHTML позволяют нежелательному тегу <isindex> появляться только в заголовке документа, большинство броузеров

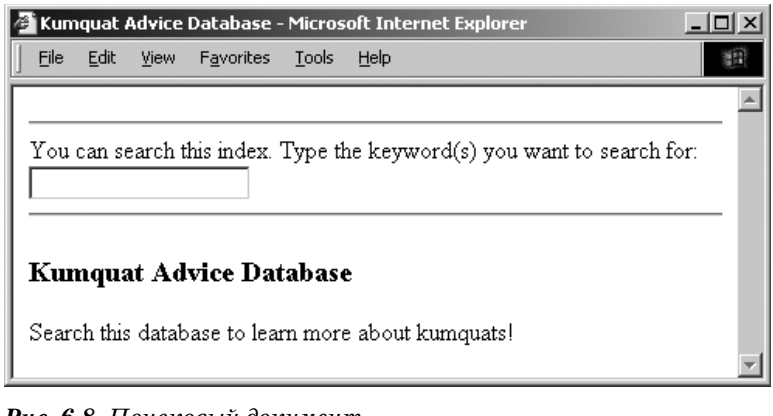

Рис. 6.8. *Поисковый документ*

разрешают ему находиться в документе где угодно и вставляют поле ввода в поток содержимого там, где встречается тег <isindex>. Это удобное расширение дает возможность добавлять инструкции и другие полезные элементы перед тем, как предоставить читателю само поле ввода.

6.6.1.1. Атрибут prompt

Перед полем ввода в поисковом документе (над ним или слева от него) броузер выводит подсказку. У броузера Internet Explorer эта подсказка меняется со временем. В версии 5, например, она такая: «You can search this index. Type the keyword(s) you want to search for:»[1] Новая подсказка версии 6 показана на рис. 6.8. Такая используемая по умолчанию подсказка не во всех случаях одинаково хороша, поэтому можно изменить ее при помощи атрибута prompt.

Значение атрибута prompt, вставленного в тег <isindex>, будет строкой текста, которую броузер выведет перед полем для ввода ключевых слов.

Мы добавили к предыдущему примеру другую подсказку. Сравните их (рис. 6.8–6.9):

```
<isindex prompt="To learn more about kumquats, enter a keyword:">
```

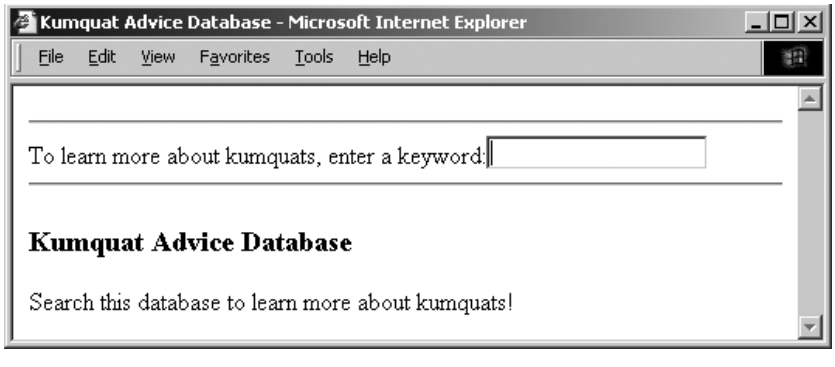

***Рис. 6.9.** Атрибут prompt позволяет установить текст подсказки в поисковом документе*

Старые броузеры проигнорируют атрибут prompt, но это не повод, чтобы лишить следящих за модой читателей уместной подсказки.[2]

6.6.1.2. URL запроса

Еще одним важным элементом поискового документа, кроме тега <isindex> в его заголовке, является URL запроса. По умолчанию это URL

[1] Разные версии даже одного броузера могут выдавать различный поясняющий текст. – *Примеч. науч. ред.*

[2] Напоминаем еще раз, что употребление тега <isindex> нежелательно. Лучше пользоваться формами. – *Примеч. науч. ред.*

именно поискового документа, что не очень хорошо, если ваш документ не умеет обрабатывать запросы. Многие авторы используют тег <base>, чтобы указать другой URL для поиска. [элемент заголовка <base>, 6.7.1]

Броузер добавляет к URL запроса вопросительный знак, за которым следуют определенные пользователем параметры поиска. Непечатаемые символы подходящим образом кодируются; элементы списка параметров разделяются знаком «плюс» (+).[1]

В предыдущем примере, если пользователь наберет в поле ввода «insect control», броузер создаст URL:

```
cgi-bin/quat-query?insect+control
```

6.6.1.3. Атрибут action

Только для Internet Explorer можно указать URL запроса, применяя атрибут action. Результат будет таким же, как если бы вы добавили этот URL в атрибут href тега <base>, – броузер передаст по указанному URL присоединенные к нему параметры.

Хотя атрибут action и «разводит» базовый URL документа и URL поискового сервера, но его использование приведет к неудаче поиска, если броузер не является Internet Explorer. Поэтому мы не рекомендуем применять атрибут action для определения URL запроса.

6.6.1.4. Атрибуты class, dir, id, lang и style

Атрибуты class и style дают возможность определить свойства отображения и имена классов для управления внешним видом тега, хотя их ценность для <isindex> представляется ограниченной. Атрибуты id и title позволяют создать имя и название для тега. Имя может употребляться в дальнейшем для ссылок. [встроенные стили: атрибут style, 8.1.1] [стилевые классы, 8.3]

Атрибуты lang и dir означают язык, применяемый для этого тега, и направление, в котором он выводится. Способ их использования с <isindex> также неясен. [атрибут dir, 3.6.1.1] [атрибут lang, 3.6.1.2]

6.6.1.5. Зависимость от сервера

Подобно серверным картам, поисковые документы нуждаются для своей работы в поддержке со стороны сервера. То, как сервер будет интерпретировать URL запроса, не определяется стандартами HTML или XHTML.

Вам придется изучить документацию сервера, чтобы узнать, как можно получить и использовать параметры поиска для нахождения нужного документа. Обычно сервер отрезает параметры от URL запроса и передает их программе, обозначенной в URL.

---

[1] В строке параметров плюсом заменяются все пробелы. – *Примеч. науч. ред.*

6.7. Отношения

Очень немногие документы существуют сами по себе. Наоборот, обычно они являются частью собрания, где каждый источник соединен с другими нитями гиперссылок, описанных в этой главе. Один документ может входить в несколько собраний, содержать ссылки на некоторые документы и являться целью ссылок из других. Читатели перемещаются от семейства к семейству, выбирая интересующие их гиперссылки.

Вы устанавливаете явное отношение между двумя документами, вставляя в один из них гиперссылку на другой. Добросовестные авторы для обозначения природы этой связи используют атрибут `rel` тега `<a>`. Кроме того, внести большую ясность о месте документа в собрании и его отношении к разным членам семейства помогут еще два тега. Эти теги, `<base>` и `<link>`, помещаются в теле тега `<head>`. [тег <head>, 3.7.1]

6.7.1. Элемент заголовка <base>

Как уже говорилось, URL в документе могут быть либо абсолютными (включающими в себя все элементы URL, явно указываемые автором), либо относительными (в которых некоторые элементы опущены и добавлены потом броузером). Обычно броузер заполняет недостающие части, используя URL текущего документа. Вы можете изменить это при помощи тега `<base>`.

<base>

Функция:	Определяет базовый URL для якорей в документе
Атрибуты:	`href`, `target`
Закрывающий тег:	Отсутствует в HTML; `</base>` или `<base ... />` в XHTML
Содержит:	Ничего
Может содержаться в:	*head_content* (содержимое заголовка)

Тег `<base>` должен появляться только в заголовке документа, но не в его теле. Впредь броузер будет трактовать базовым URL не текущего документа, а указанный в теге, и применять его для пополнения относительных URL, включая те, что он найдет в тегах `<a>`, `<link>` или `<form>`. Тег `<base>` определяет также URL, который будет использоваться при обработке запросов в поисковых документах, содержащих тег `<isindex>`. [ссылки на документы: URL, 6.2]

6.7.1.1. Атрибут href

Атрибут `href` должен получать в качестве значения допустимый URL, который броузер затем употребляет как базовый для приведения относительных URL к абсолютным. К примеру, тег `<base>` в заголовке следующего XHTML-документа:

```
<head>
<base href="http://www.kumquat.com/" />
</head>
...
```

сообщает броузеру, что все относительные URL в документе «отсчитываются» от каталога самого верхнего уровня на сервере *www.kumquat.com* независимо от того, с какой машины и из какого каталога был получен этот документ.

Вопреки вашим допустимым ожиданиям, можно сделать базовый URL относительным, а не абсолютным. Броузер должен сформировать (но отнюдь не всегда формирует) абсолютный базовый URL, заполнив пропуски в относительном, используя URL самого документа. Это свойство может сослужить добрую службу. К примеру, обрабатывая следующий HTML-текст:

```
<head>
<base href="/info/">
</head>
...
```

броузер сделает базовым URL текущего документа относящимся к каталогу */info* сервера, который, вероятно, не тот, где содержится сам документ. Вообразите, что вам пришлось бы переадресовывать каждую ссылку в документе на этот общий каталог. Тег `<base>` не только помогает сократить запись URL, имеющих общий корень, но и позволяет установить каталог, от которого отсчитываются относительные ссылки, не привязывая документ к определенному серверу.

6.7.1.2. Атрибут target

При работе с документами, расположенными во фреймах, атрибут `target` тега `<a>` гарантирует, что запрошенный документ будет выведен в нужный фрейм. Подобным образом атрибут `target` для тега `<base>` позволяет установить имя фрейма или окна, в которые будет по умолчанию помещаться документ, полученный по перенаправленной гиперссылке. [обзор фреймов, 11.1]

Если у вас нет оснований отдать предпочтение какому-либо из фреймов, вы, вероятно, захотите подумать об использовании `<base target =_top>`. Этот текст отвечает за то, что гиперссылки, не направленные в какой-либо определенный фрейм или окно, будут отображаться броузером в окне верхнего уровня. Такой подход исключает неприятную и распространенную ошибку, когда документы с других сайтов загружаются в один из фреймов вместо того, чтобы полностью занять окно броузера. Это облегчит жизнь ваших читателей.

6.7.1.3. Использование тега <base>

Главная причина использования тега `<base>` – он позволяет гарантировать, что все относительные ссылки в документе будут дополнены до

правильных адресов, даже если сам документ перемещен или переименован. Это особенно важно при создании собраний документов. Поместив правильный тег <base> в каждый из них, можно затем смело перемещать коллекцию из каталога в каталог и даже с сервера на сервер, не разрушая при этом ее ссылочной целостности. Кроме того, придется употреблять тег <base> в поисковых документах, если вы хотите, чтобы пользовательские запросы отправлялись по URL, отличному от адреса самого поискового документа.

С документами, которые одновременно содержат тег <isindex> и другие относительные URL, могут быть проблемы, если относительные адреса отсчитываются не от URL обработчика запросов. Поскольку обычно так и бывает, не употребляйте относительных ссылок в поисковых документах, содержащих тег <base>, определяющий URL запросов.

6.7.2. Элемент заголовка <link>

Используйте тег <link> для определения отношения, в котором находится документ к другому элементу собрания.

Тег <link> допустим только в теге <head> и нигде более. Атрибуты тега <link> используются так же, как атрибуты тега <a>, но они служат только для документирования отношений между документами. У тега <link> нет содержимого, и только в XHTML поддерживается закрывающий </link>.

<link>

Функция:	Определяет отношение этого документа к другому документу
Атрибуты:	charset, class, dir, href, hreflang, id, lang, media, onClick, onDblClick onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, rel, rev, style, target title, type
Закрывающий тег:	Отсутствует в HTML; </link> или <link ... /> в XHTML
Содержит:	Ничего
Может содержаться в:	*head_content* (содержимое заголовка)

6.7.2.1. Атрибут href

Как и в других тегах, атрибут href определяет URL документа-цели. Это обязательный атрибут, и его значением может быть любой действительный URL. Предполагается, что указанный в href документ имеет какое-то отношение к текущему документу, в котором атрибут задан.

6.7.2.2. Атрибуты rel и rev

Атрибуты rel и rev выражают формальные отношения между документами, являющимися источником и целью гиперссылки. Атрибут rel

определяет отношение документа-источника к цели, а атрибут rev – отношение цели к источнику. Оба атрибута могут помещаться в одном теге <link>.

Значениями атрибутов rel и rev являются списки отношений, разделителями в которых служат пробелы. Действительные названия отношений и их смысл в ваших руках, о них нет ничего в стандарте HTML, хотя некоторые из названий стали общеупотребительными. К примеру, документ, являющийся элементом последовательности документов, может содержать в своих ссылках такие значения атрибутов отношений:

```
<link href="part-14.html" rel=next rev=prev>
```

Отношение источника к цели соответствует переходу к следующему в последовательности документу. Обратное отношение соответствует переходу к предыдущему документу.

6.7.2.3. Атрибут title

Атрибут title позволяет определить название для документа, на который направлена гиперссылка. Атрибут title особенно удобен при ссылках на ресурсы, не имеющие собственного названия, на изображения или отличные от HTML документы. К примеру, броузер может употреблять название из тега <link> при отображении документа, на который направлена ссылка. В частности:

```
<link href="pics/kumquat.gif"
   title="Фотография Благородного Фрукта">
```

предлагает броузеру использовать присвоенное название при выводе изображения, на которое указывает ссылка.

6.7.2.4. Атрибут type

Атрибут type определяет MIME-тип содержимого документа, на который ссылается тег <link>. Поддерживаемый всеми популярными броузерами, включенный в стандарты HTML 4 и XHTML атрибут может использоваться с любым документом, на который направлена гиперссылка. Он часто применяется для указания типа присоединенных таблиц стилей. В таком контексте значением атрибута обычно бывает text/css. Например:

```
<link href="styles/classic.css" rel=stylesheet type="text/css">
```

создает ссылку на внешнюю таблицу стилей в заголовке документа. Подробная информация – в главе 8.

6.7.2.5. Как броузеры могут использовать <link>

Хотя стандарты и не требуют, чтобы броузеры делали что-нибудь с информацией, содержащейся в теге <link>, нетрудно вообразить, как она могла бы использоваться для улучшенного представления документа.

В простейшем случае предположим, что вы систематически применяли тег <link>, в каждом из ваших документов указывая на отношения «следующий», «предыдущий», «родительский». Броузер мог бы использовать эту информацию, помещая стандартную панель инструментов в верхней или нижней части каждого документа. Нажимая на кнопки такой панели, удавалось бы перепрыгивать к документам, находящимся с текущим в соответствующих отношениях. Переложив задачу обеспечения простых связей в коллекции на броузер, вы могли бы сосредоточиться на более важном содержимом своего документа.

В более сложном примере допустим, что броузер обнаруживает словарь, который определяется в теге <link> и в то же время является поисковым документом. Когда бы пользователь ни щелкал на слове или фразе на странице, броузер автоматически станет просматривать словарь и представлять определения выбранных слов в маленьком всплывающем окошке.

С развитием сети будут возникать все новые и новые применения тега <link> для явного задания взаимоотношений документов.

6.7.2.6. Другие атрибуты тега <link>

Стандарты HTML 4 и XHTML предоставляют тегу <link> универсальный набор атрибутов, относящихся к таблицам стилей, событиям и к языку. Вы можете заглянуть в соответствующие разделы, описывающие эти атрибуты для тега <a>, чтобы получить полное описание их употребления. [тег <a>, 6.3.1]

Поскольку тег <link> расположен в теге <head>, содержимое которого не воспроизводится, может показаться, что эти атрибуты бесполезны. Совсем не исключено, что броузеры будущего сумеют найти какой-то способ представления информации, находящейся в <link>, – возможно, в виде панели навигационных инструментов или набора рекомендуемых переходов. В таких случаях информация о способах отображения окажется полезной. В настоящее время броузеры не предоставляют этих возможностей.

6.8. Поддержка автоматической обработки и создания документов

Существует еще два тега заголовка, основная задача которых состоит наряду с поддержкой автоматической обработки во взаимодействии с веб-сервером и средствами автоматического создания документов.

6.8.1. Элемент заголовка <meta>

Познакомившись с уймой тегов заголовка, определяющих вид и связи документа, да еще таких, которыми большинство авторов не пользуются, впору было бы подумать, а не пора ли остановиться. А вот и нет!

Всегда есть кто-то с особенными нуждами. Такие авторы намереваются дать еще больше сведений о своем драгоценном материале, представить информацию, которая могла бы использоваться броузерами, читателями исходного текста или программами, составляющими индексы документов. Тег <meta> для вас – тех, кто хочет получить больше большего.

<meta>

Функция:	Доставляет дополнительную информацию о документе
Атрибуты:	charset [1], content, dir, http-equiv, lang, name, scheme
Закрывающий тег:	Отсутствует в HTML; </meta> или <meta ... /> в XHTML
Содержит:	Ничего
Может содержаться в:	*head_content* (содержимое заголовка)

Тег <meta> включается в заголовок документа и не имеет собственного содержимого. Вместо этого атрибуты тега определяют пары имя/значение, которые характеризуют документ.

В определенных случаях такие значения используются веб-сервером, чтобы предоставить броузеру тип содержимого документа.

6.8.1.1. Атрибут name

Атрибут name хранит компонент имени для пары имя/значение, устанавливаемой в теге <meta>. Ни HTML, ни XHTML не определяют стандартных имен для тега <meta>. В целом, можно свободно употребить любое название, имеющее смысл для вас и других читателей исходного текста.

Одним из распространенных имен является keywords, определяющее набор ключевых слов документа. Обнаруженные какой-нибудь из популярных в сети поисковых машин, они будут использованы для включения документа в тематический каталог. Если вы хотите, чтобы документ попал в базу данных поисковой машины, позаботьтесь о внесении подобного тега в каждый из ваших документов:

```
<meta name="keywords" content="kumquats, cooking, peeling, eating">
```

Если атрибут name отсутствует, имя для пары имя/значение берется из атрибута http-equiv.

6.8.1.2. Атрибут content

Атрибут content хранит компонент значения в паре имя/значение. Это может быть любая строка, которую необходимо заключить в кавычки, если она содержит пробелы. Атрибут content всегда определяется в па-

ре либо с атрибутом name, либо с атрибутом http-equiv. К примеру, можно поместить в документ имя автора, написав:

```
<meta name="Authors" content="Chuck Musciano & Bill Kennedy">
```

6.8.1.3. Атрибут http-equiv

Атрибут http-equiv хранит имя в паре имя/значение и предписывает серверу включить такую пару в MIME-заголовок документа, передаваемый броузеру перед самим документом. Когда сервер посылает документ броузеру, он сначала отправляет несколько пар вида имя/значение. Хотя некоторые серверы могут передавать набор таких пар, как минимум одну посылает каждый из них:

```
content-type: text/html
```

Этот текст предупреждает броузер о скором прибытии HTML-документа. Когда тег <meta> применяется с атрибутом http-equiv, сервер добавляет пары[1] имя/значение в заголовок[2], описывающий содержимое файла, отправляемого броузеру. Например, следующий текст:

```
<meta http-equiv="charset" content="iso-8859-1">
<meta http-equiv="expires" content="31 Dec 99">
```

приведет к тому, что отправленный броузеру заголовок будет содержать следующие строки, причем вставка этих дополнительных данных в заголовок имеет смысл, только если броузер принимает их и подходящим образом использует.

---

[1] Иногда используют сокращенную форму записи, когда последующие параметры (в виде имя:значение) включаются через точку с запятой (;) после значения параметра, имя которого указано в http-equiv как <meta http-equiv="content-type" content="text/html; charset: windows-1251">. – *Примеч. науч. ред.*

[2] Большинство современных веб-серверов не обрабатывают содержимое тегов <meta> передаваемых документов и не формируют из содержимого этих тегов http-заголовок. Как правило, веб-сервер формирует заголовок, исходя из собственных настроек и информации, переданной броузером при запросе. Сервер может перекодировать документ до передачи его клиенту (то же самое могут сделать с документом и промежуточные серверы (proxy)). Обычно при такой перекодировке содержимое тегов <meta> не изменяется, но если в нем был указан набор символов (charset), то он после перекодировки станет ошибочным. Стандарт HTML предполагает приоритет charset, указанного в заголовке, формируемом сервером, над charset из тега <meta>. Но некоторые броузеры (Internet Explorer 4.0, Netscape Navigator 4.x) неправильно отрабатывают приоритеты для charset, что делает документы нечитаемыми. Поэтому лучше вовсе не указывать charset в теге <meta>. Отсутствие же charset в http-заголовке автоматически соотносит передаваемый документ с кодировкой ISO8859-1 (в которой нет символов кириллицы), и такие документы обрабатываются соответствующим образом на proxy-серверах и броузером. Поэтому позаботьтесь о том, чтобы ваш веб-сервер отдавал документы с правильным значением charset в заголовке. – *Примеч. науч. ред.*

```
content-type: text/html
charset: iso-8859-1
expires: 31 Dec 99
```

6.8.1.4. Атрибут charset

Internet Explorer версии 5 и более ранних явным образом поддерживает атрибут `charset` для тега `<meta>`. Присвойте атрибуту значение, являющееся названием набора символов, который должен употребляться в документе. Это не тот способ, который следует рекомендовать для определения выбранного набора. Мы советуем использовать вместо него атрибуты `http-equiv` и `content`, как в предыдущем примере.

6.8.1.5. Атрибут scheme

Этот атрибут определяет схему, которую следует употреблять для интерпретации значений свойств. Данная схема должна быть определена в профиле, указанном в атрибуте `profile` тега `<head>`. [тег <head>, 3.7.1]

6.8.2. Элемент заголовка <nextid> (архаизм)

Данный тег не принадлежит стандартам HTML 4 или XHTML, и его не следует употреблять. Мы описываем его из уважения к истории. Замысел `<nextid>` состоял в том, чтобы предоставить некий способ автоматического индексирования идентификаторов фрагментов.

<nextid>	
Функция:	Определяет идентификатор следующего документа
Атрибуты:	`n`
Закрывающий тег:	Отсутствует
Содержит:	Ничего
Может содержаться в:	*head_content* (содержимое заголовка)

6.8.2.1. Атрибут n

Атрибут `n` определяет имя очередного по порядку генерируемого идентификатора фрагмента. Обычно это последовательность букв, за которой идет двузначное число. Типичный тег `<nextid>` выглядит примерно так:

```
<html>
<head>
<nextid n=DOC54>
</head>
...
```

Автоматический генератор документов может использовать данные из `<nextid>` для последовательного присвоения имен: `DOC54`, `DOC55` и т. д.

- Неупорядоченные списки
- Упорядоченные списки
- Тег
- Вложенные списки
- Списки определений
- Как использовать списки
- Директории
- Меню

7

Форматированные списки

Представление информации в более удобном виде – это единственное наиболее важное достоинство HTML и его потомка XHTML. Превосходная коллекция текстовых стилей и средств форматирования, входящих в состав этих языков, помогает организовать информацию в документы, которые читатели быстро просматривают, легко понимают и делают выдержки, иногда задействуя автоматические способы обработки.

Помимо возможности украсить текст с помощью специализированных текстовых тегов, HTML и XHTML предоставляют также богатый набор орудий, позволяющих упорядочить содержимое документа в виде форматированных списков. В них нет ничего волшебного или таинственного. В действительности красота списков заключена в их простоте. Они основаны на общеизвестных образцах, с которыми мы встречаемся ежедневно, таких как неупорядоченный список покупок, упорядоченный перечень действий в инструкции и похожий на словарь список определений. Все они являются хорошо знакомыми удобными способами организации информации. И все они помогают лучше усвоить ее при беглом просмотре, а также найти и извлечь из веб-страниц нужные сведения.

7.1. Неупорядоченные списки

Подобно перечню покупок или сданных в прачечную вещей, *неупорядоченный список* представляет собой совокупность однородных элементов, между которыми нет отношений порядка или следования. Самые распространенные в сети неупорядоченные списки – это собрания гиперссылок на другие документы. Общая тема, например «Сайты любителей кумкватов», связывает отдельные элементы в неупорядоченный список, который можно читать в каком угодно порядке.

7.1.1. Тег

Тег <ul> сигнализирует броузеру, что все, следующее за ним, вплоть до закрывающего тега , представляет собой неупорядоченный список элементов. Внутри неупорядоченного списка каждый элемент отмечается тегом <li>. Кроме того, в список может входить практически любое HTML/XHTML-содержимое, включая другие списки, текст и мультимедийные элементы. [тег <li>, 7.3]

Функция:	Определяет неупорядоченный список
Атрибуты:	class, compact [!], dir, id, lang, onClick, onDblClick, onKeyDown, onKeyPress onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, style, title, type [!]
Закрывающий тег:	; присутствует обязательно
Содержит:	*list_content* (содержимое списка)
Может содержаться в:	*блоке*

При отображении броузер обычно предваряет любой элемент списка специальным маркером и, располагая каждый элемент неупорядоченного списка с новой строки, применяет к нему какое-нибудь форматирование, например отступ от левого края окна. Действительное отображение неупорядоченных списков хотя и сходно у популярных броузеров (рис. 7.1), не определяется стандартами, так что не надо увлекаться точным позиционированием элементов на экране.

Вот пример неупорядоченного списка, описанного в соответствии со стандартом XHTML (рис. 7.1):

```
Popular Kumquat recipes:
<ul>
  <li>Pickled Kumquats</li>
  <li>'Quats and 'Kraut (a holiday favorite!)</li>
```

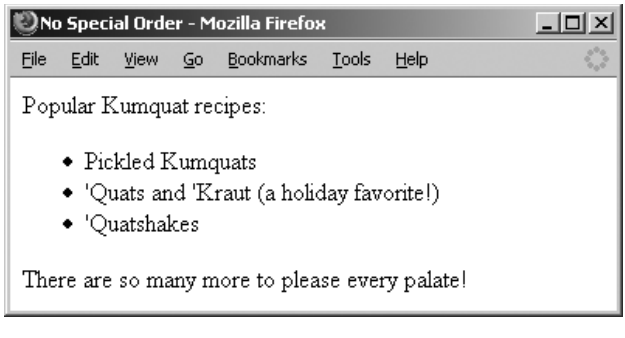

Рис. 7.1. *Простой неупорядоченный список*

```
    <li>'Quatshakes</li>
  </ul>
  There are so many more to please every palate!
```

Ловкие HTML-авторы иногда применяют вложенные неупорядоченные списки с элементами, отмеченными тегом <li>, и без него, чтобы воспользоваться автоматической системой создания последовательных отступов. Вы сможете соорудить кое-какие хитроумные фрагменты, применяя этот способ. Но только не полагайтесь на него, имея дело с различными броузерами, особенно с теми, что появятся в будущем. Лучше применить свойство border в определениях стиля в тегах абзаца (<p>) или раздела ( ), чтобы организовать отступы в секциях вашего документа, не являющихся списками (см. главу 8).

7.1.1.1. Атрибут type

Графические броузеры автоматически снабжают каждый отмеченный тегом <li> элемент неупорядоченного списка специальным маркером. В частности, Netscape и Firefox выводят ромбик, как показано на рис. 7.1, а Internet Explorer и Opera применяют сплошной закрашенный кружок. Броузеры, поддерживающие HTML 3.2 и более поздние версии, включая 4.0 и 4.1, так же как и XHTML 1.0, позволяют употреблять атрибут type для определения того, какой символ маркера будет отмечать элементы неупорядоченного списка. Этот атрибут может принимать такие значения: disc (диск), circle (круг), square (квадрат). Все элементы в списке будут отмечаться указанным маркером, если только отдельный элемент не отменит его спецификацию. Как это делается, описано далее в этой главе.

С появлением стандарта каскадных таблиц стилей W3C признал нежелательным в HTML 4 и XHTML атрибут type. Ожидается, что он выйдет из употребления.

7.1.1.2. Компактные неупорядоченные списки

Если вы любите широкие открытые пространства, вам не понравится необязательный атрибут тега <ul> compact. Он предлагает броузеру сжать неупорядоченный список в компактный блок меньшего размера. Обычно броузер уменьшает расстояние между представленными элементами. И если он что-нибудь и делает с отступами в списке, так убавляет их (обычно броузер ничего с ними не делает).

Некоторые броузеры игнорируют атрибут compact, поэтому не следует полагаться на его свойства форматирования. Кроме того, этот атрибут признан нежелательным в стандартах HTML 4 и XHTML, так что ему недолго осталось жить.

7.1.1.3. Атрибуты class и style

Атрибуты class и style дают возможность управления отображением списков при помощи каскадных таблиц стилей, что позволяет осуще-

ствлять более совершенный контроль, нежели тот, которого можно достичь с применением отдельных атрибутов, подобных type. Включайте атрибут style в тег <ul>, чтобы назначить в качестве маркеров ваши собственные пиктограммы, вместо того чтобы употреблять общепринятые кружки, диски и квадратики. Атрибут class позволяет применять к содержимому тега стиль, заранее определенный для данного класса списков. Значение атрибута class – это имя стиля, определенного в таблице стилей на уровне документа или внешним образом. [встроенные стили: атрибут style, 8.1.1] [стилевые классы, 8.3]

7.1.1.4. Атрибуты lang и dir

Атрибут lang позволяет определить язык, который употребляется в списке, а атрибут dir дает возможность посоветовать броузеру, в каком направлении следует выводить текст. Значение атрибута lang – это двухбуквенный код языка по стандарту ISO, включающий необязательный языковой модификатор. К примеру, lang=en-UK сообщает броузеру, что список написан по-английски, причем так, как говорят и пишут в Великобритании (United Kingdom). Предполагается, что броузер как-то отразит в макете и типографских решениях ваш выбор языка.

Атрибут dir сообщает броузеру, в каком направлении следует отображать содержимое списка: слева направо (dir=ltr), как в английском или русском языках, или справа налево (dir=rtl) для таких языков, как китайский или иврит. [атрибут dir, 3.6.1.1] [атрибут lang, 3.6.1.2]

7.1.1.5. Атрибуты id и title

Используйте атрибут id для обеспечения неупорядоченного списка меткой. Допустимое значение атрибута – заключенная в кавычки строка, которая уникальным образом идентифицирует список и позволяет употреблять его в качестве цели гиперссылок и селектора в таблицах стилей, для автоматического поиска и ряда других приложений. [атрибут id, 4.1.1.4]

Вы также можете использовать необязательный атрибут title и заключенную в кавычки строку – значение атрибута – для дополнительной идентификации списка. В отличие от id, значение атрибута title не обязано быть уникальным. [атрибут title, 4.1.1.5]

7.1.1.6. Атрибуты событий

Многие относящиеся к поведению пользователя события как внутри списка, так и вне него, такие как одинарный или двойной щелчок мышью в области отображения списка, распознаются современными броузерами. С помощью соответствующих on-атрибутов и их значений можно реагировать на эти события, отображая окно диалога с пользователем или активизируя какое-нибудь мультимедийное событие. [обработчики событий JavaScript, 12.3.3]

7.2. Упорядоченные списки

Используйте упорядоченные списки, когда порядок элементов в них имеет существенное значение. Инструкция по установке стиральной машины, оглавление книги или список сносок и примечаний – все это хорошие примеры упорядоченных списков.

7.2.1. Тег

Типичный броузер форматирует содержимое упорядоченного списка точно так же, как и неупорядоченного, за тем лишь исключением, что элементы списка снабжаются номерами вместо маркеров. Нумерация начинается с единицы и каждый следующий отмеченный тегом `<li>` элемент списка получает номер на единицу больше, чем предыдущий. [тег <li>, 7.3]

<ol>

Функция:	Определяет упорядоченный список
Атрибуты:	`class`, `compact`, `dir`, `id`, `lang`, `onClick`, `onDblClick`, `onKeyDown`, `onKeyPress`, `onKeyUp`, `onMouseDown`, `onMouseMove`, `onMouseOut`, `onMouseOver`, `onMouseUp`, `start`, `style`, `title`, `type`
Закрывающий тег:	`</ol>`; присутствует обязательно
Содержит:	*list_content* (содержимое списка)
Может содержаться в:	*блоке*

HTML 3.2 ввел ряд средств для создания разнообразных упорядоченных списков. Можно изменять начальное значение и выбирать один из пяти различных стилей нумерации. Вот пример упорядоченного списка в XHTML:

```
<h3>Pickled Kumquats</h3>
Here's an easy way to make a delicious batch of pickled 'quats:
<ol>
  <li>Rinse 50 pounds of fresh kumquats</li>
  <li>Bring eight gallons white vinegar to rolling boil</li>
  <li>Add kumquats gradually, keeping vinegar boiling</li>
  <li>Boil for one hour, or until kumquats are tender</li>
  <li>Place in sealed jars and enjoy!</li>
</ol>
```

Opera отобразит его так (рис. 7.2).

7.2.1.1. Атрибут start

Как правило, броузеры автоматически нумеруют элементы упорядоченного списка, начиная с арабской цифры «1». Атрибут `start` тега

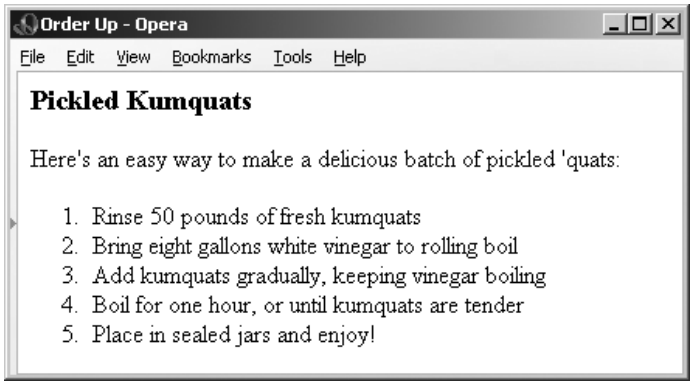

***Рис. 7.2.** Упорядоченный список*

`<ol>` позволяет изменить начальное значение. Чтобы начать нумерацию в списке, к примеру, с цифры 5, напишите:

```
<ol start=5>
  <li> Это пункт номер 5.</li>
  <li> Это номер 6!</li>
  <li> И так далее...</li>
</ol>
```

7.2.1.2. Атрибут type

По умолчанию броузеры нумеруют элементы упорядоченного списка арабскими цифрами. Помимо выбора другого начального номера, отличного от единицы, можно также использовать атрибут `type` тега `<ol>`, чтобы изменить сам стиль нумерации. Значениями атрибута `type` в теге `<ol>` могут быть: `A` – для нумерации заглавными буквами, `a` – строчными буквами, `I` – римскими цифрами в верхнем регистре, `i` – римскими цифрами в нижнем регистре, а также `1` – для обычных арабских цифр (табл. 7.1).

Таблица 7.1. Типы нумерации упорядоченных списков в HTML

Тип	Стиль	Образец последовательности
`A`	Заглавные буквы	A, B, C, D
`a`	Строчные буквы	a, b, c, d
`I`	Заглавные римские цифры	I, II, III, IV
`i`	Строчные римские цифры	i, ii, iii, iv
`1`	Арабские цифры	1, 2, 3, 4

Атрибуты `start` и `type` работают в тандеме. Атрибут `start` устанавливает начальное значение целочисленного счетчика элементов в начале упорядоченного списка. Атрибут `type` назначает стиль нумерации. В част-

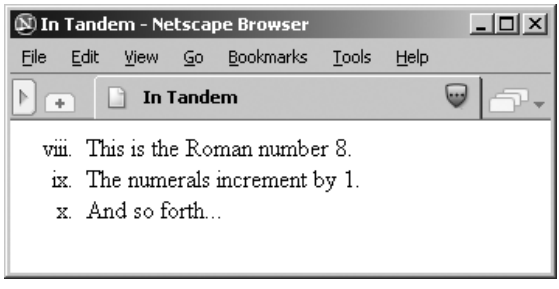

Рис. 7.3. *Атрибуты start и type работают в тандеме*

ности, приведенный ниже упорядоченный список начинает отсчет элементов с номера 8, но поскольку стиль нумерации определяется значением `i`, первым номером в списке будет римское «viii». Следующие элементы нумеруются в том же стиле, при этом их номера последовательно увеличиваются на единицу, как это определено в HTML-примере:[1]

```
<ol start=8 type="i">
  <li> This is the Roman number 8.
  <li> The numerals increment by 1.
  <li> And so forth...
</ol>
```

Результат представлен на рис. 7.3.

Тип и значение отдельных пунктов в списке могут быть не такими, как у списка в целом (см. разд. 7.3.1). Как было сказано ранее, атрибуты `start` и `type` объявлены нежелательными в HTML 4 и XHTML. Используйте вместо них таблицы стилей.

7.2.1.3. Компактный упорядоченный список

Подобно тегу `<ul>`, тег `<ol>` имеет необязательный атрибут `compact`, нежелательный в стандартах HTML 4 и XHTML. Без крайней нужды не используйте его.

7.2.1.4. Атрибуты class, dir, event, id, lang, style и title

Эти атрибуты применимы также и к упорядоченным спискам, при этом имеют тот же эффект, что и в случае неупорядоченных списков. [атрибуты class и style, 4.1.1.6] [атрибуты lang и dir, 6.3.1.7] [атрибуты id и title, 7.1.1.5] [атрибуты событий, 6.3.1.4]

---

[1] Заметьте, что в этот HTML-пример мы не включили закрывающие теги `</li>`, как это делали в предыдущих XHTML-примерах. Некоторых закрывающих тегов может не быть в HTML, но они обязательно должны присутствовать во всех XHTML-документах.

7.3. Тег

Вам должно быть уже совершенно ясно, что тег <li> определяет элемент списка. Этот универсальный тег применяется как в рассмотренных упорядоченных (<ol>) и неупорядоченных (<ul>) списках, так и в директориях (<dir>) и меню (<menu>), которые мы еще подробно обсудим в этой главе.

В связи с тем, что завершение списка всегда может быть логически выведено из окружающей структуры документа, большинство авторов опускают тег . В этом есть смысл, поскольку так легче добавлять, удалять и перемещать элементы списка. Тем не менее XHTML требует, чтобы закрывающий тег присутствовал, так что лучше включайте его в документы.

Хотя значение тега <li> универсально, употребляемый в тегах списков различных типов он подвергается некоторым вариациям и специальным ограничениям. В упорядоченных и неупорядоченных списках за тегом <li> может следовать практически все что угодно, включая другие списки и последовательности абзацев. Броузер, если он вообще использует отступы, обычно последовательно применяет их к каждому вложенному списку, и содержимое любого элемента такой иерархии выравнивается по границе самого глубокого вложения для данного элемента.

<li>

Функция:	Определяет элемент упорядоченного и неупорядоченного списков, директорий и меню
Атрибуты:	class, dir, id, lang, onClick, onDblClick, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, style, title, type, value
Закрывающий тег:	; часто опускается в HTML
Содержит:	*flow* (поток текста)
Может содержаться в:	*list_content* (содержимое списка)

Списки директорий и меню – это другое дело. Они состоят из коротких элементов, таких как отдельное слово или простое словосочетание. Соответственно элементы списка в тегах <dir> и <menu> не могут содержать других списков или других элементов блочной природы, включая абзацы, преформатированные блоки или формы.

Корректно написанные документы, полностью соответствующие стандартам HTML и XHTML, не должны допускать внутри упорядоченных или неупорядоченных списков, директорий и меню ничего, что не было бы заключено в тегах <li>. Большинство броузеров снисходительно относятся к нарушению этого правила, но нельзя винить броузер, если в исключительных случаях он позволит себе следовать стандарту.

7.3.1. Изменение стиля и порядкового номера отдельного элемента списка

Помимо изменения стиля маркирования и нумерации всех элементов, в неупорядоченном и упорядоченном списках можно изменить стиль отдельного элемента. В упорядоченных списках подобная операция допустима и для номера.

Как будет видно дальше, комбинирование изменений стиля и нумерации ведет к разнообразию структур списков, особенно при использовании вложенных объектов. Отметьте, однако, что стандарты признали нежелательными эти атрибуты в пользу их CSS-аналогов.

7.3.1.1. Атрибут type

Возможными значениями атрибута `type` в теге `<li>` являются те же значения, что используются для этого атрибута в тегах соответствующих списков: элементы неупорядоченного списка могут иметь типы `circle` (круг), `square` (квадрат) или `disc` (диск), а элементы упорядоченного списка могут иметь тип из числа приведенных в табл. 7.1.

Будьте осторожны. В старых броузерах, таких как Netscape Navigator и Internet Explorer версии 4 и более ранних, изменение типа маркирования и нумерации одного элемента списка переносилось и на следующие за ним. Иначе обстоит дело с броузерами, поддерживающими стандарт HTML 4, такими как Netscape версии 6 и Internet Explorer версии 5 и более поздний, Firefox и Opera. Для них атрибут `type` действует локально только на содержимое «своего» тега `<li>`. Стиль последующих элементов списка возвращается к принятому по умолчанию. Если вас это не устраивает, ставьте свои атрибуты `type` в последующие элементы списка.

Атрибут `type` изменяет стиль вывода номера отдельного пункта списка, причем только этого пункта, не затрагивая значение счетчика пунктов, все время увеличивающееся на единицу. Рис. 7.4 показывает влияние изменения стиля отдельного элемента упорядоченного спи-

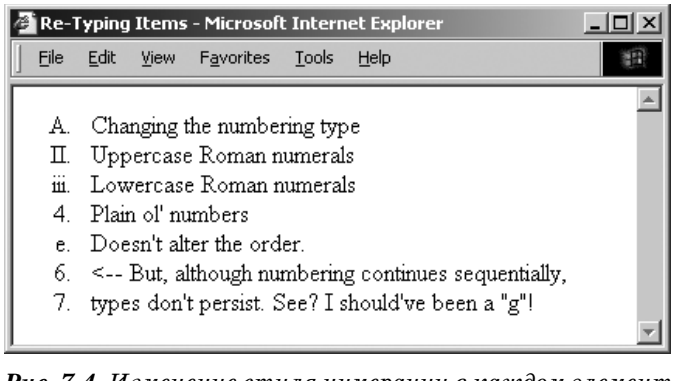

***Рис. 7.4.** Изменение стиля нумерации в каждом элементе упорядоченного списка*

ска на вид следующего за ним при отображении броузером такого фрагмента XHTML-кода:

```
<ol>
  <li type=A>Changing the numbering type</li>
  <li type=I>Uppercase Roman numerals</li>
  <li type=i>Lowercase Roman numerals</li>
  <li type=1>Plain ol' numbers</li>
  <li type=a>Doesn't alter the order.</li>
  <li> &lt;-- But, although numbering continues sequentially,</li>
  <li> types don't persist. See? I should've been a "g"!</li>
</ol>
```

Можно использовать относящиеся к таблицам стилей атрибуты `class` и `style` для воздействия на отдельные элементы упорядоченных и неупорядоченных списков, которое будет или не будет переноситься на последующие элементы списка. Смотрите главу 8 (в особенности раздел 8.4.8.5).

7.3.1.2. Атрибут value

Атрибут `value` изменяет номер определенного элемента списка и тех, что за ним следуют. Поскольку элементы последовательно нумеруются только в упорядоченных перечнях, атрибут `value` действителен лишь в теге `<li>`, содержащемся в упорядоченном списке.

Чтобы изменить номер, присоединенный к текущему элементу (и следующим за ним) в упорядоченном списке, просто присвойте `value` целое значение. Следующий XHTML-пример использует атрибут `value`, чтобы перескочить через несколько номеров:

```
<ol>
  <li>Item number 1</li>
  <li>And the second</li>
  <li value=9> Jump to number 9</li>
  <li>And continue with 10...</li>
</ol>
```

Результат действия этого кода можно видеть на рис. 7.5.

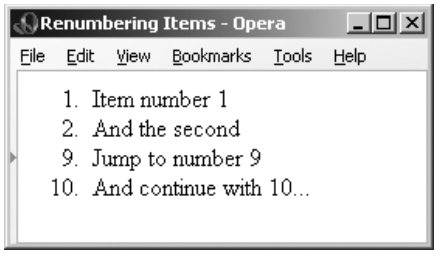

Рис. 7.5. *Атрибут value позволяет изменять номера отдельных элементов в упорядоченном списке*

7.3.1.3. Атрибуты class и style

Используйте атрибут `style` с тегом `<li>` для встроенного в документ определения стиля содержимого тега. Атрибут `class` позволяет применять к содержимому тега стиль, заранее определенный для данного класса элементов списков. [встроенные стили: атрибут style, 8.1.1] [стилевые классы, 8.3]

7.3.1.4. Атрибуты class, dir, event, id, lang, style и title

Эти атрибуты применимы также и к упорядоченным спискам и имеют тот же эффект, что и в случае неупорядоченных списков. [атрибуты class и style, 4.1.1.6] [атрибуты lang и dir, 6.3.1.7] [атрибуты id и title, 7.1.1.5] [атрибуты событий, 6.3.1.4]

7.4. Вложенные списки

Во все списки, кроме директорий и меню, можно вкладывать другие списки. Списки директорий и меню могут быть вложены в списки других типов. Отступы для каждого вложения списков накапливаются, так что позаботьтесь, чтобы глубина вложений не была слишком велика, – содержимое списков быстро превращается в узкую полоску вдоль правого края окна броузера.

7.4.1. Вложенные неупорядоченные списки

Элементы в каждом новом вложенном списке могут маркироваться по-своему, это зависит от броузера. К примеру, Internet Explorer выводит последовательность из маркеров в виде окружности, сплошного круга и квадратика для списков разных уровней вложенности, как это показывает нижеследующий фрагмент, отображенный на рис. 7.6:

```
<ul>
  <li>Morning Kumquat Delicacies
  <ul>
    <li>Hot Dishes
    <ul>
      <li>Kumquat omelet</li>
      <li>Kumquat waffles
      <ul>
        <li>Country style</li>
        <li>Belgian</li>
      </ul>
      </li>
    </ul>
    <li>Cold Dishes
    <ul>
      <li>Kumquats and cornflakes</li>
      <li>Pickled Kumquats</li>
      <li>Diced Kumquats</li>
    </ul>
```

```
    </li>
   </ul>
  </li>
 </ul>
```

Можно изменять стиль маркеров в каждом неупорядоченном списке и даже у отдельных элементов, но набор маркеров ограничен, как правило, закрашенным диском для пунктов первого уровня, незакрашенным кружком для второго уровня и закрашенным квадратиком для последующих уровней.

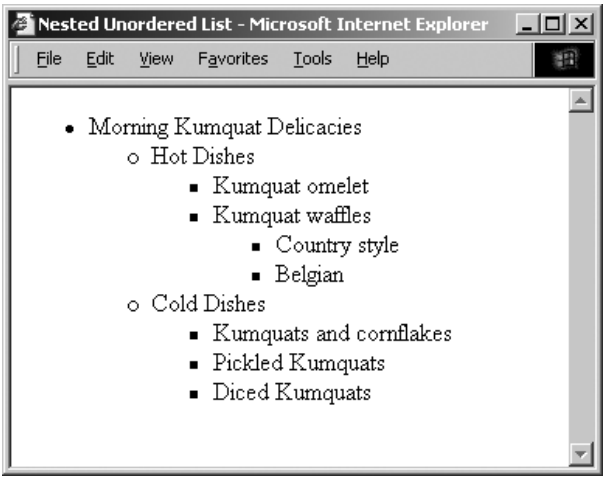

Рис. 7.6. *Маркеры во вложенных неупорядоченных списках разные*

7.4.2. Вложенные упорядоченные списки

По умолчанию броузеры нумеруют элементы упорядоченного списка, начиная с арабской цифры 1, независимо от того, вложен он или нет. Было бы здорово, если бы стандарты потребовали рационального последовательного способа нумерации вложенных упорядоченных списков. К примеру, элементы второго уровня в третьей части главного упорядоченного списка нумеровались бы последовательно: «3.2.1», «3.2.2», «3.2.3» и т. д.

Тем не менее использование атрибутов `type` и `value` предоставляет известную свободу в оформлении вложенных списков. Превосходным примером может служить традиционный способ вывода с применением различных способов нумерации, предлагаемых атрибутом `type` (рис. 7.7):

```
<ol type="A">
  <li>A History of Kumquats
  <ol type="1">
    <li>Early History
```

```
      <ol type="a">
        <li>The Fossil Record</li>
        <li>Kumquats: The Missing Link?</li>
      </ol>
    </li>
    <li>Mayan Use of Kumquats</li>
    <li>Kumquats in the New World</li>
   </ol>
  </li>
  <li>Future Use of Kumquats</li>
</ol>
```

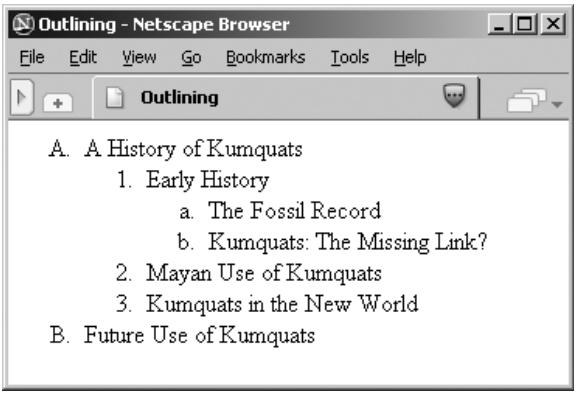

Рис. 7.7. *Атрибут type позволяет оформить упорядоченный список в традиционном стиле*

7.5. Списки определений

HTML и XHTML поддерживают также списки определений – конструкции, совершенно отличные от рассмотренных выше упорядоченных и неупорядоченных списков. Устроенные как статьи в толковом словаре или энциклопедии, заполненные текстом, картинками и другими мультимедийными элементами списки определений дают идеальный способ создания глоссариев, набора терминов или других списков с элементами вида имя/значение.

7.5.1. Тег <dl>

Список определений заключен между тегами `<dl>` и `</dl>`. Внутри этого тега каждый элемент списка состоит из двух частей – термина и следующего за ним определения или объяснения. Вместо тега `<li>` для выделения элементов используется тег `<dt>` (содержащий термин) и следующий за ним тег `<dd>` (содержащий определение или объяснение).

Если только вы не изменили характеристики отображения при помощи таблиц стилей, броузер будет выводить имя термина у левого края

<dl>	
Функция:	Устанавливает список определений
Атрибуты:	class, compact, dir, id, lang, onClick, onDblClick, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, style, title, type
Закрывающий тег:	</dl>; присутствует обязательно
Содержит:	*dl_content* (содержимое dl)
Может содержаться в:	*блоке*

демонстрационного окна, а его определение ниже и с отступом. Если описываемый термин очень короткий (обычно не длиннее трех символов), броузер может поместить первую часть определения на той же строке. На рис. 7.8 показано, как броузер отображает нижеследующий XHTML-список определений:

```
<h3>Common Kumquat Parasites</h3>
<dl>
  <dt>Leaf mites</dt>
  <dd>The leaf mite will ravage the Kumquat tree, stripping it
      of any and all vegetation.</dd>
  <dt>Trunk dropsy</dt>
  <dd>This microscopic larvae of the common opossum
      chigger will consume the structural elements of the
      tree trunk, causing it to collapse inward.</dd>
</dl>
```

Как и в списках других типов, можно увеличить расстояние между элементами, вставляя тег абзаца <p> или определяя для соответствующих тегов стили, задающие величину интервалов.

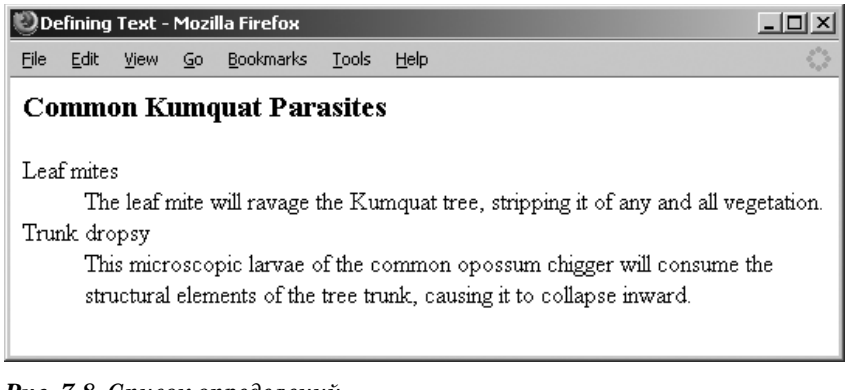

***Рис. 7.8.** Список определений*

7.5.1.1. Более компактные списки определений

Тег <dl> поддерживает атрибут compact, советующий броузеру представить список как можно компактнее. Немногие броузеры, если вообще такие есть, учтут этот совет. Данный атрибут нежелателен в HTML 4 и XHTML.

7.5.1.2. Атрибуты class, dir, event, id, lang, style и title

Многие атрибуты тега <dl> должны быть вам уже знакомы. Эти атрибуты дают возможность пометить (id) или назвать (title) содержимое тега, изменить характеристики его отображения (class, style), указать используемый язык (lang) и связанное с ним направление отображения текста (dir), кроме того, множество on-атрибутов позволяют реагировать на инициированные пользователем с помощью мыши и/или клавиатуры события, связанные с содержимым тега. Не все такие атрибуты реализованы в популярных в настоящее время броузерах для этого и многих других тегов. [атрибут dir, 3.6.1.1] [атрибут lang, 3.6.1.2] [встроенные стили: атрибут style, 8.1.1] [стилевые классы, 8.3] [атрибут id, 4.1.1.4] [атрибут title, 4.1.1.5] [обработчики событий JavaScript, 12.3.3]

7.5.2. Тег <dt>

Этот тег устанавливает компонент, соответствующий термину списка определений. Он имеет смысл, только если находится в списке перед тегом <dd>, содержащим определение или объяснение термина.

<dt>	
Функция:	Устанавливает термин в списке определений
Атрибуты:	class, dir, id, lang, onClick, onDblClick, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, style, title
Закрывающий тег:	</dt>; в HTML может опускаться
Содержит:	*текст*
Может содержаться в:	*dl_content* (содержимое dl)

Обычно определяемый термин, следующий за тегом <dt>, краток. Это слово или несколько слов. Формально он может иметь любую длину. Длинный термин броузер может продолжить за пределы окна или, разбив его на несколько частей, поместить окончание на ту же строку, где начинается определение.

Поскольку конец тега <dt> непосредственно предшествует началу соответствующего тега <dd>, пропуск </dt> не приведет к двусмысленности, и закрывающий тег не является обязательным в HTML-документах. XHTML, однако, настаивает на его присутствии, поэтому привыкайте включать его в свои документы.

7.5.2.1. Форматирование текста при помощи <dt>

На практике броузеры либо чересчур снисходительны, либо слишком молчаливы, чтобы настаивать на исполнении правил, так что некоторые хитроумные HTML-авторы злостно употребляют тег <dt> для перетаскивания левого поля вправо и влево при отображении текста. (Если вы помните, символы табуляции и ведущие пробелы не влияют на отображение обычного текста.) Мы не одобряем нарушение стандарта HTML, и уж тем более XHTML, и снова предостерегаем вас от уловок в документах. Используйте лучше таблицы стилей.

7.5.2.2. Атрибуты class, dir, event, id, lang, style и title

Тег <dt> поддерживает стандартный набор HTML 4/XHTML-атрибутов. Эти атрибуты дают возможность пометить (id) или назвать (title) содержимое тега, изменить характеристики отображения его содержимого (class, style), указать используемый язык (lang) и связанное с ним направление отображения текста (dir). Кроме того, множество on-атрибутов позволяют реагировать на инициированные пользователем с помощью мыши и/или клавиатуры события, связанные с содержимым тега. Не все такие атрибуты реализованы в популярных в настоящее время броузерах для этих и многих других тегов. [атрибут dir, 3.6.1.1] [атрибут lang, 3.6.1.2] [стилевые классы, 8.3] [встроенные стили: атрибут class, 8.1.1] [атрибут id, 4.1.1.4] [атрибут title, 4.1.1.5] [обработчики событий JavaScript, 12.3.3]

7.5.3. Тег <dd>

Тег <dd> отмечает начало описания в элементе списка определений. В соответствии со стандартами HTML 4 и XHTML тег <dd> может встречаться только внутри списка определений, непосредственно следуя за тегом <dt> и термином. Вслед за тегом <dd> идет определение или объяснение термина.

За тегом <dd> могут следовать любые HTML/XHTML-конструкции, включая другие списки, блоки текста и мультимедийные элементы.

<dd>

Функция:	Указывает на описание значения термина в списке определений
Атрибуты:	class, dir, id, lang, onClick, onDblClick, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, style, title
Закрывающий тег:	</dd>; всегда опускается в HTML
Содержит:	*flow* (поток текста)
Может содержаться в:	*dl_content* (содержимое dl)

Рассматривая все эти компоненты как обыкновенное содержимое тега, броузеры обычно отображают их с отступом. И поскольку начало другого термина, назначенного указанием (<dt>), или обязательный закрывающий тег списка </dl> недвусмысленно прекращают предшествующее определение, в теге </dd> нет необходимости, и его отсутствие делает исходный текст более пригодным для чтения. Итак, опять и снова: XHTML настаивает на присутствии закрывающих тегов в ваших документах, так что лучше привыкнуть вставлять в них </dd>.

7.5.3.1. Атрибуты class, dir, event, id, lang, style и title

Тег <dd> поддерживает стандартный набор атрибутов. Эти атрибуты дают возможность пометить (id) или назвать (title) содержимое тега, изменить характеристики отображения его содержимого (class, style), указать используемый язык (lang) и связанное с ним направление отображения текста (dir). Кроме того, множество on-атрибутов позволяют реагировать на инициированные пользователем с помощью мыши и/или клавиатуры события, связанные с содержимым тега. Не все эти атрибуты реализованы в популярных в настоящее время броузерах для данного и многих других тегов. [атрибут dir, 3.6.1.1] [атрибут lang, 3.6.1.2] [встроенные стили: атрибут style, 8.1.1] [стилевые классы, 8.3] [атрибут id, 4.1.1.4] [атрибут title, 4.1.1.5] [обработчики событий JavaScript, 12.3.3]

7.6. Как использовать списки

Используйте неупорядоченные списки для:

- Собраний гиперссылок
- Коротких, не связанных иерархически фрагментов текста
- Выделения ключевых моментов презентации

Используйте упорядоченные списки для:

- Оглавлений
- Перечня предписаний
- Последовательных разделов текста
- Присоединения номеров к коротким фразам, на которые, возможно, придется где-то сослаться

Используйте списки определений для:

- Словарей
- Создания маркеров элементов списка (поместите в тег <dt> маленькую пиктограмму)
- Любых списков вида имя/значение

7.7. Директории

Директория – это специализированная форма неупорядоченного списка, не одобренная стандартами HTML 4 и XHTML. Мы не рекомендуем употребление этого тега. [тег <ul>, 7.1.1]

7.7.1. Тег <dir> (нежелателен)

Создатели HTML предназначали тег <dir> для отображения списков файлов. Раз так, броузер, если он обращается с <dir> и <ul> по-разному (а большинство их не различает), ожидает, что элементы списка будут короткими, возможно, не длиннее 20 символов. Некоторые броузеры отображают элементы такого списка в несколько колонок и могут не использовать при этом маркеры.

<dir>

Функция:	Определяет директорию
Атрибуты:	class, dir, id, lang, onClick, onDblClick, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, style, title
Закрывающий тег:	</dir>; присутствует обязательно
Содержит:	*list_content* (содержимое списка)
Может содержаться в:	*блоке*

Как и в случае неупорядоченного списка, элементы директории определяются тегом <li>. Однако применяемый в директории тег <li> не может содержать блочных элементов, таких как абзацы, другие списки, преформатированный текст, формы.

Следующий пример использует тег <dir> в его прямом назначении – для представления списка имен файлов:

```
The distribution tape has the following files on it:
<dir>
  <li><code>README</code></li>
  <li><code>Makefile</code></li>
  <li><code>main.c</code></li>
  <li><code>config.h</code></li>
  <li><code>util.c</code></li>
</dir>
```

Заметьте, что мы употребили тег <code>, чтобы гарантировать вывод имен файлов в подходящем виде (рис. 7.9).

7.7.1.1. Атрибуты тега <dir>

Атрибуты тега <dir> идентичны тегу <ul> и имеют то же действие.

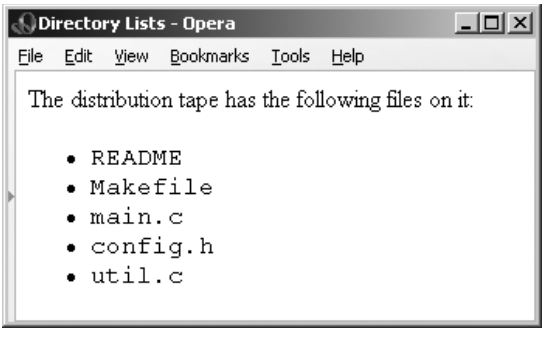

***Рис. 7.9.** Пример списка <dir>*

7.8. Меню

Меню – это еще одна специализированная форма неупорядоченного списка. Подобно `<dir>`, тег `<menu>` нежелателен в стандартах HTML 4 и XHTML, так что мы не рекомендуем его применение. [тег <ul>, 7.1.1]

7.8.1. Тег <menu> (нежелателен)

Тег `<menu>` отображает для читателя список, состоящий из коротких альтернатив, такой как меню гиперссылок на другие документы.

<menu>

Функция:	Определяет меню
Атрибуты:	`class`, `dir`, `id`, `lang`, `onClick`, `onDblClick`, `onKeyDown`, `onKeyPress`, `onKeyUp`, `onMouseDown`, `onMouseMove`, `onMouseOut`, `onMouseOver`, `onMouseUp`, `style`, `title`
Закрывающий тег:	`</menu>`; присутствует обязательно
Содержит:	*list_content* (содержимое списка)
Может содержаться в:	*блоке*

Броузер может использовать особый (обычно более компактный) способ представления его элементов в сравнении с обыкновенным неупорядоченным списком или даже выводить содержимое этого тега в виде всплывающего меню. Если элементы списка достаточно коротки, броузер может отобразить их в несколько колонок и не применять маркеров.

Как и в случае неупорядоченного списка, элементы меню определяются тегом `<li>`. Однако используемый в меню тег `<li>` не может содержать блочных элементов, таких как абзацы, другие списки, преформатированный текст, формы.

Сравните исходный текст меню и его отображение (рис. 7.10) с директорией (рис. 7.9) и неупорядоченным списком, приведенным ранее в этой главе (рис. 7.1):

```
Some popular kumquat recipes include:
<menu>
  <li>Pickled Kumquats</li>
  <li>'Quats and 'Kraut (a holiday favorite!)</li>
  <li>'Quatshakes</li>
</menu>
There are many more to please every palate!
```

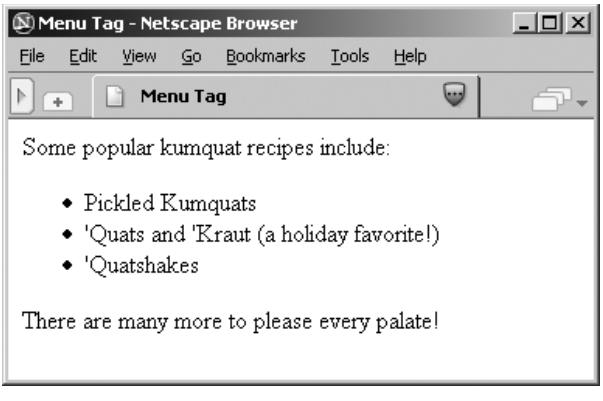

Рис. 7.10. *Образец меню*

- Элементы стилей
- Синтаксис стилей
- Стилевые классы
- Стилевые свойства
- Бестеговые стили – тег
- Применение стилей в документах

8

Каскадные таблицы стилей

С помощью таблиц стилей профессионалы в издательствах управляют общим «внешним видом» своих публикаций – фоном, шрифтами, цветами и прочими элементами оформления – от буклетов до огромных коллекций документов. Большинство настольных издательских систем поддерживают таблицы стилей так же, как это делают и популярные текстовые редакторы, так что применение таблиц стилей в HTML-документах очевидно.

Язык HTML всегда был ориентирован на содержание, а не на подачу материала. Авторам предлагалось заботиться о предоставлении информации высокого качества, а вопросы отображения материала оставить броузеру. Мы настоятельно рекомендуем принять такую философию и воплощать ее в ваших документах – не путайте стиль и содержание.

Тем не менее не следует забывать, что и подача материала служит нуждам читателя. Даже первые создатели HTML понимали взаимозависимость между стилем и удобством чтения – например, между физическим стилем и тегами заголовков. Таблицы стилей расширяют это представление всякими дополнительными эффектами, включая цвета, более богатый выбор шрифтов, даже звуковое сопровождение, чтобы пользователи еще лучше могли различать элементы документа. Но важнее всего то, что таблицы стилей позволяют управлять характеристиками отображения для всех тегов – в одном документе или в целом собрании – и делать это по единому образцу.

В начале 1996 года консорциум World Wide Web Consortium (W3C) собрал проектные разработки, определяющие каскадные таблицы стилей (CSS) для HTML. Этот проект быстро созрел до рекомендованного стандарта. В середине 1998 года W3C расширил изначальные спецификации и создал CSS2, который включил в себя стандарты представ-

ления для множества средств вывода информации, отличных от хорошо всем знакомых экранных броузеров, наряду с другими усовершенствованиями.

Консорциум W3C продолжает работать над младшей версией (2.1) и черновым проектом CSS3, но они пока не актуальны. Вообще, никакой современный броузер или другой веб-клиент сети не отвечает вполне стандарту CSS2. Поскольку и так ясно, что согласие с детищем, предложенным W3C, будет в конце концов достигнуто, мы в этой главе затронем все компоненты стандарта CSS2. Как всегда, будет точно указано, что уже реально существует, что только предлагается и что в действительности поддерживается.[1]

8.1. Элементы стилей

На самом простом уровне стиль – это не что иное, как правило, указывающее броузеру, как выводить содержимое какого-то определенного HTML- или XHTML-тега.[2] У каждого тега есть ряд ассоциированных с ним стилевых свойств, значения которых определяют, как этот тег воспроизводится броузером. Правило приписывает определенное значение одному или нескольким свойствам тега. Например, большинство тегов имеют свойство `color`, значение которого определяет цвет, который современные GUI-броузеры должны использовать при отображении содержимого тега. К числу других свойств относятся шрифт, интервал между строками, поля, рамки, громкость звука и тембр, которые будут детально рассмотрены ниже в этой главе.

Существует три способа присоединения стиля к тегу: встроенные в теги стили, стили документа и внешние таблицы стилей. Вы можете использовать одну или несколько таблиц для документа. Броузер либо объединяет определения каждого из стилей, либо переопределяет характеристики стиля для содержимого тега. Заимствованные из этих различных источников, они применяются к документу, сочетаясь и определяя свойства результирующего стиля, которые, подобно каскаду водопадов, ступенчато ниспадают, начиная с внешних таблиц, продолжая на уровне документа и оканчивая свой путь во встроенных стилях. Этот каскад свойств и стилевых правил дал имя стандарту каскадных таблиц стилей.

---

[1] Осенью 2000 г. началась работа над стандартом CSS3. Поскольку он находится еще только на стадии проектирования, а броузеры не полностью соответствуют даже стандарту CSS2, мы в этой главе сосредоточиваемся на CSS2.

[2] Мы намеренно избегаем здесь термина *отображает*, поскольку он ассоциируется с визуальным представлением, тогда как стандарт CSS2 предлагает много различных способов вывода для размеченного с помощью тегов содержимого документа.

Здесь мы обсудим основы синтаксиса каждого из трех типов таблиц стилей. А в конце главы остановимся на вопросах уместности употребления встроенных, документных и внешних таблиц стилей.

8.1.1. Встроенные стили. Атрибут style

Встроенный стиль – это простейший способ применения стиля к тегу. Просто включите в тег атрибут `style` со списком свойств и их значений. Броузер использует их при выводе содержимого в соответствии с требованиями тега.

К примеру, следующий стиль предлагает отобразить текст заголовка уровня 1, «Я такой сииниииий!», применяя не только свойственные броузеру стилевые характеристики тега `<h1>`, но также синий цвет и курсив:

```
<h1 style="color: blue; font-style: italic"> Я такой сииниииий!</h1>
```

Встроенные стили трудно поддерживать, поскольку они придают дополнительный смысл определениям тегов, затрудняя их восприятие. Кроме того, они имеют лишь локальный эффект и поэтому должны быть разбросаны по всему документу. Используйте атрибут `style` скупо и только в тех редких случаях, когда соответствующего эффекта нельзя достичь другим способом.

8.1.2. Таблицы стилей на уровне документа

Действительная сила таблиц стилей становится более очевидной, когда вы помещаете список правил представления в начало HTML- или XHTML-документа. Расположенные внутри тега `<head>` и заключенные в собственный тег `<style>` (с закрывающим `</style>`) *таблицы стилей на «уровне документа»* действуют на все вхождения тегов в документ, за исключением тех, что содержат обладающие более высоким приоритетом встроенные определения с атрибутом `style`.[1]

Все, что находится между тегами `<style>` и `</style>`, рассматривается как часть стилевых правил, которые должны быть применены к документу. На самом деле содержимое тега `<style>` не относится к HTML или XHTML и не подчиняется обычным правилам размеченного содер-

<style>

Функция:	Определяет таблицу стилей на уровне документа
Атрибуты:	`dir`, `lang`, `media`, `title`, `type`
Закрывающий тег:	`</style>`; часто опускается в HTML
Содержит:	*стили*
Может содержаться в:	*head_content* (содержимое заголовка)

[1] XHTML-таблицы стилей на уровне документа заключены в специальные CDATA-разделы документов. См. подробнее в главе 16 «XHTML».

жимого. Тег <style>, таким образом, позволяет вставить в документ чужеродную информацию, которую броузер использует для форматирования тегов.

К примеру, броузеры, понимающие таблицы стилей, отобразят синим курсивом содержимое всех тегов заголовков в HTML-документе, имеющем в своем заголовке следующее определение таблиц на уровне документа:[1]

```
<head>
<title>Все синее</title>
<style type="text/css">
  <!--
  /* делаем все заголовки 1-го уровня синими и курсивом */
  h1 {color: blue; font-style: italic}
  -->
</style>
</head>
<body>
<h1> Я такой сииниииий!</h1>
...
<h1>И я тожеее сииниииий!</h1>
```

8.1.2.1. Атрибут type

Кроме CSS, для HTML/XHTML доступны и другие таблицы стилей. Подобно стилевым таблицам JavaScript, описываемым в главе 12, они плохо поддерживаются (или вовсе не поддерживаются) современными броузерами, так что мы не станем уделять им много внимания в этой книге. Как бы то ни было, броузеру нужен способ определения таблицы стилей, используемой в вашем документе. С этой целью ставьте атрибут type в тег <style>. Все каскадные таблицы стилей имеют тип text/css. Таблицы стилей JavaScript применяют тип text/javascript. Можно опустить данный атрибут и надеяться, что броузер сам поймет, какой тип используется. Мы предпочитаем всегда включать атрибут type, избегая возможных недоразумений. [таблицы стилей JavaScript, 12.4]

8.1.2.2. Атрибут media

В наши дни HTML- и XHTML-документы можно встретить в самых неожиданных местах, даже в сотовых телефонах. Чтобы помочь броузеру выбрать лучший способ вывода документа, включите в тег <style> атрибут media. Его значением является средство обмена информацией, для которого этот документ предназначен, хотя это и не препятствует выводу документа другим устройством. Значением атрибута по умолчанию является screen (дисплей компьютера). Другие определенные в стандарте значения – это tty (телетайпы, терминалы или портатив-

[1] Это пример HTML. В XHTML-документе стили должны быть заключены в раздел CDATA, а не в комментарии HTML. См. раздел 16.3.7.

ные устройства с ограниченными возможностями дисплея), tv (устройства типа телевизора с низким разрешением и плохой цветопередачей), projection (проекционные аппараты), handheld (портативные компьютеры и сотовые телефоны), print (принтеры, печатающие чернилами на бумаге), braille (тактильные устройства), embossed (устройства для вывода текста в коде Брайля, для слепых), aural (аудиоустройства или, например, синтезаторы речи) и all (многие другие типы устройств).

Если вы хотите вместо all явно перечислить несколько типов устройств, вы можете использовать в качестве значения атрибута media заключенный в кавычки список, элементы которого разделяются запятыми. Например:

```
<style type="text/css" media="screen,print">
```

говорит броузеру, что документ содержит таблицу стилей и предназначен для вывода на принтер или на дисплей компьютера.

Будьте внимательны при указании устройств, поскольку броузер не сможет применить определяемые вами стили, если документ выводится не на одном из перечисленных устройств. Таким образом, броузер не будет применять разработанные для media="screen,print" стили из нашего примера, если пользователь соединяется с сетью с помощью, скажем, карманного компьютера.

Как обеспечить определение разных стилей для разных устройств, не создавая нескольких копий документа? Стандарт CSS2 позволяет определять стили, специфичные для того или иного устройства вывода, через свои специальные директивы (так называемые правила at) @import и @media. Подробнее об этом см. в разделе 8.1.5.

8.1.2.3. Атрибуты dir, lang и title

Как и со многими другими HTML/XHTML-элементами, можно связать с тегом <!-- <DEFANGED_STYLE> описательное название, а также указать язык и направление вывода текста с помощью атрибутов title, lang и dir соответственно. [атрибут dir, 3.6.1.1] [атрибут lang, 3.6.1.2] [атрибут title, 4.1.1.4]

8.1.3. Броузеры без поддержки стилей

Конечно, вы заметили, что в предыдущем примере, где стиль определялся на уровне документа, мы поместили содержимое тега <style> внутрь тега комментария <!--. Старые версии броузеров, не поддерживающие стили, игнорируют тег <style>, но с энтузиазмом выводят на экран его содержимое. Современные броузеры ожидают, что в HTML-комментарии перечислены стилевые правила, и обрабатывают их соответствующим образом, в то время как старые броузеры игнорируют неизвестный им тег, а к тегу комментария относятся корректно. Это работает.

Порядок тегов очень важен. В нашем примере принят следующий подход:

```
<style>
<!--
  h1 {color: blue; font-style: italic}
-->
</style>
```

Ставьте тег <style>, за ним HTML-комментарий, а за комментарием – стилевые правила на уровне документа. После этого закройте комментарий и поставьте тег </style> – именно в таком порядке.

С XHTML-документами нужно обращаться чуть иначе. В них мы помещаем стили, относящиеся ко всему документу, в секцию CDATA, а не в HTML-тег комментария. Подробности см. в разд. 16.3.7.

Со встроенными стилевыми атрибутами и их значениями броузеры, не поддерживающие стили, поступают так же, как и с остальными неизвестными им атрибутами: они их игнорируют. Так что искажения ваших документов опасаться не стоит.

8.1.4. Внешние таблицы стилей

Кроме того, можно помещать определения стилей в отдельный документ (текстовый файл, MIME-типом которого является text/css) и импортировать такие *«внешние» таблицы стилей* в документы. Чтобы придать своим документам унифицированный вид, пользуйтесь единой таблицей стилей для других документов в этом собрании и даже для других собраний документов. Поскольку внешняя таблица является отдельным файлом и загружается броузером из сети, ее можно хранить где угодно, использовать многократно и даже применять чужие таблицы стилей.

Например, допустим, что мы создали файл gen_styles.css, включающий следующее стилевое правило:

```
h1 {color: blue; font-style: italic}
```

Можно предложить броузеру прочитать файл gen_styles.css для каждого документа из нашей подборки, а он в ответ покрасит содержимое каждого тега <h1> в синий цвет и выведет текст курсивом. Конечно, это произойдет только при условии, что машина пользователя способна совершать такие фокусы со стилем, на ней установлен работающий с таблицами стилей броузер и назначения стиля не отменены на уровне документа или встроенным определением.

Загрузить в документ внешнюю таблицу стилей можно двумя способами: при помощи тега <link> или директивы @import.

8.1.4.1. Присоединение внешних таблиц стилей при помощи тега <link>

Один способ загрузить внешнюю таблицу стилей – использовать тег <link> внутри тега <head> в вашем документе:

```
<head>
<title>Присоединение стиля с помощью link</title>
<link rel=stylesheet type="text/css"
      href="http://www.kumquats.com/styles/gen_styles.css"
      title="Синева">
</head>
<body>
<h1>Я такой сиииниий!</h1>
...
<h1>И я тожеее сиииниий!</h1>
```

Напомним, что тег `<link>` связывает документ, в котором он содержится, с каким-то другим документом в сети. В примере мы сообщаем броузеру, что документ, поименованный в атрибуте `href`, имеет тип `stylesheet`, как это указано в атрибуте `type`. Ссылка на внешнюю таблицу стилей в теге `<link>` требует указания атрибутов `href` и `type`. Кроме того, мы явно, причем добровольно, сообщаем броузеру, что отношение файла к нашему документу определяется значением `stylesheet` (таблица стилей). Мы также добавили атрибут `title`, определяющий название таблицы стилей, создав возможность для ссылок на нее в будущем. [элемент заголовка <link>, 6.7.2]

Тег `<link>` и его обязательные атрибуты `href` и `type` должны появляться только в заголовке документа. URL таблицы стилей может быть абсолютным или относительным, отсчитываемым от базового URL документа.

8.1.4.2. Импортированные внешние таблицы стилей

Второй способ загрузки внешней таблицы стилей импортирует файлы, применяя специальную директиву (так называемое *правило at*) в теге `<style>`:

```
<head>
<title>Импортированный стиль</title>
<style type="text/css">
  <!--
    @import url(http://www.kumquats.com/styles/gen_styles.css);
    @import "http://www.kumquats.com/styles/spec_styles.css";
    body {background: url(backgrounds/marble.gif)}
  -->
</style>
</head>
```

Директива `@import` ожидает параметр в виде URL, указывающего путь в сети к внешней таблице стилей. Как показано в этом примере, URL может быть заключенной в двойные кавычки строкой, заканчивающейся точкой с запятой, или следовать за ключевым словом `url`, так же оканчиваясь точкой с запятой, но помещаясь при этом в скобки. URL может быть абсолютным или относительным, отсчитываемым от базового URL документа.

Команда `@import` должна появляться *до* любых традиционных стилевых правил, будь то в теге `<style>` или во внешней таблице стилей. В противном случае стандарт настаивает, чтобы броузер игнорировал ошибочную `@import`. Согласно стандарту CSS2 каскадирование происходит так: сначала импортируются различные таблицы стилей, затем обрабатываются правила, определенные на уровне документа, в результате выигрывает последний оставшийся стиль. [URL как значение свойств, 8.4.1.4]

Директива `@import` может появиться в определениях стиля на уровне документа или даже в другой таблице стилей, что позволяет создавать вложенные таблицы стилей.

8.1.5. Стили для разных устройств вывода

Помимо атрибута `media` для тега `<style>` стандарт CSS2 содержит еще два средства, которые позволяют применять различные таблицы стилей в зависимости от того, какой агент или устройство будут выводить документ. С помощью этих способов можно применять один или целую таблицу стилей, когда документ отображается на экране компьютера, и другую группу стилей, когда содержимое выводится на принтере, печатающем по системе Брайля. А как же обходиться с сотовыми телефонами в сети?

Подобно тому как атрибут `media` для тега `<style>` сообщает, в каких случаях нужно применять описанную в теге таблицу стилей, существует возможность указать обработчику документов[1], следует ли ему загружать и использовать внешнюю таблицу в зависимости от того, какое устройство будет доносить до человека содержание документа. Это можно сделать, добавив в конец команды `@import` один или несколько отделенных друг от друга запятыми типов посредников.

Следующий пример предлагает обработчику документов решить, какую из таблиц стилей загрузить и применить – предназначенную для синтезатора речи или ту, что задает стили при выводе на экран компьютера и на печать:

```
@import url(http://www.kumquats.com/styles/visual_styles.css) screen,print;
@import "http://www.kumquats.com/styles/speech_styles.css" aural;
```

В CSS2 включены следующие типы устройств вывода: `all`, `aural`, `braille`, `embossed`, `handheld`, `print`, `projection`, `screen`, `tty` и `tv`.

Другой предусмотренный CSS2 способ учесть устройство вывода состоит в применении явной директивы `@media`, которая позволяет включать в одну таблицу стилей правила, относящиеся только к определенным посредникам. Это можно делать как на уровне документа, так

[1] В наши дни веб-документы выводятся на все виды устройств, включая популярные броузеры, принтеры, печатающие по Брайлю, телевизоры, проекторы и т. д.

и во внешней таблице стилей. На уровне документа @media должна, как и директива @import, содержаться в теге <style>. При этом обе они не могут входить в иные правила. В отличие от @import, директиве @media разрешено следовать за другими стилевыми назначениями, и те, которые описаны в ней, вправе отменять определенные прежде в соответствии с каскадным стандартом.

Содержимое @media включает в себя одно или несколько разделенных запятыми наименований типов посредников, за которыми следуют фигурные скобки ({}), охватывающие набор стилевых правил. Например:

```
body {background: white}
@media tv, projection  {
     body {background: yellow}
     }
```

Атрибут yellow у специальной директивы @media окрашивает фон тела документа в желтый цвет (а не в белый, указанный в качестве умолчания в общем стилевом правиле), когда документ отображается на телевизионном экране или проекционной системой (это определяют атрибуты tv и projection).

8.1.6. <link> или @import?

На первый взгляд может показаться, что способы присоединения таблиц стилей, использующие тег <link> и директиву @import, эквивалентны и применяют различный синтаксис для достижения одной и той же цели. Это так, если в документе только один тег <link>. Когда их больше, в игру вступают специальные правила CSS2.

Когда в документе содержится один тег <link>, броузер должен загрузить стили из таблицы, на которую ссылается тег, и в соответствии с ними провести форматирование, учитывая, что при конфликте установки на уровне документа и встроенные стили отменяют внешние определения. Если в документе содержатся два и более тегов <link>, броузер должен представить пользователю список таблиц стилей, на которые ссылаются теги. Пользователь выбирает таблицу, которую броузер загружает и применяет при форматировании документа. Остальные таблицы из тегов <link> при этом игнорируются.

Если речь идет об @import, то, напротив, всякий распознающий стили броузер должен слить множество указанных в директивах таблиц в один набор стилевых правил для документа. При возникновении каких-либо взаимных противоречий приоритет имеет последняя импортированная таблица. Следовательно, если внешняя таблица стилей gen_styles.css предлагает броузеру отобразить содержимое тега <h1> синим курсивом, а spec_styles.css настаивает на красном цвете текста того же тега, то содержимое будет отображено красным курсивом. И если впоследствии мы определим для тега <h1> другой цвет, скажем желтый, на уровне документа, тег <h1> будет желтым и наклонным. Каскадный эффект. Понятно?

На практике популярные броузеры обращаются с таблицами стилей, присоединенными с помощью тега <link>, так же, как с импортированными таблицами, каскадным образом совмещая их действие. Броузеры в настоящее время не позволяют выбирать таблицу стилей. Импортированные стили замещают стили, присоединенные тегом <link>, точно так же, как определенные на уровне документа и встроенные в тег стили замещают внешние определения. Собирая все это вместе, рассмотрим пример:

```
<html>
<head>
<link rel=stylesheet href=sheet1.css type=text/css>
<link rel=stylesheet href=sheet2.css type=text/css>
<style>
<!--
  @import url(sheet3.css);
  @import url(sheet4.css);
-->
</style>
</head>
```

В соответствии с моделью CSS2 броузер должен предложить пользователю выбирать между sheet1.css и sheet2.css. Затем он должен загрузить выбранную таблицу, за которой последуют sheet3.css и sheet4.css. Стили, определенные в таблицах sheet3.css и sheet4.css, так же как и встроенные стили, будут в случае конфликта применяться вместо стилей выбранной таблицы. На практике популярные броузеры каскадным образом объединят правила таблиц из примера, обрабатывая их в том порядке, в каком они появляются, от sheet1 до sheet4.

8.1.7. Ограничения, связанные с современными броузерами

Все популярные броузеры используют тег <link> для присоединения внешней таблицы стилей к документу. Ни один не поддерживает множественные и выбираемые пользователем таблицы стилей, соответствующие стандарту CSS2. Вместо этого они каскадно объединяют указанные в тегах <link> таблицы, так что правила каждой следующей таблицы сильнее правил предыдущей.

Netscape шестой версии (но не более ранних), Internet Explorer версии 5 и более поздних, а также все версии Opera и Firefox признают и @import, и @media как на уровне документа, так и во внешних таблицах, позволяя вкладывать таблицы друг в друга.

Можно и не пытаться с помощью Netscape Navigator ранних версий получить стилевую дифференциацию по различным средствам вывода во внешних таблицах. Предполагайте, следовательно, что большая часть людей, имеющих броузеры Netscape версии 4, отображает документы на обычном экране PC, так что сделайте это устройство прини-

маемым по умолчанию. Затем вложите все стили, ориентированные на тактильный или печатный способ представления, при помощи директив `@media`, чтобы поддерживающие CSS обработчики смогли выбирать их, основываясь на устройстве вывода. Альтернативой является включение всех нужных стилей в тегах `<style>` во всех документах. И не думайте приниматься за это!

8.1.8. Комментарии к стилям

Комментарии в теге `<style>` только приветствуются, как и во внешних таблицах стилей, но вы должны обращаться с ними не так, как с комментариями языка HTML. Таблицы стилей – это не HTML. Лучше выделяйте комментарии, открывая их последовательностью символов `/*` и закрывая `*/`, как это сделали мы в предыдущем примере в разделе 8.1.2. (Те из вас, кто знаком с языком программирования C, узнают такие обозначения.) Используйте эти правила оформления комментариев как на уровне документа, так и во внешних таблицах стилей. Комментарии нельзя вкладывать друг в друга.

Мы рекомендуем документировать стили всегда, когда это возможно, особенно во внешних таблицах. Всякий раз, когда существует вероятность того, что ваши стили могут быть использованы другими авторами, комментарии позволят гораздо легче в них разобраться.

8.1.9. Приоритеты стилей

Можно импортировать несколько внешних таблиц и комбинировать их с определениями стилей на уровне документа и тега разными способами. Их эффекты складываются. Можно, например, задать тип шрифта для тега `<h1>` из нашего примера во внешней таблице, в то время как для его отображения будет использоваться стиль из определений на уровне документа.

Эффекты таблиц стилей не накапливаются, однако из ряда стилей, которые, возможно, определяют различные значения для одного и того же свойства (цвет содержимого тега `<h1>`, например), всегда существует один, обладающий приоритетом. Его можно найти, следуя правилам, перечисленным ниже, и действуя по порядку.

Ранжируем по источнику

Стиль, определенный «ближе» к тегу, имеет приоритет над установленными с «большего расстояния». Так, встроенные в тег стили обладают большим приоритетом, чем определенные на уровне документа, которые, в свою очередь, приоритетнее действия внешних таблиц.

Если можно применить несколько стилей, ранжируем их по классу

Свойства, заданные в классе тегов (раздел 8.3), имеют приоритет над свойствами, определенными для тега вообще.

Если все еще осталось несколько стилей, ранжируем их по специфичности

Свойства, определенные для более специфического окружения (раздел 8.2.3), имеют приоритет перед свойствами, введенными для менее специфического контекста.

Если стилей все еще несколько, ранжируем их по порядку вхождения

Последнее описание преобладает над всеми предыдущими.

Результат взаимодействия между свойствами стилей и традиционными атрибутами тегов почти невозможно предсказать. Диктуемые таблицами стилей цвета фона и переднего плана – определены ли они внешним образом, на уровне документа или встроены в тег – имеют предпочтение перед различными атрибутами цвета, которые могут встретиться в тегах. Атрибут `align` для встроенных в документ изображений, напротив, обычно оказывается сильнее выравнивания, предписанного стилями.

Существуют мириады комбинаций стилей и атрибутов. Нужно владеть магическим кристаллом, чтобы предсказать, какая из них выиграет, а какая проиграет в битве приоритетов. Правила, регулирующие множественные определения и взаимоотношения стилей и атрибутов, недостаточно ясно представлены в принадлежащем W3C стандарте CSS2, нет и четкого описания правил предпочтения, реализованных в броузерах, поддерживающих стили. Это особенно неприятно, потому что впереди нас ждет продолжительный период, возможно в несколько лет, когда пользователи будут работать как с поддерживающими, так и не поддерживающими стили броузерами. Чтобы добиться одного и того же эффекта, авторам придется одновременно применять средства управления, основанные и на стилях, и на атрибутах тегов.

Тем не менее мы советуем просто спасаться бегством от одноразовых, встроенных в документ, локализованных эффектов, достигаемых, скажем, тегом `<font>` или атрибутом `color`. Они сделали свое дело, их время ушло. Наступила пора вернуть (безболезненно!) последовательность и систематичность в представление документов. Используйте стили. Это стезя HTML.

8.2. Синтаксис стилей

Синтаксис стилей, как можно было заметить из предыдущих примеров, очень прямолинеен.

8.2.1. Основы

Стилевое правило состоит, по меньшей мере, из двух основных частей: *селектора*, так называется элемент разметки, который подчиняется этому правилу, следующих за ним двух фигурных скобок (`{}`) и заключенного в этих скобках списка пар вида `свойство:значение` (`property:value`), элементы которого разделяются точками с запятой:

```
selector {property1:value1; property2:value1; ...}
```

В частности, мы могли определить цвет содержимого всех заголовков уровня 1 в нашем документе, написав:

```
h1 {color:green}
```

В этом примере `h1` – это селектор и в то же время название элемента «заголовок уровня 1», `color` – свойство стиля, а `green` – значение. Просто и ясно.

Свойства требуют указания хотя бы одного значения, но могут иметь и несколько. Значения, разделенные запятыми, обычно свидетельствуют о том, что свойство принимает ряд значений, из которых берется первое допустимое. Если значения разделены пробелами, то к свойству применяется каждое. Последнее допустимое значение переопределяет предыдущее:

```
selector {property3:value1 value2 value3}
selector {property4:value1, value2, value3}
```

Например, в следующем случае фон будет черным, а не белым или серым, даже если вы укажете белый и черный цвета в стилевом правиле:

```
body {background: white black}
```

Современные поддерживающие стили броузеры игнорируют регистр букв в любом элементе стилевого правила. Так что `H1` и `h1` – это один и тот же селектор, а `COLOR`, `color`, `CoLOR` и `cOLor` – эквивалентные свойства. Когда-то требовалось, чтобы HTML-авторы записывали имена селекторов заглавными буквами, например `H1`, `P` и `STRONG`. Это и сейчас принято делать и так пишется в принадлежащем W3C CSS2-документе.

Однако действующие стандарты настаивают, особенно для документов, соответствующих XML, чтобы названия элементов записывались с применением тех же регистров, что использовались в их DTD. Например, в XHTML названия элементов (скажем, `h1`, `strong`) вводятся строчными буквами, так что их CSS2-селекторы тоже должны записываться строчными буквами. Мы придерживаемся этого последнего соглашения.

Любое допустимое название элемента (имя тега за вычетом атрибутов и заключающих скобок < и >) может быть селектором. В список селекторов можно включать более чем одно имя тега, как будет показано в следующих разделах.

8.2.2. Множественные селекторы

Ко всем перечисленным через запятую в селекторном списке элементам применяются значения свойств, определенных в стилевом правиле. Это облегчает авторам жизнь. К примеру:

```
h1, h2, h3, h4, h5, h6 {text-align: center}
```

делает в точности то же, что и:

```
h1 {text-align: center}
h2 {text-align: center}
h3 {text-align: center}
h4 {text-align: center}
h5 {text-align: center}
h6 {text-align: center}
```

Оба стилевых определения предлагают броузеру центрировать содержимое заголовков уровней 1–6. Для большинства авторов первый вариант легче набрать, понять и изменить. И он требует меньше времени и затрат других ресурсов при передаче по сети, хотя результат тривиален. Определяйте стили так, как вам удобнее. Вы не обязаны использовать множественные селекторы.

8.2.3. Контекстные селекторы

Обычно поддерживающие стили броузеры применяют определенные на уровне документа или встроенные стили к тегам везде, где они обнаруживаются в тексте документа, безотносительно к контексту. Однако стандарт CSS2 определяет способ применения стиля только к тегам, находящимся в определенном контексте, например содержащимся внутри другого тега.

Чтобы создать контекстный селектор, перечислите теги в том порядке, в котором они должны вкладываться в документ, начиная с самого внешнего. Когда броузер встретит такой порядок вложения, стилевое правило будет применено к последнему тегу списка.

Вот, например, как можно использовать контекстные стили для определения классической последовательности нумерации в упорядоченных списках: заглавные латинские буквы для самого внешнего уровня, римские цифры в верхнем регистре для следующего, затем арабские цифры и строчные латинские буквы для самого внутреннего уровня:

```
ol li {list-style: upper-roman}
ol ol li {list-style: upper-alpha}
ol ol ol li {list-style: decimal}
ol ol ol ol li {list-style: lower-alpha}
```

Следуя приведенной таблице стилей, поддерживающий стили броузер, встречая тег `<li>`, вложенный в один тег `<ol>`, применяет значение `upper-roman` для свойства `list-style` тега `<li>`. Когда тот же броузер видит тег `<li>`, вложенный в два тега `<ol>`, он использует значение `upper-alpha` свойства `list-style`. Вложите тег `<li>` в три и четыре тега `<ol>`, и вы увидите соответственно `decimal`- и `lower-alpha`-стили нумерации. Сравните рис. 8.1 с рис. 7.7, где для достижения того же эффекта в упорядоченном списке применялся атрибут `type` тега `<ol>`.

Подобным образом можно устанавливать некий стиль только для тегов, находящихся в определенном контексте. Следующее определение стиля, например, наделит красным цветом содержимое только тех те-

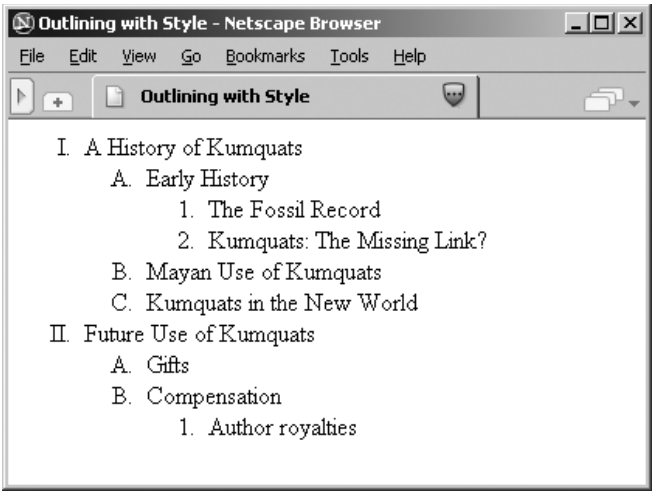

***Рис. 8.1.** Стили вложенных упорядоченных списков*

гов логического подчеркивания (), которые встречаются внутри заголовков первого уровня (в тегах <h1>), и никаких других:

```
h1 em {color: red}
```

Если существует противоречие между двумя контекстными стилями, более специфический контекст побеждает.

В селекторный список могут входить как селекторы, являющиеся именами тегов, так и контекстные селекторы, при этом все они должны отделяться друг от друга запятыми. Все элементы селекторного списка подчиняются общему для них стилевому правилу. Пример:

```
h1 em, p strong, address {color: red}
```

означает, что вы увидите красный цвет всюду, где тег содержится в теге <h1> или тег появляется внутри тега <p>, а также в содержимом тега <adress>.

Для того чтобы было применено контекстное правило, последовательность вложений не обязана точно совпадать с контекстным селектором. К примеру, если тег вложен в тег <ul>, входящий, в свою очередь, в тег <p>, он все еще подчиняется контекстному правилу p strong, определенному выше. Если некоторое вложение тегов подходит для применения нескольких стилевых правил, употребляется более специфичное. В частности, если вы определили два контекстных селектора:

```
p strong {color: red}
p ul strong {color: blue}
```

и использовали в документе последовательность <p>
, будет применено второе, более специфическое, правило, и содержимое тега будет синим.

8.2.4. Универсальные, дочерние и смежные селекторы

Стандарт CSS2 определяет дополнительные конструкции селекторов, кроме уже известных, составляемых при помощи запятых и пробелов. Следующие примеры допустимых селекторов иллюстрируют это утверждение:

```
* {color: purple; font: ZapfDingBats}
ol > li {font-size: 200%; font-style: italic}
h1 + h2 {margin-top: +4mm}
```

В первом примере звездочка, универсальный селектор, указывает, что стилевое правило должно быть применено ко всем элементам документа, так что любой текст в нем должен быть представлен символами набора ZapfDingBats.[1] Во втором примере устанавливаются отношения типа дочерний/родительский, в данном случае между элементами упорядоченного списка.

Третий пример иллюстрирует тип смежного селектора, который относится к тегу, непосредственно следующему за другим тегом в документе. В этом случае всякий раз, когда за заголовком уровня 1 непосредственно стоит заголовок уровня 2, между ними увеличивается вертикальный интервал.

8.2.5. Селекторы атрибутов

Существует возможность присоединить стиль только к тем элементам HTML/XHTML, которые обладают специфическими атрибутами. Для этого следует перечислить нужные атрибуты в квадратных скобках рядом с именем элемента до определения стиля:

```
div[align] { font-style: italic }
div[align=left] {font-style: italic }
div[title~="bibliography"] { font-size: smaller }
div[lang|="en"] {color: green }
```

Первый пример – самый простой. Он выделяет курсивом текстовое содержимое только тех тегов `<div>`, которые имеют атрибут `align`, независимо от значения этого атрибута. Второй пример чуть сложнее. Он затрагивает только те теги `<div>`, у которых атрибут `align` имеет значение `left`.

Третья строчка относится к тегам `<div>`, чей атрибут `title` содержит слово *bibliography*, отделенное одним или несколькими пробелами. Частичное совпадение снова не считается. Если бы вы указали `div[title~="a"]`, стиль относился бы только к тем тегам `<div>`, у которых атрибут `title` содержит букву «a», отделенную пробелами или стоящую в начале или конце названия.

---

[1] Если, конечно, стиль не будет переопределен.

Последний пример определяет теги , у которых атрибуту lang присвоен список слов, разделенных дефисами и начинающихся на «en». Под это условие подходят такие атрибуты как lang=en, lang=en-us и lang=en-uk.

Вы можете сочетать универсальный селектор с селекторами атрибутов, чтобы вашему определению соответствовал любой элемент с указанным атрибутом. Например:

```
*[class=comment] { display: none }
```

Этот селектор спрячет все элементы документа, у которых атрибут class имеет значение comment.

Netscape, Firefox, Opera и другие современные броузеры поддерживают селекторы атрибутов. По неизвестным причинам Internet Explorer этого не делает.

8.2.6. Псевдоэлементы

В документах встречаются отношения между элементами, которые невозможно или неудобно выражать при помощи тегов. Употребление буквицы – это общепринятый прием при печати, но как выделить первую букву абзаца? Такие способы есть, но вам придется отмечать каждый случай в отдельности. Не существует тега для первой буквы абзаца. Бывают ситуации, когда хочется, чтобы броузер автоматически генерировал содержимое, добавляя, например, префикс «элемент №», и автоматически нумеровал каждый элемент упорядоченного списка.

CSS2 вводит четыре псевдоэлемента, позволяющие определять особые отношения и стили для их отображения: first-line (первая строка), first-letter (первая буква), before (перед чем-либо) и after (после чего-либо). Объявляйте эти отношения, добавляя к стандартному элементу разметки отделенный двоеточием суффикс. К примеру:

```
p:first-line {font-size: 200%; font-style: italic}
```

означает, что броузер должен отображать первые строки абзацев курсивом и шрифтом крупнее остального текста в два раза. Подобным образом:

```
p:first-letter {font-size: 200%; float: left}
```

предлагает броузеру сделать первую букву абзаца в два раза крупнее остального текста, сдвинуть ее влево и разрешить первым двум строкам абзаца обтекать большую начальную букву (см. рис. 8.2).[1]

[1] Свойства, которые можно определять при помощи псевдоклассов first-letter и first-line, относятся к свойствам шрифта, цвета и фона, а именно text-decoration, vertical-align, text-transform, line-height и clear. Кроме того, псевдокласс first-letter принимает свойства полей, рамок, подложки и float. Псевдокласс first-line принимает также свойства word-spacing и letter-spacing.

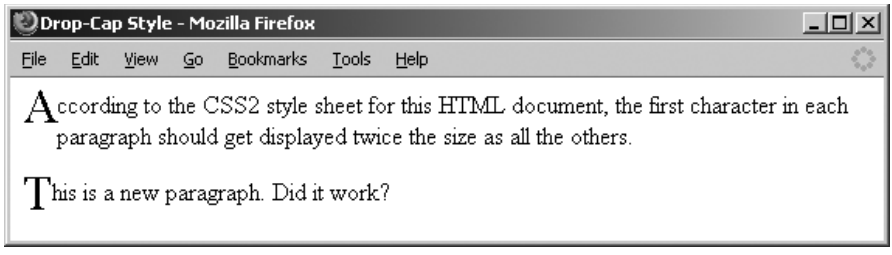

Рис. 8.2. *Использование псевдоэлемента first-letter для выделения первой буквы текста в содержимом тега*

Псевдоэлементы `:before` и `:after` позволяют указывать в документе места, куда вставляется автоматически генерируемое содержимое, в частности особые заголовки и т. д. Стало быть, эти псевдоэлементы идут рука об руку с такими свойствами CSS2, как `counter` и `content`. Следующий пример вас, возможно, заинтересует:

```
ul {counter-reset: item; list-style: none}
ul li:before {content: "Item #" counters(item), " ";
              counter-increment: item}
...
<ul>
  <li> This is item number 1.</li>
  <ul>
    <li> This is sub-item number 1.1.</li>
  </ul>
  <li> This is item number 2.</li>
  <ul>
    <li> This is sub-item 2.1.</li>
    <li> This is sub-item 2.2.</li>
... and so on
```

Все популярные броузеры поддерживают псевдоэлементы, порождающие эффекты, аналогичные показанному на рис. 8.2. При этом Internet Explorer не поддерживает свойство `content`, а Netscape не поддер-

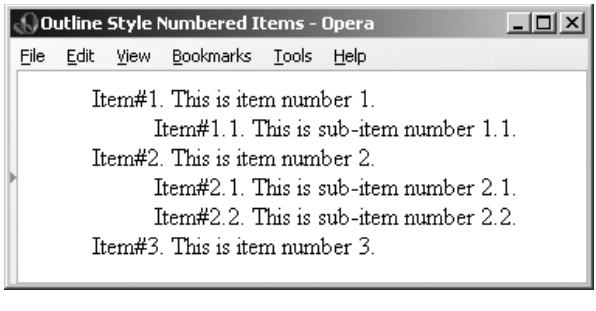

Рис. 8.3. *Стилевые счетчики вместе с псевдоэлементами создают нумерацию, отражающую структуру документа*

живает счетчики. Получается, что только «новички» Firefox и Opera корректно отображают последовательную нумерацию элементов списка, определенную в предшествующем примере и проиллюстрированную на рис. 8.3.

8.3. Стилевые классы

Стандарт CSS2 позволяет вам определить несколько различных стилей для одного элемента путем указания класса для каждого стиля на уровне документа или во внешней таблице стилей. Затем вы в документе явно выбираете стиль, задавая атрибут `class` с нужным вам значением `name` в соответствующем теге.

8.3.1. Регулярные классы

В технических документах часто хочется использовать один стиль абзацев при изложении краткого содержания, другой – для формул и третий – для цитат. Вместо этого стоит определить для каждого из них свой класс стилей:

```
<style type="text/css">
<!--
p.abstract {font-style: italic;
            margin-left: 0.5cm;
            margin-right: 0.5cm}
p.equation {font-family: Symbol;
            text-align: center}
h1, p.centered {text-align: center;
                margin-left: 0.5cm;
                margin-right: 0.5cm}
-->
</style>
```

Приведенный пример показывает: определение стилевого правила для класса состоит в том, что в селекторе к имени тега через точку приписано имя класса. В отличие от «послушного» XHTML-селектора, являющегося именем стандартного тега (который необходимо записывать строчными буквами), имя класса может быть любой последовательностью букв, цифр и дефисов, но всегда должно начинаться с буквы.[1] Будьте, однако, осторожны, регистр в этом случае имеет значение – `abstract` это не то же самое, что `AbsTRact`. Классы входят в селекторные списки наряду с обычными селекторами, отделяясь от других классов и селекторов запятыми, как это показано в третьем примере. Единственное ограничение, вводимое для классов, состоит в том, что они не

[1] Netscape поддерживает таблицы стилей JavaScript и потому не может обслуживать имена классов, которые случайно совпали с ключевыми словами JavaScript. Класс abstract, например, вызовет у Netscape ошибку.

могут вкладываться друг в друга, так что конструкция `p.equation.centered`, например, не является допустимой.

Первое правило в примере, таким образом, создает класс стилей абзаца под названием «abstract», текст которого должен выводиться курсивом с отступом на полсантиметра от правого и левого краев. Подобным образом второе определение стилевого класса «equation» предписывает броузеру центрировать текст и использовать гарнитуру шрифта Symbol при его отображении. Последнее правило создает стиль с центрированным текстом и полусантиметровыми полями, который должен применяться ко всем заголовкам уровня 1, как и к абзацам, атрибут `class` которых имеет значение `centered`.

Чтобы применить к содержимому тега определенный класс, следует вставить в тег атрибут `class`, как это делается в нижеследующем примере (рис. 8.4):

```
<p class=abstract>
This is the abstract paragraph.  See how the margins are indented?
</p>
<h3>The equation paragraph follows</h3>
<p class=equation>
a = b + 1
</p>
<p class=centered>
This paragraph's text should be centered.
</p>
```

Для каждого абзаца значение атрибута `class` – это имя того стилевого класса, который должен быть применен к данному тегу.

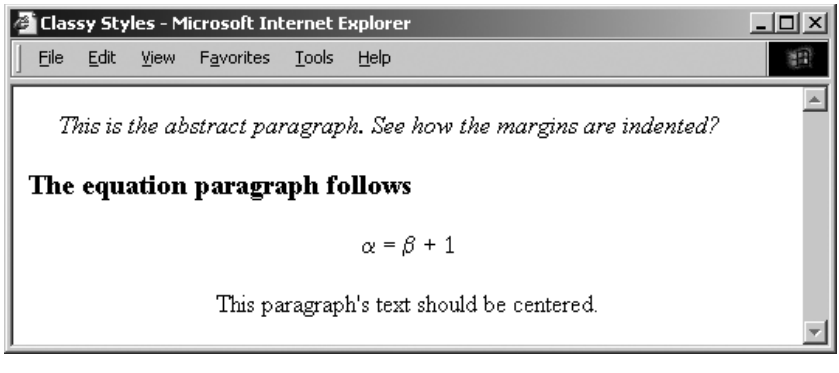

Рис. 8.4. *Используйте классы для различного отображения одного и того же тега*

8.3.2. Родовые классы

Вы можете также создавать классы, не ассоциируя их ни с одним тегом в отдельности, и применять по своему выбору к разнообразным тегам. К примеру:

```
.italic {font-style: italic}
```

создает родовой класс по имени italic. Чтобы применить его, просто включите это имя в качестве значения атрибута class в тег. Так, например, используйте конструкции <p class=italic> и <h1 class=italic>, чтобы вывести курсивом абзац или заголовок.

Родовые классы очень удобны и делают легким применение определенного стиля к разным тегам. Все популярные броузеры поддерживают родовые классы стандарта CSS2.

8.3.3. ID-классы

Почти все HTML-теги допускают атрибут id, присваивающий элементу уникальный в документе идентификатор. Атрибут id может быть целью URL и использоваться при автоматической обработке документа, а также в качестве указания применить к поименованному им элементу определенное стилевое правило.

Для создания стилевого правила, которое броузер, поддерживающий стили, применит только к участкам документа, помеченным атрибутом id, используйте тот же синтаксис, что и для стилевых классов, с тем только отличием, что вместо точки перед именем класса следует писать # перед идентификатором. Вот пример:

```
<style>
<!--
#yellow {color : yellow}
h1#blue {color : blue}
-->
</style>
```

Теперь в документе можно написать <h1 id=blue>, чтобы создать синий заголовок, или вставить id=yellow практически в любой тег, чтобы окрасить его содержимое в желтый цвет. Сочетание атрибутов class и id позволяет в некоторых ограничениях применять к одному элементу два независимых стилевых правила.

Употребление правил, созданных таким способом, обладает существенным недостатком – стандарты HTML и XHTML требуют, чтобы значение атрибута id было уникально в каждом случае его использования в документе. А здесь нам приходится неоднократно указывать одно и то же значение для применения стилевого класса.

Даже если современные броузеры могут позволить вам это сделать, мы настоятельно советуем не создавать и не применять стилевые правила такого типа. Сохраняйте приверженность традиционным классам стилей, и ваши документы будут корректными и жизнестойкими.

8.3.4. Псевдоклассы

В дополнение к традиционным стилевым классам стандарт CSS2 определяет псевдоклассы, позволяющие устанавливать стиль отображения

для некоторых состояний тегов, например изменять стиль показа, когда пользователь щелкает по гиперссылке. Псевдоклассы похожи на регулярные классы, но обладают двумя заметными отличиями – они присоединяются к имени тега двоеточием, а не точкой, и их имена заранее определены, а не произвольно выбираются вами.

Существует семь псевдоклассов, три из которых явным образом ассоциированы с тегом `<a>`.

8.3.4.1. Псевдоклассы гиперссылок

Популярные, поддерживающие стандарт CSS2 броузеры различают три специальных состояния гиперссылок, созданных при помощи тега `<a>`: неиспользованная, используемая и использованная. Броузер способен изменять внешний вид содержимого тега, обозначая его состояние подчеркиванием, например, или цветом. При помощи псевдоклассов можно управлять тем, как эти состояния отображаются, определяя стили для `a:link` (неиспользованная), `a:active` (используемая) и `a:visited` (использованная).

Псевдокласс `:link` управляет внешним видом ссылок, которые не выбраны пользователем и не посещались прежде. Псевдокласс `:active` определяет внешний вид гиперссылок, которые в настоящий момент выбраны и обрабатываются броузером. Псевдокласс `:visited` устанавливает облик ссылок, которые ранее уже были использованы.

Чтобы полностью определить все три состояния тега `<a>`, можно написать:

```
a:link {color: blue}
a:active {color: red; font-weight: bold}
a:visited {color: green}
```

В этом примере неиспользованные ссылки будут синими. Когда посетитель выберет гиперссылку, броузер переменит ее цвет на красный и шрифт сделает жирным. Однажды использованная ссылка вернется к традиционному зеленому цвету.

8.3.4.2. Интерактивные псевдоклассы

Стандарт CSS2 определяет два новых псевдокласса, которые вместе с `:active` реагируют на поступки пользователя и советуют интерактивному агенту, скажем броузеру, как отображать элемент, подвергнувшийся внешнему воздействию. Другими словами, эти два псевдокласса являются динамическими: `hover` и `focus`.

Например, броузер может изменить вид указателя мыши, когда вы проводите ею над гиперссылкой в документе. Подобный эффект нависания, скорее всего, связан со стилем отображения, который возникает только при условии, что мышь попала в область, принадлежащую элементу. В частности, если вы добавите псевдокласс `:hover` к списку стилевых правил для гиперссылки из нашего примера:

```
a:hover {color: yellow}
```

то текст неиспользованной гиперссылки будет синим, но превратится в желтый, когда на него укажут, при щелчке по нему мышью станет красным и останется зеленым после посещения.

Подобным образом псевдокласс `:focus` позволяет изменить стиль элемента, который стал объектом внимания. Элемент может попасть в сферу обзора, если вы дошли до него, нажимая клавишу «Tab», щелкнули на нем мышью или – в зависимости от броузера – подвели к нему курсор. Независимо от способа, каким ему уделено внимание, стилевое правило, ассоциированное с псевдоклассом `:focus`, будет применяться к этому элементу, пока вы не утратите к нему интереса.

8.3.4.3. Псевдоклассы вложения и языка

CSS2-псевдокласс `:first-child` позволяет указать способ, каким можно отобразить элемент, являющийся первым потомком некоего родительского элемента. К примеру, следующее правило применяется только к тем абзацам, которые входят первыми в состав раздела. Перед абзацем не должно быть никаких элементов (обратите внимание на специальное использование знака «больше» по отношению к первому элементу-потомку):

```
div > p:first-child  {font-style: italic}
```

В соответствии с этим правилом первый абзац в следующем HTML-фрагменте будет выведен курсивом, если броузер поддерживает стандарт CSS2, поскольку он является первым дочерним элементом своего раздела. Наоборот, второй абзац идет после заголовка уровня 2, являющегося первым дочерним элементом во втором разделе. Поэтому второй абзац выводится обычным текстом, поскольку не является первым элементом раздела (рис. 8.5):

```
<div>
  <p>
    Я отображаюсь курсивом, потому что мой абзац является первым
    элементом раздела.
  </p>
</div>
<div>
  <h2> Новый раздел</h2>
  <p>
    Я отображен обычным шрифтом, потому что этот параграф – второй
    потомок в разделе.
```

Наконец, стандарт CSS2 определяет новый псевдокласс, который позволяет выбирать элемент, основываясь на его языке. В частности, если включить в тег `<div>` атрибут `lang=fr`, то броузер поймет, что раздел содержит текст на французском языке, и, возможно, будет обращаться с ним в особой манере. Другой путь – самим установить определенный способ отображения, определив псевдокласс `:lang`. К примеру:

```
div:lang(it) {font-family: Roman}
```

говорит, что текст в разделах, содержащих итальянский язык, должен применять шрифты семейства Roman. Как считаете, подходит? Заметьте, что язык указывается в скобках сразу после ключевого слова `lang`. Используйте для псевдокласса `:lang` те же коды языков по стандарту ISO, что и для атрибута `lang`. [атрибут lang, 3.6.1.2]

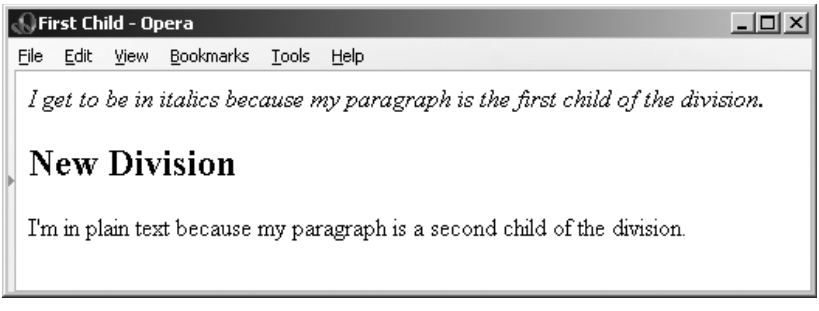

Рис. 8.5. *Псевдокласс fisrt-child в действии*

8.3.4.4. Как броузеры поддерживают псевдоклассы

Ни один из популярных броузеров пока не поддерживает псевдоклассы `:lang` и `:focus`. Все они обрабатывают псевдоклассы `:link`, `:active`, `:hover` и `:visited` для тега гиперссылки (`<a>`), а также `:first-child`. Хотя по стандарту `:active` может использоваться и с другими элементами, ни один из броузеров пока не поддерживает его употребление ни с каким отличным от `<a>` тегом.

8.3.5. Составные классы

Можно соединять псевдоклассы с регулярными классами, приписывая имя псевдокласса к имени класса в селекторе. Вот несколько способов, определяющих простые, нормальные и затейливые якоря:

```
a.plain:link, a.plain:active, a.plain:visited {color: blue}
a:link {color: blue}
a:visited {color: green}
a:active {color: red}
a.fancy:link {font-style: italic}
a.fancy:visited {font-style: normal}
a.fancy:active {font-weight: bold; font-size: 150%}
```

Простая (`plain`) версия `<a>` всегда синяя вне зависимости от состояния гиперссылки.[1] Соответственно нормальная гиперссылка имеет снача-

[1] Установка совершенно одинаковых стилей для нормальной, посещенной и активной ссылок является дурным тоном в оформлении страниц. По меньшей мере это неудобно для пользователей. – *Примеч. науч. ред.*

ла синий цвет, становится красной, когда активна, и превращается в зеленую после посещения. Затейливая (`fancy`) ссылка наследует цветовую схему нормальной, но ее текст до использования выводится курсивом, после посещения возвращается к обычному и вырастает в полтора раза и становится жирным, когда она активна.

Два слова в предупреждение по поводу класса `fancy` – указывая изменение размера шрифта для преходящего свойства, вы задаете броузеру работу по отображению окружающего содержимого. Поскольку некоторые броузеры работают на медленных машинах, этот эффект может остаться незамеченным определенной частью ваших пользователей. Кроме того, поскольку реализация такого сорта изменений в отображении довольно хлопотна, маловероятно, что большинство броузеров будут поддерживать радикальную модификацию внешнего вида в псевдоклассах тега `<a>`.

8.3.6. Наследование у классов

Классы наследуют свойства стиля от своих базовых тегов. К примеру, каждое свойство обычного тега `<p>` применяется к специально установленному классу абзацев, если класс не переопределяет этого свойства.

Классы не могут наследовать от других классов, а только от «свободного» тега. Таким образом, как можно большее число стилевых свойств необходимо определить в спецификациях тега, а в правила классов включать только те из них, что специфичны для данного класса. Это упростит поддержание и совместное использование стилевых классов, особенно в случае больших собраний документов.

8.4. Стилевые свойства

Определения стилевых правил в соответствии со стандартом CSS2 основываются на богатом наборе свойств, позволяющих управлять поддерживающими стили броузерами в вопросах, связанных со способом представления документов пользователю. Стандарт разбивает эти свойства на шесть групп: шрифт, фон и цвет, текст, рамка и макет, списки, классификация тегов. Мы будем придерживаться этой типологии и перед тем, как погрузиться в их анализ, обсудим значения свойств и вопросы наследования.

Сводку по стилевым свойствам можно найти в приложении C.

8.4.1. Значения свойств

Большинство свойств устанавливает значение для некоторой характеристики вашего документа, определяющей, как броузер выведет его на экран. В качестве примера назовем размер символов или цвет заголовков второго уровня. Как было сказано ранее при описании синтаксиса стилей, чтобы задать значение CSS2-свойства, вы ставите двоеточие после ключевого слова, обозначающего это свойство, а за двоето-

чием – одно или несколько чисел или ключевых слов, разделенных пробелами или запятыми. Например:

```
color:blue
font-family: Helvetica, Univers, sans-serif
```

Здесь `color` и `font-family` – свойства, а `blue` и названия шрифтов – их значения.

Значения свойств бывают восьми различных видов: ключевые слова, значения длины, процентные значения, URL, цвета, углы, время и частотные значения.

8.4.1.1. Значения свойств – ключевые слова

Свойство может иметь значение, являющееся ключевым словом, выражающим действие или размер. К примеру, результат присваивания свойству значений `underline` (подчеркнуть) или `line-through` (зачеркнуть) очевиден. И размеры, связанные со свойствами, вы можете выразить при помощи ключевых слов, таких как `small` (маленький) или `xx-large` (весьма большой). Некоторые ключевые слова выражают отношение: например, `bolder` (жирнее) является допустимым значением для свойства `font-weight`. Ключевые слова в качестве значений свойств не чувствительны к регистру – `Underline`, `UNDERLINE` и `underline` в равной мере возможны.

8.4.1.2. Значения свойств – длины

Так называемая «длина» `length` (термин из стандарта CSS2) явным образом устанавливает размер для свойства. Длина является числом, иногда десятичной дробью. Перед длиной могут стоять знаки «+» или «–», показывающие, что величина должна быть прибавлена к текущему значению свойства или вычтена из него. За длинами должно непосредственно (без промежуточных пробелов) следовать двухбуквенное обозначение единицы измерения.

Существуют три вида единиц измерения длин: относительные, пикселы и абсолютные. Относительные единицы определяют размер, который в качестве эталона (единицы) принимает некоторое другое свойство содержимого. В настоящее время есть только две относительные единицы: `em`, представляющая собой ширину строчной буквы «m» в текущем шрифте; и `x-height`, представляющая собой высоту буквы «x» в текущем шрифте (записывается `ex`). Пикселы – это светящиеся цветные точки на экране компьютера или телевизора, образующие текст и изображение. Пиксел как единица измерения (записывается `px`) равен экранному пикселу, так что размер некоторых свойств вы можете определять количеством точек на дисплее. Абсолютные единицы длин всем нам знакомы лучше. Это дюймы (`in`), сантиметры (`cm`), миллиметры (`mm`), пункты (`pt`, 1/72 дюйма) и пики (`pc`, двенадцать пунктов).

Ниже перечислены примеры допустимых длин, хотя не все единицы признаются поддерживающими стили броузерами:

```
1in
1.5cm
+0.25mm
-3pt
-2.5pc
+100em
-2.75ex
250px
```

8.4.1.3. Значения свойств – процентные значения

Подобно относительным длинам процентные значения описывают величины в терминах отношений к неким другим величинам, свойственным содержимому. Они могут иметь необязательный знак, показывающий, что процентное значение нужно прибавлять к текущему значению свойства или вычитать из него, а также необязательную десятичную точку. Процентные значения заканчиваются знаком процента (%). К примеру:

```
line-height: 120%
```

вычисляет расстояние между строками, которое должно стать на 20 % больше текущего (обычно зависящего от размера шрифта). Отметьте, что это значение не является динамическим – изменение размера шрифта после того, как это правило было обработано броузером, не повлияет на вычисленную высоту строки.

8.4.1.4. URL как значение свойств

Некоторые свойства также принимают (скорее даже ожидают) URL в качестве значения. Синтаксис CSS2 URL в стилевых свойствах отличается от традиционного в HTML/XHTML:

```
url(service://server.com/pathname)
```

Для свойств, определенных в стандарте CSS2, ключевое слово `url` обязательно, как и открывающая и закрывающая скобки. Не оставляйте между словом `url` и открывающей скобкой пробелов. В `url` может находиться как абсолютный, так и относительный URL. Однако имейте в виду, что относительный URL отсчитывается от URL-адреса таблицы стилей, в которой он содержится, то есть необязательно от базового URL-адреса вашего документа. Это означает, что если вы используете `url`-значение на уровне документа или во встроенном в тег стиле, то итоговый URL отсчитывается от URL текущего документа. В остальных случаях URL отсчитывается относительно внешней таблицы стилей, присоединенной к документу посредством директивы `@import` или с помощью тега `<link>`.

8.4.1.5. Значения свойств – цвета

Цветовые значения определяют цвет в свойстве (удивлены?). Цвет можно указать по названию или при помощи шестнадцатеричного

RGB-триплета так же, как это делается для обычных HTML/XHTML-атрибутов или десятичного RGB-триплета, что допустимо только для стилевых свойств. Названия цветов и шестнадцатеричная RGB-нотация описаны в приложении G.

В стандарте CSS2 вам разрешается присваивать только одну шестнадцатеричную цифру вместо двух красной, зеленой и синей составляющим цвета. Каждая цифра трехзначного кода удваивается при переводе в традиционный шестизначный триплет. Таким образом, цвет #78C эквивалентен цвету #7788CC. Трехзначные значения цвета в основном пригодны только для простых цветов.

Десятичная RGB-нотация выглядит иначе:

```
rgb(red, green, blue)
```

Значения интенсивности `red`, `green` и `blue` – это целые числа в интервале от нуля до 255 или целые проценты. Как и в случае URL, не оставляйте пробелов между `rgb` и открывающей скобкой.

К примеру, в десятичных RGB-обозначениях белый цвет – это `rgb(255, 255, 255)` или `rgb(100%, 100%, 100%)`, а нейтральный желтый – это `rgb(127, 127, 0)` или `rgb(50%, 50%, 0%)`.

8.4.1.6. Значения свойств – углы, время и частота

Для некоторых свойств, таких как направление стрелки компаса, требуется значение, задающее угол. Эти свойства принимают числовые значения, за которыми следуют соответствующие единицы измерения: `deg` (градусы), `grad` (грады, то есть сотые доли прямого угла) или `rad` (радианы). Аналогичным образом значения времени выражаются числами, после которых стоит либо `ms` (миллисекунды), либо `s` (секунды).

Частоты тоже выражаются числами, а в качестве единиц измерения можно указывать `Hz` (герцы) или `kHz` (1 килогерц = 1000 Hz). Интересно, что такие единицы, как `MHz` и `GHz`, не предусмотрены, т. к. в CSS2 имеются в виду только аудиочастоты, а не частоты электромагнитных волн, применяемых в телевидении, радиовещании, Bluetooth и других технологиях.

8.4.2. Наследование свойств

В дополнение к специальным правилам для определенных элементов свойства и их значения для тегов, содержащихся внутри других тегов, наследуются от родительского тега. Так, установив свойство для тега `<body>`, вы применяете это свойство ко всем тегам в теле вашего документа, кроме тех, в которых оно переопределено. Чтобы сделать весь текст в документе синим, нужно только написать:

```
body {color: blue}
```

вместо того чтобы создавать правило для любого используемого в нем тега.

Это наследование распространяется на все уровни. Если создать затем тег с текстом другого цвета, поддерживающий стили броузер отобразит весь текст в разделе и в тегах, содержащихся в нем, этим новым цветом. Когда завершится, цвет вернется к тому, что определен в теге <body>.

Во многих последующих описаниях свойств мы называем тег, содержащий текущий тег, *родительским элементом* этого тега.

8.4.3. Свойства шрифтов

Больше всего жалуются на то, что HTML и его потомку XHTML не хватает стилей и характеристики шрифтов, которые реализованы даже в простейших текстовых редакторах. Различные атрибуты <font> частично снимают эту проблему, но их скучно применять, поскольку каждое изменение шрифта требует отдельного тега <font>.

Таблицы стилей, конечно, меняют дело. Стандарт CSS2 предоставляет семь свойств шрифта, изменяющих вид текста, содержащегося в тегах, к которым они применяются: font-family, font-size, font-size-adjust, font-style, font-variant, font-stretch и font-weight. Вдобавок существует универсальное свойство font, которое позволяет объявить все модификации шрифта.

Только будьте добры, помните, что таблицы стилей не способны преодолеть ограниченные возможности системы клиента и броузер не сможет показывать фокусы со шрифтами, если у него их нет.

8.4.3.1. Свойство font-family

Свойство font-family принимает в качестве значения список разделенных запятыми названий шрифтов (гарнитур). Броузер использует первый упомянутый в списке шрифт[1], который при этом инсталлирован и может быть отображен дисплеем клиентской машины.

Названия шрифтов соответствуют конкретным начертаниям, скажем, Helvetica или Courier, или семействам шрифтов, определенным стандартом CSS2, таким как serif, sans-serif, fantasy и monospace. Каждый броузер сам решает, какой представитель семейства будет использован в действительности. К примеру, Courier – это самый популярный выбор моноширинного шрифта.

Поскольку доступные шрифты значительно меняются от броузера к броузеру, вы должны предусмотреть при определении их стилей возможность выбора, вплоть до указания подходящего семейства. В частности:

```
h1 {font-family: Helvetica, Univers, sans-serif}
```

[1] К сожалению, нет возможности указывать шрифты, в названии которых присутствуют символы, не входящие в набор символов Latin1. – *Примеч. науч. ред.*

предлагает броузеру поискать Helvetica, а если его нет, использовать Univers. Если ни один из этих шрифтов не окажется доступным на дисплее клиента, броузер применяет семейство начертаний `sans-serif`.

Заключайте названия шрифтов, содержащие пробелы, скажем New Century Schoolbook, в кавычки. Например:

```
p {font-family: Times, "New Century Schoolbook", Palatino, serif}
```

В объявлении стиля, встроенном в тег, пара двойных кавычек может привести к неприятностям. Используйте во встроенных стилях одинарные кавычки (апострофы):

```
<p style="font-family: Times, 'New Century Schoolbook', Palatino, serif">
```

На практике нет необходимости применять кавычки, потому что значения, определяющие имена шрифтов, разделены запятыми, а «нормальный» броузер проигнорирует пробелы. Таким образом, следующие конструкции законны:

```
p {font-family: Times, New Century Schoolbook, Palatino, serif}
<p style="font-family: Times, New Century Schoolbook, Palatino, serif">
```

Все же мы на всякий случай советуем использовать кавычки. Это хорошая привычка, и она избавляет ваш код от неоднозначностей.

8.4.3.2. Свойство font-size

Свойство `font-size`, определяющее размер шрифта, в качестве значений может принимать абсолютные и относительные длины, проценты и ключевые слова. В частности:

```
p {font-size: 12pt}
p {font-size: 120%}
p {font-size: +2pt}
p {font-size: medium}
p {font-size: larger}
```

Первое правило, вероятно, употребляется чаще всего, поскольку выглядит самым знакомым, – оно устанавливает размер шрифта равным определенному числу пунктов (двенадцати в данном примере). Второе правило устанавливает размер шрифта на 20% больше, чем в родительском элементе. Третье – увеличивает нормальный размер шрифта на 2 пункта. В четвертой строке выбирается заранее определенный броузером размер, обозначенный словом `medium`. Допустимыми ключевыми словами для абсолютного размера могут служить `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large` и `xx-large`, которые обычно соответствуют семи размерам шрифта, используемым для атрибута `size` тега `<font>`.

Последнее правило выбирает следующий по размеру шрифт по сравнению с тем, что используется в родительском элементе. Таким образом, если раньше применялся `medium`, то теперь он превратится в `large`. Вы также можете написать `smaller` с легко предсказуемым результатом.

Ни один из современных броузеров не обращается корректно с предписаниями увеличить или уменьшить размер шрифта. Они скорее проигнорируют знак увеличения/уменьшения и примут его значение за абсолютный размер. Так что результатом применения третьего правила будет шрифт размером в два пункта, а совсем не на два пункта больше нормального.

8.4.3.3. Свойство font-stretch

Помимо шрифтов различных размеров их семейства содержат уплотненные и разреженные версии, символы в которых либо сжаты, либо растянуты. Применяйте свойство `font-stretch` для выбора более сжатых или более растянутых символов вашего шрифта.

Используйте значение свойства `normal`, чтобы остановиться на версии шрифта нормального размера. Относительные значения `wider` и `narrower` выбирают следующий, более широкий или соответственно более узкий варианты начертания шрифта, но не шире и не уже крайних («ultra») версий семейства.

Оставшиеся значения свойства `font-stretch` назначают определенные варианты из семейства шрифтов. Начиная с самого плотного и заканчивая самым жидким, значения таковы: `ultra-condensed`, `extra-condensed`, `condensed`, `semi-condensed`, `semi-expanded`, `expanded`, `extra-expanded` и `ultra-expanded`.

Свойство `font-stretch` предполагает, естественно, что на компьютере поддерживаются растягиваемые шрифты. Тем не менее современные популярные броузеры игнорируют это свойство.

8.4.3.4. Свойство font-size-adjust

Если не вдаваться в тонкости, то разборчивость и видимый размер шрифта зависят в основном от его характеристического отношения (aspect ratio) – отношения высоты строчной буквы к номинальному размеру (измеряемому по высоте заглавной буквы). Шрифты с характеристическим отношением, приближающимся к единице, разборчивей при мелком размере, чем шрифты с этим отношением вблизи нуля.

Кроме того, в зависимости от величины характеристического отношения видимый размер одного шрифта может быть явно больше или меньше другого шрифта, хотя их номинальные размеры одинаковы. Поэтому, когда один шрифт оказывается недоступным для броузера, применение его заменителя может исказить внешний вид документа.

Свойство `font-size-adjust` позволяет подогнать размер замещающего шрифта с учетом его характеристического отношения, чтобы он наилучшим образом подходил для отображения. Используйте значение этого свойства `none`, чтобы игнорировать характеристическое отношение. В противном случае присвойте свойству десятичное дробное число в интервале от нуля до единицы. Обычно назначается характери-

стическое отношение для шрифта, который должен быть выбран в первую очередь. Владеющий стилями броузер выведет тогда замещающий шрифт с размером, вычисленным по формуле:

```
s = (n/a) * fs
```

где `s` – это новый размер, используемый для вывода замещающего шрифта, вычисленный как частное от деления значения `n` свойства `font-size-adjust` на характеристическое отношение замещающего шрифта `a` и умноженное затем на размер текущего шрифта `fs`. К примеру, предположим, что вы предлагаете вывести текст шрифтом Times New Roman с характеристическим отношением 0,45, но броузер его не обнаруживает и применяет вместо него шрифт Comic Sans MS с отношением 0,54. Тогда, чтобы сохранить приблизительно тот же видимый размер (для Times New Roman это 18 px), поддерживающий стандарт CSS2 броузер, учитывая значение свойства `font-size-adjust` (0,45 для Times New Roman), выведет на экран или напечатает текст шрифтом Comic Sans Microsoft с меньшим размером 0,45/0,54×18 px = 15 px. К сожалению, мы не можем показать вам, как популярные броузеры это делают, так как они это не поддерживают.

8.4.3.5. Свойство font-style

Используйте свойство `font-style`, чтобы отобразить текст наклонно. По умолчанию принимается стиль `normal`, который можно переменить на `italic` или `oblique`. К примеру:

```
h2 {font-style: italic}
```

выводит текст в заголовках уровня 2 курсивом. Netscape 4 поддерживает только значение `italic` для свойства `font-style`. Все современные броузеры воспроизводят оба значения, но обычно трудно отличить курсив от наклонного начертания.

8.4.3.6. Свойство font-variant

Свойство `font-variant` позволяет выводить текст маленькими заглавными буквами. Принятое по умолчанию значение этого свойства – `normal` – указывает на традиционную версию шрифта. Но вы можете придать этому свойству значение `small-caps`, выбирая версию, в которой строчные буквы заменяются маленькими заглавными (т. е. капителью).

Все современные броузеры поддерживают это свойство. Internet Explorer версий 4 и 5 некорректно реализует `small-caps`, выводя текст заглавными буквами.[1]

[1] Начиная с версии 5.01 Internet Explorer корректно обрабатывает данное свойство. – *Примеч. науч. ред.*

8.4.3.7. Свойство font-weight

Свойство font-weight заведует весом или жирностью букв. По умолчанию принимается значение normal. Можно установить значение bold, чтобы получить жирную версию шрифта, или использовать относительные значения bolder и lighter для применения версии, которая будет более или соответственно менее жирной, чем принятая для родительского элемента.

Для определения различных уровней «жирности» текста используются числа, кратные 100, от 100 (самый тонкий) до 900 (самый жирный). Значение 400 соответствует значению normal, а 700 соответствует bold.

Все современные броузеры поддерживают это свойство.

8.4.3.8. Свойство font

Чаще, чем хотелось бы, приходится сразу определять не одно, а несколько свойств шрифтов, чтобы отрегулировать отображение тега. Полная спецификация шрифта может оказаться несколько громоздкой:

```
p {font-family: Times, Garamond, serif;
   font-weight: bold;
   font-size: 12pt;
   line-height: 14pt}
```

Чтобы сократить эти трудоемкие и, вероятно, неудобные для восприятия описания, используйте всеобъемлющее свойство font и соберите все объявления в одно:

```
p {font: bold 12pt/14pt Times, Garamond, serif}
```

Группировка атрибутов шрифтов и порядок их перечисления существенны для свойства font. Первыми нужно указывать атрибуты стиля, жирности и варианта шрифта, за которыми должны следовать размер шрифта и высота строки, разделенные слэшем, и последним задается список гарнитур. Из всех свойств обязательны размер и гарнитура; прочие могут быть опущены.

Вот несколько примеров определения свойства font:

```
em {font: italic 14pt Times}
h1 {font: 24pt/48pt sans-serif}
code {font: 12pt Courier, monospace}
```

Первый пример сообщает поддерживающему стили броузеру, что логическое подчеркивание следует выразить, использовав курсивный шрифт Times размером 14 пунктов. Второе стилевое правило говорит о том, что текст в теге <h1> отображается имеющимся в наличии шрифтом семейства sans-serif размером 24 пункта с 24 пунктами дополнительного расстояния между строками. Наконец, для текста в теге <code> установлен Courier или, если его нет, выбранный по усмотрению броузера моноширинный шрифт размером 12 пунктов.

Вообразите сами, какие безобразия можно натворить, изобретательно применяя стили шрифтов. Если у вас не получается, загляните в последний номер журнала «Wired».[1]

8.4.4. Выбор и синтез шрифтов

Первый CSS-стандарт, CSS1, определял упрощенный алгоритм соответствия шрифтов. Если указанный шрифт отсутствует на компьютере клиента, следует подставить неспецифичный шрифт. Конечно, результаты в большинстве случаев не радуют глаз и способны вызвать полный беспорядок на экране. Кроме того, нередко оказывается, что какой-то из имеющихся шрифтов подошел бы лучше, чем шрифт общего назначения. Стандарт CSS2 значительно расширяет модель соответствия шрифтов, принятую в CSS1, и содержит новую специальную директиву (так называемое правило at), позволяющую авторам определять, загружать и использовать в документах новые шрифты.

8.4.1.1. Алгоритм соответствия шрифтов в CSS2

В стандарте CSS2 алгоритм соответствия шрифтов состоит из четырех шагов. Первый шаг заключается в применении указанного шрифта, если он находится на компьютере пользователя. Это может быть одно из нескольких семейств шрифтов, указанных в стилевом правиле и подставленных в порядке появления.

Второй шаг предпринимается, когда ни один из шрифтов, перечисленных в правиле, не установлен на компьютере пользователя. Броузер пытается найти наиболее похожий из локальных шрифтов. Например, запрос на шрифт Helvetica может привести к использованию шрифта Arial – аналогичного шрифта без засечек.

Третий шаг алгоритма состоит в попытке броузера синтезировать шрифт путем модифицирования локального шрифта в соответствии с указанным. Например, запрос на шрифт Helvetica размером 72 пункта можно удовлетворить за счет масштабирования 12-пунктового шрифта Arial до требуемого размера.

При неудачном завершении трех предыдущих шагов броузер может предпринять четвертый и загрузить требуемый шрифт, если автор предоставил подходящее определение внешнего шрифта. Такие определения создаются с помощью специальной директивы `@font-face`, имеющей такой синтаксис:

```
@font-face {
   descriptor : value;
   ...
   descriptor : value
   }
```

[1] «Wired» – один из популярных ежемесячных англоязычных журналов, специализируется в основном на футурологии. – *Примеч. науч. ред.*

Каждая специальная директива `@font-face` определяет для броузера новый шрифт. Последующие запросы на шрифты могут быть удовлетворены этими новыми шрифтами. Броузер использует различные дескрипторные значения, чтобы гарантировать соответствие подставленного шрифта запрошенному.

8.4.4.2. Базовые дескрипторы шрифта

Базовые дескрипторы шрифта, указываемые в специальной директиве `@font-face`, соответствуют шрифтовым свойствам CSS2 и принимают те же значения, что и эти свойства. Иными словами, вы можете указывать дескрипторы `font-family`, `font-style`, `font-variant`, `font-weight`, `font-stretch` и `font-size` и их допустимые значения, чтобы определить новый шрифт для броузера. Например:

```
@font-face {
   font-family : "Kumquat Sans";
   font-style : normal, italic;
   src : url("http://www.kumquat.com/foundry/kumquat-sans")
   }
```

Здесь определяется шрифт Kumquat Sans, который можно загрузить с сайта *www.kumquat.com*. Для этого загружаемого шрифта доступны обычная версия и курсив. Поскольку другие дескрипторы не указаны, броузер будет предполагать, что остальные свойства шрифта (жирность, вариант и т. д.) определяются в самом шрифте.

В общем случае отсутствие какого-либо дескриптора шрифта позволяет броузеру брать для соответствующего свойства любое значение. Указывая одно или несколько значений дескриптора шрифта, вы ограничиваете для броузера набор допустимых значений. Поэтому, когда вы определяете шрифт таким способом, вам следует быть как можно конкретнее, чтобы броузер впоследствии сделал оптимальный выбор. Например, если у шрифта нет курсивной версии и вы ничего не предложите броузеру, он может использовать неподходящий шрифт, когда встретит запрос на курсив для этого шрифта.

8.4.4.3. Дескриптор src

Дескриптор `src` в специальной директиве `@font-face` сообщает броузеру, откуда брать шрифт. Для загружаемых шрифтов значением этого дескриптора является URL его документа, выраженный в соответствии с синтаксисом CSS2 с помощью ключевого слова `url`. Чтобы сослаться на локальные шрифты (то есть установленные на компьютере пользователя), напишите ключевое слово `local` вместо `url` и укажите локальное название шрифта.

Значением дескриптора `src` может быть и список, элементы которого разделены запятыми. Например, в коде, приведенном выше, мы могли написать:

```
src : url("http://www.kumquat.com/foundry/kumquat-sans"), local("Lucida Sans")
```

Это означает команду броузеру загрузить шрифт Kumquat Sans с сайта *www.kumquat.com*, а если это не получится, поискать локальный шрифт Lucida Sans.

Вы можете дать броузеру некоторые подсказки. Стандарт CSS2 намеренно либерален в том, что касается формата шрифтового файла. Сознавая, что существует несколько различных форматов шрифтов, создатели стандарта позволяют вам использовать любой, какой только пожелаете, и предполагают, что броузер сам во всем разберется. Чтобы дать броузеру подсказку насчет формата, напишите ключевое слово `format`, а за ним одно или несколько названий формата, например, так:

```
src : url("http://www.kumquat.com/foundry/kumquat-sans") format("type-1"),
    local("Lucida Sans") format("truetype", "intellitype")
```

В этом случае внешний шрифт имеет формат Type 1, а локальные версии шрифта Lucida Sans доступны как в формате TrueType, так и в Intellifont. Другими допустимыми значениями формата являются `true-doc-pfr`, `opentype`, `embedded-opentype`, `truetype`, `truetype-gx` и `speedo`.

8.4.4.4. Дополнительные дескрипторы шрифта

В дополнение к базовым дескрипторам шрифта в CSS2 предусмотрен ряд расширенных дескрипторов, которые уточняют определение шрифта. Среднестатистическому разработчику веб-страниц эти дескрипторы не нужны, но знатоки, возможно, сочтут их небесполезными.

Дескриптор `unicode-range` принимает список значений Unicode, разделенных запятыми. Каждое значение состоит из префикса `U+` и шестнадцатеричного числа. Вы можете указывать диапазоны значений, разделяя границы диапазона дефисом. Вопросительный знак соответствует любому значению в данной позиции.

Дескриптор `unicode-range` призван точно указать глифы, определенные в шрифте. Если символы, используемые в документе, недоступны, броузер не загрузит шрифт. Так, значение `U+2A70` показывает, что шрифт содержит соответствующий глиф в этой позиции. Запись `U+2A7?` задает символы от **2A70** до **2A7F**, а запись `U+2A70-2A9F` – более широкий диапазон. Как правило, этот дескриптор применяется, чтобы ограничить использование специальных символов только теми, что определены в шрифте.

Дескриптор `units-per-em` принимает одно числовое значение, определяющее ширину самой широкой буквы в шрифте. Это значение, называемое «эм» (em), важно, если вы используете его как единицу измерения для задания значений других дескрипторов.

Дескриптор `panose-1` принимает ровно десять целочисленных значений, разделенных пробелами и соответствующих особой характеристике шрифта, носящей имя Panose-1. Обсуждение Panose-1 выходит за рамки этой книги, и мы отсылаем заинтересованного читателя к специальной литературе.

Дескрипторы stemv и stemh определяют толщину (в «эмах») вертикальных и горизонтальных штрихов в начертании символов шрифта. Аналогичным образом дескрипторы cap-height и x-height определяют высоту глифов в верхнем и нижнем регистре. Наконец, дескрипторы ascent и descent определяют максимальные размеры верхнего и нижнего элементов шрифта. Если вы используете любой из этих дескрипторов, вы должны задать дескриптор units-per-em.

Дескриптор slope определяет наклон вертикальных штрихов шрифта. Он играет важную роль при поиске курсивных и наклонных версий шрифта.

Дескрипторы baseline, centerline, mathline и topline определяют обычную, центральную, математическую и верхнюю базовые линии шрифта. Все они принимают числовые значения, выраженные в «эмах». Для них вы тоже должны задать дескриптор units-per-em.

Дескриптор bbox принимает ровно две пары координат (X, Y), определяющие левый нижний и правый верхний углы ограничивающего прямоугольника шрифта. Дескриптор bbox важен, когда броузер решает синтезировать шрифт на основе данного. Указывая размеры ограничивающего прямоугольника, вы гарантируете, что синтезированный шрифт займет столько же места, сколько и запрошенный.

Дескриптор widths принимает список диапазонов Unicode-кодов, перечисленных через запятую, за которыми следуют отделенные пробелами значения, определяющие ширину символов в диапазоне. Если вы укажете одно значение для диапазона, все символы в диапазоне будут иметь одинаковую ширину. Когда указаны несколько значений, каждое относится к соответствующему символу из диапазона. Подобно дескриптору bbox дескриптор widths обеспечивает хорошую степень совпадения синтезированного шрифта с запрошенным.

И наконец, необязательный дескриптор definitions-src задает URL-адрес файла, содержащего все дескрипторы шрифта. Это удобно при детализированном определении шрифта. Вместо того чтобы включать длинное описание в каждый документ или стилевую таблицу, где используется этот шрифт, вы задаете дескрипторы один раз в отдельном файле и ссылаетесь на него с помощью дескриптора definitions-src.

8.4.5. Свойства цвета и фона

Каждый элемент в документе обладает цветами фона и переднего плана. В некоторых случаях фон представляет собой многокрасочное изображение. Стилевые свойства цвета и фона управляют и красками, и изображениями.

Дочерний элемент в HTML/XHTML обычно наследует цвет переднего плана от родительского элемента. В частности, если вы назначили тексту в теге <body> красный цвет, поддерживающий стили броузер будет отображать текст заголовков и абзацев красным.

Свойства фона ведут себя иначе – они не наследуются. Вместо этого каждый элемент имеет принятый по умолчанию прозрачный цвет фона, позволяющий фону родительского элемента проступать сквозь него. Таким образом, установка изображения фоном для тега <body> не приведет к постоянной его перезагрузке при выводе каждого элемента в теле документа. Вместо этого броузер однажды загрузит изображение и сделает его задником документа, который будет служить фоном для всех элементов, не имеющих своего собственного, явно выраженного в цвете или рисунке.

Все современные броузеры поддерживают свойства цвета и фона.

8.4.5.1. Свойство background-color

Свойство background-color управляет (вы угадали!) цветом фона элемента. Присваивайте этому свойству в качестве значения цвет или ключевое слово transparent. Значение по умолчанию – transparent (прозрачный). Результат будет естественным.

Тогда как вы, скорее всего, привыкли устанавливать цвет фона для всего документа при помощи специальных атрибутов тега <body>, стилевое свойство background-color может применяться к любому элементу. К примеру, чтобы установить фоновый цвет для одного элемента маркированного списка, напишите:

```
<li style="background-color: blue">
```

Подобным образом, все заголовочные ячейки в таблицах документа можно сделать инверсными:

```
th {background-color: black; color: white}
```

Если хотите, чтобы текст, на важности которого вы настаиваете, действительно выделялся, покрасьте его фон в красный цвет:

```
em {background-color: red}
```

8.4.5.2. Свойство background-image

Свойство background-image помещает изображение позади содержимого элемента. Его значением является либо URL, либо ключевое слово none, которое принимается по умолчанию.

Подобно цвету, фоновое изображение можно расположить позади как всего документа, так и выбранного элемента. При помощи этого стилевого свойства легко поместить изображение под таблицу или под выбранный текст:

```
<table style="background-image: url(backgrounds/woodgrain.gif)">
li.marble {background-image: url(backgrounds/marble.gif)}
```

Первый пример использует встроенное в тег стилевое правило, чтобы подложить под таблицу узор древесных волокон. Второй – определяет класс элементов списка, который помещает мраморный фон под содержимое тегов <li> с атрибутом class=marble. А вот этот XHTML-пример:

```
<h2>Here's what's for dinner tonight:</h2>
<ul>
   <li class="marble">Liver with Onions</li>
   <li class="marble">Mashed Potatoes and Gravy</li>
   <li class="marble">Green Beans</li>
   <li class="marble">Choice of Milk, Tea, or Coffee</li>
</ul>
<h2>And for dessert:</h2>
<ul>
   <li>Creamed Quats in Milk  (YUM! YUM!)</li>
</ul>
```

приводит к результату, показанному на рис. 8.6.

Если изображение больше, чем содержащий его элемент, оно будет обрезано по границам области, занимаемой элементом. Если изображение окажется меньше, броузер «замостит» его копиями пространство, отведенное под элемент. Способ укладки изображения определяется значением атрибута `background-repeat`.

Вы управляете начальным положением изображения по отношению к элементу при помощи свойства `background-position`. Поведение изображения при прокручивании документа определяется свойством `background-attachment`.

Хотя может показаться, что цвет фона и фоновое изображение исключают друг друга, обычно следует определить и то и другое. Тогда, если изображение оказывается недоступным, что случается, когда пользователь запрещает его автоматическую загрузку, броузер отобразит вместо него цветной фон. Более того, если изображение имеет прозрачные области, фоновый цвет их заполнит.

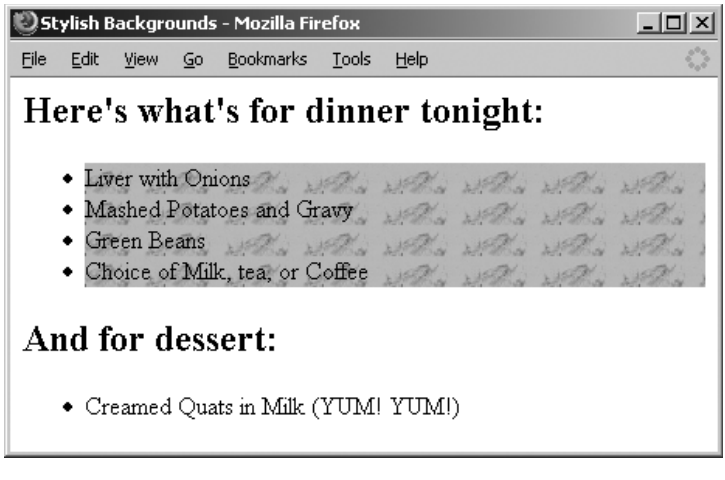

Рис. 8.6. *Изображение служит фоном элемента*

8.4.5.3. Свойство background-attachment

Если вы используете в качестве фона элемента изображение, свойство `background-attachment` позволит управлять тем, каким образом это изображение прикреплено к окну броузера. При принятом по умолчанию значении `scroll` броузер перемещает фоновое изображение вместе с элементами, когда пользователь прокручивает документ. Значение `fixed` делает изображение неподвижным.

8.4.5.4. Свойство background-position

По умолчанию поддерживающие стили броузеры выводят фоновое изображение, начиная с левого верхнего угла выделенного пространства. С помощью свойства `background-position` можно сместить начальное положение фонового изображения. Смещение может быть абсолютным (длина) и относительным (процентное значение или ключевое слово). В результате потенциально «обрезанное» изображение будет заполнять прямоугольную область от этой сдвинутой начальной точки.

Свойству `background-position` можно приписать одно или два значения. Если вы используете одиночное значение, оно применяется к положениям по горизонтали и по вертикали. В случае двух значений первое – это горизонтальное смещение, второе – вертикальное. Значения длины (с указанием единиц измерения; см. раздел 8.4.1.2) задают абсолютное расстояние от верхнего левого угла элемента, за которым вы помещаете фоновое изображение. Отрицательные значения фактически обрезают соответственно верхнюю и левую часть изображения в пределах выделенного окошка, подобно тому как изображение, не помещающееся в окно броузера, обрезается снизу и справа.

Например:

```
table {background-image: url(backgrounds/marble.gif);
       background-position: 10px 20px}
```

сдвигает мраморный фон на 10 пикселов вправо и на 20 пикселов вниз от верхнего левого угла всякого элемента `<table>` в документе.

Процентные значения устроены чуть хитрее, но они легче в использовании. Если считать, что расстояния от левого до правого края элемента, так же как и от верхнего до нижнего, равны 100 процентам, то центр элемента расположен в точке 50%, 50%. Подобным образом сдвиг на треть элемента вправо и на две трети вниз обозначается 33%, 66%. Итак, чтобы сдвинуть фон обеденного меню из нашего примера к центру пространства, в которое отображается содержимое элемента, напишем:[1]

```
background-position: 50%
```

[1] Интересно, что данное свойство работало в соответствии с описанием в версиях 4 и 5 броузера Internet Explorer, но в версии 6 (как и в других популярных броузерах) это уже не наблюдается. Смещение действует, только если вы установите свойство `background-repeat`.

Но зачем использовать числа, когда достаточно одного слова? Вы можете применять ключевые слова `left`, `center` и `right` так же, как и `top`, `center` и `bottom` вместо `0%`, `50%` и `100%` соответственно. Чтобы центрировать изображение по отношению к области, занятой содержимым тега, напишите:

```
background-position: center
```

Длины и процентные значения[1] можно использовать совместно, так что:

```
background-position: 1cm 50%
```

смещает изображение на один сантиметр вправо и к середине области тега по вертикали.

Обратите внимание, что при относительном смещении изображение сдвигается относительно содержимого тега, когда пользователь меняет размер окна броузера, потому что место, отведенное содержимому, тоже меняет свой размер. Если же вы указываете абсолютное смещение, изображение остается на том же месте по отношению к содержимому элемента.

Вероятно, вы ожидаете, что при размножаемом фоне (это свойство установлено по умолчанию, см. разд. 8.4.5.5) изображение расширяется вниз и вправо. Это не совсем так. В современных броузерах изображение «обтекает» элемент, заполняя отведенное ему пространство на экране. Взгляните внимательно на рис. 8.7 и сравните эффект размноже-

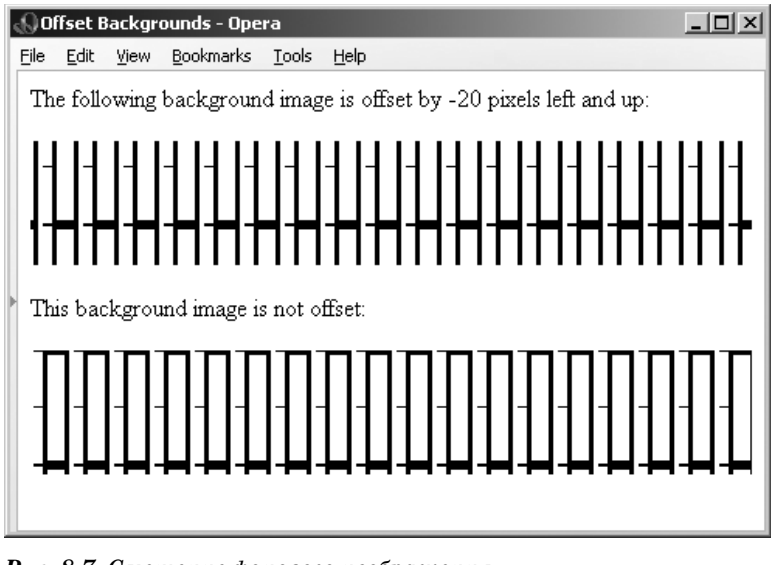

Рис. 8.7. *Смещение фонового изображения*

[1] Это так, если броузер поддерживает единицы измерения. Пока Internet Explorer и Netscape работают со скудным набором единиц длины – пикселы и проценты.

ния для смещенного и несмещенного изображения, выводимого в следующих фрагментах описания стиля:

```
<style type=css/text>
<!--
pre {background-image: url(backgrounds/vert.gif)}
pre.offset {background-image: url(backgrounds/vert.gif); background-
position: -20px -20px}
-->
</style>
...
The following background image is offset by -20 pixels left and up:
<pre class=offset>

</pre>
<p>
This background image is not offset:
<pre>

</pre>
```

8.4.5.5. Свойство background-repeat

Обычно броузер, чтобы заполнить выделенное пространство, размножает фоновое изображение и по горизонтали, и по вертикали. Используйте свойство `background-repeat`, чтобы уклониться от этого принятого по умолчанию образа действий броузера. Чтобы изображение повторялось только по горизонтали, но не по вертикали, применяйте значение `repeat-x`. Для повторения только по вертикали используйте `repeat-y`. Чтобы прекратить укладывание плитки вообще, употребляйте `no-repeat`.

Часто это свойство используют, чтобы поместить на странице водяной знак или логотип, обойдясь без многократного повторения картинки. К примеру:

```
body {background-image: url(backgrounds/watermark.gif);
      background-position: center center;
      background-repeat: no-repeat
     }
```

приведет к тому, что центр страницы будет помечен фоновым изображением.

Популярным приемом является создание вертикальной полосы вдоль правого края страницы:

```
body {background-image: url(backgrounds/ribbon.gif);
      background-position: top right;
      background-repeat: repeat-y
     }
```

8.4.5.6. Свойство background

Многие свойства фона так же не просто описывать и впоследствии читать, как характеристики шрифтов. Поэтому, подобно `font`, в CSS2 предусмотрено общее свойство `background`.

Свойство `background` принимает значения, пригодные для характеристик `background-color`, `background-image`, `background-attachment`, `background-repeat` и `background-position`, в любом порядке.

Если не указаны значения для некоторых свойств, последние задаются по умолчанию. Таким образом:

```
background: red
```

устанавливает значение свойства `background-color` в красный, а для всех остальных задает принятые по умолчанию. Более сложный пример:

```
background: url(backgrounds/marble.gif) blue repeat-y fixed center
```

устанавливает одновременно свойства фоновых цвета и изображения, что в результате дает мраморный рисунок на синем фоне (синее будет просвечивать сквозь все прозрачные участки изображения). Изображение повторяется вертикально, начиная с центра области воспроизведения содержимого элемента, и остается неподвижным, когда пользователь прокручивает документ. Отметьте, что мы включили ровно одно значение для положения (`center`), и броузер использует его для вертикального и для горизонтального смещений.

8.4.5.7. Свойство color

Свойство `color` устанавливает цвет переднего плана для содержимого тега – цвет текста, например. Значением данного свойства может быть название цвета, шестнадцатеричный RGB-триплет или десятичный RGB-триплет, как это описано в разделе 8.4.1.5. Таким образом, ниже перечислены допустимые объявления этого свойства:

```
color: mauve
color: #ff7bd5
color: rgb(255, 125, 213)
color: rgb(100%, 49%, 84%)
```

Чаще всего вы будете использовать `color` для определения цвета текста, но можете также при его помощи модифицировать и нетекстовое содержимое тега. К примеру, следующее объявление создает зеленую горизонтальную линейку:

```
hr {color: green}
```

Если цвет элемента не определен, то окраску он унаследует от родителя.

8.4.6. Свойства текста

Каскадные таблицы стилей проводят различие между свойствами шрифтов, которые управляют размером, стилями и видом текста, и свойствами текста, которые управляют тем, как текст выравнивается и представляется пользователю.

8.4.6.1. Свойство letter-spacing

Свойство `letter-spacing` вставляет дополнительные промежутки между символами в тексте при его отображении броузером. Значение свойства может быть либо длиной, либо принятым по умолчанию ключевым словом `normal`, указывающим, что броузер должен использовать обычное расстояние между буквами. К примеру:

```
blockquote {letter-spacing: 2px}
```

вставляет дополнительные два пиксела между соседними буквами в теге `<blockquote>`. А так выглядят 5 пикселов между символами (рис. 8.8).

Все популярные броузеры поддерживают это свойство.

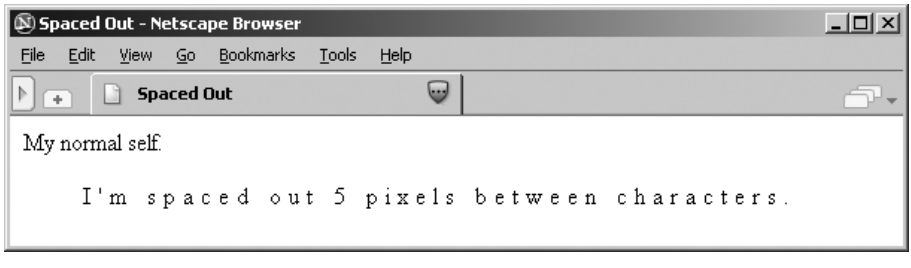

***Рис. 8.8.** Свойство letter-spacing позволяет вам растянуть текст*

8.4.6.2. Свойство line-height

Используйте свойство `line-height` (высота строки) для определения минимального расстояния между строками в тексте, являющемся содержимым тега. Когда броузер выводит текст с одинарным интервалом, верх следующей строки всего лишь на несколько пунктов ниже предыдущей. Увеличивая высоту строк, вы увеличиваете расстояние между ними.

Значением свойства `line-height` может быть абсолютная или относительная длина, процентное значение, масштабный множитель или ключевое слово `normal`. Например:

```
p {line-height: 14pt}
p {line-height: 120%}
p {line-height: 2.0}
```

Первый пример устанавливает расстояние между базовыми линиями последовательных строк текста в точности равным 14 пунктам. Вто-

рой указывает, что высота строки должна составлять 120% от размера шрифта. Последний пример использует масштабный множитель, чтобы установить высоту строки в два раза большей, чем размер шрифта, выводя текст с двумя интервалами. Принятое по умолчанию значение `normal` обычно эквивалентно масштабному множителю 1,0 или 1,2.

Имейте в виду, что абсолютные и процентные значения свойства `line-height` применяются для вычисления высоты строки, основываясь на значении `font-size`. Вычисленное значение этого свойства наследуется дочерними элементами. Последующие изменения свойства `font-size` в родительском и дочернем элементах уже не изменяют вычисленную высоту строки.

Масштабные множители, напротив, откладывают вычисление высоты строки до момента действительного отображения текста. Следовательно, варьирование `font-size` локально воздействует на `line-height`. Вообще, лучше использовать для свойства `line-height` масштабные множители, чтобы высота строки автоматически менялась вместе с размером шрифта.

Рассматриваемое обычно отдельно от характеристик шрифтов свойство `line-height` (точнее, его значение) может служить одним из элементов спецификации свойства `font`. [свойство font, 8.4.3.8]

8.4.6.3. Свойство text-align

Выравнивание текста относительно краев страницы – это элементарная функция практически всех текстовых процессоров. Свойство `text-align` предоставляет такую возможность для любого HTML-тега блочного характера. (Разработчики стандартов из W3C предпочитают, чтобы вы пользовались скорее `text-align` из CSS2, чем явным атрибутом `align` для выравнивания блочных тегов, таких как `<p>` и `<div>`). Используйте одно из четырех значений: `left`, `right`, `center` или `justify`. По умолчанию принимается, конечно, `left`.[1] К примеру:

```
div {text-align: right}
```

указывает поддерживающему стили броузеру выровнять весь текст в теге `<div>` по правому краю. Значение `justify` предлагает броузеру выровнять текст по левому и по правому краю, раздвигая символы и слова по мере необходимости.

Все популярные броузеры в настоящее время поддерживают выравнивание `left`, `right` и `center`, но не `justify`.

8.4.6.4. Свойство text-decoration

Свойство `text-decoration` создает украшения текста, некоторые из которых можно получить с применением тегов физической разметки.

[1] Для локалей, где строки читаются слева направо. В «правосторонних» локалях умолчание – `right`.

Значением этого свойства служат одно или несколько ключевых слов из множества: `underline` (подчеркнутый), `overline` (надчеркнутый), `line-through` (зачеркнутый) и `blink` (мерцающий). Значение `none` принимается по умолчанию и говорит поддерживающему стили броузеру, что текст следует представлять обычным образом.

Свойство `text-decoration` хорошо подходит для оформления гиперссылок. К примеру:

```
a:visited a:link a:active {text-decoration: underline overline}
```

помещает линии над и под гиперссылками в документе.

Это свойство текста не наследуется и не имеет влияния на нетекстовые элементы.

Интересно, что все популярные броузеры поддерживают свойство `text-decoration`, но только Internet Explorer обладает достаточно хорошим вкусом, чтобы не поддерживать значение `blink`.

8.4.6.5. Свойство text-indent

Существует традиция, в наши дни уже не такая общепринятая, начинать первую строку абзаца с отступом.[1] Некоторые же текстовые блоки, такие как определения, обычно выносят первую строку влево, создавая то, что называется *обратным* или *висячим отступом*.

Свойство `text-indent` стандарта CSS2 позволяет начинать абзац с красной строки или с висячего отступа, управляя величиной отступа первой строки в блоке. Применяйте с этим свойством длины и процентные значения. Отрицательные значения создают висячий отступ. Процентные – влекут за собой вычисление величины отступов в долях ширины родительского элемента. По умолчанию значение свойства – ноль.

Чтобы выводить все абзацы в документе с красной строки, напишите, скажем:

```
p {text-indent: 3em}
```

Использованная единица длины `em` масштабирует величину отступа в зависимости от размера шрифта на разных броузерах.

С висячими отступами дело обстоит хитрее, поскольку вам приходится следить за рамками элементов. Отрицательный отступ не смещает левого края текста, он просто сдвигает влево первую строку элемента, возможно, на поле, рамку или подложку родительского элемента. По этой причине висячие отступы работают в соответствии с ожиданиями, если вы сдвигаете левый край элемента вправо на такое же расстояние, на какое строка выступает влево, или на большее. В частности:

```
p.wrong {text-indent: -3em}
p.hang  {text-indent: -3em; margin-left: 3em}
```

[1] Как, очевидно, не делается в этой книге.

```
p.large {text-indent: -3em; margin-left: 6em}
```

создает три стиля абзацев. Первый формирует висячий отступ, выходящий на левое поле. Второй вводит традиционный висячий отступ, а третий создает абзац, тело которого сдвинуто вправо больше, чем первая строка выступает влево. Посмотрите на все три стиля в действии (рис. 8.9).

Все популярные броузеры поддерживают свойство `text-indent`.

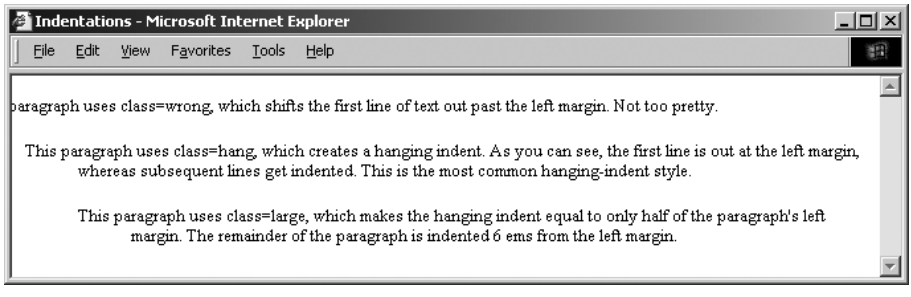

***Рис. 8.9.** Эффекты установок свойств text-indent и margin-left*

8.4.6.6. Свойство text-shadow

Свойство `text-shadow` позволяет придать вашему тексту трехмерный вид при помощи выдержавших проверку временем теней. Значениями свойства являются обязательный сдвиг и необязательные цвет и радиус растушевки. У свойства может быть несколько наборов значений, разделенных запятыми, что позволяет получить последовательность теней, причем каждый очередной набор значений находится на слое выше предыдущего, но всегда остается под оригинальным текстом.

Обязательно указываемый при определении свойства сдвиг составляется из двух длин. Первая отвечает горизонтальному смещению, вторая определяет смещение по вертикали. Положительные значения размещают тень вправо и вниз на указанное расстояние от текста. Отрицательные значения соответствуют тени, протянувшейся вверх и влево.

Необязательный параметр радиус тоже принимает значения длины и характеризует границы растушевки. Влияние этого параметра зависит от выводящего устройства. Еще один параметр тени – это цвет. Он, разумеется, может быть задан названием или RGB-триплетом и обозначает цвет тени. Если цвет не указан, `text-shadow` использует значение свойства `color`. К примеру:

```
h1 {text-shadow: 10px 10px 2px yellow}
p:first-letter {text-shadow: -5px -5px purple, 10px 10px orange}
```

Первый пример создает желтую тень от заголовков первого уровня в документе. Она протягивается на 10 пикселов вправо и 10 пикселов вниз и растушевана на 2 пиксела. Второй пример отбрасывает две тени

от первой буквы в каждом абзаце. Фиолетовая тень размещается вверх и влево от буквы на пять пикселов. Другая тень, как и в первом примере, только оранжевая, протягивается на 10 пикселов вниз и 10 пикселов вправо от первой буквы каждого абзаца.

К сожалению, мы не можем продемонстрировать вам ни один из этих эффектов, так как популярные броузеры не поддерживают данное свойство.

8.4.6.7. Свойство text-transform

Свойство `text-transform` позволяет автоматически преобразовать часть или весь текст документа в отображаемый заглавными или строчными буквами. Допустимыми значениями являются `capitalize`, `uppercase`, `lowercase`, `none`.

`Capitalize` выводит каждую первую букву любого слова в верхнем регистре, даже если текст исходного документа набран строчными буквами. Значения `uppercase` и `lowercase` выводят весь текст в верхнем и соответственно нижнем регистрах. Значение `none`, разумеется, отменяет всякое преобразование.

К примеру:

```
h1 {text-transform: uppercase}
```

выводит все буквы в заголовках уровня 1, предположительно названиях, в верхнем регистре, тогда как:

```
h2 {text-transform: capitalize}
```

гарантирует, что каждое слово в заголовках уровня 2 начинается с прописной буквы, что приемлемо, например, в заголовках разделов.

Отметьте, что тогда как `uppercase` и `lowercase` влияют на весь текст, `capitalize` действует только на первую букву каждого слова. Следовательно, попытка преобразовать слово «htML» при помощи `capitalize` приведет к «HtML». Свойство `text-transform` поддерживают все популярные броузеры.

8.4.6.8. Свойство vertical-align

Свойство `vertical-align` управляет положением элемента по отношению к содержащей его строке. Его допустимыми значениями являются:

`baseline`

Выравнивает базовую линию элемента с базовой линией охватывающего элемента.

`middle`

Выравнивает середину элемента с серединой охватывающего элемента. (Обычно середина – это верх строчной буквы).

`sub`

Элемент выводится как подстрочный индекс.

`super`

Элемент выводится как надстрочный индекс.

`text-top`

Выравнивает верхний край элемента с верхним краем шрифта охватывающего элемента.

`bottom`

Выравнивает нижний край элемента по нижнему выносному элементу текущей строки.

Кроме того, процентное значение указывает положение относительно базовой линии текущей строки, так что значение `50%` приподнимает элемент на половину высоты строки над базовой линией. Значение `100%` помещает элемент ровно на высоту строки ниже базовой линии.

Все популярные броузеры сходятся в том, как следует располагать изображения относительно линии текста для значений `baseline` (которое устанавливается по умолчанию и соответствует отсутствию выравнивания по вертикали), `middle` (но не `center`), `super` (но не `sub`), `text-top`, `text-bottom`, `top` (то же самое, что `text-top`; но не `bottom`), а также для положительных и отрицательных значений относительного смещения. На рис. 8.10 показано, как Internet Explorer интерпретирует различные значения свойства `vertical-align`.

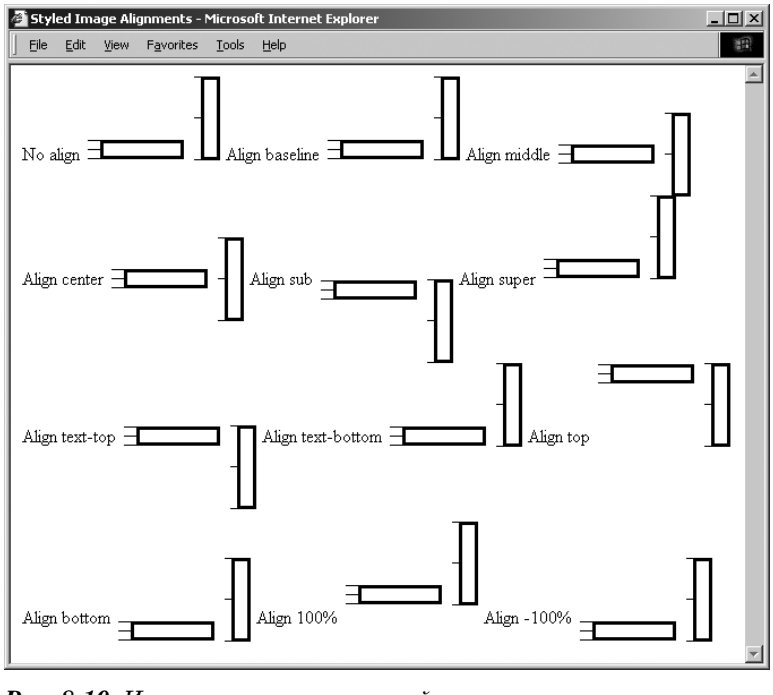

Рис. 8.10. *Интерпретация значений вертикального выравнивания броузером Internet Explorer*

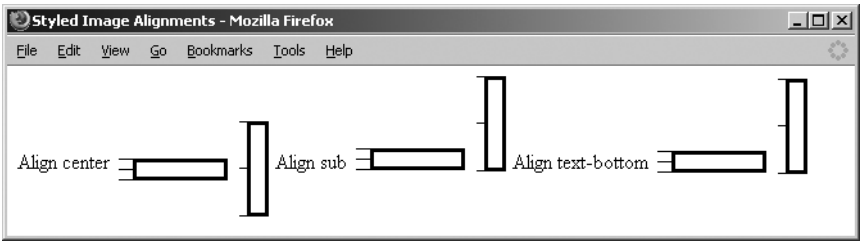

Рис. 8.11. *Интерпретация некоторых значений вертикального выравнивания броузером Firefox*

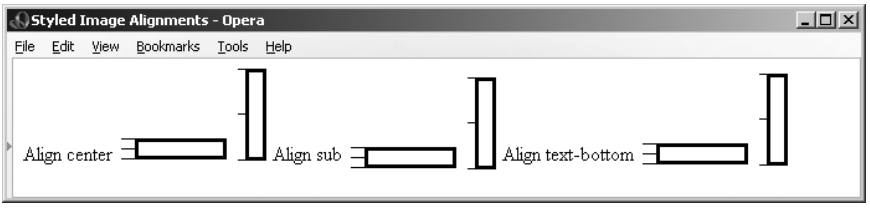

Рис. 8.12. *Интерпретация некоторых значений вертикального выравнивания броузером Opera*

А теперь обсудим различия. Броузер Firefox интерпретирует значение `center` так же, как Internet Explorer, причем иначе, чем значение `middle` (рис. 8.10), Netscape считает значение `center` идентичным значению `middle`, а Opera вообще не распознает его. Что касается значения `sub`, Netscape соглашается с Firefox и выравнивает нижнюю сторону изображения по крайней строчке нижнего выносного элемента. При этом Opera помещает изображение явно ниже базовой линии, но не так низко, как Internet Explorer, у которого изображение оказывается чуть выше следующей строчки текста.

В отношении значения `bottom` броузер Opera соглашается с Internet Explorer, выравнивая нижнюю границу изображения по линии, проведенной чуть выше следующей строчки текста. Firefox и Netscape располагают нижнюю сторону изображения по крайней строчке нижнего выносного элемента. Запутались? Возможно, рис. 8.11–8.13 помогут вам в этом разобраться, особенно если сравнивать их с рис. 8.10.

8.4.6.9. Свойство word-spacing

Используйте свойство `word-spacing`, чтобы увеличить расстояние между словами в теге. Вы можете присвоить `word-spacing` либо длину, либо ключевое слово `normal`, возвращающее обычное расстояние между словами. К примеру:

```
h3 {word-spacing: 25px}
```

помещает дополнительные 25 пикселов между словами в теге `<h3>`. Все популярные броузеры поддерживают свойство `word-spacing`.

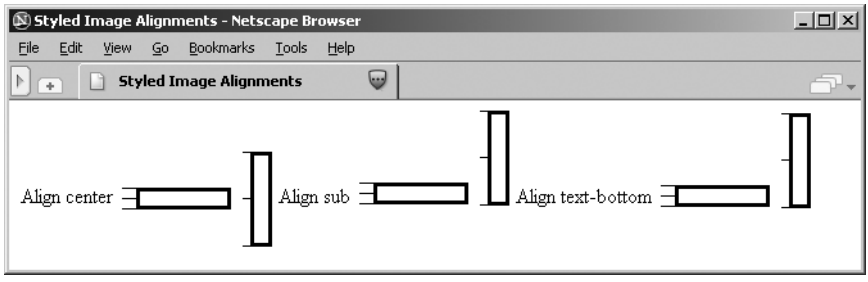

Рис. 8.13. *Интерпретация некоторых значений вертикального выравнивания броузером Netscape*

8.4.7. Свойства контейнеров

Модель CSS2 предполагает, что HTML- и XHTML-элементы всегда помещаются в прямоугольные контейнеры. Используя свойства, определяемые в этом разделе, можно управлять размером, положением и внешним видом контейнеров, содержащих элементы в документах.

8.4.7.1. CSS2-модель форматирования

Каждый элемент документа может быть помещен в прямоугольный контейнер. Авторы CSS2 называют этот контейнер «ядром» (*core content area*) и помещают его в еще три контейнера: подложку, рамку и поля. Рисунок 8.14 показывает эти контейнеры и определяет некоторые полезные понятия.

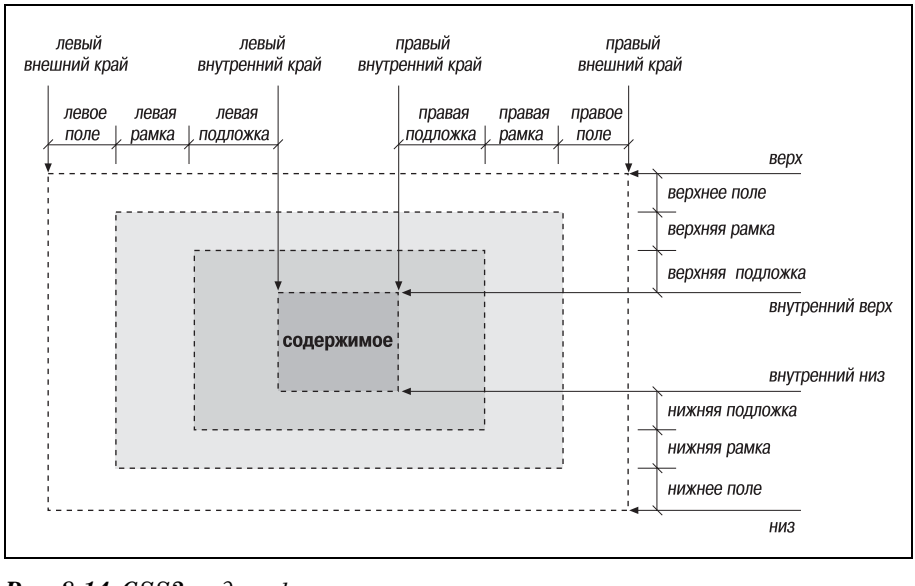

Рис. 8.14. *CSS2-модель форматирования и ее термины*

Верхний (top), нижний (bottom), левый внешний (left outer edge) и правый внешний (right outer edge) края ограничивают ядро элемента и связанные с ним области подложки (padding), рамки (border) и полей (margin space). Внутренний верхний (inner top), внутренний нижний (inner bottom), левый внутренний (left inner edge) и правый внутренний (right inner edge) края определяют границы ядра. Дополнительное пространство вокруг элемента – это область между внутренними и внешними краями, включающая подложку, рамку и поля. Броузер может опускать любое из этих дополнительных пространств, и для многих элементов внешние и внутренние края совпадают.

Когда элементы являются смежными по вертикали, нижнее поле верхнего элемента и верхнее поле нижнего элемента перекрываются так, что полное расстояние между смежными элементами будет равно величине большего из полей. К примеру, пусть у одного абзаца нижнее поле равно одному дюйму, а верхнее поле следующего абзаца равно половине дюйма, тогда между абзацами будет расстояние, равное ширине большего (однодюймового) поля. Этот метод известен как *схлопывание полей* и обычно ведет улучшению внешнего вида документов.

Поля смежных по горизонтали элементов не перекрываются. Вместо этого CSS2-модель складывает смежные горизонтальные поля. К примеру, если у абзаца левое поле один дюйм и он соседствует с изображением, правое поле которого 0,5 дюйма, тогда полное расстояние между ними будет равно полутора дюймам. Это правило относится и к вложенным элементам, так что абзац, содержащийся в разделе, будет иметь левое поле, равное сумме левых полей раздела и абзаца.

Как видно из рис. 8.14, полная ширина элемента равна сумме семи величин: левого и правого полей, левой и правой рамки, левой и правой подложки и ядра элемента. Сумма этих семи величин должна равняться полной ширине элемента. Из этих семи величин только трем может быть присвоено значение `auto`: ширине элемента и его левому и правому полю. Это означает, что броузер может вычислять значение для этих свойств. При необходимости броузер следует перечисленным ниже правилам:

- Если ни для одного из этих свойств не установлено значение `auto` и полная ширина оказывается меньше ширины родительского элемента, свойству `margin-right` назначается `auto` и правое поле увеличивается, чтобы сделать полную ширину элемента равной ширине родительского.
- Если `auto` назначено ровно одному свойству, именно оно будет сделано достаточно большим, чтобы полная ширина стала равной ширине родительского элемента.
- Если `width`, `margin-left` и `margin-right` установлены в `auto`, поддерживающий стандарт CSS2 броузер определит левое и правое поля равными нулю и назначит `width` достаточно большим, чтобы сделать полную ширину равной ширине родительского элемента.

- Если левое и правое поля установлены в `auto`, они всегда будут равными, чтобы центрировать элемент относительно родительского.

Существуют специальные правила для плавающих элементов. Поля плавающего элемента (такого как изображение с выравниванием `align=left`), если только они не отрицательные, не схлопываются с полями охватывающего или предыдущего элемента. Вот как следующий отрывок HTML может быть выведен (рис. 8.15):

```
<body>
<p>
<img align=left src="pics/img.gif">
Some sample text...
</body>
```

Броузер двигает изображение вместе с его полями влево и вверх, насколько это возможно без перекрытия левого и верхнего полей абзаца или тела документа. Левые поля абзаца и содержащего его тела документа складываются, тогда как верхние схлопываются.

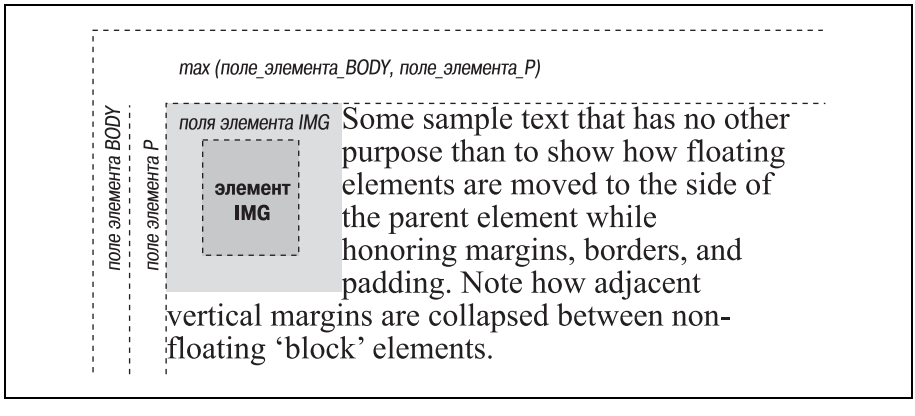

***Рис. 8.15.** Поведение полей плавающего элемента*

8.4.7.2. Свойства рамки

Окружающая элемент рамка имеет цвет, толщину и стиль. Можно использовать различные свойства рамки для управления этими тремя ее характеристиками на каждой из четырех сторон элемента. При желании можно применить краткие варианты свойств, позволяющие с легкостью определить для всей рамки один цвет, одну толщину и один стиль. Свойства рамки не наследуются; их нужно явно устанавливать для каждого элемента с рамкой.

8.4.7.3. Свойство border-color

Используйте `border-color` для задания цвета рамки. Если он не определен, броузер рисует рамку, применяя значение свойства `color` для элемента. Свойство `border-color` принимает от одного до четырех цветовых

значений. Количество значений определяет способ их применения к рамке (табл. 8.1). Если установить только одно значение, все четыре стороны рамки будут одного цвета. Два значения приводят к тому, что верх и низ рамки окрашиваются в первый цвет, а левая и правая сторона рамки – во второй. В случае трех значений первое будет цветом верха рамки, второе – левой и правой сторон, третье – нижней стороны рамки. Четыре значения определяют цвет каждой из сторон: верха, правой стороны, низа, левой стороны.

Таблица 8.1. Порядок, в котором действуют множественные значения свойств рамки, полей и подложки

Число значений	Влияние на стороны рамки, полей и подложки
1	Все элементы принимают одно и то же значение
2	Первое значение устанавливает *верх* (*top*) и *низ* (*bottom*); второе значение устанавливает *левую* (*left*) и *правую* (*right*) стороны
3	Первое значение устанавливает *верх*; второе устанавливает *левую* и *правую*; третье значение устанавливает *низ*
4	Первое значение устанавливает *верх*; второе – *правую*; третье – *низ*; четвертое – *левую*

8.4.7.4. Свойство border-width

Свойство `border-width` управляет толщиной рамки. Как и свойство `border-color`, оно принимает от одного до четырех значений, которые применяются к разным сторонам рамки в том же порядке (табл. 8.1).

Помимо длин для указания толщины рамки можно использовать три ключевых слова: `thin` (тонкая), `medium` (средняя) и `thick` (толстая). По умолчанию, если толщина не установлена явно, принимается значение `medium`. Типичные спецификации толщины рамки:

```
border: 1px
border: thin thick medium
border: thick 2mm
```

Первый пример устанавливает для всех четырех сторон рамки толщину в 1 пиксел. Второй пример делает верх рамки тонким, правую и левую стороны толстыми, а низ – средним. Последний пример делает верх и низ рамки толстыми, а для правой и левой сторон устанавливает толщину в 2 миллиметра.

Если неудобно определять толщину всех четырех сторон рамки при одном подходе, то можно использовать индивидуальные свойства: `border-top-width`, `border-bottom-width`, `border-left-width` и `border-right-width` для определения толщины каждой стороны в отдельности. Любое свойство принимает ровно одно значение, по умолчанию устанавливается `medium`.

Все популярные броузеры поддерживают это свойство.

8.4.7.5. Свойство border-style

В соответствии с моделью CSS2 можно применить к рамкам HTML-элементов множество украшений.

Свойство `border-style` принимает следующие значения: `none` (по умолчанию), `dotted`, `dashed`, `double`, `groove`, `ridge`, `inset` и `outset`. Броузер, разбирающийся в стилях рамок, применяет от одного до четырех значений этого свойства к каждой из сторон рамки в том порядке, который описан в табл. 8.1.

Броузер рисует `dotted` (точечная), `dashed` (штриховая), `solid` (сплошная) и `double` (двойная) рамки в виде плоских линий поверх фона тега. Значения `groove`, `ridge`, `inset` и `outset` создают «трехмерные» рамки: `groove` – углубленную линию (паз), `ridge` – выпуклую линию (ребро), `inset` утапливает содержимое тега и `outset` приподнимает его над поверхностью документа. Воздействие трехмерной природы последних четырех стилей на фоновое изображение не определено и оставлено на усмотрение броузера. Netscape поддерживает трехмерные эффекты.

Все популярные броузеры поддерживают стили рамок. Пример см. на рис. 8.16.

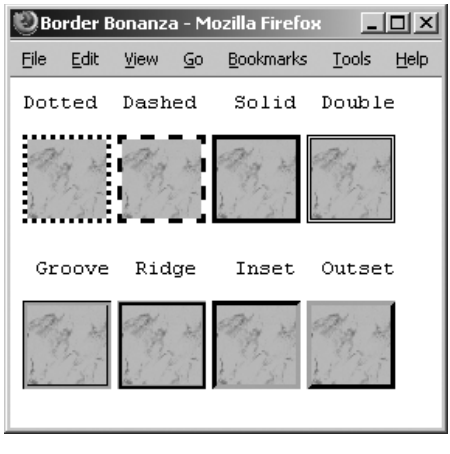

Рис. 8.16. *Свойство border-style аккуратно обрамляет изображение*

8.4.7.6. Краткие описания рамок

Поскольку описание сложной рамки может наскучить, стандарт CSS2 предоставляет пять свойств, которые принимают любое или все сразу из значений толщины, цвета и стиля для одной или всех сторон рамки. Свойства `border-top`, `border-bottom`, `border-left` и `border-right` действуют на соответствующие стороны рамки. Всеобъемлющее свойство `border` управляет всеми четырьмя сторонами рамки одновременно. Например:

```
border-top: thick solid blue
border-left: 1ex inset
border-bottom: blue dashed
border: red double 2px
```

Первое свойство выполняет верх рамки толстой сплошной синей линией. Второе – устанавливает для левой стороны эффект утопленности на толщину, равную высоте строчной буквы в шрифте, оставляя цвет рамки тем же, что и цвет элемента. Третье свойство проводит по низу элемента синюю пунктирную линию, толщина которой равна `medium` – значению по умолчанию. Наконец, последнее свойство выводит все четыре стороны рамки красными двойными линиями толщиной в 2 пиксела.

По поводу последнего свойства `border` надо сказать две вещи. Во-первых, нельзя применять к нему множественные значения для избирательного воздействия на определенные стороны рамки, как это делается с использованием `border-color`, `border-width` и `border-style`. Свойство `border` всегда действует на все четыре стороны рамки.

Во-вторых, секундное размышление подскажет, что создать двойную рамку толщиной всего два пиксела просто невозможно. В подобных случаях броузер имеет право подобрать толщину, чтобы вывести рамку надлежащим образом.

Обычно мы рассматриваем рамки, окружающие блочные элементы, такие как изображения, таблицы и абзацы, однако можно обводить и внедренные теги. Это позволяет обрамлять слово или фразу из потока текста. Реализация рамок вокруг внедренных тегов, которые охватывают несколько строк, не определена стандартом и оставлена на усмотрение броузера.

Все популярные броузеры поддерживают стили рамок.

8.4.7.7. Свойство clear

Как и родственный ему атрибут `clear` для тега `<br>`, свойство `clear` говорит броузеру, помещать ли содержимое тега рядом с «плавающим» элементом или на первой строке под ним. Текст обтекает плавающие элементы, такие как изображения и таблицы с атрибутами `align=left` и `align=right`, и HTML/XHTML-элементы, у которых свойство `float` отлично от `none`. [тег
, 4.6.1] [свойство float, 8.4.7.9]

Значением свойства `clear` может быть `none`, `left`, `right` или `both`. Значение `none`, принятое по умолчанию, подразумевает, что броузер действует обычным образом и размещает содержимое тега рядом с плавающими элементами по обе стороны, если там есть место. Значение `left` запрещает располагать содержимое слева от плавающего элемента, `right` – справа, `both` не допускает размещения содержимого тега рядом с плавающим элементом вообще.

Результат применения этого свойства совпадает с тем, который достигается, когда перед тегом помещается тег `<br>` с атрибутом `clear`. Таким образом:

```
h1 {clear: left}
```

имеет тот же результат, как если бы перед каждым тегом <h1> стояло <br clear=left>.

8.4.7.8. Свойство clip

Как правило, содержимое элемента видно целиком в области, отведенной под элемент на экране. Свойство clip определяет окошко внутри этой области, позволяя вам скрыть части элемента и привлечь внимание к какому-либо участку или аспекту содержимого.

У свойства clip значением по умолчанию является auto, согласно которому окошко просмотра совпадает с прямоугольником, включающим в себя элемент. В качестве альтернативы можно указать геометрическую фигуру, определяющую окошко просмотра в пределах области на экране, отведенной под элемент. В настоящее время единственной фигурой, поддерживаемой стандартом CSS2[1], является прямоугольник, задаваемый с помощью ключевого слова rect. Например:

```
p {overflow : hidden;
   clip : rect(15px, -10px, 5px, 10px) }
```

Эти четыре значения определяют верхнюю, правую, нижнюю и левую стороны усекающего прямоугольника. Каждое значение является смещением относительно сторон прямоугольника, включающего в себя элемент. В нашем примере верхняя сторона усекающего прямоугольника расположена на 15 пикселов ниже верхней стороны прямоугольника, содержащего элемент, правая сторона – на 10 пикселов правее соответствующей правой стороны, нижняя – на 5 пикселов выше нижней стороны, а левая – на 10 пикселов правее левой стороны.

Обратите внимание, что свойство clip имеет силу только в том случае, когда свойство overflow данного элемента имеет значение, отличное от visible. Когда overflow равно visible, усечение не происходит и свойство clip игнорируется.

Популярные броузеры пока еще не поддерживают свойство clip.

8.4.7.9. Свойство float

Свойство float описывает область отображения тега в качестве плавающего элемента и заставляет текст обтекать его определенным образом. В целом, оно аналогично атрибуту align для таблиц и изображений, но может применяться к любому элементу, включая текст. [атрибут align, 10.2.1.1]

Свойство float допускает одно из трех значений: left, right или none, принимаемое по умолчанию. Использование none выключает свойство

[1] Предполагается, что в следующих версиях стандарта появятся и другие геометрические фигуры.

float. Другие значения действуют, как их аналоги для атрибута align, приказывая броузеру поместить содержимое тега у соответствующего края потока и позволить другому содержимому обтекать его.

Таким образом, броузер разместит специфицированное при помощи float:left содержимое (включая все поля, заполнители и рамки) тега слева от левой границы потока текста, и последующее содержимое будет обтекать его справа, вниз и под содержимым тега. Пара float:right помещает содержимое тега напротив правого края потока, и другое содержимое будет обтекать его слева вниз и под содержимое тега.

Однако свойство float, чаще всего используемое с таблицами и изображениями, можно с успехом применять и к текстовым элементам. Например, следующее описание создает «встроенный» заголовок, который обтекается потоком текста, как показано на рис. 8.17.

```
h2 {float: left;
text-align: center;
margin-right: 10px }
```

Все популярные броузеры поддерживают это свойство.

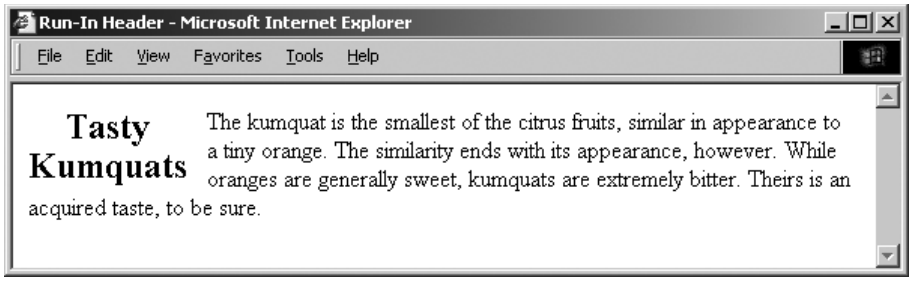

Рис. 8.17. *Применяйте свойство float для блоков текста, чтобы создавать встроенные заголовки*

8.4.7.10. Свойство height

Как вы могли заподозрить, свойство height управляет высотой области, в которую выводится содержимое тега. Чаще всего оно используется с изображениями и таблицами, но с тем же успехом может применяться для управления высотой других элементов.

Значением свойства height может быть или длина, или ключевое слово auto, принимаемое по умолчанию. Использование auto подразумевает, что тег обладает изначальной высотой, которую и следует употребить при его отображении. В противном случае для высоты необходимо установить желаемое значение. Если используются абсолютные единицы, высота устанавливается равной указанному значению. Например:

```
img {height: 100px}
```

говорит броузеру, что нужно так масштабировать изображение из тега <img>, чтобы оно стало высотой 100 пикселов. Если применяется относительное значение, то базовая величина, от которой оно отсчитывается, зависит от броузера и от тега.

При масштабировании элемента в соответствии с заданной высотой пропорции объекта могут быть защищены установкой свойства width этого тега в auto. Таким образом,

```
img {height: 100px; width: auto}
```

гарантирует, что изображение всегда будет 100 пикселов высотой с пропорционально измененной шириной. [свойство width, 8.4.7.16]

Если вы хотите задать высоту элемента в форме диапазона, а не одного конкретного значения, пользуйтесь свойствами min-height и max-height. Они принимают те же значения, что и height, и устанавливают допустимый диапазон для высоты элемента. Броузер затем отрегулирует высоту элемента в соответствии с указанным диапазоном.

Все популярные броузеры полностью поддерживают свойство height, но ни один из них не поддерживает свойства min-height и max-height.

8.4.7.11. Свойства полей

Подобно характеристикам границ, различные свойства полей позволяют управлять пространством вокруг элемента сразу за его рамкой (рис. 8.14). Поля всегда прозрачны, позволяя цвету фона или фоновому изображению объемлющего элемента просвечивать сквозь них. В результате вы можете определять только размеры полей. У них нет ни цвета, ни стиля отображения.

Все свойства margin-left, margin-right, margin-top и margin-bottom принимают либо процентные значения, либо длины, обозначающие величину резервируемого пространства. Помимо того, ключевое слово auto указывает поддерживающему стили броузеру вернуться к тем полям, которые он обычно располагает вокруг элемента. Процентные значения понимаются в отношении к ширине охватывающего элемента. По умолчанию полей нет.

Вот их допустимые установки:

```
body {margin-left: 1in; margin-top: 0.5in; margin-right: 1in}
p {margin-left: -0.5cm}
img {margin-left: 10%}
```

Первый пример создает поля шириной в один дюйм слева и справа во всем документе и полудюймовое поле сверху. Второй пример выносит тег <p> на полсантиметра на левое поле. Последний – создает слева от тега <img> поле шириной в десять процентов от ширины родительского элемента.

Для краткости записи можно употребить свойство margin, чтобы определить все четыре поля, применяя от одного до четырех значений, дей-

ствующих в ранее упомянутом порядке (табл. 8.1). С использованием этой формы поля тега <body> из предыдущего примера могут быть описаны также так:

```
body {margin: 0.5in 1in}
```

Свойства margin-left и margin-right взаимодействуют со свойством width при определении полной ширины элемента, как это описано в разделе 8.4.7.1.

Все популярные броузеры поддерживают свойства, определяющие поля, и их значения.

8.4.7.12. Свойства подложки

Подобно свойствам полей, различные свойства подложки (padding) позволяют управлять присущим ей пространством вокруг элемента, пространством между ядром элемента и его рамкой (рис. 8.14). Подложка всегда отображается с использованием цвета фона или фонового изображения. В результате определять можно только ее размеры; цвета и стиля у нее нет.

Свойства padding-left, padding-right, padding-top, padding-bottom принимают в качестве значений либо длины, либо процентные значения, указывая пространство, которое поддерживающий стили броузер должен оставить вокруг элемента. Процентные значения берутся в отношении к ширине охватывающего элемента. Подложка не может быть отрицательной. По умолчанию подложка отсутствует.

Вот допустимые установки для ее размеров:

```
p {padding-left: 0.5cm}
img {padding-left: 10%}
```

Первый пример создает полусантиметровый отступ между содержимым тега <p> и его левой рамкой. Второй пример формирует подложку слева от тега <img> шириной в десять процентов от ширины родительского элемента.

Так же как и «скорые на руку» margin и border, можно использовать свойство padding для определения размеров подложки со всех четырех сторон, приписывая ему от одного до четырех значений, которые действуют, как описано в табл. 8.1. Свойство padding не поддерживает Internet Explorer, и в этом он расходится с остальными броузерами.

8.4.7.13. Свойство overflow

Свойство overflow сообщает броузеру, как поступить с содержимым, переполняющим область вывода элемента. Значением этого свойства по умолчанию является visible. Оно предписывает броузеру выводить все содержимое, даже если оно не умещается в область, отведенную под элемент.

Не заботясь о последствиях, авторы веб-документов обычно хотят выводить все содержимое. Однако в редких случаях элементы могут накладываться друг на друга, создавая довольно некрасивую картину. Чтобы предотвратить подобные неприятности, установите свойство overflow в значение hidden, scroll и auto.

Значение hidden заставляет броузер прятать все содержимое, выходящее за дозволенные границы, то есть делает эту часть содержимого невидимой для пользователя. Значение scroll приводит к созданию полос прокрутки, позволяющих пользователю просматривать скрытое содержимое. Правда, полосы прокрутки появятся, даже если содержимое не переполняет область элемента.

Добавление постоянных полос прокрутки гарантирует, что они не будут то исчезать, то появляться по мере того, как изменяется размер элемента в динамическом документе. Отрицательной стороной такого решения является визуальная «замусоренность» страницы, порождаемая полосами прокрутки, которые отвлекают читателя от документа. Чтобы избежать всего этого, установите свойство overflow в значение auto. Когда задано значение auto, полосы прокрутки появляются только в случае необходимости. Если содержимое элемента изменяется так, что он больше не усекается, полосы прокрутки удаляются.

8.4.7.14. Свойство position

По умолчанию броузер выводит элементы документа на дисплей последовательно, единым потоком. Вы можете изменить это стандартное поведение броузера с помощью свойства position, комбинируя его со свойствами top, bottom, left и right.

Если значением свойства position является static, действуют обычные правила позиционирования и компоновки, принятые в HTML/XHTML, причем левая и верхняя сторона прямоугольника, содержащего элемент, определяются броузером. Чтобы сместить элемент в потоке вывода, установите свойство position в значение relative. В этом случае свойства top, bottom, left и right служат для вычисления положения прямоугольника по отношению к его нормальному положению в потоке вывода. Последующие элементы не затрагиваются этим изменением позиции и располагаются в потоке вывода так, словно сдвинутый элемент остается на месте.

Если свойство position равно absolute, данный элемент удаляется из потока, а последующие сдвигаются вверх. Затем позиция элемента вычисляется с помощью свойств top, bottom, left и right. Этот способ позволяет помещать элемент в фиксированное положение относительно охватывающего элемента и передвигать его в соответствии с передвижением охватывающего элемента.

Наконец, установка свойства position в значение fixed позиционирует элемент относительно окна или страницы, в которую он выводится.

Как и при абсолютном позиционировании (значение absolute), элемент удаляется из потока, содержащего его, а остальные элементы соответственно сдвигаются. Свойства top, bottom, left и right служат для установки позиции элемента относительно окна или страницы, в которую он выводится. Обратите внимание, что в устройствах последовательного вывода информации (то есть с прокруткой) элемент выводится один раз в нужной позиции. В устройствах печати элемент выводится на каждой странице в соответствующей позиции. Вы можете использовать позиционирование типа fixed для размещения верхних и нижних колонтитулов в окне броузера и на каждой напечатанной странице.

Каждое из свойств top, bottom, left и right принимает в качестве значения длину или процентное соотношение. Когда свойство position имеет значение relative, процентное соотношение рассчитывается на основе размера прямоугольника элемента. Когда свойство position установлено в значение absolute или fixed, процентное соотношение основывается на размере прямоугольника, содержащего данный элемент. Если заданы длины, они обозначают смещения от соответствующих сторон прямоугольника элемента. Например, чтобы расположить элемент так, что его нижняя сторона будет на 1 см выше нижней стороны окна броузера (или каждой напечатанной страницы), вы должны установить свойство position в значение fixed, а свойство bottom – в значение 1cm.

8.4.7.15. Свойство visibility

Свойство visibility определяет, будет ли видно на экране содержимое элемента. Место под элемент все равно отводится, и этот факт влияет на компоновку документа, однако содержимое самого элемента можно сделать невидимым.

Значением этого свойства по умолчанию является visible. Оно заставляет броузер выводить содержимое. Если установить значение hidden, то содержимое элемента станет невидимым, но ограничивающий прямоугольник элемента удален не будет, это повлияет на компоновку документа. Заметим, что вы можете убрать из документа и содержимое и ограничивающий прямоугольник элемента, установив свойство display в значение none.

Данное свойство активно применяется в динамических документах, когда изменение его значения для конкретного элемента приводит в исчезновению содержимого без переформатирования документа.

Если это свойство применить к строкам и столбцам таблицы, то у него появится дополнительное значение collapse. Использованное в этом контексте значение collapse приводит к удалению соответствующих строк и столбцов из таблицы, но не вызывает ее переформатирования. В динамических документах такой прием позволяет вам удалять элементы из таблицы, не переформатируя ее. За пределами таблицы значение collapse эквивалентно значению hidden.

8.4.7.16. Свойство width

Свойство width, подобно своему товарищу height, управляет шириной связанного с ним тега. Точнее говоря, определяет ширину ядра элемента, как это показано на рис. 8.8. Чаще всего оно используется с изображениями и таблицами, но вполне допустимо применять его и для управления шириной других элементов.

Значениями свойства width могут быть длина, процентное отношение или ключевое слово auto. Значение auto принимается по умолчанию и подразумевает, что тег имеет начальную ширину, которая и должна употребляться при отображении тега. Если применяется длина (с указанием единиц измерения), то ширина элемента будет равна ей. Использование процентных значений влечет за собой вычисление ширины в процентах от ширины охватывающего элемента. Например:

```
img {width: 100px}
```

демонстрирует изображение, на которое ссылается тег <img>, масштабированным так, чтобы его ширина была равна 100 пикселам.

При масштабировании элементов до определенной ширины можно гарантировать сохранение пропорций объекта, устанавливая свойство height тега в auto. Так:

```
img {width: 100px; height: auto}
```

обеспечивает, что ширина изображений будет равна 100 пикселам, а высота выдержана согласно пропорциям. [свойство height, 8.4.7.10]

Если вы хотите задать ширину элемента в форме диапазона, а не одного конкретного значения, пользуйтесь свойствами min-width и max-width. Они принимают те же значения, что и width, и устанавливают допустимый диапазон для ширины элемента. Броузер затем отрегулирует ширину элемента в соответствии с указанным диапазоном.

Свойство width взаимодействует со свойствами margin-left и margin-right при определении полной ширины элемента так, как это описано в разделе 8.4.7.1.

8.4.7.17. Свойство z-index

В дополнение к x- и y-позициям элемента в окне броузера или на распечатанной странице у каждого элемента есть позиция по оси z. Элементы с более высоким значением z находятся «ближе» к зрителю и заслоняют элементы, расположенные за ними.

Позиция z не является абсолютной в документе. Значения z считаются относительными к содержащему элементу. Например, два элемента в одном документе могут быть расположены так, что один окажется поверх другого. У первого можно установить значение z, равное 1, а значением второго может быть 2. Все содержимое второго элемента будет выводиться поверх содержимого первого. Если

в первом элементе содержатся элементы с z-позициями 3 или 4, они будут выводиться в рамках первого и не «выскочат» перед вторым элементом .

Вы контролируете z-позицию элемента с помощью свойства z-index. Его значением может быть целое положительное число, устанавливающее z-позицию элемента в отношении элемента, который его содержит. Свойство z-index позволяет вам динамически регулировать z-позицию элемента, делая его видимым, когда необходимо. Кроме того, вы можете, например, располагать текстовый элемент перед изображением, чтобы отметить интересные места.

8.4.8. Свойства списков

Стандарт CSS2 позволяет управлять выводом элементов списков, упорядоченных и неупорядоченных.

Броузеры форматируют элементы списков так же, как любые другие блочные элементы, за тем исключением, что в этом блоке есть некий маркер, предшествующий содержимому. В случае неупорядоченных списков маркер – это изображение или специальный символ, в нумерованных списках маркером является числовое или буквенное обозначение. В CSS2 свойства списков позволяют управлять внешним видом и положением связанных с элементом маркеров.

8.4.8.1. Свойство list-style-image

Свойство list-style-image определяет изображение, которым броузер будет отмечать элемент списка. Значением этого свойства служит URL файла с изображением или ключевое слово none, принимаемое по умолчанию.

Такое изображение будет маркером, предваряющим содержимое пунктов списка. Если оно доступно, броузер отобразит его вместо любого другого иначе определенного маркера. Если изображение недоступно или пользователь отключил его автоматическую загрузку, будет применяться маркер, определенный свойством list-style-type (раздел 8.4.8.3).

Авторы могут употреблять свойство list-style-image для определения типовых маркеров своих неупорядоченных списков. Хотя допустимо использование любых изображений в качестве маркеров, мы рекомендуем применять GIF- или JPEG-изображения небольшого размера, чтобы гарантировать привлекательный вид ваших списков.

К примеру, если вы хотите употребить в качестве маркера изображение из файла *mybullet.gif*, хранящегося на вашем сервере, можно написать:

```
li {list-style-image: url(pics/mybullet.gif); list-style-type: square}
```

В этом случае изображение будет использовано, если броузер успешно загрузит *mybullet.gif*. Иначе применяется традиционный квадратный маркер.

***Рис. 8.18.** Свойство list-style-image позволяет задать собственный маркер для списка*

Все популярные броузеры поддерживают свойство `list-style-image` (см. рис. 8.18).

8.4.8.2. Свойство list-style-position

Существует два способа разместить маркер, ассоциированный с элементом списка, – внутри связанного с элементом блока или за его пределами. В соответствии с этим свойство `list-style-position` допускает одно из двух значений: `inside` или `outside`.

По умолчанию принимается `outside`, означающее, что маркер будет висеть слева от элемента примерно так:

```
• Это маркированный список
  с маркером «outside»
```

Значение `inside` «вдвигает» маркер в элемент списка, который обтекает его примерно так же, как текст обтекает плавающее изображение:

```
• Это маркированный список
с маркером «inside»
```

Отметьте, что вторая строка текста выводится без отступа и выравнивается по левому краю маркера.

Текущие версии популярных броузеров полностью поддерживают свойство `list-style-position`.

8.4.8.3. Свойство list-style-type

Свойство list-style-type выполняет двойную задачу в том смысле, что определяет способ, которым выводятся элементы как упорядоченных, так и неупорядоченных списков броузерами, поддерживающими стили. Оно действует на элементы списков так же, как атрибут type. [атрибут type, 6.7.2.4]

Свойство list-style-type применяется к элементам неупорядоченного списка и принимает одно из четырех значений: disc, circle, square или none. Броузер помечает элемент списка одним из этих маркеров. По умолчанию для элементов самого внешнего списка принимается значение disc; броузеры изменяют применяемое по умолчанию значение в зависимости от глубины вложения списка.

Используемое с элементами упорядоченного списка свойство list-style-type принимает одно из шести значений: decimal, lower-roman, upper-roman, lower-alpha, upper-alpha или none. Эти значения форматируют номера элементов как десятичные значения, строчные римские цифры, заглавные римские цифры, строчные буквы и заглавные буквы соответственно. Большинство броузеров по умолчанию применят десятичную нумерацию.

Популярные броузеры поддерживают свойство list-style-type и свойство list-style, описываемое в следующем разделе.

8.4.8.4. Свойство list-style

Свойство list-style – это компактная версия для всех других характеристик стиля списков. Оно принимает любые значения, допустимые для свойств list-style-type, list-style-position и list-style-image, которые можно указывать в произвольном порядке, применяя значения, подходящие для типа списка, на который они действуют. Вот допустимые объявления свойства list-style:

```
li {list-style: disc}
li {list-style: lower-roman inside}
li {list-style:
url(http://www.kumquat.com/images/tiny-quat.gif) square}
```

Первый пример создает элементы списка, применяющие кружок в качестве маркера. Второй предписывает использовать строчные римские цифры, утопленные в блок элемента, для нумерации пунктов упорядоченного списка. В последнем примере в качестве маркера будет применяться квадратик, если указанное изображение окажется недоступным.

8.4.8.5. Эффективное использование свойств списков

Хотя можно приложить свойства списков к любому элементу, они подействуют только на те элементы, для которых свойство display имеет значение list-item. Обычно единственным тегом, удовлетворяющим такому условию, является тег <li>.

Однако это не должно оттолкнуть вас от использования подобных свойств где-либо еще, особенно с тегами <ul> и <ol>. Поскольку они наследуются от родительских элементов, в которых были установлены, модифицируя свойства списков для тегов <ol> и <ul>, вы модифицируете их и для тегов <li>, содержащихся в соответствующем списке. Это значительно облегчает создание списков с определенным внешним видом.

К примеру, допустим, что вы хотите создать стиль списка, применяющий строчные римские цифры. Один способ состоит в том, чтобы определить класс для тега <li> с подходящим значением свойства list-style-type:

```
li.roman {list-style-type: lower-roman}
```

Внутри списка вам придется специфицировать использование этого класса для каждого элемента:

```
<ol>
  <li class=roman>Пункт один</li>
  <li class=roman>Пункт два</li>
  <li class=roman>И так далее</li>
</ol>
```

Необходимость повторять имя класса вызывает скуку и чревато ошибками. Лучшим решением будет определить класс для тега <ol>:

```
ol.roman {list-style-type: lower-roman}
```

Каждый тег <li> списка унаследует это свойство и будет помечен строчными римскими цифрами:

```
<ol class=roman>
  <li>Пункт один</li>
  <li>Пункт два</li>
  <li>И так далее</li>
</ol>
```

Это гораздо легче понять, да и проще управиться. Если когда-нибудь в будущем понадобится изменить стиль нумерации, вам придется устанавливать другие значения свойства только в теге <ol> вместо того, чтобы находить и править каждое вхождение тега <li> в список.

Более того, работа с такими характеристиками может оказать и глобальное воздействие. Установив значения свойств списков для тега <body>, вы измените внешний вид всех списков в документе. Установка этих свойств в теге повлияет на все списки раздела.

8.4.9. Табличные свойства

Как правило, броузеры HTML/XHTML-документов выводят табличное содержимое, ориентируясь на те же свойства, которые определяют обычное содержимое документа. Однако при выводе таблиц возникают особые обстоятельства. Чтобы предоставить авторам более широ-

кие возможности управления таблицами, стандарт CSS2 определяет несколько табличных свойств. Популярные броузеры их пока не поддерживают.

8.4.9.1. Свойства border-collapse, border-spacing и empty-cells properties

Существуют две разных точки зрения на границы ячеек таблицы. Согласно первой, каждая ячейка является независимой сущностью со своими собственными границами. Вторая точка зрения сводится к тому, что соседние ячейки имеют общую границу и изменение границы одной ячейки неизбежно влияет на границу другой.

Чтобы предоставить авторам максимальный контроль, CSS2 определяет свойство `border-collapse`, позволяющее каждому составителю документов выбрать модель по вкусу. Значением этого свойства по умолчанию является `collapse`, которое заставляет соседние ячейки делить друг с другом общую границу. Альтернативным значением свойства `border-collapse` является `separate`. Оно расширяет таблицу так, что границы каждой ячейки выводятся отдельно и отчетливо.

Если вы выберете модель `separate`, вы сможете пользоваться свойством `border-spacing` и устанавливать интервалы между границами соседних ячеек. По умолчанию этот интервал равен 0, то есть границы ячеек соприкасаются. Впрочем, в броузерах могут быть приняты иные умолчания. Увеличивая это значение, вы заставляете броузер раздвигать границы ячеек так, что станет проступать фоновый цвет или изображение. Если вы указываете только одно значение свойства `border-spacing`, оно задаст интервал как для вертикальных, так и для горизонтальных границ. Когда указаны два значения, первое определяет горизонтальный, а второе – вертикальный интервал.

В рамках модели `separate` вы можете управлять изображением границ вокруг ячеек. По умолчанию границы рисуются у каждой ячейки таблицы, даже если она пустая. Вы можете изменить этот стиль, переключив значение свойства `empty-cells` с `show` (по умолчанию) на `hide`. Когда установлено это значение, в пустых ячейках виден фон таблицы. Если целая строка таблицы состоит из пустых значений, броузер удаляет ее из таблицы.

8.4.9.2. Свойство caption-side

Используйте свойство `caption-side` только с элементом `<caption>`. Оно принимает значения `top` (по умолчанию), `bottom`, `left` и `right` и сообщает, как расположить заголовок по отношению к таблице. Свойство `caption-side` предоставляет более удачный способ размещения заголовка таблицы, чем атрибут `align` тега `<caption>`, который зависит от броузера и объявлен нежелательным.

Все популярные броузеры, кроме Internet Explorer, поддерживают свойство `caption-side`.

8.4.9.3. Свойство speak-header

Броузеры с функцией аудиовоспроизведения предлагают пользователям целый ряд способов навигации, основанных на озвучивании содержимого таблицы. Простейший подход заключается в том, чтобы броузер читал содержимое таблицы по порядку, сверху вниз и справа налево. Более сложный аудиоброузер организует содержимое таблицы согласно заголовкам и читает информацию более понятным образом. Чтобы избежать путаницы, броузер должен в любом случае как-то сообщать пользователю, содержимое какой ячейки он читает.

Свойство `speak-header` определяет два способа, которыми броузер может идентифицировать ячейку или группу ячеек в таблице. Если указано `once` (значение по умолчанию), броузер читает содержимое ячейки-заголовка только один раз перед тем, как перейдет к чтению содержимого каждой ассоциированной ячейки с данными. Таким образом, пользователь, перемещающийся по строке ячеек, услышит заголовок строки и заголовок столбца первой ячейки в строке, а потом будет слышать только заголовки столбцов последующих ячеек в этой строке.

Если вы установите свойство `speak-header` в значение `always`, броузер будет предварять чтение содержимого каждой ячейки чтением ассоциированного заголовка. Это бывает полезно при чтении таблиц со сложной структурой или когда заголовок таблицы помогает понять ее содержимое, особенно если оно состоит из одних цифр.

Обратите внимание, что заголовки озвучиваются, только когда броузеру известно, как ассоциированы ячейки-заголовки с ячейками с данными. Сознательные авторы документов всегда указывают атрибут `header` для ячеек таблицы, чтобы определить связь между заголовочными ячейками и ячейками, содержащими данные.

8.4.9.4. Свойство table-layout

Компоновка таблицы всегда представляет сложную задачу для броузера. Чтобы вывести красивую таблицу, броузер должен найти самую широкую ячейку в каждом столбце и отрегулировать ширину столбца, а затем и всей таблицы. В случае большой таблицы вывод документа может сильно замедлиться, если броузер совершает несколько проходов по таблице, пытаясь настроить все размеры.

Чтобы облегчить броузеру эту задачу, пользуйтесь свойством `table-layout`. Если вы установите его в значение `fixed`, броузер определит ширину каждого столбца по ширине соответствующей ячейки в первой строке таблицы. Если вы явно укажете ширину столбцов, установка свойства `table-layout` в значение `fixed` даже ускорит процесс вывода таблицы, что, конечно, в интересах пользователей, просматривающих ваш документ.

По умолчанию свойство `table-layout` имеет значение `auto`, что заставляет броузер использовать многопроходный алгоритм, требующий

много времени, даже если вы указываете ширину столбцов таблицы. Если содержимое вашей таблицы непостоянно и вы не можете определить ширину столбцов заранее, оставьте свойству `table-layout` значение `auto`. Если же вы в состоянии зафиксировать ширину столбцов и содержимое таблицы это позволяет, задайте свойству `table-layout` значение `fixed`.

8.4.10. Классификационные свойства

Классификационные свойства – самые фундаментальные из стилевых свойств CSS2. Они не управляют напрямую тем, как поддерживающие стили броузеры выводят элементы HTML или XHTML. Вместо этого они сообщают броузеру, к какому разряду отнести встреченный тег и его содержимое и как с ним следует обращаться.

По большей части нет необходимости устанавливать значения этих свойств, если только не стремиться к какому-либо специальному эффекту.

8.4.10.1. Свойство display

Каждый элемент в HTML- или XHTML-документе может быть отнесен с точки зрения его свойств при отображении к одному из следующих разделов: блочные элементы, встроенные элементы, элементы списков. Блочные элементы, такие как заголовки, абзацы, таблицы и списки, форматируются как особые блоки текста, отделенные от предыдущих и последующих блоков. Встроенные (inline) элементы, такие как теги физической/логической разметки и якоря гиперссылок, выводятся в текущей строке текста внутри содержащего их блока. Элементы списка, более всего это относится к тегу `<li>`, выводятся как блочный элемент с *маркером*.

Свойство `display` позволит определить тип отображения элемента как один из набора: `block`, `inline`, `list-item` или `none`. Первые три устанавливают принадлежность объекта к соответствующему разряду. Значение `none` «выключает» элемент, превращая его и все его потомство в неотображаемые компоненты.

Можно представить себе все беды, которые легко натворить, переключая классификацию элементов, заставляя абзацы отображаться как элементы списка и превращая гиперссылки в блочные элементы. В действительности все это пустое ребячество, и мы не рекомендуем изменять классификацию элемента без достаточно серьезных оснований.

Все популярные броузеры поддерживают это свойство. Internet Explorer принимает только значения `block` и `none`.

8.4.10.2. Свойство white-space

Свойство `white-space` определяет, как поддерживающий стили броузер обращается с «пропусками» (пробелами, знаками табуляции и возвра-

та каретки) в блочных тегах. Ключевое слово normal (принимаемое как значение по умолчанию) схлопывает пропуски так, что один или несколько идущих подряд пробелов, возвратов каретки и знаков табуляции трактуются как один пробел между словами. Значение pre имитирует тег <pre>, в котором броузер сохраняет и отображает все пробелы, возвраты каретки и табуляции. И наконец, значение nowrap приказывает броузеру игнорировать возврат каретки и не вставлять автоматически переходов на новую строку. Все разрывы строк должны осуществляться явным применением тега
.

Как и свойство display, свойство white-space легче использовать во вред, чем во благо. Не изменяйте способа, которым элементы обрабатывают «пропуски», без серьезных на то оснований.

Internet Explorer поддерживает только значение nowrap, а другие популярные броузеры поддерживают оба значения (pre и nowrap) свойства white-space.

8.4.11. Свойства генерируемого содержимого

Идея генерируемого содержимого не нова для HTML. Даже первые версии броузеров автоматически добавляли маркеры или номера к неупорядоченным или упорядоченным спискам для улучшения внешнего вида. Впрочем, такой функциональности явно недостаточно, и авторам HTML-документов всегда хотелось, чтобы в HTML были средства для генерирования содержимого. Наконец, появился стандарт CSS2 и предоставил авторам возможность генерировать произвольное содержимое, нумерованные списки и все виды содержимого, основанного на элементах.

В основе модели генерируемого содержимого, принятой в CSS2, лежат свойства content и quotes, а также псевдоэлементы :before и :after. Свойства служат для определения необходимого вам содержимого, а псевдоэлементы – для позиционирования содержимого относительно элементов в вашем документе.

8.4.11.1. Псевдоэлементы :before и :after

Мы представили вам псевдоэлементы ранее в этой главе, и вы даже видели их в действии (см. рис. 8.2 и 8.3). Псевдоэлементы :before и :after работают аналогичным образом. Добавьте любой из них к селектору «стиль-элемент», чтобы выбрать и специфицировать смысл и свойства генерируемого содержимого. Вообще говоря, любое содержимое, сгенерированное в пределах этих псевдоэлементов, наследует атрибуты внешности элемента-родителя, так что шрифты, размер и цвета, применяемые к элементу, применяются и к его сгенерированному содержимому. Например:

```
p.note { color : blue }
p.note:before { content : "Note: " }
```

Этот пример стиля вставляет слово «Note:» перед каждым элементом <p class=note>. Вставляемый текст выводится синим цветом, как и весь абзац. Если заменить это стиль на такой:

```
p.note:before { content : "Note: "; color : red }
```

то вставляемый текст будет красным, а остальной абзац – синим.

Любое сгенерированное содержимое, будь то до или после элемента, включается в прямоугольник элемента и влияет на его форматирование, расположение, размер и компоновку.

8.4.11.2. Свойство content

Свойство content принимает широкий набор значений, от простых строк до автоматических счетчиков. Вы можете включить в состав свойства content любое количество этих значений, разделив их запятыми. Броузер конкатенирует их, формируя одно значение, которое затем вставляет в документ.

Простейшим значением свойства content является строка в кавычках. В такую строку нельзя включать разметку HTML или XHTML. Для генерирования специального текста следует пользоваться escape-последовательностями (например, \A генерирует перевод строки).

В стандарте CSS2 escape-последовательности аналогичны символьным заменам, принятым в HTML/XHTML. В то время как символьные замены начинаются с амперсанда (&), за которым следует имя или десятичное значение символа (в последнем случае должен присутствовать символ "#"), строковое значение свойства content в CSS2 содержит символы, шестнадцатеричные эквиваленты которых предваряются обратной косой чертой. Таким образом, escape-последовательность \A эквивалентна символьной замене &#010 и, если свериться с приложением F, обозначает символ перевода строки.

Свойство content в качестве значений может принимать URL-адреса. Выраженный в соответствии со стилем CSS, а не в духе HTML, адрес URL может указывать на любой объект, приемлемый для броузера, включая текст, изображения и звуковые файлы. Например, чтобы поместить декоративный символ рядом с каждым уравнением в документе, вы можете написать:

```
p.equation:before { content : url("http://www.kumquat.com/
                                        /decorative-symbol.jpg") }
```

Помните, что объект не должен содержать разметку HTML/XHTML, потому что броузер вставит содержимое объекта в документ дословно.

Свойство content поддерживает автоматическое генерирование контекстуально корректных кавычек в соответствии с локалью. Вы вставляете их с помощью ключевых слов open-quote и close-quote. Они обеспечивают появление кавычек и соответственное увеличение или уменьшение значения счетчика вложенных кавычек. Вы можете управлять внешним видом кавычек с помощью свойства quotes, описанного да-

лее. Кроме того, вы можете использовать ключевые слова no-open-quote и no-close-quote для изменения значения счетчика кавычек без постановки самих кавычек.

Положительной чертой свойства content является его способность заставить броузер вывести значение любого атрибута ассоциированного элемента. Значение attr имеет единственный параметр, задающий имя атрибута. Если этот атрибут определен для данного элемента, его значение вставляется в документ. Чтобы вывести URL-адрес изображения вслед за изображением, напишите:

```
img:after { content : "("attr(src) ") " }
```

Если атрибут для элемента не определен, то никакое содержимое не вставляется, хотя другие значения свойства content (например, скобки, определенные в предыдущем примере) будут вставлены.

Одной из самых мощных функциональных возможностей свойства content является его способность создавать нумерованные списки, которая обсуждается далее, в разд. 8.4.11.4.

Все популярные броузеры поддерживают псевдоэлементы :before и :after, но Internet Explorer не поддерживает свойство content.

8.4.11.3. Определение кавычек

Кавычки вставляются с помощью значений open-quote и close-quote свойства content, а их внешний вид определяется свойством quotes.

В качестве значения этого свойства следует указать одну или несколько пар строк. Первая пара определяет открывающие и закрывающие кавычки самого внешнего уровня цитирования в вашем документе. Следующая пара определяет следующий уровень и т. д. Если уровень цитирования превысит указанное количество параметров, броузер выберет пару для самого внешнего уровня. Заметим, что, хотя во многих языках для обозначения цитат используются односимвольные открывающие и закрывающие кавычки, вы можете для этой цели указать строки произвольной длины.

Возможно определение альтернативных кавычек в соответствии с языком документа. Вы можете воспользоваться псевдоэлементом :lang и ассоциировать различные свойства quotes с разными языками:

```
q:lang(en) { quotes : "‘" "’" "‘" "’" }
q:lang(no) { quotes : "&#171;" "&#187;" "<" ">" }
```

обеспечивает правильную постановку кавычек в документах на английском и норвежском языках.

8.4.11.4. Создание счетчиков

Вы можете создавать простейшие нумерованные списки в HTML и XHTML с помощью элемента <ol>. Однако чуть более сложные спи-

ски, например, вложенные нумерованные, создать в языках разметки уже нельзя. Зато стандарт CSS2 вводит понятие счетчика, значение которого может быть установлено и изменено, когда броузер выводит ваш документ. Вставьте значение счетчика с помощью специальных функций, известных свойству `content`, и измените формат и внешний вид счетчика с помощью других CSS2-свойств.

Согласно стандарту CSS2 каждый счетчик обладает именем. Чтобы создать счетчик, просто упомяните его имя в свойстве `counter-reset` или `counter-increment`, ассоциированном с соответствующим элементом. Если экземпляр счетчика с таким именем еще не существует на текущем уровне вложенности документа, броузер, поддерживающий стандарт CSS2, создаст его автоматически. С этого момента вы можете сбрасывать и устанавливать значение счетчика, как вам угодно. Например, предположим, что вы хотите обозначать элементом `<h1>` названия глав, а элементом `<h2>` – названия разделов. Главы и разделы пронумерованы, причем в каждой главе нумерация разделов начинается заново. Желаемый результат достигается таким способом:

```
h1:before { counter-increment : chapter; counter-reset : section }
h2:before { counter-increment : section }
```

Когда броузер, поддерживающий стандарт CSS2, встречает первый элемент `<h1>` в документе, он создает счетчики `chapter` и `section` и сбрасывает их значения в 0. Одновременно (и при каждом следующем обнаружении этого элемента) такой броузер активизирует свойство `counter-increment`, чтобы увеличить счетчик `chapter` на единицу и получить номер главы 1, 2, 3 и т. д. По мере того, как внутри главы встречаются элементы `<h2>`, счетчик `section` увеличивается в соответствии со стилевым правилом «h2», которое предписывает последовательную нумерацию разделов. Обратите внимание, что счетчик `section` сбрасывается в стилевом правиле «h1» и нумерация разделов внутри каждой главы начинается заново.[1]

Как свойство `counter-reset`, так и `counter-increment` принимают списки имен счетчиков, позволяя вам сбрасывать или увеличивать значения нескольких счетчиков в одном свойстве. Кроме того, вы можете указать число после имени счетчика, в результате чего свойство `counter-reset` проинициализирует счетчик указанным значением, а свойство `counter-increment` увеличит счетчик на указанное числовое значение. Отрицательные числа допускаются, так что вы можете нумеровать пункты в обратном порядке, если пожелаете.

Например, если мы хотим, чтобы документ начинался с главы 7, а номера разделов увеличивались на 2, мы можем переписать предыдущий пример таким образом:

---

[1] Заметим, что броузер не выводит счетчики на экран, если вы явно не попросите его об этом. См. раздел «Применение счетчиков в документах».

```
body { counter-reset : chapter 6 }
h1:before { counter-increment : chapter; counter-reset : section }
h2:before { counter-increment : section 2 }
```

Обратите внимание, что мы создали счетчик chapter как можно раньше и проинициализировали его значением, на единицу меньшим, чем нужное нам начальное значение. Когда броузер встретит первый элемент <h1>, он создаст счетчик chapter, установит его в значение 6, а затем увеличит это значение на единицу.

Областью действия имени счетчика является уровень вложенности, на котором оно было определено. Совсем не обязательно, чтобы эта область охватывала весь документ. Если вы используете такое же имя счетчика в элементе-потоке, броузер создаст на этом уровне новый счетчик. В нашем примере все элементы <h1> и <h2> находятся на одном уровне вложенности, так что единственные экземпляры счетчиков chapter и section обслуживают весь уровень. Если бы вы вложили в этот элемент тег , который бы, в свою очередь, содержал элементы <h1> и <h2>, то на этом новом уровне появились бы новые экземпляры обоих счетчиков.

Такое поведение распространяется на вложенные нумерованные списки. Если вы ассоциируете счетчик с элементом <li>, а затем вложите друг в друга несколько нумерованных списков, то на каждом уровне будет создан отдельный экземпляр счетчика со своей нумерационной последовательностью.

8.4.11.5. Применение счетчиков в документах

В счетчиках мало пользы, если вы не показываете их значения в своих документах. Вывод значений не происходит автоматически. Чтобы в документе появился счетчик, вызовите функции counter() и counters(), которые являются специальными значениями свойства content.

Для функции-значения counter() вы должны указать в скобках имя счетчика и при желании спецификацию формата. Броузер выведет значение указанного счетчика в составе генерируемого содержимого в заданном формате. Форматом может быть любой списковый формат, принимаемый свойством list-style-type (см. разд. 8.4.8.3 ранее в этой главе).

Чтобы вывести на экран номера глав и разделов в нашем примере, расширим стилевые правила для элементов <h1> и <h2>:

```
h1:before { counter-increment : chapter;
   counter-reset : section;
   content : "Chapter " counter(chapter) ": " }
h2:before { counter-increment : section;
   content : "Section " counter(section) ": "}
```

Теперь, когда броузер, поддерживающий CSS2, встретит в документе элемент:

```
<h1>Kumquat Growers</h1>
```

он передаст его так, как показано на рис. 8.19. Чтобы нумеровать главы и разделы римскими цифрами, нам придется изменить свойства таким образом:

```
h1:before { counter-increment : chapter;
   counter-reset : section;
   content : "Chapter " counter(chapter, upper-roman) ": " }
h2:before { counter-increment : section;
   content : "Section " counter(section, lower-roman) ": "}
```

Значение `counter()` определяет значение счетчика на текущем уровне вложенности. Чтобы получить значения одноименных счетчиков на всех уровнях вложенности, воспользуйтесь функцией-значением `counters()`. В скобках укажите имя счетчика и разделительную строку. Броузер выведет эту строку между списками значений счетчика. Кроме того, вы можете указать формат нумерации, отличный от принятого по умолчанию.

Значение `counters()` особенно полезно при создании вложенных нумерованных списков. Рассмотрим следующие свойства:

```
ol { counter-reset: item }
li:before { counter-increment: item ;
   content: counters(item, ".") }
```

Если вы вложите друг в друга несколько элементов `<ol>`, каждый элемент `<li>` будет содержать значения счетчиков, разделенные точками. Это соответствует распространенному способу нумерации разделов[1] 1,

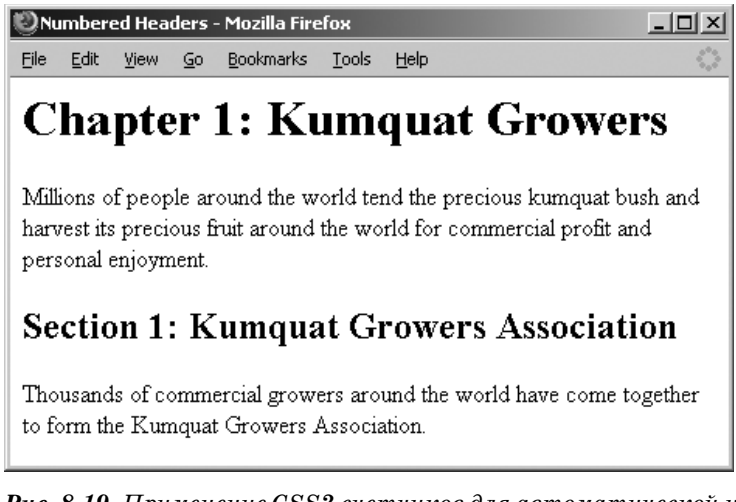

Рис. 8.19. *Применение CSS2-счетчиков для автоматической нумерации глав и разделов*

[1] Конечно же, вы обратили на него внимание в этой книге!

1.1, 1.1.1 и т. д., продемонстрированному ранее в этой главе (см. рис. 8.3).

Как это было уже не раз, только новички на рынке броузеров, Firefox и Opera, корректно выводят счетчики и содержимое, сгенерированные с помощью таблиц стилей.

8.4.11.6. Создание маркеров

Согласно стандарту CSS2 броузер должен помещать содержимое, сгенерированное в соответствии с таблицами стилей, до или после обычного HTML/XHTML-содержимого элемента, в результате чего сгенерированное содержимое становится составной частью содержимого элемента. Этот подход неприемлем для нумерованных списков, где номер должен появляться отдельно от содержимого пункта списка. Чтобы добиться этого, добавьте в генерируемое содержимое свойство `display` и установите его в специальное значение `marker`. Чтобы сделать наш пример с вложенными нумерованными списками безупречным, установим следующие правила:

```
ul { counter-reset: item }
li:before { display : marker;
   counter-increment: item ;
   content: counters(item, ".") }
```

В результате сгенерированное значение счетчика появится слева от содержимого элемента. Аналогичным образом вы можете помещать маркеры после элемента. Например, ниже показано, как создавать нумерованные уравнения в тексте главы (элемент `<blockquote>` очерчивает уравнение):

```
h1:before { counter-increment : chapter;
   counter-reset : equation }
blockquote:after { counter-increment : equation;
   display : marker;
   content : "("counter(chapter, upper-roman) "-" counter(equation) ")" }
```

При выводе маркера броузер определяет, куда его поместить, относительно содержимого элемента. Вы управляете этим поведением с помощью свойства `marker-offset`. Оно принимает числовое значение, равное расстоянию от края маркера до края ассоциированного элемента. Например, чтобы номер уравнения отстоял от уравнения на 0.5 дюйма, мы пишем следующее правило:

```
h1:before { counter-increment : chapter;
   counter-reset : equation }
blockquote:after { counter-increment : equation;
   display : marker;
   content : "("counter(chapter, upper-roman) "-" counter(equation) ")";
   marker-offset : 0.5in }
```

8.4.12. Аудиосвойства

Язык HTML скромно начинал свой путь как средство визуализации данных на экране компьютера. Хотя по мере его развития внимание уделялось и другим носителям информации, стандарт CSS2 по сути является первой попыткой полноценной интеграции невизуальных данных в HTML/XHTML-документы.

Например, в стандарте CSS2 содержится прогноз, что когда-нибудь броузеры будут в состоянии озвучивать текстовое содержимое документа с помощью некоторой технологии преобразования текста в речь. Такой броузер был бы огромным подспорьем для людей с нарушениями зрения, а также позволил бы пользователям путешествовать по Всемирной паутине с помощью телефона или другого устройства, дисплей которого плохо подходит для отображения содержимого сайтов. Представьте, как замечательно было бы вести машину, прослушивая содержимое веб-страниц![1]

CSS2 стандартизирует альтернативные способы вывода информации, определяющие звуковой опыт веб-слушателя. Ни один из них не поддерживается сейчас популярными броузерами, однако мы предвидим, что в ближайшем будущем вам удастся воспользоваться некоторыми из этих свойств.

8.4.12.1. Свойство volume

Самым важным звуковым свойством является `volume` (громкость). Оно принимает числовые значения, абсолютные или относительные, а также несколько ключевых слов, соответствующих предустановленным значениям громкости.

Числовые значения лежат в диапазоне от 0 до 100, причем 0 соответствует минимальному уровню громкости, при котором звук еще слышен, а 100 – максимальному комфортному уровню. Обратите внимание, что 0 не означает отсутствие звука, поскольку минимальный уровень слышимости в шумном окружении (например, в производственном помещении) может быть довольно высоким.

Относительные значения выражают процентное соотношение громкости элемента к уровню громкости его контейнера. Отрицательные значения приравниваются к нулю, а значения, превышающие 100, усекаются до ста. То есть, чтобы заставить элемент звучать вдвое громче его родителя, установите его свойство `volume` в значение 200%. Если громкость родителя равна 75, элемент-потомок получает максимально возможный уровень громкости 100.

---

[1] А также представьте, как ужасно будет, если кто-то затеет прослушивать веб-страницы, когда вы спокойно сидите в кафе или смотрите фильм в кинотеатре. Мы постоянно убеждаемся, что успехи технологии имеют и негативную сторону.

Вы можете указать ключевое слово, определяющее значение свойства volume. Так, значение silent отключает звук. Значение x-soft соответствует нулевому, soft – числовому значению 25, medium равно 50, loud равно 75, а x-loud – это 100.

8.4.12.2. Речевые свойства

Три свойства управляют преобразованием текста в речь. Первое, speak, включает и выключает эту функциональную возможность. По умолчанию значением свойства speak является normal. Оно показывает, что текст преобразуется в речь в соответствии со стандартными, локальспецифичными правилами произношения, грамматики и интонации речи. Если вы установите свойство speak в значение none, речь будет отключена. Вы можете использовать этот прием, чтобы подавить проговаривание второстепенного содержимого или содержимого, плохо воспринимаемого на слух, например таблиц.

Наконец, вы можете указать для свойства speak значение spell-out, которое заставит броузер читать в словах отдельные буквы. Это очень полезно при чтении аббревиатур. Например, если мы укажем:

```
acronym { speak : spell-out }
```

то название телеканала ОРТ прозвучит как «оэртэ», а не «орт».

По умолчанию свойство speak-punctuation устанавливается в значение none, и в результате пунктуация отражается в синтезированной речи в виде пауз и изменения интонации. Если вы зададите этому свойству значение code, то знаки препинания будут проговариваться. Это полезно при речевом воспроизведении фрагментов программного кода и еще в некоторых случаях.

Свойство speak-numeral по умолчанию имеет значение continuous, в результате чего все цифровые последовательности произносятся как число. Например, 1234 прозвучит как «одна тысяча двести тридцать четыре». Установив значение digits, вы заставите броузер произносить «один два три четыре».

8.4.12.3. Голосовые характеристики

Стандарт CSS2 определяет ряд свойств, меняющих проговариваемое содержимое, что обогащает впечатления от прослушивания. Для разного содержимого можно назначать разные голоса, ускорять речь, а также менять высоту звука и силу ударений.

Свойство speech-rate принимает числовое значение, определяющее количество слов в минуту. Умолчание зависит от локали, потому что в разных культурах существуют разные представления о «нормальной» скорости речи. Вместо конкретного значения вы можете воспользоваться ключевыми словами x-slow, slow, medium, fast и x-fast, которые соответствуют скоростям 80, 120, 180, 300 и 500 слов в минуту. Ключевое слово faster увеличивает скорость проговаривания на 40 слов в ми-

нуту по сравнению со скоростью в контейнерном элементе, а ключевое слово slower уменьшает скорость аналогичным образом.

Свойство voice-family (семейство голосов) является звуковым аналогом свойства font-family. Семейство голосов определяет стиль и тип речи. Эти качества, как и шрифт, в большой степени зависят от броузера и платформы. Предполагается, что броузеры будут включать в себя общие семейства голосов, такие как «мужской», «женский» и «детский», и несколько специфических семейств, таких как «диктор» или «сказочник». Значением свойства voice-family должен быть список имен таких семейств. Броузер будет просматривать этот список, пока не найдет семейство голосов, с помощью которого он может озвучить текст элемента.

Свойство pitch управляет средней высотой звука, измеряемой в герцах (Гц). Базовая высота голоса определяется семейством, к которому он принадлежит. Изменение высоты позволит вам варьировать характеристику голоса подобно тому, как вы меняете размер шрифта. Например, увеличив высоту, вы можете заставить «сказочника» пищать, как мышь.[1]

Свойству pitch можно придать числовое значение, например 120hz или 210hz (средние частоты типичного мужского и женского голосов), или одно из ключевых слов x-low, low, medium, high или x-high. В отличие от остальных ключевых слов, характеризующих речь, эти слова не соответствуют каким-то конкретным частотам и сильно зависят от базовой частоты голосового семейства. Единственное требование, которое к ним предъявляется, сводится к тому, что они должны отражать постепенное повышение или понижение частоты.

В то время как свойство pitch устанавливает среднюю высоту звука, свойство pitch-range определяет диапазон изменения частоты при прочтении текста броузером. Значением этого свойства должно быть число от 0 до 100 (умолчание – 50). Установка свойства pitch-range в ноль дает «плоский», монотонный голос; значения, превышающие 50, позволяют воспроизводить живые голоса с богатой модуляцией.

Свойство stress управляет силой ударений в проговариваемом тексте. В различных языках существуют разные манеры произношения ударных слогов и варьирования интонации в зависимости от грамматики. Свойство stress принимает значения от 0 до 100. Умолчание (50) соответствует «нормальным» ударениям. Значение 0 полностью выключает интонационные характеристики содержимого. Значения, превышающие 50, увеличивают интонационное выделение определенных звуковых элементов.

Свойство richness определяет качество, или богатство, голоса. Насыщенный голос заполняет помещение и производит более сильное впе-

[1] Если, конечно, он еще не подражает голосу мыши.

чатление на слушателя. Подобно свойствам pitch и stress, свойство richness принимает числовое значение от 0 до 100 с умолчанием 50. Значения, близкие к нулю, делают голос мягче и тише; значения, превышающие 50, делают его насыщенным и громоподобным.

8.4.12.4. Свойства, определяющие паузы

Аналогично пробелам и отступам в тексте, паузы в проговариваемом содержимом служат для привлечения внимания и облегчения восприятия.

Свойства pause-before и pause-after вставляют паузы непосредственно до и сразу после звукового содержимого элемента. Эти свойства принимают в качестве значений либо абсолютный отрезок времени (выраженный в секундах, s, или миллисекундах, ms), либо процентное значение. Во втором случае продолжительность паузы определяется относительно времени, необходимого для проговаривания одного слова. Например, если темп речи равен 120 словам в минуту, то одно слово произносится в среднем за 0,5 секунды. Тогда 100-процентная пауза будет длиться 0,5 секунды, а 20-процентная – 0,1 секунды.

Свойство pause устанавливает свойства pause-before и pause-after одновременно. Одиночное значение свойства pause влияет на оба свойства, а если указано два значения, то свойство pause-before примет первое из них, а свойство pause-after – второе.

8.4.12.5. Маркерные свойства

Маркерные свойства позволяют вам вставлять звуковые метки до или после элемента. Например, вы можете предварять каждую главу книги музыкальной заставкой или отмечать конец цитаты специальным сигналом.

Свойства cue-before и cue-after принимают в качестве значения URL-адреса звукового файла, который броузер должен загрузить и воспроизводить до или соответственно после элемента. Технически звук может иметь любую продолжительность, но предполагается, что звуковые маркеры будут короткими и ненавязчивым. В противном случае звуковой пользовательский опыт окажется негативным.

Свойство cue устанавливает свойства cue-before и cue-after одновременно. Одиночное значение свойства cue влияет на оба свойства, а если указано два значения, то свойство cue-before примет первое из них, а свойство cue-after – второе.

8.4.12.6. Аудиомикширование

Чтобы создать позитивный звуковой пользовательский опыт, вы, вполне возможно, захотите воспроизводить фоновую музыку одновременно с речевым фрагментом. Свойство play-during поможет вам в этом. Его значениями являются URL-адрес звукового файла и несколько ключевых слов, управляющих воспроизведением.

Ключевое слово repeat повторяет фоновое аудио, пока не закончится проговаривание содержимого. Если это слово опустить, фоновое аудио проигрывается один раз, даже если оно короче содержимого. Фоновый звук, превышающий по продолжительности содержимое, обрывается по окончании содержимого.

Ключевое слово mix предписывает броузеру, поддерживающему стилевые таблицы CSS2, микшировать фоновое звучание с любыми другими фоновыми звуками, воспроизводимыми родительским элементом. Если вы опустите это ключевое слово, фоновое звучание элемента-потомка заменит звуковой фон родителя, воспроизведение которого возобновится по окончании текущего элемента.

Вместо URL-адреса, определяющего фоновый звук, вы можете поставить значение none. Это отключит все фоновые звуки, в том числе и фон родительских элементов, на время проговаривания текущего содержимого.

8.4.12.7. Пространственное позиционирование

В то время как визуальный документ существует на двухмерной странице, проговариваемое содержимое может доноситься из любой точки трехмерного пространства, окружающего слушателя. Стандарт CSS2 определяет свойства azimuth и elevation, которые позволяют располагать звуковое содержимое элементов в разных точках вокруг слушателя. Свойство azimuth определяет направление на источник звука в плоскости слуховых органов, а elevation – расположение источника по вертикали, выше или ниже этой плоскости.

Свойство azimuth принимает либо угол, либо ключевое слово, обозначающее позицию вокруг пользователя. Направление прямо перед слушателем соответствует нулю градусов; справа от него – 90 градусов; сзади – 180 градусов; слева – 270 или -90 градусов.

Ключевые слова, определяющие позиционирование, могут быть модифицированы ключевым словом behind. Их соответствие угловым значениям показано в таблице 8.2.

Ключевое слово leftwards приводит к вычитанию 20 градусов из значения azimuth родительского элемента, а ключевое слово rightwards – к прибавлению 20 градусов к этому значению. Этот процесс может продолжаться, пока вы не обойдете вокруг слушателя. Значения прибавляют или вычитают 20 градусов, каким бы ни был азимут родителя.

Свойство elevation принимает угловое значение от -90 градусов до 90 градусов. Крайние значения соответствуют точкам непосредственно под слушателем и над ним. Ноль градусов определяет уровень ушей. Вместо значений -90, 0 и 90 можно использовать ключевые слова below, level и above соответственно.

Ключевое слово higher прибавляет 10 градусов к значению elevation родительского элемента, а ключевое слово lower вычитает 10 градусов из значения родительского свойства elevation.

Таблица 8.2. Угловые эквиваленты азимутальных ключевых слов

Ключевое слово	Угол	Угол при использовании модификатора behind
left-side	270	270
far-left	300	240
left	320	220
center-left	340	200
center	0	180
center-right	20	160
right	40	140
far-right	60	120
right-side	90	90

8.4.13. Вывод на печать

Печать никогда не была сильной стороной HTML. На самом деле стандарты HTML и XHTML намеренно игнорировали вывод на печать, потому что он предполагает верстку печатаемой страницы, а HTML и XHTML инструментами верстки не являются.

Авторы веб-страниц используют каскадные таблицы стилей для форматирования и верстки HTML/XHTML-документов, поэтому не удивительно, что стандарт CSS2 вводит некоторые базовые возможности для разбивки на страницы, которые дают броузеру подсказку, как лучше распечатывать документ. Эти функциональные возможности делятся на две группы: те, что определяют компоновку конкретной страницы, и те, что управляют разбивкой документа на страницы.

8.4.13.1. Определение страниц

В качестве расширения модели контейнеров стандарт CSS2 определяет *контейнер страницы* – прямоугольник конечного размера, в который выводится содержимое. Контейнер страницы не обязательно соответствует физическому листу бумаги. Пользовательский агент отображает один или несколько страничных контейнеров в листы бумаги в процессе печати. Несколько маленьких страничных контейнеров могут уместиться на одном листе, а большие могут быть пропорционально уменьшены до размеров одного листа или же разбиты на несколько листов по усмотрению броузера.

В процессе печати содержимое попадает в страничный контейнер, претерпевает соответствующую разбивку на физические страницы и передается на целевой лист печатающего устройства. Размеры контейнера страницы могут отличаться от окна вывода в броузере, так что внешний вид распечатанного документа может отличаться от его экранного представления. Как всегда, получение конкретного, наперед заданного внешнего вида документа в общем случае невозможно. Впрочем, вы

можете воспользоваться возможностями разбивки на страницы, предлагаемыми стандартом CSS2, и помочь броузеру красиво распечатать ваш документ.

Вы определяете контейнер с помощью специальной директивы `@page`. Сразу за ключевым словом `@page` идет необязательное имя страницы, а за ним – заключенный в фигурные скобки список свойств, отделенных друг от друга символами ";". Эти свойства определяют размер, поля и внешний вид контейнера страницы.

Для определения размера страничного контейнера пользуйтесь свойством `size`. В качестве значения оно принимает одно или два числа или одно из специальных ключевых слов: `portrait`, `landscape` или `auto`. Если вы укажете единственное числовое значение, будет создан квадрат со стороной, равной этому числу. Два числовых значения задают ширину и высоту страницы соответственно. Ключевое слово `portrait` определяет страницу, у которой высота больше ширины (конкретные размеры устанавливаются броузером и, как правило, составляют 8×11 дюймов), а `landscape` – страницу, у которой ширина больше высоты (как правило, 11×8). Ключевое слово `auto` определяет страничный контейнер того же размера, что и лист бумаги, на котором распечатывается документ.

Вообще говоря, вам следует использовать специальные ключевые слова при указании размера, чтобы ваш документ хорошо распечатывался на других компьютерах. Строка

```
@page normal { size : 8.5in 11in }
```

годится для США, но сможет привести к нежелательным последствиям в европейских локалях. Лучше указать:

```
@page normal { size : portrait }
```

Тогда в Америке будет выбрана страница 8.5×11 дюймов, а в Европе – страница формата A4.[1]

Чтобы установить поля страницы, задавайте свойства `margin`, `margin-top`, `margin-bottom`, `margin-left` и `margin-right` в специальной директиве `@page`. Помните, что броузер может установить собственные поля для вывода страничного контейнера на лист бумаги, так что ваши поля будут складываться с этими. По умолчанию поля страничного контейнера не определены и зависят от настроек броузера.

Свойство `marks` используется внутри специальной директивы `@page` для постановки меток обрезки и меток совмещения вне страничного контейнера на целевом листе бумаги. По умолчанию никакие метки не печатаются. Вы можете указать одно или оба ключевых слова `crop` и `cross`, чтобы поставить на листе метки обрезки и совмещения соответственно.

[1] Слово «normal» в этих примерах означает, конечно же, имя страницы.

8.4.13.2. Левая, правая и первая страницы

Часто при печати документов авторы хотят, чтобы первая страница имела особый формат, а также чтобы левая и правая страницы двухсторонних документов имели разные форматы. Стандарт CSS2 удовлетворяет эти потребности с помощью трех псевдоклассов, добавляемых к имени страницы.

Псевдокласс `:first` применяет страничный формат к первой странице документа. Атрибуты компоновки страницы, задаваемые на странице `:first`, переопределяют соответствующие атрибуты в общей компоновке. Вы можете использовать псевдокласс `:first` в комбинации с именованной компоновкой. Соответствующая компоновка первой страницы будет применена, если первая страница документа выводится как именованная страница.

Аналогичным образом псевдоклассы `:left` и `:right` определяют левую и правую компоновку страниц вашего документа. И опять именованные страницы могут иметь левую и правую версии. Тогда броузер автоматически применяет соответствующую левую и правую компоновку к страницам документа, если такие компоновки существуют.

Нет необходимости указывать именованные страницы, чтобы воспользоваться любым из этих псевдоклассов. Но в большинстве документов это и не делается. Например, если вы используете такие установки:

```
@page :first { margin-top : 3in }
@page :left { margin-left : 2in; margin-right : 1in }
@page :right { margin-left : 1in; margin-right : 2in }
```

без последующих переопределений, то у первой страницы документа будет трехдюймовое верхнее поле (и соответствующие левое и правое поля в зависимости от локали и от того, является ли первая страница документа левой или правой). Последующие страницы будут поочередно иметь то левую, то правую компоновку.

8.4.13.3. Использование именованных страниц

Когда вы создаете именованную компоновку страницы, вы можете использовать ее в своем документе, добавив свойство `page` к стилю, который впоследствии будет применен к элементу документа. Если элемент имеет компоновку страницы, отличную от компоновки предшествующего или контейнерного элемента, в документ вставляется символ перехода на новую страницу и форматирование возобновляется в соответствии с новой компоновкой страницы. Когда область действия элемента заканчивается, восстанавливается предыдущая компоновка, естественно со вставкой символа перехода на новую страницу в случае необходимости.

Например, следующий стиль предписывает выводить все таблицы на страницы с альбомной ориентацией:

```
@page { size : portrait }
@page rotated { size : landscape }
table { page : rotated }
```

Если броузер во время печати встречает элемент <table>, а текущая компоновка страницы в качестве умолчания устанавливает портретную ориентацию, то выполняется переход на новую страницу, и таблица распечатывается с альбомной ориентацией. Если вслед за таблицей идет нетабличное содержимое, броузер вставляет еще один символ перехода на новую страницу, и поток вывода возобновляется на страницы с портретной ориентацией, установленной по умолчанию. Несколько таблиц, идущих подряд, будут распечатаны на одном листе с альбомной ориентацией при условии, что они на нем поместятся.

8.4.13.4. Управление разбивкой на страницы

Если вы не предпринимаете никаких специальных действий, переход на новую страницу совершается, когда изменяется формат страницы или когда содержимое переполняет текущий страничный контейнер. Чтобы расставить или, наоборот, подавить переходы на новую страницу, пользуйтесь свойствами page-break-before, page-break-after и page-break-inside.

Свойства page-break-before и page-break-after принимают в качестве значений ключевые слова auto, always, avoid, left и right. По умолчанию устанавливается значение auto. Оно позволяет броузеру генерировать переходы на новую страницу по мере необходимости. Ключевое слово always принудительно переводит печать на новую страницу до или после элемента, а ключевое слово avoid подавляет переход на новую страницу до или после элемента. Ключевые слова left и right вставляют один или два символа перехода на новую страницу так, чтобы элемент печатался на левой или на правой странице.

Использование свойств, обеспечивающих разбивку на страницы, не составляет никакого труда. Предположим, что в вашем документе главы имеют заголовки первого уровня, а разделы – второго. Вы хотите, чтобы каждая глава начиналась с новой страницы, причем с правой. Кроме того, заголовки разделов не должны отрываться от текста из-за перехода на новую страницу. Вы пишете следующее CSS2-правило для печати:

```
h1 { page-break-before : right }
h2 { page-break-after : avoid }
```

Со свойством page-break-inside следует использовать только значения auto и avoid. Значение auto разрешает переходы на новую страницу внутри элемента (это поведение устанавливается по умолчанию), а avoid подавляет их. Впрочем, элементы, не умещающиеся на физический лист, все равно разрываются. Поэтому выбрано ключевое слово avoid (избегать), а не prevent (предотвращать).

Если вы хотите, чтобы ваши таблицы по возможности не разрывались переходами на новую страницу, напишите такое правило:

```
table { page-break-inside : avoid }
```

8.4.13.5. Управление висячими строками

На типографском жаргоне *висячей* называется одна строка абзаца, оторванная от него переходом на новую страницу (то есть первая строка, оставшаяся внизу страницы, или последняя строка, перешедшая на следующую страницу). Вообще говоря, висячие строки в напечатанном тексте выглядят некрасиво. Большинство принтеров старается оставить хотя бы две строчки текста вверху или внизу каждой страницы.

Если вы хотите управлять этим поведением, можете применять к элементу свойства `widows` и `orphans`. Значением каждого из них является минимальное количество строчек текста, оставляемых соответственно наверху и внизу страницы. Значением по умолчанию является 2. Согласно этой установке броузер генерирует переходы на новую страницу по мере необходимости, гарантируя, что хотя бы две строки текста от каждого элемента будут находиться внизу или наверху каждой страницы. Вы, скорее всего, захотите применить это свойство ко всем элементам в документе, чтобы обеспечить красивую разбивку на страницы.

8.5. Бестеговые стили – тег

До сих пор мы применяли каскадные таблицы стилей для управления внешним видом содержимого определенных тегов. Однако в некоторых случаях вы, возможно, захотите изменить внешний вид только части содержимого тега – обычно текста. Обозначайте эти специальные сегменты тегом `<span>`.

Тег `<span>` просто ограничивает некую часть содержимого (в соответствии с обычными правилами вложения тегов, разумеется). Броузер интерпретирует тег `<span>` как еще один тег физической или логической разметки. Единственное различие, конечно, состоит в том, что действие тега `<span>`, принимаемое по умолчанию, – оставить текст в покое.

****

Функция:	Выделяет текстовый фрагмент любого размера
Атрибуты:	`class`, `dir`, `id`, `lang`, `onClick`, `onDblClick`, `onKeyDown`, `onKeyPress`, `onKeyUp`, `onMouseDown`, `onMouseMove`, `onMouseOut`, `onMouseOver`, `onMouseUp`, `style`, `title`
Закрывающий тег:	`</span>`; присутствует обязательно
Содержит:	*html_content* (содержимое html)
Может содержаться в:	*body_content* (содержимое тела)

Тег был введен в язык HTML, чтобы вы могли применять управление стилями, отображением и событиями к произвольному фрагменту содержимого документа. Что касается определения стиля для тега , то обращайтесь с ним как с любым другим HTML-или XHTML-тегом:

```
span {color: purple}
span.bigger {font-size: larger}
```

и используйте его как любой другой HTML-или XHTML-тег стиля:

```
Ожидается урожай кумкватов <span class=bigger>больший, чем когда-либо</span>!
```

Подобным образом внешний вид тега может быть определен при помощи встроенного объявления стиля:

```
Ожидается урожай кумкватов <span style="font-size: larger">больший, чем
                                                    когда-либо</span>!
```

Как всякий другой тег физической или логической разметки, может вкладываться в другие теги и содержать их в себе.

Тег также поддерживает (хотя их использование нежелательно) много общепринятых атрибутов. Эти атрибуты позволяют пометить (id) или назвать (title) содержимое тега, изменить характеристики отображения содержимого (class, style), указать используемый язык (lang) и связанное с ним направление отображения текста (dir), кроме того, множество on-атрибутов позволяют реагировать на инициированные пользователем с помощью мыши и/или клавиатуры события, связанные с содержимым тега. Не все они реализованы в популярных в настоящее время броузерах для этого и многих других тегов. [атрибут dir, 3.6.1.1] [атрибут lang, 3.6.1.2] [встроенные стили: атрибут style, 8.1.1] [стилевые классы, 8.3] [атрибут id, 4.1.1.4] [атрибут title, 4.1.1.5] [JavaScript обработчики событий, 12.3.3]

8.6. Применение стилей в документах

Есть несколько вопросов, которые вы должны задать себе перед тем, как использовать таблицы стилей в документах или их собраниях в процессе работы над ними и по ее окончании. Первый, наиглавнейший вопрос состоит в том, употреблять ли их вообще. Честно говоря, немногие эффекты могут быть достигнуты только с применением таблиц стилей, большая их часть осуществляется, хотя и с большим трудом и с гораздо меньшей последовательностью, при использовании тегов физической и логической разметки, подобных <i> и , и различных атрибутов тегов, таких как color и background.

8.6.1. Использовать ли таблицы стилей?

Мы считаем, что стандарт CSS2 – победитель. Не только в связи со стандартами, основанными на JavaScript, но более всего из-за предоставляемых им удобств и эффективности при работе со всевозможными раз-

меченными документами, включая HTML-, XHTML-разработки и любые другие, созданные в согласии со стандартом XML. Большинство используемых в настоящее время броузеров поддерживают стили CSS1 и многие из стилей CSS2. Выигрыш очевиден. Так почему же не применять таблицы стилей?

Хотя мы настоятельно рекомендуем вам изучить и использовать в документах таблицы стилей CSS2, мы все же понимаем, что их создание требует затрат времени и сил, которые окупятся только через продолжительный промежуток времени. Разработка таблиц стилей для двухтрехстраничных документов – это не лучшим образом потраченное время, особенно если вы не будете применять их для других документов. Но, вообще говоря, мы полагаем, что проблема не в том, использовать таблицы стилей или не использовать, а в том, *когда* их использовать.

8.6.2. Какой тип таблиц стилей применять и когда

Раз уж вы решили применять каскадные таблицы стилей (на горе себе или на радость), следующим вопросом будет, какого типа стили – встроенные, определенные таблицами на уровне документа или внешние – следует употреблять и когда? У каждого типа есть свои достоинства и недостатки. Каждый из них лучше всего подходит в определенных обстоятельствах.

8.6.2.1. Достоинства и недостатки внешних таблиц

Когда таблицы стилей должны обеспечить стилистическое единство документов, внешние таблицы позволяют наилучшим и самым легким способом управлять стилями во всем собрании документов. Просто внесите нужное стилевое правило в таблицу стилей и примените их к нужным документам. И поскольку одна таблица стилей действует на все документы, преобразование всего собрания к новому стилю окажется не труднее, чем изменение одного стилевого правила в соответствующей внешней таблице.

Даже в том случае, когда документы могут различаться по стилю, часто оказывается возможным собрать несколько основных правил в одну таблицу, которой смогут совместно пользоваться несколько документов, различных в остальных отношениях. Эти стилевые правила могут включать в себя:

- Цвет фона
- Фоновое изображение
- Размеры и начертания шрифтов
- Поля
- Выравнивание текста

Еще одно преимущество употребления таблиц стилей состоит в том, что другие авторы сетевых документов, которые захотят скопировать

ваш стиль, могут с легкостью получить такую таблицу и сделать свои страницы похожими на ваши. Подражание – это самая искренняя форма лести, так что не стоит огорчаться, что кто-то решил сымитировать внешний облик и характер ваших страниц. Более того, вы не сможете защитить свои таблицы от подключения, так что расслабьтесь. Как и обычные HTML-документы, таблицы стилей нельзя зашифровать или как-то еще спрятать, чтобы окружающие не могли их видеть и пользоваться ими.

Самая большая проблема с внешними таблицами стилей заключается в том, что они увеличивают время, необходимое для получения доступа к странице. Мало того, что броузер должен ее загрузить, он должен также загрузить таблицу перед тем, как страницу удастся представить на обозрение. Хотя большинство таблиц стилей относительно невелики, их присутствие можно с определенностью ощутить, когда соединение с сетью медленное.

Без соответствующей дисциплины внешние таблицы быстро становятся большими и громоздкими. При их создании помните, что включать в таблицы следует только стили, общие для всех обращающихся к ней документов. Если ряд стилевых правил употребляется только одним или двумя документами, лучше выделить их в отдельную таблицу или вставить в документ, используя стили на его уровне. В противном случае вы можете обнаружить, что тратите непомерно много усилий на нейтрализацию действия внешних стилей во множестве отдельных документов.

8.6.2.2. Достоинства и недостатки определения стилей на уровне документа

Определение стиля на уровне документа полезней всего при подгонке документа под особые требования. Такие определения позволяют переиначить одно или несколько правил, определенных во внешних таблицах, чтобы создать слегка отличающийся документ.

Иногда хочется попробовать новые стилевые правила на уровне документа, прежде чем применять их в таблицах стилей. Добавляя и изменяя правила на уровне документа, вы избегаете риска поместить во внешнюю таблицу плохо или совсем не работающий стиль, который испортит внешний вид всех документов, применяющих эту таблицу.

Самая большая проблема таблиц на уровне документа состоит в том, что можно поддаться соблазну использовать только их, вместо того чтобы создать правильную внешнюю таблицу для управления всей коллекцией. Это так легко – добавлять правила в новый документ при его создании – вырезал и вставил. К несчастью, управление собранием документов со стилями, определенными на уровне документа, – это скучное и чреватое ошибками занятие. Внесение даже простого изменения может потребовать нескольких часов редактирования (с возможными ошибками).

В качестве рабочего подхода можно принять, что любое правило, которое используется тремя или большим числом документов, следует вынести во внешнюю таблицу стилей и применить к ним с помощью тега <link> или команды @import. Приверженность этому правилу при создании семейств документов окупит себя со временем, когда придет время менять стили.

8.6.2.3. Достоинства и недостатки встроенных стилей

И наконец, расположенные в самом низу каскада встроенные в теги стили могут переопределить более общие стили. Возьмите в привычку употреблять встроенные стили редко и только для этой цели. Встроенные стили не подлежат повторному использованию, что приводит к более трудному контролю над ними. Кроме того, нужные изменения разбросаны по всему документу, а это делает обнаружение и правку встроенных стилей источником ошибок.

Всякий раз, применяя встроенные стили, хорошо подумайте, нельзя ли достичь того же результата, используя определение класса стилей. К примеру, лучше написать:

```
<style type="text/css">
<!--
  p.centered {text-align: center}
  em.blue {color: blue}
-->
</style>
```

а затем применить это определение:

```
<p class=centered>
<em class=blue>
```

чем использовать эквивалентную по результату конструкцию:

```
<p style="text-align: center">
<em style="color: blue">
```

Классы стилей легче находить и обслуживать, и их можно употреблять повторно в других документах.

9

Формы

- Формы – основные понятия
- Тег <form>
- Простой пример формы
- Получение данных из форм при помощи электронной почты
- Тег <input>
- Тег <button>
- Многострочные области ввода текста
- Элементы множественного выбора
- Атрибуты формы общего назначения
- Группировка элементов формы и обеспечение их надписями
- Эффективные формы
- Программирование форм

Формы, бланки, анкеты – мы заполняем их по всякому поводу с момента нашего рождения и до самой смерти. Проза жизни в канцелярском изложении. Чем же объясняются эти восторги по поводу HTML-форм? Ответ прост – они делают HTML и, конечно же, XHTML действительно интерактивными.

Если задуматься, то взаимодействие с веб-страницей сводится, в основном, к нажиманию кнопок – щелкни здесь, щелкни там, поди туда, поди сюда. С пользователем практически нет обратной связи, а то, что есть, – полностью деперсонализировано. Программы, например, апплеты, сервлеты, а также JSP- и ASP-страницы, предоставляют широкие возможности для взаимодействия с пользователем, но их бывает трудно писать. Формы же без труда пишутся на HTML/XHTML и позволяют создавать документы, собирающие и обрабатывающие пользовательский ввод, и формировать ответ, адресованный конкретному получателю.

Этот мощный механизм имеет большие перспективы, особенно в электронной коммерции. Формы завершают онлайновые каталоги, давая покупателям возможность немедленно заказать продукты и услуги. Они обеспечивают некоммерческим организациям способ принятия новых членов. Они позволяют исследователям рынка собирать данные о пользователях. Они предоставляют автоматизированные средства для взаимодействия с читателями.

Подумайте о том, каким образом вы хотели бы взаимодействовать со своими читателями, пока мы будем подробно обсуждать создание форм на стороне клиента и на стороне сервера.

9.1. Формы – основные понятия

Формы состоят из одного или нескольких полей ввода, кнопок, окон множественного выбора и даже всплывающих меню и карт, которые содержатся все вместе в теге <form>. В документе может быть несколько форм, и в каждой из них может находиться также обычное содержимое, включая текст и изображения. Текст особенно удобен для предоставления пользователю инструкций по заполнению формы и для снабжения элементов формы метками и подсказками. Кроме того, с каждым из элементов можно использовать обработчики событий JavaScript для разных действий, таких как проверка и подтверждение правильности ее заполнения или суммирование стоимости заказов.

Пользователь заполняет различные поля формы, затем нажимает кнопку «Submit» (Отправить) (или клавишу <Enter>), чтобы послать форму серверу. Броузер собирает, если необходимо, кодирует предоставленные посетителем данные и результаты его выбора и отправляет их серверу или по заданному адресу электронной почты[1]. Сервер передает данные поддерживающей программе или приложению, которое обрабатывает информацию и генерирует ответ, обычно на HTML. Ответ может просто выражать благодарность или подсказывать пользователю, как следует правильно заполнять форму, или содержать указания на пропущенные поля. Сервер посылает ответ броузеру клиента, который, в свою очередь, выводит его пользователю. Если связь осуществляется по электронной почте, информация просто попадает в чей-то почтовый ящик. В этом случае нет уведомления об отправке формы.

Вопросы, связанные с обработкой и приемом данных на сервере, не являются частью стандартов HTML 4 или XHTML, они определяются программным обеспечением сервера. Хотя полное обсуждение программирования форм со стороны сервера выходит за пределы этой книги, мы бы проявили недостаточное внимание к интересам читателей, если бы не включили сюда хотя бы простой пример для первого знакомства с темой. С этой целью мы поместили в конце главы несколько набросков, иллюстрирующих ряд общепринятых стилей программирования форм на стороне сервера.

И последнее замечание. Верный себе консорциум W3C уже вовсю работает над XML-версией стандарта для форм. Эта новая разновидность форм, известная под названием XForms, сейчас находится в состоянии «рабочего документа», который будет пересматриваться и редактироваться. Концепция XForms отличается от обычной модели форм бук-

[1] Некоторые броузеры, в частности Netscape и Internet Explorer, могут также шифровать информацию, защищая ее, например, от злоумышленников, ворующих номера кредитных карт. Однако шифровальное устройство должно существовать также и на стороне сервера. Обратитесь к производителю сервера за разъяснениями.

вально по каждому пункту. Формы по-другому определяются, корректность данных проверяется по-другому, и информация передается на сервер тоже по-другому. Как нетрудно догадаться, модель XForms сейчас не поддерживается ни одним броузером и сервером, хотя предварительная версия XForms доступна для тестирования как часть проекта Mozilla XForms Project. Учитывая практическое отсутствие поддержки со стороны броузеров, радикальное отличие новой модели от общепринятой и то, сколько времени потребуется модели XForms для замещения миллионов форм, находящихся в использовании, – мы не станем уделять внимания этой модели в этой главе. Вместо этого мы просто опишем формы в том виде, в каком они определены в стандартах HTML и XHTML, а вас предупредим, что в неопределенном будущем новая модель форм может выйти на первый план.

9.2. Тег <form>

Форма может располагаться в любом месте тела документа, между тегом `<form>` и его закрывающим тегом `</form>`. Вы можете, и мы рекомендуем вам это делать, включать в формы обычное содержимое тела документа, чтобы надписывать поля ввода и снабжать пользователя инструкциями.

Броузеры вставляют специальные элементы формы в содержащий ее абзац, как если бы они были маленькими изображениями, вложенными в текст. Нет специальных правил их макетирования, так что вам придется пользоваться другими средствами, скажем, таблицами и таблицами стилей, чтобы управлять расположением элементов в потоке текста.

Необходимо определить как минимум два специальных атрибута формы, чтобы указать имя сервера, который будет ее обрабатывать, и метод, с применением которого следует передавать серверу ее параметры. Третий, необязательный атрибут позволяет изменять способ кодировки, применяемый для защиты данных при передаче их по сети.

<form>

Функция:	Определяет форму
Атрибуты:	`accept`, `action`, `charset`, `class`, `dir`, `enctype`, `id`, `lang`, `method`, `name`, `onClick`, `onDblClick`, `onKeyDown`, `onKeyPress`, `onKeyUp`, `onMouseDown`, `onMouseMove`, `onMouseOut`, `onMouseOver`, `onMouseUp`, `onReset`, `onSubmit`, `style`, `target`, `title`
Закрывающий тег:	`</form>`; присутствует обязательно
Содержит:	*form_content* (содержимое формы)
Может содержаться в:	*блоке*

9.2.1. Атрибут action

Обязательный атрибут `action` тега `<form>` задает URL приложения, которое должно получить и обработать данные формы. Большинство вебмастеров хранят обрабатывающие формы приложения в специальном каталоге на своем веб-сервере, который обычно называется *cgi-bin*, что означает Common Gateway Interface-binaries[1]. Хранение этих специальных программ и приложений, обрабатывающих формы, в едином каталоге укрепляет безопасность сервера и облегчает его администрирование.

Типичный тег `<form>` с атрибутом `action` выглядит примерно так:

```
<form action="http://www.kumquat.com/cgi-bin/update">
...
</form>
```

URL из примера говорит броузеру, чтобы тот связался с сервером *www* в домене *kumquat.com* и передал значения пользовательской формы приложению с названием *update*, расположенному в каталоге *cgi-bin*.

В целом, если вы видите URL, указывающий на документ в каталоге *cgi-bin*, то можно с большой долей уверенности считать, что этот документ в действительности является приложением, которое всякий раз при вызове динамически создает новую HTML-страницу.

9.2.2. Атрибут enctype

Перед тем как передавать данные серверу, броузер их специальным образом кодирует, чтобы они не были перемешаны или испорчены при передаче. И это дело сервера – раскодировать параметры или передать их приложению в закодированном виде.

Стандартной кодировкой является Internet Media Type «application/x-www-form-urlencoded». Вы можете изменить эту кодировку при помощи не являющегося обязательным атрибута `enctype` тега `<form>`. Единственными возможными вариантами в настоящее время являются «multipart/form-data» и «text/plain».

Вариант multipart/form-data необходимо использовать с формами, содержащими поле выбора файла, который загружает пользователь. Формат text/plain следует применять вместе с `mailto`-URL в атрибуте `action` при посылке форм не на сервер, а по адресу электронной почты. Если в форме нет полей выбора файла и если не нужно применять `mailto`-URL в атрибуте `action`, вероятно, следует игнорировать этот атрибут и просто положиться на то, что броузер и обрабатывающий сервер будут пользоваться принятым по умолчанию типом кодировки. [поле выбора файла, 9.5.1.3]

---

[1] Common Gateway Interface (CGI – общий шлюзовой интерфейс) определяет протокол, по которому сервер взаимодействует с программами, обрабатывающими данные форм.

9.2.2.1. Кодировка application/x-www-form-urlencoded

Стандартная кодировка – application/x-www-form-urlencoded – преобразует все пробелы в значениях формы в знаки «плюс» (+), символы, отличные от латинских букв и цифр, – в знак процента (%), за которым следуют две шестнадцатеричные цифры, являющиеся ASCII-кодом символа, а разрывы строк в многострочных данных – в `%0D%0A`.

Стандартная кодировка включает также имя каждого поля формы. «Поле» – это отдельный элемент формы, значение которого может быть почти любым текстом (от одного числа до нескольких строк), скажем, адресом пользователя. Если в поле несколько значений, они разделяются амперсандами.

Например, вот что броузер посылает серверу после того, как пользователь заполнил форму с двумя полями ввода, имена которых `name` и `address`. В первом поле только одна строка текста, тогда как второе содержит несколько строк ввода:

```
name=O'Reilly+Media&address=1005+Gravenstein+Highway+North%0D%0A
Sebastopol,%0D%0ACA+95472
```

Мы разбили значение на две строки[1] для удобства чтения, но в реальности броузер посылает данные в виде сплошной записи. Поле `name` содержит «O'Reilly Media», и значение поля `address` после раскодирования таково:

```
1005 Gravenstein Highway North
Sebastopol,
CA 95472
```

9.2.2.2. Кодировка multipart/form-data

Кодировка multipart/form-data включает поля формы в виде отдельных частей в состав одного документа, соответствующего стандарту MIME. У каждого поля в результирующем файле есть соответствующий сегмент, отделенный от сегментов соседей стандартным разделителем. В каждом сегменте одна или несколько заголовочных строк определяют имя поля, за которым следуют одна или несколько строк, содержащих его значение. Поскольку часть, содержащая значения, может заключать в себе двоичные данные или не отображаемые по другой причине символы, по отношению к передаваемым данным не предпринимается никакого кодирования или символьного преобразования.

Эта кодировка по своей природе более сложная и длинная по сравнению с application/x-www-form-urlencoded. Поэтому она применяется, только когда атрибуту `method` тега `<form>` присвоено значение `post`, как описано в разделе 9.2.4.

---

[1] На самом деле там 3 строки, что и видно в следующей записи в раскодированном виде. – *Примеч. науч. ред.*

Простой пример позволяет легко уяснить себе этот формат. Здесь приведен наш предыдущий пример, но переданный в кодировке multipart/form-data:

```
-----------------------------146931364513459
Content-Disposition: form-data; name="name"

O'Reilly Media
-----------------------------146931364513459
Content-Disposition: form-data; name="address"

1005 Gravenstein Highway North
Sebastopol,
CA 95472
-----------------------------146931364513459--
```

Первая передаваемая строка определяет разделитель, который будет появляться перед каждым сегментом документа. Разделитель всегда состоит из тридцати тире (–) и длинного случайного числа, отличающего его от другого текста, который может появиться в настоящих значениях полей.

Следующие строки содержат заголовочные поля первого сегмента. Среди них всегда есть поле `Content-Disposition`, указывающее, что сегмент включает данные формы, и сообщающее имя элемента, значение которого передается в этом сегменте. Бывают и другие заголовочные поля, в частности, некоторые поля выбора файла содержат заголовочное поле `Content-Type`, указывающее тип данных в передаваемом файле.

После заголовков идет одна пустая строка, за которой следует действительное значение поля в одной или нескольких строках. Сегмент завершается повторением разделительной строки, с которой начинается передача. Следующий сегмент начинается сразу за разделителем и далее по шаблону, пока все параметры формы не будут переданы. Конец передачи обозначается двумя дополнительными тире в конце последней разделительной строки.

Как мы уже говорили, используйте кодировку multipart/form-data только в случае, когда форма содержит поле выбора файла. Приведем пример того, как может выглядеть передача поля выбора файла:

```
-----------------------------146931364513459
Content-Disposition: form-data; name="thefile"; filename="test"
Content-Type: text/plain

First line of the file[1]
...
Last line of the file
-----------------------------146931364513459--
```

[1] В этом месте могло бы идти и побайтное содержимое двоичного файла, вообще не подлежащее какому-либо отображению на печати, – это очень важная возможность такого формата. – *Примеч. науч. ред.*

Единственное существенное отличие состоит в том, что поле Content-Disposition содержит добавочный элемент filename, определяющий имя передаваемого файла. В таком сегменте может находиться и поле Content-Type, дополнительно описывающее содержимое файла.

9.2.2.3. Кодировка text/plain

Используйте эту кодировку только в том случае, если у вас нет доступа к обрабатывающему формы серверу и нужно послать информацию по электронной почте (атрибут формы action имеет значением mailto-URL). Традиционные кодировки предназначены для компьютеров, text/plain разрабатывался для людей.

В этой кодировке каждый элемент формы помещается на отдельной строке, имя поля формы и его значение разделяются знаком равенства. Возвращаясь к нашему примеру с именем и адресом пользователя, в этой кодировке мы получим:

```
name=O'Reilly Media
address=1005 Gravenstein Highway North%0D%0ASebastopol,%0D%0ACA 95472
```

Нетрудно заметить, что закодированными в этом формате остаются только символы возврата каретки и перевода строки в многострочных полях ввода. Во всех других отношениях результат легко читать и, в общем, можно произвести синтаксический анализ с применением простых средств.

9.2.3. Атрибут accept-charset

Атрибут accept-charset введен в стандарте HTML 4.0. Он позволяет указать список наборов символов (character set), которые должен поддерживать сервер, чтобы надлежащим образом интерпретировать данные формы. Значением этого атрибута служит заключенный в кавычки список названий наборов символов по стандарту ISO. Если среди приемлемых наборов отсутствует тот, который установлен пользователем, броузер может самостоятельно решать, следует ли ему проигнорировать форму или обработать ее особым образом. По умолчанию значением атрибута является unknown, подразумевающее, что наборы символов формы и документа, ее содержащего, совпадают.

9.2.4. Атрибут method

Второй обязательный атрибут тега <form> устанавливает метод, посредством которого броузер передает серверу для обработки данные формы. Существуют два подхода: метод POST и метод GET. Если атрибут method не указан, применяется метод GET.

Следуя методу POST, броузер передает данные в два шага: во-первых, броузер устанавливает связь с обрабатывающим форму сервером, адрес которого указан в атрибуте action, и, во-вторых, когда связь установлена, посылает данные серверу в отдельном сеансе связи.

Ожидается, что со стороны сервера уже запущенные приложения типа POST начнут читать данные, расположенные стандартным образом. Прочитанные данные должны быть декодированы, прежде чем удастся использовать значения формы. Именно сервер в точности определяет, на что могут рассчитывать POST-приложения при получении своих параметров.

Метод GET, напротив, соединяется с обрабатывающим форму сервером и отправляет данные за один шаг: броузер через вопросительный знак дописывает к URL данные из атрибута `action`.

Обычные броузеры могут передавать данные формы любым из этих методов. Некоторые серверы получают данные либо тем, либо другим способом. Вы указываете, какой из двух методов – POST или GET – использует ваш обрабатывающий формы сервер, присваивая значение атрибуту `method` тега `<form>`. Вот полный тег, включающий способ передачи GET в атрибуте `method` для формы из предыдущего примера:

```
<form method=GET
   action="http://www.kumquat.com/cgi-bin/update">
  ...
</form>
```

9.2.4.1. POST или GET?

Какой из двух методов следует использовать, если сервер поддерживает оба – и POST, и GET? Вот несколько практических советов:

- Для наиболее быстрого получения результатов передачи маленькие формы с небольшим числом коротких полей посылайте методом GET.
- Поскольку некоторые операционные системы серверных машин ограничивают длину командной строки с аргументами, передаваемыми приложению за один прием, используйте метод POST для отправки форм, у которых много полей или последние содержат длинные текстовые записи.
- Если у вас нет опыта в написании приложений, обрабатывающих формы со стороны сервера, применяйте GET. Лишние действия, связанные с чтением и декодированием параметров, передаваемых по методу POST, хотя и не слишком трудны, все же, возможно, не заслуживают того, чтобы с ними связываться.
- Если защищенность данных представляется важной, используйте POST. GET помещает параметры формы прямо в URL приложения, где они могут быть легко перехвачены сетевыми анализаторами пакетов или выделены из регистрационного журнала сервера. Если параметры содержат секретную информацию, такую как номер кредитной карты, вы можете нанести вред своим пользователям. Хотя у POST-приложений есть свои огрехи в обеспечении безопасности, они во всяком случае могут воспользоваться шифровкой параметров, передаваемых в отдельном сеансе связи с сервером.

- Если вы хотите вызвать находящееся за пределами царства форм серверное приложение, требующее передачи ему параметров, используйте GET, поскольку он позволяет включить в URL приложения параметры тем же способом, что и параметры формы. Приложения типа POST, напротив, ожидают дополнительной передачи данных от броузера после URL, чего нельзя достичь, употребляя традиционный тег <a>.

9.2.4.2. Явная передача параметров

Предпошлем следующему совету необходимые объяснения. Предположим, у вас есть простая форма с двумя элементами по имени x и y. В закодированном виде значения этих элементов выглядят примерно так:

```
x=27&y=33
```

Если форма применяет метод GET, броузер добавляет в виде префикса URL-адрес приложения, выполняющего обработку на сервере, например, так:

```
http://www.kumquat.com/cgi-bin/update?x=27&y=33
```

Не существует ничего, что могло бы вам помешать создать обычный тег <a>, вызывающий форму, с какими угодно параметрами, скажем, так:

```
<a href="http://www.kumquat.com/cgi-bin/update?x=19&y=104">
```

Одно только нехорошо – амперсанд, разделяющий параметры, служит также для обозначения вставленного кода символа. Броузер, встретив амперсанд в атрибуте href тега <a>, попытается заменить следующие за ним символы на символ, который, как представляется броузеру, здесь закодирован.

Чтобы этого не случилось, вы должны заменить символ амперсанда его кодом: либо &, либо & (см. приложение F). После этой подстановки наш пример не использующего формы обращения к серверному приложению будет выглядеть так:

```
<a href="http://www.kumquat.com/cgi-bin/update?x=19&amp;y=104">
```

Из-за возможной путаницы, возникающей из необходимости избегать амперсанда в URL, создатели серверов решились допустить в качестве разделителя параметров точку с запятой. Вы, вероятно, захотите заглянуть в документацию по вашему серверу, чтобы выяснить, присоединяется ли он к этой конвенции (см. приложение F).

9.2.5. Атрибут target

Применяя фреймы, можно перенаправить результаты обработки формы в другое окно или фрейм. Просто добавьте в тег <form> атрибут target и присвойте ему имя фрейма или окна, которые должны принять результат.

Как и в случае атрибута target для тега <a>, можно использовать множество специальных имен с атрибутом target в теге <form> для создания нового окна или для замещения содержимого существующих окон или фреймов. [атрибут target тега <a>, 11.7.1]

9.2.6. Атрибуты id, name и title

Атрибут id позволяет прикрепить к форме метку, представляющую собой уникальную в документе строку, предназначенную для обращения к этой форме со стороны программ (апплетов или сценариев) и гиперссылок. До того как HTML 4.0 ввел id, Netscape использовал атрибут name для получения подобного результата, хотя его и нельзя было применять в гиперссылках. Для достижения согласия с как можно большим числом броузеров мы рекомендуем употреблять с тегом <form> оба атрибута – name и id, если это оказывается необходимым. В будущем для такой цели можно будет использовать только id.

Атрибут title определяет заключенное в кавычки строковое значение, которое помечает форму. Однако название помечает ее только в качестве сегмента документа – это значение не может применяться для ссылок на форму в апплетах, сценариях и гиперссылках. [атрибут id, 4.1.1.4] [атрибут title, 4.1.1.5]

9.2.7. Атрибуты class, style, lang и dir

Атрибут style создает встроенный стиль для элементов формы, преодолевая любые другие действующие стилевые правила. Атрибут class позволяет применять к содержимому тега стиль, заранее определенный для данного класса тегов <form>. Значение атрибута class – это имя данного класса. [встроенные стили: атрибут style, 8.1.1] [стилевые классы, 8.3]

Действительный эффект атрибута style с тегом <form>, однако, предсказать трудно. В целом стилевые свойства действуют на содержимое тела, в частности текст, которое можно включить как часть содержимого формы, но стили тега <form> будут действовать на характеристики отображения элементов формы.

Например, можно создать стили цвета фона и особого начертания шрифта для формы. Текстовые метки формы, но не текст в полях ввода, будут отображаться указанным шрифтом на указанном фоне. Подобным образом текстовые подписи рядом с кнопками, но не сами кнопки, будут выводиться с применением стиля, определенного в форме.

Атрибут lang дает возможность определить язык, который употребляется в форме, а атрибут dir позволяет рекомендовать броузеру, в каком направлении следует выводить текст. Значение атрибута lang – это двухбуквенный код языка по стандарту ISO, включающий необязательный языковой модификатор. К примеру, lang=en-UK сообщает броузеру, что содержимое формы написано по-английски, причем так, как говорят и пишут в Великобритании (United Kingdom). Предпола-

гается, что броузер как-то отразит в макете и типографских решениях ваш выбор языка.

Атрибут `dir` сообщает броузеру, в каком направлении следует отображать содержимое формы: слева направо (`dir=ltr`), как в русском, английском или французском языках, или справа налево (`dir=rtl`) для таких языков, как китайский или иврит.

Атрибуты `dir` и `lang` поддерживаются популярными броузерами, хотя для конкретных языков какого-то особого поведения броузеров не предусмотрено. [атрибут dir, 3.6.1.1] [атрибут lang, 3.6.1.2]

9.2.8. Атрибуты событий

Как и большинство других элементов документа, тег `<form>` допускает стандартные атрибуты событий, инициируемых с помощью мыши и клавиатуры, которые распознают броузеры, следующие стандартам. Мы подробно описываем большинство этих атрибутов в главе 12. [обработчики событий JavaScript, 12.3.3]

У форм есть два особых атрибута событий: `onSubmit` и `onReset`. Значением этих атрибутов событий служат заключенные в кавычки одно или несколько разделенных точками с запятой выражений JavaScript, методов или ссылок на функции. С атрибутом `onSubmit` броузер исполняет эти команды перед тем, как действительно передать данные формы серверу или послать их по электронной почте.

Вы можете использовать событие `onSubmit` для самых разных эффектов. Самый популярный состоит в проверке со стороны клиента правильности заполнения формы. Программа просматривает данные в форме и подсказывает пользователю заполнить один или несколько пропущенных им элементов. Другое и гораздо более простое употребление состоит в информировании пользователя, что форма с `mailto`-URL отправляется по электронной почте.

Атрибут `onReset` применяется точно так же, как атрибут `onSubmit`, за исключением того, что ассоциированный с ним программный код исполняется, только когда пользователь нажимает кнопку «Reset» формы.

9.3. Простой пример формы

Через мгновение мы приступим к детальному изучению элементов формы. Давайте перед этим бросим взгляд на простой пример, чтобы понять, как они устроены в целом. Эта HTML-форма (рис. 9.1) собирает основные демографические данные о пользователе:

```
<form method=POST action="http://www.kumquat.com/demo">
  Name:
    <input type=text name=name size=32 maxlength=80>
  <p>
  Sex:
```

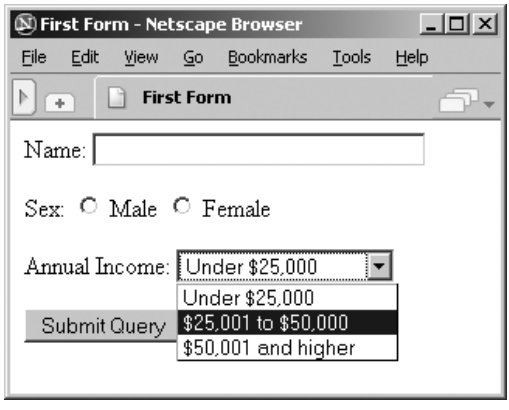

Рис. 9.1. *Простая форма*

```
    <input type=radio name=sex value="M"> Male
    <input type=radio name=sex value="F"> Female
  <p>
  Annual Income:
    <select name=income size=1>
      <option>Under $25,000
      <option>$25,001 to $50,000
      <option>$50,001 and higher
    </select>
  <p>
  <input type=submit>
</form>
```

Первая строка в примере начинает форму и указывает, что мы применяем метод POST передачи данных. Элементы пользовательского ввода следуют далее, каждый из них определяется тегом `<input>` и атрибутом `type`. В нашем примере есть три элемента ввода, каждый из которых содержится в своем абзаце.

Первый элемент – это традиционное поле для ввода текста, разрешающее пользователю ввести до 80 символов, но показывающее одновременно только 32. Второй – это переключатель, позволяющий выбрать одну из двух радиокнопок. Третий – это раскрывающееся меню для выбора одного из трех вариантов. Последний элемент – простая кнопка отправки, которая запускает обработку формы, если предварительно была нажата пользователем.

9.4. Получение данных из форм при помощи электронной почты

Появляется все больше авторов, у которых нет другого доступа к серверу, кроме возможности выложить на нем свои документы. Как следствие, они не могут создавать и администрировать свои CGI-программы.

Фактически некоторые интернет-провайдеры (Internet Service Providers, ISP), особенно те, что предоставляют место для размещения сотен или даже тысяч сайтов, как правило, отказывают в CGI-обслуживании, чтобы не перегружать сервер или из соображений безопасности.

Если вы работаете с одним из множества сайтов, на которых не удается обеспечить обработку формы, даже если это необходимо для спасения вашей жизни, не отчаивайтесь, не все еще потеряно. Можно указать в качестве значения атрибута `action` вашей формы `mailto`-URL. Современные броузеры автоматически отправят параметры и значения формы по указанному адресу электронной почты. Получатель почты может затем обработать ее и предпринять соответствующие действия.

Если вы замените тег `<form>` из предыдущего примера следующим текстом:

```
<form method=POST action="mailto:chuckandbill@oreilly.com"
    enctype="text/plain"
    onSubmit="window.alert('This form is being sent by email, even
    though it may not appear that anything has happened...')">
```

данные формы – когда пользователь нажмет кнопку «Submit» – будут отправлены Чаку и Билли, не подвергаясь никакой обработке на сервере. Отметьте еще, что в форму вставлено простое JavaScript-предупреждение, появляющееся, когда броузер готов отправить данные. Это сообщение говорит пользователю, чтобы он не ждал подтверждения отправки данных (рис. 9.2). Кроме того, если эта функция не была отключена или вы опустили спецификацию атрибута `method=POST`, броузер обычно предупредит читателя, что он вот-вот отправит незашифрованную («text/plain») и, следовательно, незащищенную информацию по сети, и даст ему возможность отменить отправку. В противном случае форма посылается по электронной почте без предуведомления.

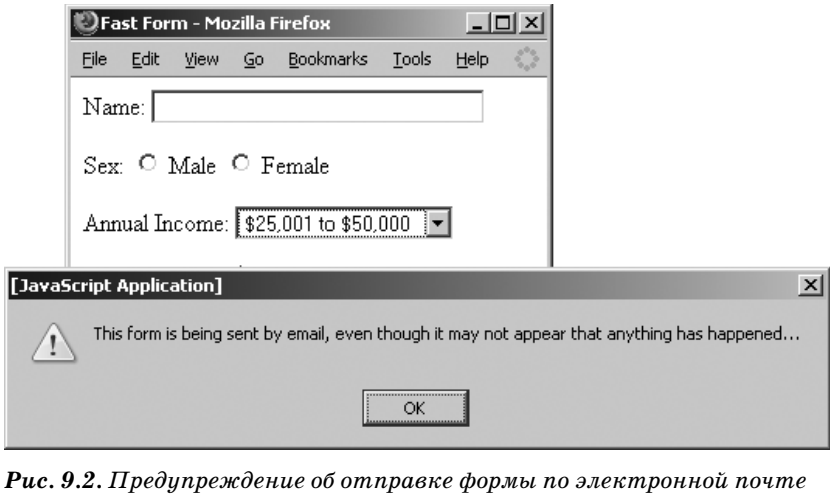

Рис. 9.2. *Предупреждение об отправке формы по электронной почте*

Тело передаваемого по электронной почте сообщения, содержащего результаты заполнения формы, будет выглядеть примерно так:

```
name=Bill Kennedy
sex=M
income=Under $25,000
```

9.4.1. Проблемы, связанные с отправкой форм по электронной почте

Если вы решились использовать `mailto` или какое-либо средство для отправки формы с помощью email-сообщения, вам, возможно, придется столкнуться с несколькими проблемами:

- Ваши формы не будут работать на броузерах, которые не поддерживают `mailto`-URL в качестве значения атрибута `action`. Все популярные в настоящее время броузеры поддерживают формы `mailto`.
- Некоторые броузеры, среди которых ранние (до пятой) версии Internet Explorer, не помещают надлежащим образом данные формы в тело сообщения электронной почты и могут даже открыть диалоговое окно почты, дезориентируя пользователя.
- При отправке данных по `mailto` пользователи не получают страницы, подтверждающей, что их форма обработана. После ее отправки пользователь по-прежнему созерцает форму, как если бы ничего не произошло. (Подражая приведенному выше примеру, применяйте JavaScript с атрибутами `onSubmit` и `onClick`, чтобы разрешать понятное недоумение.) [обработчики событий JavaScript, 12.3.3]
- Ваши данные могут быть получены в таком виде, что их будет трудно или невозможно прочитать, если не был указан пригодный для чтения тип кодировки `enctype=text/plain`.
- Вы теряете любые способы защиты информации формы, которые мог бы предоставить сервер.

Последняя проблема заслуживает дополнительных объяснений. Некоторые веб-провайдеры поддерживают особые «защищенные» серверы, которые при посылке страницы на броузер пользователя присоединяют к ней шифровальный ключ. Популярные броузеры применяют этот ключ, зашифровывая все данные, которые документ, возможно, отправляет такому серверу, включая результаты заполнения формы пользователем. Поскольку только броузер клиента и сервер знают ключ шифра, лишь этот сервер может расшифровать информацию, поступающую ему в ответ от броузера клиента, что эффективно защищает ее от подслушивания и от хакеров.

Однако, если вы для получения данных из формы используете электронную почту, сервер сначала расшифрует их и только потом поместит содержащуюся в форме информацию в тело сообщения и отправит его вам. Электронная почта просто создана для перехвата и расшифровки. Ее содержимое совершенно не защищено.

Так что если вы применяете электронную почту для получения из формы секретных данных, скажем личной информации или сведений о кредитных картах, помните о возможных последствиях. И не дурачьте себя и своих пользователей даже мыслью о «защищенном» сервере, когда из-под него торчит беззащитный хвост электронной почты.

Несмотря на все эти проблемы, передача данных форм по электронной почте представляет привлекательную альтернативу для веб-авторов, доступ которых к серверу ограничен. Наш совет: применяйте CGI-программы, когда только это возможно, а если ничего другого не остается, отступайте на позиции `mailto`-URL.

9.5. Тег <input>

Используйте тег `<input>` при определении одного из ряда обычных элементов управления форм (*controls*, как они называются в стандартах HTML и XHTML), в числе которых поля ввода текста, списки множественного выбора, карты и кнопки. Хотя у тега `<input>` много атрибутов, только атрибут `name` должен присутствовать в каждом элементе (существует исключение из этого правила для кнопок отправки и обновления формы; объяснения следуют далее). И как мы детально обсудим далее, элементы формы каждого типа используют только некоторое подмножество из числа возможных атрибутов тега. Добавочные атрибуты тега `<input>` могут оказаться обязательными, в зависимости от того, какого типа элемент определяется.

<input>

Функция:	Создает элемент для ввода информации в форме
Атрибуты:	`accept`, `accesskey`, `align`, `alt`, `border`, `checked`, `class`, `dir`, `disabled`, `id`, `lang`, `maxlength`, `name`, `notab`, `onBlur`, `onChange`, `onClick`, `onDblClick`, `onFocus`, `onKeyDown`, `onKeyPress`, `onKeyUp` `onMouseDown` `onMouseMove`, `onMouseOut`, `onMouseOver`, `onMouseUp`, `onSelect`, `size`, `src`, `tabindex`, `taborder`, `title`, `type`, `usemap`, `value`
Закрывающий тег:	Отсутствует в HTML; `</input>` или `<input ... />` в XHTML
Содержит:	Ничего
Может содержаться в:	*form_content* (содержимое формы)

Таблица 9.1 суммирует различные типы тега `<input>` и соответствующие атрибуты, обязательные и необязательные.

Тип элемента, который следует включить в форму, указывается с помощью обязательного атрибута `type` тега `<input>`; имя поля (используемое в процессе передачи формы серверу; см. предыдущие объяснения) задается атрибутом `name`. Если вы его опустите, умолчанием для поля

type будет text. Хотя, с формальной точки зрения, значением атрибута name может быть любая строка, мы рекомендуем употреблять имена без пробелов и специальных знаков. Если придерживаться использования только латинских букв и цифр (но не начинать имя с цифры), а пробелы представлять подчеркиванием (_), то проблем будет заметно меньше. Например, «cost_in_dollars» (цена в долларах) и «overhead_percentage» (накладные расходы в процентах) – это хорошо выбранные имена элементов; а «$COST», «накладные расходы %» или даже «overhead %» могут привести к трудностям.

Обратите внимание, что назначенное вами имя элемента формы прямо ассоциировано с данными, вводимыми пользователем, которые будут переданы затем обрабатывающему форму серверу. Это вовсе не то же самое и играет совсем иную роль, нежели атрибут name для гиперссылки или фрейма.

Таблица 9.1. Обязательные и некоторые основные атрибуты элементов формы

Тег формы или тип поля ввода <input>	Атрибуты (× = обязательный; • = необязательный; пусто = не поддерживается)																										
	Accept	**accesskey**	**align**	**alt**	**border**	**cols**	**checked**	**disabled**	**maxlength**	**multiple**	**name**	**notab**	**onBlur**	**onChange**	**onClick**	**onFocus**	**onSelect**	**readonly**	**rows**	**size**	**src**	**tabindex**	**taborder**	**usemap**	**value**	**wrap**
button		•						•			×	•	•		•	•						•	•		×	
checkbox		•					•	•			×	•			•			•				•	•		×	
file	•	•						•	•		×	•	•	•	•	•		•		•		•	•		•	
hidden											×														×	
image		•	•	•	•			•			•	•			•						×	•	•	•		
password		•						•	•		×	•	•	•	•	•	•	•		•		•	•		×	
radio		•					•	•			×	•			•			•				•	•		×	
reset		•						•				•			•							•	•		•	
submit		•						•			•	•			•							•	•		•	
text		•						•	•		×	•	•	•	•	•	•	•		•		•	•		•	
<button>		•						•			×		•		•	•						•			•	
<select>								•		•	×		•	•	•	•				•		•				
<textarea>		•				•		•			×		•	•	•	•	•	•	•			•				•

9.5.1. Поля ввода текста

Стандарты HTML 4 и XHTML позволяют включать в формы поля ввода текста четырех видов: традиционные поля ввода, маскированные

поля для ввода секретной информации, поля для ввода имени файла, который должен передаваться как часть данных формы, и специальное многострочное окно ввода – тег <textarea>. Первые три определяются при помощи тега <input>, четвертое представляется отдельным тегом, который будет рассмотрен в разделе 9.7.

9.5.1.1. Традиционные поля ввода

Самыми распространенными элементами формы являются поля ввода для имен пользователей, адресов и других уникальных данных. Поле ввода выглядит в окне броузера как пустое окно, содержащее строку, и принимает в себя одну строку пользовательского ввода, которая становится значением этого поля после того, как пользователь отправляет форму серверу. Чтобы создать поле ввода внутри формы в документе, присвойте значение text атрибуту type тега <input>. Включите в тег атрибут name – он обязательный.

Что такое строка текста, различные броузеры могут понимать по-разному. К счастью, HTML и XHTML дают возможность при помощи атрибутов size и maxlength определять ширину окна ввода и допустимое количество принятых символов. Значение каждого из атрибутов – это число, равное максимальному количеству символов, которое пользователь может видеть и соответственно набрать в окне ввода. Если maxlength больше, чем size, текст перемещается в окне вперед и назад. Если maxlength меньше, чем size, в окне ввода всегда будет оставаться свободное пространство. Это и составляет разницу между двумя атрибутами.

Принимаемое по умолчанию значение атрибута size зависит от броузера и обычно составляет 80 символов. Если maxlength опущен, длина ввода не ограничена. Мы рекомендуем устанавливать значения этих атрибутов. Подберите величину size так, чтобы окно ввода не выходило за пределы обычного окна броузера (около 60 символов с очень короткой подсказкой). Присвойте maxlength разумное значение: 2 – для сокращенных названий штатов, 12 – для телефонных номеров и т. д.

Поле текстового ввода обычно пусто, пока посетитель не наберет в нем что-нибудь. Можно, однако, указать его начальное, принимаемое по умолчанию значение при помощи атрибута value. Пользователь может его, конечно, изменить. Если он нажмет кнопку «Сброс» (Reset), поле снова примет начальное значение. [кнопки «сброса», 9.5.4.2]

Вот допустимые объявления полей ввода:

```
<input type=text name=comments>
<input type=text name=zipcode size=10 maxlength=10>
<input type="text" name="address" size="30" maxlength="256" />
<input type="text" name="rate" size="3" maxlength="3" value="100" />
```

Первый пример соответствует стандарту HTML и создает поле ввода текста с максимальной длиной и шириной, которые по умолчанию задает сам броузер. Как уже говорилось, такое решение – не самое удач-

ное, поскольку принятые по умолчанию значения широко варьируются от броузера к броузеру, и макет формы будет наверняка плохо выглядеть на некоторых из них. Лучше фиксируйте ширину окна ввода и максимальное число символов, которое допустимо принять от пользователя, как мы это делаем во втором примере. Он позволяет набрать не более десяти символов внутри окна шириной в десять символов. Когда пользователь отправит форму серверу, значение этого поля будет послано с именем «zipcode».

Третий пример соответствует стандарту XHTML и предлагает броузеру отобразить окно для ввода текста шириной в 30 символов, в котором пользователю разрешается набрать до 256 символов. Броузер автоматически будет прокручивать текст в окне, показывая символы, вводимые сверх его ширины.

Последнее окно ввода, тоже XHTML, имеет ширину в три символа, допускает ввод трех символов, и его начальное значение – 100.

Отметьте, что во втором и четвертом примерах подразумевается, что пользователь будет вводить данные определенного вида – почтовый индекс и соответственно числовую величину. Позволяя ограничить *количество* вводимых символов, ни HTML, ни XHTML не предоставляют способа указать, *какие* знаки можно набирать в окне ввода. К примеру, в последнее поле пользователь может ввести «ABC», хотя вы рассчитывали, что это будет число меньше 1000. Ваше приложение со стороны сервера или сценарий со стороны клиента должны отлавливать ошибочный ввод, проверять полноту заполнения формы и отправлять пользователю соответствующее сообщение, если что-то оказалось не в порядке. Все это обернется изрядной скукой, поэтому стоит опять подчеркнуть: подготовьте ясные и точные инструкции и подсказки. Убедитесь, что ваша форма ясно говорит пользователю, какого рода ввод вы от него ожидаете, уменьшая тем самым число ошибок, которые он может сделать при ее заполнении.

9.5.1.2. Маскированное поле ввода

Подобно маске на лице Одинокого рейнджера или Зорро, маска на поле ввода служит благим целям. Само поле ведет себя так же, как и традиционное поле ввода, за одним лишь исключением: вносимые пользователем символы не появляются на экране. Броузеры скрывают их в маскированном тексте, чтобы спрятать пароли и другие секретные коды от любопытных глаз.

Чтобы создать маскированное поле ввода, присвойте атрибуту `type` значение `password`. Все другие атрибуты и семантика традиционного поля применяются и в этом случае. Следовательно, вы должны снабдить поле именем и можете установить для него `size`, `maxlength` и (мы рекомендуем это сделать) начальное значение `value`.

Не обольщайтесь – маскированное поле не такое уж защищенное. Набранное значение скрыто только на экране, а броузер передает его сер-

веру незашифрованным, если только вы не работаете на веб-сервере, использующем SSL (Secure Sockets Layer), например на `https`-сервере. Таким образом, хотя любопытные глаза, возможно, ничего не увидят на экране, не исключено, что хитроумные злодеи украдут информацию электронным способом.

9.5.1.3. Поле выбора файла

Как следует из названия, поле выбора файла дает возможность пользователям выбрать хранящийся на их компьютере файл и послать его серверу вместе с формой. Поле выбора файла в форме создается, когда атрибуту `type` присваивается значение `file`. Как и для других текстовых полей, следует установить подходящие значения атрибутов `size` и `maxlength`. По умолчанию броузер открывает окно ввода шириной в 20 символов.

Броузер выводит это поле на экран, как другие поля текстового ввода, добавляя справа кнопку «Browse» (Обзор). Пользователь либо набирает прямо в поле путь к файлу, либо, нажав на кнопку «Browse», ищет имя локально хранящегося файла при помощи диалогового окна, вид которого определяется системой компьютера.

Кнопка «Browse» (Обзор) открывает специфичное для платформы диалоговое окно, позволяющее выбрать значение поля. В этом случае в поле размещается полный путь к файлу, даже если его длина превышает установленное значение `maxlength`.

Используйте атрибут `accept`, чтобы определить типы файлов, которые броузер позволит выбрать, даже если он и не контролирует, что именно вводят пользователи в качестве пути к файлу. Значением этого атрибута служит список разделенных запятыми MIME-кодов. Пользователи смогут отправлять только те файлы, типы которых есть в списке. К примеру, чтобы ограничить выбор изображениями, стоит добавить `accept="image/*"` в тег `<input>`.

В отличие от других элементов ввода, поле выбора файла работает корректно только с определенной кодировкой данных формы и специфическим методом передачи. Если в форму включено одно или несколько таких полей, следует присвоить атрибуту `enctype` тега `<form>` значение `multipart/form-data`, а атрибуту `method` значение `post`. В противном случае поле выбора файла ведет себя как обычное текстовое поле, передавая серверу свое значение (являющееся путем к файлу) вместо содержимого самого файла.

Все это проще, чем может показаться. Вот пример HTML-формы, которая запрашивает личное имя пользователя и его любимый файл:

```
<form enctype="multipart/form-data" method=post
    action="cgi-bin/save_file">
Ваше имя: <input type=text size=20 name=the_name>
<p>
```

```
Ваш любимый файл: <input type=file size=20 name=fav_file>
</form>
```

Данные, которые в результате заполнения и отправки формы будут посланы броузером серверу, состоят из двух частей. Первая включает значение поля «the_name», а вторая охватывает имя и содержимое выбранного пользователем файла:

```
-----------------------------6099238414674
Content-Disposition: form-data; name="the_name"

Одна строка содержимого текстового поля
-----------------------------6099238414674
Content-Disposition: form-data; name="fav_file"; filename="abc"
Content-Type: text/plain[1]
Первая строка файла
...
Последняя строка файла
-----------------------------6099238414674--
```

Броузер не проверяет, указал ли пользователь действительное имя файла. Если имя отсутствует, раздел `filename` заголовка `Content-Disposition` будет пустым. Если указанного файла не существует, его имя появится в подзаголовке `filename`, но не будет заголовка `Content-Type` и последующих строк содержимого файла. Существующие файлы могут содержать неотображаемые или двоичные данные. Нет возможности ограничить типы файлов, выбираемых пользователем. В свете этих потенциальных проблем приложение, обрабатывающее форму со стороны сервера, должно быть достаточно устойчивым, чтобы справляться с потерянными или чрезвычайно большими файлами либо с теми из них, что содержат некорректные данные или хранятся в необычных и неожиданных форматах.

9.5.2. Выключатели

Выключатели (checkbox), или флажки, в форме дают пользователю способ быстро и легко проводить или отменять выбор объекта. Кроме того, их можно группировать, представляя список объектов, каждый из которых можно выбирать независимо от других.

Для создания отдельного выключателя присвойте атрибуту `type` тега `<input>` значение `checkbox`. Включите в тег обязательные атрибуты `name` и `value`. В передаваемой форме будут содержаться только значения элементов, выбранных пользователем. Необязательный атрибут `checked` (установленный) предлагает броузеру вывести выключатель с заранее установленным флажком и вписать соответствующее значение в пере-

[1] В оригинале этой строки нет, но она упоминается в абзаце ниже, показана в аналогичном примере раздела 9.2.2.2, и именно она определяет тип содержимого файла. – *Примеч. науч. ред.*

даваемый серверу пакет, если только пользователь не щелкнул на выключателе сам, убрав флажок.

Значением выбранного выключателя, которое будет передано серверу, является строка, указанная в обязательном атрибуте `value`. К примеру, следующий XHTML-текст:

```
<form>
  What pets do you own?
  <p>
    <input type="checkbox" name="pets" value="dog" /> Dog
  <br />
    <input type="checkbox" checked="checked" name="pets" value="cat" /> Cat
  <br />
    <input type="checkbox" name="pets" value="bird" /> Bird
  <br />
    <input type="checkbox" name="pets" value="fish" /> Fish
  </p>
</form>
```

создает группу выключателей, показанную на рис. 9.3.

Несмотря на то что каждый выключатель на экране является частью группы, он предоставляет независимую возможность выбора. Заметьте также, что при всем уважении к любителям собак, птиц и рыб, мы заранее отметили флажком выключатель, сопоставленный кошке (Cat), поместив атрибут `checked` в соответствующий тег. Мы также снабдили выключатели текстовыми метками; похожие на них значения атрибутов `value` не отображаются в окне броузера, но являются значениями, которые передаются серверу вместе с ассоциированным именем, если пользователь установил этот флажок. Вам также нужно применять теги абзаца или новой строки для управления макетом группы выключателей, как и других элементов формы.

В приведенном примере, если флажки стоят в окошках с метками «Cat» и «Fish», то значениями, включенными в список параметров посылаемой серверу формы, будут:

```
pets=cat
pets=fish
```

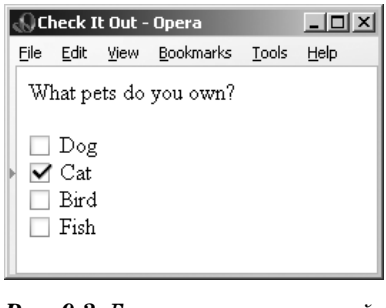

Рис. 9.3. *Группа выключателей (флажков)*

9.5.3. Переключатели

Переключатели очень похожи на выключатели, только пользователь может выбрать в группе лишь один объект[1]. Для создания кнопки переключателя присвойте атрибуту `type` тега `<input>` значение `radio`. Как и выключатели, каждая кнопка переключателя имеет в качестве обязательных атрибуты `name` и `value`. Кнопки переключателя с одним и тем же именем являются членами одной группы. Одну из кнопок можно выбрать изначально, включив в ее тег атрибут `checked`. Если ни один элемент в группе не будет выбран, броузер не будет передавать на сервер ни имени группы, ни ее значений.

Вам следует присвоить каждой кнопке переключателя свое, отличное от других значение, чтобы после отправки формы обрабатывающий ее сервер мог знать о пользовательском выборе.

Здесь представлен предыдущий пример, переписанный в HTML и позволяющий указать одно домашнее животное в качестве любимого (рис. 9.4):

```
<form>
  Which type of animal is your favorite pet?
  <p>
    <input type=radio name=favorite value="dog"> Dog
    <input type=radio checked name=favorite value="cat"> Cat
    <input type=radio name=favorite value="bird"> Bird
    <input type=radio name=favorite value="fish"> Fish
</form>
```

Как и в предыдущем примере с выключателями, мы отдали предпочтение кошкам, сделав кнопку «Cat» принимаемым по умолчанию выбором. Если, например, взять другую, «Bird», броузер автоматически отменит выбор «Cat» и выделит «Bird». Когда пользователь отправит форму серверу, броузер включит в список параметров формы только одно значение с именем `favorite`; `favorite=bird`, если это ваш выбор.

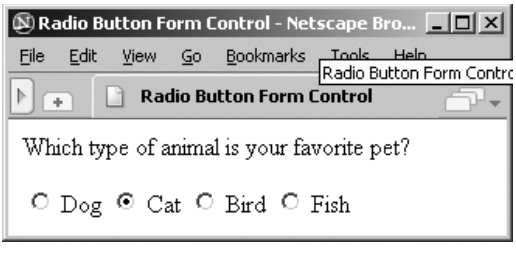

Рис. 9.4. *Переключатель позволяет выбрать один объект из группы*

[1] Кое-кто еще помнит, что раньше у радиоприемников были механические кнопки для выбора станции. При нажатии кнопки та, что была утоплена прежде, выскакивала назад. Это механическая реализация схемы выбора «один из многих».

Один из элементов в переключателе всегда выбран, поэтому нет никакого смысла делать его с одной кнопкой – переключатель должен представлять собой группу из двух или большего числа кнопок. Для элементов управления формы типа включить/выключить, да/нет используйте выключатели.

9.5.4. Активные кнопки

Хотя применяемая терминология и кажется запутанной, но в форме существует еще один класс кнопок. В отличие от выключателей и переключателей, описанных выше, они отрабатывают немедленно, их действие необратимо и они влияют на все содержимое формы, а не только на значение отдельного поля. Эти активные (за неимением лучшего термина) кнопки включают в себя кнопки отправки и сброса формы, обычные кнопки и активные изображения. Кнопки отправки и активные изображения, если они выбраны пользователем, вызывают отправку броузером всех заполненных параметров формы обрабатывающему ее серверу. Обычные кнопки не отправляют форму, но их можно использовать для вызова сценария, с целью манипулирования формой или проверки правильности ее заполнения. Кнопка сброса действует локально, стирая все, что ввел пользователь, и возвращая форму в первоначальное (принятое по умолчанию) состояние.

В этом разделе мы опишем активные кнопки, которые можно создавать при помощи стандартного элемента формы `<input>`. В следующем разделе будет детально рассмотрен новейший тег `<button>`, который позволяет достигать тех же результатов и дает больше возможностей для управления представлением и отображением кнопок в форме.

9.5.4.1. Кнопки отправки

Кнопка отправки (`<input type=submit>`) действует в соответствии с названием, запуская осуществляемый броузером процесс отправки формы серверу. В форме может быть несколько кнопок отправки. В элементе существуют атрибуты `name` и `value`.

Простейшую кнопку отправки (без атрибутов `name` и `value`) броузер отображает в виде маленького прямоугольника или овала с принятой по умолчанию надписью «Submit»[1]. В противном случае броузер помещает текст, который был включен в виде значения атрибута `value`. Если был добавлен атрибут `name`, значение атрибута `value` кнопки отправки вносится в список параметров формы, который броузер отправляет серверу. Это хорошо, так как дает возможность определить, какая кнопка в форме была выбрана пользователем, позволяя обслуживать одну или несколько разных форм при помощи одного обрабатывающего приложения.

Далее следуют допустимые объявления кнопок отправки:

[1] Надпись по умолчанию зависит от версии броузера. – *Примеч. науч. ред.*

```
<input type=submit>
<input type=submit value="Заказ кумкватов">
<input type="submit" value="Срочная доставка" name="ship_style" />
```

Первое написано на HTML и является простейшим – броузер отображает кнопку с надписью «Submit», которая активизирует последовательность обработки формы, если была нажата пользователем. Она не добавляет элемента к списку параметров, который броузер передает обрабатывающим форму серверу и приложению.

Второй HTML-пример включает в себя атрибут `value`, который отображает надпись на кнопке «Заказ кумкватов». Но, как и в первом примере, значение атрибута `value` не попадет в список параметров формы.

Последний пример, соответствующий стандарту XHTML, устанавливает надпись на кнопке и делает ее частью списка параметров. Если пользователь нажал эту кнопку отправки, она добавит параметр `ship-style="Срочная доставка"` в список параметров формы.

9.5.4.2. Кнопки сброса

Само название кнопок сброса («Reset»), создаваемых при помощи тега формы `<input>`, объясняет их действие: они позволяют сбросить (стереть или установить принятые по умолчанию) значения всех элементов формы. В отличие от всех остальных, кнопка сброса не инициирует обработку формы. Обновлением элементов формы занимается броузер. Сервер не знает, нажимал ли кто-нибудь эту кнопку.

По умолчанию броузер отображает кнопку сброса с надписью «Reset»[1]. Надпись можно заменить, указав ее в качестве значения атрибута `value`.

Вот два примера кнопок сброса:

```
<input type=reset>
<input type="reset" value="Use Defaults" />
```

Первая, на HTML, создает кнопку сброса с надписью «Reset» по умолчанию. Второй пример, на XHTML, предписывает броузеру нанести на кнопку надпись «Use Defaults». В ответ на нажатие обеих кнопок броузер сбросит форму в первоначальное состояние.

9.5.4.3. Активные изображения

Элементы `<input type=image>` создают специальные кнопки, являющиеся активными изображениями. Эти кнопки формируются из указанных изображений, а когда пользователь их нажимает, они сообщают броузеру, чтобы он отправил форму серверу и включил в список ее параметров координаты указателя мыши (x,y) в пределах данного изображения, что очень напоминает карты, которые обсуждались в главе 6.

---

[1] В русифицированном Internet Explorer выводится кнопка с надписью «Сброс». – *Примеч. науч. ред.*

Кнопки типа image должны содержать атрибут src с URL файла с изображением; в них также можно включить атрибут name и атрибут описания alt в расчете на неграфические броузеры. Хотя и нежелательно в HTML 4, вы можете также применять атрибут align для управления выравниванием изображения в текущей строке. Используйте атрибут border для определения ширины рамки (если она имеется), в которую Netscape и Firefox заключают изображения в форме так же, как атрибут border для тега <img> (Ни Internet Explorer, ни Opera не рисуют рамки вокруг изображения тега <input> в формах.)

Вот пара допустимых активных изображений:

```
<input type="image" src="pics/map.gif" name="map" />
<input type=image src="pics/xmap.gif" align=top name=map>
```

Броузер выводит указанное изображение в потоке содержимого формы. Изображение второй кнопки будет выровнено по верху примыкающего текста, как это определяется атрибутом align. Netscape и Firefox заключат его в рамку, как делают это с изображениями, входящими в якорь (тег <a>), сигнализируя, что объект является активным.

Когда пользователь щелкает на изображении, броузер посылает серверу горизонтальное смещение (в пикселах) указателя мыши от левого края изображения и вертикальное смещение – от верхнего края. Эти значения присваиваются имени изображения, определенному атрибутом name, за которым следуют .x и .y соответственно. Таким образом, если кто-то щелкнул мышью по кнопке-картинке в нашем примере, броузер передает серверу параметры, поименованные map.x и map.y.

Активные изображения очень похожи на карты (usemap), и подобно программам или тегам <map> со стороны клиента, их обслуживающим, ваш обработчик формы может использовать координаты (x,y) указателя мыши для выбора определенного образа действий. Следует применять активные изображения, когда для обработки пользовательского запроса нужна дополнительная информация, содержащаяся в форме. Если все, что вам нужно, – это разместить набор ссылок на одном изображении, используйте карту. Карты имеют еще одно преимущество: со стороны сервера им обеспечена автоматическая поддержка распознавания отобранных фигур в изображении, что позволяет обращаться с последним как с набором объектов, из числа которых совершается выбор. Активные изображения требуют от вас создания программ, которые будут определять, где пользователь щелкнул и как эта точка изображения должна быть преобразована в соответствующее действие сервера.

Странно, что стандарты HTML 4 и XHTML позволяют употреблять атрибут usemap с активными изображениями, но не объясняют, как при этом разрешается конфликт с обычной обработкой сервером координат (x,y) положения мыши. Мы рекомендуем не смешивать их, применяя карты за пределами форм, а активные изображения – в формах.

9.5.4.4. Обычные кнопки

Используя тег <input type=button> (или тег <button>, описанный в разделе 9.6), удается создать кнопку, которую пользователь может нажать, не вызвав при этом ни сброса формы, ни ее отправки. Атрибут value позволит установить надпись на кнопке. Определение атрибута name приведет к включению значения атрибута value в список параметров, передаваемых процедуре, обрабатывающей форму.

Вас, возможно, заинтересует, какой же от этих кнопок толк? Небольшой или даже никакого – пока вы не включите в их определение один или несколько атрибутов событий с фрагментами кода на JavaScript, которые должны быть реализованы при взаимодействии пользователя с кнопкой. Оснащенные таким образом, эти кнопки предоставляют пользователю возможность запуска проверки допустимости введенных данных, обновления отдельных полей, манипулирования документом и проявления иной активности со стороны клиента. [обработчики событий JavaScript, 12.3.3]

9.5.4.5. Кнопки в форме

В вашей форме может быть несколько кнопок одного или разных типов. Даже простые формы часто содержат, например, кнопки сброса и отправки. Чтобы их различать, убедитесь, что все они обладают разными значениями атрибута value, которые броузер использует для надписей на кнопках. В зависимости от того, как было запрограммировано обрабатывающее форму приложение, вы можете сделать разными значения атрибута name у всех кнопок, но обычно легче назвать все аналогично действующие кнопки одинаково и позволить обрабатывающей подпрограмме различать их по значениям. К примеру (все написано на HTML):

```
<input type=submit name=edit value="Добавить">
<input type=submit name=edit value="Удалить">
<input type=submit name=edit value="Изменить">
<input type=submit name=edit value="Отмена">
```

Когда пользователь выбирает одну из этих кнопок, серверу будет послан параметр под именем edit. Значением этого параметра будет одно из названий кнопок. Приложение со стороны сервера получает значение и действует соответственно.

Поскольку у активных изображений нет атрибута value, единственный способ различать кнопки-картинки в одной форме заключается в том, чтобы снабдить их разными именами.

9.5.5. Скрытые поля

Последний тип элементов ввода формы, который будет рассмотрен в этой главе, невидим глазу. Нет, мы не пытаемся что-то утаить. Это способ включить в вашу форму информацию, которую ни броузер, ни

пользователь не могут ни проигнорировать, ни изменить. Значения обязательных атрибутов `name` и `value` тега `<input type=hidden>` автоматически вносятся в список параметров отправляемой формы. Они служат меткой формы и могут оказать неоценимую помощь при выделении различных форм или их версий из множества уже отправленных или сохраненных.

Кроме того, скрытые поля применяются для управления взаимодействием пользователь/сервер. В частности, они помогают серверу узнать, что эта форма пришла от пользователя, который делал аналогичный запрос несколько мгновений назад. Обычно сервер не хранит такую информацию, и каждая транзакция между сервером и клиентом не зависит от других.

К примеру, первая форма, отправленная пользователем, запрашивала у него основную информацию, такую как его имя и адрес. Опираясь на этот первый контакт, сервер может создать вторую форму, задающую более конкретные вопросы. Поскольку пользователю будет скучно заново вводить основную информацию из первой формы, сервер можно запрограммировать так, чтобы он поместил эти значения во вторую форму в виде скрытых полей. Когда вторая форма вернется, вся важная информация из обеих форм будет в ней, и вторую форму, если это необходимо, можно сопоставить с первой.

Скрытые поля могут побудить сервер к исполнению каких-то определенных действий. Например, можно вложить в форму следующее скрытое поле:

```
<input type=hidden name=action value=change>
```

Тогда, если у вас со стороны сервера одно приложение обслуживает несколько форм, каждая форма будет содержать свой код действия, помогающий серверному приложению разобраться, что к чему.

9.6. Тег <button>

Как уже говорилось выше, вы создаете активную кнопку в стандартном HTML- или XHTML-коде, включая спецификацию ее типа в стандартный тег `<input>`. Например, элемент формы `<input type=submit>` создает кнопку, которая, если была выбрана пользователем, просит броузер послать содержимое формы обрабатывающему серверу или по адресу электронной почты (вариант `mailto`). Что касается отображения, то у вас нет никаких средств управления внешним видом кнопки, если не считаться с тем, что можно заменить принятую по умолчанию надпись «Submit» на другое слово или короткую фразу (например, «Нажми» или «Вот здесь!»).

Введенный впервые стандартом HTML 4.0, тег `<button>` действует так же, как кнопка `<input>`, но предоставляет больше возможностей для управления тем, как броузер будет отображать этот элемент. В частно-

сти, все атрибуты, которые можно использовать с элементом <input type=button>, пригодны для тега <button>.

<button>	
Функция:	Создает кнопку в форме
Атрибуты:	accesskey, class, dir, disabled, id, lang, name, notab [!], onBlur, onClick, onDblClick, onFocus, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, style, tabindex, taborder [!], title, type, value
Закрывающий тег:	</button>; присутствует обязательно
Содержит:	*button_content* (содержимое кнопки)
Может содержаться в:	*form_content* (содержимое формы)

9.6.1. Кнопка <button>

Ни HTML-, ни XHTML-стандарты не излагают ясно, какие еще усовершенствования в отображение кнопки в форме должен привносить элемент <button>, кроме предложения выводить его содержимое «трехмерным» образом и визуально изображать нажатие кнопки, выбранной пользователем, утапливая ее и отпуская назад. Все популярные броузеры поддерживают тег <button>.

Элемент <button> приспособлен для размещения значительно более разнообразного и богатого содержимого, чем его <input>-аналог. Все, что находится между тегами <button> и </button>, становится содержимым кнопки, включая любое приемлемое содержимое тела документа, такое как текст или мультимедийные элементы. К примеру, вы могли бы включить в кнопку изображение и относящийся к нему текст, создав таким образом кнопку с привлекательной пиктограммой, обладающей подписью. Единственный *недопустимый* элемент – это карта, поскольку ее чувствительное к мыши и клавиатуре поведение входит в конфликт с функцией кнопки в форме.

9.6.2. Атрибут type

Используйте атрибут type тега <button> для определения действия кнопки. Значением его могут быть submit, reset или button. Подобно его <input>-аналогу, выбранный пользователем <button type=submit> предлагает броузеру упаковать и отправить содержимое формы обрабатывающему серверу или послать ее по электронной почте mailto получателю. Использование type=reset создает традиционную кнопку сброса формы, а включение type=button приводит к обычной кнопке.

Например, посмотрите, как броузер выводит следующую вставку пиктограммы *exclaim.gif* в трехмерную кнопку, которая нажимается и выскакивает обратно, когда пользователь щелкает на ней мышью (рис. 9.5).

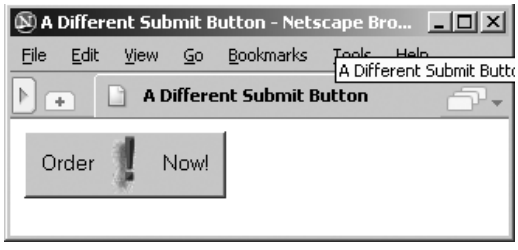

***Рис. 9.5.** Кнопка отправки <button>*

Изображая движение кнопки, броузер одновременно отправляет форму серверу:

```
<button type=submit>
  Order <img src="icons/exclaim.gif" align=middle alt="Order Now"> Now!
</button>
```

Отметьте, что можно применять богатый набор атрибутов тега `<img>`, включая `align` и `alt`, когда изображение содержится в кнопке `<button>`.

Если тег `<button>` так похож на элемент `<input type=button>`, зачем он нужен? Единственный резон – допустить в кнопку гораздо более богатое содержимое.

Если ваши кнопки – это обычные текстовые кнопки, вам хватит тега `<input>`. Если же вы хотите создать причудливую кнопку со смешанным содержимым, вам придется использовать тег `<button>`.

9.7. Многострочные области ввода текста

Традиционные и скрытые поля ввода текста в формах ограничивают пользовательский ввод одной строкой символов. Тег формы `<textarea>` снимает это ограничение.

9.7.1. Тег <textarea>

Как часть формы тег `<textarea>` создает многострочную область ввода текста в окне пользовательского броузера. В ней можно набрать практически неограниченное число строк текста. При отправке формы броузер собирает все строки текста, отделяя их друг от друга последовательностью символов «`%0D%0A`» (возврат каретки/перевод строки), и посылает их серверу в качестве значения элемента формы, используя имя, указанное в обязательном для этого тега атрибуте `name`.

Вы можете поместить между тегом `<textarea>` и его закрывающим тегом гладкий текст. Вставляемый по умолчанию, он должен быть собственно текстом – не содержать тегов и каких-либо еще специальных элементов. Пользователь может модифицировать этот текст, а броузер использует его в качестве значения, принимаемого по умолчанию, когда

<textarea>	
Функция:	Создает многострочную область ввода
Атрибуты:	accesskey, class, cols, dir, disabled, id, lang, name, notab, onBlur, onChange, onClick, onDblClick, onFocus, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, onSelect, readonly, rows, style, tabindex, taborder, title, wrap
Закрывающий тег:	</textarea>; присутствует обязательно
Содержит:	*plain_text* (гладкий текст)
Может содержаться в:	*form_context* (содержимое формы)

тот нажимает кнопку обновления формы. Следовательно, текстовое содержимое тега употребляется чаще всего для инструкций и примеров:

```
Расскажите немного о себе:
<textarea name=address cols=40 rows=4>
  Укажите здесь ваше имя
  1234 Улица такая-то
  Такой-то город, страна, почтовый индекс
</textarea>
```

9.7.1.1. Атрибуты rows и cols

Область ввода многострочного текста расположена на экране отдельно – содержимое тела выводится выше и ниже нее, но не обтекает. Тем не менее вы можете управлять ее размерами, присваивая значения атрибутам cols и rows (столбцы и строки), определяющим прямоугольную область, выделяемую броузером для многострочного ввода. Мы советуем устанавливать эти атрибуты. Обычные броузеры имеют привычку оставлять самый маленький, едва пригодный для чтения прямоугольник, и пользователи не могут изменить область, отведенную под ввод тегом <textarea>. Оба атрибута требуют целых значений, выражающих соответствующий размер в символах. Броузер автоматически прокручивает текст, выходящий за эти пределы.

9.7.1.2. Атрибут wrap

Обычно набранный пользователем текст передается серверу точно так, как он набран, с разрывами строк только в местах, где пользователь нажимал клавишу <Enter>. Поскольку часто это не то, чего хочет посетитель, можно включить в области ввода автоматический перевод строк текста. Когда пользователь набирает строку, которая превышает ширину области ввода, броузер автоматически перемещает лишний текст на следующую строку, разрывая ее в ближайшей точке между словами.

Если атрибуту wrap присвоено значение virtual, текст переносится в области ввода только для представления пользователю, но передается

серверу, как если бы никаких автопереносов строки не было, за исключением тех случаев, когда нажималась клавиша <Enter>.

Если атрибуту wrap присвоено значение physical, текст переносится в окне и отправляется серверу, как если бы пользователь его именно так и набрал. Это самый полезный способ употребления атрибута wrap, поскольку текст отправляется именно в том виде, в каком пользователь видит его в области ввода. Действию по умолчанию соответствует установка wrap=off.

В качестве примера рассмотрим следующие 69 символов текста, набираемые в области ввода в 40 символов шириной:

```
Автоматические переносы - это средство облегчить жизнь пользователям.
```

При установке wrap=off область ввода будет содержать одну строку и пользователю придется прокручивать ее вправо, чтобы увидеть весь текст. Серверу будет отправлена одна строка текста.

При установке wrap=virtual область ввода будет содержать две строки текста с разрывом после слова «средство». Серверу будет передана только одна строка – целиком, без вставленных в нее символов новой строки.

При установке wrap=physical область ввода будет содержать две строки с разрывом после слова «средство». Серверу будут переданы две строки, разделенные символом новой строки после слова «средство».

9.8. Элементы множественного выбора

Выключатели и переключатели предоставляют мощное средство для создания вопросов и ответов с множественным выбором, но их использование ведет к построению длинных форм, которые скучно писать и трудно размещать на уже и так загроможденном экране. Тег <select> предлагает две компактные альтернативы – разворачивающееся меню и прокручиваемые списки.

9.8.1. Тег <select>

Помещая список отмеченных тегами <option> элементов в тег формы <select>, вы волшебным образом создаете разворачивающееся меню. На рис. 9.2, приведенном ранее в этой главе, показано меню <select>.

Здесь, как и в других тегах формы, атрибут name обязателен и используется броузером при отправке серверу списка выбора в теге <select>. Если ни один из элементов списка не выбран пользователем, при отправке формы никакие значения серверу не посылаются. В противном случае броузер, передавая серверу данные тега формы <select>, посылает выбранный элемент, включая в него значение атрибута name.

9.8.1.1. Атрибут multiple

Чтобы допустить выбор нескольких вариантов одновременно, вставьте в тег <select> атрибут multiple. В результате элемент в <select> будет

<select>	
Функция:	Создает меню единичного и множественного выбора
Атрибуты:	class, dir, disabled, id, lang, multiple, name, notab, onBlur, onChange, onClick, onDblClick, onFocus, onKeyDown onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, size, style, tabindex, taborder, title
Закрывающий тег:	</select>; присутствует обязательно
Содержит:	*select_content* (содержимое списка выбора)
Может содержаться в:	*form_content* (содержимое формы)

вести себя, как в <input type=checkbox>. Когда форма представлена, броузер собирает все выделенные варианты в один параметрический список, разделив их запятыми, например так:

```
pets=dog,cat,mouse
```

Если атрибута multiple нет, возможность выбора ограничивается одним элементом, как в случае переключателя.

9.8.1.2. Атрибут size

Атрибут size определяет число одновременно отображаемых для пользователя вариантов выбора. Его значением должно быть положительное целое число. Если size не определен, принимается значение 1. При size=1, если атрибута multiple нет, броузер обычно представляет список <select> в виде разворачивающегося меню. Значение атрибута size, превосходящее единицу, или включение в тег атрибута multiple приводят к выводу элемента <select> в виде прокручиваемого списка.

В следующем XHTML-примере мы преобразовали наш вышеприведенный пример выключателя в прокручиваемый список с множественным выбором. Результат показан на рис. 9.6.

Отметьте также, что атрибут size предписывает броузеру показывать три варианта выбора одновременно.[1]

```
What pets do you have?
  <select name="pets" size="3" multiple="multiple">
    <option>Dog</option>
    <option>Cat</option>
    <option>Bird</option>
    <option>Fish</option>
  </select>
```

[1] Обратите внимание на закрывающий тег </option>. Он не включается обычно в документы, соответствующие стандарту HTML, но обязателен в XHTML.

Рис. 9.6. *Элемент <select>, форматированный указанием size=3*

9.8.2. Тег <option>

Используйте тег `<option>` для определения каждого варианта в элементе формы `<select>`.

Броузер отображает содержимое тега `<option>` в качестве элемента разворачивающегося меню или прокручиваемого списка тега `<select>`, поэтому его содержимое может быть только текстовым, без разметки какого-либо рода.

<option>

Функция:	Определяет вариант выбора в теге `<select>`
Атрибуты:	`class`, `dir`, `disabled`, `id`, `label`, `lang`, `onClick`, `onDblClick`, `onKeyDown`, `onKeyPress`, `onKeyUp`, `onMouseDown`, `onMouseMove`, `onMouseOut`, `onMouseOver`, `onMouseUp`, `selected`, `style`, `title`, `value`
Закрывающий тег:	`</option>`; в HTML обычно опускается
Содержит:	*plain_text* (гладкий текст)
Может содержаться в:	*select_content* (содержимое списка выбора)

9.8.2.1. Атрибут value

Используйте атрибут `value` для установки значения, которое ассоциируется с вариантом выбора. Если пользователь выбирает вариант, броузер передает ассоциированное с ним значение серверу. Если атрибут `value` не указан, значением варианта выбора будет содержимое тега `<option>`. В качестве примера рассмотрим варианты выбора в HTML-исполнении:

```
<option value=Dog>Dog
<option>Dog
```

У обоих – одно и то же значение. Первое указано в теге `<option>` явным образом, второе принято по умолчанию равным содержимому тега: «Dog».

9.8.2.2. Атрибут selected

По умолчанию ни один из вариантов в теге множественного выбора <select> не является выбранным и поэтому не включается в список параметров, когда пользователь передает форму серверу. Включайте атрибут selected в тег <option>, чтобы изначально принять один или несколько вариантов, от которых пользователь может впоследствии отказаться.

У HTML-версии атрибута selected значений нет, а XHTML-версия имеет значение selected="selected". В теге <select> типа «один из многих», если ни один из элементов не выбран явно, по умолчанию показывается[1] первый элемент.

9.8.2.3. Атрибут label

Обычно при отображении варианта выбора он помечается содержимым тега <option>. Если в теге присутствует атрибут label, его значение используется в качестве метки варианта выбора.

9.8.3. Тег <optgroup>

Меню в формах могут быть весьма большими, оттого их порой трудно отображать и использовать. В этих случаях имеет смысл группировать родственные варианты выбора, представляя их затем пользователю в виде каскада вложенных меню. Введенный в стандарте HTML 4.0 тег <optgroup> предоставляет, хотя и в ограниченной степени, эту возможность для HTML- и XHTML-форм.

<optgroup>

Функция:	Группирует родственные элементы <option> в меню <select>
Атрибуты:	class, dir, disabled, id, label, lang, onClick, onDblClick, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, style, title
Закрывающий тег:	</optgroup>; в HTML может опускаться
Содержит:	*optgroup_contents* (содержимое optgroup)
Может содержаться в:	*select_content* (содержимое списка выбора)

Применяйте тег <optgroup> только внутри тега <select>, в нем самом могут содержаться только теги <option>. Для каждого тега <optgroup> броузер создает в главном меню тега <select> свое субменю. К примеру, в HTML можно использовать <optgroup> для представления в форме меню штатов, сгруппированных по регионам (рис. 9.7):

[1] Показывается, но не выбирается. – *Примеч. науч. ред.*

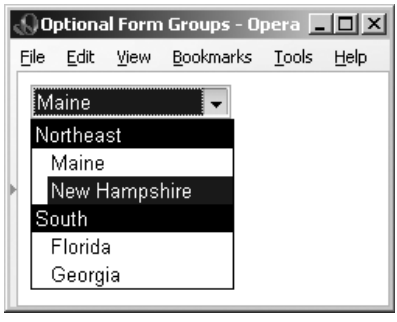

***Рис. 9.7.** Тег <optgroup> помогает организовать меню <select>*

```
<select name=state>
   <optgroup label=Northeast>
      <option>Maine
      <option>New Hampshire
      ...
   </optgroup>
   <optgroup label=South>
      <option>Florida
      <option>Georgia
   </optgroup>
   ...
</select>
```

Подобно броузеру Opera (см. рис. 9.7) другие популярные броузеры с графическим пользовательским интерфейсом делают отступы перед пунктами `<optgroup>` в прокручивающемся меню. Другие броузеры выделяют заголовки групп курсивом и полужирным шрифтом.

У тега `<optgroup>` есть существенный недостаток – он не допускает вложений, что заставляет обходиться только одним уровнем субменю. Это ограничение, предположительно, будет снято в будущей версии XHTML.

9.8.3.1. Атрибут label

Употребляйте атрибут `label` для определения названия субменю, представляемого пользователю. Вам следует придерживаться коротких названий и стремиться обеспечить возможность отображения меню на самых разных дисплеях.

9.9. Атрибуты формы общего назначения

Теги многих элементов формы могут содержать общие атрибуты, которые, как и для большинства других тегов, служат в основном для того, чтобы снабжать тег метками, устанавливать характеристики его отображения, расширять языковые возможности и обеспечивать его программные расширения.

9.9.1. Атрибуты id, name и title

Атрибут `id`, как и для большинства стандартных тегов, позволяет прикрепить к форме метку, представляющую собой уникальную в документе строку, предназначенную для обращения к этой форме со стороны программ (апплетов) и гиперссылок. Этот идентификатор отличается от имени, которое присваивается элементу формы при помощи атрибута `name`. Метки, определенные в атрибуте `id`, не передаются серверу при ее обработке.

Атрибут `title`, как и `id`, определяет заключенное в кавычки строковое значение, которое помечает элемент формы. Однако название помечает его только в качестве сегмента документа – это значение не может использоваться для ссылок на форму в апплетах, сценариях и гиперссылках. Броузер может применять значение этого атрибута во всплывающих подсказках или при невизуальном представлении формы. [атрибут id, 4.1.1.4] [атрибут title, 4.1.1.5]

9.9.2. Атрибуты событий

Как и большинство других элементов документа, элементы формы допускают использование стандартных атрибутов событий, инициируемых с помощью мыши и клавиатуры, которые распознают броузеры, следующие стандартам HTML 4/XHTML. Мы подробно описываем большинство этих атрибутов в главе 12. [обработчики событий JavaScript, 12.3.3]

9.9.3. Атрибуты class, style, lang и dir

Атрибут `style` создает встроенный в тег стиль для элементов формы, преодолевая любые другие действующие стилевые правила. Атрибут `class` позволяет применять к содержимому тега стиль, заранее определенный для данного класса тегов `<form>`. Значение атрибута `class` – это имя этого класса. [встроенные стили: атрибут style, 8.1.1] [стилевые классы, 8.3]

Атрибут `lang` позволяет определить язык, который употребляется в элементе формы, а атрибут `dir` дает возможность посоветовать броузеру, в каком направлении следует выводить текст. Значение атрибута `lang` – это двухбуквенный код языка по стандарту ISO, включающий необязательный языковой модификатор. Например, `lang=en-UK` сообщает броузеру, что в элементе формы используется английский язык, причем такой, на котором говорят и пишут в Соединенном Королевстве (United Kingdom). Предполагается, что броузер как-то отразит выбор языка в макете и типографских решениях.

Атрибут `dir` говорит броузеру, в каком направлении следует отображать содержимое элемента формы: слева направо (`dir=ltr`), как в английском или французском языках, или справа налево (`dir=rtl`) для таких языков, как китайский или иврит.

Атрибуты dir и lang поддерживаются популярными броузерами, хотя для конкретных языков какого-то особого поведения броузеров не предусмотрено. [атрибут dir, 3.6.1.1] [атрибут lang, 3.6.1.2]

9.9.4. Атрибуты tabindex, taborder (!) и notab (!)

По умолчанию все элементы (за исключением скрытых полей) включены в «табуляционную последовательность» формы. Когда пользователь нажимает клавишу <Tab>, броузер поочередно активизирует элементы внутри формы. Для большинства броузеров табуляционный порядок совпадает с порядком появления элементов в теге <form>. При помощи атрибута tabindex можно изменить как порядок, так и позицию элементов в табуляционной последовательности.

Чтобы переместить элемент в последовательности, установите значение этого атрибута равным порядковому номеру, под которым вы хотели бы его видеть (первый элемент последовательности имеет номер один). Если вы действительно намерены изменить табуляционную последовательность, мы предлагаем включить атрибут tabindex в каждый элемент формы, присваивая ему подходящие значения. Тогда вы будете уверены, что явным образом определили его место в табуляционной последовательности и что пользователя не ожидают сюрпризы, когда он будет перемещаться по форме с помощью клавиши <Tab>.

Значение атрибута tabindex – это положительные целые числа, указывающие позицию содержимого тега в общей табуляционной последовательности документа. Первыми членами этой последовательности являются те элементы, для которых явно указано значение tabindex, начиная с наименьших значений и идя по их возрастанию. Теги с одинаковыми значениями перебираются клавишей <Tab> в порядке их появления в документе. Все другие теги последовательности, такие как различные элементы форм и гиперссылки, идут после них в том порядке, в котором они встречаются в документе. Чтобы исключить элемент из табуляционной последовательности, установите tabindex равным нулю. При перемещении пользователя по форме с помощью клавиши <Tab> этот элемент будет пропускаться.

Internet Explorer ввел концепцию управления табуляционным порядком с принадлежащими только ему атрибутами taborder и notab. Атрибут taborder функционирует в точности так же, как атрибут tabindex, тогда как notab эквивалентен tabindex=0. Internet Explorer версии 5 и более поздних поддерживает tabindex, как и другие популярные броузеры. Поэтому мы настоятельно рекомендуем вам использовать атрибут tabindex, а не taborder.

9.9.5. Атрибут accesskey

Многие пользовательские интерфейсы продвигают идею горячих клавиш, короткие последовательности нажатий на которые дают быстрый доступ к пользовательскому интерфейсу. HTML 4 и XHTML под-

держивают такую возможность своим атрибутом accesskey. Значение атрибута accesskey – это один символ, который, если был нажат вместе с некоторой специальной клавишей, немедленно активизирует ассоциированный с ним элемент. Эта специальная клавиша варьируется в зависимости от пользовательского интерфейса. В Windows пользователи нажимают на клавишу <Alt>, а в Unix – на клавишу <Meta>.

К примеру, вставив accesskey="T" в элемент <textarea>, вы дадите возможность пользователю Windows активизировать эту область ввода нажатием пары клавиш <Alt>+<T>. Отметьте, что значением атрибута accesskey служит один символ и он чувствителен к регистру (заглавная «T» – это не то же самое, что ее строчная родственница).

Все популярные броузеры поддерживают атрибут accesskey. Тем не менее вы должны аккуратно протестировать свои горячие клавиши. Например, <Alt>+<f> в броузере Internet Explorer позволяет «перепрыгнуть» на тег, имеющий атрибут accesskey="f", а в Netscape эта комбинация клавиш открывает меню File (Файл).

Кроме того, обратите внимание, что опция accesskey не только переводит фокус, но и выделяет ассоциированный элемент формы. Поэтому если вы, например, ассоциируете атрибут accesskey с кнопкой-переключателем, то нажатие на соответствующую комбинацию клавиш не просто переместит фокус на эту кнопку-переключатель, но и выделит ее, как будто пользователь уже щелкнул мышью по этому элементу. То же самое справедливо для всех элементов формы: происходит перенос фокуса и выделения.

9.9.6. Атрибуты disabled и readonly

Стандарты HTML 4 и XHTML позволяют определить, но при этом «выключить» элемент формы, просто вставив в тег атрибут disabled. Выключенный элемент формы отображается, но не может быть активизирован при помощи клавиши <Tab> или мыши. Параметры такого элемента не передаются серверу при отправке формы.

Броузеры могут отображать выключенные элементы особенным образом, а также по-другому выводить связанные с ними метки. Популярные броузеры закрашивают серым неработающие кнопки-переключатели и кнопки отправки, как показано в следующем фрагменте HTML-кода (см. также рис. 9.8).

```
<form>
  Name:
    <input type=text name=name size=32 maxlength=80 readonly>
  <p>
  Sex:
    <input type=radio name=sex value="M" disabled> Male
    <input type=radio name=sex value="F" accesskey="z"> Female
  <p>
  Income:
    <select name=income size=1 disabled>
```

```
      <option>Under $25,000
      <option>$25,001 to $50,000
      <option>$50,001 and higher
    </select>
   <p>
  <input type=submit disabled>
</form>
```

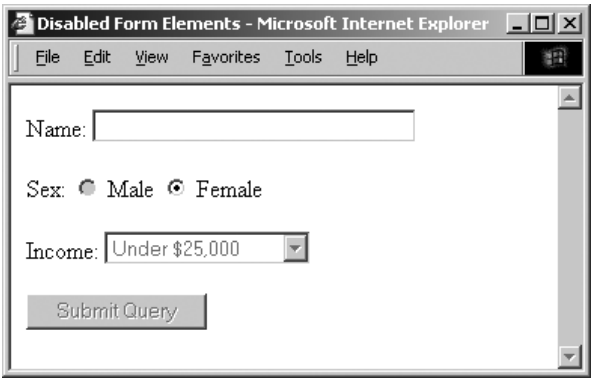

***Рис. 9.8.** Отключенные элементы формы закрашены серым цветом*

Подобным образом, текстовые варианты тега `<input>` и элемент `<textarea>`, в которых указано `readonly`, не могут быть изменены пользователем. Такие элементы остаются в табуляционной последовательности, их можно выбрать и их значение посылается серверу, когда пользователь отправляет форму. Он только не может изменить это значение. Так, по смыслу, элементы формы вида `readonly` – это видимый аналог элементов `<input type=hidden>`.

В чем смысл всех этих скрытых и нередактируемых элементов формы? В автоматизации. Автоматическое генерирование включенных и отключенных элементов формы позволяет вам подстраивать форму под конкретного пользователя. Например, если в одной из форм пользователь указывает свой пол, то имеет смысл в последующих формах хранить эту информацию в скрытом атрибуте. Некоторые элементы формы можно выводить, чтобы придать ей узнаваемость, а остальные – отключить, упростив тем самым навигацию по форме.

9.10. Группировка элементов формы и обеспечение их надписями

Обычный текст, используемый для создания надписей и меток, так же как и содержимое других видов, объясняющее форму, – статичен. Если не считать визуальной близости с элементами ввода в форме, эти метки и инструкции никак более не связаны с теми элементами, к которым они относятся. По этой причине формы нелегко понимать и по

ним трудно перемещаться, особенно тем людям, у которых неважно со зрением. Попробуйте сами. Вызовите простую форму, запрашивающую личные данные, закройте глаза и попробуйте найти место, куда следует ввести ваше имя.

Стандарт HTML 4.0 ввел три новых тега, которые облегчают перемещение по форме, особенно для пользователей с физическими недостатками. Они предоставляют возможность группировать элементы формы и снабжать эти группы метками. Предоставляют они и новые возможности для создания надписей для отдельных элементов формы. Предполагается, что броузер сможет обращаться с ними особым образом, выводить их содержимое через синтезатор речи, например, и что к ним будет обеспечен удобный доступ с пользовательской клавиатуры. Это случится, когда броузеры станут соответствовать стандартам HTML 4 и XHTML.

9.10.1. Тег <label>

Используйте тег `<label>` для установления связи между элементом формы, таким как текстовое поле ввода, и одной или несколькими текстовыми метками.

В соответствии с последними стандартами броузер должен обращаться с текстом метки специальным образом. Броузеры могут выбирать особый стиль их отображения (вы тоже, применяя таблицы стилей). И когда пользователь останавливается на метке, броузер должен автоматически активизировать соответствующий элемент формы.

<label>

Функция:	Создает метку для элемента формы
Атрибуты:	`accesskey`, `class`, `dir`, `for`, `id`, `lang`, `onBlur`, `onClick`, `onDblClick`, `onFocus`, `onKeyDown`, `onKeyPress`, `onKeyUp`, `onMouseDown`, `onMouseMove`, `onMouseOut`, `onMouseOver`, `onMouseUp`, `style`, `title`
Закрывающий тег:	`</label>`; присутствует обязательно
Содержит:	*label_contents* (содержимое метки)
Может содержаться в:	*form_content* (содержимое формы)

9.10.1.1. Явное и неявное привязывание меток

Одна или несколько меток могут связываться с элементом формы одним из двух способов: неявно, путем включения элемента формы в виде содержимого в теге `<label>`, или явно, указанием ID-элемента в атрибуте `for` тега `<label>`.

Примеры в XHTML:

```
<label for="SSN">Номер социального страхования:</label>
<input type="text" name="SocSecNum" id="SSN" />
```

```
<label>Дата рождения: <input type="text" name="DofB" /></label>
```

Первая метка явно связывает текст «Номер социального страхования» с полем ввода текста (SocSecNum), поскольку значение ее атрибута `for` идентично `id` поля ввода, «SSN». Вторая метка («Дата рождения») не требует атрибута `for`, как и ассоциированное с ней поле не нуждается в атрибуте `id`, поскольку они неявно соединены тем, что тег `<input>` помещен внутри тега `<label>`.

Будьте осторожны и не путайте атрибутов `name` и `id`. Первый создает имя элемента, используемое обработчиком формы со стороны сервера, второй указывает имя, которое может применяться тегом `<label>` и в URL. Заметьте также, что хотя метка может ссылаться только на один элемент формы, несколько меток могут ссылаться на один элемент. Это дает возможность перемещать пользователя к конкретному полю ввода из разных областей документа.

9.10.1.2. Другие атрибуты метки

К меткам относятся многие общие атрибуты тегов: отображения, доступа и событий, описанные в разделе 9.9. В дополнение к атрибутам событий стандартов HTML 4 и XHTML метки поддерживают также атрибуты `onfocus` и `onblur`.

9.10.2. Формирование групп

Помимо создания меток отдельных элементов, можно объединить в группу ряд элементов формы и определить метки для этой группы при помощи тегов `<fieldset>` и `<legend>`. Здесь снова стандарты HTML 4 и XHTML пытаются сделать формы более доступными, особенно для пользователей с физическими недостатками. Явное предназначение группы элементов позволяет особым образом отображать содержимое формы и вообще обращаться с ним иначе.

9.10.2.1. Тег <fieldset>

Тег `<fieldset>` заключает в себе часть содержимого формы, создавая группу связанных элементов. У тега `<fieldset>` нет обязательных и свойственных только ему атрибутов.

<fieldset>

Функция:	Группирует в форме родственные элементы
Атрибуты:	`class`, `dir`, `id`, `lang`, `onClick`, `onDblClick`, `onKeyDown`, `onKeyPress`, `onKeyUp`, `onMouseDown`, `onMouseMove`, `onMouseOut`, `onMouseOver`, `onMouseUp`, `style`, `title`
Закрывающий тег:	`</fieldset>`; присутствует обязательно
Содержит:	*form_content* (содержимое формы)
Может содержаться в:	*form_content* (содержимое формы)

Когда группа элементов формы находится в теге <fieldset>, броузер может отображать их особым способом – заключая группу в рамку, применяя 3D-эффекты или даже создавая субформу, содержащую элементы группы.

9.10.2.2. Тег <legend>

Используйте тег <legend> для создания метки группы элементов формы. Этот тег может находиться только внутри тега <fieldset>. Как и в случае <label>, содержимое тега <legend> может специальным образом трактоваться броузером в соответствии со стандартами HTML 4 и XHTML, активизируя ассоциированную с ним группу, когда пользователь выбирает легенду, облегчая тем самым доступ к набору <fieldset>.

<legend>

Функция:	Создает легенду для набора полей формы
Атрибуты:	accesskey, align, class, dir, id, lang, onClick, onDblClick, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, style, title
Закрывающий тег:	</legend>; в HTML может опускаться
Содержит:	*legend_content* (содержимое легенды)
Может содержаться в:	*form_content* (содержимое формы)

В дополнение к свойственным множеству элементов форм атрибутам, описанным в разделе 9.9, тег <legend> допускает атрибуты acesskey и align. Значением align могут быть top, bottom, left и right, указывающие броузеру, как должна располагаться легенда по отношению к группе элементов из <fieldset>.

Собирая вместе описанные теги, мы представляем ниже легенду к группе, содержащей небольшое количество элементов, у каждого из которых есть своя метка. Посмотрите, какой аккуратной рамочкой окружает Firefox группу элементов, включая в нее легенду, никак иначе не форматируя содержимое тега <fieldset> (рис. 9.9). Очевидно, что вам придется выполнить некоторое форматирование самостоятельно:

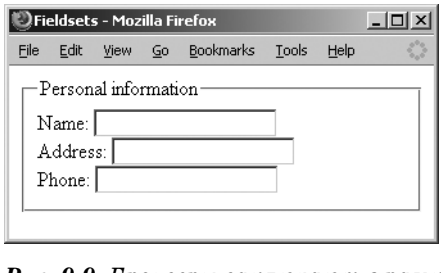

Рис. 9.9. *Броузеры заключают в рамку всю группу элементов*

```
<fieldset>
   <legend>Personal information</legend>
   <label>Name:<input type="text" /></label>
   <label>Address:<input type="text" /></label>
   <label>Phone:<input type="text" /></label>
```

9.11. Эффективные формы

Сделанные надлежащим образом формы могут предоставлять вашим читателям эффективный пользовательский интерфейс. Используя некоторые приемы программирования со стороны сервера, вы можете применять формы для индивидуального подхода в документах к нуждам, вкусам и возможностям каждого пользователя, существенно повышая тем самым ценность ваших сетевых презентаций.

9.11.1. Ограниченность броузеров

В отличие от других графических пользовательских интерфейсов, отображение при помощи броузера является статичным. Броузеры не предоставляют почти никаких возможностей по проверке внесенных данных в реальном времени или по корректировке значений формы в соответствии с пользовательским вводом, не оказывая ни помощи, ни руководства.[1] Следовательно, плохо сконструированные формы будет, скорее всего, трудно заполнять.

Убедитесь, что ваши формы максимально помогают пользователям. Например, подбирайте размеры полей ввода текста так, чтобы намекнуть пользователю на приемлемую форму – пятисимвольные (или девятисимвольные) поля для почтового индекса[2], например. Применяйте где можно выключатели, переключатели и списки выбора, сужая число доступных вариантов решений.

Проверьте также, что вы адекватно документировали вашу форму. Объясните, как следует ее заполнять, предоставив примеры для каждого поля. Вставьте подходящие гиперссылки к документации, описывающей любое поле, если это необходимо.

Когда форма отправлена, убедитесь, что приложение со стороны сервера исчерпывающим образом проверяет правильность введенных пользователем данных. Обнаружив погрешности, составляйте понятные

---

[1] Это не совсем верно. Хотя ни HTML, ни XHTML не предоставляют возможностей для проверки формы на месте или оказания помощи пользователю, к элементам формы можно присоединять апплеты Java и сценарии JavaScript, которые очень неплохо справляются с проверкой заполнения формы, обновлением полей для ввода данных пользователем и с инструкциями по перемещению в форме.

[2] На территории стран бывшего СССР, например, принят шестисимвольный почтовый индекс. – *Примеч. науч. ред.*

сообщения об ошибках и варианты их исправления. Один из самых удручающих моментов при заполнении форм состоит в необходимости начинать все заново, когда бы сервер ни обнаружил ошибку. Чтобы облегчить задачу пользователя и избавиться от безобразной избыточности действий, не пожалейте дополнительного времени и ресурсов сервера, чтобы возвращать форму, в которой просто помечены неправильно заполненные поля, с предложением их исправить.

Хотя эти рекомендации потребуют от вас значительных усилий, они окупятся многократно, облегчая жизнь пользователям. Помните, что вы создаете форму однажды, а заполняться она будет тысячи и миллионы раз.

9.11.2. Дисплеи с ограниченными возможностями

Хотя большая часть персональных компьютеров усовершенствована, предоставляя теперь значительно лучшее разрешение, чем 600×480, которое было обычным, когда мы готовили первое издание этой книги, многие устройства (WebTV, сотовые телефоны со встроенными броузерами, портативные компьютеры) требуют, чтобы проектирование форм оставалось консервативным. Наилучший компромисс – это предполагать, что окно просмотра документов имеет приблизительно 75 символов в ширину и от 30 до 50 строк в высоту.[1] Вам следует разрабатывать формы (и все документы), чтобы ими было удобно пользоваться на экранах такого размера.

Вам нужно структурировать формы так, чтобы они естественно распадались на два или три логических раздела. Пользователь может заполнить один раздел и перейти на следующую страницу, заполнить второй раздел – и на следующую, и т. д.

Следует также избегать широких элементов ввода. Довольно трудно иметь дело с полем и многострочной областью, в которых прокручивается текст, даже если не приходится гонять по горизонтали сам документ, пытаясь увидеть не поместившиеся на экране части элементов ввода.

9.11.3. Соображения о пользовательском интерфейсе

Решив создать форму, вы мгновенно оказываетесь также в роли конструктора пользовательского интерфейса. Хотя полное обсуждение разработки такого интерфейса выходит за пределы этой книги, полезно знать несколько основных правил дизайна, чтобы создавать эффективные и привлекательные формы.

---

[1] Дисплеи некоторых устройств, например сотовых телефонов, так малы, что на них умещается не более четырех строк. Более удачный подход (обсуждение которого не является темой этой книги) состоит в преобразовании вашего документа с помощью XSLT (Extensible Stylesheet Transformations, преобразования XSL).

В любом пользовательском интерфейсе одновременно присутствуют несколько уровней. Формы не исключение. На нижнем уровне ваш мозг распознает образы в документе, пытаясь классифицировать элементы формы. На следующем уровне вы читаете текстовые указания и подсказки, стараясь понять, что следует вводить. На самом верхнем уровне вы пробуете выполнить поставленную задачу, пользуясь интерфейсом как орудием.

Хорошая форма отвечает всем этим трем потребностям человеческого восприятия. Элементы ввода должны быть организованы в логические группы, чтобы рассудок мог обработать макет формы, выделив в нем куски, содержащие связанные между собой поля. Последовательные, хорошо написанные подсказки и вспомогательный текст помогают пользователю и руководят им, чтобы он вводил правильную информацию. Кроме того, текстовые подсказки напоминают ему о выполняемой задаче и дополнительно его мотивируют.

9.11.4. Создание работающей формы

Пользователи обрабатывают формы в предсказуемом порядке, один элемент за другим, и ищут следующий элемент, покончив с предыдущим. Стремясь к согласию с этим процессом, вы должны конструировать формы так, чтобы одно поле естественно приводило к другому, а родственные поля были собраны вместе. Подобным образом группы должны естественно вести пользователя к другим группам, и их оформление должно быть единообразным.

Просто собрать вместе ряд полей – это не значит создать эффективную форму. Вы должны поставить себя на место своих пользователей, которые впервые ее заполняют. Протестируйте эту форму на ничего не подозревающих друзьях и коллегах, прежде чем представите ее на всеобщее обозрение. Легко ли понять назначение формы? Откуда начинать ее заполнение? Сможет ли пользователь найти кнопку, которую нужно щелкнуть, чтобы отправить форму? Предоставлена ли возможность подтверждения решений? Понимают ли пользователи, чего от них ожидает каждое поле?

Ваши формы должны естественно проводить пользователя через весь процесс предоставления приложению необходимых данных. Вам не следует спрашивать адрес прежде, чем вы спросите имя пользователя. Другие правила могут определять порядок других групп элементов ввода. Чтобы понять, работает ли форма действительно, непременно просмотрите ее на нескольких броузерах, попросите ряд людей ее заполнить и отозваться о ее эффективности.

9.11.5. Улучшенный макет старой формы

На первый взгляд, основное правило HTML и XHTML – содержание, а не форма – находится в прямом противоречии с основным правилом создания хороших интерфейсов – точный, последовательный макет.

Если даже и так, существует возможность использовать некоторые элементы, чтобы значительно улучшить макет и удобство чтения большинства форм.

Традиционный макет страницы использует сетку столбцов для выравнивания элементов на странице. Возникающие при этом воображаемые вертикальные и горизонтальные края смежных элементов придают странице организованный и упорядоченный вид и облегчают ее просмотр и чтение.

С HTML и XHTML это труднее, но можно выполнить макетирование подобного рода для форм. Например, стоит группировать родственные элементы и отделять группы друг от друга пустыми абзацами и горизонтальными линейками.

Вертикальное выравнивание более трудно, но не невозможно. В целом, формы легче заполнять, если элементы ввода упорядочены по вертикали и выровнены по общему краю. Один популярный макет формы выравнивает левые края элементов ввода, а метки элементов помещает рядом с ними, слева. Это делается с помощью размещения элементов и их меток в таблицах. Ниже представлен наш старый пример HTML-формы (результат на рис. 9.10), в которой теперь метки помещаются в первом столбце и соответствующие элементы – во втором:

```
<form method=POST action="http://www.kumquat.com/demo">
  <table border=0>
    <tr valign=top>
      <td align=right>Name:</td>
      <td align=left><input type=text name=name size=32 maxlength=80>
      </td>
    </tr>
    <tr valign=top >
      <td align=right>Sex:</td>
      <td align=left>
        <input type=radio name=sex value="M"> Male <br>
        <input type=radio name=sex value="F"> Female
      </td>
    </tr>
    <tr valign=top >
      <td align=right>Income:</td>
      <td align=left>
        <select name=income size=1>
          <option>Under $25,000
          <option>$25,001 to $50,000
          <option>$50,001 and higher
        </select>
      </td>
    </tr>
    <tr valign=top>
      <td colspan=2 align=center>
        <input type=submit value="Submit Query">
```

```
      </td>
    </tr>
  </table>
</form>
```

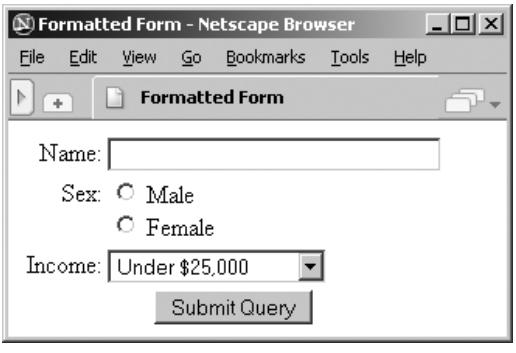

Рис. 9.10. *Использование общего вертикального поля для выравнивания элементов формы*

Отметьте, глядя на результат (см. рис. 9.10), что таблица поместила каждый элемент ввода на отдельной строке. Атрибуты `align` в ячейках таблицы сдвигают метки направо, а элементы – налево. В результате слияния ячеек в последней строке таблицы кнопка отправки центрирована по отношению к форме в целом. В общем, такое использование таблиц облегчает макетирование форм и позволяет его проводить единообразно в разных документах. Если вас затруднил этот пример, откройте главу 10, в которой таблицы раскрываются во всей своей красе.

Вы можете найти другие шаблоны для макетирования ваших форм. Весь смысл здесь в том, чтобы отыскать стиль макета, работающий на разных броузерах, и придерживаться его. Хотя HTML и XHTML обладают ограниченными средствами для управления макетированием и позиционированием, пользуйтесь тем, что есть, стараясь сделать ваши формы более привлекательными и удобными для посетителей.

9.12. Программирование форм

Если вы конструируете формы, рано или поздно вам понадобится создавать приложение со стороны сервера для их обработки. Не паникуйте. Нет ничего волшебного в программировании со стороны сервера, и это не слишком трудно. При некоторой настойчивости и после небольшой практики вы будете штамповать обрабатывающие приложения.

Самый важный совет, который мы можем дать по поводу программирования форм, легко запомнить – копируйте чужие работы. Писать обрабатывающее форму приложение с чистого листа очень трудно. Копировать работающее приложение и модифицировать его для нужд своей формы – гораздо легче.

К счастью, поставщики серверов знают об этом и обычно поставляют образцы обрабатывающих формы приложений вместе с сервером. Покопайтесь немного в каталоге *cgi-src*, и вы найдете ряд полезных примеров, которые сможете скопировать и применять.

Не стоит даже рассчитывать воспроизвести все то полезное, что поставляется вместе с вашим сервером, не можем мы и предоставить полное руководство по программированию форм. Все, что мы можем сделать, – это предложить простые примеры GET- и POST-приложений, давая вам почувствовать характер этой работы и, может быть, подталкивая вас в верном направлении.

Прежде чем мы начнем, примите во внимание, что не все серверы вызывают эти приложения одинаково. Наши примеры покрывают широкий класс серверов, произошедших от оригинального HTTP-сервера, созданного в центре NCSA (National Center for Supercomputing Applications, Национальный центр по использованию суперкомпьютеров). Они также должны работать с очень популярным и легко доступным сервером Apache. Во всех случаях обращайтесь за разъяснениями к серверной документации. Подробную информацию можно найти в книгах: «CGI Programming with Perl»[1] Скотта Гулича (Scott Guelich), Шишира Гундаварама (Shishir Gundavaram) и Гюнтера Бирзнекса (Gunther Birznieks), «Webmaster in a Nutshell»[2] Стивена Спейнауэра (Stephen Spainhour) и Роберта Экштейна (Robert Eckstein), выпущенных издательством «O'Reilly».

Альтернативой CGI-программированию является модель Java-сервлетов[3], описанная в книге «Java Servlet Programming»[4], принадлежащей перу Джейсона Хантера (Jason Hunter) и Вильяма Кроуфорда (William Crawford) (O'Reilly). Сервлеты могут применяться для обработки как GET-, так и POST-форм, хотя они посвящены более общим вопросам. В этой книге нет примеров сервлетов.

9.12.1. Возвращение результатов

Прежде чем начать, мы должны обсудить, как завершается работа приложения со стороны сервера. Все приложения со стороны сервера передают свои результаты назад серверу (и далее – пользователю), направляя их в стандартный вывод приложения в виде файла в MIME-кодировке. Следовательно, первая строка вывода должна быть MIME-

---

[1] С. Гулич, Ш. Гундаварам, Г. Бирзнекс «CGI программирование на Perl», издательство «Символ-Плюс», 2001 г.

[2] С. Спейнауэр, Р. Экштейн «Справочник вебмастера», издательство «Символ-Плюс», 2001 г.

[3] Сервлет (servlet) – приложение на Java (по аналогии с апплетом), но запускаемое на стороне сервера, а не на стороне клиента. – *Примеч. науч. ред.*

[4] Д. Хантер, В. Кроуфорд «Программирование Java-сервлетов», издательство «Символ-Плюс», II кв. 2002 г.

описателем типа содержимого. Если ваше приложение возвращает HTML-документ, первая строка это:

```
Content-type: text/html
```

Вторая строка должна быть совершенно пустой. Ваше приложение может возвращать содержимое и другого типа – только укажите правильно MIME-тип. Изображениям GIF, например, предшествует:

```
Content-type: image/gif
```

Собственно текст, который не должен интерпретироваться как HTML, может возвращаться как:

```
Content-type: text/plain
```

Это часто оказывается полезным, когда приходится возвращать вывод других команд, генерирующих простой текст вместо HTML.

9.12.2. Обработка GET-форм

Когда применяется метод GET, броузер передает параметры формы в виде части URL-адреса, который вызывает приложение формы со стороны сервера. Типичный вызов приложения типа GET может использовать примерно такой URL:

```
http://www.kumquat.com/cgi-bin/dump_get?name=bob&phone=555-1212
```

Когда сервер *www.kumquat.com* обрабатывает этот URL, он вызывает приложение под названием *dump-get*, хранящееся в каталоге *cgi-bin*. Все, что идет после знака вопроса, передается приложению в качестве параметров.

Далее дело может обстоять по-разному, в соответствии с природой URL типа GET. В то время как формы помещают в URL пары имя/значение, можно вызывать приложения типа GET с URL, содержащим только значения. Таким образом:

```
http://www.kumquat.com/cgi-bin/dump_get?bob+555-1212
```

это тоже допустимый вызов с параметрами, разделенными знаком «плюс» (+). Такой вызов обычно встречается, когда на приложение ссылается поисковый документ с тегом `<isindex>`. Параметры, набранные пользователем в поле ввода, передаются приложению со стороны сервера как безымянные параметры, разделенные знаком «плюс».

Если вызывать GET-приложение с поименованными параметрами, сервер передаст эти параметры приложению одним способом. Безымянные параметры передаются по-другому.

9.12.2.1. Использование поименованных параметров с GET-приложениями

Поименованные параметры передаются GET-приложениям путем создания переменной окружения `QUERY_STRING` и присваивания ей в каче-

стве значения всей той части URL, которая следует за вопросительным знаком. Если взять наш предыдущий пример, значением QUERY_STRING будет:

```
name=bob&phone=555-1212
```

Ваше приложение должно получить эту переменную и выделить из нее пары имя/значение. К счастью, большинство серверов поставляются вместе с рядом стандартных программ, выполняющих эту работу, так что простая C-программа, просто распечатывающая параметры, может выглядеть примерно так:

```
#include <stdio.h>
#include <stdlib.h>

#define MAX_ENTRIES 10000

typedef struct {char *name;
                char *val;
               }entry;

char *makeword(char *line, char stop);
char x2c(char *what);
void unescape_url(char *url);
void plustospace(char *str);

main(int argc, char *argv[])

{  entry entries[MAX_ENTRIES];
    int num_entries, i;
    char *query_string;

/* Получить значение переменной окружения QUERY_STRING */
    query_string = getenv("QUERY_STRING");

/* Извлечь параметры, построить таблицу вхождений */
    for (num_entries = 0; query_string[0]; num_entries++) {
       entries[num_entries].val = makeword(query_string, '&');

       plustospace(entries[num_entries].val);
       unescape_url(entries[num_entries].val);
       entries[num_entries].name =
          makeword(entries[num_entries].val, '=');
       }

/* Выдать "рыбу" HTML */
    printf("Content-type: text/html\n");
    printf("\n");

    printf(<html>);
    printf(<head>);
    printf("<title>Именованные параметры</title>\n");
    printf("</head>");
    printf(<body>);
    printf("Вы ввели следующие параметры:\n");
    printf("<ul>\n");
```

```
/* Выдать полученные от пользователя параметры */
    for(i = 0; i < num_entries; i++)
        printf("<li> %s = %s\n", entries[i].name,
                  entries[i].val);

/* И завершить выдачу "рыбы" */
    printf("</ul>\n");
    printf("</body>\n");
    printf("</html>\n");
}
```

Эта программа начинается с нескольких объявлений, которые определяют стандартные функции, просматривающие строку символов и выделяющие имена параметров и их значения.[1] Тело программы получает значение переменной окружения `QUERY_STRING`, применяя системный вызов `getenv()`, употребляет стандартные функции для выделения параметров из этого значения и затем порождает простой HTML-документ, который отображает эти значения для пользователя.

В реальных приложениях вы должны вставить действительные коды обработки параметров после их выделения и перед генерацией HTML. Разумеется, вам также нужно изменить выдаваемый HTML-текст в соответствии с функцией вашего приложения.

9.12.2.2. Использование безымянных параметров с GET-приложениями

Безымянные параметры передаются приложению как параметры командной строки. Это делает разработку приложения со стороны сервера почти тривиальной. Вот простая последовательность команд, печатающая значения параметров:

```
#!/bin/csh -f
#
# Вывод заданных пользователем неименованных параметров типа GET

echo "Content-type: text/html"
echo
echo '<html>'
echo '<head>'
echo '<title>Вывод неименованных параметров</title>'
echo '</head>'
echo '<body>'
echo 'Вы задали следующие параметры:'
echo '<ul>'

foreach i ($*)
   echo '<li>' $i
end
```

[1] Эти программы обычно поставляются вместе с сервером. Они входят в стандартные библиотеки C или UNIX.

```
echo '</ul>'
echo '</body>'

exit 0
```

Мы снова следуем тому же общему правилу: выводим типовой заголовок документа, включающий MIME-тип содержимого, за ним – параметры и стандартное завершение. Чтобы преобразовать это в реальное приложение, замените цикл `foreach` командами, которые что-то действительно делают.

9.12.3. Обработка POST-форм

Приложения, принимающие параметры HTML/XHTML для метода POST, будут читать закодированные параметры из своего стандартного ввода. Подобно GET-приложениям с поименованными параметрами они могут воспользоваться стандартными функциями сервера, чтобы разобрать эти параметры:

```
#include <stdio.h>
#include <stdlib.h>

#define MAX_ENTRIES 10000

typedef struct {char *name;
                char *val;
               } entry;

char *makeword(char *line, char stop);
char *fmakeword(FILE *f, char stop, int *len);
char x2c(char *what);
void unescape_url(char *url);
void plustospace(char *str);

main(int argc, char *argv[])

{  entry entries[MAX_ENTRIES];
    int num_entries, i;

/* Извлечь параметры из stdin, построить таблицу вхождений */
    for (num_entries = 0; !feof(stdin); num_entries++) {
       entries[num_entries].val = fmakeword(stdin, '&', &cl);
       plustospace(entries[num_entries].val);
       unescape_url(entries[num_entries].val);
       entries[num_entries].name =
          makeword(entries[num_entries].val, '=');
       }
 /* Выдать "рыбу" HTML */
    printf("Content-type: text/html\n");
    printf("\n");
    printf(<html>);
    printf(<head>);
    printf("<title>Именованные параметры</title>\n");
    printf("</head>");
```

```
    printf(<body>);
    printf("Вы ввели следующие параметры:\n");
    printf("<ul>\n");

/* Выдать полученные от пользователя параметры */
    for(i = 0; i < num_entries; i++)
        printf("<li> %s = %s\n", entries[i].name,
                  entries[i].val);

/* And close out with more boilerplate */
    printf("</ul>\n");
    printf("</body>\n");
    printf("</html>\n");
}
```

И снова мы следуем той же общей схеме. Первые строки программы состоят из объявлений разных стандартных процедур, необходимых для разбора параметров, и объявления структуры, в которой будет содержаться их список. Настоящий код начинается с чтения списка параметров из стандартного ввода и построения списка имен и значений параметров в массиве под названием `entries`. Когда с этим покончено, в стандартный вывод печатается типовой заголовок документа, за которым идут параметры и шаблонное завершение.

Как и другие примеры, эта программа вполне пригодна для проверки параметров, передаваемых серверному приложению, пока вы новичок в формах и отладке приложений. Ее также можно использовать как основу для других приложений, вставляя подходящий обрабатывающий код после того, как список параметров построен, и изменяя вывод программы, чтобы отправлять пользователю соответствующие результаты.

- Стандартная модель таблиц
- Основные теги таблицы
- Новейшие теги таблицы
- За пределами обычных таблиц

10

Таблицы

Из всех расширений, нашедших свое место в HTML и XHTML, ни одно так не приветствовалось, как таблицы. Они не только хороши для представления данных, их роль важна и в управлении макетом документа. Творческое использование таблиц, как мы покажем в этой главе, может неплохо оживить унылый документ. К различным элементам таблицы можно применять все CSS-стили, получая в результате документ, внешний вид которого не уступает печатным публикациям.

10.1. Стандартная модель таблиц

Стандартная модель таблиц довольно проста: таблица – это набор слов и чисел, выстроенных в строки и столбцы ячеек. Большинство ячеек содержат данные, другие – заголовки строк и столбцов, описывающие данные.

Определите таблицу и заключите все ее элементы между тегом `<table>` и ее закрывающим тегом `</table>`. Элементы таблицы, такие как элементы данных, заголовки строк и столбцов, подписи, имеют свои теги разметки. Двигаясь сверху вниз и слева направо, последовательно описываются заголовок и данные в каждой ячейке столбцов таблицы.

Новейшие стандарты предоставляют также расширенный состав атрибутов тегов, поддерживающихся основными броузерами, которые прежде были популярными расширениями HTML. Они улучшают внешний вид таблиц, и среди прочих возможностей имеются специальные средства для выравнивания табличных данных и заголовков, настройки параметров рамки и табличных линеек, автоматической установки размеров ячеек, в зависимости от их содержимого. Различные популярные броузеры имеют слегка различающиеся наборы атрибутов таблиц – мы укажем на эти различия по ходу изложения.

10.1.1. Содержимое таблиц

В ячейку таблицы можно поместить почти все, что может содержаться в теле HTML- или XHTML-документа, включая изображения, формы, линейки, заголовки и даже другие таблицы. Броузер обращается с каждой ячейкой как с целым окном, заполняя ее пространство потоком содержимого, но с некоторыми ограничениями и дополнениями в том, что касается форматирования.

10.1.2. Пример таблицы

Вот небольшой пример, показывающий, как выглядит исходный текст HTML-таблицы и как таблица выводится броузером (рис. 10.1).

Что более важно, этот пример показывает базовую структуру таблицы, в которой можно увидеть многие элементы, порядок и синтаксис тегов, атрибуты и т. д. и в которую можно заглядывать при чтении дальнейших подробных описаний:

```
<table border cellspacing=0 cellpadding=5>
  <caption align=bottom>
    Kumquat versus a poked eye, by gender</caption>
  <tr>
    <td colspan=2 rowspan=2></td>
    <th colspan=2 align=center>Preference</th>
  </tr>
  <tr>
    <th>Eating Kumquats</th>
    <th>Poke In The Eye</th>
  </tr>
  <tr align=center>
    <th rowspan=2>Gender</th>
    <th>Male</th>
```

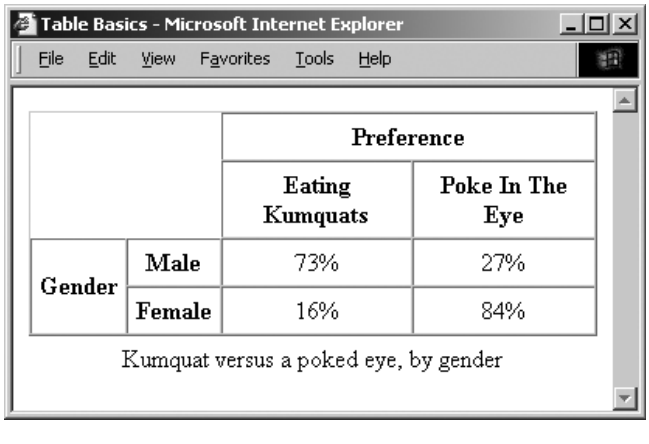

Рис. 10.1. *Пример HTML-таблицы*

```
    <td>73%</td>
    <td>27%</td>
  </tr>
  <tr align=center>
    <th>Female</th>
    <td>16%</td>
    <td>84%</td>
  </tr>
</table>
```

10.1.3. Отсутствующие возможности

Некогда стандартные HTML-таблицы не предоставляли возможностей, присущих полноценным инструментам создания таблиц, которые можно найти в популярных текстовых редакторах. Вместо этого популярные броузеры, такие как Internet Explorer и Netscape, поддерживали свои собственные расширения языка.

Чего не хватало, так это поддержки скользящих[1] заголовков строк вверху (в шапке) и внизу (в подвале) таблицы, особенно полезных при отображении больших таблиц. Не хватало и управления линейками в таблице и разделами таблиц.

В наши дни некоторые броузеры отстали от стандартов; при этом HTML 4 и XHTML включили в стандарт многие расширения и предложили новые решения.

10.2. Основные теги таблицы

Все многообразие таблиц можно создать всего лишь пятью тегами: тегом `<table>`, который выделяет таблицу и ее элементы из содержимого тела документа; тегом `<tr>`, который определяет строку таблицы; тегами `<th>` и `<td>`, определяющими заголовки в таблице и ячейки данных, и тегом `<caption>`, который определяет название таблицы. Помимо этих основных тегов существуют теги `<colgroup>`, `<tbody>`, `<thead>` и `<tfoot>`, позволяющие определять целые разделы таблиц и управлять ими, в частности с их помощью создаются скользящие заголовки. У каждого из тегов есть обязательные и необязательные атрибуты, некоторые из которых действуют не только на сам тег, но и на связанные с ним.

10.2.1. Тег <table>

Тег `<table>` и его закрывающий тег `</table>` описывают таблицу в теле документа. Если другое расположение таблицы в окне броузера не предписано таблицами стилей или выравниваниями абзацев или разделов, броузер, встретив тег `<table>`, останавливает поток текста, пере-

[1] Скользящей мы называем ту часть таблицы, которая автоматически повторяется на каждой странице, если таблица целиком не помещается на одной странице. – *Примеч. науч. ред.*

<table>	
Функция:	Определяет таблицу
Атрибуты:	align, background, bgcolor, border, bordercolor, bordercolordark, bordercolorlight, cellpadding, cellspacing, class, cols, dir, frame, height, hspace, id, lang, nowrap, onClick, onDblClick, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, rules, style, summary, title, valign, vspace, width
Закрывающий тег:	</table>; присутствует обязательно
Содержит:	*table_content* (содержимое таблицы)
Может содержаться в:	*блоке*

водит строку, начинает на новой строке таблицу, а по ее окончании возобновляет поток текста со следующей за таблицей строки.

Единственно допустимыми внутри тега <table> являются один или несколько тегов <tr>, которые определяют каждую строку таблицы, а также различные теги разделов таблицы: <thead>, <tfoot>, <tbody>, <col> и <colgroup>.

10.2.1.1. Атрибут align (нежелателен)

Стандарты HTML 4 и XHTML признали нежелательным этот атрибут в пользу свойства align, предоставляемого каскадными таблицами стилей (CSS). Пока он все же остается популярным и поддерживается популярными броузерами.

Подобно изображениям таблицы являются прямоугольными объектами, выровненными каким-либо образом относительно потока текста. Обычно броузер выравнивает таблицу по левому краю, прижимая ее левый край к левому краю окна броузера. Или же таблица может быть центрирована, если находится под действием тега , центрированного абзаца или раздела. В отличие от изображений, однако, таблицы не являются внедренными в поток текста объектами. Текстовое содержимое обычно располагается над ней и под ней, но не сбоку от нее. Такое поведение таблицы при отображении можно изменить, применяя атрибут align или каскадное определение стиля для тега <table>.

Атрибут align принимает одно из трех значений: left, right или center, означающих, что таблица должна быть выровнена либо по левому или по правому полю потока текста, при этом текст обтекает таблицу; либо по центру, тогда текст располагается сверху и снизу.

Отметьте, что атрибут align в теге <table> отличается от употребленного в тегах элементов таблицы <tr>, <td> и <th>. В последних атрибут align управляет выравниванием текста внутри ячеек таблицы, а не выравниванием таблицы в окружающем ее потоке текста.

10.2.1.2. Атрибуты bgcolor и background

Можно сделать так, чтобы фон таблицы был отличен от фона документа, применив атрибут `bgcolor` тега `<table>`. Значением атрибута `bgcolor` должен быть либо RGB-триплет, либо стандартное название цвета. Синтаксис кодов цветов и допустимые названия цветов можно найти в приложении G.

Популярные броузеры придадут этот цвет фону всех ячеек в таблице (включая название таблицы). Используя атрибут `bgcolor` или атрибут `style` для отдельных строк и ячеек, можно установить для них другой цвет фона.

Атрибут `background` – не входящее в стандарт расширение, поддерживаемое популярными броузерами, – принимает в качестве значения URL изображения, которым выкладывается фон таблицы. Если изображение больше таблицы, оно будет обрезано. Используя этот атрибут с таблицами без обрамления, можно поместить текст поверх содержащегося в документе изображения.

10.2.1.3. Атрибут border

Необязательный в теге `<table>` атрибут `border` предписывает броузеру провести линии вокруг таблицы, ее строк и ячеек. По умолчанию обрамление отсутствует. Можно присвоить атрибуту `border` значение, но HTML этого не требует. Если значение атрибута не определено, он просто включает обрамление и некоторый набор характеристик, принятых по умолчанию. В XHTML для достижения того же результата пишите `border="border"`. В HTML или XHTML, присвоив атрибуту `border` целое значение, вы достигнете трехмерного представления выпуклой рамки шириной в указанное количество пикселов.

10.2.1.4. Атрибуты frame и rules

Для Netscape 4 атрибут `border` давал либо все, либо ничего, определяя одновременно ширину и вид как рамки вокруг таблицы, так и линеек, разделяющих ячейки таблицы. Internet Explorer версии 4 и более поздних версий, Netscape версии 6 и более поздних, а также Firefox и Opera позволяют по отдельности модифицировать линии, составляющие рамку вокруг таблицы (`frame`) и обрамление ячеек (`rules`).

Принадлежащий стандарту атрибут `frame` модифицирует действие атрибута `border` для линий, окружающих таблицу. Принимаемое по умолчанию значение – то, что вы получаете, если вовсе не используете `frame`, – это `box`, которое предписывает броузеру провести линии со всех четырех сторон таблицы. Значение `border` делает то же, что и `box`. Значение `void` убирает рамку. Значения атрибута `frame` – `above`, `below`, `lhs` и `rhs` – влекут появление рамочных линий сверху, снизу, слева и справа от таблицы соответственно. Значение `vsides` проводит линии рамки слева и справа от таблицы (по вертикальным сторонам рамки); значение `hsides` проводит линии рамки по верхней и нижней границам таблицы.

В соответствии со стандартами, которые теперь имеют поддержку со стороны всех популярных броузеров, при помощи атрибута rules можно управлять также толщиной линий, обрамляющих ячейки внутри таблицы. Принятое по умолчанию поведение, соответствующее значению all, означает наличие рамок вокруг ячеек. Значение groups делает более толстыми линии между группами строк и столбцов, определенными тегами <thead>, <tbody>, <tfoot>, <col> и <colgroup>. Значения rows и cols проводят линии только между строками и столбцами соответственно, тогда как значение none убирает обрамление всех ячеек в таблице.

10.2.1.5. Атрибуты bordercolor, bordercolorlight и bordercolordark

Обычно популярные броузеры создают рамку вокруг таблицы с использованием трех цветов для достижения эффекта трехмерности: цвета фона документа и его темной и светлой вариаций. Не входящий в стандарт, но реализованный в броузере Internet Explorer атрибут bordercolor позволяет установить цвет рамки вокруг таблицы и обрамления ячеек, не совпадающий с цветом фона (если, конечно, рамка включена). Значениями атрибута bordercolor могут быть либо шестнадцатеричный RGB-триплет, либо стандартное название цвета, описанные в приложении G.

Internet Explorer позволяет также устанавливать цвета сторон рамки по отдельности при помощи специальных атрибутов расширения bordercolorlight и bordercolordark, значения которых соответствуют бликам и теням на светлой и темной сторонах рамки.

Эффективность имитации трехмерности связана с отношением между двумя этими цветами. В общем случае светлый цвет должен быть приблизительно на 25% ярче, чем цвет рамки, а темный – на 25% темнее. Не следует забывать, что если вы пользуетесь этими атрибутами, то лишь владельцы броузера Internet Explorer увидят эти эффекты.

10.2.1.6. Атрибут cellspacing

Атрибут cellspacing управляет расстоянием между соседними ячейками в таблице и шириной поля по внешним краям ячеек, расположенных вдоль границ таблицы.

Броузеры обычно оставляют между ячейками два пиксела и столько же вдоль границы таблицы. Если включить в тег <table> атрибут border, расстояние между ячейками увеличивается на два пиксела (вместе становится четыре), чтобы было где изобразить выпуклую линию границы между ячейками. Внешние поля граничных ячеек увеличиваются на значение атрибута border.

Включая в тег <table> атрибут cellspacing, можно увеличивать и уменьшать расстояние между ячейками. К примеру, чтобы сделать его минимальным, нужно включить в тег атрибут border и положить cellspacing равным нулю.

10.2.1.7. Атрибут cellpadding

Атрибут `cellpadding` заведует расстоянием между краем ячейки и его содержимым. По умолчанию это расстояние равно одному пикселу. Можно сделать так, чтобы содержимое ячейки касалось границ соответствующей ячейки, положив `cellpadding=0` в теге `<table>`. Это расстояние можно увеличить, положив значение атрибута равным величине, превосходящей единицу.

10.2.1.8. Комбинации атрибутов border, cellspacing и cellpadding

Результат совместного использования атрибутов `border`, `cellspacing` и `cellpadding` в теге `<table>` может оказаться неочевидным. На рис. 10.2 показано, как эти атрибуты создают внутренние и внешние рамки различной ширины.

Хотя допустимы любые комбинации атрибутов `border` и `cellspacing`, ниже приведены встречающиеся чаще всего:

- `border=1` и `cellspacing=0` создают внешние и внутренние рамки наименьшей возможной толщины – два пиксела.
- `border=n` и `cellspacing=0` устанавливают минимально возможное расстояние между ячейками (два пиксела) и внешнюю рамку толщиной `n` плюс один пиксел.
- `border=1` и `cellspacing=n` создают обрамление одинаковой толщины – внешние и внутренние рамки имеют толщину `n` плюс два пиксела, причем два пиксела отводятся на тени – по одному с каждой стороны.

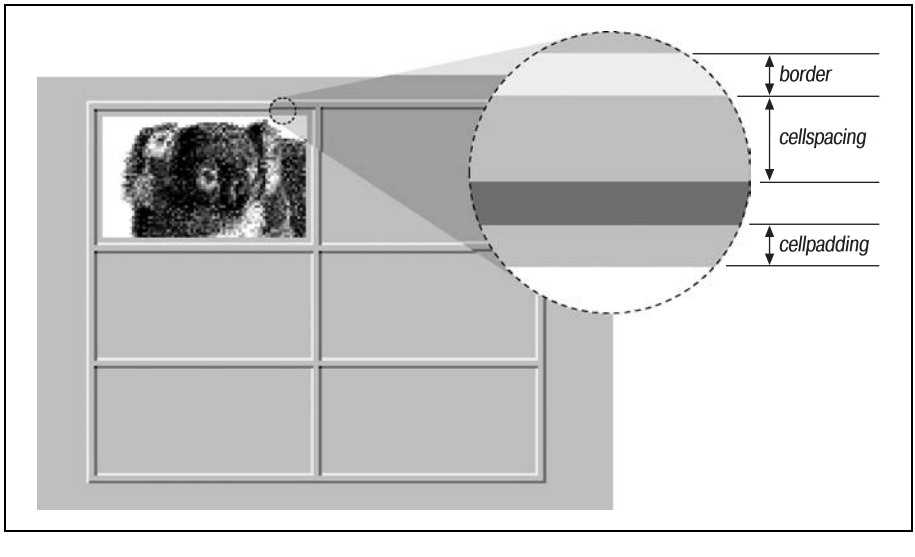

***Рис. 10.2.** Атрибуты border, cellspacing и cellpadding тега <table>*

10.2.1.9. Атрибут cols

Прежде чем изобразить таблицу, броузер должен прочитать ее целиком, чтобы определить число столбцов в таблице и ширину каждого из них. Для больших таблиц это может быть долгим процессом, вынуждающим пользователей дожидаться появления вашей страницы. Не входящий в стандарт атрибут `cols` заранее сообщает броузеру, сколько столбцов будет в таблице. Значением этого атрибута должно быть целое число, означающее количество столбцов в таблице.

Атрибут `cols` – это не более чем совет броузеру. Если указано неправильное число столбцов, броузер вправе проигнорировать атрибут `cols`, с тем чтобы вывести таблицу правильно. Вообще, включать атрибут `cols` в тег `<table>` – правильно, хотя бы для того, чтобы помочь броузеру быстрее вывести таблицу.

10.2.1.10. Атрибуты valign и nowrap

Атрибут `valign` задает общее для всех ячеек таблицы выравнивание по вертикали. Допустимыми значениями `valign` в теге `<table>` являются `top` (по верхней границе), `bottom` (по нижней границе), `middle` (по центру) и `baseline` (по базовой линии). По умолчанию содержимое ячейки центрируется по вертикали.

Броузеры рассматривают каждую ячейку таблицы, как если бы она была окном броузера, заполняя ячейку содержимым так же, как они это делают при отображении содержимого тела документа (с учетом особых свойств выравнивания для ячеек таблицы). Соответственно броузеры автоматически переносят строки текста, заполняя отведенное им в ячейке пространство. Атрибут `nowrap`, включенный в тег `<table>`, останавливает эти переносы текста во всех строках таблицы. При наличии атрибута `nowrap` броузер располагает все содержимое ячейки в одну строку, пока не встретит теги `<br>` или `<p>`, вызывающие разрыв строки, вслед за которым поток содержимого начинается с новой строки внутри ячейки таблицы.

Для тега `<table>` только Opera поддерживает атрибут `valign`. Ни один броузер не поддерживает атрибут `nowrap` на этом уровне. Впрочем, того же результата вы можете достигнуть, изменяя атрибуты `valign` и `nowrap` тегов `<tr>`, `<td>` и `<th>`, и такой подход приветствуется всеми популярными броузерами.

10.2.1.11. Атрибуты width и height

Автоматически броузер создает таблицу такой ширины, какая требуется для правильного отображения всего содержимого ячеек. При необходимости можно при помощи атрибута `width` сделать таблицу шире.

Значение атрибута `width` – это либо целое число пикселов, либо процент от ширины окна броузера, причем допускаются значения, превосходящие 100%. К примеру:

```
<table width=400>
```

предлагает броузеру отобразить таблицу шириной в 400 пикселов, с учетом всего ее внешнего обрамления. Если таблица шире 400 пикселов, броузер игнорирует атрибут. Альтернативным образом:

```
<table width="50%">
```

предлагает броузеру создать таблицу в половину ширины окна броузера. Опять в ширину таблицы входит все внешнее обрамление таблицы и атрибут игнорируется, если таблица без этого атрибута была бы шире.

Относительную ширину следует использовать для тех таблиц, ширина которых должна автоматически изменяться с изменением ширины окна броузера. К примеру, для таблицы, которая должна располагаться во всю ширину окна броузера, следует написать `<table width=100%>`. Абсолютные значения ширины следует использовать для аккуратно отформатированных таблиц, содержимое которых будет трудно прочитать в широком окне.

Популярные броузеры позволяют вам использовать нестандартный атрибут `height` для установки рекомендуемой высоты таблицы. Броузер отобразит таблицу не меньшей высоты, но может сделать ее выше, если это необходимо для размещения содержимого таблицы. Этот атрибут полезен, когда нужно растянуть таблицу, чтобы заполнить фрейм или какую-то специальную область документа, но употребляется редко, в частности из-за того, что не входит в стандарт.

10.2.1.12. Атрибут summary

Атрибут `summary` введен стандартом HTML 4.0. Его значением служит заключенная в кавычки строка, описывающая назначение и подытоживающая содержание таблицы. В соответствии со стандартом назначение этого атрибута состоит в предоставлении расширенного доступа невизуальным броузерам, особенно для пользователей с физическими недостатками.

10.2.1.13. Атрибуты hspace и vspace

Как и в случае с изображениями, рекомендуется оставить вокруг таблицы некоторое свободное пространство. Воспользуйтесь нестандартными атрибутами `hspace` и `vspace` в теге `<table>`, каждый из которых принимает в качестве значения количество пикселов, определяющее свободное пространство слева и справа и соответственно сверху и снизу от таблицы, то есть пространство между таблицей и окружающим ее текстом. Интересно, что все популярные броузеры, кроме Internet Explorer, поддерживают эти атрибуты в теге `<table>` при том, что Internet Explorer поддерживает их в теге `<img>`.

10.2.2. Стандартные атрибуты таблиц

Стандарты HTML 4 и XHTML в соединении со стандартом каскадных таблиц стилей (CSS) предоставляют множество атрибутов, которые яв-

ляются общими не только для тега <table> и тегов, используемых при описании таблиц, но также и для многих других тегов.

10.2.2.1. Атрибуты id и title

Атрибут id и заключенная в кавычки строка – значение атрибута – используется для снабжения таблиц документа уникальными метками, позволяющими ссылаться на них в гиперссылках, таблицах стилей, сценариях или апплетах. Необязательный атрибут title и заключенная в кавычки строка – значение атрибута – используются для привязывания к таблице фразы, описывающей таблицу. Название не обязано быть уникальным, и, возможно, броузер им никак не воспользуется. Популярные броузеры, например, отображают текстовое значение атрибута title всякий раз, когда указатель мыши попадает на содержимое тега. [атрибут id, 4.1.1.4] [атрибут title, 4.1.1.5]

10.2.2.2. Атрибуты dir и lang

Хотя содержимое Сети использует, главным образом, английский язык, Сеть является всемирной. Стандарты HTML 4 и XHTML способствуют распространению национальных языков на сетевых страницах. Мы от всего сердца поддерживаем это стремление. Атрибуты dir и lang представляют лишь малую часть этого процесса.

Атрибут lang позволяет явно указать язык, который употребляется в таблице. Значение атрибута lang – это двухбуквенный код языка по стандарту ISO, включающий необязательный языковой модификатор, добавляемый через дефис.

Атрибут dir советует броузеру, в каком направлении следует отображать текст: слева направо (dir=ltr), как в обычных западных языках, например английском или немецком, или справа налево (dir=rtl), как это обычно в восточных языках, таких как китайский или иврит.

Все последние версии популярных броузеров поддерживают атрибуты dir и lang. [атрибут dir, 3.6.1.1] [атрибут lang, 3.6.1.2]

10.2.2.3. Атрибуты class и style

Стандарт каскадных таблиц стилей (CSS) представляет собой санкционированный способ определения характеристик отображения HTML/XHTML-элементов и быстро становится единственным таким способом. Атрибут style используется для определения характеристик отображения таблицы и ее элементов, замещая любые другие правила из таблицы стилей, действующие по отношению ко всему документу. Атрибут class используется для ссылки на таблицу стилей, которая определяет уникальные характеристики отображения таблицы и ее элементов.

Мы подробно обсуждаем атрибуты class и style и стандарт CSS в главе 8. Их действие затрагивает все аспекты работы с таблицами, и они хорошо поддерживаются всеми популярными броузерами. [встроенные стили: атрибут style, 8.1.1] [стилевые классы, 8.3]

10.2.2.4. Атрибуты событий

В популярные броузеры встроены механизмы, обнаруживающие различные инициированные пользователем с помощью мыши и/или клавиатуры события, которые возможны в таблицах, их элементах и их окрестности. К примеру, пользователь может щелкнуть указателем мыши в одной из ячеек таблицы или выделить заголовок таблицы, а затем нажать клавишу <Enter>.

При помощи разных атрибутов событий (например *onClick* или *onKeyDown*) можно заставить броузер реагировать на такие события, исполняя одну или несколько JavaScript-команд или апплетов, сослаться на которые можно, указывая их в качестве значений соответствующих атрибутов событий. Подробнее см. в главе 12.

10.2.3. Тег <tr>

Для определения строки в таблице используется тег <tr>. В нем помещаются одна или несколько ячеек, содержащих заголовки, обозначаемые тегом <th>, и содержащих данные, определяемые тегом <td> (см. раздел 10.2.4). Тег <tr> имеет множество специальных атрибутов, управляющих его поведением, вместе с множеством общих атрибутов таблицы, описанных в разделе 10.2.2.

<tr>	
Функция:	Определяет в таблице строку
Атрибуты:	align, background, bgcolor, bordercolor, bordercolordark, bordercolorlight, char, charoff, class, dir, id, lang, nowrap, onClick, onDblClick, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, style, title, valign
Закрывающий тег:	</tr>; в HTML может опускаться
Содержит:	*tr_content* (содержимое строки таблицы)
Может содержаться в:	*table_content* (содержимое таблицы)

Во всех строках таблицы столько же ячеек, сколько в самой длинной строке таблицы. Броузер автоматически добавляет пустые ячейки в строки с меньшим числом ячеек.

10.2.3.1. Атрибуты align и valign

Атрибут align для тега <table>, может быть, и нежелателен в стандартах HTML 4 и XHTML, но он по-прежнему хорош в теге <tr> и других элементах таблицы. Атрибут align для тега <tr> позволяет изменять общее для всех ячеек строки выравнивание их содержимого по горизонтали. Атрибут действует на все ячейки в текущей, но не в следующей строке.

В соответствии со значениями атрибута `align` – `left`, `right`, `center`, `justify` и `char` – броузер выравнивает содержимое каждой ячейки в строке по левому или правому краю ячейки, по ее центру, по обоим краям или по определенному символу соответственно.

Подобным образом можно изменять общее для всех ячеек строки выравнивание их содержимого по вертикали при помощи атрибута `valign`. Обычно броузеры выводят содержимое ячеек центрированным по вертикали. Присвоив атрибуту `valign` в теге `<tr>` одно из значений: `top`, `bottom`, `center`, `middle` или `baseline` (только Internet Explorer), вы предлагаете броузеру поместить содержимое ячеек строки в верхней, в нижней части соответствующей ячейки, отцентрировать или выровнять по базовой линии верхней строки текста в других ячейках таблицы (рис. 10.3):

```
<table border="border">
  <tr>
    <th>Alignment</th>
    <th>Top</th>
    <th>Baseline</th>
    <th>Center</th>
    <th>Middle></th>
    <th>Bottom</th>
  </tr>
  <tr align="center">
    <th><h1>Baseline_  _<br />Another line</h1></th>
    <td valign="top">AAyy</td>
    <td valign="baseline">_AAyy_</td>
    <td valign="center">AAyy</td>
    <td valign="middle">AAyy</td>
    <td valign="bottom">AAyy</td>
  </tr>
</table>
```

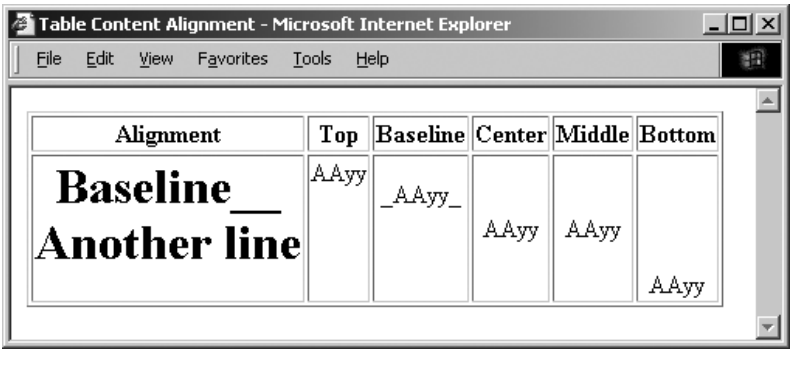

Рис. 10.3. *Действие атрибута valign; только Internet Explorer поддерживает значение baseline для этого атрибута*

Можно также определить горизонтальное и вертикальное выравнивание для отдельных ячеек в строке (раздел 10.2.3.1). Используйте атрибуты выравнивания в теге <tr>, указывая чаще всего встречающееся в строке выравнивание (если оно отличается от принятого по умолчанию), и используйте атрибуты align и valign со своими значениями для каждой ячейки, выравнивание в которой отличается от общего.

Таблица 10.1 содержит значения атрибутов горизонтального (align) и вертикального (valign) выравнивания. Значения в скобках приняты по умолчанию в популярных броузерах.

Таблица 10.1. Значения атрибутов выравнивания содержимого ячеек таблицы

	Netscape и Internet Explorer	
Атрибут	**Заголовки**	**Данные**
align	Left	(Left)
	(Center)	Center
	Right	Right
	Justify	Justify
	Char[a]	Char[a]
valign	Top	Top
	(Center)	(Center)
	(Middle)	(Middle)
	Bottom	Bottom
	Baseline	Baseline

[a] Это значение пока не поддерживается.

10.2.3.2. Атрибуты char и charoff

Даже простые текстовые редакторы позволяют выравнивать числа в таблице по десятичной точке. До появления стандарта HTML 4.0 языку недоставало этой возможности. Теперь можно использовать атрибут char, чтобы указать, какой символ в ячейках строки таблицы должен быть осью такого выравнивания. Нет необходимости присваивать значение атрибуту char. Если в тег включен атрибут char и ему не присвоено никакого значения, то принимаемый по умолчанию символ будет зависеть от языка – в английском это будет точка, во французском или русском – запятая. Если атрибуту char присвоить значение, состоящее из одного символа, – будет определена другая ось выравнивания.

Атрибут charoff с целым значением задает положение первого вхождения символа, по которому производится выравнивание, в каждой строке. Если строка не содержит символа, по которому производится выравнивание, ее конец выравнивается по указанной в charoff позиции.

Атрибуты char и charoff введены только в HTML 4 и XHTML и пока не поддерживаются ни одним из популярных броузеров.

10.2.3.3. Атрибуты bgcolor и background

Подобно своему родственнику из тега <table>, атрибут bgcolor в теге <tr> устанавливает цвет фона для всей строки.[1] Значением атрибута bgcolor должен быть либо RGB-триплет, либо стандартное название цвета. Синтаксис кодов цветов и допустимые названия цветов можно найти в приложении G.

У каждой ячейки в строке будет фон указанного цвета. Цвета отдельных ячеек могут быть переопределены спецификацией атрибута bgcolor для этих ячеек.

Нестандартный атрибут background, принимающий в качестве значения URL-адрес файла с изображением, помещает картинку (размноженную, если необходимо) позади текста в строку таблицы. Например, следующий тег заполняет строку таблицы кирпичиками:

```
<tr background="bricks.gif">
```

Все популярные броузеры поддерживают атрибут bgcolor, и все они, кроме Internet Explorer, поддерживают расширение background.

10.2.3.4. Атрибуты bordercolor, borderlight, bordercolordark

Internet Explorer позволяет использовать эти атрибуты аналогично тому, как это делается для тега <table>, для определения цвета рамок в текущей строке.

Атрибуты в строке переопределяют значения, указанные для таблицы в целом. Смотрите подробное описание этих расширений в разделе 10.2.1.5. Цветовыми значениями этих атрибутов могут быть либо RGB-триплет, либо стандартное название цвета, полностью описанные в приложении G.

10.2.3.5. Атрибут nowrap

Броузеры рассматривают каждую ячейку таблицы, как если бы она была окном броузера, заполняя ячейку содержимым так же, как они это делают при отображении содержимого тела документа (с учетом особых свойств выравнивания для ячеек таблицы). Соответственно броузеры автоматически переносят строки текста, заполняя отведенное им в ячейке пространство. Атрибут nowrap, включенный в тег <tr>, останавливает эти переносы текста во всех ячейках строки. При наличии атрибута nowrap броузер располагает все содержимое ячейки в одну строку, пока не встретит теги
 или <p>, вызывающие разрыв стро-

[1] В отличие от тега <table> для Internet Explorer, тег <tr> не поддерживает фоновые изображения.

ки, вслед за которым поток содержимого начинается с новой строки внутри ячейки таблицы.

10.2.4. Теги <th> и <td>

Теги <th> и <td> применяются в тегах таблицы <tr> для создания заголовочных ячеек и ячеек с данными соответственно и определения содержимого ячеек в строке. Оба тега действуют подобным образом, единственное между ними отличие состоит в том, что броузер отображает текст заголовка – считается, что он представляет название или описание данных в таблице – жирным шрифтом и что принятое по умолчанию выравнивание содержимого заголовка может отличаться от такового для данных. Данные обычно по умолчанию выравниваются влево, а заголовки – по центру (табл. 10.1).

<th> и <td>

Функция:	Определяют ячейки данных и заголовков
Атрибуты:	abbr, align, background, bgcolor, bordercolor, bordercolordark, bordercolorlight, char, charoff, class, colspan, dir, headers, height, id, lang, nowrap, onClick, onDblClick, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut onMouseOver, onMouseUp, rowspan, scope, style, title, valign, width
Закрывающий тег:	</th> или </td>; в HTML может опускаться
Содержит:	*body_content* (содержимое тела)
Может содержаться в:	*tr_content* (содержимое строки таблицы)

Подобно тегу строки таблицы (<tr>) теги ячеек поддерживают богатое множество атрибутов стиля и выравнивания содержимого, которые можно применять к одиночной ячейке заголовка или данных. Эти атрибуты имеют приоритет перед своими аналогами, общими для ячеек строки. Имеются также специальные атрибуты, определяющие число строк и столбцов в таблице, которое может охватывать ячейки. Теги <th> и <td> допускают также общие атрибуты таблицы, описанные в разделе 10.2.2.

Содержимым тегов <th> и <td> может быть все, что можно поместить в тело документа, включая текст, изображения, формы и т. д., даже другие таблицы. И как это было описано выше, броузер автоматически создает таблицу достаточно большой – и по горизонтали и по вертикали, – чтобы отобразить содержимое всех ячеек.

Если в некоторой строке оказалось меньше ячеек, чем в других строках, броузер помещает пустые ячейки в конец строки, заполняя ее. Если вы хотите создать пустую ячейку в начале или в середине строки, чтобы обозначить, например, отсутствующие данные, создайте ячейку данных или заголовка без содержимого.

Пустые ячейки выглядят иначе, чем ячейки с данными или заголовками: когда у таблицы имеется обрамление, пустые ячейки не будут иметь выпуклую рамку, а просто будут пустыми. Если нужно сделать пустую ячейку с выпуклой рамкой, как у всех других ячеек в таблице, позаботьтесь о том, чтобы поместить в ячейку минимальное содержимое[1] – одиночный тег
, например.

10.2.4.1. Атрибуты align и valign

Атрибуты align и valign идентичны одноименным атрибутам тега строки (<tr>, см. раздел 10.2.3), но используются с тегами <th> или <td> и управляют выравниванием содержимого по горизонтали и вертикали только в текущей ячейке. Они имеют приоритет перед установками выравнивания для строки в целом, но не действуют на выравнивание в следующих за текущей ячейках. Смотрите подробности в табл. 10.1.

Атрибуту align можно присвоить значения left, right, center, приказывая броузеру выровнять содержимое ячейки по левому или правому краю ячейки или по ее центру соответственно.

В ранних версиях Internet Explorer поддерживалось значение justify, равномерно заполнявшее ячейку словами, как в газетной колонке. Теперь этому пришел конец.

Атрибут valign принимает одно из значений: top (по умолчанию), bottom, center, middle или baseline, выравнивая содержимое ячейки по верху, по низу, по центру или в середине (только для IE) ячейки или по базовой линии первой строки текста в других ячейках строки таблицы (см. рис. 10.3).

10.2.4.2. Атрибут width

Подобно своему близнецу из тега <table>, который позволяет сделать таблицу шире, атрибут width для тегов ячеек позволяет сделать шире отдельную ячейку и, следовательно, весь столбец, в котором она содержится. Значением атрибута width могут быть либо целое число пикселов, либо процентное значение, выражающее ширину ячейки в отношении к ширине всей таблицы. Например:

```
<th width=400>
```

устанавливает ширину текущей ячейки заголовка и, следовательно, всего столбца, равной 400 пикселам. Альтернативно:

```
<td width="40%">
```

создает ячейку данных в столбце, занимающем 40 процентов всей ширины таблицы.

---

[1] Обычно в таком случае в ячейку помещают либо символ неразрывного пробела ( ), либо прозрачный рисунок-распорку размером 1×1 пиксел. – *Примеч. науч. ред.*

Поскольку популярные броузеры делают все ячейки столбца одной и той же ширины, следует вставлять атрибут width только в одну ячейку столбца, предпочтительно в первой строке, чтобы легче было читать исходный текст. Если окажется, что несколько ячеек в столбце содержат атрибут width, будет учитываться больший из них. Нельзя сделать столбец уже, чем необходимо, чтобы отобразить все ячейки столбца. Поэтому если броузер решит, что столбец должен быть не уже 150 пикселов, для того чтобы в нем поместилось все содержимое ячеек, то в любой из ячеек этого столбца броузер проигнорирует атрибут width, который пытается установить ширину ячейки всего в 100 пикселов.

10.2.4.3. Атрибут height

Атрибут height позволяет определить минимальную высоту текущей ячейки в пикселах. Поскольку все ячейки в строке имеют одинаковую высоту, этот атрибут достаточно указать в одной из ячеек строки, желательно в первой. Если какой-то другой ячейке в строке нужно быть выше, чтобы вместить свое содержимое, этот атрибут игнорируется, и все ячейки в строке будут большего размера.

По умолчанию все ячейки в строке имеют высоту самой высокой ячейки в строке, которая в точности такова, чтобы в ней поместилось все ее содержимое.

10.2.4.4. Атрибут colspan

В таблице принято помещать заголовки, описывающие несколько расположенных под ними столбцов, как это сделано на рис. 10.1. Атрибут colspan используется в тегах заголовочных ячеек и ячеек данных, чтобы объединить несколько ячеек в строке. Установите значение атрибута colspan равным целому числу ячеек строки, которые вы хотите объединить. Например:

```
<td colspan="3">
```

предлагает броузеру создать ячейку, занимающую то же пространство по горизонтали, как три ячейки в строках, расположенных выше или ниже. Броузер разместит содержимое такой ячейки на всем выделенном пространстве.

Что будет, если справа от ячейки нет достаточного числа ячеек? Броузер объединит все ячейки до правого края таблицы, он не станет добавлять в каждую последующую (и предыдущую) строку таблицы недостающие до значения атрибута colspan ячейки. Можно освободиться от этого ограничения, создав в одной строке необходимые лишние ячейки[1], оставив их при этом пустыми. (В каждую из них стоит что-

[1] То есть необходимо добиться, чтобы количество ячеек в каждой строке таблицы (с учетом ячеек, «слитых» с помощью colspan) было одинаково. – *Примеч. науч. ред.*

нибудь поместить, например одиночный тег
, чтобы броузер окружил их трехмерной рамкой.)

10.2.4.5. Атрибут rowspan

Подобно тому как атрибут colspan объединяет несколько ячеек в одной строке, атрибут rowspan объединяет ячейку с несколькими находящимися под ней.

Атрибут rowspan включается в теги <th> и <td> в самой верхней строке таблицы[1], пересекающей создаваемую объединенную ячейку, и ему присваивается значение, равное числу строк, которые эта ячейка охватывает. Созданная ячейка будет занимать пространство текущей ячейки и соответствующего числа ячеек под ней. Броузер будет размещать содержимое объединенной ячейки по всему ее пространству. Например:

```
<td rowspan="3">
```

создает ячейку, которая объединяет текущую и еще две нижележащие строки.

Как и в случае атрибута colspan, броузер игнорирует слишком большие значения rowspan и объединяет только ячейки, явно определенные тегами <tr>. Броузер не станет добавлять в таблицу пустые строки, чтобы удовлетворить чрезмерно большому значению rowspan.

10.2.4.6. Комбинирование атрибутов colspan и rowspan

Можно распространить одну ячейку одновременно по вертикали и по горизонтали, включив в теги заголовка таблицы или данных таблицы атрибуты colspan и rowspan вместе.[2] Например:

```
<th colspan="3" rowspan="4">
```

создает заголовочную ячейку, которая, как и следовало ожидать, пересекает три столбца и четыре строки, включает текущую ячейку, две ячейки справа и три ячейки под текущей ячейкой. Броузер располагает содержимое такой ячейки по всему ее пространству, выравнивая его в соответствии с выравниванием, принятым для текущей строки или указанным явно в теге этой ячейки, как это описано выше.

---

[1] Не обязательно только в самой верхней строке. Атрибут rowspan сработает для любой строки таблицы (разумеется, если под данной ячейкой имеется заданное в rowspan количество строк). – *Примеч. науч. ред.*

[2] Будьте внимательны: совместное применение этих атрибутов может создать ситуацию, когда ячейка, объединенная ранее по вертикали, попадет в область ячеек, объединенных по горизонтали. В зависимости от броузера такие ошибочно заданные объединения ячеек либо отобразятся некорректно, либо объединения не произойдет вовсе. – *Примеч. науч. ред.*

10.2.4.7. Атрибут nowrap

Броузеры рассматривают каждую ячейку таблицы, как если бы она была окном броузера, заполняя ячейку содержимым так же, как они это делают при отображении содержимого тела документа (с учетом особых свойств выравнивания для ячеек таблицы). Соответственно броузеры автоматически переносят строки текста, заполняя отведенное им в ячейке пространство. Атрибут `nowrap`, включенный в теги заголовочных ячеек или ячеек данных, останавливает эти переносы текста в ячейке. При наличии атрибута `nowrap` броузер располагает все содержимое ячейки в одну строку, пока не встретит теги `<br>` или `<p>`, вызывающие разрыв строки, вслед за которым поток содержимого начинается с новой строки внутри ячейки таблицы.

10.2.4.8. Атрибуты bgcolor и background

Для отдельной ячейки также можно изменить цвет фона с помощью атрибутов `bgcolor` и `background`. Значением атрибута `bgcolor` должен быть либо RGB-триплет, либо стандартное название цвета. Синтаксис кодов цветов и допустимые названия цветов можно найти в приложении G.

Атрибут `background` принимает в качестве значения URL изображение, которым выкладывается фон ячейки. Если изображение больше ячейки, оно будет обрезано. Любопытно, что броузер Internet Explorer солидарен с другими популярными броузерами в поддержке атрибута `background`, примененного к одиночной ячейке, но, в отличие от них, не поддерживает этот атрибут для тегов `<table>` и `<tr>`.

Значения атрибутов `background` и `bgcolor` могут быть переопределены соответствующими свойствами из таблиц стилей.

10.2.4.9. Атрибуты bordercolor, borderlight, bordercolordark

Internet Explorer позволяет использовать эти атрибуты для изменения цвета рамок в отдельной ячейке, если, конечно, рамки включены атрибутом `border` для таблицы. Смотрите подробное описание этих атрибутов в связи с тегом `<table>` в разделе 10.2.1.5.

Эти атрибуты переопределяют значения аналогичных атрибутов, указанных в тегах `<table>` и `<tr>` для текущей ячейки. Значениями этих трех атрибутов могут быть либо RGB-триплет, либо стандартное название цвета, описанные в приложении G.

10.2.4.10. Атрибуты char и charoff

Как и для тега `<tr>`, можно использовать атрибут `char` с тегами `<th>` и`<td>`, чтобы указать, какой символ в ячейках строки таблицы должен быть осью выравнивания, что обычно делается для десятичных дробей. Нет необходимости присваивать значение атрибуту `char`. Если этот атрибут включен в тег без указания его значения, то принимаемый по умолчанию символ будет зависеть от языка – в английском это будет точка, в русском или французском – запятая. Если атрибуту `char`

присвоить значение, состоящее из одного символа, будет определен другой центр выравнивания.

Атрибут `charoff` с целым значением используется для определения положения первого вхождения символа, по которому производится выравнивание, в каждой строке. Если строка не содержит символа, по которому производится выравнивание, ее конец выравнивается по указанной в `charoff` позиции.

Атрибуты `char` и `charoff` включены в стандарты HTML 4 и XHTML, но пока не поддерживаются ни одним из популярных броузеров.

10.2.4.11. Атрибуты headers и scope

Атрибут `headers` ассоциирует ячейки заголовков с ячейкой данных таблицы. Значением этого атрибута служит заключенный в кавычки список имен, определенных для различных заголовочных ячеек при помощи атрибута `id`. Атрибут `headers` особенно полезен для невизуальных броузеров, которые, возможно, будут произносить вслух содержимое заголовка, перед тем как представлять содержимое ассоциированных с ним ячеек данных.

Атрибут `scope` используется для связывания ячеек данных с ячейкой заголовка. Если `scope=row`, все ячейки строки заголовка ассоциируются с ячейкой заголовка. Если `scope=col`, с текущей ячейкой связываются все ячейки текущей строки. Значения `rowgroup` и `colgroup` связывают заголовочную ячейку со всеми ячейками группы строк (определенной тегами `<thead>`, `<tbody>` или `<tfoot>`) и ячейками группы столбцов (определенной тегами `<col>` или `<colgroup>`) соответственно.

10.2.4.12. Атрибут abbr

Значением этого атрибута должно быть сокращенное описание содержимого ячейки. При недостатке места броузер может решить вывести это сокращенное описание или использовать его в невизуальном контексте.

10.2.4.13. Атрибут axis

Таблицы обычно переполнены данными, вызывающими у просматривающего их дальнейшие вопросы. Сведенный в таблицу отчет о расходах естественным образом порождает, например, такие вопросы: «Сколько я потратил на еду?» или «Сколько всего ушло на такси?» В будущем броузеры, возможно, будут обслуживать такие запросы с помощью атрибута `axis`.

Значение этого атрибута – это заключенный в кавычки список имен категорий, который может использоваться для создания запросов. В результате, если в ячейках, соответствующих покупкам продуктов, будет написано `axis=meals`, броузер сможет найти такие ячейки, выделить их содержимое и вычислить сумму.

10.2.5. Тег <caption>

У таблицы обычно бывает заголовок, объясняющий ее содержимое, поэтому популярные броузеры предоставляют тег заголовка таблицы. Авторы обычно помещают тег <caption> непосредственно после тега <table>, но можно помещать его практически где угодно внутри таблицы между тегами строк. Тег заголовка может включать любое содержимое тела документа, так же как ячейка таблицы.

<caption>	
Функция:	Определяет заголовок таблицы
Атрибуты:	align, class, dir, id, lang, onClick, onDblClick, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, style, title, valign
Закрывающий тег:	</caption>; присутствует обязательно
Содержит:	*body_content* (содержимое тела)
Может содержаться в:	*table_content* (содержимое таблицы)

10.2.5.1. Атрибуты align и valign

По умолчанию броузеры помещают содержимое заголовка над таблицей центрированным по горизонтали. Можно поместить его под таблицей, присвоив атрибуту align значение bottom (значение top, разумеется, эквивалентно принятому по умолчанию).

Вы также можете использовать атрибут align для определения горизонтального положения заголовка, однако популярные броузеры поразному интерпретируют альтернативные значения. Например, Internet Explorer и Opera в случае установки атрибута align в значение left или right выравнивают (соответственно влево или вправо) текст заголовка вдоль горизонтального верхнего края таблицы. Зато в Netscape и Firefox текст помещается рядом с левой или правой стороной таблицы, в верхней ее части.

Кроме того, Internet Explorer поддерживает атрибут valign со значениями top или bottom у тега <caption>. В комбинации с атрибутом align этот атрибут позволяет вам выравнивать заголовок таблицы по любому из четырех ее углов, но не вдоль какой-либо стороны. Остальные броузеры игнорируют атрибут valign.

Рассмотрим пример. На рис. 10.4 показано, как Internet Explorer выводит заголовок внизу таблицы и выравнивает по левому краю. В то же время Firefox, игнорирующий атрибут valign и интерпретирующий левое выравнивание по-другому, помещает заголовок вдоль левого края таблицы (рис. 10.5):

```
<caption valign=bottom align=left>
```

```
    Kumquat versus a poked eye, by gender
</caption>
```

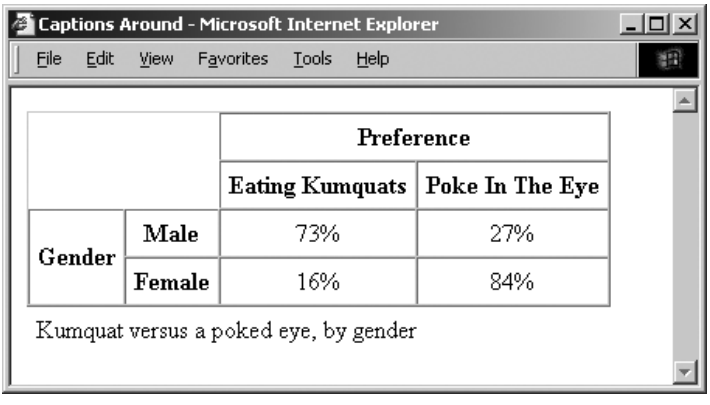

Рис. 10.4. *Комбинация атрибутов valign и align в теге <caption> в броузере Internet Explorer позволяет поместить текст у любого из четырех углов таблицы, а также отцентрировать его у верхнего или нижнего края*

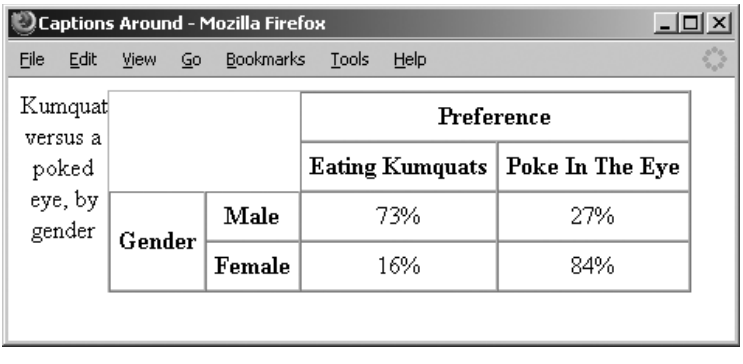

Рис. 10.5. *Firefox, как и Netscape, игнорирует атрибут valign и помещает текст слева от таблицы*

10.2.5.2. И многие другие атрибуты

Подобно другим тегам таблицы тег `<caption>` поддерживает множество атрибутов языка, событий и стилей, которые описаны в разделе 10.2.2. Используйте их на здоровье. Но не забывайте использовать контекстный селектор `TABLE CAPTION`, когда ссылаетесь на стили заголовка во внешних или размещенных на уровне документа таблицах стилей.

10.3. Новейшие теги таблицы

Хотя простую таблицу можно создать быстро, сложные таблицы с меняющимися стилями рамок, скользящими заголовками и макетирова-

нием столбцов нелегко было сконструировать в старой модели таблиц HTML 3.2. Microsoft отчасти преодолел этот недостаток, добавив в Internet Explorer версии 3.0 множество элементов управления макетом таблицы. Эти очень полезные расширения вошли в стандарт HTML 4.0 и, следовательно, в XHTML. Они предоставляют возможности группировки строк и создания скользящих заголовков так же, как и возможность макетирования столбцов.

10.3.1. Определение разделов таблиц

Все строки в таблицах создаются одинаковыми. Однако в реальных таблицах некоторые строки отличаются от других. Например, многие таблицы имеют заголовочные строки (header, верхний колонтитул таблицы) и подвалы (footer, нижний колонтитул таблицы), которые повторяются на следующих страницах, если таблица не помещается целиком на одной странице. В больших таблицах смежные строки группируются и обрамляются особым образом, чтобы таблицу было легче читать и понимать. HTML 4 и XHTML поддерживают эти возможности с помощью тегов `<thead>`, `<tfoot>` и `<tbody>`.

10.3.2. Тег <thead>

Тег `<thead>` используется для определения группы заголовочных строк. Этот тег может появляться в теге `<table>` только один раз – в самом начале. В теге `<thead>` можно поместить один или несколько тегов `<tr>`, определяющих строки табличных заголовков. Тогда броузер, поддерживающий стандарты HTML 4/XHTML, будет повторять эти заголовочные строки при распечатке таблицы или при отображении ее в нескольких разделах. В частности, он будет повторять эти строки на каждой странице, если таблица занимает больше чем одну страницу.

Закрывающий тег `</thead>` в HTML не является обязательным. Поскольку тег `<thead>` появляется в таблице только там, где, предположительно, другие строки будут обозначены как входящие в тело или в нижний колонтитул таблицы, тег `<thead>` автоматически закрывается, когда броузер встречает теги `<tbody>` или `<tfoot>` либо закрывающий тег таблицы.

<thead>

Функция:	определяет верхний колонтитул таблицы
Атрибуты:	`align`, `char`, `charoff`, `class`, `dir`, `id`, `lang`, `onClick`, `onDblClick`, `onKeyDown`, `onKeyPress`, `onKeyUp`, `onMouseDown`, `onMouseMove`, `onMouseOut`, `onMouseOver`, `onMouseUp`, `style`, `title`, `valign`
Закрывающий тег:	`</thead>`; в HTML может опускаться
Содержит:	*table_content* (содержимое таблицы)
Может содержаться в:	*table_content* (содержимое таблицы)

Многие атрибуты тега <thead> действуют так же, принимают те же значения и влияют на содержимое включенных в <thead> тегов <tr> точно так, как если бы они были применены по отдельности в каждой строке. К примеру, атрибут align принимает значения left, right и center, управляя выравниванием текста по горизонтали во всех заголовочных ячейках. Подобным же образом атрибут valign принимает значения top, middle, baseline (только для Internet Explorer) и bottom, предписывая вертикальное выравнивание текста во всех заголовочных строках. Если какие-либо выравнивания или стили не определены, броузер центрирует заголовочный текст по вертикали и по горизонтали в соответствующих ячейках, что эквивалентно спецификации align=center и valign=middle в каждой ячейке. Разумеется, определения, относящиеся к отдельным строкам и ячейкам, так же как и определения таблиц стилей, имеют приоритет перед определениями в <thead>.

10.3.3. Тег <tfoot>

Тег <tfoot> используется для создания в таблице группы строк нижнего колонтитула (подвала). Тег <tfoot> может появляться в теге <table> только один раз, перед самым концом таблицы. Подобно тегу <thead> он может содержать один или несколько тегов <tr>, определяющих те строки, которые современные броузеры будут рассматривать как нижний колонтитул таблицы. Определенные таким образом строки броузер повторяет на каждой физической и виртуальной странице, на которые окажется разбита таблица. Чаще всего броузер повторяет подвальные строки таблицы внизу каждой страницы при распечатке таблицы, занимающей несколько страниц.

Закрывающий тег </tfoot> не является обязательным в HTML, так как нижний колонтитул заканчивается на следующем теге <tbody> или вместе с таблицей.

<tfoot>

Функция:	Определяет нижний колонтитул таблицы
Атрибуты:	align, char, charoff, class, dir, id, lang, onClick, onDblClick, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, style, title, valign
Закрывающий тег:	</tfoot>; в HTML может опускаться
Содержит:	*table_content* (содержимое таблицы)
Может содержаться в:	*table_content* (содержимое таблицы)

10.3.4. Тег <tbody>

Используйте тег <tbody> для разбиения таблицы на фрагменты. Тег <tbody> содержит одну или несколько строк, образующих группу в со-

ставе таблицы. Вполне допустимы таблицы без тега <tbody>, но если он был включен в таблицу, то их, вероятно, окажется не менее двух. Выделив таким образом группу строк, можно отделить ее от предшествующих и последующих строк линейками специальной толщины. Внутри тега <tbody> могут содержаться только определения строк с применением тега <tr>. И по определению фрагмент <tbody> замкнут. Например, нельзя объединить ячейки одного фрагмента <tbody> с другим.

<tbody>	
Функция:	Определяет раздел в таблице
Атрибуты:	align, char, charoff, class, dir, id, lang, onClick, onDblClick, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, style, title, valign
Закрывающий тег:	</tbody>; в HTML может опускаться
Содержит:	*table_content* (содержимое таблицы)
Может содержаться в:	*table_content* (содержимое таблицы)

Закрывающий тег </tbody> не является обязательным в HTML, поскольку фрагмент завершается с появлением очередного тега <tbody> или тега <tfoot>, или с концом таблицы. У <tbody> так же много атрибутов, как у <tfoot>, но они не поддерживаются даже популярными броузерами. Если вы хотите применить к этому фрагменту особое выравнивание, вам понадобится определить его для каждой строки в теге <tbody>.

10.3.5. Использование фрагментов таблицы

С точки зрения представления данных, самое важное, для чего можно применять теги <thead>, <tfoot> и <tbody>, это для разбиения таблицы на разделы, имеющие разный смысл и значение, и отделения их друг от друга при отображении специальными разделительными рамками. По умолчанию Internet Explorer ничего особенного не делает с рамками вокруг верхнего и нижнего колонтитулов и фрагментов таблицы. Применяя в теге <table> атрибут rules, можно, однако, провести более толстые линии между разделами <thead> и <tbody>, между разделами <tbody> и между <tbody> и <tfoot>, помогая тем самым пользователю разобраться в организации таблицы. [атрибут align, 10.2.1.1]

К примеру, вот простая таблица, которую вы уже видели в этой главе, но теперь в нее добавлены верхний и нижний колонтитулы. Следует отметить, что мы опустили множество закрывающих тегов, чтобы текст стал короче и его было легче читать. Это допустимо в HTML, но в документе, соответствующем XHTML, эти теги должны появиться:

```
<table border cellspacing=0 cellpadding=5 rules=groups>
  <caption align=bottom>Kumquat versus a poked eye, by gender</caption>
```

```
<thead>
  <tr>
    <td colspan=2 rowspan=2>
    <th colspan=2 align=center>Preference
  </tr>
  <tr>
    <th>Eating Kumquats
    <th>Poke In The Eye
  </tr>
</thead>
<tfoot>
  <tr>
    <td colspan=4 align=center>
      Note: eye pokes did not result in permanent injury
  </tr>
</tfoot>
<tbody>
  <tr align=center>
    <th rowspan=2>Gender
    <th>Male
    <td>73%
    <td>27%
  </tr>
  <tr align=center>
    <th>Female
    <td>16%
    <td>84%
  </tr>
</tbody>
</table>
```

Получившаяся таблица в том виде, в каком ее выведет Opera, показана на рис. 10.6. Обратите внимание на то, что линейки под шапкой и перед подвалом толще, чем рамки вокруг остальных строк таблицы. Это результат того, что мы включили в тег `<table>` специальный атрибут `rules=groups`. Установка атрибутов `rules=rows` или `rules=all` приведет к похожему эффекту.[1]

Длинные таблицы часто выигрывают, когда каждые несколько строк отделены от предыдущих более толстой линией – так их легче читать. Сделать это можно, сгруппировав строки таблицы несколькими тегами `<tbody>`. Каждое семейство строк, содержащееся в одном теге `<tbody>`, будет отделяться от предыдущих и последующих строк более толстыми линиями.

[1] В Internet Explorer параметр `groups` вызовет появление рамок только вокруг колонтитулов и фрагментов (что видно на рисунке), параметр `rows` вызовет появление дополнительных горизонтальных линий внизу всех строк, а `all` – появление рамок вокруг всех строк, толщина которых для пустых ячеек будет меньше, чем для всех остальных. – *Примеч. науч. ред.*

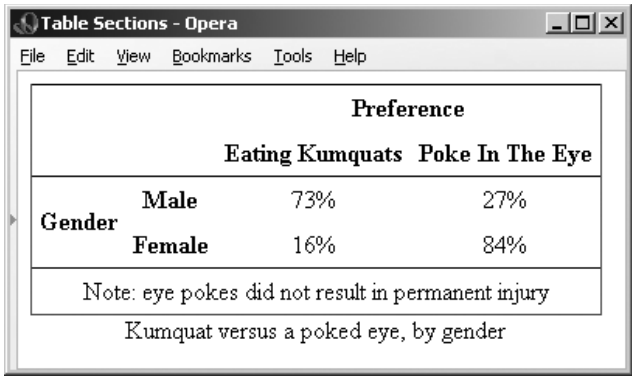

Рис. 10.6. *Используйте табличные теги HTML 4 и XHTML для разбиения таблицы на части, имеющие различное смысловое значение*

Ниже представлен подробный вариант предыдущей таблицы с дополнительными строками и разбиением на фрагменты.

```
<table border cellspacing=0 cellpadding=5 rules=groups>
    <caption align=bottom>Kumquat versus a poked eye, by gender</caption>
    <thead>
      <tr>
        <td colspan=2 rowspan=2>
        <th colspan=2 align=center>Preference
      <tr>
        <th>Eating Kumquats
        <th>Poke In The Eye
  <tfoot>
    <tr>
      <td colspan=4 align=center>
        Note: eye pokes did not result in permanent injury
    <tbody>
      <tr align=center>
        <th rowspan=4>Gender
        <th>Males under 18
        <td>94%
        <td>6%
      <tr align=center>
        <th>Males over 18
        <td>73%
        <td>27%
    <tbody>
      <tr align=center>
        <th>Females under 18
        <td>34%
        <td>66%
      <tr align=center>
        <th>Females over 18
        <td>16%
```

```
        <td>84%
    </table>
```

Результат показан на рис. 10.7. Обратили внимание на ячейку Gender? Броузер Netscape версии 4 и более ранних помещает ее слева, отцентрировав между строками Males и Females, что вполне естественно. Однако в стандартах HTML 4 и XHTML явным образом запрещены подобные трюки с разделами `<tbody>`, так что броузеры, уважающие эти стандарты, выведут нашу таблицу в четыре строчки, разбив их на две группы. Внутри таблицы вы можете создавать сколько угодно групп, добавляя теги `<tbody>`.

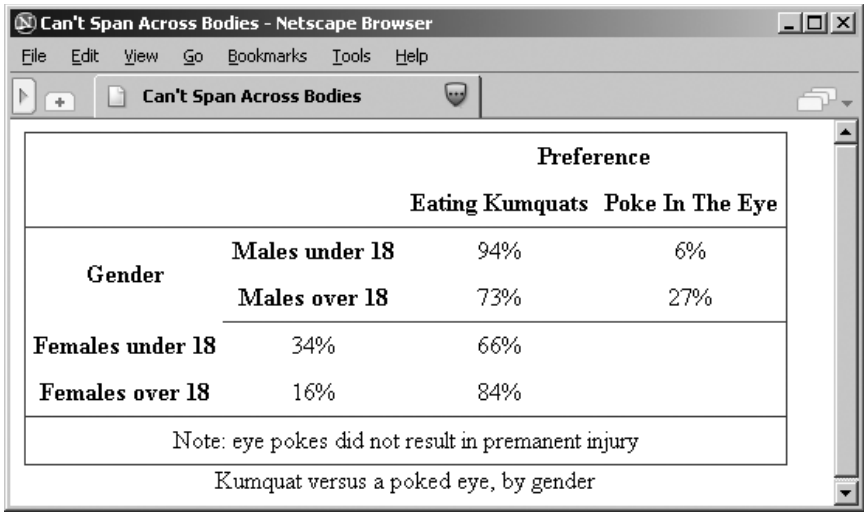

Рис. 10.7. *Сегменты <tbody> разделяют таблицу на части, объединить которые невозможно*

10.3.6. Определение групп столбцов

Основная модель таблиц ориентирована на строки. Иногда, однако, легче обращаться с таблицей как с набором столбцов. С помощью тегов `<colgroup>` и `<col>`, появившихся первоначально в Internet Explorer в качестве расширений, теперь и стандарты HTML 4 и XHTML позволяют вам повернуть таблицу и думать о ней в терминах столбцов.

В отличие от тегов, описанных в предыдущих разделах, которые могут перемежаться со строками таблицы при определении колонтитулов и фрагментов таблицы, теги столбцов не могут смешиваться с содержимым таблицы. Необходимо размещать их в самом начале таблицы, до ее содержимого. Они определяют схему, в соответствии с которой броузеры, соответствующие стандартам HTML 4 и XHTML, будут отображать столбцы.

10.3.7. Тег <colgroup>

Тег <colgroup> определяет группу столбцов и может быть использован двумя способами: как одно определение для нескольких идентичных столбцов или как контейнер для нескольких не похожих друг на друга столбцов. Тег <colgroup> может появляться только в теге <table>. Закрывающий тег </colgroup> редко используется в HTML, но является обязательным в XHTML. В языке HTML <colgroup> заканчивается, когда встречаются теги <colgroup>, <thead>, <tbody>, <tfoot> или <tr>.

Все популярные броузеры поддерживают тег <colgroup> и его атрибуты.

<colgroup>

Функция:	Определяет группу строк в таблице
Атрибуты:	align, char, charoff, class, dir, id, lang, onClick, onDblClick, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, span, style, title, valign, width
Закрывающий тег:	</colgroup>; обычно опускается в HTML
Содержит:	*column_content* (содержимое столбца)
Может содержаться в:	*table_content* (содержимое таблицы)

10.3.7.1. Атрибут span

Используйте атрибут span для достижения группировки столбцов первого типа. Значение атрибута span – это целое число столбцов, на которые действует тег <colgroup>. К примеру, таблица с шестью столбцами – четырьмя в первой группе и двумя во второй – будет выглядеть в исходном тексте так:

```
<colgroup span="4">
<colgroup span="2">
```

Когда поддерживающий стандарты HTML 4 и XHTML броузер будет разбирать ячейки таблицы по столбцам, он отнесет первые четыре ячейки каждой строки в первую группу столбцов, а две следующие ячейки – во вторую группу. Все другие атрибуты, содержащиеся в отдельном теге <colgroup>, будут применяться к столбцам, входящим в эту группу.

10.3.7.2. Когда использовать span, а когда <col>

Когда тег <colgroup> используется в качестве контейнера для различных столбцов, следует забыть про атрибут span, но необходимо включить в каждый тег <colgroup> отдельный тег <col> для каждого столбца группы. К примеру, на HTML:

```
<colgroup>
  <col>
  <col>
```

```
    <col>
    <col>
  <colgroup>
    <col>
    <col>
```

Этот метод создает группы с тем же числом столбцов в каждой, что получалось при использовании атрибута `span`, но он позволяет определять атрибуты для каждого столбца по отдельности. По-прежнему можно определять атрибуты для всей группы через тег `<colgroup>`, но при необходимости можно заместить эти атрибуты, переопределив их в тегах `<col>`.

К примеру, допустим, что вы хотите, чтобы каждый столбец первой группы занимал 20% таблицы, а оставшиеся два столбца занимали по 10% всей ширины таблицы. Этого легко достичь при помощи атрибута `span`:

```
<colgroup span=4 width="20%">
<colgroup span=2 width="10%">
```

Эта же структура может быть получена при определении индивидуальных столбцов (на HTML):

```
<colgroup width="20%">
  <col>
  <col>
  <col>
  <col>
<colgroup width="10%">
  <col>
  <col>
```

Вы можете использовать в одной таблице оба способа. К примеру, мы могли бы определить ту же, что и в предыдущем примере, группировку столбцов той же ширины так:

```
<colgroup span=4 width="20%" align=right>
<colgroup width="10%">
  <col align=left>
  <col align=right>
```

Обратите внимание, что здесь мы смогли по-разному выровнять содержимое двух последних столбцов (по умолчанию содержимое центрируется).

10.3.7.3. Другие атрибуты тега <colgroup>

Множество общих для тегов таблицы атрибутов управляет хорошо знакомыми вам аспектами отображения каждого столбца в группе, заданной тегом `<colgroup>`. Эти атрибуты принимают те же значения и ведут себя так же, как эквивалентные атрибуты тега `<td>`.

10.3.8. Тег <col>

Используйте тег <col> для управления внешним видом одного или нескольких столбцов в группе.

Тег <col> может появляться только в теге <colgroup> в таблице. У этого тега нет содержимого и, следовательно, нет закрывающего тега в HTML. Используйте </col> или одну наклонную линию в конце тега (<col />), чтобы удовлетворять требованиям XHTML об обязательном завершении тегов. Тег <col> предоставляет один или несколько столбцов в группе <colgroup>, к которым поддерживающий стандарты HTML 4 и XHTML броузер применяет атрибуты тега <col>.

Все популярные броузеры поддерживают тег <col> и его атрибуты.

<col>

Функция:	Определяет столбец в группе столбцов
Атрибуты:	align, char, charoff, class, dir, id, lang, onClick, onDblClick, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, span, style, title, valign, width
Закрывающий тег:	В HTML отсутствует; </col> или <col ... /> в XHTML
Содержит:	Ничего
Может содержаться в:	*column_content* (содержимое столбца)

10.3.8.1. Атрибут span

Атрибут span в теге <col>, как и в теге <colgroup>, позволяет указать, сколько последовательно идущих столбцов подвергаются воздействию тега <col>. По умолчанию <col> действует только на один столбец. Для примера создадим группу <colgroup>, содержащую пять столбцов. Мы выровняем первый и последний столбцы влево и вправо соответственно, тогда как три средние будут центрированы:

```
<colgroup>
  <col align=left>
  <col align=center span=3>
  <col align=right>
```

Тег <col> следует использовать только в тегах <colgroup>, не содержащих атрибута span. В противном случае броузеры, соответствующие стандартам HTML 4 и XHTML, проигнорируют теги <col> и их атрибуты.

10.3.8.2. Другие атрибуты тега <col>

Множество общих для всех тегов таблицы атрибутов, управляющих аспектами отображения столбца, можно использовать и с тегом <col>.

Эти атрибуты принимают те же значения и ведут себя так же, как эквивалентные атрибуты тега <td>.

10.3.9. Использование групп столбцов

Группы столбцов использовать легче, чем это может показаться. Думайте о них как о шаблоне, в соответствии с которым будут формироваться столбцы вашей таблицы. Их главное назначение состоит в создании групп, которые будут разделяться более толстыми линиями, и в организации процесса массового применения атрибутов форматирования ко всем ячейкам в одном или нескольких столбцах.

Возвращаясь к нашему примеру таблицы, мы можем провести более толстые линии между столбцами меток и ячейками данных, поместив столбцы с метками в одну группу, а ячейки данных – в другую (на HTML).

```
<table border= cellspacing=0 cellpadding=5 rules=groups>
    <caption align=bottom>Kumquat versus a poked eye, by gender</caption>
    <colgroup span=2>
    <colgroup span=2>
    <thead>
      <tr>
        <td colspan=2 rowspan=2>
        <th colspan=2 align=center>Preference
      <tr>
        <th>Eating Kumquats
        <th>Poke In The Eye
    <tbody>
      <tr align=center>
        <th rowspan=4>Gender
        <th>Males under 18
        <td>94%
        <td>6%
      <tr align=center>
        <th>Males over 18
        <td>73%
        <td>27%
      <tr align=center>
        <th>Females under 18
        <td>34%</td>
        <td>66%</td>
      <tr align=center>
        <th>Females over 18
        <td>16%
        <td>84%
    <tfoot>
      <tr>
        <td colspan=4 align=center>
          Note: eye pokes did not result in permanent injury
  </table>
```

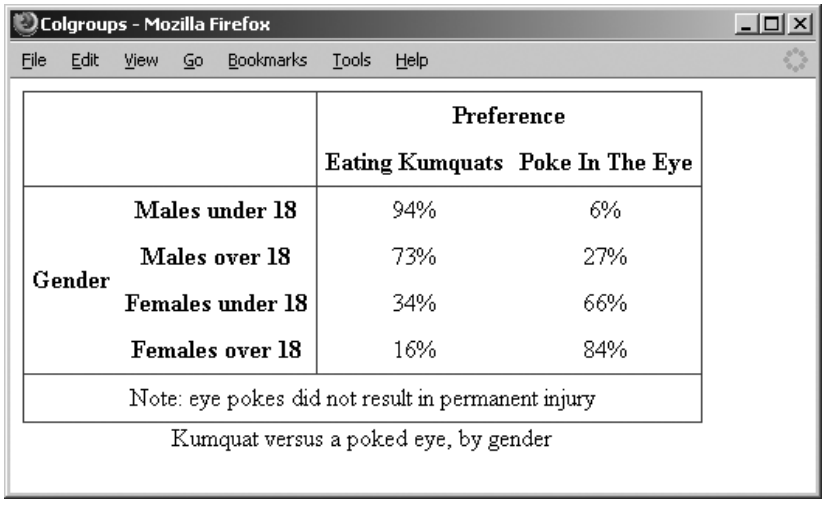

Рис. 10.8. *Пример, демонстрирующий различные средства таблиц, появившиеся в стандартах HTML 4 и XHTML*

Результат показан на рис. 10.8. Все, что мы добавили, – это два тега `<colgroup>`. Дополнительные линейки проведены благодаря действию атрибута `rules=group` в теге `<table>`. Чтобы между группами столбцов была проведена рамочная линия, атрибуту `rules` необходимо присвоить одно из значений: `groups`, `cols` или `all`.

10.4. За пределами обычных таблиц

С виду таблицы представляются весьма заурядной вещью – просто способом, которым пользуются ученые и другие любители сухих материй, располагая элементы в строках и столбцах, чтобы их было легче сравнивать. Однако если заглянуть глубже, то можно понять, что таблицы – это необыкновенная вещь. Если не считать тега `<pre>`, тег `<table>` со своими атрибутами предоставляет единственный способ управления *макетом* документа. Содержимое тега `<pre>`, разумеется, весьма ограничено. Таблицы, напротив, могут содержать практически все, что допустимо в качестве содержимого тела документа, включая мультимедийные элементы и формы. При этом структура таблицы позволяет явно управлять тем, где в окне пользовательского броузера появятся эти элементы. При подходящей комбинации атрибутов таблицы позволяют располагать текст в нескольких колонках. Они позволяют также сделать формы более легкими для чтения, понимания и заполнения. И это только начало.

Мы не знаем, можем ли мы рекомендовать вам увлекаться макетированием страниц – при помощи таблиц или без них. Помните, главное – не внешний вид, а содержание. Однако...

Невозможно спорить с тем, что, по меньшей мере, таблицы данных выигрывают от возможности контролировать макет, а за ними сразу следуют формы. Таблицы предоставляют единственный способ создания предсказуемого, независимого от броузера макета для ваших страниц. Используемые с умеренностью и заполненные качественным содержимым, таблицы представляют собой орудие, которым должен владеть каждый автор.

И теперь, когда мы пробудили у вас интерес к макетированию страниц при помощи таблиц, не отчаивайтесь, решив, что мы, заканчивая эту главу, оставим вас без примеров, – у нас есть несколько в главе 17.

- Обзор фреймов
- Теги фреймов
- Макетирование фреймов
- Содержимое фреймов
- Тег <noframes>
- Встроенные фреймы
- Окна и фреймы в качестве цели
- Модель XFrames

11

Фреймы

Начиная с Netscape Navigator 2.0, HTML-авторы получили возможность разделять основное окно броузера на несколько независимых *фреймов* (*frames*), в каждом из которых отображался свой документ. Получалось что-то вроде стены из мониторов в операторской комнате на телевидении. Компания Netscape изобрела фреймы в середине 1990-х. Немедленно завоевавшие популярность фреймы теперь входят в стандарты HTML 4 и XHTML.

11.1. Обзор фреймов

На рис. 11.1 показан простой пример фреймов. На нем окно документа разбито на строки и столбцы отдельных фреймов, отделенных друг от друга линейками и полосами прокрутки.[1] Хотя этого и не видно на рисунке, каждый фрейм в окне отображает свой документ. В каждом из фреймов может находиться любое содержимое, которое броузер умеет отображать, включая XHTML-документы и мультимедийные презентации. Если в содержимом фрейма есть гиперссылка, которую выбирает пользователь, содержимое нового документа – даже если это еще один документ с фреймами – может отобразиться в том же фрейме, в другом фрейме или занять все окно броузера.

Фреймы создаются особым фреймовым документом – фреймсетом. Его содержимое не отображается. Фреймовый документ содержит особые теги, которые сообщают броузеру, как следует разделить основное окно на отдельные фреймы и какие документы нужно в них разместить.

---

[1] К обилию фреймов на странице следует относиться очень осторожно, т. к. в устоявшейся практике HTML-проектирования множество фреймов на странице часто расценивается как «архитектурное излишество» и признак дурного вкуса. – *Примеч. науч. ред.*

Отдельные документы, на которые ссылается фреймовый документ и которые отображаются во фреймах, живут до известной степени независимой жизнью. Фреймовый документ управляет окном в целом. Можно, однако, приказать документу из одного фрейма разместить новое содержимое в другом фрейме. Это делается путем присваивания фрейму имени и объявления этого фрейма мишенью гиперссылки при помощи специального атрибута тега гиперссылки <a>.

11.2. Теги фреймов

Для создания фреймового документа нужно знать только два тега: <frameset> и <frame>. Вдобавок к ним стандарты HTML 4 и XHTML предлагают тег <iframe>, который можно использовать для создания *встроенных* (*inline*), или *плавающих* (*floating*) фреймов, и тег <noframes> для броузеров, неспособных выводить фреймы.

Тег <frameset>, образуя *набор фреймов* (*фреймсет*, *frameset*), позволяет создавать фреймы в окне броузера. Атрибуты определения строк и столбцов для тега <frameset> позволяют вам определить число и начальные размеры столбцов фреймов. Тег <frame> определяет, какой документ – HTML или какой-то другой – попадает поначалу в определенный фрейм, а также служит местом, где можно определить имя фрейма, которое затем может использоваться в гиперссылках.

Вот исходный текст HTML-страницы, изображенной на рис. 11.1:

```
<html>
<head>
<title>Frames Layout</title>
</head>
<frameset rows="60%,*" cols="65%,20%,*">
  <frame src="frame1.html">
  <frame src="frame2.html">
  <frame src="frame3.html" name="fill_me">
  <frame scrolling=yes src="frame4.html">
  <frame src="frame5.html">
  <frame src="frame6.html" id="test">
  <noframes>
    Sorry, this document can be viewed only with a
    frames-capable browser.
    <a href = "frame1.html">Take this link</a>
    to the first HTML document in the set.
  </noframes>
</frameset>
</html>
```

Обратите внимание на несколько обстоятельств в приведенном примере и в его отображении на рис. 11.1. Во-первых, броузер заполняет фреймы по строкам, во-вторых, четвертый фрейм выставил полосы прокрутки, поскольку мы ему приказали это сделать, несмотря на то, что его содержимое и так помещается во фрейм безо всякой прокрут-

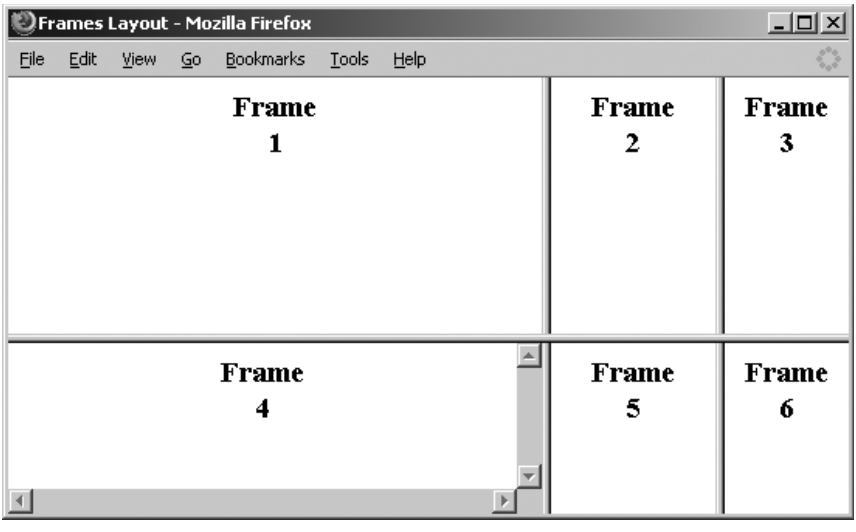

Рис. 11.1. *Простой фреймовый макет, состоящий из шести фреймов*

ки. (Полосы прокрутки появляются автоматически, когда содержимое переполняет фрейм, если только их появление не запрещено при помощи атрибута `scrolling` в теге `<frame>`.)

Также представляет интерес атрибут `name` в одном из тегов фреймов. Дав фрейму имя[1], можно указать его в качестве места, в котором следует расположить запрашиваемый документ, или выполнить некоторое автоматизируемое действие. Чтобы добиться этого, нужно добавить специальный атрибут `target` в тег гиперссылки `<a>`. Например, чтобы направить документ под названием `new.html` в третий фрейм, которому мы присвоили имя «fill_me», следует написать гиперссылку примерно так:

```
<a href="new.html" target="fill_me">
```

Если пользователь выберет эту ссылку, скажем, в первом фрейме, документ *new.html* заместит в третьем находящийся там документ *frame3.html*. [атрибут target тега <a>, 11.7.1]

11.2.1. Что следует размещать во фреймах?

Всякий, кто открывал на экране своего компьютера несколько окон одновременно, чтобы сравнить содержимое документов или чтобы оперировать взаимодействующими приложениями, инстинктивно почувствует силу фреймов.

---

[1] Речь идет не об идентификаторе (`id`), хотя этот атрибут существует и для фреймов и может применяться для определения других элементов HTML/XHTML в качестве целей для гиперссылок. – *Примеч. науч. ред.*

Одно простое применение фреймов[1] состоит в том, чтобы поместить в один фрейм общее для всего собрания документов содержимое, такое как информация о защите авторских прав, рекомендации и навигационные инструменты, размещая все другое содержимое документов в соседнем фрейме. Когда пользователь будет переходить к новым страницам, они будут загружаться во фрейм с переменным содержимым, тогда как содержимое первого фрейма будет оставаться на месте.

Описываемое в более содержательном фреймовом документе окружение предоставляет средства навигации для коллекции ваших документов. К примеру, пусть один фрейм содержит оглавление и различные средства поиска для вашей коллекции, а в другом фрейме находится выбранное пользователем содержимое документов. Когда пользователи загружают ваши различные страницы во фрейм с содержимым, они не теряют из виду навигационные инструменты, расположенные в другом фрейме.

Другое удачное использование фреймовых документов служит для сравнения возвращенных пользователю результатов заполнения формы с ее оригиналом с целью проверки правильности ее заполнения. Поместив форму в один фрейм, а результат ее отправки – в другой, вы позволяете пользователю быстро проверить, соответствует ли результат введенным данным. Если результат неправильный, форма, которую нужно снова заполнить, оказывается у пользователя под рукой.

11.3. Макетирование фреймов

Макетирование фреймов подобно макетированию таблиц. Используя тег `<frameset>`, можно расположить фреймы по строкам и столбцам, задавая одновременно их абсолютные или относительные размеры.

11.3.1. Тег <frameset>

Тег `<frameset>` позволяет определять набор фреймов и управлять их размерами и обрамлением. Используйте тег `<frameset>` для определе-

[1] Веб-разработчики не любят фреймы. В том числе из-за того, что сайты с фреймами некорректно индексируются некоторыми поисковиками; на страницах, являющихся содержимым фреймов, отсутствует навигация (пользователь, пришедший по ссылке с поисковой машины, не сможет попасть на другие страницы данного сайта); содержимое фрейма, не помещающегося в окне броузера, невозможно просмотреть. В адресной строке броузера не отображаются URL страниц, загруженных во фреймы (виден только URL фреймового документа). Некоторые версии популярных броузеров при установке закладок сохраняют вместо конкретной комбинации загруженных во фреймы документов ссылку на исходный фреймовый документ. Поэтому, собираясь применять фреймы, взвесьте все «за» и «против» и проработайте навигацию, чтобы избежать нареканий со стороны пользователей. – *Примеч. науч. ред.*

<frameset>	
Функция:	Определяет набор фреймов
Атрибуты:	border, bordercolor, class, cols, frameborder, framespacing, id, onLoad, onUnload, style, title
Завершающий тег:	</frameset>; присутствует обязательно
Содержит:	*frameset_content* (содержимое фреймсета)
Может содержаться в:	*html_content* (содержимое html)

ния наборов фреймов. Тег <frameset> допускает вложения, что предоставляет богатые возможности для макетирования.

Используйте во фреймовых документах тег <frameset> на месте тега <body>. Во фреймовом документе не может быть никакого другого содержимого, помимо тегов <head> и <frameset>. Попытка добавить описание фреймов в обычный документ, содержащий раздел <body>, может повлечь за собой непредсказуемое поведение броузера.

11.3.1.1. Атрибуты rows и cols

В теге <frameset> обязательно должен присутствовать, по меньшей мере, один из двух атрибутов: rows или cols. Они определяют число и размер столбцов (cols) или строк (rows), в которые упорядочены фреймы и вложенные наборы фреймов. Оба атрибута принимают в качестве значений заключенные в кавычки списки величин, определяющих либо абсолютную (в пикселах), либо относительную (в процентах от оставшегося пространства) ширину (для столбцов) или высоту (для строк) фреймов. Число элементов в списке определяет число строк и столбцов с фреймами, которые броузер отобразит в окне фреймового документа.

Как и в случае с таблицами, броузер попытается выполнить указания относительно размеров фреймов с наилучшей возможной точностью. Броузер тем не менее не станет растягивать основное окно, если описанный набор фреймов в нем не помещается, как и не оставит в окне пустого места, если описанный набор фреймов не заполняет всего окна. Вместо этого броузер выделит каждому фрейму пространство, пропорциональное описанным ширине и высоте, и в точности заполнит все окно документа. (Заметили ли вы, что в окне фреймового документа нет полос прокрутки?)

Например:

```
<frameset rows="150,300,150">
```

Такой тег создаст три строки фреймов, каждая из которых занимает в ширину все окно документа. Первая и последняя строка описаны как имеющие высоту 150 пикселов, а вторая строка – 300. В реальности, если только окно броузера не равно в точности 600 пикселам в высоту, броузер автоматически и пропорционально растянет или сожмет

первую и последнюю строку так, чтобы каждая из них занимала четверть окна по высоте. Средняя строка займет оставшуюся половину высоты окна.

Выражение размеров строк и столбцов в процентах от размеров окна представляется более осмысленным. К примеру, нижеприведенный текст функционально идентичен предыдущему:

```
<frameset rows="25%,50%,25%">
```

Разумеется, если сумма процентных значений не равна 100 процентам, броузер автоматически и пропорционально изменит размеры строк, компенсируя несоответствие.

Если вы похожи на нас, то умение складывать числа – не самая сильная ваша сторона. Возможно, некоторые конструкторы фреймов тоже встречаются с этим затруднением, вот почему они включают очень удобную звездочку в списки значений атрибутов `rows` и `cols` в теге `<frameset>`. Звездочка предписывает броузеру установить размеры соответствующих строки и столбца такими, чтобы заполнить оставшееся после размещения смежных фреймов место в окне.

К примеру, если броузер встречает следующий тег:

```
<frameset cols="100,*">
```

он создает столбец фиксированной ширины в 100 пикселов, а затем создает столбец, занимающий все оставшееся в окне пространство.

Вот более интересный пример макетирования:

```
<frameset cols="10,*,10">
```

Этот тег создает два очень узких столбца по краям окна и предоставляет оставшееся посередине место второму столбцу.

Звездочку можно использовать неоднократно в списке размеров строк или столбцов. В этом случае соответствующие строки и столбцы делят доступное пространство поровну. К примеру:

```
<frameset rows="*,100,*">
```

создает в середине окна строку высотой в 100 пикселов и строки одинаковой высоты под и над ней.

Если перед звездочкой поставлено целое число, соответствующая строка или столбец получит пропорционально большую долю свободного пространства. К примеру:

```
<frameset cols="10%,3*,*,*">
```

создает четыре столбца. Первый столбец занимает 10 процентов всей ширины окна. Затем броузер отведет второму столбцу 3/5 оставшегося пространства, а третьему и четвертому столбцам – по 1/5 оставшегося пространства.

Использование звездочки (особенно с числовым префиксом) облегчает распределение свободного пространства в окне броузера.

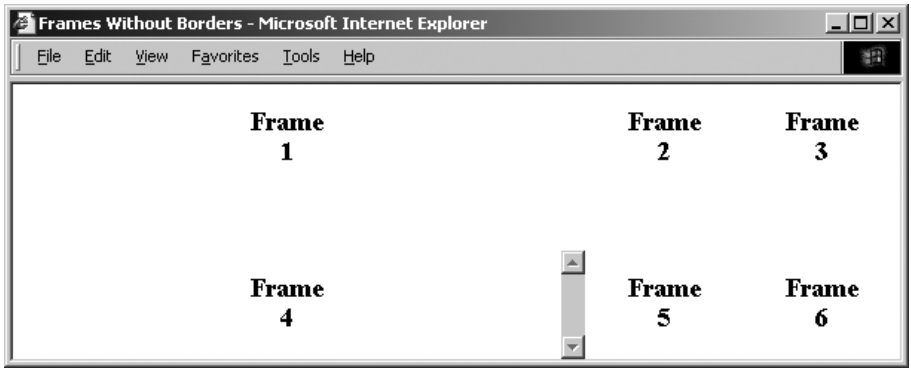

Рис. 11.2. *Атрибуты border и frameborder позволяют вам убирать рамки между фреймами*

Имейте также в виду, что, если только это не будет явно запрещено, броузер позволит пользователю изменять размеры отдельных строк и столбцов фреймового документа, а следовательно, изменять относительную долю пространства, которое занимает каждый фрейм в окне броузера этого пользователя. Чтобы предотвратить эти изменения, используйте атрибут `noresize`[1] для тега `<frame>`. [тег <frame>, 11.4.1]

11.3.1.2. Атрибуты border, frameborder, framespacing и bordercolor

Популярные броузеры предоставляют дополнительные атрибуты, расширяющие возможности тега `<frameset>`, которые можно использовать для определения и варьирования параметров рамок, окружающих фреймы. Стандарты HTML 4 и XHTML предлагают устанавливать эти свойства обрамления при помощи каскадных таблиц стилей (Cascading Style Sheet, CSS).

По умолчанию любой фрейм в наборе фреймов окружается тонкой «трехмерной» рамкой (см. рис. 11.1). Вы можете сделать эти рамки толще или вовсе избавиться от них с помощью атрибута `border` тега `<frameset>`. Нулевое значение атрибута `border` убирает рамки (см. рис. 11.2). Значение 1 задает толщину, принятую по умолчанию. Чтобы одновременно увеличить толщину рамок у всех фреймов в наборе, присвойте атрибуту `border` целое число, превышающее 1.

Атрибут `frameborder` со значением `1` или `yes` включает вывод рамок, а значение `0` или `no` убирает их. Атрибут `framespacing` с целочисленным

[1] Использование атрибута `noresize` считается дурным тоном. Невозможно заранее предугадать, какого размера будет окно броузера у заглянувшего на ваш сайт пользователя. В таких условиях полной неопределенности никакая дополнительная возможность регулировки не будет лишней. – *Примеч. науч. ред.*

значением 1 и более увеличивает рамку между фреймами. Для чего нужны два отдельных атрибута, дающие тот же эффект, что и один атрибут border? Причины, главным образом, исторические. Достаточно сказать, что некоторая путаница существует до сих пор. Все популярные броузеры поддерживают атрибут border для тега <frameset>, так что пользуйтесь им, а не альтернативами.

Все популярные броузеры, кроме Opera (причины «отступничества» этого броузера неизвестны), позволяют вам управлять цветом рамок фреймов с помощью атрибута bordercolor (рис. 11.3). В качестве значения он принимает стандартное название цвета или шестнадцатеричный триплет. Например, хотя это и не видно на черно-белой иллюстрации, рамки фреймов на рис. 11.3 имеют светло-зеленый цвет, соответствующий RGB-значению «00CC00». (Для ясности мы увеличили ширину рамок с помощью атрибута border.) Полный список названий цветов и соответствующих значений приведен в приложении G.

11.3.1.3. Фреймы и JavaScript

Все популярные броузеры поддерживают обработчики событий JavaScript-, позволяющие вашим фреймовым документам реагировать на первоначальную загрузку и на изменения размеров фреймов (onLoad); на выгрузку документа пользователем (onUnload); на перемещение фокуса ввода от окна, содержащего фреймовый документ, к другому объекту, например к другому окну (onBlur); на активизацию окна, содержащего фреймы (onFocus). Включенные в виде атрибутов в тег <frameset>, эти обработчики событий принимают в качестве значений заключенные в кавычки списки JavaScript-команд и вызовов функций. Например, можно уведомить пользователя о том, что все содержимое наконец загружено в соответствующие фреймы большого набора фреймов:

```
<frameset onLoad="window.alert('Все необходимое загружено.
                                Вы можете продолжать просмотр.')">
```

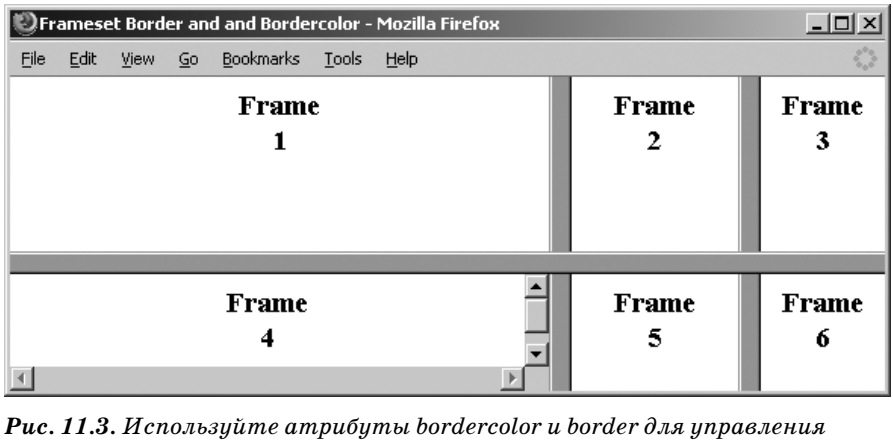

Рис. 11.3. *Используйте атрибуты bordercolor и border для управления цветом и толщиной рамок между фреймами*

Эти четыре атрибута могут использоваться и в теге <body>. Мы подробно обсудим обработчики событий JavaScript в разделе 12.3.3.

11.3.1.4. Другие атрибуты тега <frameset>

Подобно большинству стандартных тегов тег <frameset> допускает использование с ним четырех стандартных атрибутов: class, style, title и id.

Используйте атрибут class для ассоциирования заранее определенного класса стилей с этим фреймом и, в силу наследования стилей, с его содержимым в качестве альтернативы, используйте атрибут style для непосредственного определения стилей в теге <frameset>. Мы подробнее обсуждали стили в главе 8.

Атрибут id создает уникальный идентификатор фрейма, тогда как title создает название фрейма, которое может быть представлено пользователю или использовано невизуальным броузером. [атрибут id, 4.1.1.4] [атрибут title, 4.1.1.5]

11.3.2. Вложение тегов <frameset>

Можно создать достаточно сложную страницу, использовав только один тег <frameset>, но ее макет будет довольно пресным. Вкладывая в тег <frameset> верхнего уровня новые теги <frameset>, можно создать несимметричные фреймы и другие усложненные макеты.[1]

К примеру, создадим макет из двух столбцов, в первом из которых будет две строки, а во втором – три (как показано на рис. 11.4), вложив

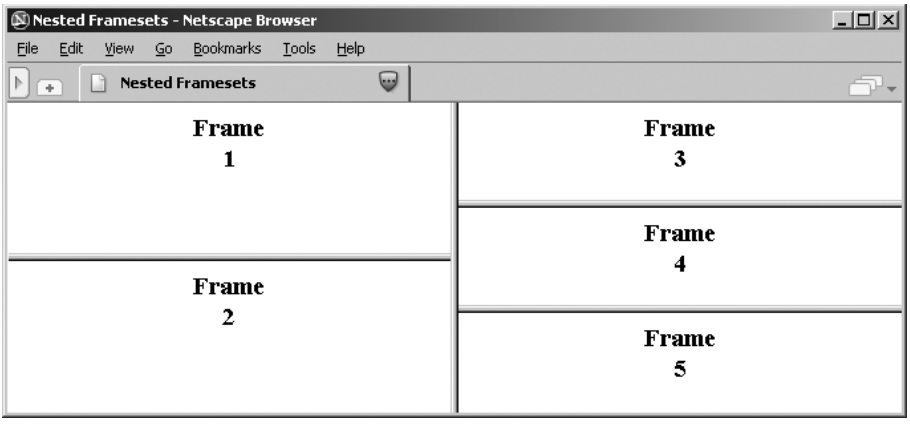

Рис. 11.4. *Использование вложенных тегов <frameset> для расположения фреймов «друг в друге»*

[1] Постарайтесь не переусердствовать с вложенными фреймами. Веб-дизайнеры считают, что если уж пришлось использовать фреймы, то по крайней мере их не должно быть больше трех. – *Примеч. науч. ред.*

два тега <frameset> с описанием строк в тег <frameset> верхнего уровня, в котором описаны столбцы.

```
<frameset cols="50%,*">
  <frameset rows="50%,*">
    <frame src="frame1.html">
    <frame src="frame2.html">
  </frameset>
  <frameset rows ="33%,33%,*">
    <frame src="frame3.html">
    <frame src="frame4.html">
    <frame src="frame5.html">
  </frameset>
</frameset>
```

11.4. Содержимое фреймов

Ничто из содержимого фреймового документа (фреймсета) не отображается, за исключением, быть может, сообщения для броузера, не поддерживающего фреймов. Теги <frame>, заключенные в один или несколько тегов <frameset> (которые составляют содержимое фреймового документа), предоставляют ссылки в виде URL на соответствующие документы, помещаемые в каждый фрейм. [тег <noframes>, 11.5]

11.4.1. Тег <frame>

Тег <frame> может появляться только внутри тега <frameset>. Используйте атрибут src для указания URL документа, который должен быть помещен в соответствующий фрейм.

Фреймы размещаются в каждом фреймовом документе по строкам – столбец за столбцом, слева направо, а затем строка за строкой сверху вниз, поэтому последовательность и число тегов <frame> имеют значение.

Теги <frame>, не содержащие атрибута src, будут отображены пустыми. Также незаполненными останутся фреймы, описанные в теге <frameset>, которым не хватило тегов <frame>, если, например, ваш документ с фреймами требует три столбца, а вы предоставили только два. Эти сироты так и останутся пустыми – впоследствии в них нельзя помес-

<frame>

Функция:	Определяет отдельный фрейм в <frameset>
Атрибуты:	bordercolor [IE], class, frameborder [IE], id, longdesc, marginheight, marginwidth, name, noresize, scrolling, src, style, title
Завершающий тег:	</frame>; редко включается в HTML
Содержит:	Ничего
Может содержаться в:	*frameset_content* (содержимое фреймсета)

тить какое-либо содержимое, даже если у них есть имя, которое может использоваться в качестве метки при перенаправлении документов. [атрибуты name и id, 6.3.1.3]

11.4.1.1. Атрибут src

Значением атрибута `src` в теге `<frame>` является URL документа, который должен отображаться в этом фрейме. Не существует другого способа заполнить фрейм содержимым. Не следует, в частности, включать во фреймовый документ тег `<body>` – броузер проигнорирует теги фреймов и просто отобразит содержимое тега `<body>`, если встретит его первым, и наоборот.

Атрибут `src` может ссылаться на любой действительный документ или любой отображаемый объект, включая изображения и мультимедийные объекты. В частности, документ, на который указывает `src`, может сам состоять из одного или нескольких фреймов. Фреймы будут выведены в тот фрейм, который на них ссылается, что дает еще одно средство создания сложных макетов с использованием вложенных фреймов.

Поскольку фрейм может ссылаться на целый документ, все возможности HTML/XHTML реализуются внутри фрейма, включая определения фона, цветов, таблиц, шрифтов и тому подобного. К несчастью, это означает также, что разные фреймы, содержащиеся в одном окне броузера, могут конфликтовать друг с другом. В частности, если у каждого вложенного во фрейм документа (не обычного HTML- или XHTML-документа) имеется свой тег `<title>`, в качестве названия всего окна броузера будет выбрано название фреймового документа, загруженного в последнюю очередь. Самый легкий способ избежать этой проблемы состоит в том, чтобы присвоить всем связанным фреймовым документам одно и то же название.[1]

11.4.1.2. Атрибуты name и id

Необязательный атрибут `name` тега `<frame>` связывает с фреймом метку, которая впоследствии может использоваться для ссылок на фрейм в атрибуте `target` тега гипертекстовой ссылки `<a>` и тега `<form>`. Таким образом, можно изменить содержимое фрейма, выбрав гиперссылку в другом фрейме. В противном случае, как это делается в обычном окне броузера, документ, на который указывает гиперссылка, заменит во фрейме документ, содержащий эту гиперссылку. Мы обсудим подробнее имена фреймов и атрибут `target` далее в этой главе. [атрибут target тега <a>, 11.7.1]

---

[1] Еще более простое решение – разместить тег `<title>` в документе с исходным фреймсетом (ведь именно эта строка будет использована броузером в качестве названия окна броузера), тем более что наличие тега `<title>` в документе необходимо по стандартам HTML 4/XHTML. – *Примеч. науч. ред.*

Аналогичным образом, атрибут id уникально идентифицирует фрейм, но броузеры не поддерживают его использование в качестве цели гиперссылки, хотя применение атрибута id в этом качестве в других тегах они допускают.

Значением атрибута name или id служит строка текста, заключенная в кавычки.

11.4.1.3. Атрибут noresize

Даже если размеры фреймов указаны явным образом при помощи атрибутов тега <frameset>, пользователь может вручную изменить размеры строк и столбцов фреймов. Чтобы помешать пользователю это сделать, включите атрибут noresize в теги <frame> тех строк и столбцов, относительные размеры которых вы не разрешаете изменять. Для фреймовых документов два на два, например, атрибут noresize в любом из четырех тегов <frame> заморозит относительные размеры всех фреймов.

Атрибут noresize особенно полезен для фреймов, содержащих изображения фиксированного размера, например кнопочную панель или логотип, используемые для рекламы. Зафиксировав размер фреймов, в котором в точности помещается изображение, и выставив в нем атрибут noresize, вы гарантируете, что изображение будет выведено так, как вы этого хотите, а оставшаяся часть[1] окна броузера остается в распоряжении прочих фреймов документа.

11.4.1.4. Атрибут scrolling

Если содержимое фрейма не помещается в него целиком, то броузер выведет фрейм с вертикальными и горизонтальными полосами прокрутки. Если для содержимого хватает места, полосы прокрутки исчезнут. Атрибут scrolling в теге <frame> позволяет явным образом управлять тем, появятся ли полосы прокрутки или нет.

Если в теге написано scrolling="yes", все популярные броузеры, кроме Netscape, вставят в описываемый фрейм полосы прокрутки, даже если в нем нечего прокручивать. Если присвоить атрибуту scrolling значение no[2], полосы прокрутки не появятся даже в том случае, когда содержимое фрейма больше его самого. Значение auto по умолчанию работает так же, как если бы вы не включали атрибут scrolling в тег.

---

[1] При условии, конечно, что в этот остаток что-нибудь поместится (что, вообще говоря, никто не гарантирует). Еще раз напоминаю, что использование атрибута noresize (во фреймах, которые и без того не одобряются сетевой общественностью) является дурным тоном. – *Примеч. науч. ред.*

[2] Будьте внимательны: если содержимое не помещается в окно фрейма при установке scrolling=no, пользователь никаким образом не увидит содержимое документа, которое не попало в видимую область. – *Примеч. науч. ред.*

11.4.1.5. Атрибуты marginheight и marginwidth

Обычно броузер оставляет небольшое расстояние между краем фрейма и его содержимым. Эти поля можно изменять при помощи атрибутов `marginheight` и `marginwidth`, каждый из которых принимает значения, равные точному числу пикселов, которые следует оставить вокруг содержимого фрейма.

Нельзя сделать поля уже, чем в один пиксел, или сделать их такими большими, чтобы не осталось места для содержимого фрейма. Это потому, что они, как и большинство других атрибутов в HTML, только советуют броузеру, но не приказывают ему. Если броузер не может в точности удовлетворить спецификациям полей, он проигнорирует их и выведет фрейм, как сумеет.

11.4.1.6. Атрибуты frameborder и bordercolor

В ранних версиях броузера Internet Explorer добавить или убрать рамку вокруг отдельного фрейма можно при помощи атрибута `frameborder`. Значения `yes` или `1` и `no` или `0` соответственно включают или выключают рамку вокруг фрейма и при установке имеют приоритет перед спецификациями атрибута `frameborder`, заданными в любом содержащем этот фрейм теге `<frameset>`. Не применяйте его.

В популярных броузерах, кроме Opera, можно также управлять цветом рамки отдельного фрейма при помощи атрибута `bordercolor`. Используйте в качестве его значения название цвета или шестнадцатеричный триплет. Если два смежных фрейма имеют различные значения атрибута `bordercolor`, то окончательный цвет рамки не будет определен. Полный список названий цветов и кодов можно найти в приложении G.

11.4.1.7. Атрибуты title и longdesc

Как и в случае большинства других стандартных тегов, можно присвоить фрейму название при помощи атрибута `title`. Значением его служит заключенная в кавычки строка, описывающая содержимое фрейма. Броузеры могут отображать название тогда, например, когда мышь проходит по фрейму.

Если недостаточно атрибута `title`, можно использовать атрибут `longdesc`. Его значение – URL документа, описывающего фрейм. Предположительно, это длинное описание может иметь какой-то альтернативный формат для использования невизуальными броузерами.

11.5. Тег <noframes>

У фреймового документа нет тела, и он не содержит тега `<body>`. Более того, он не должен их содержать, так как броузер проигнорирует любой тег фреймов, если он обнаружит любое содержимое, характерное для тега `<body>`, прежде чем он встретит первый тег `<frameset>`. Фреймовый до-

кумент, таким образом, совершенно невидим для броузеров, не поддерживающих фреймов. Тег <noframes> предоставляет возможность увидеть запрашиваемую страницу тем, чей броузер не отображает фреймов.

<noframes>

Функция:	Доставляет содержимое для отображения не поддерживающими фреймы броузерами
Атрибуты:	class, dir, id, lang, onClick, onDblClick, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, style, title
Завершающий тег:	</noframes>; иногда опускается в HTML
Содержит:	*body_content* (содержимое тела)
Может содержаться в:	*frameset_content* (содержимое фреймсета)

Использовать тег <noframes> следует только внутри самого внешнего тега <frameset> фреймового документа. То, что находится между тегом <noframes> и его обязательным закрывающим тегом </noframes>, не отображается ни одним из поддерживающих фреймы броузеров, но отображается вместо альтернативного содержимого фреймового документа броузерами, которые не умеют обращаться с фреймами. Содержимым тега <noframes> может быть любое обычное содержимое тела документа, включая сам тег <body>.

Хотя этот тег не является обязательным, опытные авторы обычно включают в свои фреймовые документы тег <noframes> с содержимым, которое предупреждает пользователя, броузер которого не поддерживает фреймы, что он не увидит содержимого. А толковые авторы предоставят для таких пользователей какой-либо выход из сложившейся ситуации[1], когда отсутствует прямой доступ к отдельным документам, составляющим содержание фреймов. Помните наш первый в этой главе пример фреймов? Рис. 11.5 показывает, что будет, если фреймовый документ будет загружен старой версией Mosaic.

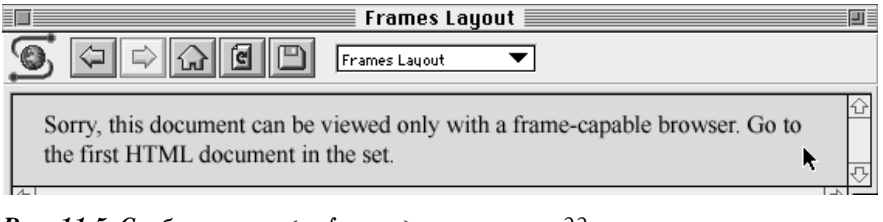

Рис. 11.5. *Сообщение из <noframes> в окне не поддерживающего фреймы броузера*

[1] Обычно таким выходом бывает ссылка на документ, содержащий оглавление, включающее ссылки на все документы, которые могут быть загружены в данный фреймсет. – *Примеч. науч. ред.*

Код HTML, выдающий это сообщение, выглядит так:

```
<noframes>
  Sorry, this document can be viewed only with a
  frame-capable browser. Go to the <a href="frame1.html">
 first HTML document</a> in the set.
</noframes>
```

Причина, по которой <noframes> работает, состоит в том, что броузеры крайне терпимы к ошибочным тегам и некорректным документам. Не поддерживающий фреймы броузер просто игнорирует все теги фреймов. Все, что после этого остается, – это содержимое тега <noframes>, которое броузер добросовестно отображает.

Однако если броузер строго придерживается какой-то версии HTML или XHTML, которая не поддерживает фреймов, он может просто выдать сообщение об ошибке и отказаться отображать документ, если даже он содержит тег <noframes>.

11.5.1. Атрибуты тега <noframes>

У тега <noframes> нет специфических атрибутов, но с ним можно использовать любой из шестнадцати стандартных атрибутов: они позволяют вам пометить (id) или назвать (title) содержимое тега, изменить характеристики отображения содержимого тега (class, style), указать используемый язык (lang) и связанное с ним направление отображения текста (dir) и множество on-атрибутов, позволяющих вам реагировать на инициированные пользователем с помощью мыши и/или клавиатуры события, связанные с содержимым тега <noframes>. [атрибут dir, 3.6.1.1] [атрибут lang, 3.6.1.2] [встроенные стили: атрибут style, 8.1.1] [стилевые классы, 8.3] [атрибут id, 4.1.1.4] [атрибут title, 4.1.1.5] [обработчики событий JavaScript, 12.3.3]

11.6. Встроенные фреймы

До сих пор наше обсуждение было посвящено в основном фреймам, которые определялись в рамках наборов фреймов (фреймсетов), заполняющих окно броузера. Такой фреймсет занимает место традиционного тела документа и предоставляет пользователю содержимое через содержащиеся в нем фреймы.

Стандарты HTML 4 и XHTML позволяют действовать иначе – можно определить фрейм, который будет содержаться внутри традиционного документа и отображаться как часть потока содержимого документа. Эти фреймы ведут себя очень похоже на встроенные в документ изображения, почему их и называют *встроенными* (*inline*) фреймами.

Все популярные броузеры поддерживают встроенные фреймы.

11.6.1. Тег <iframe>

Встроенные фреймы[1] определяются при помощи тега <iframe>. Тег <iframe> не используется в теге <frameset>. Вместо этого он может появиться в вашем документе везде, где может появиться тег <img>. Тег <iframe> определяет прямоугольную область в документе, в которой броузер отображает отдельный документ с его рамкой и полосами прокрутки.

<iframe>

Функция:	Определяет в потоке текста встроенный фрейм
Атрибуты:	align, class, frameborder, height, id, longdesc, marginheight, marginwidth, name, scrolling, src, style, title, width
Завершающий тег:	</iframe>; присутствует обязательно
Содержит:	*body_content* (содержимое тела)
Может содержаться в:	*тексте*

Воспользуйтесь атрибутом src тега <iframe> для определения URL документа, который займет место внутри встроенного фрейма. Все другие атрибуты тега <iframe>, включая атрибуты class, frameborder, id, longdesc, marginheight, marginwidth, name, scrolling, style и title, могут быть опущены. Эти необязательные атрибуты используются подобно аналогичным атрибутам тега <frame>. [тег <frame>, 11.4.1]

Используйте содержимое тега <iframe> для предоставления информации пользователям броузеров, не поддерживающих встроенных фреймов. Поддерживающие встроенные фреймы броузеры проигнорируют это содержимое, тогда как все другие броузеры проигнорируют тег <iframe> и, следовательно, отобразят его содержимое, как если бы оно было регулярным содержимым тела документа. К примеру, используйте содержимое тега <iframe>, чтобы объяснить пользователям, что они потеряли:

```
...предыдущее содержимое документа…
<iframe src="sidebar.html" width=75 height=200 align=right>
Ваш броузер не поддерживает встроенных фреймов. Чтобы увидеть полное
<a href="sidebar.html">документа</a> содержимое, вам необходимо установить
более современную версию броузера.
</iframe>
...последующее содержимое документа
```

В этом примере мы сообщаем пользователю, что он пытается воспользоваться возможностью, которую не поддерживает его броузер, и даем ему ссылку на пропущенный документ.

---

[1] Другие авторы очень часто называют такие фреймы «плавающими». – *Примеч. науч. ред.*

11.6.1.1. Атрибут align

Подобно атрибуту `align` для тегов `<table>` и `<img>` этот атрибут встроенного фрейма позволяет управлять тем, помещается ли фрейм в строку с окружающим текстом или смещается к краю документа, позволяя тексту обтекать его.

Для вывода фрейма в строку используйте значения `top`, `middle` или `bottom` атрибута `align`. Фрейм будет выровнен по верху, середине или по низу окружающего текста соответственно. Чтобы позволить тексту обтекать встроенный фрейм, используйте значения `left` и `right` этого атрибута. Фрейм тогда сместится соответственно влево или вправо по отношению к потоку текста, так что остальное содержимое документа будет обтекать его. Значение `center` помещает встроенный фрейм в центр окна, причем текст будет располагаться выше и ниже фрейма.

11.6.1.2. Атрибуты height и width

Популярные броузеры помещают по умолчанию содержимое встроенных фреймов в контейнер 150 пикселов высотой и 300 пикселов шириной. Чтобы изменить размеры этого контейнера, присвойте атрибутам `height` и `width` желательные величины в пикселах.

11.6.2. Использование встроенных фреймов

Хотя вы, вероятно, будете избегать их в большинстве ваших веб-страниц, встроенные фреймы могут быть полезны, особенно для предоставления информации, связанной с текущим документом, подобно тому, как это делают в традиционных печатных публикациях в примечаниях на полях.

За исключением того, что они расположены внутри традиционного документа, встроенные фреймы ведут себя точно так, как обычные. Можно загружать другие документы во встроенный фрейм, используя его имя (см. следующий раздел), и ссылаться на другие документы из встроенного фрейма.

11.7. Окна и фреймы в качестве цели

Как говорилось в разделе 11.4.1, можно создать метку фрейма, используя в теге `<frame>` атрибут `name`.[1] Снабженный именем (`name`) или идентификатором (`id`) фрейм может служить окном для отображения документа, на который указывает гиперссылка в документе, отображенном в другом фрейме. Осуществить это перенаправление можно, используя в гиперссылке, ссылающейся на этот документ, атрибут `target`.

---

[1] Атрибут `id` также предоставляет возможность создания уникальной метки, но вы не можете использовать его для перенаправления. Броузер проигнорирует целевой кадр, обозначенный с помощью идентификатора, и выведет указанный документ в новом окне.

11.7.1. Атрибут target тега <a>

Если включить в тег <a> атрибут target (цель), броузер загрузит и отобразит документ, упомянутый в атрибуте href, во фрейм или в окно с именем, соответствующим значению атрибута target. Если не существует ни фрейма, ни окна с таким именем или идентификатором, броузер откроет новое окно, присвоит ему указанное имя и загрузит новый документ в это окно. По окончании этого процесса новое окно может служить целью для гиперссылок.

Гиперссылки с заданной целью позволяют легко создавать эффективные навигационные инструменты. Например, документ, содержащий оглавление, может перенаправлять документы в отдельное окно:

```
<h3>Table of Contents</h3>
<ul>
  <li><a href="pref.html" target="view_window">Preface</a>
  <li><a href="chap1.html" target="view_window">Chapter 1</a>
  <li><a href="chap2.html" target="view_window">Chapter 2</a>
  <li><a href="chap3.html" target="view_window">Chapter 3</a>
</ul>
```

Когда пользователь первый раз выберет одну из гиперссылок оглавления, броузер откроет новое окно, назовет его «view_window» и отобразит в нем содержимое выбранного документа. Если пользователь выберет новую гиперссылку в оглавлении, а окно «view_window» еще открыто, броузер опять отобразит выбранный документ в этом окне, заместив предыдущий документ.

На протяжении всего этого процесса окно, содержащее оглавление, остается доступным пользователю. Щелкая на гиперссылках в одном окне, пользователь изменяет содержимое другого окна.

Чаще атрибут target используется не для того, чтобы открывать совершенно новое окно броузера, а для того, чтобы направлять содержимое документа, на который указывает гиперссылка, в один из фреймов, заданных тегом <frameset>, или в окно <iframe>. Например, можно разместить оглавление в одном фрейме двухфреймового документа, а соседний фрейм использовать для отображения выбранных документов:

```
<frameset cols="150,*">
  <frame src="toc.html">
  <frame src="pref.html" name="view_frame">
</frameset>
```

Когда броузер первый раз отображает оба эти фрейма, левый фрейм содержит оглавление, а правый – предисловие (см. рис. 11.6).

Когда пользователь выбирает гиперссылку из оглавления в левом фрейме (например Chapter 1), броузер загружает и отображает ассоциированный с гиперссылкой документ в правый фрейм, названный «view_window» (рис. 11.7). Когда выбираются другие гиперссылки,

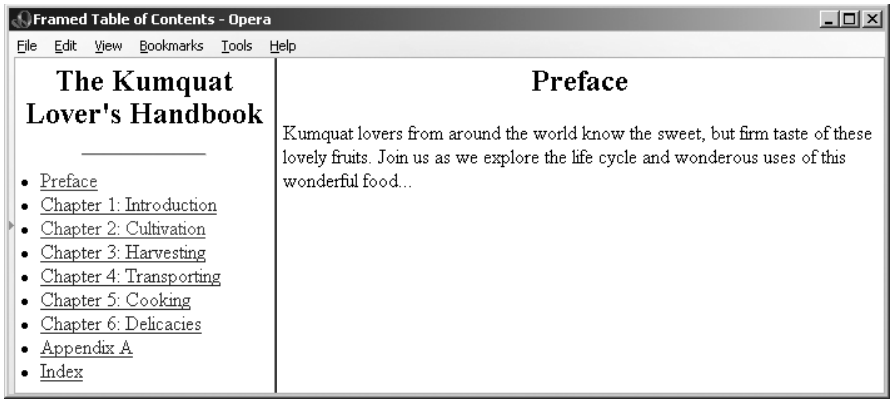

***Рис. 11.6.** Фрейм с оглавлением определяет содержимое соседнего фрейма*

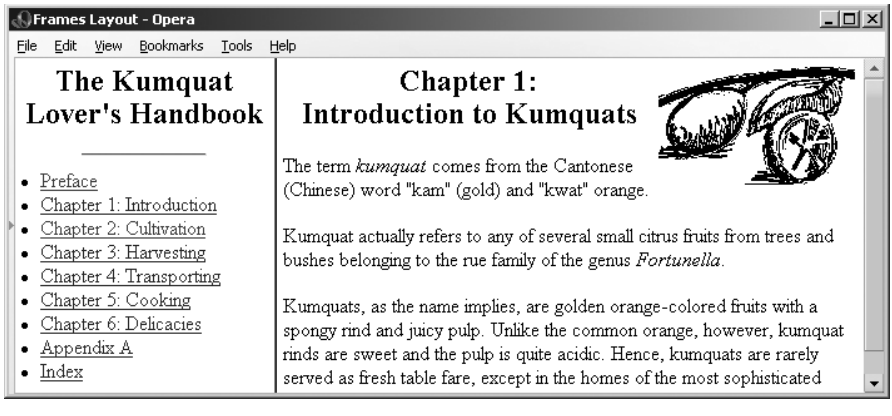

***Рис. 11.7.** Содержимое первой главы выводится в соседнем фрейме*

содержимое правого фрейма меняется, в то время как оглавление в левом фрейме остается постоянно доступным для пользователя.

11.7.2. Специальные значения атрибута target

Существуют четыре зарезервированных значения атрибута `target` для специальных видов перенаправления документов:

`_blank`

Броузер всегда загружает документ, на который указывает гиперссылка с `target="_blank"`, во вновь открываемое безымянное окно.

`_self`

Это значение принимается по умолчанию для тегов `<a>`, в которых этого атрибута нет, и означает, что документ, на который указывает гиперссылка, должен быть загружен и отображен в том же окне или фрейме, что и документ, содержащий гиперссылку.

_parent

Присваивание target="_parent" означает, что документ должен быть загружен в родительское окно или должен заместить набор фреймов, в котором содержится фрейм с документом, содержащим гиперссылку. Если ссылка содержится в окне или фрейме верхнего уровня, значение эквивалентно значению _self.

Короткий пример может помочь прояснить, как работают эти ссылки. Рассмотрим гиперссылку во фрейме, который является частью фреймсета, состоящего из трех столбцов. Этот фреймсет, в свою очередь, включен как содержимое фреймсета верхнего уровня, занимающего все окно броузера. Схема такого каскада фреймов показана на рис. 11.8.

Если для гиперссылки цель не указана, тогда гиперссылка загружается в содержащий ее фрейм. Если в качестве цели указано _parent, документ загружается в область, занятую фреймсетом, состоящим из трех столбцов, в которые входит фрейм, содержащий гиперссылку.

_top

Такая цель представляет собой все окно, в котором находится гиперссылка. Вызванный документ заместит собой все ранее отображавшиеся в этом окне фреймы.

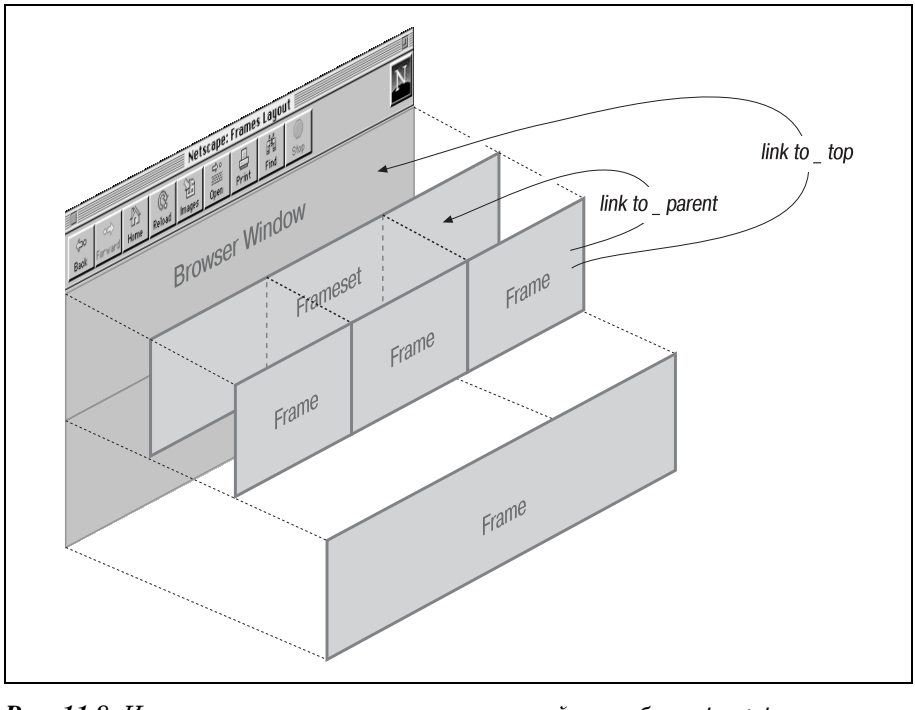

Рис. 11.8. *Использование специальных значений атрибута target с вложенными фреймами и фреймсетами*

В терминах иерархии фреймов, изображенной на рис. 11.8, присваивание target="_top" повлечет за собой удаление всех содержащихся в окне броузера фреймов и загрузку в это окно нового документа.

Все эти четыре значения атрибута target начинаются с символа подчеркивания, всякое другое имя окна или фрейма, начинающееся с подчеркивания, игнорируется броузером, поэтому не используйте подчеркивание в качестве первого символа определяемых в ваших документах имен и идентификаторов.

11.7.3. Определение принимаемой по умолчанию цели в теге <base>

Определение целей для всех гиперссылок в документах – скучное занятие, особенно если большинство из них направляют документ в одно и то же окно или фрейм. Можно облегчить свою жизнь, вставив атрибут target в тег <base>. [элемент заголовка <base>, 6.7.1]

Атрибут target в теге <base> устанавливает принимаемую по умолчанию цель для всех гиперссылок в документе, не содержащих атрибут target. Например, в содержащем оглавление документе из нашего примера все гиперссылки указывают, что документ должен быть отображен во фрейме «view_frame». Вместо того чтобы включать в каждую гиперссылку атрибут target, следует указать общую для всех гиперссылок цель в теге <base>, содержащуюся в заголовке документа:

```
<head>
<title>Table of Contents</title>
<base target="view_frame">
</head>
<body>
<h3>Table of Contents</h3>
<ul>
  <li><a href="pref.html">Preface</a></li>
  <li><a href="chap1.html">Chapter 1</a></li>
  <li><a href="chap2.html" >Chapter 2</a></li>
  <li><a href="chap3.html">Chapter 3</a></li>
</ul>
</body>
```

Заметьте, что мы не включали никаких других указаний на цель в список гиперссылок, поскольку броузер загрузит и отобразит все указанные в гиперссылках документы во фрейм «view_frame», как это предписано спецификацией атрибута target в теге <base>.

11.7.4. Традиционное поведение ссылок

До появления фреймов всякий раз при выборе гиперссылки соответствующий документ замещал содержимое окна броузера. С появлением фреймов этот образ действий претерпел изменения, так что теперь соответствующий документ замещает содержимое фрейма, в котором на-

ходится ссылка. Такое поведение часто оказывается нежелательным и может запутать людей, изучающих собрание ваших документов.

Допустим, что все документы на сайте должны быть представлены в трех фреймах: навигационном в верхней части окна броузера, прокручиваемом фрейме для просмотра документов – посередине, и формы для обратной связи – в нижней части окна. Имя фрейма для просмотра документов задано атрибутом `name` тега `<frame>` документа верхнего для вашей коллекции документов уровня и используете атрибут `target` в теге `<base>` во всех документах вашего сайта, чтобы гарантировать, что все ссылки будут загружаться в средний фрейм.

Эти установки прекрасно работают для всех документов на вашем сайте, но что будет, когда пользователь выберет ссылку, которая перенесет его на другой сайт? Вызванный документ по-прежнему загрузится в средний фрейм. Теперь пользователь оказывается перед документом с другого сайта, находящимся в окружении ваших навигационного фрейма и фрейма обратной связи![1] Очень неловко.

Решение проблемы заключается в том, чтобы все гиперссылки, указывающие на удаленный документ, имели мишенью `_top`. Тогда при выборе кем-либо ссылки, уводящей с вашего сайта, удаленный документ заместит содержимое всего окна броузера, включая ваши фреймы с навигационными документами и средствами обратной связи. Если большая часть ссылок в ваших документах ведет к другим сайтам, можно подумать о возможности включения в тег `<base>` ваших документов спецификации `target="_top"` и явном использовании атрибута `target` в ссылках на ваши локальные документы.

11.8. Модель XFrames

История фреймов – типичная история Золушки. Пройдя путь от нестандартного расширения в броузере Netscape до стандартной конструкции языков HTML и XHTML, фреймы продемонстрировали свою важность в мире HTML-документов. Тем не менее с фреймами связано несколько проблем, до сих пор не получивших полного разрешения:

- навигация с помощью кнопки броузера Назад может вести себя непредсказуемо;
- вы не можете напрямую сослаться на документ в наборе фреймов
- нельзя сослаться на конкретный набор фреймов с помощью одного URL-адреса;
- поисковые механизмы часто пропускают содержимое фреймов.

Чтобы исправить эти недостатки, сохранив при этом всю мощь фреймов, консорциум W3C предложил чуть измененную модель помеще-

[1] Загляните в главу 17, чтобы узнать, как развернуть на все окно броузера ваши страницы, если они случайно будут загружены в чужой фрейм.

ния содержимого во фреймы. Она все еще находится в состоянии разработки и не реализована ни в одном броузере. Тем не менее мы вкратце опишем ее, чтобы авторы HTML-документов знали, чего им следует ожидать от фреймов в ближайшем будущем.

11.8.1. XFrames–документ

В языках HTML и XHTML фреймы замещают тег <body> документа, не затрагивая теги <html> и <head>. В модели XFrames соответствующий документ заменяет собой весь <html>-документ, неся в себе собственный заголовок <head> и содержимое во фреймах. В теге <head> авторы могут указать теги <title> и <style>, а содержимое фреймов определяется с помощью тегов <group> и <frame>. Небольшой XFrames-документ может выглядеть примерно так:

```
<frames xmlns="http://www.w3.org/2002/06/xframes/">
  <head>
    <title>Kumquat Lore</title>
    <style type="text/css">
      #header {height: 10em }
      #toc, #nav {width: 20%}
      #footer {height: 4em }
    </style>
  </head>
  <group compose="vertical">
    <frame xml:id="header" source="lore.xhtml"/>
    <group compose="horizontal">
      <frame xml:id="toc" source="toc.xhtml"/>
      <frame xml:id="main" source="intro.xhtml"/>
      <frame xml:id="nav" source="main-nav.xhtml"/>
    </group>
    <frame xml:id="footer" source="copyright.xhtml"/>
  </group>
</frames>
```

Тег <head> устанавливает заголовок документа и определяет стили, которые будут влиять на вывод соответственно названных фреймов из набора, образующего документ. Тег <group> аналогичен тегу <frameset> и определяет группу фреймов и другие группы, компоновкой которых управляет атрибут compose. Тег <frame> определяет отдельный документ, содержимое которого выводится во фрейме. В нашем примере пять фреймов организованы в три ряда, причем верхний и нижний ряды содержат по одному фрейму, а в среднем находится три фрейма. В этом ряду левый и правый фреймы занимают по 20% доступного пространства, а центральный – остальную часть. Отдельные фреймы именуются с помощью атрибута xml:id, и ссылки на эти имена используются при загрузке нового содержимого во фрейм, при ассоциировании стилей с фреймом и при создании URL-адреса с целью вывода конкретного набора фреймов на экран (к этой теме мы еще вернемся).

Атрибут `compose` в теге `<group>` предоставляет вам дополнительные возможности компоновки, отсутствующие у традиционных фреймов. В то время как эффект от значений `horizontal` и `vertical` вполне очевиден, значения `single` и `free` гораздо интереснее. Установка атрибута `compose` в значение `single` заставляет броузер выводить только один фрейм из группы за раз, причем предполагается присутствие некой индикации наличия других фреймов и способа добраться до них. Например, это может быть раскрывающееся меню, позволяющее выбрать фрейм.

Значение `free` атрибута `compose` выводит группу фреймов в виде набора окон, свободно перемещаемых в области вывода. Пользователь может передвигать окна, как ему заблагорассудится, и даже перекрывать одно окно другим! Предполагается, что это позволит создать аналог рабочего стола в окне броузера и предоставить фреймы пользователю в полное распоряжение.

11.8.2. URL-адреса в модели XFrames

Чтобы поддержать явные ссылки на фреймы, входящие в состав документа, модель XFrames расширяет определение URL, добавляя в него ключевое слово `#frames`. Эта функциональная возможность позволит вам определять содержимое отдельно для каждого фрейма в документе, что совершенно невозможно во фреймовой модели, принятой сейчас в языках HTML и XHTML.

Чтобы воспользоваться этой функциональностью, добавьте ключевое слово `#frames` в конец URL-адреса, ссылающегося на документ с фреймами. После ключевого слова укажите в скобках список идентификаторов фреймов и их содержимое. Думаете, это трудно? Ничуть:

```
http://www.kumquat.com/lore.html#frames(toc=section7.xhtml,main=arctic-
quats.xhtml)
```

Этот URL заставляет броузер открыть документ с фреймами, названный lore.html, и загрузить во фреймы `toc` и `main` указанные страницы. В другие фреймы, упомянутые в документе, загружается содержимое, установленное по умолчанию, поскольку про них в этом URL ничего не сказано. Фреймы, не имеющие содержимого по умолчанию, остаются пустыми.

Этот мощный синтаксис сулит огромные возможности авторам документов и конечным пользователям. Авторы смогут конструировать ссылки, которые позволят открыть комплект документов с фреймами легко воспроизводимым способом. Пользователи смогут делать закладки на документы с фреймами и не сомневаться, что при следующем открытии документа они увидят фреймы с тем же содержимым, которое определил URL.

- Апплеты и объекты
- Вложенное содержимое
- JavaScript
- Таблицы стилей JavaScript (устарели)

12

Исполняемое содержимое

Одним из самых полезных достижений в веб-технологиях является возможность доставки с сервера исполняемых модулей и приложений прямо на броузер пользователя. Эти, обычно маленькие, программы выполняют на компьютере клиента простые задачи – от реагирования на действия пользователя с мышью или клавиатурой до поддержки мультимедийных данных, оживляющих ваши веб-страницы.

Можно добавлять в ваши документы сценарии, используя язык программирования, который называется JavaScript[1]. Также можно загружать из Интернета использующие Java-технологию независимые от платформы приложения, называемые *апплетами*, и исполнять их на своем компьютере. В процессе исполнения эти программы могут генерировать динамическое содержимое, взаимодействовать с пользователем, проверять ввод данных в формы или даже создавать окна и запускать приложения, независимые от ваших страниц. Возможностям несть числа, и они могут идти гораздо дальше, чем это предусмотрено изначально в простой модели HTML-документа.

В этой главе на простых примерах мы вам покажем, как вкладывать и включать исполняемое содержимое – сценарии и апплеты – в ваши документы. Мы не станем, однако, и пытаться научить вас тому, как следует писать и отлаживать выполняемое содержимое. В конце концов, эта книга об HTML и XHTML. Лучше обратитесь к специалистам –

[1] Кроме JavaScript существует еще несколько языков сценариев, поддерживаемых популярными броузерами. Прежде всего это аналог JavaScript от Microsoft под названием JScript, который во многом совпадает с «оригиналом», но имеются и различия (в Internet Explorer все сценарии JavaScript исполняются как JScript, что вносит дополнительную путаницу и ошибки), и язык VBScript, который, являясь ограниченной версией Visual Basic, поддерживается только Internet Explorer. – *Примеч. науч. ред.*

возьмите одну из книг Дэвида Флэнагана (David Flanagan) от O'Reilly: «JavaScript: Definitive Guide» [1], «Java in a Nutshell»[2].

12.1. Апплеты и объекты

Апплеты, как и клиентские карты, представляют собой некоторое изменение базовой модели веб-коммуникаций. До сих пор почти всю вычислительную работу в Сети выполняли серверы, а броузеры клиентов служили всего лишь в качестве усовершенствованных терминалов. С появлением апплетов веб-технологии смещаются в сторону клиента, перенося часть или даже всю вычислительную работу на плечи клиента и его броузера.

Апплеты предоставляют также способ расширения возможностей броузера, не заставляющий пользователя приобретать новый броузер, когда разработчики вводят новый тег или расширяют множество атрибутов HTML. Пользователям нет нужды также доставать и инсталлировать новые программы – *плагины* (*plugin*) и *вспомогательные приложения* (*helper*).[3] Это означает, что если у пользователя есть броузер, поддерживающий апплеты (все популярные броузеры их поддерживают), можно немедленно получить апплет, осуществляющий, например, графические или мультимедийные нововведения.

12.1.1. Модель объектов

Java-апплеты – программы, на которые ссылаются веб-страницы, загружаемые с сетевого сервера и исполняемые на пользовательском компьютере-клиенте, – в действительности составляют подмножество объектов, которые стандарты HTML 4 и XHTML называют *включениями* (*inclusion*). Как и в случае изображений, броузер сначала загружает HTML-документ, затем изучает его на предмет включений – добавочного, отдельного и самостоятельного содержимого, которое должен обработать броузер. GIF-изображения – это один из типов включений, звуковые *.wav*-файлы – второй тип, MPEG-видео – третий, а программа «часы», написанная на Java, принадлежит к еще одному типу включений.

Обычно стандарты HTML 4 и XHTML называют содержимое включений *объектами* (*object*). В документе можно сослаться практически на любой файл в сети и загрузить его при помощи универсального тега `<object>`, который мы подробно обсуждаем далее в этой главе. [тег <object>, 12.2.1]

---

[1] Д. Флэнаган «JavaScript. Подробное руководство», 4-е издание, СПб.: Символ-Плюс, 2004

[2] Д. Флэнаган «Java. Справочник», 4-е издание, СПб.: Символ-Плюс, 2004

[3] На самом деле пользователям броузера Internet Explorer 6 приходится загружать и устанавливать программное обеспечение для поддержки Java. Подробности см. далее.

Стандарты требуют, чтобы загруженный файл был как-то отображен при помощи внутренних или внешних по отношению к броузеру механизмов. В случае их отсутствия необходимый механизм отображения могут предоставлять плагины и вспомогательные приложения. Например, Internet Explorer имеет собственный механизм воспроизведения AVI-роликов, а другие броузеры при показе роликов полагаются на программы от третьих фирм, такие как RealPlayer или QuickTime.

12.1.1.1. Модель апплетов

Для Java-апплетов броузер создает внутри окна отображения документа отдельную область. Можно управлять размером и положением этой области, апплет заведует тем, что представлено внутри нее.

Апплет – это исполняемая программа. Соответственно броузер не только предоставляет область отображения, но и с помощью операционной системы и ресурсов компьютера-клиента помещает апплет в операционную среду – обычно это виртуальная машина Java[1] (Java Virtual Machine, JVM).

В процессе исполнения апплет получает ограниченный доступ к ресурсам на пользовательском компьютере. Апплеты, например, имеют доступ к мыши и клавиатуре и могут получать данные от пользователя. Если позволяет принятая на компьютере политика безопасности, они могут инициировать сетевые соединения и получать данные с других серверов в Интернете. Короче говоря, апплеты – это полноценные программы, укомплектованные разнообразными механизмами ввода-вывода так же, как и полным набором сетевых сервисов.

В одном документе может быть несколько апплетов. Все они исполняются параллельно и могут общаться друг с другом. Хотя броузер может ограничивать доступ апплетов к операционной системе компьютера, на котором он работает, апплеты имеют полный контроль[2] над своим виртуальным окружением в броузере.

12.1.1.2. Преимущества апплетов

Имеется несколько преимуществ использования апплетов, не последнее из которых – возможность создания более проработанных пользовательских интерфейсов внутри веб-страницы. Апплет может, например, создавать свой набор меню, текстовых полей и других средств пользовательского ввода, отличных от тех, что предоставляет броузер.

---

[1] Такая виртуальная машина служит «прослойкой» между Java-программой и операционной системой, переводя команды с внутреннего языка Java (так называемого байт-кода) в команды, понятные конкретной операционной системе. – *Примеч. науч. ред.*

[2] Поскольку все взаимодействие с системой происходит через виртуальную машину, которая строго следит за тем, чтобы апплет не мог совершить «противоправных» действий, одним из плюсов апплетов является безопасность их использования. – *Примеч. науч. ред.*

Когда пользователь нажимает кнопку в интерактивной области отображения апплета, апплет может ответить, выводя результаты в этой области, сигнализируя другому апплету или даже загружая совершенно новую страницу в броузер.

Мы не хотим сказать, что единственное применение апплетов состоит в улучшении пользовательского интерфейса. Апплет – это полноценная программа, способная выполнять на компьютере-клиенте любое число вычислительных задач и задач взаимодействия с пользователем. Апплет способен реализовать отображение видеоинформации в режиме реального времени; использоваться в качестве тренажера, играть с пользователем, предоставлять интерфейс для «разговоров» (чатов) в Интернете и т. д.

12.1.1.3. Корректное использование апплетов

Апплет – это не что иное, как еще одно средство, которое можно использовать для создания завершенных и полезных веб-страниц. Имейте в виду, что апплеты используют вычислительные ресурсы клиента в ходе своей работы и таким образом создают дополнительную нагрузку на компьютер пользователя. Это может замедлить работу системы в целом. Кроме того, если апплет для исполнения своих задач занимает большую часть пропускной способности сетевого соединения (для передачи видео в реальном времени, например), это может сделать весь прочий сетевой обмен невыносимо медленным. Хотя такие приложения могут казаться забавными, они вызовут разве что только раздражение вашей аудитории.

При правильном использовании апплетов нагрузки на броузер и сервер уравновешиваются. Для каждой страницы вы должны решить, какие задачи лучше оставить серверу (обработку форм, поиск в базах данных и т. п.), а какие задачи лучше выполнять локально (усовершенствование пользовательского интерфейса, отображение данных в реальном времени, незатейливую анимацию, проверку заполнения форм и т. д.). Распределите обработку соответствующим образом. Помните, что у многих пользователей сетевое соединение и компьютер медленнее, чем у вас, и разрабатывайте ваши апплеты так, чтобы удовлетворить большую часть вашей аудитории.

Корректно используемые апплеты могут существенно улучшить страницы и впечатление, производимое ими на аудиторию. В случае неправильного применения они лишь расходуют ресурсы, отпугивают пользователей и портят страницы.

12.1.1.4. Написание апплетов

Создание апплетов – это задача по программированию, не входящая в обязанности HTML- или XHTML-авторов. За подробностями мы рекомендуем обратиться к одному из множества руководств по программированию апплетов, включая книги, выпущенные издательством O'Reilly.

Разработанный корпорацией Sun Microsystems в Montain View, California, язык Java поддерживает объектно-ориентированный стиль программирования, в котором классы апплетов могут использоваться снова и снова для построения сложных приложений. По своему замыслу апплеты, написанные на этом языке, должны работать с любым броузером, поддерживающим Java. Хотя в большинстве случаев так оно и есть, реальность устроена несколько сложнее. До шестых версий броузеров Netscape и Internet Explorer броузеры имели собственные виртуальные машины Java (JVM), и их реализация, особенно выполненная в корпорации Microsoft, могла быть довольно экстравагантной. Некоторые решения, принятые разработчиками Internet Explorer версии 4 и более ранних версий, привели к тому, что корректные Java-апплеты не могли работать с этим броузером. Корпорация Microsoft устранила эти проблемы в версии Internet Explorer 5, но из-за судебной тяжбы с компанией Sun не стала включать JVM в шестую версию Internet Explorer.[1] Хотя может показаться, что для апплетов настали плохие времена, на самом деле это не так. Броузер Internet Explorer 6 предлагает пользователю загрузить JVM от Microsoft. Что касается плагина Java Plug-in от компании Sun, то он распространяется в Интернете бесплатно. Пользователи любого броузера могут установить Java Plug-in и получить самую современную поддержку языка Java.

Мы воспользуемся представившейся возможностью, чтобы упомянуть также ActiveX, альтернативную технологию программирования апплетов[2], поставляемую Microsoft. ActiveX – это коммерческий продукт, тесно связанный с различными версиями Microsoft Windows и работающий только с Internet Explorer, хотя альтернативные плагины существуют в настоящее время для всех броузеров.

Элементы управления ActiveX (таково их официальное название) будут работать с версиями Internet Explorer, предназначенными для разных версий Windows, но некоторые элементы управления ActiveX не будут работать с этими версиями без перекомпиляции. Это их самое важное отличие от апплетов Java; Java-апплет может быть однажды написан и скомпилирован и с этого момента будет готов работать с широким множеством броузеров и операционных систем.[3]

---

[1] После того как мы написали эти строки, ситуация могла измениться. Не исключено, что корпорация Microsoft передумает и включит JVM в стандартный пакет для Windows XP. Однако пока нет никаких признаков включения JVM в загружаемые экземпляры Internet Explorer 6.

[2] Вообще говоря, таких альтернативных технологий программирования апплетов существует больше двух, и их количество, судя по всему, будет увеличиваться. Так, например, в последнее время успешно развивается технология, использующая язык Python. – *Примеч. науч. ред.*

[3] При условии, что у пользователя установлена та или более поздняя версия Java SDK (Java Software Developer's Kit), под которую был написан апплет (версии Java SDK не совместимы друг с другом). – *Примеч. науч. ред.*

ActiveX-апплеты представляют также неприемлемо высокий риск нарушения безопасности для всякого пользователя, броузер которого поддерживает ActiveX-технологию.[1] Хотя с годами броузеры стали больше заботиться о безопасности, до смешного легко проникнуть в компьютер, на котором работает броузер, допускающий выполнение ActiveX-апплетов[2], и нанести ему ущерб. Поэтому все популярныые броузеры, включая Internet Explorer, позволяют пользователям заблокировать выполнение апплетов ActiveX. По этой причине мы не можем считать ActiveX жизнеспособной технологией реализации апплетов и даже позволим себе рекомендовать пользователям отключить поддержку ActiveX на их броузерах, особенно на Internet Explorer.[3]

12.2. Вложенное содержимое

В этом разделе мы опишем три тега, поддерживающих вложенное содержимое. Тег `<object>` входит в стандарты HTML 4 и XHTML. Это универсализированный гибрид нежелательного тега `<applet>` для вложенных апплетов, в особенности Java-апплетов, и тега расширения `<embed>`, который позволяет вам включать объекты, MIME-тип которых указывает на плагины, необходимые для обработки и, возможно, отображения этих объектов.

Новые стандарты настоятельно рекомендуют использование тега `<object>` для включения апплетов так же, как и множества разных самостоятельных включений в ваших документах, даже изображений (хотя стандарты не заходят так далеко, чтобы признать нежелательным тег `<img>`). Используйте тег `<object>` с атрибутом `classid` для включения Java и других апплетов в документ в сопровождении параметров их исполнения в виде содержимого ассоциированного тега `<param>`. Используйте атрибут `data` в теге `<object>` для загрузки и отображения не HTML/XHTML-содержимого, такого как мультимедийные объекты, в пользовательском окружении. Данные объекта могут обрабатываться и отображаться вложенными апплетами, утилитами, поставляемыми вместе с броузером, или плагином (вспомогательным приложением), предоставляемым пользователем.

Для апплетов броузер создает область отображения в потоке содержимого точно так, как для встроенных изображений или для тега `<iframe>` –

---

[1] Чтобы узнать, чем вы рискуете, посетите сайт *http://www.digicrime.com/activex*.

[2] Это верно. Перед загрузкой и запуском ActiveX-приложения броузер проверяет наличие и правильность цифрового сертификата приложения, затем предоставляет пользователю самому принять решение о допустимости запуска данного приложения. Но в самом деле, кто читает предупреждения, выдаваемые броузером? – *Примеч. науч. ред.*

[3] В настоящее время технология ActiveX применительно к HTTP-апплетам отходит в прошлое. – *Примеч. науч. ред.*

без разрывов строк и в виде одного большого прямоугольника. Затем броузер загружает и исполняет программный код апплета, если он указан, и сразу после загрузки и отображения всего документа загружает и отображает вложенные данные. Исполнение апплета продолжается до тех пор, пока программа не закончится или пользователь не уйдет со страницы, содержащей апплет.

Встречая вложенные данные, броузер расшифровывает тип данных объекта и либо сам отображает эти данные (изображения GIF, PNG и JPEG, например), либо вызывает для работы соответствующее вспомогательное приложение.

12.2.1. Тег <object>

Тег `<object>` был первоначально введен Microsoft для поддержки своих ActiveX-апплетов и только потом стал поддерживать Java. Подобным образом, Netscape вначале ввела альтернативные теги `<embed>` и `<applet>` для включения объектов, а затем предоставила ограниченную поддержку тегу `<object>`.

<object>

Функция:	Вкладывает в документ объект или апплет
Атрибуты:	`align`, `archive`, `border`, `class`, `classid`, `codebase`, `codetype`, `data`, `declare`, `dir`, `height`, `hspace`, `id`, `lang`, `name`, `notab`  , `onClick`, `onDblClick`, `onKeyDown`, `onKeyPress`, `onKeyUp`, `onLoad`, `onMouseDown`, `onMouseMove`, `onMouseOut`, `onMouseOver`, `onMouseUp`, `shapes`, `standby`, `style`, `tabindex`, `title`, `type`, `usemap`, `vspace`, `width`
Завершающий тег:	`</object>`; присутствует обязательно
Содержит:	*object_content* (содержимое объекта)
Может содержаться в:	*тексте*

Из-за склок между гигантами рынка броузеров[1] мы даже колебались, следует ли вообще рекомендовать использование тега `<object>` в предыдущих изданиях этой книги. Теперь мы опираемся на стандарт HTML 4 и особенно XHTML, и кроме того, современные популярные броузеры поддерживают этот тег. Но надо понимать, что броузеры поразному интерпретируют теги `<object>` и `<embed>`. Так, Internet Explorer все еще считает содержимое тега `<object>` элементами управления ActiveX и запускает вспомогательную программу для вывода данных. Зато содержимое тега `<embed>` он выводит в составе документа.

Содержимым тега `<object>` может быть любое допустимое HTML- или XHTML-содержимое, а также тег `<param>`, передающий апплету пара-

[1] Хотите верьте, хотите – нет, но когда-то Netscape был вне конкуренции.

метры. Если броузеру удается получить запрашиваемый объект и успешно обработать его, запустив апплет или обработав данные объекта при помощи плагина (вспомогательного приложения), содержимое тега <object> игнорируется, если не считать тега <param>. Если возникают проблемы в процессе получения и обработки объекта, броузер не станет вставлять объект в документ, а отобразит вместо этого содержимое тега <object>, за исключением тега <param>. Короче говоря, следует обеспечить альтернативное содержимое для броузеров, которые не умеют обращаться с тегом <object>, и на случай проблем с загрузкой объекта.

12.2.1.1. Атрибут classid

Атрибут classid используется для указания расположения объекта, обычно Java-класса, который должен быть получен броузером. Значением атрибута может быть абсолютный или относительный URL соответствующего объекта. Относительные URL отсчитываются от URL, указанного в атрибуте codebase. Если эта спецификация отсутствует, они отсчитываются от URL текущего документа.

Так, чтобы запустить апплет clock, содержащийся в файле под названием *clock.class*, можно поместить в HTML-документ следующий код:

```
<object classid="java:clock.class">
</object>
```

Броузер найдет код апплета, использовав URL текущего документа в качестве базового. Следовательно, если URL текущего документа это:

```
http://www.kumquat.com/harvest_time.html
```

Броузер будет искать код апплета нашего класса clock по адресу:

```
http://www.kumquat.com/clock.class
```

12.2.1.2. Атрибут codebase

Используйте атрибут codebase для предоставления альтернативного базового URL, от которого броузер будет отсчитывать адреса объектов. Значение этого атрибута – это URL, указывающий на директорию, содержащую объект, на который указывает атрибут classid. URL, указанный в качестве значения codebase, имеет приоритет перед используемым по умолчанию базовым URL документа, но не замещает его полностью. [ссылки на документы: URL, 6.2]

Теперь, продолжая иметь дело с нашим предыдущим примером, предположим, что ваш документ пришел с сервера *http://www.kumquat.com*, но апплет clock хранится в отдельной директории под названием classes. Вы не сможете получить апплет, написав classid="classes/clock.class". Вместо этого включите в тег атрибут codebase и новый базовый URL:

```
<object classid="clock.class" codebase="http://www.kumquat.com/classes/">
</object>
```

который разрешится в URL:

```
http://www.kumquat.com/classes/clock.class
```

Хотя в этом примере мы использовали абсолютный URL, вы можете использовать и относительный URL. Обычно апплеты хранятся на том же сервере, что и содержащий их документ, поэтому будет лучше (из соображений переносимости) указать в качестве базового относительный URL:

```
<object code="clock.class" codebase="/classes/">
</object>
```

Атрибут `classid` подобен атрибуту `code` тега `<applet>`, указывающему имя файла, содержащего апплет, и используется в сочетании с атрибутом `codebase` для определения полного URL объекта, который нужно получить и поместить в документ.

12.2.1.3. Атрибут archive

По соображениям общей эффективности работы можно решить заранее загрузить набор объектов, содержащийся в одном или нескольких архивах. Это особенно верно для Java-приложений, где один Java-класс может при работе опираться на множество других классов. Значением атрибута `archive` служит заключенный в кавычки список URL, каждый из которых указывает на архив, который должен быть загружен броузером перед отображением или исполнением объекта.

12.2.1.4. Атрибут codetype

Атрибут `codetype` требуется, только если броузер не может определить MIME-тип апплета по атрибуту `classid` или если сервер не доставляет правильный MIME-тип при отправке объекта. Этот атрибут почти идентичен атрибуту `type` (раздел 6.7.2.4), за исключением того, что он используется для определения программного кода, тогда как `type` следует использовать для определения типа файлов данных.

Следующий пример явным образом сообщает броузеру, что код объекта – это Java:

```
<object code="clock.class" codetype="application/java">
</object>
```

12.2.1.5. Атрибут data

Атрибут `data` используется для указания – если такой имеется – файла данных, который должен обрабатываться объектом. Значение атрибута `data` – это URL файла, абсолютный или отсчитываемый относительно базового URL документа или от того, который вы указали в атрибуте `codebase`. Броузер определяет тип данных по типу объекта, который вставляется в документ.

Этот атрибут подобен атрибуту `src` тега `<img>` тем, что он указывает на данные, подлежащие отработке со стороны включаемого объекта. Раз-

ница состоит, разумеется, в том, что атрибут data позволяет включить файл практически любого типа, а не только содержащий изображение. Фактически тег <object> ожидает, хотя и не требует этого, что вы явно назовете обрабатывающее приложение объекта в спецификации атрибута classid или укажете MIME-тип файла посредством атрибута type, чтобы помочь броузеру решить, как обрабатывать и отображать данные.

К примеру, вот изображение, включенное в документ в виде объекта, а не файла, указанного в теге <img>:

```
<object data="pics/kumquat.gif" type="image/gif">
</object>
```

12.2.1.6. Атрибут type

Атрибут type позволяет явно указать MIME-тип данных в файле, объявленном в атрибуте data. (Используйте codetype для обозначения MIME-типа апплета.) Если вы не предоставляете данных или если MIME-тип данных с очевидностью следует из URL или сообщается сервером, этот атрибут можно пропустить. Мы рекомендуем вам все же включать его, чтобы гарантировать, что броузер и включенный объект будут обращаться с вашими данными надлежащим образом.

Примеры MIME-типов вы можете обнаружить, заглянув в настройки броузера. Там вы можете найти список множества типов файлов данных, которые распознает ваш броузер, и приложений – если это не сам броузер, – которые обрабатывают и отображают файлы этого типа.

12.2.1.7. Атрибуты align, class, border, height, hspace, style, vspace и width

Имеются несколько атрибутов, которые позволяют управлять внешним видом области отображения объекта из тега <object>, совершенно так же, как это делают соответствующие атрибуты тега <img>. Атрибуты height и width управляют размерами области отображения. Атрибуты hspace и vspace определяют поля вокруг области отображения. Значения этих атрибутов указываются в пикселах.

Атрибут align определяет то, как броузер должен расположить область отображения объекта по отношению к окружающему тексту.[1] Используйте значения top, texttop, middle, absmiddle, baseline, bottom или absbottom, чтобы выровнять область отображения объекта по отношению к примыкающему тексту, или значения left и right, чтобы позволить тексту обтекать ее.

Размеры области отображения часто должны соответствовать нуждам апплета, так что позаботьтесь о том, чтобы эти размеры были согласо-

[1] Атрибут align объявлен нежелательным в стандартах HTML 4 и XHTML из-за появления стандарта CSS, но он по-прежнему популярен и поддерживается броузерами.

ваны с автором-программистом этого апплета. Иногда апплет может масштабировать свой вывод в соответствии с указанными вами размерами области отображения.

Например, предположим, что апплет clock из нашего примера должен уметь растягиваться или сжиматься, чтобы поместиться почти во всякой области отображения. Тогда можно создать квадратные часы 100×100 пикселов:

```
<object classid="clock.class" height="100" width="100">
</object>
```

Как в случае тега <img>, используйте атрибут border для управления толщиной рамки, окружающей область отображения объекта, входящего в гиперссылку. Нулевое значение (border=0) убирает рамку. [тег <img>, 5.2.6]

Используйте атрибуты class и style для управления стилем отображения содержимого тега и для форматирования его содержимого в соответствии с заранее определенным классом стиля для тега <object>. [встроенные стили: атрибут style, 8.1.1] [стилевые классы, 8.3]

12.2.1.8. Атрибут declare

Атрибут declare позволяет определить объект, но удерживает броузер от его загрузки и обработки. Используемое вместе с атрибутом name, это средство подобно предварительным объявлениям в традиционных языках программирования, которые позволяют вам отложить загрузку объекта до момента его фактического использования в документе.

12.2.1.9. Атрибуты id, name и title

Атрибут id или атрибут name используется для создания уникальной метки объекта. Атрибут title используется для того, чтобы просто назвать тег. Значением каждого из этих атрибутов служит строка символов. Броузер может решить представить название объекта пользователю или как-то еще использовать его при отображении документа. Значения id и name используются для ссылок на объект из других элементов ваших документов, включая гиперссылки и другие объекты.

Для примера допустим, что в документе имеются два апплета, изображающие часы, вместе с двумя апплетами, устанавливающими на них время. Снабдите апплеты часов уникальными метками при помощи атрибутов id или name, а затем передайте эти метки устанавливающим время апплетам, используя тег <param>, обсуждаемый далее в этой главе:

```
<object classid="clock.class" id="clock1">
</object>
<object classid="clock.class" id="clock2">
</object>
<object classid="setter.class">
  <param id="clockToSet" value="clock1">
```

```
</object>
<object classid="setter.class">
  <param id="clockToSet" value="clock2">
</object>
```

Поскольку нам нет необходимости различать устанавливающие время апплеты, мы решили не снабжать идентификаторами их экземпляры.

12.2.1.10. Атрибуты shapes и usemap

Напомним, что, как это подробно обсуждалось при описании гиперссылок в главе 6, можно разделить картинку на геометрические области и ассоциировать с каждой из них гиперссылку, создав то, что мы называем картой. Атрибуты `shapes` и `usemap` для тега `<object>` переносят эту возможность на объекты других типов.

Стандартный атрибут `shapes` информирует броузер, что содержимое тега `<object>` включает в себя последовательность гиперссылок и определений геометрических фигур. Атрибут `usemap` и его обязательное значение, являющееся URL, указывает на карту `<map>`, в которой определяются фигуры и ассоциированные с ними гиперссылки точно так же, как это делается с картами со стороны клиента, обсуждавшимися в разделе 6.5.2.

К примеру, вот карта, описанная нами в главе 6, переписанная на XHTML как объект с «фигурами»:

```
<object data="pics/map.gif" shapes="shapes">
  <a shape="rect" coords="0,0,49,49" href="main.html#link1"></a>
  <a shape="rect" coords="50,0,99,49" href="main.html#link2"></a>
  <a shape="rect" coords="0,50,49,99" href="main.html#link3"></a>
  <a shape="rect" coords="50,50,99,99" href="main.html#link4"></a>
</object>
```

и как более знакомая карта:

```
<object data="pics/map.gif" usemap="#map1">
</object>
...
<map name="map1">
  <area coords="0,0,49,49" href="main.html#link1" />
  <area coords="50,0,99,49" href="main.html#link2" />
  <area coords="0,50,49,99" href="main.html#link3" />
  <area coords="50,50,99,99" href="main.html#link4" />
</map>
```

Можно также воспользоваться всеми атрибутами, ассоциированными с гиперссылками, картами и тегами `<area>` для определения и обустройства области карты. К примеру, мы рекомендуем, чтобы вы включали альтернативные (атрибут `alt`) текстовые описания для каждой чувствительной области вашей карты.

12.2.1.11. Атрибут standby

Атрибут `standby` позволяет отобразить сообщение – значение атрибута, являющееся строкой текста, – на то время, пока броузер будет загружать данные объекта. Если объекты велики или если предполагается, что пользователь может соединяться по медленному каналу связи, добавьте этот атрибут в порядке любезности по отношению к пользователям.

12.2.1.12. Атрибуты tabindex и notab

Для Internet Explorer и только для объектов ActiveX атрибут `notab` исключает объект из табуляционной последовательности документа.

В качестве альтернативы использованию мыши пользователи могут переходить между активными элементами документа при помощи клавиши <Tab>. Затем пользователь может активизировать выбранный элемент, нажав клавишу <Enter> или <Return>, а броузеры, возможно, предоставят другие механизмы выделения содержимого. В типичном случае каждый раз, когда пользователь переходит на следующий объект (например, нажимая на клавишу <Tab>), броузер перешагивает с одного активного элемента на другой в том порядке, в каком они появляются в документе. При помощи атрибута `tabindex` этот порядок можно изменить. Значение этого атрибута – целое число, указывающее позицию этого элемента в общей последовательности переходов для текущего документа.

12.2.1.13. Атрибуты dir и lang

Атрибут `lang` позволяет явным образом указать язык для содержимого тега `<object>`. Атрибут `dir` советует броузеру, в каком направлении следует отображать текст. [атрибут dir, 3.6.1.1] [атрибут lang, 3.6.1.2]

12.2.1.14. Обработка событий, связанных с объектом

При наступлении в области действия объекта событий, инициированных с помощью мыши и/или клавиатуры, вы, возможно, захотите совершить некоторые действия. Тогда вы можете использовать десять стандартных атрибутов событий, чтобы по наступлении этих событий передать управление коду JavaScript. Подробнее мы описываем обработчики событий JavaScript в разделе 12.3.3.

12.2.1.15. Броузеры, не поддерживающие апплеты или тег <object>

Поскольку некоторые броузеры, возможно, не поддерживают апплеты или тег `<object>`, вам, может быть, захочется сообщить пользователям, что они теряют. Заключив соответствующее сообщение (а это может быть любое разрешенное в теле документа содержимое) между тегами `<object>` и `</object>`, вы добьетесь желаемого.

Поддерживающие тег <object> броузеры игнорируют дополнительное содержимое тега. Броузеры, не поддерживающие объекты, разумеется, не распознают тега <object>. Вообще терпимые к очевидным ошибкам, броузеры обычно игнорируют нераспознанные теги и попросту продолжают отображать все, что было внутри тега. Вот и все. Следующий фрагмент сообщает пользователям броузеров, не поддерживающих объектов, что они не увидят часов из нашего примера:

```
<object classid=clock.class>
  Если бы ваш броузер умел работать с апплетами, то вы бы смогли увидеть
  прямо здесь стильные часы!
</object>
```

Что более существенно, поддерживающие объекты броузеры отобразят содержимое тега <object>, если не смогут загрузить, исполнить или вывести объект. Если есть несколько объектов сходного назначения, но с различающейся доступностью, можно включить их теги друг в друга, броузер будет перебирать эти объекты по очереди и остановится на первом, с которым он умеет обращаться. Так, например, самый внешний объект может быть полноценным видео. Внутри его тега <object> можно поместить более простое MPEG-видео, а внутри его тега <object> – простое GIF-изображение. Если броузер поддерживает полноценное видео, то пользователи его увидят. Если видео этого уровня оказывается недоступным, броузер может попробовать загрузить менее ресурсоемкий видеоклип в формате MPEG. Если это не удалось, броузер может просто вывести изображение. Если и это оказалось невозможным, самый внутренний тег <object> может содержать словесное описание объекта.

12.2.2. Тег <param>

Тег <param> предоставляет параметры тегам <object> или <applet>, в которых он содержится. (Мы обсуждаем нежелательный тег <applet> в разделе 12.2.3.)

Тег <param> не имеет содержимого[1], и в HTML у него нет закрывающего тега. Он может находиться, возможно, наряду с другими тегами <param>

<param>

Функция:	Доставляет параметры вложенному объекту
Атрибуты:	id, name, type, value, valuetype
Завершающий тег:	В HTML отсутствует; </param> или <param ... /> в XHTML
Содержит:	Ничего
Может содержаться в:	*applet_content* (содержимое апплета)

[1] Но имеет атрибуты, «несущие» в себе значения параметров, которые ожидает получить апплет. – *Примеч. науч. ред.*

только внутри тегов <object> или <applet>. Используйте тег <param> для передачи параметров вложенному объекту, такому как Java-апплет, если это нужно для его правильного функционирования.

12.2.2.1. Атрибуты id, name и value

У тега <param> два обязательных атрибута: name или id, и value. Вы уже видели такое в формах. Вместе они определяют пару имя/значение, которую броузер передает апплету.

Наш апплет clock из предыдущего примера может предоставлять пользователю возможность указать часовой пояс, в котором он живет. Чтобы передать нашему апплету параметр под названием «timezone» с присвоенным ему значением «EST», определите параметры следующим образом:

```
<object classid="clock.class">
  <param id="timezone" value="EST" />
</object>
```

Броузер передает апплету пару имя/значение, но это не гарантирует, что апплет ожидает эти параметры, что имя и значение – правильные или что апплету вообще нужны параметры. Правильные имена параметров с учетом регистра и допустимые их значения определяются автором апплета. Разумный автор веб-страниц будет работать в тесном контакте с автором апплета или обзаведется подробной документацией, чтобы гарантировать, что параметры апплета названы правильно и им присвоены допустимые значения.

12.2.2.2. Атрибуты type и valuetype

Используйте атрибуты type и valuetype для определения типа параметра, передаваемого броузером вложенному объекту, и того, как объект должен интерпретировать значение этого параметра. Атрибут valuetype принимает одно из трех значений: data, ref и object. Значение data указывает, что значение параметра – это простая строка. Это значение принимается по умолчанию. Если valuetype=ref, то значение параметра – это URL какого-то другого ресурса Сети. Наконец, object указывает, что значение – это имя другого вложенного объекта в данном документе. Это может понадобиться для поддержки коммуникации между объектами документа.

Значение атрибута type – это MIME-тип значения параметра. Оно обычно не имеет никакой важности, когда значение параметра – простая строка, но может быть существенным, когда значение является URL, указывающим на некий другой объект в Сети. В этих случаях вложенному объекту бывает нужно знать MIME-тип объекта, чтобы правильно его использовать. К примеру, следующее описание параметра сообщает вложенному объекту, что параметр на самом деле является документом Microsoft Word:

```
<param id="document" value="http://kumquats.com/quat.doc"
   type="application/msword" valuetype="ref" />
```

12.2.3. Тег <applet> (нежелателен)

Используйте тег <applet> в вашем документе, чтобы загрузить и запустить апплет. Используйте также этот тег для определения области отображения апплета в документе. В тег <applet> можно включить альтернативное содержимое, которое будут отображать броузеры, не поддерживающие апплетов.

<applet>	
Функция:	Вставляет апплет в поток текста
Атрибуты:	align, alt, archive, class, code, codebase, height, hspace, id, mayscript, name, object, style, title, vspace, width
Завершающий тег:	</applet>; присутствует обязательно
Содержит:	*applet_content* (содержимое апплета)
Может содержаться в:	*тексте*

Большинству апплетов нужны один или несколько параметров, управляющих исполнением апплета, которые передаются им из вашего документа. Параметры размещаются между тегом <applet> и закрывающим тегом </applet> в теге <param>. Броузер передаст апплету параметры, определенные в документе, при его запуске. [тег <param>, 12.2.2]

Тег <applet> нежелателен в стандартах HTML 4 и XHTML. Его функцию теперь выполняет универсальный тег <object>, который может все то же, что и <applet>, и еще многое другое. Тем не менее <applet> – это популярный тег, и его по-прежнему поддерживают популярные броузеры.

12.2.3.1. Отображение апплетов

Броузер создает для апплета область отображения в потоке содержимого точно так же, как и для встроенных изображений, – без разрывов строк и в виде одного большого прямоугольника. Затем броузер загружает и исполняет программный код апплета, если он определен, и сразу после загрузки и отображения всего документа загружает и отображает вложенные данные. Исполнение апплета продолжается до тех пор, пока программа не закончится или пользователь не уйдет со страницы, содержащей апплет.

12.2.3.2. Атрибут align

Как и в случае изображения или тега <iframe>, атрибут align позволяет вам управлять областью вывода апплета по отношению к окружающему его тексту, хотя создатели стандартов и предпочли бы, чтобы вы

пользовались свойствами выравнивания, определенными в CSS. Используйте значения top, texttop, middle, absmiddle, baseline, bottom или absbottom, чтобы выровнять область отображения объекта по отношению к примыкающему тексту, или значения left и right, чтобы позволить тексту обтекать область отображения апплета (см. раздел 5.2.6).

12.2.3.3. Атрибут alt

Атрибут alt предоставляет способ вежливо уведомить пользователей, что что-то потерялось, когда по какой-то причине апплет не может или не хочет исполняться на их компьютерах. Значение этого атрибута – это заключенная в кавычки строка сообщения, которая, как и в случае атрибута alt для изображений, отображается вместо самого апплета.

Сообщение в атрибуте alt предназначено только для тех броузеров, которые поддерживают апплеты. Смотрите раздел 12.2.1.15, чтобы узнать, как следует информировать пользователей броузеров, не поддерживающих апплеты, о том, почему они не могут увидеть апплета.

12.2.3.4. Атрибут archive

Атрибут archive собирает часто использующиеся классы Java в одну библиотеку, которая кэшируется на локальном диске пользователя. Когда библиотека кэширована, броузеру нет необходимости использовать сеть, чтобы получить апплет, – он извлекает нужную программу из своих локальных запасов, исключая тем самым задержки, связанные с дополнительным использованием сети для загрузки класса.

Значением атрибута archive является URL, указывающий на архивный файл. Суффиксом имени архивного файла может быть либо *.zip*, либо *.jar*. Архивные файлы *.zip* имеют хорошо известный формат ZIP-архивов. Архивные файлы *.jar* имеют формат Java-архивов. Архивные файлы *.jar* поддерживают сжатие данных и передовые возможности, такие как механизмы электронной подписи.

Атрибут archive можно использовать в теге <applet>, даже если класс, на который указывает атрибут тега code, отсутствует в архиве. Если класс не обнаруживается в архиве, броузер просто пытается получить класс по URL, указанному в URL документа или в URL атрибута codebase, если последний определен.

12.2.3.5. Атрибуты code и codebase

Атрибут code является обязательным для тега <applet>. Используйте атрибут code для определения имени файла (но *не* URL) класса Java, который должен быть запущен броузером. Как и в случае тега <object>, чтобы поиск производился относительно другого места хранения, используйте атрибут codebase, описанный в разделе 12.2.1.2, или атрибут archive, описанный в разделах 12.2.1.3 и 12.2.3.4. Расширение имени файла должно быть *.class*. Если расширение опущено, некоторые броузеры автоматически допишут *.class* при поиске апплета.

Вот знакомый нам апплет с часами часов, но уже с тегом <applet>:

```
<applet code="clock.class" codebase="http://www.kumquat.com/classes/">
</applet>,
```

который броузер получит по адресу:

```
http://www.kumquat.com/classes/clock.class
```

12.2.3.6. Атрибут name

Атрибут name позволяет снабдить уникальным именем экземпляр класса – ту копию апплета, которая исполняется на персональном компьютере пользователя. Как и в случае других именованных элементов документа, ассоциирование имени с апплетом позволяет другим частям документа, включая другие апплеты, ссылаться на него и взаимодействовать с ним, например совместно используя результаты вычислений.

12.2.3.7. Атрибуты height, hspace, vspace и width

Используйте атрибуты height и width (идентичные их аналогам для тегов <img> и <object>) для определения размера области отображения апплетов в документе. Атрибуты hspace и vspace используются для того, чтобы окружить область апплета некоторым свободным пространством и, тем самым, выделить его из текста. Все эти атрибуты принимают значения, обозначающие размеры в пикселах. [атрибуты height и width, 5.2.6.10]

12.2.3.8. Атрибут mayscript

Атрибут mayscript позволяет апплету взаимодействовать с функциями JavaScript. Обычно попытка апплета получить доступ к функциям JavaScript вызывает ошибку броузера. Если апплет использует JavaScript, вы должны включить в тег <applet> атрибут mayscript.

12.2.3.9. Атрибут title

Значением этого атрибута является заключенная в кавычки строка, которая используется для присвоения апплету названия, если это оказывается необходимым.

12.2.3.10. Атрибут object

Этот неудачно названный атрибут и его строковое значение указывают имя ресурса, который содержит класс, заставляющий апплет[1] переинициализироваться. Что и как делает этот атрибут, остается загадкой – ни один из популярных броузеров его не поддерживает.

---

[1] Дело в том, что при переходе на другую страницу броузер не выгружает код апплетов: они «засыпают» и при возврате на страницу продолжают исполняться с того места, где их застал переход. Вызов класса из атрибута object должен был принудительно перезапустить апплет (если бы, конечно, этот атрибут поддерживался каким-либо броузером) либо привести апплет в какое-нибудь новое состояние. – *Примеч. науч. ред.*

12.2.4. Тег <embed> (расширение)

Когда-то тег <embed> был единственным способом включения в документ ссылки, предписывающей броузеру запустить некоторый специальный добавляемый модуль и, возможно, обработать данные для него. Сегодня стандартом является тег <object> с атрибутом data, и мы рекомендуем вам использовать его вместо тега <embed>. Впрочем, <embed> пользуется поддержкой всех популярных броузеров.

<embed>

Функция:	Вкладывает объект в документ
Атрибуты:	align, border, height, hidden, hspace, name, palette, pluginspage, src, type, units, vspace, width
Завершающий тег:	Отсутствует
Содержит:	Ничего
Может содержаться в:	*тексте*

Используя тег <embed>, ссылайтесь на объект данных при помощи атрибута src, значением которого является URL файла, который должен быть загружен броузером. Броузер использует MIME-тип объекта, указанного в src для определения того, какой плагин нужен для обработки объекта. В качестве альтернативы можно использовать атрибут type для указания MIME-типа вне объекта и, таким образом, инициировать исполнение плагина, если он имеется на компьютере пользователя.

Как у всех других тегов, у нестандартного тега <embed>, принадлежащего к расширениям Internet Explorer и Netscape, есть множество свойственных ему атрибутов, определяющих его параметры и модифицирующих его поведение. В отличие от большинства других тегов, однако, броузеры позволяют вам включать в тег <embed> специфичные для плагина пары имя/значение, которые не изменяют действие самого тега, но передаются плагину для дальнейшей обработки.

Вот, например, следующий тег:

```
<embed src=movie.avi width=320 height=200 autostart=true loop=3>
```

имеет атрибуты, обрабатываемые броузером для тега <embed> (src, width, heigth), и два других, которые не распознаются броузером, но вместо этого передаются плагину, ассоциированному с AVI видеоклипами: autostart (автозапуск) и loop (количество повторов).[1]

Невозможно задокументировать всевозможные атрибуты, которые могут понадобиться множеству разных плагинов и должны переда-

[1] Поддержка AVI встроена в IE, а пользователи других броузеров должны скачать и установить специальный плагин.

ваться при помощи тега <embed>. Чтобы узнать обо всех обязательных и необязательных атрибутах всех тех элементов, которые вы намерены использовать на ваших страницах, вам придется обратиться к разработчикам плагинов.

12.2.4.1. Атрибуты align, border, height, width, hspace и vspace

Броузер отводит отдельную область для отображения вложенных объектов внутри окна документа. Атрибуты тега <embed> align, border, height, width, hspace и vspace позволяют управлять внешним видом области отображения объекта точно так же, как они это делают для тега <img>, так что мы не будем говорить о них здесь подробно. [тег <img>, 5.2.6]

Вкратце, атрибуты height и width управляют размером области отображения. Обычно следует указывать эти размеры в пикселах, но можно использовать и другие единицы измерения, если присвоить значение атрибуту units (см. раздел 12.2.4.8). Атрибуты hspace и vspace определяют ширину полей в пикселах вокруг области отображения объекта. Атрибут align определяет, как броузер должен выравнивать область отображения объекта по отношению к окружающему тексту, а атрибут border определяет толщину рамки (если таковая имеется) вокруг области отображения.

Все популярные броузеры поддерживают атрибуты height, width и align, но Internet Explorer не поддерживает атрибуты border, hspace и vspace для тега <embed>, поддерживая их, однако, для тегов <applet> и <object>.

12.2.4.2. Атрибут hidden

Атрибут hidden делает объект невидимым для пользователя, устанавливая его высоту и ширину равными нулю. Обратите внимание на то, что при использовании этого атрибута броузер не отображает в документе пустую область, а полностью удаляет объект из обрамляющего его потока текста.

Этот атрибут полезен[1] при включении в документ аудиовставок. Следующий HTML-код:

```
<embed src=music.wav hidden autostart=true loop=true>
```

вкладывает в страницу аудиообъект. Броузер ничего не показывает пользователю, но вместо этого включает музыкальное сопровождение страницы. Для сравнения плагин, присоединенный посредством

```
<embed src=music.wav>,
```

[1] Имейте в виду, что ничто так не раздражает посетителя страницы, как звук, который раздается внезапно и который нельзя отключить. Включение в документ проигрывающихся автоматически, а тем более неуправляемых аудиофайлов является дурным тоном в создании веб-страниц. – *Примеч. науч. ред.*

может вывести на страницу панель управления, при помощи которой пользователь может включать и выключать воспроизведение записи, регулировать громкость и т. д.

12.2.4.3. Атрибут name

Подобно другим атрибутам `name` этот атрибут позволяет снабдить вложенный объект меткой, используемой в дальнейшем для ссылок на этот объект со стороны других элементов документа, в частности, других объектов. Значением атрибута служит строка символов.

12.2.4.4. Атрибут palette

Атрибут `palette` поддерживают и Netscape, и Internet Explorer, но совершенно по-разному. Для Netscape значением атрибута `palette` могут быть либо `foreground`, либо `background`, указывающие, какую палитру системных цветов окна использует при отображении плагин.

Для Internet Explorer значением этого атрибута служит, напротив, пара шестнадцатеричных кодов цветов, разделенных вертикальной чертой. Первое значение указывает цвет переднего плана, используемого плагином, а второе – цвет фона. Таким образом, следующее определение атрибута `palette`:

```
palette=#ff0000|#00ff00
```

заставит плагин использовать красный цвет для переднего плана и зеленый – для фона. Полное описание шестнадцатеричных кодов цветов вы найдете в приложении G.

12.2.4.5. Атрибут pluginspage

Атрибут `pluginspage`, в прежние времена поддерживаемый только Netscape, указывает URL веб-страницы, содержащий информацию о том, где можно получить и каким образом установить плагин, ассоциированный с вложенным объектом. В настоящее время все популярные броузеры предложат вам загрузить поддерживающие плагины с соответствующих домашних страниц.

12.2.4.6. Атрибут src

Подобно мириадам его аналогов, ссылающихся на документы в других тегах, атрибут `src` представляет URL объекта данных, который вкладывается в HTML-документ. Сервер, поставляющий этот объект, должен быть сконфигурирован так, чтобы сообщать броузеру правильный MIME-тип объекта. Если этого не происходит, броузер использует суффикс (расширение имени файла в пути URL) последнего элемента URL для определения типа объекта. По MIME-типу броузер определяет, какой плагин надо запустить для обработки объекта.

Если не включить в тег <embed> атрибут src, то нужно включить атрибут type, чтобы явно указать MIME-тип объекта и, таким образом, необходимый для него плагин.

12.2.4.7. Атрибут type

Атрибут type используется в качестве дополнения к атрибуту src или вместо него. Значение этого атрибута явным образом указывает MIME-тип вложенного объекта, который, в свою очередь, определяет, какой плагин будет вызван броузером для обработки объекта. Этот атрибут не является обязательным, если включен атрибут src, и броузер может определить тип объекта на основании URL объекта или сообщения сервера. Если атрибут src не задан, то атрибут type должен быть указан обязательно.

Может показаться странным использование тега <embed> без атрибута src, ссылающегося на какой-либо объект, но это обычное дело, когда плагину не нужны данные или он их получает динамически после запуска. В этих случаях атрибут type необходим, чтобы броузер знал, какой плагин следует запустить.

12.2.4.8. Атрибут units

По умолчанию в качестве единиц измерения для атрибутов height и width, регулирующих размеры области отображения тега <embed>, используются пикселы.[1] Атрибут units позволяет явным образом указать, что единицами размеров являются пикселы (pixels) или относительные единицы en, равные половине текущего значения высоты шрифта в пунктах. Используя единицы en, вы выкраиваете область отображения объекта пропорционально его непосредственному окружению, размеры которого регулируются пользователем.

Например:

```
<embed src=movie.avi height=200 width=320 units=pixels>
```

создает область отображения объекта размером 200×320 пикселов. Заменив значение атрибута units на en, в окружении текста размером в 12 пунктов вы выделяете для отображения объекта область размером 1200×1920 пикселов.

---

[1] Все написанное в этом разделе верно только для Netscape. Internet Explorer по умолчанию также использует пикселы. Однако обозначаются они иначе: px. В качестве относительных единиц измерения Internet Explorer использует единицы em и ex. Первая равна высоте текущего шрифта, а вторая – высоте латинской буквы «x» в текущем шрифте. Как нетрудно догадаться, размер en из Netscape и размер ex из Internet Explorer примерно одинаково соотносятся с окружающим текстом. – *Примеч. ред.*

12.2.5. Тег <noembed> (расширение)

Тег <noembed>, хотя и не прописан в стандартах, поддерживается популярными броузерами, то есть они игнорируют текст, заключенный в этот тег. С другой стороны, броузеры, не распознающие тег <embed>, игнорируют и <noembed>, выводя заключенный в него текст и, тем самым, предоставляя пользователям альтернативное содержимое, сообщающее им, что они теряют из-за недоступности им содержимого тега <embed>.

<noembed>	
Функция:	Предоставляет содержимое для отображения не поддерживающими тега <embed> броузерами
Атрибуты:	Отсутствуют
Завершающий тег:	</noembed>; всегда присутствует
Содержит:	*ничего*
Может содержаться в:	*тексте*

Содержимое тега <noembed> позволяет информировать пользователя о недостаточности возможностей его броузера:

```
<embed src=cool.mov autostart=true loop=true>
<noembed>Чтобы увидеть крутой ролик, вам необходимо установить броузер,
который поддерживает тег &lt;embed&gt;!</noembed>
```

Мы рекомендуем использовать сообщение в теге <noembed> только в тех случаях, когда вложенный объект имеет решающее значение для понимания и использования вашего документа. При этом следует предоставить ссылку на документ, который может обойтись без вложенного объекта, или вежливо объяснить возникшее затруднение.

12.3. JavaScript

Все элементы исполняемого содержимого, о которых мы до сих пор говорили, имеют одну общую черту – они существуют отдельно от броузера и HTML/XHTML-документа, представляя собой отдельный блок данных или отдельную программу.

Совсем иное дело – JavaScript. Это встроенный язык сценариев, обращающийся к собственным возможностям броузера. Можно раскидать предложения JavaScript по вашим документам – в виде ли блоков программного кода или в виде отдельных операторов, привязанных к индивидуальным тегам. Броузеры, поддерживающие JavaScript, в том числе все все популярные броузеры, интерпретируют и исполняют введенные в текст документов фрагменты кода JavaScript, изменяя внешний вид документов, управляя характеристиками отображения,

манипулируя элементами форм и проверяя правильность их заполнения, выполняя, наконец, вычислительные задачи общего назначения.

Как и в случае Java, мы не претендуем на то, чтобы учить вас в этой книге программированию на JavaScript. Мы покажем вам, как вкладывать JavaScript в ваши документы и как запускать эти программы, а полное определение языка JavaScript вы сможете найти в книге «JavaScript: Definitive Guide» (O'Reilly).

12.3.1. Тег <script>

Один из способов включения JavaScript-кода в документы заключается в использовании стандартного HTML- и XHTML-тега `<script>`.

<script>	
Функция:	Определяет в документе исполняемый сценарий
Атрибуты:	`charset`, `defer`, `language`, `src`, `type`
Завершающий тег:	`</script>`; присутствует обязательно
Содержит:	*сценарии*
Может содержаться в:	*head_content* (содержимое заголовка), *body_content* (содержимое тела)

Все, что содержится между тегами `<script>` и `</script>`, будет обрабатываться броузером как исполняемые выражения (функции) JavaScript и данные. Внутри этого тега нельзя размещать разметку HTML и XHTML – броузер расценит это как ошибку.

Однако броузеры, не поддерживающие тега `<script>`, обрабатывают его содержимое – к недоумению пользователей – как обычный HTML. По этой причине мы рекомендуем заключать содержимое тега `<script>` в HTML-комментарии, как это делается CSS-правилами `<style>`:

```
<script language="JavaScript">
<!--
   Здесь будут выражения JavaScript
// -->
</script>
```

Для броузеров, игнорирующих тег `<script>`, это содержимое будет скрыто обозначениями комментариев `<!--` и `-->`. Поддерживающие JavaScript броузеры, напротив, автоматически распознают и интерпретируют операторы JavaScript, выделенные тегами комментариев. Используя этот шаблон во всех своих тегах `<script>`, можно быть уверенным, что броузеры обработают документы, если даже и не полностью, то аккуратно.

К несчастью, как мы об этом говорим в главе 16, XHTML требует, чтобы содержимое сценария заключалось в специальные объявления CDATA, а не в комментарии. Следовательно, HTML-броузеры проиг-

норируют XHTML-сценарии, и наоборот. Единственное, что мы можем здесь порекомендовать, это ориентироваться на популярные броузеры – писать на HTML, но, готовясь к будущему, включать в ваши документы как можно больше черт XHTML.

Можно включить в документ не один тег `<script>`, а несколько, располагая их как в заголовках (`<head>`), так и в теле (`<body>`) документа. Поддерживающий JavaScript броузер исполняет предложения по очереди. Переменные и функции, определенные в одном теге, могут использоваться в других JavaScript-предложениях других тегов `<script>`. Фактически один распространенный стиль программирования на JavaScript состоит в помещении в документ одного тега `<script>`, содержащего определения глобальных переменных и функций общего назначения, и последующих вызовов этих функций и ссылок на эти переменные в предложениях JavaScript, разбросанных по документу.

12.3.1.1. Атрибуты language и type

Атрибуты `language` и `type` тега `<script>` используются для объявления языка сценария, использованного при составлении содержимого тега. Атрибут `language` признан нежелательным в стандартах HTML 4 и XHTML в пользу атрибута `type`. К сожалению, значения этих атрибутов различны.

Если вы используете JavaScript – до сих пор самый распространенный в Сети язык сценариев, – пишите `language=JavaScript` или `type=text/javascript`. Иногда вы можете увидеть в качестве значения атрибута `language=VBScript` (`text/vbscript` для `type`), что означает, что текст внутри тега написан на Microsoft Visual Basic Script.[1]

В случае JavaScript можно также указать в качестве языка `"JavaScript 1.2"`, тем самым уточняя, что данный скрипт написан для броузеров, поддерживающих версию 1.2 этого языка (а большинство современных броузеров ее действительно поддерживает). Netscape 2.0, поддерживающий JavaScript 1.0, не будет обрабатывать сценарий, объявленный как `"JavaScript 1.1"`, но какова доля ваших клиентов, все еще использующих Netscape 2.0?

12.3.1.2. Атрибуты src и charset

Коды особенно больших или часто используемых JavaScript-программ, возможно, имеет смысл сохранять в отдельных файлах. В таких случаях броузер будет загружать эти отдельные файлы при помощи атрибута `src`. Значение атрибута – это URL файла, содержащего JavaScript-программу. MIME-тип такого файла должен быть `application/x-javascript`, но правильно сконфигурированный сервер будет с ним обращаться надлежащим образом, если суффикс в имени файла будет *.js*.

---

[1] Применение в HTML сценариев на MS Visual Basic Script воспринимается сегодня как явный архаизм. – *Примеч. науч. ред.*

Например:

```
<script type="text/javascript" src="http://www.kumquat.com/quatscript.js">
</script>
```

предлагает поддерживающему тег <script> броузеру загрузить с сервера JavaScript-программу под названием *quatscript.js*. Хотя у тега <script> нет содержимого, закрывающий тег </script> все же обязателен.

Используемый совместно с атрибутом src, атрибут charset сообщает броузеру, какой набор символов использовался при записи программы на JavaScript. Его значением может служить любой набор кодов символов стандарта ISO.

12.3.1.3. Атрибут defer

Некоторые JavaScript-сценарии используются для создания содержимого документа с применением метода document.write, а некоторые – нет. Если ваши сценарии не изменяют содержимого документа, вставьте в тег <script> атрибут defer, чтобы ускорить обработку вашего документа. Поскольку броузер будет знать, что он может без опаски прочитать остаток вашего документа, не исполняя ваши сценарии, он отложит интерпретацию сценариев до окончания отображения документа на устройстве пользователя.[1]

12.3.2. Тег <noscript>

С помощью тега <noscript> можно оставить сообщение для тех пользователей, чьи броузеры не поддерживают тег <script>. Вы уже видели множество примеров применения этого тега, так что знаете, как поступить...

<noscript>

Функция:	Предоставляет содержимое для отображения не поддерживающими тега <script> броузерами
Атрибуты:	class, dir, id, lang, onClick, onDblClick, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, style, title
Завершающий тег:	</noscript>; присутствует обязательно
Содержит:	*body_content* (содержимое тела)
Может содержаться в:	*тексте*

[1] Атрибут defer в точности означает «отложенное исполнение»: исполнение таких скриптов будет производиться только после завершения всего исходного HTML-документа; все остальные трюки с использованием defer уже проистекают как следствие из такого поведения. Атрибут defer введен Microsoft и из всего множества броузеров поддерживается только Internet Explorer. – *Примеч. науч. ред.*

Очень старые, поддерживающие тег <script> броузеры, такие как Netscape 2 и Internet Explorer 3, ничтоже сумняшеся отобразят – к недоумению пользователей – содержимое тега <noscript>. Учитывая немногочисленность владельцев этих броузеров, мы сомневаемся в необходимости всей затеи, но тем не менее существуют другие способы определения того, как броузеры относятся к тегу <script>, и соответствующего с ними обращения – они подробно описаны во всякой хорошей книге по JavaScript.

Тег <noscript> поддерживает шесть стандартных атрибутов HTML 4/XHTML: class и style для управления стилями, lang и dir для выбора типа языка и направления вывода, title и id для озаглавливания и именования содержимого, а также атрибуты событий для обработки действий пользователя. [атрибут dir, 3.6.1.1] [атрибут lang, 3.6.1.2] [атрибут id, 4.1.1.4] [атрибут title, 4.1.1.5] [встроенные стили: атрибут style, 8.1.1] [стилевые классы, 8.3] [JavaScript-обработчики событий, 12.3.3]

12.3.3. JavaScript-обработчики событий

Одной из самых важных возможностей, предоставляемых JavaScript, является способность отмечать события, происходящие во время загрузки документа[1], его вывода и изучения пользователем, и реагировать на них. Код JavaScript, обрабатывающий эти события, может размещаться в теге <script>, но чаще он ассоциируется с определенным тегом при помощи одного или нескольких специальных атрибутов тега.

Например, вы можете захотеть вызывать некую функцию JavaScript, когда пользователь проводит мышью над гиперссылкой в документе. Поддерживающие JavaScript броузеры распознают специальный атрибут обработки события «мышь сверху» для тега <a>, именуемый onMouseOver и позволяющий написать следующее:

```
<a href="doc.html" onMouseOver="status='Click me!'; return true">
```

Когда мышь попадает на ссылку из этого примера, броузер исполняет оператор JavaScript. (Отметьте, что два оператора JavaScript заключены в кавычки и разделяются точкой с запятой и что текстовое сообщение в первом предложении заключено в одинарные кавычки.)

Хотя полное объяснение приведенного кода находится за пределами нашего обсуждения, чистый его эффект состоит в том, что броузер помещает в строку состояния своего окна сообщение «Click me!» (Нажми меня). Обычно авторы используют эту простую функцию JavaScript для более содержательного описания гиперссылки, чем тот, часто за-

[1] Ко времени загрузки можно отнести, и то условно, только единственное событие onLoad: полное завершение загрузки HTML-документа, содержащего JavaScript-фрагменты. Все остальные события определяют реакцию на действия пользователя над уже загруженным документом. – *Примеч. науч. ред.*

гадочный URL, который в строке состояния традиционно отображает броузер.[1]

И HTML, и XHTML поддерживают очень много обработчиков событий при помощи соотвествующих «onEvent» («поСобытию») атрибутов тегов. Значением каждого из JavaScript-обработчиков событий является заключенная в кавычки строка, содержащая один или несколько операторов JavaScript, разделенных точками с запятой. Чрезвычайно длинные последовательности операторов можно при необходимости разбить на несколько строк. Следует также обратить внимание на необходимость использования кодов символов двойных кавычек (\" или \x22) внутри операторов во избежание синтаксических ошибок при обработке значений атрибутов.

12.3.3.1. Стандартные атрибуты обработчиков событий

Таблица 12.1 представляет имеющееся в настоящее время множество стандартных обработчиков событий в качестве атрибутов тегов. Большинство из них поддерживаются популярными броузерами, которые поддерживают также и многообразие нестандартных обработчиков событий, помеченных в таблице звездочками.

Таблица 12.1. Обработчики событий

Обработчики событий	HTML/XHTML-теги
onAbort*	<img>
onBlur	<a>, <area>, <body>, <button>, <frameset>, <input>, <label>, <select>, <textarea>
onChange	<input>, <select>, <textarea>
onClick	Большинство тегов
onDblClick	Большинство тегов
onError*	<img>
onFocus	<a>, <area>, <body>, <button>, <frameset>, <input>, <label>, <select>, <textarea>
onKeyDown	Большинство тегов
onKeyPress	Большинство тегов
onKeyUp	Большинство тегов

[1] Тем не менее замещение стандартного сообщения, выводимого в строку состояния, на часто бессодержательное сообщение типа «Нажми меня!» и тем более прокрутка в строке состояния какого-либо рекламного текста, мешающего посетителю нормально ориентироваться в ссылках, считается дурным тоном. Существует множество других, менее деструктивных способов привлечь внимание к ссылке: изменить шрифт, цвет отображения, вставить или изменить под мышью пиктограмму, в конце концов просто заполнить атрибут title. – *Примеч. науч. ред.*

Обработчики событий	HTML/XHTML-теги
onLoad	<body>, <frameset>, <img>*
onMouseDown	Большинство тегов
onMouseMove	Большинство тегов
onMouseOut	Большинство тегов
onMouseOver	Большинство тегов
onMouseUp	Большинство тегов
onReset	<form>
onSelect	<input>, <textarea>
onSubmit	<form>
onUnload	<body>, <frameset>

Мы разделяем обработчики событий на две категории: пользовательские и документные. К пользовательским мы относим события, инициируемые мышью и/или клавиатурой, происходящие, когда пользователь производит какие-либо действия с каким-нибудь из этих устройств. Пользовательские события вездесущи, они появляются в качестве атрибутов почти во всех стандартных тегах (даже если они, возможно, не поддерживаются никаким броузером), поэтому мы не перечисляем соответствующие теги в табл. 12.1. Вместо этого мы отметим, какие теги *не имеют* этих атрибутов событий: <applet>, <base>, <basefont>, <bdo>,
, <font>, <frame>, <frameset>, <head>, <html>, <iframe>, <isindex>, <meta>, <param>, <script>, <style> и <title>.

Некоторые события, однако, встречаются редко и только с особыми тегами. Это относится к особым событиям и состояниям, случающимся в процессе отображения документа и его элементов броузером и в процессе управления документом и его элементами.

12.3.3.2. События, связанные с мышью

Атрибуты OnClick, onDblClick, onMouseDown и onMouseUp относятся к кнопкам мыши. Событие OnClick происходит тогда, когда пользователь нажимает и быстро отпускает кнопку мыши. Если только пользователь быстро щелкнет кнопкой мыши еще раз, то происходит другое событие – onDblClick.

Если нужно обратить внимание на каждую стадию щелчка мыши, используйте атрибуты onMouseDown и onMouseUp. Когда пользователь нажимает кнопку мыши, происходит событие onMouseDown, а когда отпускает кнопку, – onMouseUp.

События onMouseMove, onMouseOut и onMouseOver происходят, когда пользователь перемещает указатель мыши. Событие onMouseOver наступает, когда указатель мыши впервые попадает на ассоциированный с этим событием HTML-элемент. Вслед за этим событие onMouseMove наступа-

ет, когда мышь перемещается в пределах элемента. Наконец, событие onMouseOut наступает, когда мышь покидает элемент.

Для некоторых элементов событию onMouseOver соответствует onFocus, а событию onMouseOut – onBlur.

12.3.3.3. События, связанные с клавиатурой

С клавиатурой связаны в настоящее время только три принадлежащие стандартам HTML 4 и XHTML атрибута: onKeyDown, onKeyUp и onKeyPress. Событие onKeyDown наступает, когда пользователь нажимает на клавишу на клавиатуре, а событие onKeyUp – когда он ее отпускает. onKeyPress происходит, когда пользователь нажимает клавишу и отпускает. Обычно обрабатывается либо первая пара событий, либо третье составное событие, но не все вместе.

12.3.3.4. События, связанные с документом

Большинство обработчиков событий, связанных с документом, относятся к действиям и состояниям элементов формы. К примеру, onReset и onSubmit случаются, когда пользователь активизирует кнопки типа reset (сбросить) и submit (отправить) соответственно. Подобным образом onSelect и onChange происходят, когда пользователь взаимодействует с определенными элементами формы. Обратитесь, пожалуйста, к главе 9, где события, относящиеся к формам, обсуждаются подробно.

Имеются также несколько событий, относящихся к документу, происходящих, когда различные элементы документа обрабатываются броузером. Например, событие onLoad может произойти, когда закончено отображение набора файлов или когда тело HTML- или XHTML-документа загружается и отображается броузером. Подобным образом onUnload происходит, когда документ покидает фрейм или окно.

12.3.4. javascript–URL

Вы можете заменить любую традиционную URL-ссылку в документе на одно или несколько предложений JavaScript. Тогда, вместо того чтобы загружать новый документ, броузер выполнит этот JavaScript-код всякий раз, когда броузеру укажут на этот URL. Результат последнего оператора принимается в качестве «документа», на который указывает URL, и соответственно отображается броузером. Результат последнего оператора – это *не* URL документа, а действительное содержимое, которое должен отобразить броузер.

Чтобы создать javascript-URL, используйте javascript в качестве указания протокола URL:

```
<a href="javascript:generate_document()">
```

В этом примере JavaScript-функция generate_document() исполняется всякий раз, когда пользователь выбирает эту гиперссылку. Предполо-

жительно, значение, возвращаемое этой функцией, – это сгенерированный сценарием текст HTML- или XHTML-документа, который выводится и отображается броузером.

Может так случиться, что исполняемое предложение не возвращает никакого значения. В таком случае текущий документ остается без изменений. Следующий javascript-URL, например:

```
<a href="javascript:alert('Error!')">
```

выводит на экран диалоговое окно с предупреждением об ошибке и ничего более. Документ, содержащий гиперссылку, остается видимым после того, как диалоговое окно появится на экране и будет закрыто пользователем.

12.3.5. JavaScript-макросы

Коды символов (символьные ссылки) HTML и XHTML состоят из амперсанда (&), названия символа или его номера и завершаются точкой с запятой. Чтобы, например, вставить сам символ амперсанда в поток текста документа, используйте последовательность символов `&`. Подобным образом в JavaScript можно определить макросы, которые состоят из амперсанда, следующих за ним одного или нескольких операторов JavaScript, заключенных в фигурные скобки, и закрывающей точки с запятой. К примеру:

```
&{document.fgColor};
```

Если в фигурных скобках больше чем один оператор, они разделяются точками с запятой. Значение последнего (или единственного) оператора преобразуется в строку и замещает в документе JavaScript-код.

Обычно макросы могут встречаться в документе где угодно. Макросы JavaScript предназначены исключительно для значений атрибутов тегов. Это позволяет вам создавать «динамические теги», атрибуты которых неизвестны, пока документ не загружен и JavaScript не исполнен. Например:

```
<body text=&{favorite_color()};>
```

установит цвет текста в документе в соответствии с кодом цвета, возвращаемым, вероятно, заданной автором функцией `favorite_color()`. Поддержка макросов JavaScript не согласована между броузерами, и по этой причине мы не рекомендуем вам пользоваться макросами.

12.3.6. Тег <server>

Тег `<server>` – это зверь особой породы. Он обрабатывается только вебсервером, и его не видит броузер. Поэтому то, что вы можете делать с этим тегом, зависит от используемого вами сервера, а не от пользовательского броузера.

Netscape-серверы (не путайте их с броузером Netscape Navigator) используют тег `<server>`, чтобы позволить вам вставлять в документ Java-

<server>	
Функция:	Определяет JavaScript, исполняемый на стороне сервера
Атрибуты:	Отсутствует
Завершающий тег:	</server>; присутствует обязательно
Содержит:	*операторы_JavaScript*
Может содержаться в:	*head_content* (содержимое тела)

Script операторы, которые будет обрабатывать сервер. Результат исполнения JavaScript затем вставляется в документ на место тега <server>. Полное обсуждение этого так называемого JavaScript со стороны сервера находится совершенно за пределами настоящей книги. Мы включаем сюда это краткое упоминание, только чтобы задокументировать тег <server>.

Подобно тегу <script> тег <server> содержит JavaScript-код. Последний из них, однако, вместе со своим JavaScript-содержимым может находиться только в заголовке документа. Он извлекается из документа и исполняется сервером, когда документ запрашивается.

Очевидно, что JavaScript со стороны сервера тесно связан с сервером, а не с броузером. Чтобы использовать все возможности этого тега, равно как и преимущества, проистекающие из JavaScript со стороны сервера или других языков программирования со стороны сервера, обратитесь к документации по вашему веб-серверу.

12.4. Таблицы стилей JavaScript (устарели)

Манипулирование отображением составляет значительную часть работы броузера, и значительная часть кодов, управляющих отображением, пишется на JavaScript. Поэтому естественно, что разработчики из Netscape реализовали таблицы стилей JavaScript (JavaScript Stylesheets, JSS), и думается, что это было относительно просто сделать.[1] Эта альтернативная технология стилей документа, основывающаяся на рекомендованной W3C модели каскадных таблиц стилей (CSS; см. главу 8), позволяет вам устанавливать свойства отображения различных элементов HTML как в виде атрибутов, встроенных в тег, так и на уровне документа, либо в виде внешнего файла стилей, используемого для целого собрания документов.

Таблицы JSS устарели. Даже их изобретатель отказался от поддержки JSS в пользу стандарта CSS2.

---

[1] Последствия такого «простого» решения до сих пор расхлебывают (недолго уж им осталось) пользователи Netscape Navigator. Ведь в этом броузере отключение выполнения JavaScript влечет за собой невозможность отработки стилей CSS. – *Примеч. науч. ред.*

Мы твердо стоим за разумную стандартизацию, и теперь, когда модель CSS2 полностью поддерживается стандартами HTML 4 и XHTML, мы не можем рекомендовать вам использовать что-либо, кроме таблиц стилей, соответствующих стандарту CSS.

Мы подробно обсудили идеи и понятия, стоящие за таблицами стилей – особенно каскадными таблицами стилей, – в главе 8, так что не будем здесь повторяться. Вместо этого мы коснемся только того, как следует создавать стили JavaScript и манипулировать ими (исключительно из любви к истории). Прежде чем продолжать чтение этого раздела, мы рекомендуем вам усвоить информацию главы 8.

12.4.1. Синтаксис таблиц стилей JavaScript

В Netscape версии 4 и более ранних JSS реализованы путем расширения нескольких существующих тегов HTML и определения нескольких новых объектов для хранения стилей ваших документов. Netscape больше не поддерживает JSS, да и любой другой броузер – тоже.

12.4.1.1. Внешние, на уровне документа и встроенные JSS

Как и в случае CSS, можно указать и загрузить внешние JSS-файлы при помощи тега `<link>`. Например:

```
<link href="styles.js" rel=stylesheet type=text/JavaScript>
```

Единственное существенное различие между этим тегом и таким же для внешних таблиц стилей CSS состоит в том, что атрибуту `type` тега `<link>` присваивается значение `text/JavaScript` вместо `text/CSS`. Файл, на который указывает тег, *styles.js*, содержит предложения JavaScript, определяющие стили и классы, используемые затем Netscape для управления отображением текущего документа.

JSS на уровне документа определяется в теге `<style>` в заголовке документа, точно так же, как в случае CSS. И снова имеется только одно существенное различие – в том, что атрибуту `type` тега `<style>` присваивается значение `text/JavaScript` вместо `text/CSS`.

Содержимое тега `<style>` JSS, однако, совершенно отлично от такового в CSS. Например:

```
<style type=text/JavaScript>
<!--
    tags.BODY.marginLeft = "20px";
    tags.P.fontWeight = "bold";
  // -->
</style>
```

Заметьте, во-первых, что мы используем стандартные JavaScript- и HTML-комментарии, окружающие наши JSS-определения, чтобы не поддерживающие JSS броузеры не стали обрабатывать их как HTML-содержимое. Заметьте также, что синтаксис определений стиля соот-

ветствует JavaScript, где помимо всего прочего регистр букв имеет значение.

Вы ассоциируете встроенные стилевые правила JavaScript с определенным тегом, используя атрибут `style` так же, как и для встроенных стилей CSS. Значением атрибута является список JSS-присваиваний, разделенных точками с запятой. Например:

```
<p style="color = 'green'; fontWeight = 'bold'">
```

создает абзац с жирным зеленым текстом. Заметьте, во-первых, что вам нужно заключать значения встроенных стилей в одиночные, а не в двойные кавычки, которые вы могли использовать в определениях стиля на уровне документа и во внешних таблицах. Это вызвано тем, что значение атрибута `style` должно заключаться в двойные кавычки (а тип кавычек внутри описания значения должен отличаться от кавычек, ограничивающих значение атрибута).

Отметьте также, что встроенные в тег JSS-определения используют только название свойства и не содержат имени тега, обладающего этим свойством. Это допустимо, поскольку встроенные в тег JSS-стили влияют только на текущий тег, а не на все вхождения тега в документ.

12.4.1.2. JSS-значения

В общем, все значения, которые вы можете использовать для CSS, могут использоваться также и в JSS-определениях. В случае ключевых слов, длин и процентных значений заключайте только эти значения в кавычки и используйте их так, как вы делали бы это с любым строковым значением в JavaScript. Таким образом, CSS-значение `bold` превращается в `"bold"` или `'bold'` в JSS-стилях на уровне документа или во встроенных в тег стилях соответственно. `12pt` в CSS превращаются в `'12pt'` или в `"12pt"` в JSS.

Указывайте цветовые значения в виде названий цветов или шестнадцатеричных кодов, заключая их в одиночные или двойные кавычки. Десятичные RGB-обозначения CSS не поддерживаются в JSS.

В JSS URL значения являются строками, содержащими соответствующий URL. Таким образом, CSS URL, определенный как `url(http://www.kumquat.com)`, превращается в `'http://www.kumquat.com'` во встроенных стилях JSS и в `"http://www.kumquat.com"` – в стилях на уровне документа.

Одно преимущество, свойственное только JSS, состоит в том, что любое значение может вычисляться динамически, в процессе обработки документа броузером. Вместо статического определения размера шрифта, например, вы можете вычислить его на ходу:

```
tags.P.fontSize = favorite_font_size();
```

Мы предполагаем здесь, что JavaScript-функция `favorite_font_size()` как-то определяет желательный размер шрифта и возвращает строковое значение, содержащее этот размер. Оно, в свою очередь, присваи-

вается свойству fontSize для тега <p>, определяя размер шрифта во всех абзацах документа.

12.4.1.3. Определение стилей для тегов

JavaScript определяет новое свойство tags объектов document, содержащее свойства стилей для всех тегов документа. Для определения стиля для тега просто установите необходимое значение нужного свойства в свойстве tags объекта document. К примеру:

```
document.tags.P.fontSize = '12pt';
document.tags.H2.color = 'blue';
```

Эти два JSS-определения устанавливают размер шрифта для тега <p> в 12 пунктов, а все теги <h2> предписывают выводить синим цветом. Эквивалентные CSS определения выглядят так:

```
p {font-size : 12pt}
h2 {color : blue}
```

Поскольку свойство tags всегда относится к текущему документу, вы можете опустить document в любом JSS-определении стилей тега. Мы могли записать предыдущие два стиля как:

```
tags.P.fontSize = '12pt';
tags.H2.color = 'blue';
```

Более того, как мы отмечали выше, можно опустить имя тега так же, как свойства document и tags, во встроенных в тег JSS, использующих атрибут style.

Регистр в JSS имеет значение. Имена тегов в свойстве tags должны набираться целиком заглавными буквами. Заглавные буквы в середине названий свойств тегов имеют значение – любое уклонение от точного побуквенного их воспроизведения влечет ошибку, и Netscape не станет учитывать такое JSS-объявление. Все нижеследующие JSS-определения недействительны, хотя причина этого далеко не очевидна:

```
tags.P.fontSize = '12pt';
tags.Body.Color = 'blue';
tags.P.COLOR = 'red';
```

Их правильные варианты таковы:

```
tags.P.fontSize = '12pt';
tags.BODY.color = 'blue';
tags.P.color = 'red';
```

Может быть очень скучным занятием определять множество свойств для одного тега, так что можете воспользоваться JavaScript-предложением with, чтобы уменьшить объем набора. Следующие стили:

```
tags.P.fontSize = '14pt';
tags.P.color = 'blue';
tags.P.fontWeight = 'bold';
```

```
tags.P.leftMargin = '20%';
```

могут быть записаны короче:

```
with (tags.P) {
  fontSize = '14pt';
  color = 'blue';
  fontWeight = 'bold';
  leftMargin = '20%';
  }
```

Можно применить один набор стилей к нескольким тегам:

```
with (tags.P, tags.LI, tags.H1) {
  fontSize = '14pt';
  color = 'blue';
  fontWeight = 'bold';
  leftMargin = '20%';
  }
```

12.4.1.4. Определения классов стилей

Как и в CSS, можно применять JSS-стили к тегам, особым образом используемым в документе. В JSS свойство `classes` определяет различные стили для одного тега. У свойства `classes` нет заранее определенных свойств, но любое указанное свойство определяется в качестве класса, который может быть применен в текущем документе. Например:

```
classes.bold.P.fontWeight = 'bold';
with (classes.abstract.P) {
   leftMargin = '20pt';
   rightMargin = '20pt';
   fontStyle = 'italic';
   textAlign = 'justify';
   }
```

Первый стиль определяет класс `bold` для тега `<p>`, который будет отображаться полужирным шрифтом. Следующий стиль использует предложение `with` для создания класса тегов `<p>` под названием `abstract`. Эквивалентные CSS правила были бы такими:

```
P.bold {font-weight : bold}
P.abstract {left-margin : 20pt;
            right-margin : 20pt;
            font-style : italic;
            text-align : justify
           }
```

Определенные по правилам JSS-классы используются так же, как классы CSS, – с применением атрибута `class` и имени класса.

Как CSS, так и JSS позволяют вам также определять классы без указания тега, к которому будет применяться класс. Это позволяет определять родовые классы, которые вы затем сможете применять к любому

тегу. Для создания родового класса стилей используйте специальное свойство `all`:

```
classes.green.all.color = "green";
```

Впоследствии вы сможете вставить `class = "green"` в любой тег, чтобы Netscape вывел его содержимое зеленым цветом. Эквивалентное CSS определение:

```
.green {color : green}
```

12.4.1.5. Использование контекстных стилей

Одно из самых мощных свойств CSS заключается в возможности создания контекстных стилей, когда броузер применяет стиль только к тегам, определенным образом вложенным в документ. JSS также поддерживает контекстные стили при помощи метода `contextual()` свойства `tags`. Параметрами этого метода являются теги и классы, определяющие контекст, в котором Netscape применит этот стиль. Например:

```
tags.contextual(tags.UL, tags.UL, tags.LI).listStyleType = 'disc';
```

определяет контекст, в котором элемент (`tags.LI`) неупорядоченного списка, вложенного в еще один неупорядоченный список, использует диск в качестве маркера. CSS-эквивалент:

```
ul ul li {list-style-type : disc}
```

Вы можете смешивать теги и классы в методе `contextual()`. Например:

```
tags.contextual(classes.abstract.P,tags.EM).color = 'red';
```

предлагает броузеру отобразить содержимое тега `<em>` красным, когда он встретится внутри абзаца из класса `abstract`. CSS-эквивалент таков:

```
p.abstract em {color : red}
```

Поскольку объект `tags` недвусмысленно включен в метод `contextual()`, можно опустить его в определении. Следовательно, наш пример с вложенными списками можно переписать так:

```
tags.contextual(UL, UL, LI).listStyleType = 'disc';
```

12.4.2. Свойства таблиц стилей JavaScript

В JSS поддерживается подмножество свойств CSS. Стилевые свойства JSS, их CSS-эквиваленты и разделы, в которых эти свойства полностью задокументированы, представлены в табл. 12.2.

В JSS определены также три метода, позволяющие вам определять ширину полей, подложки и рамки в одном стилевом свойстве. Каждый из этих трех методов: `margins()`, `paddings()` и `borderWidths()` – получает четыре параметра, соответствующих верхней, правой, нижней и левой ширине поля, подложки и рамки. В отличие от своих CSS-аналогов (`margin`, см. раздел 8.4.7.11; `padding`, см. раздел 8.4.7.12 и `border-width`,

см. раздел 8.4.7.4), эти JSS-методы требуют, чтобы всегда были указаны все четыре параметра. В JSS нет сокращенного способа установки нескольких значений ширины полей, подложки или рамки при помощи одного значения.

Таблица 12.2. Свойства JSS и их эквиваленты в каскадных таблицах стилей

JSS-свойство	CSS-свойство	См. раздел
align	float	8.4.7.9
backgroundImage	background-image	8.4.5.2
backgroundColor	background-color	8.4.5.1
borderBottomWidth	border-bottom-width	8.4.7.4
borderLeftWidth	border-left-width	8.4.7.4
borderRightWidth	border-right-width	8.4.7.4
borderStyle	border-style	8.4.7.5
borderTopWidth	border-top-width	8.4.7.4
cClear	clear	8.4.7.7
display	display	8.4.10.1
fontSize	font-size	8.4.3.2
fontStyle	font-style	8.4.3.5
height	height	8.4.7.10
lineHeight	line-height	8.4.6.2
listStyleType	list-style-type	8.4.7.3
marginBottom	margin-bottom	8.4.7.11
marginLeft	margin-left	8.4.7.11
marginRight	margin-right	8.4.7.11
marginTop	margin-top	8.4.7.11
paddingBottom	padding-bottom	8.4.7.12
paddingLeft	padding-left	8.4.7.12
paddingRight	padding-right	8.4.7.12
paddingTop	padding-top	8.4.7.12
textDecoration	text-decoration	8.4.6.4
textTransform	text-transform	8.4.6.7
textAlign	text-align	8.4.6.3
textIndent	text-indent	8.4.6.5
verticalAlign	vertical-align	8.4.6.7
whiteSpace	white-space	8.4.10.2
width	width	8.4.7.16

- Обзор динамических документов
- Client-pull-документы
- SP-документы

13

Динамические документы

Модель документа в стандартах HTML 4 и XHTML является статической. Отображенный броузером документ не меняется, пока пользователь не начинает какой-либо деятельности, подобной выбору гиперссылки. Разработчики Netscape посчитали это ограничение неприемлемым и встроили в свои броузеры специальные средства, позволяющие динамически изменять содержимое HTML-документа. На деле они представили два различных механизма для динамических документов, их мы и опишем подробно в этой главе. Internet Explorer поддерживает некоторые из этих механизмов, что мы также обсудим.

Следует отметить, что множество черт динамических документов позже вытеснены плагинами к броузерам и некоторыми апплетами, а также новой технологией Ajax (Asynchronous JavaScript and XML). Тем не менее Netscape и Internet Explorer продолжают поддерживать динамические документы, и мы полагаем, что это технология имеет свои преимущества, о которых вы должны быть осведомлены, даже если и не собираетесь пользоваться ими при создании ваших HTML-документов. [апплеты и объекты, 12.1]

13.1. Обзор динамических документов

Вы, возможно, помните из главы 1, что броузер клиента инициирует передачу данных по Сети, контактируя с сервером и запрашивая у него документ. Затем клиент отображает содержимое документа пользователю. В случае обычного документа все, что нужно для его получения и представления пользователю, – это одиночная транзакция, инициированная со стороны клиента. Однажды отображенный, этот документ уже не изменяется.

Динамические документы, напротив, являются результатом множественных транзакций, инициированных либо со стороны сервера, либо

со стороны клиента, либо с обеих сторон. Динамический документ, *автоматически запрашиваемый клиентом* (*client-pull documents*), инициирует множественные транзакции со стороны клиента. Когда инициатором общения является сервер, динамический документ называется *SP-документом*, или *server-push document*.

В документах, автоматически запрашиваемых клиентом, специальный HTML-код приказывает клиенту периодически запрашивать и загружать новые документы с одного или нескольких серверов в сети, динамически обновляя отображение.

Документы, *поставляемые сервером*, тоже улучшают общение сервера с клиентом. Обычно в Интернете клиент устанавливает соединение с сервером ровно на то время, которое требуется, чтобы получить один запрошенный документ. В случае документов, поставляемых сервером, соединение остается открытым и сервер продолжает периодически посылать клиенту данные, пополняя или замещая предыдущее содержимое.

Броузеры, основанные на движке Mozilla, такие как Netscape и Firefox, способны корректно работать с SP-документами. Поскольку динамические документы неработоспособны без HTTP-сервера, их невозможно разрабатывать и тестировать, если они хранятся как локальные файлы и если отсутствует локально работающий сервер.

13.1.1. Еще одно предостережение

Как всегда, мы точно расскажем вам, как использовать эти вдохновляющие, но нестандартные возможности, но мы советуем вам не использовать их, если только у вас нет убедительных и неоспоримых оснований так поступить. Мы так настаиваем на этом предостережении в отношении динамических документов не только потому, что они не входят в стандарт HTML, но и потому, что они пожирают сетевые ресурсы. Они требуют больших объемов и большей продолжительности передачи данных, чем их статические аналоги. И они требуют большего числа (в случае CP-документов) и более продолжительных (в случае SP-документов) соединений с сервером. Лишь ограниченное число пользователей из многих миллионов одновременно смогут установить множественные соединения с одним сервером. Нам бы не хотелось видеть, как вы теряете ваших читателей оттого только, что вы создали прыгающую картинку в динамическом документе, без которой статический документ мог бы быть эффективным и легко доступным и насладиться им смогли бы множество людей.

13.2. Client-pull-документы

Создать такой динамический документ относительно легко, и выполнять его вы можете без какого бы то ни было HTTP-сервера. Дело в том, что документ этого типа заставляет броузер запросить и загру-

зить другой документ, например хранящийся локально. Все, что для этого нужно, – это включить в заголовок вашего HTML- или XHTML-документа тег <meta>. Этот специальный тег предлагает броузеру отображать текущий документ в течение указанного периода времени, а затем загрузить и отобразить новый документ, точно так, как если бы пользователь выбрал новый документ при помощи гиперссылки. (Заметьте, что в настоящее время нет способа динамически изменить только часть документа с применением CP-технологии.) [<meta>, 6.8.1]

13.2.1. Заголовок Refresh

Client-pull динамические документы работают со всеми популярными броузерами, поскольку эти броузеры реагируют на специальное поле HTTP-заголовка, именуемое Refresh (Обновить).

Вы, возможно, помните из наших предыдущих обсуждений, что когда бы HTTP-сервер ни посылал броузеру клиента документ, он предпосылает данным документа одно или несколько заголовочных полей. Одно заголовочное поле, например, содержит описание типа содержимого документа, используемое броузером при решении вопроса, как следует отображать содержимое документа. HTML-документам, например, сервер предпосылает заголовок Content-type: text/html, значение которого совершенно очевидно: он указывает, что содержимое документа будет передаваться текстом и интерпретировать его надо как HTML-документ.

Как мы долго обсуждали в главе 6 «Гиперссылки и Сети», можно добавить к HTTP-заголовку HTML-документа особые поля, вставляя в заголовок (<head>) документа тег <meta>. [элемент заголовка <meta>, 6.8.1]

HTTP-поле Refresh организует Client-pull динамический HTML-документ, включаемый тегом <meta> следующего формата:

```
<meta http-equiv="Refresh" content="field value">
```

Атрибут http-equiv предписывает HTTP-серверу включить в последовательность заголовков, отправляемую им броузеру клиента непосредственно перед отправкой остального содержимого документа, поле Refresh (обновить), которому присваивается значение, указываемое в атрибуте content (как всегда, заботливо заключенное в кавычки). Броузер интерпретирует заголовок Refresh в качестве признака динамического HTML-документа и реагирует так, как это описано ниже.

13.2.2. Значение поля Refresh

Значением атрибута content в теге <meta> Refresh определяется, как и когда броузер обновляет текущий документ. Присвойте ему целое значение и броузер отложит автоматическую загрузку нового документа на это число секунд. Вы можете присвоить ему нулевое значение, означающее отсутствие задержки. В этом случае броузер загружает следующий документ сразу после того, как закончит вывод текущего до-

кумента, в результате чего вы можете получить очень грубую анимацию. [атрибут content, 6.8.1.2]

13.2.2.1. Обновление текущего документа

Если значение поля Refresh, указанного в content, это просто число секунд, броузер будет заново загружать текущий документ снова и снова, выдерживая указанный временной интервал между циклами, до тех пор, пока пользователь не перейдет к другому документу или не закроет броузер.

К примеру, броузер будет перегружать следующий Client-pull-документ каждые 15 секунд:

```
<html>
<head>
<meta http-equiv="Refresh" content="15">
<title>Цены на рынке кумкватов</title>
</head>
<body>
<h3>Цены на рынке кумкватов</h3>
Текущие торги кумкватов проходят по $1.96 за фунт.
</body>
</html>
```

Знатоки финансов из числа наших читателей могут заметить, что при помощи специальных программных трюков со стороны сервера можно обновлять цену кумкватов в документе, так что этот документ будет работать как динамический ценник, обновляя текущую цену кумкватов каждые 15 секунд.

13.2.2.2. Обновление при помощи другого документа

Вместо того чтобы раз за разом загружать один и тот же документ, можно предписать броузеру динамически загрузить другой документ. Это делается путем добавления абсолютного URL этого документа через точку с запятой после времени задержки в атрибуте content тега <meta>. Например:

```
<meta http-equiv="Refresh"
 content="15; URL=http://www.kumquat.com/next.html">
```

повлечет за собой то, что броузер будет должен получить документ *next.html* с сервера *www.kumquat.com*, продержав на экране текущий документ в течение 15 секунд.

13.2.2.3. Зацикливание документов

Имейте в виду, что действие тега <meta> Refresh относится только к тому документу, в котором он появился. Поэтому, чтобы создать цикл, содержащий несколько документов, нужно включить тег <meta> Refresh в каждый из них. Для достижения такого эффекта укажите в ка-

честве значения атрибута content в каждом документе абсолютный URL, указывающий на следующий документ, тогда как последний документ должен указывать на первый, чтобы замкнуть цикл.

К примеру, ниже следуют теги <meta> для заголовков каждого из трех замкнутых в цикл HTML-документов.

Документ *first.html* содержит:

```
<meta http-equiv="Refresh"
 content="30; URL=second.html">
```

Документ *second.html* содержит:

```
<meta http-equiv="Refresh"
 content="30; URL=third.html">
```

И третий документ содержит в своем заголовке (не считая других безумных мыслей):

```
<meta http-equiv="Refresh"
 content="30; URL=first.html">
```

Если оставить броузер в покое, он будет бесконечно крутиться между этими тремя документами с 30-секундным интервалом.

Зацикленные документы превосходно удерживают внимание, заставляя, например, прохожих останавливаться у веб-киоска. Пользователи могут затем самостоятельно побродить по более широкому множеству документов киоска, выбирая гиперссылки на одной, а затем и на другой странице заманчивого калейдоскопа.[1]

Чтобы вернуться к циклу страниц-приманок, каждый документ в остальной части коллекции киоска должен содержать свое поле Refresh, которое в конце концов указывает на приманку. Для страниц, не входящих в приманку, следует указать гораздо больший интервал времени – от 120 до 300 секунд или даже больше, – чтобы киоск не обновлялся автоматически, когда пользователь читает текущий документ. Период задержки, однако, должен быть достаточно короток, чтобы киоск возвращался в режим калейдоскопа через разумное время после того, как пользователь закончил ознакомление с документом.

13.2.3. Получение содержимого, отличного от HTML

Client-pull-возможности Netscape и Internet Explorer не ограничиваются HTML-документами, хотя, конечно, легче всего создать динамический документ при помощи HTML. Используя совсем небольшое усилие по программированию со стороны сервера, можно добавить по-

[1] Здесь уместно отметить, что пользователи могут в любой момент прекратить динамическое поведение, определенное полем Refresh, выбрав гиперссылку, например, пока не закончился перерыв между именами документов. Броузер всегда пренебрегает действием Refresh в пользу действий пользователя.

ле Refresh в HTTP-заголовок документа любого типа, включая изображения, аудиофайлы и видеоклипы.

Например, создайте видеопоток, добавляя заголовочное поле Refresh к каждому из последовательных изображений, полученных с видеокамеры и затем оцифрованных. Включите нулевую задержку и URL, указывающий на следующее изображение, так что, как только броузер выведет одно изображение, он начнет получать следующее. Если сеть работает хорошо, результатом будет грубое (очень грубое) TV.

Поскольку броузер очищает окно перед тем, как представить очередное изображение, мерцание и мелькание сделают почти невозможным показ связной последовательности изображений. Эта техника более эффективна, когда представляется последовательность изображений, сконструированная так, чтобы ее можно было разглядывать как слайдшоу, где пользователю предоставляется возможность совершить какие-либо действия между появлениями очередных изображений.

Возможно, лучшим употреблением Client-pull-возможностей будет их сочетание с длинными мультимедийными документами, для отображения которых популярные броузеры используют специальные вспомогательные приложения. На многозадачных компьютерах, на которых стоят Linux или Windows, броузер загружает один документ, в то время как вспомогательное приложение проигрывает другой. Комбинируйте Client-pull-возможности с этой многозадачностью для улучшения исполнения мультимедийных документов. Чтобы не приходилось ожидать загрузки одного большого документа, такого как видео- или аудиофайл, прежде чем начать его проигрывать, разбейте его на фрагменты меньшего размера, чтобы каждый из них автоматически загружался вслед за предыдущим (при помощи заголовка Refresh). Броузер будет проигрывать первый фрагмент во время загрузки следующего, затем третьего, затем четвертого и т. д.

13.2.4. Комбинирование поля Refresh с другими полями заголовка

Можно заставить Client-pull-документы выполнять некоторые изящные трюки, комбинируя эффекты поля Refresh и других полей HTTP-заголовка. Одна комбинация полезна особенно – поле Refresh с полем Redirect.

Поле Redirect позволяет серверу предложить броузеру получить запрошенный документ в каком-то другом месте, указывая в сопровождающем поле значение URL этого места. Броузер клиента автоматически перенаправляет свой запрос по новому URL и получает документ из другого места, обычно не уведомляя об этом пользователя. Мы можем все время получать перенаправленные документы и не замечать этого. Самая обычная причина перенаправления состоит в том, что собрание HTML-документов переместилось в новый каталог или на новый сер-

вер. В качестве любезности вебмастер программирует свой прежний сервер так, чтобы тот отправлял HTTP-заголовок, включающий поле Redirect и новый URL (но не включающий тела документа), всякому броузеру, запрашивающему документ по его прежнему адресу. Таким образом, новое расположение документа оказывается доступным пользователям, и им не приходится обновлять закладки своего броузера.

Но иногда нужно, чтобы пользователи переустановили свои закладки на новое расположение, потому что старое не будет перенаправлять броузеры вечно, возможно, потому, что выключается из Сети вообще. Один из способов оповестить пользователей о новом местоположении состоит в том, чтобы URL перенаправления стал указывать на HTML-документ, не являющийся домашней страницей новой коллекции, а содержащий информацию о новом местоположении. Однажды уведомленные, пользователи затем выберут гиперссылку «Продолжить», ведущую к новому расположению домашней страницы, и соответственно установят свои закладки.

Комбинируя поля Redirect и Refresh, можно заставить этот документ с объявлением автоматически переходить к новой странице. Если броузер получает HTTP-заголовок с обоими полями, он учтет их оба – он немедленно отправится по перенаправленному URL и отобразит его, а также установит таймер обновления и URL замещающего документа, если они указаны. Когда время выйдет, броузер автоматически перейдет по следующему URL, указывающему на новое расположение вашей домашней страницы.

13.2.4.1. Генератор случайных URL

Другим приложением сочетания полей Redirect и Refresh HTTP-заголовка является случайное блуждание по URL. Вам понадобятся некоторые навыки в программировании для создания приложения со стороны сервера, которое выбирает случайный URL из заранее заготовленного списка и выводит поле Redirect, указывающее на этот URL, и поле Refresh, которое после некоторой задержки снова обращается к приложению, выдающему случайные URL.

Когда броузер получит полный заголовок, он немедленно загрузит и отобразит случайно выбранный документ, указанный в URL поля Redirect. Подождав указанное в поле Refresh время, броузер вернется к генератору случайных URL на сервере (как это указано в Refresh URL), и цикл начнется заново. Результат состоит в бесконечном цикле случайных URL, отображаемых с регулярным интервалом.

13.2.5. Соображения об эффективности

Документы, запрашиваемые клиентом (client-pull documents, или CP-документы), потребляют лишние сетевые ресурсы, особенно когда интервал между обновлениями мал, поскольку каждое обновление может требовать совершенно нового соединения с сервером. Броузеру мо-

жет понадобиться несколько секунд, чтобы соединиться с сервером и начать получение документа.[1] В результате быстрое обновление обычно оказывается недостижимым, особенно при медленном сетевом соединении.

Используйте Client-pull-документы для редко происходящих обновлений целого документа или для циклического перехода между документами, не требующего вмешательства пользователя.

13.3. SP–документы

Динамические документы, рассылаемые сервером, были изобретены в фирме Netscape. Соединение клиент-сервер остается открытым после первоначальной передачи данных, и сервер периодически посылает клиенту новые данные, обновляя отображение документа. SP-поведение делается возможным за счет некоего специального программирования со стороны сервера и осуществляется при помощи специального компонента MIME-стандарта, принятого в компьютерной индустрии для передачи через Интернет мультимедийных документов.

13.3.1. Тип передачи данных multipart/mixed

Как мы уже отмечали выше в этой главе при обсуждении CP-документов, HTTP-сервер передает броузеру клиента данные, состоящие из двух частей: заголовка, описывающего документ, и следующего за ним собственно документа. MIME-тип документа – это часть HTTP-заголовка. Обычно перед отправкой HTML-документа сервер включает в заголовок `Content-type: text/html`. Заменив этот тип содержимого на `multipart/mixed`, вы можете послать HTML-документ или несколько документов, разбив их на несколько частей, а не одним куском. Однако броузеры, основанные на движке Mozilla, такие как Netscape и Firefox, понимают эту запись в заголовке, сообщающую, что документ состоит из нескольких частей, и правильно на нее реагируют. Другие броузеры либо игнорируют добавочные части, либо отключают документ целиком.

Общая форма заголовка, сообщающего о принадлежности передаваемых данных MIME-типу `multipart/mixed`, выглядит примерно так:

```
Content-type: multipart/mixed;boundary="СлучайнаяСтрока"
```

Этот компонент HTTP-заголовка говорит броузеру Mozilla, что ожидается прибытие документа, состоящего из нескольких частей, и что следует обратить внимание на разграничитель `СлучайнаяСтрока`, отделяющий части друг от друга. Эта строка, разграничивающая части, должна быть уникальной и не должна появляться нигде внутри отдельных

[1] Кроме того, при установке соединения могут возникать произвольные (и достаточно длительные) паузы в обслуживании сервером, что случается при большой загрузке сервера остальными клиентами. – *Примеч. науч. ред.*

передаваемых частей. Содержимое данных, передаваемых от сервера клиенту, выглядит примерно так:

```
--СлучайнаяСтрока
Content-type: text/plain

Данные для первой части
--СлучайнаяСтрока
Content-type: text/plain

Данные для второй части

--СлучайнаяСтрока--
```

В этом примере документ состоит из двух чисто текстовых частей. Сервер посылает каждую часть, вставляя перед ней два тире и разграничитель `СлучайнаяСтрока`, за которыми идут поле `Content-type` и данные частей документа. Последняя передача от сервера к клиенту – это строка-разграничитель с двумя дополнительными тире после нее, обозначающими, что передана последняя часть документа.

При получении каждой части броузер Mozilla автоматически добавляет поступившие данные к отображению текущего документа.

Чтобы сделать возможным функционирование SP-документа такого типа, необходимо написать специальное приложение для HTTP-сервера. Это приложение должно создавать специальный HTTP MIME-заголовок `multipart/mixed` и посылать различные документы, разделяя их разграничителем.

13.3.2. Тип передачи данных multipart/x-mixed-replace

Авторы SP-документов могут использовать экспериментальный вариант MIME-типа `multipart/mixed`, известный как `multipart/x-mixed-replace`. Разница между этим особым типом содержимого и его предшественником состоит в том, что вместо того чтобы просто добавлять содержимое к текущему отображению, в replace-версии каждая последующая часть замещает предыдущую.

Формат замещающего HTTP-заголовка очень похож на его аналог для типа `multipart/mixed`, единственная разница состоит в `Content-type`:

```
multipart/x-mixed-replace; boundary=СлучайнаяСтрока
```

Все другие правила, касающиеся формата содержимого, состоящего из нескольких частей, – те же самые, включая использование разграничителя в виде строки и указания типа данных в каждой части содержимого.

13.3.3. Использование документов, состоящих из нескольких частей

Легко понять, как можно использовать эти два специальных составных MIME-типа для создания SP-документов. Передавая отдельные части с временными промежутками, вы можете создать автоматически прокручивающееся сообщение в окне броузера Mozilla. Или, замещая фрагменты документа с использованием MIME-типа `x-mixed-replace`, вы можете включить в ваш документ динамическую доску объявлений и даже, возможно, анимацию.

Заметьте, что составные SP-документы не обязаны принадлежать к числу HTML или других текстовых документов. Изображения – это тоже содержимое, кодирующееся в соответствии со стандартом MIME, поэтому HTTP-сервер может также передавать несколько изображений по очереди в качестве частей составной передачи. Поскольку можно также добиться, чтобы каждое следующее изображение замещало предыдущее, результатом будет грубая анимация. Если сделать это правильно, при достаточной скорости передачи результат может оказаться вполне удовлетворительным.

13.3.3.1. Соображения об эффективности

Документы, рассылаемые сервером (server-push documents, или SP-документы), поддерживают открытым соединение между клиентом и сервером на все время действия динамического документа. От некоторых серверов это может потребовать лишних затрат сетевых ресурсов, а также сохранения в активном состоянии нескольких процессов, обслуживающих открытое соединение. Убедитесь, что каждый SP-процесс (и, следовательно, каждое соединение клиент-сервер) прекращается после завершения передачи или по истечении некоторого периода молчания. В противном случае кто-нибудь нечаянно зависнет на бесконечно передающемся SP-документе и отберет ресурсы у сервера и, в конечном счете, у других пользователей.

Прежде чем решиться на создание SP-документов, убедитесь, что ваш сервер способен выдержать добавочную нагрузку по обработке данных и их отправке. Имейте в виду, что активными могут оказаться сразу много SP-документов, а это многократно увеличит нагрузку на сервер и может серьезно повлиять на его работу в целом.

13.3.4. Создание SP–документов

Создайте специальное приложение, работающее вместе с HTTP-сервером, чтобы обеспечить функционирование динамических документов, рассылаемых сервером. Это приложение должно создавать специальные MIME-заголовки `Content-type`, уведомляющие броузер Mozilla, что следующий за ними документ придет по частям – добавляющимся к текущему документу или замещающим часть текущего документа.

Приложение должно также создавать подходящую строку-разграничитель и посылать для каждой передаваемой части заголовок Content-type и собственно данные, возможно, отделяя передачу последовательных частей некоторым промежутком времени. Вам понадобится обратиться к документации вашего сервера, чтобы узнать, как следует создавать приложения со стороны сервера, которые могут вызываться при обращении к определенному URL на сервере. Для некоторых серверов это может оказаться не труднее, чем поместить приложение в некоторый каталог на сервере. Для других вам придется долго и мучительно искать обходные пути.

13.3.4.1. Пример SP-приложения для NCSA и Apache httpd

Программы NCSA (National Center for Supercomputing Applications) и Apache *httpd* как веб-серверы используются на большинстве Unix- и Linux-систем. Администраторы обычно конфигурируют сервер так, чтобы он запускал приложения со стороны сервера, хранящиеся в каталоге *cgi-bin*.

Ниже следует простой сценарий командного процессора Unix, иллюстрирующий, как следует посылать составные документы броузеру Netscape или Firefox при помощи *httpd*:[1]

```
#!/bin/sh
#
# Информируем клиента о том, что мы посылаем составной документ
# с граничной строкой "NEXT"
#
echo "HTTP/1.0 200"
echo "Content-type: multipart/x-mixed-replace;boundary=NEXT"
echo ""
echo "--NEXT"
while true
do
#
# Посылаем следующую часть, сопровождаемую граничной строкой
# После бездействуем пять секунд и повторяем
#
  echo "Content-type: text/html"
  echo ""
  echo <html>
  echo <head>
  echo "<title>Список серверных процессов</title>"
```

[1] NCSA *httpd* свойственна болезненная нетерпимость к пробелам в поле Content-type, которое предшествует вашему составному документу. Некоторые авторы любят оставлять пробел после точки с запятой перед ключевым словом boundary. Не позволяйте себе такого с NCSA *httpd*. Записывайте все поле Content-type подряд без пробелов, чтобы дать серверу возможность правильно распознать тип составного документа.

```
    echo "</head>"
    echo <body>
    echo "<h3> Список серверных процессов</h3>"
    echo "Дата:"
    date
    echo <p>
    echo <pre>
    ps -el
    echo "</pre>"
    echo "</body>"
    echo "</html>"
    echo "--NEXT"
    sleep 5
  done
```

Приведенный сценарий обновляет список процессов, идущих на машине сервера, каждые пять секунд. Это обновление продолжается до тех пор, пока броузер не прервет соединение, перейдя к другому документу.

Мы предлагаем этот пример сценария в качестве иллюстрации основной логики, стоящей за каждым генератором SP-документов. Попробуйте создать свое приложение со стороны сервера, используя более традиционный язык программирования, такой как Perl или C. Приложение будет работать более эффективно и сможет лучше обнаруживать разрыв связи клиент-сервер.

- Мобильный Интернет
- Фактор устройства
- Язык XHTML Basic
- Эффективный веб-дизайн мобильного содержимого

14

Мобильные устройства

Сейчас, когда большинство веб-разработчиков поднаторело в производстве захватывающих страниц для популярных броузеров, работающих на персональных компьютерах, перед ними встает нелегкая задача по созданию столь же эффектных страниц для вездесущих крошечных мобильных устройств. Появление мобильных приборов, способных подключаться ко Всемирной паутине, невозможно было предвидеть в начале 1990-х, когда появился язык HTML. Впрочем, от современных стандартов тоже мало проку. В этой главе мы обсудим существующие мобильные устройства, обладающие веб-функциональностью, а также проблемы, которые они ставят перед веб-дизайнерами, и подмножество XHTML, ориентированное специально на эти устройства. Мы также дадим несколько советов (и окажем моральную поддержку) тем, кто собирается создавать веб-страницы, выглядящие элегантно на дисплеях самых разных мобильных устройств.

14.1. Мобильный Интернет

Учитывая, что Всемирная паутина прочно вошла в нашу повседневную жизнь, мы должны считать нормальным желание пользователей иметь доступ к веб-содержимому, где бы они ни находились и в любой момент времени. Реагируя на спрос, производители выбрасывают на рынок огромное количество устройств и множество технических способов их подключения к Интернету. Хотя можно насчитать несколько сотен типов мобильных устройств, этот сектор рынка можно разбить на несколько основных категорий.

14.1.1. Устройства

Большинство современных мобильных устройств – сотовых телефонов и портативных компьютеров – оборудованы цифровыми дисплеями,

как правило, жидкокристаллическими, и встроенными процессорами. Что мешает установить на них броузер?

14.1.1.1. Мобильные телефоны

Программа-броузер хранится в основной оперативной памяти сотового телефона, и конечному пользователю не так-то просто расширить ее функциональность. Кроме того, как мы вскоре убедимся, у мобильного броузера гораздо меньше возможностей, чем у «настольного». Дополнительные функции доступны только очень упорному пользователю, готовому продраться сквозь ужасный интерфейс.

Операторы сотовой связи предоставляют выход в Интернет с использованием любой из имеющихся технологий, причем некоторые ограничивают доступ к содержимому или затрудняют обращение к данным за пределами их порталов.

14.1.1.2. Портативные компьютеры

Бесспорно, портативные компьютеры обеспечивают самый эффективный мобильный доступ в Интернет. На рынке доминируют устройства с операционной системой Palm OS от PalmSource, Inc. (бывшая Palm Computing, Inc.) или с системой Windows Mobile от Microsoft. Все производители оснащают свою продукцию высококачественными броузерами, обладающими функциональностью, характерной для настольных броузеров. Конечный пользователь может обновить версию броузера или расширить его возможности с относительной легкостью. Другие сетевые приложения, такие как почтовые клиенты или клиенты FTP, тоже доступны.

Большинство портативных компьютеров придерживается стандарта IEEE 802.11, известного как WiFi, при установлении беспроводного Ethernet-соединения с сетью оператора и, в конечном счете, с Интернетом. В некоторых портативных компьютерах применяется Bluetooth, альтернативная беспроводная технология, позволяющая соединиться с другим устройством, таким как сотовый телефон, ноутбук или сетевой терминал Bluetooth, и опять-таки получить в конечном счете выход во Всемирную паутину.

14.1.1.3. Конвергентные устройства

Конвергентные устройства – это попытка сочетать удобство мобильного телефона с гибкостью и мощностью портативного компьютера. Они устанавливают соединение с оператором сотовой связи, но могут также соблюдать и стандарт IEEE 802.11. На них можно установить большинство приложений, работающих на портативных компьютерах, и они в некоторой степени обеспечивают интеграцию пользовательского опыта на портативном компьютере с обычной функциональностью сотового телефона. Конвергентные устройства предлагаются в настоящее время компанией PalmSource (с предустановленной опе-

рационной системой Palm OS или Windows Mobile) и различными производителями сотовых телефонов (с операционной системой Windows Mobile). Конвергентные устройства предлагают определенный компромисс между мощью и большим экраном портативного компьютера и компактностью и простотой в работе, характерными для сотового телефона.

14.1.2. Мобильный доступ к сети

Пользователи могут выбрать не только тип устройства, но и план доступа к сети, позволяющий их мобильным устройствам подключаться к Интернету. Существуют сотни планов доступа, но, как это было и в случае с типами устройств, они могут быть разбиты на небольшое количество категорий.

14.1.2.1. Низкоскоростной доступ

Часто называемый доступом первого или второго поколения, низкоскоростной доступ к сотовой сети обеспечивает скорость обмена данными, близкую к скорости модема (56 Кбит/с, сеансовое подключение). Операторы обычно устанавливают плату за трафик, измеряемый в байтах (!), предоставляя ежемесячные пакеты из фиксированного количества байтов. Из-за низкой скорости и относительно высокой цены такой способ доступа годится лишь при нерегулярном выходе в Интернет для решения ограниченного круга задач. Постоянное подключение в этом случае нерационально, как по финансовым соображениям, так и в силу чрезвычайно низкой скорости обмена данными. Такой тип доступа присущ сотовым телефонам и некоторым конвергентным устройствам от самых разных производителей.

14.1.2.2. Высокоскоростной доступ

Последние успехи сотовой технологии позволяют операторам предлагать абонентам доступ со скоростью 1,5 мегабит в секунду. При таких скоростях пользователям обеспечен высококачественный доступ к видео- и аудиосодержимому Всемирной паутины. Оценив потенциал рынка, многие операторы предлагают высокоскоростной доступ по безлимитному плану за фиксированную цену. Продвигаемый под различными брендами (например, EDGE и EVDO), такой тип связи первоначально появился на сотовых телефонах, но быстро распространяется на портативные компьютеры, конвергентные устройства и некоторые модели ноутбуков.

14.1.2.3. WiFi

Многие портативные компьютеры и некоторые конвергентные устройства поддерживают беспроводной выход в сеть в соответствии со стандартом IEEE 802.11 и, следовательно, могут подключаться к совместимым беспроводным терминалам, столь широко распространившимся

за последнее время. Во многих населенных пунктах и студенческих городках существует беспроводной и, следовательно, мобильный доступ ко Всемирной паутине. В то время как большинство устройств поддерживает версию «b» этой технологии (то есть IEEE 802.11b), которая обеспечивает максимальную скорость 11 мегабит в секунду, некоторые последние модели придерживаются версии «g», при которой скорость возрастает до 54 мегабит в секунду. Стоимость доступа варьируется от бесплатного обмена данными (домашние и офисные сети и общественные терминалы) до нескольких десятков долларов в месяц у независимых провайдеров, таких как Cingular и T-Mobile.

Основываясь на приведенной классификации устройств и способов доступа, разработчик мобильного веб-содержимого должен принимать во внимание девять вариантов потенциальной пользовательской среды. К сожалению, в реальности ситуация гораздо сложнее, потому что у каждой конкретной модели и у каждого плана доступа имеются собственные ограничения и противопоказания. Как мы увидим в следующем разделе, на пользовательский опыт в мобильном Интернете влияют десятки факторов.

14.2. Фактор устройства

При создании содержимого для мобильных устройств разработчик должен держать в уме множество ограничений, накладываемых конструкцией. Если он пренебрегает каким-то одним из них, получившиеся веб-страницы будет трудно, а то и невозможно читать на мобильном устройстве. Мы полагаем, что профессиональный веб-дизайнер мобильной части Всемирной паутины должен всегда помнить об ограничениях, исходящих от броузера, способа ввода, сети и дисплея.

14.2.1. Ограничения броузера

Различия в броузерах представляют самую большую проблему для веб-дизайнера мобильного содержимого. «Втиснутые» в портативное устройство, мобильные броузеры не могут поддерживать многие теги, доступные в настольном броузере. Те теги, которые все-таки поддерживаются, имеют несогласованные реализации в разных мобильных устройствах. В результате веб-дизайнер должен тщательно отбирать теги, которыми он может воспользоваться, и часто вынужден жертвовать сложностью страниц ради их корректного отображения в большинстве устройств.

Помимо выбора тегов, существуют и дополнительные соображения. Мобильные броузеры могут не поддерживать сценарии, таблицы стилей, фреймы, внедренные объекты, слои, файлы cookie и прочие элементы содержимого страницы. Даже широко распространенные теги `<meta>` (например, `refresh`) могут оказаться без поддержки в каком-нибудь броузере. Вообще говоря, осторожному веб-дизайнеру следует из-

бегать подобных элементов при разработке содержимого. Хотя результат будет весьма примитивным, его гарантированно можно будет увидеть на многих устройствах. В отличие от настольных броузеров, которые пытаются хоть как-то вывести страницу, когда сталкиваются с неподдерживаемыми тегами, многие мобильные броузеры просто отказываются выводить сложное содержимое. Самая большая неприятность для разработчика мобильного веб-содержимого – это сообщение броузера «не удалось вывести страницу» в ответ на попытку пользователя просмотреть это содержимое.

Даже в том случае, когда разработчик содержимого придерживается только «безопасных» тегов, обсуждаемых далее в этой главе, результаты могут быть непредсказуемыми на некоторых броузерах. Стандарт на шрифты отсутствует, а некоторые мобильные броузеры предлагают один шрифт, одного размера и без таких излишеств, как полужирное начертание или курсив. Большинство устройств позволяет владельцу менять размер шрифта (например, человек, страдающий дальнозоркостью, может захотеть увеличить буквы, а другой пользователь, наоборот, выберет шрифт помельче, чтобы сообщение целиком умещалось на маленьком экране). Подобные скачки размера шрифта могут существенно повлиять на представление содержимого на экране устройства.

Некоторые мобильные броузеры вообще не способны обрабатывать графику, но эта проблема постепенно отходит на второй план по мере появления новых моделей мобильных устройств. Практически все мобильные броузеры с трудом справляются с крупными изображениями и могут проигнорировать или исказить их. Четкое определение того, какое изображение следует считать «крупным», отсутствует, и броузер принимает решение «на глаз», руководствуясь как линейными размерами изображения, так и длиной графического файла. Форматирование и обход изображения текстом по-разному реализованы в разных броузерах, к тому же мобильные броузеры часто игнорируют атрибут выравнивания изображения. Нет нужды добавлять, что нынешнее поколение устройств не может обработать внедренное видеоизображение, flash-анимацию и любой другой вид анимированного содержимого.

14.2.2. Ограничение способа ввода

У сотовых телефонов нет мыши – самого удобного устройства ввода при работе с броузером. Портативные компьютеры и конвергентные устройства не страдают от этого ограничения, позволяя прикасаться к экрану световым пером. Тем не менее значительная часть целевой аудитории мобильного содержимого будет пользоваться сотовыми телефонами. Таким образом, навигация внутри страницы представляет проблему на мобильных устройствах. Перенос фокуса для выбора ссылки будет в лучшем случае утомительным, особенно когда таких ссылок много.

Вводить текст на сотовом телефоне еще труднее. Большинство телефонов предлагает два режима ввода текста: либо каждое нажатие на кла-

вишу прокручивает буквы, которые на ней написаны, либо включается режим предсказания ввода, в котором пользователь набирает буквы, а телефон ищет в словаре подходящее слово, начинающееся с введенной последовательности. Первый способ утомителен, но более точен; второй помогает сэкономить время, но в большинстве случаев не проходит, так как ввод URL-адресов, самая востребованная операция, не подчиняется общепринятым правилам орфографии.

В обоих случаях ввод пунктуации затруднен, поскольку немногочисленные знаки препинания нередко вводятся одной клавишей, прокручивающей десяток других служебных символов.

14.2.3. Сетевые ограничения

Большинство разработчиков мобильного содержимого отдает себе отчет в ограничениях, накладываемых низкой скоростью передачи данных. Однако многие не задумываются над тем, сколько пользователи платят за каждый байт мобильного веб-содержимого. По иронии судьбы сегодня дизайнерам мобильного веб-содержимого приходится возвращаться к методам, господствовавшим в середине 1990-х годов, когда самые быстрые модемы обеспечивали скорость передачи 56 Кбит/с, а время соединения измерялось интернет-провайдером. Настолько ли ценно ваше содержимое, чтобы пользователи соглашались платить деньги всякий раз, когда им хочется просмотреть ваши веб-страницы?

Вдобавок к невысокой скорости передачи информации пользователи мобильных устройств сталкиваются с такими ограничениями, обусловленными технологией, которые вызвали бы возмущение у пользователей персональных компьютеров в сети. Некоторые URL-адреса блокируются, а данные от других пропускаются через прокси-серверы, которые преобразуют содержимое под мобильные устройства. В общем случае трудно предсказать, как конкретный оператор будет интерпретировать конкретную страницу. Самая надежная стратегия должна состоять в том, чтобы до предела упрощать содержимое, тем самым избегая странных преобразований ваших страниц.

И наконец, сетевая связь мобильных устройств непостоянна. Пользователь, посетивший ваш сайт, может просмотреть пару страниц, а затем внезапно потерять связь с сетью, выйдя из зоны охвата. Содержимое, предполагающее активную навигацию со страницы на страницу, может вызвать проблемы у пользователей, находящихся в пограничных областях зоны охвата их оператора.

14.2.4. Ограничения дисплея

Бесспорно, у всех мобильных устройств есть одна общая особенность: маленький, можно сказать, крошечный дисплей. Даже дисплеи конвергентных устройств, изготовители которых хвастаются, что их дисплеи самые продвинутые на рынке мобильных телефонов, не идут ни

в какое сравнение с окнами настольных броузеров. Многие мобильные устройства обеспечивают вертикальную прокрутку, когда данные оказываются за нижней границей дисплея, но лишь отдельные модели имеют прокрутку по горизонтали. Поэтому разработчик должен сознательно подстраиваться под маленький дисплей со строго фиксированной шириной страницы и весьма ограниченной длиной.

Ситуацию усложняет тот факт, что размеры дисплеев мобильных устройств имеют большой разброс. В отличие от настольных дисплеев, имеющих, как правило, размер 800×600 или 1024×768, дисплеи мобильных моделей могут иметь размер от 128×128 у некоторых сотовых телефонов до 320×480 и больше у портативных компьютеров. Встречаются сотовые телефоны с совсем странными дисплеями, например 176×220 или 122×96. Вообще говоря, вы не можете делать никаких предположений о размере дисплея пользователя фиксированного размера страниц. Впрочем, этот совет подходит для *любой веб-страницы в любой среде!*

14.3. Язык XHTML Basic

Отдавая себе отчет в неизбежных ограничениях мобильного серфинга по Всемирной паутине и пытаясь выдвинуть стандартную модель содержимого, предназначенного для мобильных устройств, консорциум W3C определил сокращенную версию XHTML специально для этого сектора рынка. Известная под названием *XHTML Basic*[1], эта версия языка содержит стандартный набор тегов, достаточный для создания эффективного содержимого под мобильные устройства и в то же время такой простой, что широкий набор мобильных броузеров сможет интерполировать его единообразно.

Не стоит обольщаться: из того, что стандарт нацелен на мобильные устройства, не следует, что мобильные броузеры будут его поддерживать. Нередко реализации препятствуют аппаратные ограничения.

14.3.1. Поддерживаемые теги

К языку XHTML Basic можно относиться как к нескольким группам тегов, которые сообща определяют минимальную, но работоспособную версию XHTML.

14.3.1.1. Базовое содержимое

Язык XHTML Basic был бы пустой затеей, если бы он не поддерживал четыре основных тега, определяющих документ: `<html>`, `<head>`, `<title>` и `<body>`. Вы не должны создавать документ без этих тегов, которые следует использовать для ограничения документа и его частей.

---

[1] Базовое подмножество языка XHTML. – *Примеч. науч .ред.*

Более сложная структура документа не допускается. Язык XHTML Basic явным образом исключает фреймы и слои из веб-содержимого, предназначенного для мобильных броузеров.

В теле документа язык XHTML Basic поддерживает базовый набор тегов структурирования текста, в том числе шесть заголовочных тегов (с <h1> по <h6>) и теги
, <p>, <pre> и <blockquote>. Этого достаточно для создания текста, организованного в абзацы и блоки, идентифицируемые заголовками различных уровней. В результате должен получиться документ, читаемый любым броузером.

Внутри текста язык XHTML Basic поддерживает все стилевые теги, включая <abbr>, <acronym>, <address>, <cite>, <code>, <dfn>, , <kbd>, <q>, <samp>, и <var>. Однако, если учесть малочисленность шрифтов в большинстве мобильных устройств, особенно в сотовых телефонах, мобильный броузер, возможно, не станет по-разному отображать эти теги. То же самое справедливо и в отношении заголовочных тегов, поскольку маловероятно, что мобильные телефоны смогут предложить шесть размеров шрифта для отображения всех шести тегов заголовков.

Ограничения, накладываемые бедностью шрифтов, заставили разработчиков стандарта XHTML Basic отказаться от физических стилевых тегов, определяющих полужирный шрифт и курсив. Не имея гарантии доступности этих стилей, неразумно поддерживать эквивалентные теги. Разные направления вывода текста тоже не поддерживаются; многие мобильные устройства и обычный-то текст выводят с трудом.

Язык XHTML Basic в большей степени ориентирован на использование таблиц стилей при выводе мобильного содержимого. Однако таблицы стилей исключены из контекста страницы, и сам тег <style> не поддерживается. Вместо этого язык XHTML Basic полагается на внешние таблицы стилей и для их поддержки определяет теги и , позволяющие разбивать содержимое на части и применять стили по мере необходимости. Атрибут class позволяет ассоциировать стиль с текстом.

Конечно, XHTML Basic поддерживает тег <a>, чтобы у вас была возможность ссылаться на другие документы.

14.3.1.2. Изображения, объекты и сценарии

В языке XHTML Basic все-таки имеется тег <img>, хотя пользоваться им следует с оглядкой. Вообще, рекомендуется вставлять изображения в документ лишь при наличии веских оснований, а для мобильного содержимого это справедливо вдвойне, потому что картинка может существенно замедлить загрузку документа и даже создать проблему для некоторых броузеров, если она достаточно велика. Далее в этой главе мы дадим несколько советов по эффективному использованию изображений в ваших документах.

XHTML Basic поддерживает и более общие способы внедрения объектов в мобильное содержимое, опирающиеся на теги <object> и <param>.

Как нередко случается с благими намерениями, этими тегами вымощена дорога в ад для мобильного содержимого. Реализация должна сильно зависеть от броузера и устройства, а мобильный рынок еще недостаточно устоялся, чтобы авторы веб-документов могли предполагать широкую поддержку внедренного содержимого, за исключением простейших изображений. Тем не менее, если вы нацеливаетесь на конкретную модель, обеспечивающую необходимую поддержку, то эти теги к вашим услугам.

XHTML Basic не поддерживает сценарии и обработку событий. Не поддерживается ни один атрибут обработки событий; отказано в поддержке и тегам `<script>` и `<noscript>`. Учитывая ограниченные вычислительные возможности типичного мобильного устройства, в этой функциональности нет смысла. Оставим динамические и управляемые сценариями страницы полноценным настольным броузерам.

14.3.1.3. Списки

С целью дальнейшего структурирования документа стандарт XHTML Basic поддерживает упорядоченные (`<ol>`) и неупорядоченные (`<ul>`) списки, а также списки определений (`<dfn>`). Естественно, поддерживаются теги `<li>`, `<dl>`, `<dd>` и `<dt>`. Эти списки реально помогают организовать и структурировать содержимое, особенно страницы, имеющие большое количество ссылок.

В частности, сочетание нумерованного списка ссылок с атрибутом `accesskey` в соответствующих тегах `<a>` облегчает пользователю мобильного броузера навигацию по страницам, сводя ее до нажатия на клавиши.

14.3.1.4. Формы

Интерактивность – важный момент веб-серфинга, поэтому XHTML Basic предоставляет поддержку форм, включая базовую структуру и элементы ввода: `<form>`, `<input>`, `<label>`, `<select>`, `<option>` и `<textarea>`. В спецификации языка не ограничивается диапазон элементов формы, разрешенных к употреблению, но вы должны помнить, что некоторые мобильные устройства не в состоянии отобразить меню из большого количества пунктов.

Единственными элементами формы, явно запрещенными в стандарте XHTML Basic, являются элементы выгрузки файла и изображения. По иронии судьбы именно они могли бы стать привлекательными для владельцев сотовых телефонов со встроенными фотокамерами, потому что эти элементы позволили бы пользователям выгружать картинки на веб-сервер.

14.3.1.5. Таблицы

Веб-дизайнеры традиционно пользуются таблицами для структурирования содержимого на экране. Хотя вы можете добиться аналогичных эффектов в мобильном броузере, будьте бдительны. Подобная практи-

ка неявно осуждается в стандарте XHTML, который предполагает, что таблицы будут применяться для вывода табличного содержимого, а не с целью компоновки.

XHTML Basic поддерживает только основные табличные теги: <table>, <tr>, <td>, <th> и <caption>. Более сложные элементы, такие как определение интервалов в столбцах или вложенные таблицы, явным образом запрещены в стандарте XHTML Basic. Сложные таблицы могут быть выведены некорректно, и узкий дисплей запросто исказит то, что вы задумали. Тонкие эффекты, такие как индивидуальные поля в ячейках или варьирующаяся толщина рамок, почти наверняка будут по-разному обрабатываться разными броузерами, так что ради большей доступности содержимого от них лучше отказаться.

14.3.1.6. Заголовок документа

XHTML Basic поддерживает некоторые теги, обычно встречающиеся в заголовке (<head>) документа, а именно <meta>, <link> и <base>. Первоначально теги <link> и <base> были задуманы для обеспечения ссылок на таблицы стилей из мобильного содержимого. Будьте осторожны с тегом <meta> – мобильные броузеры поддерживают не все его атрибуты.

14.3.2. Намерения и результат

Хотя стандарт XHTML Basic определяет специальный набор тегов, которые должны работать на любом мобильном броузере, соблюдающем этот стандарт, не стоит впадать в заблуждение, что вы сможете использовать элементы языка XHTML Basic по максимуму во время разработки содержимого. Рынок мобильных устройств еще не устоялся, и броузеры еще слишком «сырые», чтобы поддерживать все варианты тегов этого языка. Дизайнеры старшего поколения вспомнят, как они работали в середине 1990-х, когда создание содержимого одновременно под Netscape Navigator и ранние версии Internet Explorer было, в лучшем случае, трудной задачей, а в большинстве ситуаций – пыткой. Оба броузера пытались реализовать тогдашний стандарт HTML, но настолько по-разному, что разработчики содержимого не могли рассчитывать на легкую жизнь.

Цель стандарта на мобильное веб-содержимое состоит в том, чтобы предоставить разработчику небольшой набор тегов, воспринимаемых широким спектром мобильных устройств, от сотовых телефонов до портативных компьютеров. Консорциум W3C даже приводит список приборов (в который входят холодильники и стиральные машины с микрокомпьютерами внутри), которые могут быть целью взаимодействия через веб-страницы. Хорошие дизайнеры постараются не отступать от задуманного, разумно применяя теги, не злоупотребляя трюками и аккуратно кодируя каждую страницу. Получившееся содержимое будет красиво выглядеть на всех устройствах, и дизайнеры станут добрее и счастливее.

14.4. Эффективный веб–дизайн мобильного содержимого

Нет никакого секрета в том, как следует создавать эффективное мобильное содержимое. Советы, которые мы давали на протяжении всей книги, справедливы в отношении мобильных устройств в той же степени, что и в отношении их настольных собратьев: изучите целевую аудиторию, ее потребности и ее способ выхода в Интернет. И все-таки опыт мобильного веб-серфинга отличается от «настольного» так значительно, что мы поступили бы недобросовестно, если бы не предложили несколько специальных советов по улучшению качества вашего веб-содержимого.

В условиях бурного роста популярности мобильного Интернета нет недостатка в советах по веб-дизайну мобильного содержимого, как удачных, так и не очень. В последующих разделах мы внесем свой вклад в эту копилку, основанный на нашем личном опыте и посещении многочисленных мобильных сайтов.

14.4.1. Изучите своего пользователя

Люди прибегают к помощи мобильного броузера совсем по иным причинам, чем при выходе в Интернет с настольного компьютера или ноутбука. В большинстве случаев они не стремятся скачать материал для вдумчивого чтения и не пытаются получить очередной кредит на приобретение недвижимости. Их интересуют сведения, объем которых невелик и получить которые можно довольно быстро: заголовки новостей, прогноз погоды, расписание авиарейсов, результаты спортивных состязаний и т. д. Броузеры стали интерфейсом ко многим другим сетевым устройствам, так что мобильный броузер может иметь коммерческое и промышленное применение. Мобильные пользователи не стремятся загрузить большие картинки или полнометражные фильмы. Они, скорее всего, захотят выяснить, как доехать до нужного места, узнать цены на товары, заказать билеты в реальный кинотеатр, настроить работу какого-нибудь хитроумного устройства, да мало ли что еще...

Помните об этом, когда будете разрабатывать содержимое. Какую информацию вы предоставляете пользователям? Почему они захотят ее получить через мобильный броузер? Настолько ли важны ваши данные, что пользователи согласятся просмотреть их на ходу, держа телефон в руке, параллельно делая какое-то другое дело? Не пытайтесь втиснуть свой сайт в мобильный формат только для того, чтобы потом хвастаться на эту тему. Отбирайте и распространяйте только то содержимое, которое ценно для людей, находящихся в «мобильном» состоянии. Почти в любом случае тщательное редактирование содержимого является первым шагом в превращении его в мобильное.

Когда вы поняли, кто будет просматривать и использовать ваше содержимое с помощью мобильного броузера, подумайте и о том, в каком

окружении они будут одновременно с кем-то разговаривать. Ваше содержимое должно будет «продраться» сквозь отвлекающие факторы, обеспечить пользователя нужной информацией и больше не удерживать его внимание. Необходимо, чтобы оно было доступно и понятно и не требовало усилий при навигации. Из-за технологических ограничений ваше содержимое, вероятнее всего, будет поступать в броузер медленно, так не ухудшайте ситуацию, заставляя пользователя совершать дополнительные действия. Здесь ключевыми словами являются «быстрота» и «мобильность».

14.4.2. Ссылки и навигация

Излишне большой размер и плохо продуманная навигация являются самыми серьезными недостатками большинства мобильных веб-страниц.

Многие страницы содержат важную информацию, но навигация к ней настолько затруднена, что пользователи пытаются получить ее по другим каналам. Создается впечатление, что разработчики, построившие сложные навигационные структуры для настольных броузеров, чувствуют себя обязанными повторно задействовать тот же код для крошечных мобильных броузеров. Кроме того, становится очевидно, что сами эти разработчики не пробовали пользоваться своим содержимым в мобильной среде. В противном случае, они непременно упростили бы свои страницы.

Перемещаться по странице в мобильном броузере гораздо труднее, чем в настольном. Прокрутка представляет отдельную проблему, так как для нее приходится многократно нажимать на маленькие кнопки. Перенос фокуса со ссылки на ссылку ничуть не проще, причем нажимать нужно уже на другие маленькие кнопки. Вы должны заботиться о пользователях: разрабатывайте навигацию таким образом, чтобы максимально исключить прокрутку и перенос фокуса. Если вы создаете традиционные ссылки «в начало», «далее» и «назад», размещайте их на самом верху, где они заметнее и доступнее. Не заставляйте пользователя прокручивать всю страницу в поисках элементов навигации. Применяйте небольшое количество навигационных элементов, которые ясно показывают, куда они приведут.

Некоторые броузеры поддерживают атрибут `accesskey`, позволяя вам ассоциировать клавишу со ссылкой или элементом формы. Нажатие на эту клавишу равносильно выбору ссылки или переносу фокуса на соответствующий элемент формы. Если вы организуете ссылки в нумерованный список и добавите атрибуты `accesskey`, пользователи получат возможность за одно нажатие перейти по нужной ссылке, не перебирая их все. Например:

```
Kumquat Resources:
<ol>
   <li><a accesskey="1" href="growers.html">Growers</li>
   <li><a accesskey="2" href="vendors.html">Vendors</li>
```

```
    <li><a accesskey="3" href="fanclubs.html">Fan Clubs</li>
</ol>
```

Здесь пользователю достаточно нажать на клавишу 1, 2 или 3, чтобы перейти по интересующей его ссылке. Если то же самое реализовать в виде ряда обычных гиперссылок, то количество нажатий на клавиши увеличится. Экономичные решения, аналогичные этому, значительно влияют на пользовательский опыт в целом.

Вообще говоря, переход по ссылке в мобильном броузере – дорогое удовольствие, в смысле времени и денег. Четко идентифицируйте свои ссылки, чтобы пользователь не сомневался, куда они приведут и что он от этого получит. Безымянные ссылки «щелкните здесь» раздражают. Пользователи не желают заниматься исследованием вашего сайта; им хочется быстро получить нужную информацию. Если ссылка вызовет загрузку большого объема данных, например картинки, предупредите об этом пользователя, чтобы он имел возможность отказаться.

Особенно следует избегать ссылок, основанных на изображениях, за исключением тех случаев, когда изображения очень малы. Многие мобильные броузеры позволяют пользователям выполнять навигацию по странице и выбирать ссылку до того, как страница будет загружена целиком. Помните, что страница загружается раньше, чем сопутствующие ей файлы, например изображения. Текстовые ссылки появляются практически сразу, а ссылки, основанные на изображениях, заставляют пользователей ждать. Как бы то ни было, вы не должны базировать навигацию на изображениях, потому что области такой навигационной карты не будут такими очевидными и легко доступными, как в настольном броузере при наличии мыши.

Не делайте ссылки на другие окна с помощью атрибута `target`. Многие мобильные броузеры просто не в состоянии отобразить несколько окон, и они отбросят содержимое страницы, где находится ссылка. Пользователи запутаются, а содержимое не будет представлено так, как вы задумали.

14.4.3. Формы

Формы представляют особую трудность для разработчика мобильного содержимого. Чтобы сделать пользовательский опыт интерактивным, вы должны включать формы в свои страницы, позволяя пользователям вводить запросы и параметры для конкретизации содержимого, получаемого с сайта. К сожалению, большинство форм плохо переносится на мобильные броузеры, в которых ввод текста и выбор поля затруднены и чреваты ошибками. Пользователи мобильных устройств хотят быстро получить конкретную информацию. Разрабатывайте формы так, чтобы ими было удобно пользоваться, и клиенты будут снова и снова посещать ваш сайт.

Как всегда, хорошее содержимое начинается с хорошего редактирования, и формы не являются исключением. Добейтесь того, чтобы фор-

мы были лаконичными и уместными. Четко и недвусмысленно пометьте различные поля и элементы, чтобы пользователи поняли, чего вы от них ждете. Там, где это возможно, обеспечьте значения полей по умолчанию, чтобы пользователям не приходилось «трогать» каждый элемент перед отправкой формы. Это особенно важно, когда пользователь вынужден возвращаться к форме, чтобы исправить ошибки. Недопустимо заставлять пользователя заново вводить все данные каждый раз, когда он заполняет форму.

Ввод текста – отдельная тема, особенно если речь идет о вводе паролей и других замаскированных сведений. Не все мобильные устройства корректно обрабатывают замаскированный текст, а ввод замаскированного пароля в режиме многократного нажатия на кнопку превращается в огромную проблему. В некоторых мобильных броузерах ввод текста выполняется в отдельном всплывающем окне, и пользователь проходит через несколько этапов выбора и подтверждения, прежде чем ему удается ввести в поле одно текстовое значение.

Будьте проще. Формы со многими элементами ввода плохо представляются на маленьком экране мобильного устройства. Когда пользователи прокручивают форму, чтобы заполнить ее, они не видят предыдущие элементы, а также те, что им еще предстоит заполнить. Такая потеря ориентации затрудняет правильное обращение с формами в мобильном окружении. Имеет смысл разбивать большие формы на несколько маленьких и позволять пользователям вводить информацию поэтапно. Если вы предпримете такой подход, обязательно проверяйте корректность данных по мере их получения. Не накапливайте шесть экранов с пользовательской информацией лишь для того, чтобы потом заставить пользователя вернуться к первому экрану и исправить ошибку.

14.4.4. Компоновка и представление страницы

Мобильная часть Всемирной паутины – не место для оригинальных компоновок и блестящих презентаций. Ограниченный набор тегов языка XHTML Basic подтверждает это, и разумный дизайнер мобильного содержимого не будет пытаться выходить за установленные рамки. Хотя переход от богатого функциональными возможностями настольного броузера к минималистской мобильной среде может оказаться непростым, разработчик содержимого должен помнить, что его цель – информировать пользователя, а не произвести впечатление на других разработчиков.

14.4.4.1. Таблицы стилей

Впрочем, для тех, кто хочет создать привлекательное мобильное содержимое, не все потеряно. Хорошие дизайнеры пользуются таблицами стилей для отделения содержимого от атрибутов его представления. Такой подход, кроме прочего, облегчает сопровождение единственного источника содержимого, внешний вид которого управляется

разными таблицами стилей, в зависимости от целевого устройства. Поскольку применение встроенных стилей не рекомендуется и не поддерживается стандартом для мобильного содержимого, вы должны использовать внешние ссылки на таблицы стилей мобильного содержимого. Например:

```
<link rel="stylesheet" type="text/css" media="handheld" href="sheet.css">
```

В этой ссылке атрибут `media` является ключевым. Он гарантирует, что эта таблица стилей будет применяться к содержимому, просматриваемому с помощью мобильного устройства, и игнорироваться в противном случае. Вы должны будете пользоваться маленькими таблицами стилей, так как они увеличивают время загрузки содержимого при медленном мобильном соединении.

Кроме того, вы должны отдавать себе отчет в том, что не все мобильные броузеры поддерживают таблицы стилей, а у броузеров, поддерживающих их, могут возникнуть проблемы при загрузке таблицы стилей из-за неустойчивой связи. Протестируйте содержимое без таблицы стилей и убедитесь, что оно удовлетворительно выглядит и в этом случае.

14.4.4.2. Шрифты

Шрифты – еще одна «головная боль» разработчиков мобильного содержимого. В отличие от настольных броузеров с сотнями шрифтов в разных вариациях, мобильные броузеры часто имеют не больше одного. Он, как правило, представлен в одном или двух размерах и не имеет полужирной или курсивной версии. Реальность такова, что крошечные дисплеи мобильных устройств не в состоянии отображать сложные шрифты, и производители не реализуют шрифты, которые все равно невозможно будет разобрать.

Чтобы справиться со всеми этими ограничениями, реализуйте разные размеры шрифтов с помощью заголовочных тегов (если они доступны), а не с помощью относительного или абсолютного масштабирования шрифтов. Большинство мобильных броузеров проводит различие между тегами `<h1>`, `<h2>` и `<h3>`, так что вы можете пользоваться ими для вывода названий страниц, заголовков разделов и прочих структурирующих элементов. Помните, что многие мобильные броузеры интерпретируют все теги выделения (полужирный шрифт, курсив, подчеркивание) одинаково, то есть выводят выделенный текст полужирным шрифтом. Если вы попытаетесь по-разному выделять текст на одной странице, пользователи, весьма вероятно, увидят только один тип выделения.

14.4.4.3. Поля и интервалы

Узкий дисплей мобильного устройства ограничивает возможности компоновки страниц. Избегайте полей, – это нерациональная трата ценного места на экране. То же самое можно сказать про вложенные списки: глубокая вложенность отодвинет содержимое страницы вправо, превратив его в колонку слов у правого края дисплея.

Абсолютное указание интервалов и управление разметкой – это очередная проблема на мобильных устройствах. Трюки, типичные для настольного содержимого, такие как изображения размером в один пиксел и прозрачные GIF-изображения, не пройдут на мобильном устройстве. Стандартные элементы HTML, такие как фреймы и слои, обычно не поддерживаются, а ограниченная поддержка страниц затрудняет попытки основать компоновку на таблицах. Вообще говоря, вам следует рассматривать мобильное содержимое как вертикальный поток, и пусть броузер форматирует его по своему усмотрению, без вашего вмешательства.

Наконец, отдавайте себе отчет в том, что может произойти адаптация вашего содержимого. Это автоматическое преобразование данных в форму, наиболее удобную клиенту. Оно может иметь дело на сервере, когда в клиенте, запрашивающем содержимое, распознается мобильное устройство. Оно нередко выполняется оператором сети, причем со страницы удаляются изображения и некоторые теги, что позволяет уменьшить ее объем. Кроме того, неявную адаптацию может произвести мобильный броузер, игнорирующий не поддерживаемые им теги и атрибуты.

Вы не в состоянии бороться с адаптацией. Самое лучшее, что вы можете сделать, – это постараться избежать ее, создавая простое содержимое, которое не подвергнется адаптации ни на каком уровне. Короче говоря, чем проще ваше содержимое, тем больше вероятность того, что на экране мобильного устройства оно появится в том виде, как вы задумали.

14.4.5. Изображения

На заре существования Всемирной паутины изображения усложняли жизнь пользователям. Сеансовое подключение через модем не позволяло быстро загрузить картинку, пользователи испытывали разочарование, а страницы оставались непосещенными. Веб-серферы со стажем помнят времена, когда за ссылкой в скобках указывался объем загружаемого содержимого. Работая с модемом 28,8 Кбит/с и выбрав ссылку, за которой стоит «(132K)», вы могли выпить чашечку кофе за то время, пока загружалось изображение.

Поразительные успехи технологии сделали нормой веб-страницы, насыщенные графикой. Дизайнеры привыкли украшать страницы крупными изображениями. К сожалению, подобный дизайн не подходит для медленных мобильных устройств, а также и для быстрых устройств, но с небольшой памятью. В результате изображения, особенно крупные, оказываются роскошью в мобильном окружении.

Сказанное не означает, что изображения следует исключить из мобильного содержимого. Просто употреблять их надо экономно. Маленький логотип уместен на страницах сайта, а навигационные пиктограммы определенно упростят просмотр ваших страниц. Если вам приходится доставлять крупное изображение, сделайте на него отдельную ссылку и предупредите пользователя о размере объекта. Тогда пользователи смогут оценить стоимость щелчка по ссылке.

Поставляя крупное изображение, пользуйтесь широко распространенными форматами, такими как GIF89a и JPEG. Мы не знаем ни одного мобильного броузера, который был бы не в состоянии справиться с этими давно установившимися форматами файлов. Помня о размере дисплеев мобильных устройств, используйте изображения, которые на них поместятся. Ни в коем случае не отправляйте огромное изображение в надежде на то, что броузер его масштабирует до размеров дисплея. Просто непорядочно отправлять изображение 1024×768 по медленному соединению, забивать до предела память устройства и заставлять это устройство в конечном счете масштабировать картинку под экран. Чтобы помочь броузеру и сообщить ему размеры будущей картинки, всегда ставьте атрибуты `height` и `width` в теге `<img>`.

14.4.6. Общие советы

В заключение дадим еще один совет: чем меньше, тем лучше. Мобильная часть Всемирной паутины – это не то место, где следует демонстрировать потрясающее мастерство компоновки страниц или фантастическую библиотеку изображений. Здесь главное – быстрая доставка важной информации, нужной для конкретных целей в конкретный момент времени.

Придерживаясь этого минималистского подхода, учтите следующие рекомендации консорциума W3C относительно разработки мобильных веб-страниц:

- Разрабатывайте все страницы в расчете на дисплей шириной 120 пикселов. Хотя многие современные модели имеют более широкие дисплеи, нацеленность на этот небольшой размер обеспечит хорошее представление страницы на любом устройстве, как старом, так и новом.
- Используйте изображения в формате GIF89a и JPEG. Как было сказано выше, это гарантирует показ картинки почти на любом мобильном устройстве.
- Не отклоняйтесь от стандарта XHTML Basic. Применение тегов, не поддерживаемых этим стандартом, почти наверняка приведет к ошибкам в большинстве мобильных устройств.
- Пользуйтесь таблицами стилей для отделения содержимого от представления. Стандарт XHTML Basic определяет оптимальный метод интегрирования таблиц стилей и мобильного содержимого.
- Объем страницы не должен превышать 20 Кбайт. Сюда входят базовое содержимое страницы, все ассоциированные с ним таблицы стилей и все подключаемые изображения.

Следуя этим рекомендациям, тщательно редактируя содержимое и структурируя страницы с целью упростить навигацию, вы создадите отличное мобильное содержимое и получите положительные отзывы пользователей.

- Языки и метаязыки
- Документы и DTD
- Как читать XML DTD
- Грамматика элементов
- Атрибуты элементов
- Условные разделы
- Построение XML DTD
- Использование XML

15

XML

Выросший на вольных просторах Всемирной Сети, как мустанг на равнинах Дикого Запада, HTML лишь в общих чертах соответствует теоретическим нормам проектирования и реализации электронных документов. HTML возник из необходимости упорядоченного соединения текста, графики и другой информации в теле одного электронного документа, который можно было бы передавать по глобальной сети Интернет. В начале бума Интернета спрос на все лучшие и лучшие броузеры и серверы со стороны ненасытных полчищ новых пользователей, жаждавших все новых и еще круче оформленных страничек, не оставлял времени для размышлений о правилах и стандартах.

Без общих принципов HTML превратился бы в Вавилонское столпотворение. Что почти и случилось во время броузерных войн второй половины 90-х годов. Но хаос не может быть основой индустрии, в которую вложены триллионы долларов. И хотя специалистам из W3C удалось обуздать неукротимый HTML стандартом 4.0, он пока еще недостаточно цивилизован, чтобы войти в благородное сообщество языков разметки.

Язык стандарта HTML 4.01 описан как подмножество SGML (Standardized Generalized Markup Language, стандартизованного обобщенного языка разметки). SGML более чем достаточен для формализации HTML, но слишком сложен, чтобы быть общепринятым средством расширения и усовершенствования HTML. Вместо него W3C разработал новый стандарт – расширяемый язык разметки (Extensible Markup Language), или XML. Он основан на простейших элементах SGML, гибче и вполне подходит как база для создания и развития новых языков разметки. Благодаря ему HTML возродился уже как XHTML.

В этой главе мы раскроем основы XML, включая то, как его читать, как создавать простые XML DTD (Document Type Definitions, определения типа документа) и как с помощью XML повысить качество работы с Интернетом. В следующей главе мы исследуем глубины XHTML.

Нет необходимости понимать все в XML, чтобы писать на XHTML. Мы думаем, что это было бы небесполезно, но если вы спешите начать работать, без опасений переходите к следующей главе. Однако, быть может, вы все же захотите предварительно познакомиться с многообещающими примерами использования XML, описанными в конце этой главы, начиная с раздела 15.8.

В этой главе дается только обзор XML. Наша цель здесь – познакомить вас с XML и разогреть ваш аппетит. Для полного овладения XML обратитесь к книгам:[1] Erik T. Ray «*Learning XML*» и W. Scott Means and Elliotte Rusty Harold «*XML in a Nutshell*». Обе книги выпущены издательством O'Reilly.

15.1. Языки и метаязыки

Язык состоит из символов, которые мы некоторым образом объединяем, чтобы выразить свои мысли и передать информацию так, чтобы она была понятна окружающим. В английском языке, например, есть правила грамматики, которые определяют, как соединять его символы (то есть слова), чтобы получались предложения, абзацы и, наконец, книги, вроде той, что вы держите в руках. Если вы знаете слова и понимаете правила языка, вы уже можете читать книгу, даже если не понимаете ее содержания.

Важное отличие между естественными языками и языками компьютерными состоит в том, что естественные языки описывают себя сами. Мы пользуемся английскими предложениями и абзацами, чтобы определить, как правильно составлять английские предложения и абзацы. Наш мозг – это чудесная машина, которая без труда понимает, что для описания языка годится он сам. Компьютерные языки, однако, не так богаты, и компьютеры не настолько умны, чтобы возможно было с легкостью описать компьютерный язык тем же самым языком. Вместо этого можно определить другой язык – *метаязык,* который описывал бы правила и символы компьютерного языка.

Разработчики программного обеспечения используют метаязык[2] для определения правил описания языка и затем в соответствии с этими правилами создают новые языки. Метаязык используется также раз-

---

[1] Натанья Питц-Моултис, Черил Кирк «XML», СПб.: БХВ-Петербург, 2001. – *Примеч. науч. ред.*

[2] Метаязыки уже давно широко используются в мире программирования. Язык программирования C, например, содержит набор правил и обозначений, определяемых одним из метаязыков, включая *yacc*. Программисты пользуются *yacc* при создании компиляторов, которые, в свою очередь, преобразуют тексты исходных файлов в машинный код. Отсюда и его название: Yet Another Compiler Compiler (еще один компилятор компиляторов). Единственное назначение *yacc* в том, чтобы помогать разработчикам при создании новых языков программирования.

работчиками автоматизированных программ-посредников, которые отображают (или каким-то другим образом обрабатывают) содержимое документов, созданных с помощью этого языка.

XML – это метаязык, созданный W3C и применяемый программистами для описания разных языков разметки, таких как, например, XHTML. Создатели броузеров опираются на правила метаязыка XML при создании автоматических обработчиков, читающих описания элементов языка XHTML и, в конечном счете, отображающих или иначе обрабатывающих документы на языке XHTML.

Зачем надо ломать голову над метаязыком разметки? Затем, что, выражаясь образно, W3C хочет дать нам удочку и научить ловить рыбу себе на пропитание. XML дает нам способ единообразного описания языков разметки, специально созданных, чтобы удовлетворить самые разнообразные нужды, и призванных заменить беспорядочное множество расширений HTML. Математикам нужно средство отображения множества специальных значков и формул; композиторам надо представить на обозрение свои партитуры; промышленники хотят руководить заводами при помощи сетевых технологий; предприятия желают через Интернет получать заказы, а врачи – обмениваться опытом. Все эти группы пользователей нуждаются в приемлемом, гибком способе отображения этих разнообразных видов информации, чтобы индустрия программного обеспечения могла создавать программы, обрабатывающие и отображающие эти разнообразные документы.

Все это может XML. Для всех предметных областей – торговли, управления производством, науки – можно определить язык разметки, который удовлетворяет конкретным требованиям, предъявляемым к обмену информацией и обработке данных. Программисты могут написать XML-совместимые обработчики-анализаторы (парсеры, parsers), которые читают описания новых языков и позволяют серверу соответствующим образом обрабатывать документы, написанные на этих языках.

15.1.1. Информация и ее отображение

Неограниченная свободой при описании новых языков разметки с помощью XML порождает проблему отображения информации, записанной на новом языке. Когда вы пишете на HTML, броузер знает, что делать с тегом `<h1>`, поскольку этот тег описан в HTML DTD.

При помощи XML можно, к примеру, создать новый DTD для описания рецептов.[1] Это поможет собрать и привести в порядок все многочисленные рецепты, скопившиеся в ящике кухонного стола. Пользуясь специальными тегами `<ингредиент>` и `<количество>`, очень просто бу-

[1] Альтернативой DTD являются схемы XML. Они предлагают функциональные возможности, связанные с вводом данных, и ориентированы, скорее, на программу, чем на документ. Подробности можно прочитать в книге *XML Schema* (автор Eric van der Vlist, издательство O'Reilly).

дет записывать рецепты и понимать их. Броузер, однако, не будет знать, что делать с этими новыми тегами, пока вы не присоедините к основному тексту таблицу стилей, задающую способ их обработки. Без таблицы стилей XML-совместимый броузер обработает эти теги по своему разумению, совсем не так, как того заслуживают ваши оригинальные рецепты.

Даже при использовании таблиц стилей сохраняются ограничения на представление информации, основанной на XML. Допустим, вы хотите создать нечто более амбициозное, например DTD для музыкальной нотации или схемы микрочипа. Хотя описать подобные типы данных не представляет особой сложности, графическое отображение такой информации находится за пределами возможностей известных нам таблиц стилей. Потребуются специальные средства для надлежащего отображения такой богатой графики.

Тем не менее запись рецептов в формате DTD весьма полезна при их хранении и передаче. Как мы увидим далее в этой главе, XML годится не только для создания языков разметки, предназначенных для броузеров. Он является многообещающим средством управления информационными потоками, так что описанные вами блюда будут сохранены для будущих поколений. Имейте только в виду, что в дополнение к описанию нового XML-языка в некоем DTD вам наверняка придется позаботиться и о создании соответствующей таблицы стилей.[1]

15.1.2. Немного истории

Чтобы покончить с оставшимися «зачем» и «почему», полезно узнать, откуда пошли все эти языки разметки.

Вначале был SGML, стандартизованный обобщенный язык разметки. SGML претендовал на роль единственного метаязыка, на основе которого должны были бы создаваться все языки разметки. На SGML можно описать все – от иероглифов до HTML, – так что потребность в каком-либо другом метаязыке вроде бы отпадает.

Проблема в том, что SGML настолько всеобъемлющ и широк, что простой смертный не может с ним управиться. Эффективное использование SGML требует очень дорогих и сложных инструментов, совершенно недоступных человеку, который просто хочет быстро состряпать HTML-страничку. В результате создавались другие языки разметки, с гораздо меньшими возможностями, но более простые в использовании. Сами стандарты HTML первоначально определялись подмножеством SGML, не содержавшим его наиболее продвинутых средств. DTD использует это подмножество SGML для определения стандарта HTML 4.01.

---

[1] На самом деле можно создавать XML-документы, используя только таблицы стилей. Создание DTD настоятельно рекомендуется, но не является обязательным. См. подробности на *http:// www.w3c.org/TR/xml-stylesheet*.

Учитывая все возрастающую потребность в новых языках разметки и то, что SGML слишком громоздок для описания HTML-подобных языков, W3C (World Wide Web Consortium) создал XML. XML – метаязык разметки, включающий в себя избранные элементы SGML и предназначенный для определения HTML-подобных языков. В нем отсутствуют элементы SGML, не применимые к языкам типа HTML, а другие элементы упрощены, чтобы облегчить их понимание и использование.

XML, лежащий между SGML и HTML, служит удобным средством для описания широкого спектра языков разметки. Важность XML будет возрастать по мере того, как WWW выходит за пределы броузеров и проникает в область прямого обмена данными между людьми, компьютерами и другими различными, часто несовместимыми системами. Небольшое число людей займется созданием новых языков разметки с помощью XML, и очень многим понадобится уметь понимать XML DTD для того, чтобы использовать все эти новые языки.

15.2. Документы и DTD

Для большей точности поясним, что буквосочетание «XML» употребляется в несколько различных смыслах. «XML-документ» – это документ, содержимое которого соответствует правилам языка разметки, определенного по стандарту XML. «XML Document Type Definition» (XML DTD) – это совокупность правил, официально именуемых объявлениями элементов и сущностей, которые определяют язык разметки XML, то есть как расположены теги в корректном («действительном») XML-документе. Все оказывается еще более запутанным оттого, что объявления элементов и сущностей могут встречаться как в XML DTD, так и в самом теле XML-документа.

XML-документ содержит текст, состоящий из собственно содержимого, разметки в виде тегов и XML-объявлений. Таким образом,

```
<blah>harrumph</blah>
```

является строкой *корректного* (*well-formed*) XML-документа. Корректные XML-документы соблюдают определенные правила, такие как требование наличия закрывающего тега для каждого открывающего тега. Эти правила в отношении XHTML представлены в главе 16.

Чтобы считаться *действительным* (*valid*), а действительный XML-документ соответствует DTD, XML-документ должен содержать необходимое количество объявлений XML (или ссылок на них), описывающих предназначение тегов. Эти объявления могут включаться прямо в XML-документ или же хранятся отдельно в XML DTD. Если существует XML DTD, определяющее тег `<blah>`, наш корректный XML-документ является действительным, если предварить его тегом `<!DOCTYPE>`, объясняющим, где можно найти соответствующее DTD:

```
<?xml version="1.0"?>
<!DOCTYPE blah SYSTEM "blah.dtd">
```

```
<blah>harrumph</blah>
```

Представленный фрагмент начинается с необязательной директивы `<?xml>`, объявляющей используемую документом версию XML. За ней следует директива `<!DOCTYPE>`, указывающая на DTD, которое будет использоваться какой-нибудь автоматизированной системой, например броузером, для обработки содержимого документа и, возможно, вывода его на экран. В нашем случае DTD `blah.dtd` должно быть доступно броузеру[1], чтобы он мог определить, является ли тег `<blah>` действительным в документе.

В XML DTD содержатся только объявления элементов и сущностей XML, тогда как XML-документы могут содержать как объявления элементов XML, так и обычное содержимое, использующее объявленные элементы для создания документа. Такое смешение содержательного текста и правил его прочтения в одном документе вполне приемлемо для компьютерной обработки, но может запутать тех, кто изучает XML. Поэтому в этой главе мы сосредоточим внимание на тех возможностях объявлений элементов и сущностей XML, с помощью которых вы сможете определять новые теги и типы документов. Другими словами, мы займемся только особенностями DTD. То, что относится к содержательной части XML-документов, буквально повторяет правила и требования, известные вам по опыту создания HTML-страниц.

15.3. Как читать XML DTD

Чтобы использовать язык разметки, определенный через XML, необходимо уметь читать и понимать элементы и сущности, встречающиеся в его XML DTD. Но не пугайтесь: хотя XML DTD многословны, полны непонятных знаков и предназначены в первую очередь для компьютерной обработки, но стоит лишь снять с них всю эту синтаксическую шелуху, они легко откроются вашему пониманию. Помните, что у вашего мозга больше способностей к языкам, чем у любого компьютера.

Как мы уже говорили, XML DTD – это набор объявлений элементов и сущностей XML, а также комментариев. Сущности – это пары имя/значение, которые делают DTD более легким для чтения и понимания, тогда как элементы – это действительные теги разметки, определяемые в DTD, как, например, теги `<p>` и `<h1>` в HTML. DTD определяют также допустимое содержимое и грамматические правила для всех тегов языка. Наряду с объявлениями элементов вы найдете объявления атрибутов, используемых с тегами, обозначенными в объявлениях элементов.

---

[1] Мы здесь говорим о броузерах, поскольку именно они обычно применяются для обработки XML-документов. Описание стандарта XML использует более общее понятие – «приложение-обработчик», потому что в некоторых случаях XML-документ обрабатывается не обыкновенным броузером, а какой-то другой программой, умеющей интерпретировать XML-документы.

Четкого порядка нет, хотя заботливые авторы DTD упорядочивают объявления так, чтобы людям было легче находить и понимать их. Уважаемые создатели DTD включают в них также много комментариев, объясняющих объявления и то, как их можно использовать при создании документов. В этой главе мы пользуемся примерами, взятыми из XHTML 1.0 DTD, которое можно найти в полном виде на веб-сайте W3C. Заметьте, что оно, хотя и длинно, но хорошо написано, полно и, при некотором опыте, просто для понимания.

В XML возможно также создание условных разделов в DTD, что позволяет обработчику DTD по выбору включать или исключать группы объявлений. Это удобно, когда DTD на самом деле определяет несколько версий языка разметки; нужная версия может быть установлена путем включения или исключения соответствующих разделов. Например, XHTML 1.0 DTD определяет как обычную версию HTML, так и версию, поддерживающую фреймы. За счет того, что анализатор DTD может включать только нужные разделы, правила для тега `<html>` могут меняться, чтобы поддерживался либо тег `<body>`, либо тег `<frameset>`.

15.3.1. Комментарии

Синтаксис комментариев в XML DTD совершенно такой же, как в HTML, – они начинаются с `<!--` и заканчиваются `-->`. Все, что находится между этими двумя элементами, обработчик XML игнорирует. Комментарии нельзя вкладывать друг в друга.

15.3.2. Сущности

Сущности – это причудливое обозначение для констант. Сущности играют решающую роль при создании структурированных, понятных для чтения DTD. Во многом различаясь, все сущности, однако, связывают имя с определенной строкой символов. Если имя сущности употребляется где-либо в DTD или в XML-документе, анализатор языка заменит имя соответствующими символами. Возьмем пример из HTML: имя `<` заменяется символом `<`, где бы оно ни появилось в HTML-документе.

Сущности бывают двух видов: *обрабатываемые* (*parsed*) и *необрабатываемые* (*unparsed*). Обрабатываемые сущности рассматриваются XML-анализатором (парсером), необрабатываемые им игнорируются. Подавляющее большинство сущностей являются обрабатываемыми. Необрабатываемые сущности зарезервированы для использования в списках атрибутов некоторых тегов; они представляют собой не что иное как заменители строк, используемых в качестве значений атрибутов тега.

Далее в группе обрабатываемых сущностей можно выделить *общие* (*general*) сущности и *параметрические* (*parameter*) сущности. Общие сущности используются в XML-документе, а параметрические – в XML DTD.

Вы могли не сознавать, что пользуетесь общими сущностями, создавая HTML-документы. Как и для всех других общих сущностей, имени сущности предшествует символ амперсанда. К примеру, сущность для символа копирайта (©), `©` – это общая сущность, установленная в HTML DTD. Все известные и наиболее распространенные общие сущности перечислены в приложении F.

Для большей простоты XML заранее определяет пять самых распространенных сущностей, которые можно использовать в любом XML-документе. Эти пять сущностей доступны любому XML-автору, хотя все-таки их лучше явно определять в каждом использующем их DTD:

```
amp;            &
&apos;          '
&gt;            >
&lt;            <
&quot;          "
```

Параметрические сущности во множестве встречаются в любом корректно написанном DTD, включая HTML DTD. Перед именами таких сущностей ставится знак процента (%). Этот знак говорит XML-процессору, чтобы он нашел имя сущности в списке параметрических сущностей, вставил значение сущности в DTD вместо ссылки на нее и обработал значение сущности как часть DTD.

Последнее особенно важно. При обработке содержимого параметрической сущности как части DTD XML-обработчик позволяет вам использовать в качестве значения сущности любое действительное XML-содержимое. Многие параметрические сущности содержат пространные XML-определения и могут даже содержать определения других сущностей. Параметрические сущности – это рабочие лошадки XML DTD; создание DTD без них было бы нелегким делом.[1]

15.3.3. Объявления сущностей

Определим сущность при помощи тега `<!ENTITY>` в XML DTD. Внутри этого тега поставим имя сущности и ее значение, а также обозначим, общая это сущность или параметрическая:

```
<!ENTITY name value>
<!ENTITY % name value>
```

Первое объявление определяет общую сущность, а второе – параметрическую (это обозначено знаком процента).

---

[1] Программисты на C и C++, возможно, признают, что механизм сущностей в XML похож на механизм макроопределений `#define` в C и C++. Сущности в XML допускают только простые подстановки строк символов и не используют более разработанные механизмы макропараметров языков C.

Для сущностей обоих типов имя (name) – это просто последовательность символов, начинающаяся с буквы, двоеточия или подчеркивания, за которыми может следовать любая комбинация букв, цифр, точек, дефисов, подчеркиваний и двоеточий. Единственное ограничение заключается в том, что имена не должны начинаться с символа[1], отличного от двоеточия или подчеркивания, или с последовательности «xml» (в верхнем или нижнем регистре).

Значение сущности – это либо строка символов, заключенная в кавычки (в отличие от HTML-разметки, следует использовать кавычки даже тогда, когда строка представляет собой «слово», непрерывную последовательность букв), либо ссылка на другой документ, содержащий значение сущности. Для таких внешних значений сущностей используется либо ключевое слово SYSTEM, за которым следует URL документа, содержащего значение сущности, либо ключевое слово PUBLIC, за которым следует формальное имя документа и его URL.

Несколько примеров все прояснят. Вот простое объявление общей сущности:

```
<!ENTITY fruit "kumquat or other similar citrus fruit">
```

В этом объявлении сообщается, что где бы в документе ни встретилась сущность "&fruit;", ее следует заменить на фразу «kumquat or other similar citrus fruit».

А вот объявление параметрической сущности:

```
<!ENTITY % ContentType "CDATA">
```

Теперь, где бы в вашем DTD ни встретилась ссылка %ContentType;, она будет заменена на слово «CDATA». Это типичный способ использования параметрических сущностей: создание лучше запоминающегося термина для часто употребляемого в DTD родового параметра.

Вот объявление внешней общей сущности:

```
<!ENTITY boilerplate SYSTEM "http://server.com/boilerplate.txt">
```

В нем XML-обработчику предлагается получить содержимое файла *boilerplate.txt* с *server.com* и использовать его в качестве значения сущности «boilerplate». Везде, где в документе вы напишете &boilerplate;, на этом месте окажется содержимое указанного файла.

Вот объявление внешней параметрической сущности, извлеченное из HTML DTD и ссылающееся на общедоступный внешний документ:

```
<!ENTITY % HTMLlat1 PUBLIC "~//W3C//ENTITIES Latin 1 for XHTML//EN" "xhtml-
lat1.ent">
```

В нем определяется сущность по имени HTMLlat1, содержимое которой должно быть взято из общедоступного документа, обозначенного как

[1] Из числа «специальных символов», отличных от латинских литер и цифровых символов. – *Примеч. науч. ред.*

«*~//W3C//ENTITIES Latin 1 for XHTML/ /EN*». Если у обработчика под рукой нет копии этого документа, он может использовать URL «*xhtml-lat1.ent*», чтобы найти его. Этот общедоступный документ в действительности очень длинный и содержит все объявления общих сущностей для кодировок символов набора Latin 1 в HTML.[1] Соответственно, написав в HTML DTD просто:

```
%HTMLlat1;
```

все эти общие сущности можно определить как часть языка.

Авторы DTD могут использовать в объявлениях общих и параметрических сущностей внешние значения PUBLIC и SYSTEM. Необходимо структурировать внешние определения так, чтобы DTD и документы было легко читать и понимать.

Как вы помните, в начале этого раздела, посвященного сущностям, мы упомянули о необрабатываемых сущностях, единственная функция которых – употребляться в качестве значений некоторых атрибутов. Вы объявляете необрабатываемую сущность, приписывая в конец объявления внешней общей сущности ключевое слово NDATA, за которым следует имя необрабатываемой сущности. Если в приведенном выше примере мы хотим преобразовать общую сущность «boilerplate» в необрабатываемую общую сущность, чтобы воспользоваться ею в качестве значения атрибута, напишем:

```
<!ENTITY boilerplate SYSTEM "http://server.com/boilerplate.txt" NDATA text>
```

В соответствии с этим объявлением атрибуты типа ENTITY (как это описано в разделе 15.5.1) могут использовать `boilerplate` в качестве одного из своих значений.

15.3.4. Элементы

Элементы – это определение тегов, которые могут использоваться в документах, написанных на вашем XML-языке. В некотором смысле объявления элементов проще, чем объявления сущностей, так как единственное, что вам нужно сделать, – это определить имя тега и вид содержимого, которое он может заключать в себе:

```
<!ELEMENT name contents>
```

Имя подчиняется тем же правилам, что и имена в определениях сущностей. Содержимое может описываться одним из четырех перечисленных ниже способов:

- Ключевое слово `EMPTY` определяет теги без содержимого, такие как `<hr>` и `<br>` в HTML. При работе с пустыми элементами в XML нужно учитывать некоторые нюансы, описанные в разделе 15.4.5.

[1] Вы можете сами ознакомиться с этим документом по адресу *http://www.w3.org/TR/xhtml1/DTD/ xhtml-symbol.ent*.

- Ключевое слово ANY означает, что тег может заключать любое содержимое, без ограничений или дальнейшей обработки XML-обработчиком.
- Содержимое может представлять собой множество грамматических правил, определяющих порядок следования тегов и их вложенность в определяемый элемент. Этот тип содержимого используется, когда определяемый тег содержит только другие теги, но не обычное содержимое. В HTML таким является тег <ul>, поскольку он может содержать только теги <li>.
- Смешанное содержимое, обозначаемое списком имен элементов, разделенных запятыми, и ключевым словом #PCDATA, заключается в скобки. Этот тип содержимого позволяет тегам включать содержимое, определяемое пользователем, наряду с другими элементами разметки. Например, содержимое тега <li>, равно как и других тегов, может определяться пользователем.

Два последних типа содержимого составляют основу большей части объявлений элементов в DTD. Тут-то все и начинается.

15.4. Грамматика элементов

Грамматика естественных языков изобилует разнообразными синтаксическими конструкциями, временами глаголов и всякого рода случайностями и исключениями. Тем не менее уже трехлетний ребенок владеет большинством из них. Грамматика компьютерных языков обычно проста, регулярна, и в ней мало исключений. Фактически компьютерные грамматики используют четыре типа правил для определения устройства элементов языка: это правила последовательности вхождения, правила выбора, правила кратности вхождения и правила группировки.

15.4.1. Последовательность, выбор, группировка и кратность

Правила *последовательности* определяют точный порядок следования элементов в конструкциях языка. К примеру, если грамматическое правило последовательности говорит, что за элементом A следует элемент B, а за ним C, элементы A, B и C в вашем документе должны появиться точно в этом порядке. Пропуск элемента (A, а за ним C, без B, например), включение лишнего элемента (A, B, E, C) или изменение порядка следования (C, A, B) нарушают правило и противоречат грамматике.

Во многих грамматиках, включая XML, правила последовательности определяются простым перечислением (через запятую и в правильном порядке) соответствующих элементов. Соответственно правило последовательности из нашего примера выглядело бы в DTD просто как A, B, C.

Правила *выбора* придают гибкость конструкциям языка, позволяя авторам DTD выбирать один элемент из группы возможных. К примеру, правило выбора гласит, что вы можете выбирать из трех элементов: D, E или F. Любой из них отвечает грамматическим нормам. Как и во многих других грамматиках, правило выбора в XML обозначается перечислением допустимых вариантов через вертикальную черту (|). Таким образом, наше простое правило выбора может быть записано в DTD как `D|E|F`. Если вы будете читать вертикальную черту как слово «или», правило выбора будет легко уяснить.

Правила *группировки* собирают два или большее число правил в одно, позволяя строить более богатые и пригодные к употреблению языки. К примеру, правило группировки допускает последовательность элементов, за которой следуют выбор и затем другая последовательность. Обозначить группу, входящую в правило, можно заключив ее в скобки. Например:

```
Document ::= A, B, C, (D | E | F), G
```

Такое правило требует, чтобы документ начинался с элементов A, B, C, за которыми следует один из элементов – D, E или F согласно правилу выбора, и завершает конструкцию элемент G.

Правила *кратности* позволяют регулировать допустимое количество повторений одного или нескольких элементов. В XML, как и во многих других языках, кратность обозначается путем приписывания суффикса в виде специального символа к элементу или группе элементов правила. Без суффикса элемент или группа употребляется только один раз. К этим специальным символам относятся: знак «плюс» (+), означающий, что элемент может появиться несколько раз, но по меньшей мере однажды; звездочка (*), означающая, что элемент может появиться несколько раз или не появиться совсем; вопросительный знак (?), означающий, что элемент может не появиться или появиться ровно один раз.

К примеру, следующее правило:

```
Document ::= A, B?, C*, (D | E | F)+, G*
```

создает неограниченное множество документов с элементами от A до F. В соответствии с этим правилом каждый документ должен начинаться с A, за ним может следовать (а может и не следовать) B, за ним следует (нуль или больше раз) C, затем следует по меньшей мере один, но, возможно, и больше раз либо D, либо E, либо F, за которым следует нуль или большее число элементов, обозначенных как G. Все нижеследующие документы (и множество других) соответствуют этому правилу:

```
ABCDG
ACCCFFGGG
ACDFDFGG
```

Вы, возможно, захотите проработать эти примеры, чтобы убедиться в том, что все они действительно соблюдают правила кратности.

15.4.2. Составные грамматические правила

Теперь вам, может быть, кажется, что записать грамматику всего языка в одном правиле трудно, хотя и возможно. К несчастью, в результате получилась бы неудобочитаемая последовательность правил, смысл которых едва ли был бы ясен. Спасает в этой ситуации то, что составляющими правила, кроме собственно элементов языка, могут быть сами правила, содержащие, в свою очередь, другие правила и элементы. В таких случаях элементы грамматики, которые сами являются правилами, называются *нетерминальными символами*, а элементы грамматики, представляющие собой элементы языка, называются *терминальными символами*. В конечном счете, все нетерминальные символы должны сводиться к последовательности терминальных символов, иначе грамматике не соответствовал бы ни один документ.

Мы можем выразить грамматику из нашего примера посредством двух правил:

```
Document ::= A, B?, C*, Choices+, G*
Choices ::= D | E | F
```

В этом примере Document и Choices – это нетерминальные символы, тогда как A, B, C, D, E, F и G – терминальные символы.

В грамматике XML, как и в большинстве других, нет правил, ограничивающих количество нетерминальных элементов в ней. Большинство грамматик пользуются нетерминальными элементами всякий раз, когда это способствует ясности и удобству.

15.4.3. Грамматика элементов в XML

Правила, определяющие содержимое элемента, соответствуют грамматическим правилам, которые мы только что обсудили. Для определения допустимого содержимого элемента можно использовать комбинации правил выбора, последовательности, группировки и кратности. Нетерминальные элементы правил должны быть именами других элементов, определенных в вашем DTD.

Несколько примеров покажут, как это работает. Рассмотрим объявление тега `<html>`, взятое из HTML DTD:

```
<!ELEMENT html (head, body)>
```

Оно определяет элемент под названием `html`, содержимым которого является элемент `head`, за которым следует элемент `body`. Отметьте, что в DTD вы не заключаете имена элементов в угловые скобки – это обозначение применяется только тогда, когда элементы используются в документе.

В HTML DTD вы можете найти объявление тега `<head>`:

```
<!ELEMENT head (%head.misc;,
     ((title, %head.misc;, (base, %head.misc;)?) |
      (base, %head.misc;, (title, %head.misc;)))*)>
```

Ух! Что бы все это могло значить? Во-первых, отметим, что здесь имеется параметрическая сущность с именем head.misc, неоднократно упомянутая в объявлении. Давайте ее разыщем:

```
<!ENTITY % head.misc "(script|style|meta|link|object)*">
```

Теперь все начинает проясняться: head.misc определяет группу элементов, из которой вы можете выбрать один. Звездочка в конце, однако, указывает, что вы можете включить нуль или большее число этих элементов. В совокупности это означает, что всюду, где встречается %head.misc;, вы можете включить любое число элементов script, style, meta, link или object в любом порядке или не включить ни одного. Знакомо, не правда ли?

Возвращаясь к объявлению head, мы видим, что можно начать с любого числа разных элементов заголовка. Затем мы должны сделать выбор между группой, состоящей из элемента title, необязательных членов head.misc и необязательного элемента base, за которым следуют другие элементы, и группой из элемента base, за которым следуют еще элементы, элемент title и еще какое-то число разных элементов.

Зачем понадобилось такое закрученное правило для тега <head>? Почему бы не написать просто:

```
<!ELEMENT head (script|style|meta|link|object|base|title)*>,
```

что позволяло бы появиться любому числу элементов заголовка – или ни одному? Потому что стандарт HTML требует, чтобы каждый тег <head> содержал ровно один тег <title>. А также он требует, чтобы тегов <base> было не больше одного. В остальном стандарт допускает любое количество других элементов заголовка в любом порядке.

Проще говоря, объявление элемента head, показавшееся нам сначала запутанным, построено таким образом, чтобы в элементе head обязательно имелся ровно один элемент title и не более одного элемента base. Все другие элементы могут появляться в любом количестве и любом порядке.

Уже один этот пример прекрасно демонстрирует могущество XML – его способность определять обычно используемые элементы при помощи параметрических сущностей и применять грамматические правила для определения синтаксиса документов. Если вы смогли разобраться в объявлении элемента head и понять его, то вы делаете большие успехи в умении читать XML DTD.

15.4.4. Смешанное содержимое элементов

Смешанное содержимое элементов расширяет правила грамматики элементов включением в состав терминальных символов специального ключевого слова #PCDATA. Слово «PCDATA» является сокращением слов «parsed character data» (обработанные символьные данные) и означает, что содержимое элемента должно быть исследовано XML-обра-

ботчиком на предмет наличия общих сущностей. После того как все общие сущности заменены, символьные данные передаются XML-приложению для дальнейшей обработки.

Смысл этого состоит в том, что обработанные символьные данные – это действительное содержимое вашего XML-документа. Элементы, которые допускают в качестве содержимого обработанные символьные данные, могут включать в себя простой текст, а также другие теги, которые DTD разрешает им содержать.

К примеру:

```
<!ELEMENT title (#PCDATA)>
```

означает, что элемент `title` может содержать только текст с сущностями. Стандартом HTML не допускаются никакие другие теги.

Более сложный пример представляет тег `<p>`, объявление которого таково:

```
<!ELEMENT p %Inline;>
```

Еще одна параметрическая сущность! Сущность `%Inline;` определена в HTML DTD:

```
<!ENTITY % Inline "(#PCDATA | %inline; | %misc;)*">
```

что расширяется при раскрытии сущностей до:

```
<!ENTITY % special "br | span | bdo | object | img | map">
<!ENTITY % fontstyle "tt | i | b | big | small">
<!ENTITY % phrase "em | strong | dfn | code | q | sub | sup | samp |
          kbd | var | cite | abbr | acronym">
<!ENTITY % inline.forms "input | select | textarea | label | button">
<!ENTITY % misc "ins | del | script | noscript">
<!ENTITY % inline "a | %special; | %fontstyle; | %phrase; |
          %inline.forms;">
```

Что мы получили в результате? Сущность `%Inline;` определяет содержимое элемента `p` как разобранные символьные данные плюс любые элементы, определенные в `%inline;`, а также любые элементы, определенные в `%misc;`. Отметьте, что регистр имеет значение: `%Inline;` – это не то же, что `%inline;`.

Сущность `%inline;` включает массу всего: специальные элементы, элементы стилей шрифтов, фразовые элементы и встроенные элементы форм. `%misc;` включает элементы `ins`, `del`, `script` и `noscript`. Вы можете изучить HTML DTD, где имеются другие объявления сущностей, чтобы узнать, какие еще элементы допускаются в качестве содержимого элемента `p`.

Почему авторы HTML DTD разбивают все эти элементы на отдельные группы? Если бы они просто определяли элементы, которые можно включить в элемент `p`, они создали бы один большой список. Однако правила HTML определяют, где в документе могут встречаться встро-

енные элементы. Авторы сгруппировали сходные элементы в отдельные сущности, на которые в DTD можно сослаться несколько раз. За счет этого DTD легче читать и понимать, как и легче изменять его нужным образом.

15.4.5. Пустые элементы

Элементы, содержимое которых пусто по определению, заслуживают особого упоминания. XML вводит новые правила обозначения таких элементов, отличающиеся от традиционных правил HTML.

HTML-авторы привыкли обозначать такие элементы одним тегом, как, например теги `<br>` или `<img>`. XML требует, чтобы у каждого элемента были открывающий и закрывающий теги, так что тег изображения должен записываться как `<img></img>` без вложенного содержимого. Другие пустые элементы должны записываться подобным же образом.

Поскольку этот формат хорошо работает для непустых тегов, но кажется несколько избыточным для пустых, можно использовать специальное сокращенное обозначение пустых тегов. Чтобы записать такой тег в XML, достаточно поместить символ наклонной черты (/) перед закрывающей угловой скобкой тега. Таким образом, разрыв строки может быть обозначен как `<br/>`, а тег изображения – как `<img src="myimage.gif"/>`. Отметьте, что если у пустого элемента имеются атрибуты, они записываются перед закрывающей наклонной чертой и скобкой.[1]

15.5. Атрибуты элементов

Завершают головоломку под названием DTD атрибуты. Вы с ними знакомы – это пары имя/значение, включаемые в тег и управляющие его внешним видом и поведением. Для определения в XML DTD атрибутов и их допустимых значений используйте директиву `<!ATTLIST>`:

```
<!ATTLIST element attributes>
```

Здесь `element` – это имя элемента, к которому относится атрибут. Поле `attributes` содержит список объявлений атрибутов элемента. Объявление каждого из атрибутов состоит из имени атрибута, его типа и принимаемого по умолчанию значения (если такое есть).

15.5.1. Значения атрибутов

Значения атрибутов бывают нескольких типов, каждый из которых в определении атрибута обозначается одним из следующих ключевых слов:

---

[1] То есть тег может быть «пустой» в смысле отсутствия обрамляемого им содержимого, но содержать массу информации в значениях атрибутов. – *Примеч. науч. ред.*

`CDATA`

Показывает, что значением атрибута является символ или строка символов. Этот тип атрибутов предназначен для указания URL или других произвольных данных пользователя.

`ID`

Подразумевает, что значением атрибута является уникальный в пределах документа идентификатор. Этот тип атрибута используется с атрибутом, таким как атрибут `id` в HTML, значение которого определяет ID в документе. Эта тема обсуждается в приложении B в разделе «Основные атрибуты».

`IDREF` или `IDREFS`

Означают, что атрибут принимает в качестве значений ID, определенные где-то в документе при помощи атрибута типа `ID`. Тип `ID` используется при определении разных идентификаторов, а `IDREF` и `IDREFS` используются при ссылках на один идентификатор или на их список соответственно.

`ENTITY` или `ENTITIES`

Означают, что атрибут получает имя или список имен необрабатываемых общих сущностей, определенных где-то в другом месте DTD. Определение и использование необрабатываемых общих сущностей описаны в разделе 15.3.2.

`NMTOKEN` или `NMTOKENS`

Означают, что атрибут принимает действительное XML-имя или список имен. Эти имена передаются обрабатывающему приложению как значения атрибута. Приложение определяет, что с ними делать.

Вдобавок к этим типам, обозначенным ключевыми словами, вы можете создавать перечисляемые типы, указывая последовательно все допустимые значения атрибута. Чтобы создать перечисляемый тип, перепишите допустимые значения через вертикальную черту и заключите список в скобки. Для примера посмотрите, как в HTML DTD определяется атрибут `method` для тега `<form>`:

```
method      (get|post)      "get"
```

Атрибут `method` принимает одно из двух значений: либо `get`, либо `post`. По умолчанию принимается значение `get`, на тот случай, если в теге ничего конкретно не сказано.

15.5.2. Обязательные и принимаемые по умолчанию атрибуты

После определения имени и типа атрибута вы должны указать, как XML-обработчик должен обращаться с обязательными и принимаемыми по умолчанию значениями атрибута. Можно сделать это, указав после типа атрибута одно из четырех значений.

Если вы используете ключевое слово #REQUIRED, определяемый атрибут обязательно должен присутствовать при всяком появлении элемента в документе. В XHTML DTD атрибут src для тега <img> является обязательным, поскольку тег <img> не имеет смысла, если не указано изображение, которое следует отобразить.

Ключевое слово #IMPLIED означает, что атрибут можно, но не обязательно использовать и что по умолчанию с ним никакое значение не ассоциируется. Если этот атрибут не указан автором документа, XML-обработчик не присвоит этому атрибуту никакого значения при обработке элемента. Для тега <img> атрибуты width и height являются подразумевающимися, поскольку если эти атрибуты не специфицированы, броузер примет решение о размерах изображения, основываясь на самом изображении.

Если значение указано, оно становится для этого атрибута принимаемым по умолчанию. Если пользователь не задаст значение атрибута в тексте на определяемом языке, XML-обработчик вставит значение по умолчанию, которое определено в DTD.

Если вы вставите перед значением по умолчанию ключевое слово #FIXED, это значение не только будет приниматься по умолчанию, но и будет *единственным* допустимым значением атрибута при его явном включении.

Для примера изучим список атрибутов элемента form, в сокращенном виде взятый нами из HTML DTD:

```
<!ATTLIST form
action          CDATA#REQUIRED
method          (get|post)"get"
enctype         CDATA"application/x-www-form-urlencoded"
onsubmit        CDATA#IMPLIED
onreset         CDATA#IMPLIED
accept          CDATA#IMPLIED
accept-charset  CDATA#IMPLIED
>
```

Этот пример связывает с элементом form семь атрибутов. Атрибут action является обязательным и принимает в качестве значения строку символов. Атрибут method принимает одно из двух значений – либо get, либо post. Значение get принимается по умолчанию, поэтому автор документа не обязан включать атрибут method в тег формы, XML-обработчик автоматически будет полагать method=get.

Атрибут enctype для элемента form принимает в качестве значений строку символов и, если не определяется в теге явно, по умолчанию принимает значение application/x-www-form-urlencoded. Все остальные атрибуты принимают в качестве значений строки символов, не являются обязательными и по умолчанию не принимают никаких значений.

Если вы посмотрите на список атрибутов элемента `form` в HTML DTD, то увидите, что он не совпадает в точности с нашим примером. Это вызвано тем, что мы изменили наш пример, чтобы показать типы атрибутов после раскрытия всех параметрических сущностей. В самом HTML DTD типы атрибутов указаны в виде параметрических сущностей, имена которых намекают на то, какого рода значения ожидаются атрибутом. Например, тип атрибута `action` выглядит как `%URL;`, а не `CDATA`, но где-то в DTD определен как `CDATA`. Действуя подобным образом, авторы DTD дают вам понять, что строковое значение данного атрибута должно быть URL, а не просто какой-то строкой. Аналогично для атрибутов `onSubmit` и `onReset` указан тип `%Script;`. Это подсказывает, что строка символов должна быть именем сценария, который следует выполнить при отправке или обновлении формы.

15.6. Условные разделы

Как мы уже отмечали в этой главе, XML позволяет включать или игнорировать целые разделы в DTD, так что вы можете подгонять язык «по фигуре». HTML DTD, например, содержит переходную, строгую и фреймовую версии языка. Авторы DTD могут выделить части DTD, которые они планируют включить или проигнорировать, используя условные директивы XML:

```
<![INCLUDE [
   ...любое XML-содержимое ...
]]>
```

или

```
<![IGNORE [
   ...любое XML-содержимое ...
]]>
```

XML-обработчик соответственно включит или проигнорирует содержимое. Условные разделы могут включаться друг в друга с соблюдением правила: разделы, содержащиеся в игнорируемом разделе, будут пропущены, даже если сами они объявлены в качестве включенных.

Вы редко увидите DTD, в которых явно прописаны ключевые слова `INCLUDE` и `INGNORE`. Вместо этого вы увидите параметрические сущности, на основании которых разделы включаются или игнорируются. Допустим, вы создаете DTD для обмена чертежами конструкций между строителями. Поскольку вы ориентируетесь на международную клиентуру, вы строите DTD, которое поддерживает и британскую, и метрическую систему мер. Вы можете определить две параметрические сущности так:

```
<!ENTITY % English "INCLUDE">
<!ENTITY % Metric "IGNORE">
```

Вы можете затем поместить объявления, специфичные для английской системы, в условный раздел и подобным же образом изолировать метрические объявления:

```
<![%English [
   ...Здесь английские определения...
]]>
<![%Metric [
   ...Здесь метрические определения...
]]>
```

Чтобы использовать DTD для строительных работ в США, определите `%English;` как `INCLUDE`, а `%Metric;` — как `IGNORE`, в результате чего в вашем DTD будут использоваться английские объявления. Для метрических чертежей сделайте противоположные установки, игнорируя английский раздел и включив метрический.

15.7. Построение XML DTD

Изучив XML DTD в анатомическом театре, давайте посмотрим на них в живом виде. Построим простой пример. Вы можете создавать DTD при наличии любого текстового редактора и ясного представления о том, как вы собираетесь размечать XML-документы. Вам понадобятся XML-обработчик (парсер) и приложение, способное работать с XML, чтобы интерпретировать и использовать ваше DTD, а также таблица стилей, которая позволит XML-совместимому броузеру отобразить ваш документ.

15.7.1. DTD адреса XML

Давайте создадим простое XML DTD, определяющее язык разметки для создания документов, содержащих имена и адреса. Мы начнем с элемента `address`, включающего в себя другие элементы, которые размечают адрес. Наш элемент `address` имеет один атрибут, указывающий, домашний ли это адрес или рабочий:

```
<!ELEMENT address (name, street+, city, state, zip?)>
<!ATTLIST address type (home|business) #REQUIRED>
```

Ну вот! Первое объявление создает элемент, именуемый `address`, который содержит элемент `name`, один или несколько элементов `street` (улица), элементы `city` (город) и `state` (штат) и необязательный элемент `zip` (почтовый код). У элемента `address` имеется один атрибут `type`, который обязательно должен быть определен и может принимать либо значение `home`, либо значение `business`.

Давайте сначала определим элемент `name`:

```
<!ELEMENT name (first, middle?, last)>
<!ELEMENT first (#PCDATA)>
<!ELEMENT middle (#PCDATA)>
<!ELEMENT last (#PCDATA)>
```

Элемент name тоже содержит другие элементы: имя, необязательное второе имя и фамилию – каждый из этих элементов определен в последующих строках DTD. Они не могут содержать тегов, но только обработанные символьные данные, то есть действительное имя лица.

Остальные элементы адреса тоже легко описываются:

```
<!ELEMENT street (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT zip (#PCDATA)>
<!ATTLIST zip length CDATA "5">
```

Все эти элементы содержат обработанные символьные данные. У элемента zip есть атрибут length, обозначающий длину почтового индекса. Если атрибут length не определен, он устанавливается равным пяти.

15.7.2. Использование DTD адреса

Определив DTD адреса, мы можем использовать это DTD для разметки документа типа «адрес». Например:

```
<address type="home">
   <name>
      <first>Chuck</first>
      <last>Musciano</last>
   </name>
   <street>123 Kumquat Way</street>
   <city>Cary</city>
   <state>NC</state>
   <zip length="10">27513-1234</zip>
</address>
```

При наличии подходящего XML-анализатора и приложения, которое использовало бы эти данные, мы смогли бы обрабатывать и хранить адреса, создавать программы, которые могли бы представлять адреса в самых разных стилях и на самых разных носителях информации. Хотя наше DTD простое, оно дает нам стандартный способ ввода адресных данных, который легко понять и использовать.

15.8. Использование XML

Наш пример с адресом тривиален. Он едва скользнул по поверхности широкого океана возможных применений XML. Чтобы раздразнить ваш аппетит, приведем здесь несколько обычных направлений использования XML, которые вы, без сомнения, сможете увидеть уже сейчас или в будущем.

15.8.1. Создание новых языков разметки

Мы уже коснулись этого вопроса, когда отметили, что HTML в последних версиях формулируется как DTD, согласующееся с XML. Мы обсудим влияние, оказанное XML на HTML, в следующей главе.

Но что еще важнее – XML позволяет сообществу пользователей создавать языки, наилучшим образом приспособленные для выражения их неповторимых идей и данных. Математики, химики, музыканты и профессионалы сотен других специальностей могут стандартным способом создавать особенные теги, которые будут отображать понятия, свойственные только их дисциплине. Даже если еще нет броузеров, которые правильно отобразят эти теги, возможность охватывать и упорядочивать информацию имеет огромное значение для будущих исследователей и интерпретаторов этих идей.

В отношении традиционных применений XML, ориентированных на устоявшуюся аудиторию, легко предвидеть создание настраиваемых броузеров, которые смогут отображать новую информацию. Менее широкие области применения и рынки будут в большей степени заинтересованы в создании собственных языков разметки, чтобы воспользоваться всеми возможностями новых броузеров. Создание настраиваемого средства отображения языков разметки – это трудная задача. Распространение его на различные платформы дорого. Как мы уже отмечали, острота некоторых из этих проблем с отображением информации может быть смягчена соответствующим применением таблиц стилей. К счастью, возможности XML выходят за пределы представления документов.

15.8.2. Обмен документами

Поскольку XML возник благодаря оглушительному успеху HTML, многие люди рассматривают XML как очередное средство отображения документов. На самом деле XML по-настоящему могуч даже не в этой области, а в том, что касается ввода данных и обмена ими.

Несмотря на миллиарды компьютеров, разбросанных по всему миру, совместное использование данных остается таким же утомительным и чреватым ошибками занятием. Конкурирующие приложения не принимают в расчет общепринятые форматы хранения документов, поэтому отправка одного документа множеству получателей сопряжена с риском. Даже когда поставщики пытаются создать формат обмена, он все равно склонен оставаться собственностью фирмы и часто рассматривается как источник преимущества в конкуренции. У поставщиков программной продукции мало стимулов для публикации исходных кодов приложений, что облегчило бы обмен документами.

XML свободен от этих проблем. Всеобщий, независимый от платформ, он способен выполнить почти любые задачи, связанные с приемом данных. Он одинаково доступен всем поставщикам и легко может быть интегрирован в большинство приложений. Стабилизация стандарта XML, растущее число людей, способных писать на XML, возрастающая доступность XML-парсеров позволят все с большей легкостью создавать XML-языки разметки для сбора документов и обмена ими.

Всего важнее то, что обмен документами редко требует представления документов, что позволяет вычеркнуть из уравнения члены с «трудно-

стями отображения». Существующие приложения будут использовать XML, чтобы включать данные из другого источника, затем используя свои собственные возможности отображения документов для представления данных конечному пользователю.[1] Стоимость добавления к действующему приложению средств обмена данными, основанных на XML, относительно невелика.

15.8.3. Системы соединения

Вслед за приложениями идет потребность в системах обмена данными. Эта потребность растет даже быстрее, чем объем коммуникаций. В прошлом это означало, что кто-то должен разработать протокол кодировки и обмена. Если же воспользоваться XML, то обмен данными требует только определения DTD и интегрирования анализатора с уже существующим приложением.

Передаваемые наборы данных могут в действительности быть очень маленькими. Вообразим себе поиск нового компьютера через Сеть. Если вы сумели изложить свои требования к системе в виде маленького документа, пользуясь XML DTD, вы можете разослать эту спецификацию сотням разных поставщиков с просьбой сообщить цену. Если вы расширите эту модель так, чтобы она включала почти все, что можно купить – от автомобилей до косметики, – XML даст вам отличную базу для коммуникаций между поставщиками, сотрудничающими в Интернете.

Практически все данные, введенные и хранящиеся к настоящему моменту, при помощи XML гораздо легче сделать доступными множеству пользователей. Для многих систем XML DTD будет определять протокол передачи данных и ничего больше. Данные, возможно, никогда не будут в действительности храниться в соответствии с разметкой, определяемой XML, – они могут принимать согласующуюся с XML форму только на время передачи данных между системами.

Одним из применений XML, набирающим все большую популярность, являются веб-службы. Они позволяют разным приложениям распознавать друг друга и «гладко» обмениваться данными через Интернет независимо от языка программирования и архитектуры. Более подробную информацию по веб-службам вы найдете в книге Этана Керами (Ethan Cerami) «Web Services Essentials», выпущенной издательством O'Reilly.

Вместе с обменом данными на основе XML для описания внешнего вида и определения данных, представленных в соответствии с XML DTD,

[1] Ярким примером реализованной схемы является успех «свободного» офисного пакета OpenOffice, текстовый процессор которого хранит документы в формате XML, и они одинаково представляются на самых разных платформах: Windows, Linux и др. – *Примеч. науч. ред.*

все больше будет применяться расширяемый язык таблиц стилей (XSL, Extensible Stylesheet Language). Подобно каскадным таблицам стилей и их способности трансформировать HTML-документ, XSL будет поддерживать создание таблиц стилей для любых XML DTD. CSS тоже могут использоваться с XML-документами, но их средства программирования не так богаты, как в XSL. Тогда как CSS останавливается на таблицах стилей, XSL – это целый язык стилей. XSL, конечно, в первую очередь отвечает на потребность в отображении данных, но также предоставляет богатые средства, которые позволяют контролируемым и предсказуемым образом приводить данные, соответствующие одному DTD, в соответствие с другим DTD. Полное обсуждение XSL выходит за пределы этой книги. Обратитесь за подробностями к книге Doug Tidwell «XSLT», выпущенной издательством O'Reilly.

Потенциал XML во многом превосходит возможности традиционных средств разметки и представления данных. И сейчас нам видна и нами задействуется лишь ничтожная часть потенциала этой технологии.

15.8.4. Стандартизация HTML

И последнее – по порядку, но не по значению: XML используется в настоящее время для определения стандартной версии HTML, известной под названием XHTML. XHTML сохраняет почти все черты HTML 4.01, но содержит также множество маленьких (и несколько не столь маленьких) отличий. Следующая глава сравнивает и сопоставляет друг с другом HTML и XHTML, отмечая их различия, так что вы сможете приступить к созданию документов, которые будут соответствовать обоим стандартам, HTML 4 и XHTML.

- Зачем нужен XHTML?
- Создание документов XHTML
- HTML и XHTML
- XHTML 1.1
- Использовать ли XHTML?

16

XHTML

Несмотря на свое название, расширяемый язык разметки (XML) не применяется непосредственно для создания и разметки веб-документов. Вместо этого посредством XML-технологии определяются новые языки разметки, которые затем уже используются при разметке веб-документов. Это наверняка не окажется сюрпризом для тех, кто прочел предыдущую главу этой книги. Не удивитесь вы и тому, что одним из первых языков, определенных при помощи XML, стала модифицированная версия HTML – самого популярного языка разметки всех времен. XML теперь дисциплинирует и очищает HTML, чтобы вернуть его в лоно большой семьи языков разметки. Этот стандарт называется XHTML 1.0.[1]

Дурная наследственность и странности, присущие HTML, сделали нелегкой работу W3C по описанию HTML средствами XML. На самом деле некоторые правила HTML, о которых мы еще расскажем, нельзя представить, пользуясь XML. Тем не менее, если W3C добьется своего, XHTML в конце концов заменит HTML, который сейчас все мы знаем и любим.

В XHTML так много совпадений с действующим стандартом HTML версии 4.01, что почти все, о чем говорится в других главах этой книги, может быть применено и к HTML, и к XHTML. Отличия – в лучшую и в худшую сторону – детально излагаются в этой главе. Чтобы овладеть XHTML, вам нужно сначала усвоить остальное содержимое этой книги – и тогда ваш разум будет готов понять то, о чем мы говорим в этой главе.

---

[1] Всюду в этой главе мы пишем «XHTML», имея в виду стандарт XHTML 1.0. Грядущий стандарт XHTML 1.1 строже, чем XHTML 1.0, и расходится с HTML 4.01. Мы обсудим наиболее заметные черты стандарта XHTML 1.1 в разделе 16.4.

16.1. Зачем нужен XHTML?

Как было сказано в предыдущей главе, HTML появился как простой язык разметки, похожий по виду и применению на другие, основанные на SGML языки разметки. В первые годы его жизни не прилагалось больших усилий к тому, чтобы он полностью соответствовал SGML. В результате разные странности и небрежное отношение к правилам стали непременной составляющей как HTML, так и броузеров, обрабатывающих HTML-документы.

По мере превращения Сети из эксперимента в индустрию потребность в стандартной версии HTML привела к созданию нескольких официальных версий, вершиной которых совсем недавно стал HTML 4.01. С тех пор как HTML устоялся в этой последней версии, броузеры стали ближе друг другу в том, что касается их поддержки различных возможностей HTML. В целом, мир HTML уложился в знакомое множество конструкций и грамматических правил.

К несчастью, HTML предлагает лишь ограниченное число примитивов для создания документов; он не умеет обращаться с нетрадиционным содержимым, таким как химические формулы, нотная запись или математические выражения, и ему не удается адекватно поддерживать альтернативные средства отображения, такие как маленькие портативные компьютеры или умные сотовые телефоны. Нам нужны новые способы передачи информации, с которой можно было бы делать все что угодно, – анализировать, обрабатывать, отображать ее – при помощи множества различных коммуникационных технологий, возникших с той поры, как десять лет назад Сеть высекла искру, возгоревшуюся пламенем революции в цифровых коммуникациях.

Вместо того чтобы пытаться обуздать еще один табун диких, нестандартных языков, W3C представил XML в качестве стандартного способа описания новых языков разметки. XML – это опорная конструкция, на основе которой организации могут разрабатывать свои собственные языки разметки, соответствующие потребностям их пользователей. XML – это обновленная версия SGML, рационализированная и усовершенствованная для нужд сегодняшних динамических систем. И хотя W3C изначально задумал XML как средство создания языков разметки документов, XML оказался весьма полезным и для определения микроскопических языков, используемых как протоколы обмена данными между различными приложениями.

Мы, разумеется, не хотим отказываться от огромного множества документов, уже размеченных при помощи HTML, или от инфраструктуры знаний, средств и технологий, которые в настоящее время поддерживают HTML и Сеть. Однако мы также не хотим упустить и возможности, которые дает нам XML. XHTML – вот связующее звено между ними. Он использует возможности XML для определения языка размет-

ки, который практически идентичен стандарту HTML 4.01 и в то же время позволяет нам вступить на путь, открываемый XML.

16.1.1. XHTML DTD

HTML 4.01 является нам в трех вариантах, каждый из которых определен в отдельном SGML DTD. Подобным образом существует и три варианта XHTML с XML DTD, соответствующими трем SGML DTD, которые определяют HTML 4.01. Для создания XHTML-документа необходимо выбрать одно из этих DTD, а затем создать документ, использующий его особые элементы и правила.

Первое XHTML DTD соответствует «строгому» (strict) HTML DTD. Строгое определение исключает все нежелательные элементы (теги и атрибуты) в HTML 4.01, заставляя авторов использовать только те возможности, которые полностью поддерживаются в HTML. Многие элементы HTML, касающиеся представления и внешнего вида, такие как тег `<font>` и атрибут `align`, отсутствуют в XHTML DTD, где они заменены эквивалентными свойствами модели каскадных таблиц стилей.

Большинство HTML-авторов находят строгое XHTML DTD слишком жестким, поскольку многие из нежелательных элементов и атрибутов до сих пор широко употребляются в Сети. Что еще важнее, огромный объем содержимого во Всемирной паутине основан на старых элементах и атрибутах, а популярные броузеры по-прежнему поддерживают морально устаревшие элементы. Единственное реальное преимущество использования строгого XHTML DTD состоит в том, что соответствующие ему документы в любом случае будут полностью поддерживаться в будущих версиях XHTML.[1]

Большинство авторов выберут, вероятно, «переходное» (transitional) XHTML DTD. Оно ближе всего к действующему стандарту HTML и включает все эти замечательные, но нежелательные возможности, которые облегчают жизнь HTML-авторам. Пользуясь переходным XHTML DTD, можно пристать к берегам XML, продолжая в то же время плыть по течению индустрии броузеров.

Третье DTD предназначено для фреймов (frameset). Во всех остальных отношениях оно тождественно переходному DTD. Единственное отличие состоит в замене тела документа подходящими фреймовыми элементами. Вы, может быть, подумаете, что по соображениям полноты должны существовать строгое и переходное фреймовые DTD, но W3C решил, что если уж использовать фреймы, то с тем же успехом можно употреблять и все нежелательные элементы.

---

[1] Если W3C будет твердо придерживаться избранного направления, HTML не изменится после версии 4.01. Больше никакого HTML – все дальнейшие разработки будут на XHTML и множестве других языков на основе XML.

16.2. Создание документов XHTML

По большей части, создание XHTML-документа ничем не отличается от создания HTML-документа. Просто вставьте в документ в вашем любимом текстовом редакторе элементы разметки в правильном порядке и потом откройте документ в своем любимом броузере. Чтобы быть строго корректным («действительным» (valid), как принято говорить в W3C), ваш XHTML-документ должен начинаться со стандартного объявления, где определены использованное при создании документа DTD и пространство имен документа.

16.2.1. Объявление типа документа

Чтобы XHTML-броузер верно проанализировал и отобразил ваш XHTML-документ, следует сообщить ему, какая версия XML используется при создании документа. Также нужно указать, в каком XHTML DTD определены элементы вашего документа.

Объявление версии XML использует специальную директиву XML-обработчика. Как правило, эти директивы начинаются с <? и завершаются ?>, а в остальном выглядят, как обычные теги вашего документа.[1] Чтобы объявить, что вы используете XML версии 1.0, поместите в первой строке документа следующую директиву:

```
<?xml version="1.0" encoding="UTF-8"?>
```

В ней броузеру сообщается, что вы используете XML 1.0 и восьмибитовое представление символов Unicode, которое чаще всего употребляется сегодня.[2] Значение атрибута `encoding` должно отражать ваш локальный набор символов. Обратитесь к соответствующим стандартам ISO за названиями других кодировок.

Управившись с важной задачей указания версии XML, следует затем объявить DTD языка разметки:

```
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

В этом предложении вы объявляете, что корневым элементом вашего документа является `html` в том виде, как он определен в DTD с публичным именем `"-//W3C//DTD XHTML 1.0 Strict//EN"`. Броузер, возможно, знает, как найти DTD, соответствующее этому публичному идентификатору. Если нет, он может воспользоваться URL, следующим за публичным идентификатором, в качестве альтернативного местоположения DTD.

---

[1] <! уже занято.

[2] Редактор, выбранный для подготовки текста XHTML-документа, должен уметь поддерживать именно ту кодировку, которая указана в `encoding`, а UTF-8 (Unicode) поддерживают далеко не все из них. – *Примеч. науч. ред.*

Как вы могли заметить, директива <!DOCTYPE> предписывала броузеру использовать строгое XHTML DTD. А вот та директива, которой вы, вероятно, воспользуетесь в своих «переходных» XHTML-документах:

```
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

И, как вы, наверное, и ожидали, директива <!DOCTYPE> для фреймовой версии XHTML будет:

```
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

16.2.2. Понятие пространства имен

Как говорилось в предыдущей главе, XML DTD определяет любое количество имен элементов и атрибутов как часть языка разметки. Эти имена элементов и атрибутов хранятся в уникальном для каждого DTD *пространстве имен* (*namespace*). Когда в своем документе вы ссылаетесь на элементы и атрибуты, броузер заглядывает в пространство имен, чтобы узнать, как их следует использовать.

Например, имя тега <a> (a) и такие атрибуты, как href и style, определены в XHTML DTD, и их имена находятся в пространстве имен DTD. Любой «обрабатывающий агент» – обычно это броузер[1], но той же цели могут служить ваши глаза и мозг – может найти это имя в соответствующем DTD и узнать, что именно этот элемент разметки означает и что он должен делать.

При работе с XML в документе можно в действительности использовать несколько DTD и, следовательно, несколько пространств имен. Можно, например, создать переходный XHTML-документ, но включить в него специальную разметку для математических выражений в соответствии с неким математическим XML-языком. Что же происходит, когда XHTML DTD и математическое DTD используют одно и то же имя для определения разных элементов, как, например, <a> для гиперссылок XHTML и <a> для абсолютных значений в математическом языке? Как броузер узнает, каким пространством имен он должен воспользоваться?

Ответ дает атрибут xmlns.[2] Пользуйтесь им для определения одного или нескольких пространств имен в ваших документах. Его можно помес-

---

[1] Применительно к XML это любая обрабатывающая программа, из которых броузеры – это лишь один класс таких программ, хотя и самый известный. – *Примеч. науч. ред.*

[2] Пространство имен XML (XML namespace) – xmlns – понимаете? Вот почему XML не разрешает начинать имена элементов и атрибутов с трехбуквенной приставки «xml» – она зарезервирована для специальных элементов и атрибутов XML.

тить в открывающий тег любого элемента в вашем документе, и его значение, похожее на URL[1], обозначит пространство имен, которое броузер должен использовать при обработке всего содержимого этого элемента.

В XHTML в соответствии с новыми правилами XML необходимо, по меньшей мере, включить атрибут `xmlns` в тег `<html>` документа, определяя таким образом первичное пространство имен, используемое при его обработке:

```
<html xmlns="http://www.w3.org/TR/xhtml1">
```

В тот момент, когда вам, возможно, понадобится математическая разметка, вы снова примените атрибут `xmlns`, чтобы определить математическое пространство имен. Так, например, можно было бы использовать атрибут `xmlns` в каком-нибудь специфически математическом теге обычного в других отношениях XHTML-документа (предполагая, конечно, что этот элемент имеется в области `MATH`):

```
<div xmlns="http://www.w3.org/1998/Math/MathML>x2/x</div">
```

В этом случае XML-совместимый броузер, используя пространство имен *http:// www.w3.org/1998/Math/MathML*, догадывается, что это математическая, а не XHTML-версия тега `<div>` и, следовательно, его содержимое должно быть отображено как выражение деления.

Вам очень скоро наскучило бы вставлять атрибут `xmlns` во все теги `<div>` всякий раз, когда захочется вставить в документ формулу с делением. Будет лучше – особенно если вы намерены применять ее ко многим различным элементам вашего документа – определить пространство имен в начале документа, снабдив его меткой, а затем ссылаться на это пространство имен, включая метку в виде приставки к имени соответствующего элемента. Например:

```
<html xmlns="http://www.w3.org/TR/xhtml1"
      xmlns:math="http://www.w3.org/1998/Math/MathML">
```

Область имен `math` теперь может упоминаться в вашем документе под сокращенным именем «math». Так что упрощенная форма:

```
<math:div>x2/x</div>
```

будет иметь тот же эффект, что и длинная, использовавшая атрибут `xmlns` версия тега `<div>` из предыдущего примера.

Большинству XHTML-авторов придется определять многочисленные пространства имен, и, таким образом, не придется использовать полностью определенные названия, содержащие префиксы пространств имен. Но даже если это и так, вы должны понимать, что множествен-

---

[1] Оно выглядит как URL, и можно подумать, что оно указывает на документ, содержащий пространство имен, но, увы, это не так. Это просто уникальное имя, идентифицирующее пространство имен. Отображающий агент использует его для ссылки на свои собственные ресурсы, содержащие информацию о том, как следует интерпретировать то или иное имя элемента или атрибута.

ные пространства имен существуют и что вам понадобится управляться с ними, если вы захотите вложить содержимое, описанное одним DTD, в содержимое, описанное другим.

16.2.3. Минимальный XHTML-документ

Чтобы помочь начинающим XHTML-авторам, мы представим здесь минимальный и корректный XHTML-документ, включающий все необходимые объявления XML, XHTML и пространства имен. После того как эта самая трудная часть работы проделана, вам понадобится только вставить содержимое, чтобы получить завершенный XHTML-документ:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
          PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
          "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/TR/xhtml1" xml:lang="en" lang="en">
  <head>
    <title>Каждый документ должен иметь заголовок</title>
  </head>
  <body>
    ...сюда поместите содержимое документа...
  </body>
</html>
```

Рассматривая минимальный документ поэлементно, мы увидим, что он начинается с указания на то, что мы основываем этот документ на стандарте XML 1.0 и используем восьмибитовое представление (UTF-8) Unicode для записи его содержимого и разметки. Затем в знакомом, похожем на HTML предложении `<!DOCTYPE>` мы объявляем, что следуем правилам разметки, определенным в переходном XHTML 1.0 DTD, которое позволяет нам употреблять в документе практически любые элементы HTML 4.01.

Действительное содержимое документа начинается с тега `<html>`, который содержит атрибут `xmlns`, объявляющий, что пространство имен XHTML будет применяться по умолчанию для всего документа. Обратите также внимание на атрибут `lang` в пространстве имен XML и XHTML, объявляющий, что языком документа будет английский.

Наконец, мы включаем знакомые теги документа `<head>` и `<body>`, равно как и обязательный тег `<title>`.

16.3. HTML и XHTML

Большая часть HTML полностью совместима с XHTML, и наша книга посвящена как раз этой части. В этой главе, однако, мы говорим о меньшей части, где стандарты HTML 4.01 и XHTML DTD расходятся. Если вы на самом деле хотите создавать документы, соответствующие стандартам HTML 4 и XHTML, вы должны учесть разнообразные

предупреждения и предостережения, которые найдете в нижеследующих разделах.

Самое большое различие – Различие с большой буквы, обещающее немалые трудности, – состоит в том, что написание XHTML-документов требует дисциплины и внимания, какие и не снились самым дотошным HTML-авторам. На языке W3C это называется безупречной «корректностью» («well-formed»). На протяжении всей истории HTML – и этой книги – авторов поощряли к созданию корректных документов, но для того чтобы ваши документы можно было рассматривать как корректные с точки зрения стандартов XML, вам понадобится порвать со стандартами HTML.

Ваши усилия по овладению XHTML тем не менее будут вознаграждены корректностью документов и чувством удовлетворения от игры по всем правилам. Вы также получите реальный выигрыш в будущем – при посредстве XML ваши документы смогут появиться в таких местах, о каких вы и не мечтали (большей частью, надеемся, в хороших местах).

16.3.1. Корректно вложенные элементы

Одно из требований корректности XHTML-документов состоит в том, что элементы должны вкладываться корректно. Здесь нет ничего отличного от стандарта HTML – просто закрывайте все элементы разметки в порядке, соответствующем порядку их открытия. Если один элемент содержится в другом, завершающий тег внутреннего элемента должен появиться перед завершающим тегом внешнего.

Таким образом, в следующем корректном XHTML-фрагменте мы завершаем тег курсива перед завершением тега жирного шрифта, поскольку мы открывали тег курсива после тега жирного шрифта:

```
<b>Закройте тег курсива <i>в первую очередь</i></b>.
```

Напротив, следующий пример:

```
<b>Такое завершение тегов <i>некорректное!</b></i>
```

не является корректным.

XHTML строго настаивает на соблюдении других ограничений, касающихся вложения тегов. Эти ограничения всегда были частью HTML, но не всегда обязательно соблюдались. Эти ограничения с формальной точки зрения не входят в XHTML DTD, но они определены как часть стандарта XHTML, основанного на этом DTD.[1]

[1] Это тонкость стандарта XHTML. В XML нет механизма, определяющего, какие теги нельзя помещать в другие теги. В SGML, на котором базируется XML, такие средства есть, но они исключены из XML, чтобы сделать последний более легким для употребления и программной реализации. В результате эти ограничения просто перечислены в приложении стандарта XHTML, а не выражены явно в XHTML DTD.

К ограничениям вложенности относятся следующие:

- Тег <a> не может содержать другой тег <a>.
- Тег <pre> не может содержать теги <img>, <object>, <big>, <small>, <sub> или <sup>.
- Тег <button> не может содержать теги <input>, <select>, <textarea>, <label>, <button>, <form>, <fieldset>, <iframe> или <isindex>.
- Тег <label> не может содержать теги <label>.
- Тег <form> не может содержать теги <form>.

Эти ограничения применяются к вложениям на любую глубину. К примеру, тег <a> не может содержать другие теги <a>, равно как и иные теги, которые, в свою очередь, содержат тег <a>.

16.3.2. Закрывающие теги

Как подчеркивалось на протяжении всей книги, каждый тег HTML, заключающий в себе содержимое или другие теги, имеет соответствующий закрывающий тег. Однако характерной чертой HTML (в соответствии со стандартом 4.01) является то, что разрешается опускать закрывающие теги, если их наличие может быть логически выведено обрабатывающим агентом. Вот почему большинство из нас, пишущих на HTML, обычно опускают закрывающий тег </p> между соседними абзацами. Списки и таблицы, освобожденные от всех этих , </td>, </th> и </tr>, продираться через которые было бы нелегко, тоже делают HTML несколько яснее и легче для чтения.

В XHTML не так. Каждый тег, содержащий другой тег или другое содержимое, должен иметь закрывающий тег, явно выписанный с учетом правил вложенности тегов. Отсутствующий закрывающий тег – это ошибка, из-за которой документ будет признан несоответствующим стандарту. Хотя эта мера кажется драконовской, в сочетании с правилами вложенности она исключает любую двусмысленность по поводу того, где начинается один тег и заканчивается другой.

16.3.3. Пустые элементы

В XML и, следовательно, в XHTML у каждого тега должен быть соответствующий закрывающий тег – даже у тех, которые не могут содержать что-либо. Таким образом, XHTML ожидает, что разрыв строки будет обозначен в вашем документе в виде
</ br>. Вот так.

К счастью, существует приемлемая альтернатива – вставить перед завершающей угловой скобкой тега наклонную черту, чтобы обозначить его конец, как, например, в
. Если у тега есть атрибуты, наклонная черта пишется после атрибутов, так что рисунок может быть определен так:

```
<img src="kumquat.gif" />
```

Хотя такое обозначение может выглядеть как написанное на иностранном языке и вызывать у HTML-авторов раздражение, оно действительно служит полезной цели. Любой элемент XHTML, не имеющий содержимого, может быть записан таким образом. Так что пустой абзац можно записать как <p />, а пустую ячейку таблицы – как <td />. Это удобный способ отмечать пустые ячейки таблицы.

Хотя такой способ записи пустых тегов кажется разумным, он может смутить HTML-броузер.

Поэтому, чтобы избежать проблем с совместимостью, вы можете перехитрить HTML-броузер, поместив при обозначении пустого элемента по версии XHTML пробел перед завершающим слэшем. Используйте, например,
 с пробелом между br и / вместо его XHTML-эквивалентов
 и
</br>. В табл. 16.1 представлены все пустые HTML-теги, выраженные в их приемлемой XHTML-форме (переходное DTD).

Таблица 16.1. Пустые теги HTML в формате XHTML

<area />	<base />	<basefont />
 	<col />	<frame />
<hr />	<img />	<input />
<isindex />	<link />	<meta />
<param />		

16.3.4. Зависимость от регистра

Если вы думали, что написание XHTML-документов делают трудным все эти обязательные закрывающие теги и необходимость расчистки всех случайных ошибок при записи вложения тегов, то сейчас держитесь за стул. Значения *всех имен тегов и атрибутов в XHTML зависят от регистра. В XHTML-документе* <a> и <A> – это разные теги, src и SRC – разные атрибуты. Точно так же, как и sRc и Src! Каким добрым и снисходительным теперь кажется HTML.

XHTML DTD определяет все теги и атрибуты, заимствованные из HTML, используя строчные буквы. Записанные заглавными буквами имена тегов и атрибутов не являются допустимыми тегами и атрибутами XHTML.

Это может поставить в затруднительное положение любого автора, желающего привести уже существующие HTML-документы в соответствие со стандартом XHTML. Во множестве веб-страниц используются теги и атрибуты, записанные заглавными буквами, чтобы выделить их из окружающего содержимого, записанного строчными буквами. Чтобы сделать их соответствующими XHTML, все эти имена необходимо преобразовать в строчные. Даже те, что вы использовали в своих определениях таблиц стилей CSS. К счастью, исправления такого рода легко выполнить при помощи разных средств редактирования. А специа-

лизированные редакторы XHTML должны выполнять такое преобразование сами.

16.3.5. Кавычки в значениях атрибутов

И, как будто всех этих чувствительных к регистру имен атрибутов еще недостаточно для усугубления трудностей языка, XHTML требует вдобавок, чтобы все значения атрибутов – даже числовые – были заключены в двойные кавычки. В HTML можно заключать в кавычки все, что вздумается, но обязательны кавычки только в случае, если значения атрибутов включают в себя пробелы или другие специальные символы. XHTML требует, чтобы все значения атрибутов были заключены в кавычки. К примеру,

```
<table rows=3>
```

в XHTML будет неправильным. Корректно это записывается так:

```
<table rows="3">
```

16.3.6. Явное указание значений атрибутов

В HTML имеется небольшое количество атрибутов без значений. При этом само присутствие этих атрибутов модифицирует поведение тега. Они представляют собой нечто вроде переключателей вкл./выкл., как, например, атрибут `compact` в различных тегах списков или атрибут `ismap` в теге `<img>`.

В XHTML значение обязательно для каждого атрибута. Атрибуты без значений должны использовать в качестве таковых собственные имена. Таким образом, `compact` в XHTML корректно записывается в виде `compact="compact"`, а вместо `checked` – `checked="checked"`. Каждый атрибут теперь имеет заключенное в кавычки значение. Атрибуты, получившие такие вот новые значения, представлены в табл. 16.2.

Таблица 16.2. Новые XHTML-значения атрибутов для атрибутов HTML без значений

`checked="checked"`	`compact="compact"`	`declare="declare"`
`defer="defer"`	`disabled="disabled"`	`ismap="ismap"`
`multiple="multiple"`	`noresize="noresize"`	`noshade="noshade"`
`nowrap="nowrap"`	`readonly="readonly"`	`selected="selected"`

Учтите, что новые значения атрибутов в XHTML могут заставить некоторые старые HTML-броузеры проигнорировать атрибут совсем. У современных броузеров таких проблем не будет, так что большинство пользователей не заметят никаких отличий. У этой проблемы нет лучшего решения, кроме как снабдить всех нуждающихся броузерами, которые поддерживали бы HTML 4.0.

16.3.7. Как обращаться со специальными символами

XHTML гораздо чувствительнее, чем HTML, к употреблению в ваших документах символов < и & в объявлениях JavaScript и CSS. В HTML, чтобы избежать возможных конфликтов, сценарии и таблицы стилей заключают в комментарии (<!-- и -->). XML-броузеры, однако, могут просто удалить все содержимое комментариев из документа вместе со всеми скрытыми сценариями и таблицами стилей.

Чтобы как следует защитить специальные символы от XML-броузеров, заключайте ваши стили или сценарии в секции CDATA. Таким образом XML-броузеру сообщается, что все содержащееся внутри – это просто символы без всяких специальных значений. Например:

```
<script language="JavaScript">
<![CDATA[
   Здесь помещаем операторы JavaScript...
 ]]>
</script>
```

Это, однако, не решает проблему. HTML-броузеры игнорируют содержимое XML-тега CDATA, но учитывают содержимое сценариев и таблиц стилей, заключенное в кавычки, тогда как XML-броузеры делают в точности противоположное. Мы советуем помещать сценарии и стили во внешние файлы и ссылаться на них в документах при помощи соответствующих внешних ссылок.

Специальные символы в значениях атрибутов тега также представляют проблему для XHTML. В частности, амперсанд в значениях атрибутов должен всегда записываться как &, а не просто как символ &. Подобным же образом, проявляя осторожность, следует кодировать знаки «меньше» и «больше», заменяя их на < и >. Например, хотя

```
<img src=seasonings.gif alt="Salt & pepper">
```

совершенно законно в HTML, нужно переписать это как:

```
<img src="seasonings.gif" alt="Salt &amp; pepper" />
```

чтобы все согласовалось с XHTML.

16.3.8. Атрибуты id и name

В ранних версиях HTML атрибут name употреблялся с тегом <a> для создания идентификатора фрагмента документа. Этот идентификатор мог затем использоваться в URL для указания на определенное место документа. Впоследствии атрибут name был добавлен к другим тегам, таким как <frame> и <img>, что позволило также ссылаться на имена этих элементов из других мест документа.

В HTML 4.0 W3C добавил почти ко всем тегам атрибут id. Как и name, id позволяет почти с любым элементом документа ассоциировать идентификатор для последующих ссылок и использования его, скажем, в гиперссылках или сценариях.

XHTML явно отдает предпочтение атрибуту id. Атрибут name определен для тех элементов, которые исторически использовали его, но формально нежелателен. Поскольку в настоящее время широко поддерживается HTML 4.0, лучше избегать атрибута name, где только это возможно, и вместо него присваивать имена элементам документов, пользуясь атрибутом id. Если в определенных тегах вам придется использовать атрибут name, включайте в них идентичный атрибут id – это гарантирует, что тег поведет себя так же, когда будет обрабатываться XHTML-броузером.

16.4. XHTML 1.1

В мае 2001 года консорциум W3C выпустил обновленный стандарт XHTML 1.1. В то время как большинство стандартов расширяется от версии к версии, в случае с XHTML 1.1 был предпринят необычный шаг по выпуску более строгой подверсии. Если вы считаете XHTML 1.0 громоздким, придирчивым и отнимающим у вас драгоценное время, то вы обнаружите, что XHTML 1.1 обладает этими качествами в еще большей степени. По нашему мнению, XHTML 1.1 является примером выхода процесса стандартизации на абсурдный уровень. Возможно, стандарт и стал рафинированным с академической точки зрения, но пользоваться им практически невозможно.

16.4.1. Нововведения в XHTML 1.1

Стандарт XHTML 1.1 начинается со строгого DTD-описания стандарта XHTML 1.0 и внесения некоторых модификаций. Поддерживая строгую версию XHTML 1.0, версия 1.1 исключает все нежелательные элементы и все броузерные расширения, которые все еще широко распространены во Всемирной паутине. Кроме того, были внесены следующие, не столь значительные изменения:

- атрибут lang был убран из всех элементов; вместо него теперь следует использовать атрибут xml:lang;
- атрибут name убран из элементов <a> и <map>; ему на смену пришел атрибут id.

Наконец, в стандарте XHTML 1.1 определен новый набор элементов, реализующих одну особенность типографских документов, называемую «рубин». Текст «рубин» – это небольшие полосы текста, размещенные рядом с основным текстом и используемые для комментариев или уточнения произношения. Такая манера оформления документов пришла с Востока и типична для китайских учебников и японских книг и газет. Текст «рубин»[1] обычно печатается более мелким шрифтом, чем

[1] Термин «рубин» заимствован из полиграфии, где этим словом обозначается шрифт размером 5,5 пункта, используемый в английских газетах для таких примыкающих полос текста.

основной, и следует определенным правилам выравнивания, гарантирующим, что он будет находиться рядом с соответствующим элементом основного текста.

Вы можете определять текст «рубин» и управлять им с помощью набора элементов, обеспечивающих группирование и компоновку. Скажем без обиняков: эта новая функциональная возможность настолько вычурна и неинтересна основной массе HTML-авторов (даже тех, кто обречет себя на соблюдение стандарта XHTML 1.1), что она не заслуживает подробного обсуждения в этой книге. Интересующийся читатель найдет подробную информацию о тексте «рубин» по адресу *http://www.w3.org/TR/ruby*.

А остальным читателям достаточно знать, что в XHTML 1.1 появилось несколько новых элементов, которые лучше не использовать в своих DTD-документах, разве что в тех случаях, когда вы хотите избежать путаницы с DTD XHTML 1.1. Новый набор элементов выглядит так:

`<ruby>`

Определяет сегмент текста «рубин».

`<rb>`

Определяет основной текст для «рубина».

`<rt>`

Определяет текст «рубин», ассоциированный с основным.

`<rp>`

Используется в качестве «скобок рубин» для группирования элементов «рубин».

`<rbc>`

Служит контейнером основного текста и группирует его элементы.

`<rtc>`

Служит контейнером текста «рубин» и группирует его элементы.

Когда вы встретите один из этих элементов в каком-нибудь документе, обратитесь к вышеупомянутой спецификации, если вам нужно разобраться в тонкостях их употребления. Вообще говоря, вы обнаружите один внешний элемент `<ruby>`, содержащий хотя бы по одному элементу `<rb>` и `<rt>`. Несколько элементов `<rb>` и `<rt>` могут быть сгруппированы внутри элемента `<rp>` или внутри контейнерного элемента `<rbc>` или `<rtc>`.

16.5. Использовать ли XHTML?

С точки зрения авторов, привыкших к HTML, XHTML, очевидно, требует больше хлопот и ведет себя менее великодушно. Если мы некогда гордились способностью набело писать HTML при помощи карандаша и бумаги, то создание XHTML-документов без помощи специальных

приложений для их подготовки – гораздо более скучное занятие. Зачем кому-то понадобится взваливать на себя этот лишний груз?

16.5.1. Проблема пыльных полок

Всего лишь за несколько лет Сеть заполнилась миллиардами страниц. Можно смело биться об заклад, что многие из этих страниц не согласуются ни с одной из определявшихся версий HTML. С еще большей уверенностью можно утверждать, что подавляющее большинство их не согласуются с XHTML.

Горькая правда в том, что эти миллиарды страниц никогда не будут преобразованы в XHTML. У кого найдется время, чтобы вернуться, раскопать эти старые страницы и переделать их по стандарту XHTML, особенно если конечный результат с точки зрения пользователей не изменится? Подобно пыльным полкам с программами на COBOL, десятилетия пролежавшими нетронутыми, пока, наконец, проблема Y2K не заставила программистов чихать, извлекая их, – точно так же и эти пыльные полки с веб-страницами останутся в неприкосновенности до тех пор, пока какое-нибудь столь же драматическое событие не заставит обновить их.

Тем не менее проблема пыльных полок – это не основание для того, чтобы и впредь отказываться писать документы, согласующиеся со стандартом. Оставим старые документы в покое, но не будем создавать с каждым новым документом очередную проблему преобразования. Немного старания – и ваши документы смогут работать в будущем на еще большем количестве броузеров.

16.5.2. Автоматическое преобразование

Если чувство ответственности подталкивает вас к тому, чтобы заняться преобразованием ваших уже существующих HTML-документов в XHTML, вы найдете, что утилита Tidy исключительно полезна для этой цели. Написанная Дэйвом Реггетом (Dave Raggett), одним из активных деятелей W3C, она автоматизирует значительную часть работы, требующейся для преобразования HTML-документов в XHTML.

Хотя возможности Tidy слишком разнообразны и удивительны, чтобы их полностью здесь перечислять, все же заверим вас, что во всяком случае Tidy отслеживает и корректирует использование регистров, наличие кавычек в значениях атрибутов и правильность вложения элементов. Полный перечень возможностей и последнюю версию Tidy для любых компьютерных платформ можно найти на *http://tidy.sourceforge.net*.

16.5.3. Ленивые авторы и снисходительные броузеры

Есть один принцип, касающийся обмена данными, особенно в Интернете, – быть снисходительным к тому, что вы получаете, и строгим – к тому, что выдаете. Это не рассуждение о социальной политике, а ско-

рее прагматический совет относиться терпимо к двусмысленностям и ошибкам в получаемых данных и вместе с тем стремиться, чтобы все, что вы посылаете, было точным до мелочей.

Броузеры являются хорошим примером терпимости к полученным данным. Большинство современных веб-страниц содержат какие-нибудь ошибки, хотя часто это всего лишь пропуски. Тем не менее броузеры принимают эти ошибочные тексты и представляют пользователям разумный результат. Благодаря этой снисходительности авторам все сходит с рук, и они подчас даже не знают, что допустили ошибку.

Большинство авторов прекращают работать над страницей, когда она уже хорошо выглядит и делает то, чего они от нее хотят. Очень немногие из них тратят время на то, чтобы в поисках возможных ошибок просмотреть свои страницы на разных средствах проверки грамматики HTML. Многие из тех, кто пробовал проверить, соответствуют ли их работы правилам языка, обнаруживали такое множество совершенных ими мелких ошибок, что они просто сдавались и продолжали создавать плохие страницы, которые сносно выводят добрые броузеры.

Поскольку количество плохих страниц продолжает расти, броузеры не могут позволить себе стать строгими. Пользователи будут избегать любых броузеров, которые попытаются настаивать на соблюдении даже самых основных правил стандарта HTML – пользователи хотят видеть веб-страницы, а не сообщения об ошибках. Порочный круг замыкается – плохие страницы заставляют использовать снисходительные броузеры, стимулирующие создание плохих страниц. Разорвите этот круг, поклявшись в том, что всегда, когда только сможете, вы будете создавать лишь документы, соответствующие правилам XHTML.

16.5.4. Время, деньги и стандарты

XHTML разрабатывался как представление стандарта HTML средствами XML. Если заглянуть вперед, то он задумывался с целью стать единственным стандартом, которому должны были бы следовать все, кто создает содержимое Сети.

Если бы мир был совершенным, все принимали бы стандарты и пользовались ими. Полное соответствие стандарту было бы обязательным для всех размещаемых в Сети документов. Преобразование документов, унаследованных с «достандартных» времен, происходило бы незамедлительно.

Но в реальном мире дефицит времени и денег препятствует тому, чтобы стандарты стали всеобщими. Необходимо быстро предоставить что-нибудь работающее – и разработчики выпускают страницы, которые едва работают. Поскольку броузеры мирятся с существованием в Сети второсортного содержимого, необходимость соответствия стандартам отодвигается на второй план, который слишком быстро выпадает из поля зрения при стремительном развитии сети.

16.5.5. Человек против машины

Не все, однако, потеряно. Хотя людям трудно и скучно писать на XHTML, для машин это очень легко. Постоянно растет количество средств разработки веб-страниц, и составленные с их помощью документы должны абсолютно соответствовать XHTML. Хотя в необходимости тратить время на правильную расстановку всех этих завершающих тегов не видно большого экономического смысла, тем не менее кажется разумным, чтобы программист, разрабатывающий специализированный редактор для создания веб-страниц, непременно позаботился о том, чтобы его программа правильно расставляла эти завершающие теги. Усилия, затраченные автором страницы, окупаются однократно, а усилия создателя программы – всякий раз, когда с ее помощью создается очередной документ.

Нам представляется, что действительное будущее XHTML лежит в сфере автоматической генерации содержания. XHTML слишком придирчив, чтобы им могли успешно пользоваться миллионы обычных авторов, создающих маленькие веб-сайты. Но если эти же авторы будут делать свои страницы при помощи подходящего приложения, они смогут создавать страницы, соответствующие стандарту XHTML, даже не задумываясь об этом.

Если вы принадлежите к немногочисленному сообществу создателей программ, генерирующих вывод в виде HTML, то вы крайне плохо обслуживаете множество ваших возможных клиентов, если вывод вашей программы не согласуется в точности со стандартом XHTML. Не существует никаких технических оснований для того, чтобы какое-либо приложение не могло бы генерировать согласующийся с XHTML вывод. Если речь идет о совместимости, то есть о том, как этот вывод могут использовать, скажем, броузеры, не поддерживающие XHTML, то пусть в программе будет опция, позволяющая авторам в качестве варианта выбирать согласующийся с XHTML вывод.

16.5.6. Что же делать?

Мы советуем всем HTML-авторам потратить время на то, чтобы усвоить изложенные в этой главе различия между HTML и XHTML. При наличии ресурсов и подходящего случая надо стараться, чтобы страницы ваших сайтов согласовались с XHTML. Вы, разумеется, должны выбирать специализированные средства создания страниц, поддерживающие XHTML и позволяющие вам создавать такие страницы.

Рано или поздно XHTML может заменить HTML в качестве официального стандарта в Сети. Но в любом случае Сеть останется переполненной не согласующимися со стандартом страницами, что заставит броузеры в течение ближайших пяти-десяти лет уважать старые HTML-конструкции. Хорошо это или плохо, но на годы вперед HTML останется *фактическим* стандартом для сетевых авторов.

- Главный совет
- Доработка документа после HTML-редактора
- Трюки с таблицами
- Фокусы с окнами и фреймами

17

Трюки, штуки и советы

На страницах этой книги мы рассыпали множество подсказок, фокусов и обходных маневров наряду с советами относительно стиля, примерами и инструкциями. Зачем нужна отдельная глава с советами, фокусами и обходными маневрами? Потому что HTML и XHTML – это языки, которые при всей своей ограниченности сделали Сеть таким волнующим местом. И заинтересованные читатели хотят знать: «Могу ли я сделать не хуже?»

17.1. Главный совет

Самый важный совет, который можно дать даже опытным авторам, это самостоятельно исследовать Сеть. Мы можем показать и объяснить несколько изящных трюков, чтобы помочь вам начать, но сотни тысяч авторов в Сети составляют и пересоставляют теги HTML и XHTML, и перетряхивают содержимое документов, стараясь сделать их краше и полезнее.

Все популярные броузеры позволяют просматривать исходный код загруженных страниц. Изучайте (но не воруйте) их страницы на предмет привлекательности и эффективности и используйте их в своих творениях. Получите впечатление от самых эффективных веб-коллекций. Как документы в них организованы? Какой размер имеют отдельные документы? И так далее.

Мы все узнали из опыта, теперь настала ваша очередь.

17.1.1. Дизайн с оглядкой на аудиторию

На протяжении всей этой книги мы постоянно утверждаем, что важнее всего содержание, а не внешний вид. Это вовсе не значит, что оформление не играет никакой роли.

Эффективные документы отвечают ожиданиям вашей целевой аудитории, предоставляя ей в процессе поиска и сбора информации знакомое окружение. Серьезные ученые, например, ожидают, что трактат по физиологии кумкватов будет иметь вид научного журнала, полного многозначительных слов, рисунков, диаграмм, и содержать минимум легкомысленных украшательств, таких как вычурные маркеры или эпатирующий шрифт. Не оскорбляйте зрение читателей, если только вы не хотите поразить их чувства или произвести на них яркое художественное впечатление. Представив себе свою аудиторию и конструируя документы, отвечающие ее вкусам, вы также мягко оттолкнете от своих страниц нежелательных посетителей.

Можно, например, использовать приглушенные цвета и деликатные вариации шрифтов от раздела к разделу, представляя коллекцию художественного музея, чтобы сымитировать тихую обстановку в настоящем музее классического искусства. Типичный сетевой маньяк с рок-н-роллом в голове, вероятно, только взглянет на нее и уйдет, а миллионер-меценат, возможно, останется.

Используйте также эффектное макетирование, чтобы направлять внимание ваших читателей к интересным частям вашего документа. Делайте это, придерживаясь основных правил макетирования и конструирования документов, таких как размещение рисунков и диаграмм рядом (если не в строку) с относящимся к ним текстом. Ничего нет хуже, чем прокручивать окно броузера вверх и вниз в безнадежных поисках картинки, которая наконец все объяснит.

Мы не станем лгать, уверяя вас, что мы специалисты по дизайну. Мы – нет, но их нетрудно найти. Так что вот еще один совет серьезным веб-авторам: обращайтесь за помощью к профессионалам. Лучше всего, если у вас есть собственный дизайнерский опыт. Хорошо иметь кого-то, кто заглядывал бы вам через плечо, на худой конец, того, кому можно позвонить.

Сходите в местную библиотеку и почитайте там что-нибудь сами. Еще лучше – просмотрите разные руководства в Интернете. Загляните в книгу Jennifer Niederst Robbins «Web Design in a Nutshell» издательства O'Reilly. Вашим читателям это пойдет на пользу. [инструменты веб-дизайнера, 1.6]

17.1.2. Единообразие документов

Следующий по важности совет, который мы можем вам дать, это повторно использовать ваши документы. Не начинайте каждый раз с чистого листа. Наоборот, разработайте стандартный скелет текста вплоть до контуров содержимого, на котором вы сможете размещать подробности и которому сможете придать затем окончательный вид. Вы можете даже попытаться создать таблицы стилей на базе CSS2, чтобы сделать внешний вид и поведение всех ваших страниц единообразными.

17.2. Доработка документа после HTML-редактора

Хотя вы можете создавать и редактировать HTML/XHTML-документы в текстовом редакторе, таком как vi или Notepad, большинство авторов HTML-документов использует какое-нибудь приложение, предназначенное специально для разработки веб-страниц. Некоторые из таких приложений бесплатны, многие имеют бесплатные ознакомительные версии, и почти все могут быть загружены из Всемирной паутины. И все-таки мы на основании своего опыта хотим вас предупредить: вам редко (если когда-либо) удастся создать с помощью одного из этих редакторов веб-документ, не требующий проверки, добавлений, редактирования и даже устранения ошибок из сгенерированного HTML-кода. В следующих разделах мы обсудим некоторые важные моменты, о которых вы должны знать, и которые вы всегда должны принимать во внимание.

17.2.1. Во что превратился мой документ?

Одно из первых наблюдений, которое вы сделаете, будет заключаться в том, что многие HTML-редакторы автоматически вставляют в ваш документ разметку, которую вы никак не запрашивали. Помните простенький HTML-документ, с которого мы начали в гл. 2?

```
<html>
<head>
<title>My first HTML document</title>
</head>
<body>
<h2>My first HTML document</h2>
Hello, <i>World Wide Web!</i>
 <!-- No "Hello, World" for us -->
<p>
Greetings from<br>
<a href="http://www.ora.com">O'Reilly Media</a>
<p>
Composed with care by:
<cite>(insert your name here)</cite>
<br>&copy;2000 and beyond
</body>
</html>
```

Вот как он будет выглядеть, когда вы загрузите его в Microsoft Word из Office XP:

```
<html xmlns:o="urn:schemas-microsoft-com:office:office"
xmlns:w="urn:schemas-microsoft-com:office:word"
xmlns="http://www.w3.org/TR/REC-html40">

<head>
<meta http-equiv=Content-Type content="text/html; charset=windows-1252">
<meta name=ProgId content=Word.Document>
```

```
<meta name=Generator content="Microsoft Word 10">
<meta name=Originator content="Microsoft Word 10">
<link rel=File-List href="html_files/filelist.xml">
<title>&lt;html&gt;</title>
<!--[if gte mso 9]><xml>
 <w:WordDocument>
  <w:Compatibility>
   <w:BreakWrappedTables/>
   <w:SnapToGridInCell/>
   <w:WrapTextWithPunct/>
   <w:UseAsianBreakRules/>
  </w:Compatibility>
  <w:BrowserLevel>MicrosoftInternetExplorer4</w:BrowserLevel>
 </w:WordDocument>
</xml><![endif]-->
<style>
<!--
 /* Style Definitions */
 p.MsoNormal, li.MsoNormal, div.MsoNormal
    {mso-style-parent:"";
    margin:0in;
    margin-bottom:.0001pt;
    mso-pagination:widow-orphan;
    font-size:12.0pt;
    font-family:"Times New Roman";
    mso-fareast-font-family:"Times New Roman";}
p.MsoPlainText, li.MsoPlainText, div.MsoPlainText
    {margin:0in;
    margin-bottom:.0001pt;
    mso-pagination:widow-orphan;
    font-size:10.0pt;
    font-family:"Courier New";
    mso-fareast-font-family:"Times New Roman";}
@page Section1
    {size:8.5in 11.0in;
    margin:1.0in 65.95pt 1.0in 65.95pt;
    mso-header-margin:.5in;
    mso-footer-margin:.5in;
    mso-paper-source:0;}
div.Section1
    {page:Section1;}
-->
</style>
<!--[if gte mso 10]>
<style>
 /* Style Definitions */
 table.MsoNormalTable
    {mso-style-name:"Table Normal";
    mso-tstyle-rowband-size:0;
    mso-tstyle-colband-size:0;
    mso-style-noshow:yes;
```

```
    mso-style-parent:"";
    mso-padding-alt:0in 5.4pt 0in 5.4pt;
    mso-para-margin:0in;
    mso-para-margin-bottom:.0001pt;
    mso-pagination:widow-orphan;
    font-size:10.0pt;
    font-family:"Times New Roman";}
</style>
<![endif]-->
</head>

<body lang=EN-US style='tab-interval:.5in'>

<div class=Section1>

<p class=MsoPlainText>&lt;html&gt;<o:p></o:p></p>

<p class=MsoPlainText>&lt;head&gt;<o:p></o:p></p>

<p class=MsoPlainText>&lt;title&gt;My first HTML document&lt;/
title&gt;<o:p></o:p></p>

<p class=MsoPlainText>&lt;/head&gt;<o:p></o:p></p>

<p class=MsoPlainText>&lt;body&gt;<o:p></o:p></p>

<p class=MsoPlainText>&lt;h2&gt;My first HTML
                                    document&lt;/h2&gt;<o:p></o:p></p>

<p class=MsoPlainText>Hello, &lt;i&gt;World Wide Web!&lt;/i&gt;<o:p></o:p></p>

<p class=MsoPlainText><span style='mso-spacerun:yes'> </span>&lt;!-- No
&quot;Hello, World&quot; for us --&gt;<o:p></o:p></p>

<p class=MsoPlainText>&lt;p&gt;<o:p></o:p></p>

<p class=MsoPlainText>Greetings from&lt;br&gt;<o:p></o:p></p>

<p class=MsoPlainText>&lt;a href=&quot;http://www.ora.com&quot;&gt;O'Reilly
Media&lt;/a&gt;<o:p></o:p></p>

<p class=MsoPlainText>&lt;p&gt;<o:p></o:p></p>

<p class=MsoPlainText>Composed with care by: <o:p></o:p></p>

<p class=MsoPlainText>&lt;cite&gt;(insert your name here)&lt;/
cite&gt;<o:p></o:p></p>

<p class=MsoPlainText>&lt;br&gt;&amp;copy;2000 and beyond<o:p></o:p></p>

<p class=MsoPlainText>&lt;/body&gt;<o:p></o:p></p>

<p class=MsoPlainText>&lt;/html&gt;</p>

</div>

</body>

</html>
```

Что произошло? Во что превратился документ? Избыточная разметка сделала его абсолютно нечитаемым. Особенно возмущает таких ревностных сторонников чистоты документа, как мы, во-первых, тот факт, что была добавлена куча вещей, о которых мы не просили и которые нам не нужны, а во-вторых, что Word автоматически считает любой текстовый документ с HTML-разметкой сырьем для своей фабрики. Вы можете убрать расширение html или htm из имени файла, вы можете убрать теги `<html>` и `<head>` из документа, – все бесполезно. Word вас достанет.

Microsoft Word – не единственный редактор, засоряющий исходный код. Большинство HTML-редакторов, как минимум, добавляет тег `<meta>` с информацией о самом редакторе. Многие идут дальше и «подправляют» ваш документ в соответствии с текущими стандартами и общепринятой практикой. Например, они добавляют все эти закрывающие теги абзацев и элементов списка, которые язык HTML разрешает опускать. (Мы должны признать, что с позиций XHTML такое вмешательство оправданно.)

К чести редактора Word, он работает довольно устойчиво, в отличие от других приложений, которые завершались аварийно, когда мы боролись с их интерпретацией разметки. Более того, Microsoft предлагает плагин для Word, убирающий дополнительную разметку и позволяющий восстановить первоначальный вид документа.[1]

17.2.2. Когда и зачем нужно редактировать работу редактора

Независимо от того, насколько хорош HTML-редактор, вам все равно придется редактировать текст, который он сгенерирует (как минимум, замусоренный). Это показывает и наш личный опыт, и опыт всех веб-разработчиков, с которыми мы общаемся на протяжении последних лет.

Не все HTML-редакторы предоставляют удобный способ добавления JavaScript-сценариев в документы, а многие даже не соответствуют последним стандартам HTML/XHTML и CSS2. Вспомним также, что популярные броузеры, и даже разные версии одного броузера, иногда расходятся в интерпретации многих тегов. Кроме того, даже лучшие HTML-редакторы не обязаны поддерживать расширения языка.

Так или иначе, вы будете вынуждены вникать в исходный текст либо для того, чтобы включить в него какую-то особенность HTML, еще не поддерживаемую редактором (например, новое свойство из стандарта CSS2), либо чтобы добавить ключевое слово или значение атрибута, либо чтобы откорректировать значения, указанные редактором.

---

[1] Этот плагин можно найти здесь: *http://office.microsoft.com/downloads/2000/Msohtmf2.aspx*

Совет: вначале составьте документ. Постарайтесь сделать его ясным и законченным. С самого начала сосредоточьтесь на содержимом, а спецэффектами займитесь на заключительной стадии. Пользуйтесь хорошим HTML-редактором или же готовьте документ за два этапа, с помощью других разных инструментов: хорошего текстового редактора и хорошего HTML-редактора, особенно когда собираетесь распространять документ еще и в формате, отличном от HTML.

17.2.3. Используйте лучшее

Если вы создаете веб-страницы, трудно представить, что вы обходитесь без хоть какого-то HTML-редактора. Уж очень велик соблазн. Однако выбирайте редактор внимательно, – некоторые просто ужасны, и у вас больше времени уйдет на исправление сгенерированного кода, чем на собственно разработку документа. Главный намек: вы получаете то, за что заплатили.

Неудивительно, что HTML-редакторы сильно различаются по своим функциональным возможностям. Многие редакторы позволяют вам переключаться с исходного текста на вид в окне броузера. В других вы всего лишь добавляете теги и модифицируете атрибуты с помощью раскрывающихся меню и горячих клавиш. Третьи представляют собой WYSIWYG-инструменты, облегчающие сопровождение страниц графикой и мультимедийным содержимым. Дополнительными функциональными возможностями HTML-редакторов являются внедрение и тестирование апплетов и сценариев.

Вообще говоря, HTML-редакторы можно разбить на две категории: либо они являются хорошими средствами макетирования, позволяющими подключать стили и динамическое содержимое, либо они прекрасно подходят для создания содержимого и управления им. Очевидно, что если вы создаете броские коммерческие веб-страницы, опирающиеся на современные технологии и включающие в себя массу различных стилей и динамическое содержимое, вам следует работать с инструментом, облегчающим макетирование. Если же результатом вашего труда являются документы с насыщенным содержимым, вы должны предпочесть инструмент, обеспечивающий качественное редактирование.

Независимо от того, какой тип HTML-редактора вам нужен, существуют общие соображения, из которых нужно исходить при выборе конкретного приложения:

Соответствует ли оно последним стандартам?

Ни один HTML-редактор не соответствует последним стандартам (особенно CSS2) на все сто процентов. Читайте технические характеристики и своевременно обновляйте редактор.

Входит ли в состав приложения редактор исходного кода?

Хотя ничто не мешает вам загрузить документ, сгенерированный HTML-редактором, в другой текстовый редактор и модифицировать

его там, гораздо удобнее, если HTML-редактор сам позволяет вам редактировать исходный текст. Убедитесь, что HTML-редактор не станет автоматически «исправлять» то, что вы написали вручную.

Допускает ли редактор настройку предпочтений?

В идеальном случае HTML-редактор позволяет вам настраивать его «под себя». Например, он, как минимум, должен разрешить вам выбирать шрифт, а также его цвет и цвет фона, если эти характеристики автоматически включаются в генерируемый текст.

Надежен ли редактор, и можете ли вы позволить себе его приобретение?

Мы не устанем повторять, что вы получаете то, за что заплатили. Если создание веб-страниц для вас нечто большее, чем временное увлечение, приобретите самый лучший редактор, какой сможете найти. Следует предпочесть тот, который лучше поддерживается производителем и получил наиболее благоприятные отзывы HTML-авторов. Поспрашивайте знакомых, поучаствуйте в группах новостей, поищите самую свежую информацию о продуктах.

17.3. Трюки с таблицами

По своей конструкции таблицы позволяют авторам создавать привлекательные и легко доступные таблицы информации. Но табличные теги можно также использовать для создания новаторского дизайна страниц, который иначе нельзя осуществить в стандартах HTML и XHTML.

17.3.1. Страницы с несколькими столбцами

Один очень распространенный и популярный элемент макетирования, отсутствующий в стандартах HTML 4 и XHTML, – это вывод текста в несколько колонок. Приведем совет, как достичь такого эффекта.[1]

17.3.1.1. Базовый макет вывода в несколько столбцов

Базовый макет в две колонки с применением тега `<table>` включает в себя одну строку с тремя ячейками данных – по одной на каждый столбец – и одной пустой ячейкой в промежутке, чтобы красивее отделить столбцы друг от друга. Мы вставили также атрибут `cellspacing` с большим значением, чтобы увеличить расстояние между столбцами.

Следующий HTML-пример таблицы дает превосходный шаблон для простого макета в две колонки.

```
<table border=0 cellspacing=7>
  <tr>
    <td>Copy for column 1...
```

[1] Известно, что ранние версии Netscape поддерживали расширение `<multicol>`. Но этому пришел конец. Мы приведем универсальное решение.

```
    <td><br>
    <td>Copy for column 2...
</table>
```

На рис. 17.1 можно видеть результат.

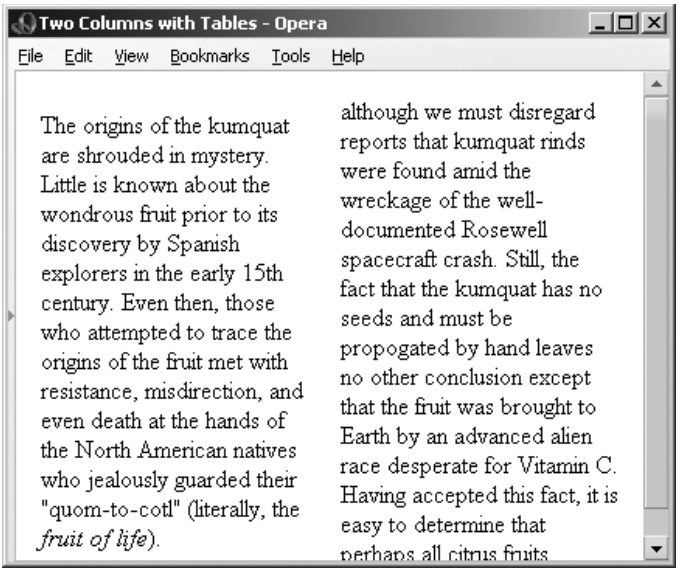

Рис. 17.1. *Простой макет в две колонки*

Единственная неприятность состоит в том, что броузер не будет автоматически балансировать текст в столбцах, добиваясь, чтобы соседние колонки имели приблизительно одинаковую длину. Вам придется поэкспериментировать с документом, вручную перемещая текст из столбца в столбец, пока вы не получите хорошо сбалансированную страницу. Имейте в виду, однако, что пользователи могут изменить размер окна броузера и содержимое столбцов соответственно сдвинется. Так что не тратьте слишком много времени, точно выравнивая последние предложения столбцов. Им все равно суждено разойтись при другой ширине окна броузера.

Разумеется, можно с легкостью преобразовать макет нашего примера в три и более колонок, распределив текст по большему числу ячеек таблицы. Имейте только в виду, что страницы с более чем тремя колонками могут оказаться трудными для чтения на маленьких дисплеях, на которых фактически ширина столбцов может быть очень мала.

17.3.1.2. Широкие заголовки

Базовое многоколоночное форматирование – это только начало. Добавляя ячейки, охватывающие столбцы, можно создавать заголовки. Подобным образом вы можете разместить рисунки, охватывающие не-

сколько столбцов, – просто вставьте в ячейку, содержащую заголовок или рисунок, атрибут `colspan`. Рисунок 17.2 демонстрирует привлекательный макет в три колонки с заголовком над всеми столбцами и рисунком, размещенным в двух столбцах. Он получается при исполнении следующего исходного HTML-текста, содержащего теги таблицы.

```
<table border=0 cellspacing=7>
  <tr>
    <th colspan=5><h2>The History of the Kumquat</h2>
  <tr valign=top>
    <td rowspan=2>Copy for column 1...
    <td rowspan=2 width=24><br>
    <td>Copy for column 2...
    <td width=24><br>
    <td>Copy for column 3...
  <tr>
    <td colspan=3 align=center><img src="pics/fruit.gif">
    <p>
    <i>The Noble Fruit</i>
</table>
```

Чтобы получить этот красивый макет, мы использовали атрибут `colspan` в ячейке первой строки, охватывающей все пять столбцов таблицы (три с текстом и два средника). Мы использовали атрибут `rowspan`

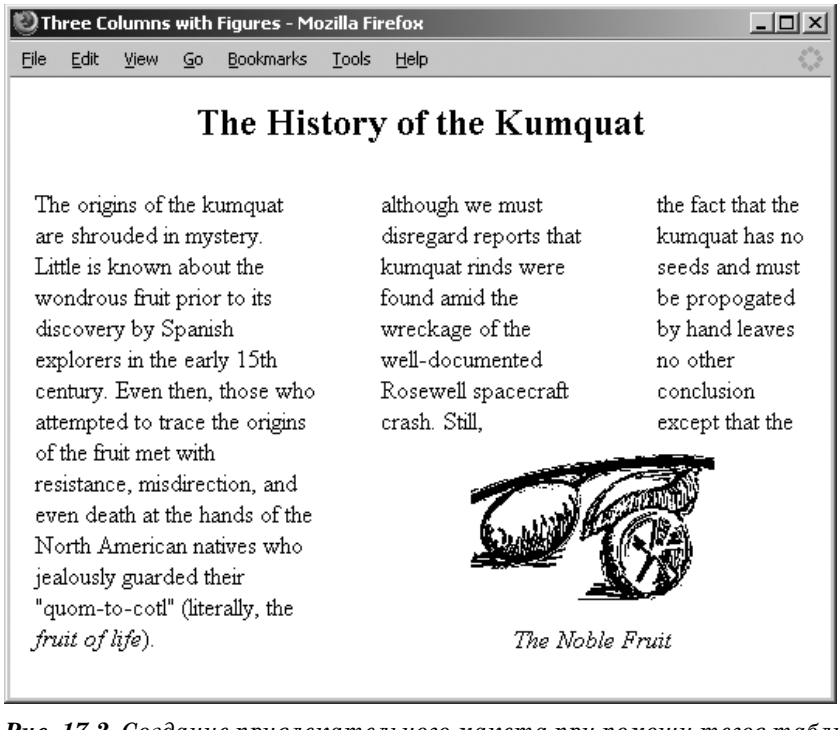

***Рис. 17.2.** Создание привлекательного макета при помощи тегов таблицы*

в первом столбце и в прилегающем к нему среднике, чтобы растянуть вниз столбец слева от рисунка. В ячейке с рисунком содержится атрибут `colspan`, поэтому ее содержимое охватывает другие два столбца и пространство между ними.

17.3.2. Боковые заголовки

Единственными предусмотренными в HTML и XHTML заголовками являются теги `<h1>`–`<h6>`. Эти теги вкладываются в поток текста, разделяя смежные абзацы текста. При помощи многостолбцовости можно добиться другого стиля, в котором заголовки помещаются сбоку, в отдельном столбце, проходящем в вертикальном направлении вдоль текста документа.

На рис. 17.3 можно увидеть довольно-таки симпатичную пару боковых заголовков, результат исполнения следующего кусочка XHTML-кода таблицы:

```
<table>
   <tr>
     <th width="20%" align="right">
       <h3>Section 1</h3></th>
     <td></td>
     <td>
       Copy for section 1 goes on and on a bit
       so that it will take up more than one line in the
       table cell window... </td>
   </tr>
   <tr>
     <th align="right">
       <h3>Section 2</h3></th>
     <td></td>
     <td>
       Copy for section 2 goes on and on a bit
       so that it will take up more than one line in the
       table cell window...</td>
   </tr>
 </table>
```

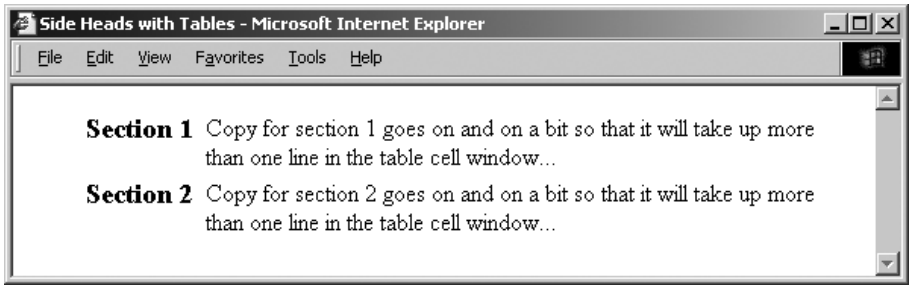

***Рис. 17.3.** Теги таблицы создали эти боковые заголовки*

Обратите внимание на то, как мы отодвинули эти относительно привлекательные боковые заголовки от левого края окна броузера, подобрав подходящую ширину первой заголовочной ячейки и выровняв ячейки правого столбца по правому краю.

Точно так же, как и в примере многоколоночного макета, в макете с боковыми заголовками используется пустой столбец для создания промежутка между узким левым столбцом с заголовками и более широким правым столбцом, содержащим ассоциированный с этими заголовками текст. Лучше всего указывать ширину этого столбца в процентах от ширины таблицы, а не в пикселах, гарантируя тем самым, что заголовочный столбец масштабируется в соответствии с широким или узким окном броузера.

17.3.3. Улучшенное макетирование форм

Из всех элементов HTML и XHTML формам более всего не достает средств управления их макетом. В отличие от других структурированных элементов, формы выглядят лучше, когда они жестко размещены на экране, с точно определенными полями и вертикальным выравниванием элементов. Однако за исключением тщательно спланированного форматирования элементов формы при помощи тега `<pre>`, обычный язык не дает нам никаких специальных средств для управления размещением формы на экране. Вы можете многого достичь с помощью таблиц стилей, но они быстро усложняют проект. Зато обыкновенные таблицы позволят вам хорошо скомпоновать формы.

17.3.3.1. Базовый макет формы

Формы почти всегда будут выглядеть лучше и в них будет легче разбираться читателям, если применить для структурирования и выравнивания элементов формы таблицу. В формах, например, можно использовать вертикальное выравнивание, размещая слева надписи, а справа – выровненные по смежному вертикальному полю элементы формы. Не пытайтесь добиться этого только при помощи стандартов HTML и XHTML. Подготовьте лучше форму, содержащую таблицу с двумя столбцами. Следующий исходный HTML-текст делает то, что можно увидеть на рис. 17.4:

```
<form method=post action="http:/cgi-bin/process">
  <table>
    <tr>
      <th align=right>Name:
      <td><input type=text size=32>
    <tr>
      <th align=right>Address:
      <td><input type=text size=32>
    <tr>
      <th align=right>Phone:
      <td><input type=text size=12>
```

```
    <tr>
      <td colspan=2 align=center>
      <input type=submit value="Register">
  </table>
</form>
```

Конечно, при помощи таблиц можно сделать и более сложный макет. Мы рекомендуем вам выполнить сначала набросок формы на бумаге и спланировать то, как различные комбинации элементов таблицы, включая слияние ячеек по строкам и по столбцам, могут быть использованы для реализации этого макета.

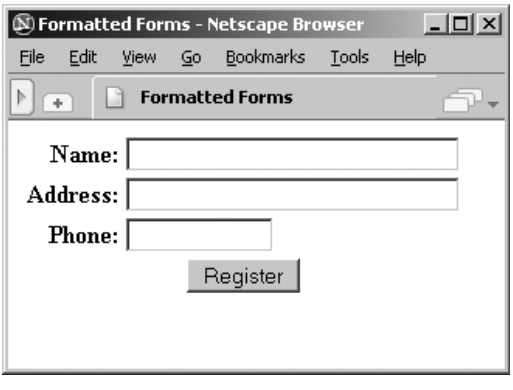

Рис. 17.4. *Аккуратно выровняйте вашу форму при помощи таблицы*

17.3.3.2. Построение форм с применением вложенных таблиц

Как мы отмечали выше, таблицу можно поместить в ячейке другой таблицы, что позволяет создать более сложную структуру таблицы. В применении к формам это может быть полезным при организации подмножества элементов формы внутри таблицы большего размера, содержащей форму целиком. Наилучшим применением вложенных таблиц в формах служит планировка выключателей (флажков) и переключателей.

Вставьте, например, следующую строку, содержащую таблицу, в таблицу формы из предыдущего примера. Она создает выключатели с четырьмя вариантами выбора:

```
<tr>
  <th align=right valign=top>Preferences:
  <td>
  <table>
    <tr>
    <td><input type=checkbox name=pref>Lemons
    <td><input type=checkbox name=pref>Limes
    <tr>
```

```
    <td><input type=checkbox name=pref>Oranges
    <td><input type=checkbox name=pref>Kumquats
  </table>
```

На рис. 17.5 показано, как привлекательно размещаются выключатели при помощи вложенной таблицы, без которой броузер просто вывел бы их в одну строку без надлежащего выравнивания.

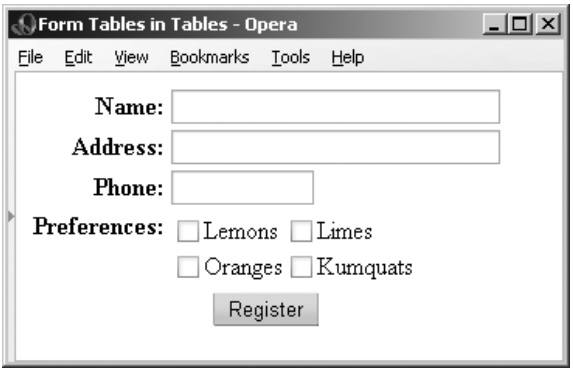

***Рис. 17.5.** Вкладывание таблиц более низкого уровня для форматирования формы*

17.4. Фокусы с окнами и фреймами

В подавляющем большинстве случаев вы захотите, чтобы загружаемый после щелчка по ссылке документ отображался в том же самом окне, замещая предыдущий. Это имеет смысл, поскольку пользователи обычно перебирают документы коллекции последовательно.

Но иногда имеет смысл открыть документ в новом окне так, чтобы и новый документ и старый были непосредственно доступны, находясь на экране пользователя. Например, имеет смысл держать перед глазами оба документа, если новый документ ссылается на предыдущий. В других случаях, возможно, потребуется открыть несколько документов в различных окнах набора фреймов. Однако чаще всего новый документ ставит пользователя как бы перед новой сетью документов, а вы, возможно, хотите, чтобы пользователи помнили и видели, откуда они пришли.

Независимо от причины можно с легкостью открыть новое окно броузера из вашего документа. Все, что для этого необходимо, – это вставить в соответствующий тег `<a>` атрибут `target`.

17.4.1. Направление документа в окно

Обычно мы используем атрибут `target` для загрузки документа в определенный фрейм, имеющий в наборе фреймов собственное имя. Он

также может служить для создания нового окна с использованием одного из двух методов:

Ссылка на новое имя

Если вы используете не определенное заранее имя в качестве значения атрибута `target` в гиперссылке, популярные броузеры автоматически создадут новое окно с этим именем и загрузят в это окно указанный документ. Такой способ создания нового окна наиболее предпочтителен, поскольку впоследствии можно использовать данное новое имя и загружать в это окно новые документы. Используя эту технику, можно управлять тем, куда и какой документ загружается.

Создание безымянного окна

Все популярные броузеры поддерживают специальное значение атрибута `target`, называющееся `_blank` и позволяющее создавать новое окно. Возможности использования таким образом созданного окна, однако, ограничены, поскольку оно остается безымянным, – в это окно не удастся направить никакой другой документ. (Новые документы, загружаемые в результате исполнения гиперссылок в этом окне, разумеется, будут загружаться в него.)

17.4.2. Преодоление чужих нацеливаний

Случалось ли вам посещать сайты с домашней страничкой, представляющей собой фреймовый документ, который никогда не сдается? Ну, вы знаете, это те, что оставляют свой большой-пребольшой логотип вверху окна, а сбоку пускают оглавление сайта, от которого не удается избавиться, даже уйдя с сайта по гиперссылке. Что, если набор фреймов вашего сайта окажется захвачен одним из фреймов их окна? Что делать? (Их вебмастеры, очевидно, ничего не слышали о значении `_blank` атрибута `target`.)

Быстрое решение состоит в использовании JavaScript, чтобы принудительно открыть новое окно для ваших документов, но это может запутать пользователей, у которых, возможно, уже открыто окно, готовое принять ваш документ. Поэтому пусть сначала JavaScript определит, предназначен ли ваш документ для целого окна или для фрейма в углу.

Ниже следует образец сценария, который загружает веб-страницу под названием *index2.html* в целиком принадлежащее ей окно. Заметьте, что поддерживающие JavaScript броузеры не позволят вам очистить прежде загруженный документ, если только окно не принадлежит вашему документу. Поэтому в том случае, если целью является не все окно («self» это не «window.top»), сценарий из нашего примера откроет новое окно, которое станет целью для ваших страниц. Пользователь может выбирать, закрыть ли ваше окно и вернуться к другому или поступить наоборот.

```
<html>
<head>
```

```
<title>мне необходимо собственное окно</title>
<script language="JavaScript">
<!--
  if (self != window.top)
      window.open("http://www.kumquats.com/index2.html");
  else
    self.location.href = "http://www.kumquats.com/index2.html";
//-->
</script>
</head>
<body>
ваш броузер не поддерживает JavaScript. Пожалуйста, выберите ссылку
<a href="http://www.kumquats.com/index2.html">чтобы перейти
                                              на наш сайт вручную.</a>
</body>
</html>
```

17.4.3. Несколько фреймов в одной гиперссылке

Загрузка нового документа по гиперссылке – это просто щелчок мыши, даже если новый документ помещен во фрейм или окно, не совпадающие с тем, в котором содержится гиперссылка. Однако может так случиться, что вы захотите загрузить документы в два фрейма, когда пользователь щелкает только на одной гиперссылке. Проявляя некоторую изобретательность, можно загрузить два или большее число фреймов одновременно, если только они определенным образом расположены в окне броузера.

Рассмотрим следующий фреймовый макет:

```
<frameset rows=2>
  <frameset cols=2>
    <frame name=A>
    <frame name=B>
  </frameset>
  <frameset>
    <frame name=C>
    <frame name=D>
  </frameset>
</frameset>
```

Если кто-то щелкает на ссылке во фрейме A, то все, что можно сделать, это обновить один из четырех фреймов. Предположим, что нужно обновить одновременно фреймы B и D. Фокус состоит в том, чтобы заменить фреймы B и D одним фреймом, так, например:

```
<frameset cols=2>
  <frameset rows=2>
    <frame name=A>
    <frame name=C>
  </frameset>
```

```
  <frame name=BD>
</frameset>
```

Ага! Теперь у нас один фрейм, в который следует загрузить документ, – это фрейм BD. Документ, который загружается в этот фрейм, должен, в свою очередь, содержать два фрейма – B и D – в одном столбце, например так:

```
<frameset cols=2>
  <frame name=B>
  <frame name=D>
</frameset>
```

Два фрейма заполняют фрейм BD. Когда обновится фрейм BD, оба фрейма будут заменены, это создаст впечатление, что два фрейма обновлены одновременно.

Единственный недостаток этого способа состоит в том, что фреймы должны быть смежными и группироваться в один документ. Для большинства страниц, однако, такое решение работает вполне удовлетворительно.

Мы только скользнули здесь по поверхности фокусов и трюков с HTML и XHTML. Наш последний совет: продолжайте выдумывать!

A

Грамматика HTML

По большей части, броузеры не настаивают жестко на точном соблюдении синтаксических правил HTML или даже XHTML. Это дает авторам большую свободу при создании документов и ведет к росту числа документов, которые работают на большинстве броузеров, но на самом деле не согласуются со стандартами HTML и XHTML. Придерживайтесь стандартов, если только ваши документы не мотыльки-однодневки.

Стандарты явным образом определяют порядок следования и допустимую вложенность тегов и элементов документов. Этот синтаксис содержится в соответствующем определении типа документа (DTD, Document Type Definition), и в нем не просто разобраться тем, кто не владеет SGML (для HTML 4.01 см. приложение D) или XML (для XHTML 1.0 см. приложение E). Поэтому мы предоставляем определение допустимого HTML- и XHTML-синтаксиса в другой форме, используя весьма обычное орудие, именуемое грамматикой.

Грамматику, определяет ли она английские предложения или HTML-документы, можно разбить на два множества: *терминальных символов*, или *терминалов* (слов языка), и *нетерминальных символов*, или, соответственно, *нетерминалов* (все прочие грамматические правила). В HTML и XHTML словам соответствуют теги разметки и текст документа.

Чтобы применить грамматику для создания действительного документа, следуйте последовательности правил, чтобы понять, где при создании действительного документа могут размещаться теги и текст.

Принятые обозначения

Мы применяем условные обозначения и специальную пунктуацию, чтобы в наших правилах было легче разобраться.

Условные обозначения и соглашение об именах

В нашей грамматике мы обозначаем терминалы моноширинным шрифтом. Нетерминальные символы обозначаются курсивом.

Мы также следуем особому соглашению об именах, касающемуся большинства наших нетерминальных символов, – если правило определяет синтаксис определенного тега, имя этого правила будет начинаться с имени этого тега, за которым следует суффикс *_tag*. Если нетерминал определяет допустимое содержимое определенного тега, его имя будет именем тега, за которым будет следовать *_content*.

Если, например, вы хотите знать, какие в точности элементы могут содержаться в теге `<a>`, вы можете заглянуть в правило *a_content* нашей грамматики. Подобным образом, чтобы узнать правильный синтаксис списка определений с тегом `<dl>`, см. правило *dl_tag*.

Пунктуация

Каждое правило в грамматике начинается с имени правила, за которым следует символ замены (`::=`) и значение правила. Стараясь сделать грамматику простой, мы все же используем три пунктуационных элемента для обозначения альтернативы, повторения и необязательных элементов.

Альтернатива

Альтернатива означает, что правило может иметь несколько различных значений и что вы должны выбрать ровно одно из них. Альтернативные варианты разделяются в правилах вертикальной чертой (`|`).

Правило *heading*, например, эквивалентно одному из шести тегов заголовков HTML, и, таким образом, выглядит в таблице как:

```
heading       ::=       h1_tag
                |       h2_tag
                |       h3_tag
                |       h4_tag
                |       h5_tag
                |       h6_tag
```

Правило *heading* говорит нам, что где бы в правиле ни появился нетерминальный символ *heading*, вы можете заменить его ровно одним из перечисленных тегов заголовка.

Повторение

Повторение означает, что элемент в правиле может быть повторен несколько раз. Повторяющиеся элементы заключаются в фигурные скобки. При закрывающей скобке внизу приписывается число, отличное от единицы, обозначающее минимальное количество повторений этого элемента.

К примеру, тег <ul> может содержать только теги <li>, но может быть и пустым. Правило, следовательно, будет таким:

```
ul_tag          ::=  <ul>
                       {li_tag }0
                     </ul>
```

Это правило говорит, что синтаксис тега <ul> требует открывающего тега <ul>, нулевого или большего числа тегов <li>, за которыми следует закрывающий тег . Мы записали это правило в нескольких строках и применили отступы только для того, чтобы его было легче читать – это не означает, что документы должны на самом деле так форматироваться.

Необязательные элементы

Некоторые элементы могут появляться в документе, но не являются обязательными. Необязательные элементы заключаются в квадратные скобки ([и]).

Тег <table>, например, имеет необязательный заголовок (<caption>)

```
table_tag       ::=  <table>
                     [ caption_tag ]
                       {tr_tag }0
                     </table>
```

В совокупности правило гласит, что таблица начинается с тега <table>, за которым следует необязательный заголовок, некоторое, возможно, нулевое число тегов строк, и что завершается она тегом </table>.

Подробности

Наша грамматика останавливается на уровне тегов, она не проникает глубже, не показывает синтаксиса каждого тега и его атрибутов. Эти подробности можно найти в карточке краткого справочника, включенного в эту книгу.

Первичные нетерминальные символы

Стандарты HTML и XHTML определяют несколько особых видов содержимого, соответствующих различным типам текста. Это:

literal_text (литеральный текст)

Текст здесь интерпретируется точно так, как написан, – ни замены символов, ни теги стилей не распознаются.

plain_text (гладкий текст)

Обычные символы в кодировке документа, а также замены символов их кодами, обозначенные символом амперсанда.

style_text (отформатированный текст)

Как *plain_text*, но допускаются также теги физической и логической разметки.

Грамматика

Грамматика составлена из тегов стандартов HTML 4.01 и XHTML 1.0 и специальных расширений языка, поддерживаемых последними версиями Netscape Navigator и Microsoft Internet Explorer.

Правила перечисляются в алфавитном порядке. Начальное правило для целого документа называется *html_document*.

a_content[a]	::=	*heading*
	|	*text*
a_tag	::=	<a>
		{*a_content*}0
		
abbr_tag	::=	<abbr> *text* </abbr>
acronym_tag	::=	<acronym> *text* </acronym>
address_content	::=	*p_tag*
	|	*text*
address_tag	::=	<address>
		{*address_content*}0
		</address>
applet_content	::=	{<param>}0
		body_content
applet_tag	::=	<applet> *applet_content* </applet>
b_tag	::=	*text*
basefont_tag	::=	<basefont> *body_content* </basefont>
bdo_tag	::=	<bdo> *text* </bdo>
big_tag	::=	<big> *text* </big>
blink_tag	::=	<blink> *text* </blink>
block	::=	{*block_content*}0
block_content	::=	<isindex>
	|	*basefont_tag*
	|	*blockquote_tag*
	|	*center_tag*
	|	*dir_tag*
	|	*div_tag*
	|	*dl_tag*

[a] *a_content* не может содержать *a_tag*; нельзя вкладывать теги <a> в другие теги <a>.

	|	*form_tag*
	|	*listing_tag*
	|	*menu_tag*
	|	*multicol_tag*
	|	*nobr_tag*
	|	*ol_tag*
	|	*p_tag*
	|	*pre_tag*
	|	*table_tag*
	|	*ul_tag*
	|	*xmp_tag*
blockquote_tag	::=	<blockquote> *body_content* </blockquote>
body_content	::=	<bgsound>
	|	<hr>
	|	*address_tag*
	|	*block*
	|	*del_tag*
	|	*heading*
	|	*ins_tag*
	|	*layer_tag*
	|	*map_tag*
	|	*marquee_tag*
	|	*text*
body_tag	::=	<body>
		{*body_content*}**0**
		</body>
caption_tag	::=	<caption> *body_content* </caption>
center_tag	::=	*body_content*
cite_tag	::=	<cite> *text* </cite>
code_tag	::=	<code> *text* </code>
colgroup_content	::=	{<col>}**0**
colgroup_tag	::=	<colgroup>
		colgroup_content
content_style	::=	*abbr_tag*
	|	*acronym_tag*

	|	*cite_tag*
	|	*code_tag*
	|	*dfn_tag*
	|	*em_tag*
	|	*kbd_tag*
	|	*q_tag*
	|	*strong_tag*
	|	*var_tag*
dd_tag	::=	<dd> *flow* </dd>
del_tag	::=	<del> *flow* </del>
dfn_tag	::=	<dfn> *text* </dfn>
dir_tag[a]	::=	<dir>
		{*li_tag*}
		</dir>
div_tag	::=	*body_content*
dl_content	::=	*dt_tag dd_tag*
dl_tag	::=	<dl>
		{*dl_content*}
		</dl>
dt_tag	::=	<dt>
		text
		</dt>
em_tag	::=	text
fieldset_tag	::=	<fieldset>
		[*legend_tag*]
		{*form_content*}0
		</fieldset>
flow	::=	{*flow_content*}0
flow_content	::=	*block*
	|	*text*
font_tag	::=	 *style_text*
form_content[b]	::=	<input>

[a] *li_tag* в *dir_tag* не может содержать блочных элементов.

[b] *form_content* не может содержать *form_tag*, так как нельзя вкладывать форму в другую форму.

	|	<keygen>
	|	*body_content*
	|	*fieldset_tag*
	|	*label_tag*
	|	*select_tag*
	|	*textarea_tag*
form_tag	::=	<form>
		{*form_content*}0
		</form>
frameset_content	::=	<frame>
	|	*noframes_tag*
frameset_tag	::=	<frameset>
		{*frameset_content*}0
		</frameset>
h1_tag	::=	*text*
h2_tag	::=	*text*
h3_tag	::=	*text*
h4_tag	::=	*text*
h5_tag	::=	*text*
h6_tag	::=	*text*
head_content	::=	<base>
	|	<isindex>
	|	<link>
	|	<meta>
	|	<nextid>
	|	*style_tag*
	|	*title_tag*
head_tag	::=	<head>
		{*head_content*}0
		</head>
heading	::=	*h1_tag*
	|	*h2_tag*
	|	*h3_tag*
	|	*h4_tag*
	|	*h5_tag*

	|	*h6_tag*
html_content	::=	*head_tag body_tag*
	|	*head_tag frameset_tag*
html_document	::=	*html_tag*
html_tag	::=	<html> ***html_content*** </html>
i_tag	::=	***text***
ilayer_tag	::=	<ilayer> ***body_content*** </ilayer>
ins_tag	::=	<ins> ***flow*** </ins>
kbd_tag	::=	<kbd> ***text*** </kbd>
label_content[a]	::=	<input>
	|	*body_content*
	|	*select_tag*
	|	*textarea_tag*
label_tag	::=	<label>
		{*label_content*}0
		</label>
layer_tag	::=	<layer> ***body_content*** </layer>
legend_tag	::=	<legend> ***text*** </legend>
li_tag	::=	***flow***
listing_tag	::=	<listing> ***literal_text*** </listing>
map_content	::=	{<area>}0
map_tag	::=	<map> ***map_content*** </map>
marquee_tag	::=	<marquee> ***style_text*** </marquee>
menu_tag[b]	::=	<menu>
		{*li_tag*}0
		</menu>
multicol_tag	::=	<multicol> ***body_content*** </multicol>
nobr_tag	::=	<nobr> ***text*** </nobr>
noembed_tag	::=	<noembed> ***text*** </noembed>
noframes_tag	::=	<noframes>
		{*body_content*}0
		</noframes>
noscript_tag	::=	<noscript> ***text*** </noscript>

[a] В тег <label> не могут вкладываться теги <label> и <form>.

[b] *li_tag* в *menu_tag* не может содержать блочные элементы.

object_content	::=	{<param>}0
		body_content
object_tag	::=	<object> *object_content* </object>
ol_tag	::=	
		{*li_tag*}
optgroup_tag	::=	<optgroup>
		{*option_tag*}0
		</optgroup>
option_tag	::=	<option> *plain_text* </option>
p_tag	::=	*text*
physical_style	::=	*b_tag*
	|	*bdo_tag*
	|	*big_tag*
	|	*blink_tag*
	|	*font_tag*
	|	*i_tag*
	|	*s_tag*
	|	*small_tag*
	|	*span_tag*
	|	*strike_tag*
	|	*sub_tag*
	|	*sup_tag*
	|	*tt_tag*
	|	*u_tag*
pre_content	::=	
	|	<hr>
	|	*a_tag*
	|	*style_text*
pre_tag	::=	<pre>
		{*pre_content*}0
		</pre>
q_tag	::=	<q> *text* </q>
s_tag	::=	<s> *text* </s>
samp_tag	::=	<samp> *text* </samp>

script_tag[a]	::=	<script> ***plain_text*** </script>
select_content	::=	*optgroup_tag*
	|	*option_tag*
select_tag	::=	<select>
		{*select_content*}0
		</select>
server_tag[b]	::=	<server> ***plain_text*** </server>
small_tag	::=	***text***
span_tag	::=	***text***
strike_tag	::=	<strike> ***text*** </strike>
strong_tag	::=	***text***
style_tag	::=	<style> ***plain_text*** </style>
sub_tag	::=	_{***text***}
sup_tag	::=	^{***text***}
table_cell	::=	*td_tag*
	|	*th_tag*
table_content	::=	<tbody>
	|	<tfoot>
	|	<thead>
	|	*tr_tag*
table_tag	::=	<table>
		[*caption_tag*]
		{*colgroup_tag*}0
		{*table_content*}0
		</table>
td_tag	::=	<td> *body_content* </td>
text	::=	{*text_content*}0
text_content	::=	
	|	<embed>
	|	<iframe>
	|	<img>

[a] *script_tag* может помещаться где угодно в HTML-документе, невзирая на синтаксические правила.

[b] *server_tag* может помещаться где угодно в HTML-документе, невзирая на синтаксические правила.

	|	<spacer>
	|	<wbr>
	|	*a_tag*
	|	*applet_tag*
	|	*content_style*
	|	*ilayer_tag*
	|	*noembed_tag*
	|	*noscript_tag*
	|	*object_tag*
	|	*physical_style*
	|	*plain_text*
textarea_tag	::=	<textarea> *plain_text* </textarea>
th_tag	::=	<th> *body_content* </th>
title_tag	::=	<title> *plain_text* </title>
tr_tag	::=	<tr>
		{*table_cell*}0
		</tr>
tt_tag	::=	<tt> *text* </tt>
u_tag	::=	<u> *text* </u>
ul_tag	::=	
		{*li_tag*}
var_tag	::=	<var> *text* </var>
xmp_tag	::=	<xmp> *literal_text* </xmp>

B

Краткий справочник по тегам HTML/XHTML

В этом приложении в алфавитном порядке перечислены все известные (и некоторые недокументированные) теги HTML и XHTML и их атрибуты, поддерживаемые хотя бы одним из современных популярных броузеров.

Основные атрибуты

До появления HTML 4.0 имелось совсем немного атрибутов, которые можно было бы систематически использовать со всеми HTML-тегами. HTML 4.0 изменил это положение вещей, определив набор из шестнадцати основных атрибутов, которые могут применяться почти ко всем элементам HTML и XHTML. Чтобы не повторять эти основные атрибуты в главной таблице, мы перечислим их здесь:

`class=`*name*	Определяет класс стилей, управляющий внешним видом содержимого тега
`dir=`*dir*	Определяет направление вывода текста либо слева направо (`ltr`), либо справа налево (`rtl`)
`id=`*name*	Определяет уникальное в документе имя тега для последующих ссылок
`lang=`*language*	Определяет язык, использованный для содержимого этого тега, с применением двухсимвольного наименования языка по стандарту ISO639 и, возможно, кода диалекта
`onclick=`*applet*	Определяет программный код, который должен быть исполнен, когда пользователь щелкает кнопкой мыши на области отображения тега

ondblclick=*applet*	Определяет программный код, который должен быть исполнен, когда пользователь дважды щелкает кнопкой мыши на области отображения тега
onkeydown=*applet*	Определяет программный код, который должен быть исполнен, когда пользователь нажимает клавишу в случае, когда содержимое тега активно
onkeypress=*applet*	Определяет программный код, который должен быть исполнен, когда пользователь нажимает и отпускает клавишу в случае, когда содержимое тега активно
onkeyup=*applet*	Определяет программный код, который должен быть исполнен, когда пользователь отпускает клавишу в случае, когда содержимое тега активно
onmousedown=*applet*	Определяет программный код, который должен быть исполнен, когда пользователь нажимает на кнопку мыши в случае, когда указатель мыши находится в области отображения содержимого тега
onmousemove=*applet*	Определяет программный код, который должен быть исполнен, когда пользователь ведет указатель мыши по области отображения содержимого тега
onmouseout=*applet*	Определяет программный код, который должен быть исполнен, когда пользователь выводит указатель мыши из области отображения содержимого тега
onmouseover=*applet*	Определяет программный код, который должен быть исполнен, когда пользователь вводит указатель мыши в область отображения содержимого тега
onmouseup=*applet*	Определяет программный код, который должен быть исполнен, когда пользователь отпускает кнопку мыши в случае, когда указатель мыши находится в области отображения содержимого тега
style=*style*	Определяет для тега встроенный стиль
title=*string*	Определяет для тега название

Лишь небольшая часть тегов не принимает ни одного из этих атрибутов или принимает только некоторые из них. Это:

<applet>	<base>	<basefont>
<bdo>	 	<comment>
<embed>	<font>	<frame>
<frameset>	<head>	<hr>
<html>	<iframe>	<isindex>
<keygen>	<marquee>	<meta>
<nextid>	<nobr>	<noembed>

<param>	<script>	<server>
<spacer>	<style>	<title>
<wbr>		

Для удобства мы пометили в главной таблице эти теги звездочкой (*) и перечислили все атрибуты, поддерживаемые этими нетипичными тегами, включая и основные атрибуты. Для всех других тегов (не помеченных звездочкой) предполагается, что они поддерживают все вышеперечисленные атрибуты. Заметьте, однако, что популярные броузеры поддерживают *не* все атрибуты стандарта HTML 4.0, независимо от того, основные эти атрибуты или нет. Обратитесь за подробностями к основному тексту книги.

Краткий справочник по HTML

Мы используем иконку в крайнем правом столбце таблицы для обозначения тегов и атрибутов, которые являются расширениями стандартов HTML 4.01 и XHTML 1.0.

Мы используем пиктограмму броузера Internet Explorer для обозначения тегов и атрибутов расширений, присущих исключительно этому броузеру и плохо поддерживаемых другими. Хотя мы и включили в основной текст устаревшие и признанные нежелательными элементы и атрибуты, которые отсутствуют в стандартах и больше не поддерживаются ни одним броузером, мы не указываем их здесь.

Мы записываем все возможные атрибуты тега (некоторые из них обязательные) с отступом после соответствующих тегов. В описании мы по возможности указываем допустимые значения атрибута в виде области целых чисел или в виде списка вариантов.

<a> ... 	Создает якорь гиперссылки (атрибут href) или идентификатор фрагмента (атрибут id)	
accesskey=*char*	Определяет горячую клавишу для гиперссылки	
charset=*encoding*	Определяет набор символов, используемый в документе-цели гиперссылки	
coords=*list*	Определяет список координат, связанных с фигурами	
href=*url*	Определяет URL цели гиперссылки	
hreflang=*language*	Определяет язык документа-цели	
name=*name*	Определяет имя идентификатора фрагмента	
rel=*relationship*	Обозначает *отношение* текущего документа к документу-цели	

rev=*relationship*	Обозначает *отношение* документа-цели к текущему документу	
shape=*shape*	Определяет форму области как `circ`, `circle`, `poly`, `polygon`, `rect` или `rectangle`	
tabindex=*value*	Определяет положение якоря в табуляционной последовательности документа	
target=*name*	Определяет имя фрейма или окна, в котором следует разместить документ, на который указывает гиперссылка	
type=*type*	Определяет MIME-тип документа-цели	
`<abbr>` ... `</abbr>`	Заключенный в нем текст является сокращением	
`<acronym>` ... `</acronym>`	Заключенный в нем текст является акронимом	
`<address>` ... `</address>`	Заключенный в нем текст является адресом	
`<applet>` ... `</applet>`	Определяет в потоке текста исполняемый апплет	*
align=*position*	Выравнивает область апплета как `top`, `middle`, `bottom` (по умолчанию), `left`, `right`, `absmiddle`, `baseline` или `absbottom` по отношению к строке текста	
alt=*string*	Определяет альтернативный текст для помещения в область апплета для броузеров, которые поддерживают тег `<applet>`, но не могут исполнить приложение	
archive=*url*	Определяет архив классов, который должен скачать броузер, чтобы найти в нем нужный класс	
class=*name*	Определяет класс стилей, управляющий внешним видом этого тега	
code=*class*	Определяет имя класса кода, который должен исполняться (обязательный)	
codebase=*url*	Определяет URL, с которого должен быть получен код	
height=*n*	Определяет высоту области апплета в пикселах	
hspace=*n*	Определяет добавочное пространство справа и слева от области апплета в пикселах	
id=*name*	Определяет имя апплета, уникальное в документе	
mayscript	Разрешает апплету доступ к JavaScript на странице	

`name=`***name***	Определяет имя этого конкретного вхождения тега `<applet>`	
`object=`***data***	Определяет представление состояния исполнения объекта	
`style=`***style***	Определяет встроенный стиль для этого тега	
`title=`***string***	Предоставляет апплету название	
`vspace=`***n***	Определяет добавочное пространство сверху и снизу от области апплета в пикселах	
`width=`***n***	Определяет ширину области апплета в пикселах	
`<area>`	Определяет чувствительную область на карте, обрабатываемой со стороны клиента	
`accesskey=`***char***	Определяет горячую клавишу для этой области	
`alt=`***string***	Предоставляет альтернативный текст, который должен отображаться неграфическими броузерами	
`coords=`***list***	Определяет разделенный запятыми *список* координат, характерных для геометрических фигур, которые определяют границы этой области	
`href=`***url***	Определяет URL цели гиперссылки, ассоциированной с этой областью	
`nohref`	Обозначает, что с этой областью не ассоциирован никакой документ; щелчок на этой области будет безрезультатным	
`notab`	Исключает эту область из табуляционной последовательности	
`onblur=`***applet***	Определяет программный код, который следует исполнить, когда мышь покидает область	
`onfocus=`***applet***	Определяет программный код, который следует исполнить, когда мышь попадает в эту область	
`shape=`***shape***	Определяет, что область имеет форму `circ`, `circle`, `poly`, `polygon`, `rect` или `rectangle`	
`tabindex=`***value***	Определяет положение этой области в табуляционной последовательности документа	
`taborder=`***n***	Определяет положение этой области в табуляционной последовательности документа	

target=*name*	Определяет фрейм или окно, в которые должен быть помещен документ, ассоциированный с этой областью	
...	Форматирует заключенный в нем текст, используя полужирный шрифт	
<base>	Определяет базовый URL для всех относительных URL в этом документе	*
href=*url*	Определяет базовый URL (обязательный)	
target=*name*	Определяет значение атрибута target по умолчанию для всех гиперссылок <a> в текущем документе	
<basefont>	Определяет размер шрифта для последующего текста (нежелателен)	 *
color=*color*	Определяет цвет основного шрифта	
face=*name*	Определяет гарнитуру шрифта, которая должна использоваться для текста, расположенного в области действия этого тега	
id=*name*	Определяет имя для этого тега, уникальное в этом документе	
name=*name*	Определяет гарнитуру шрифта, которая должна использоваться для текста, расположенного в области действия этого тега	
size=*value*	Устанавливает размер шрифта от 1 до 7 (обязательный; по умолчанию устанавливается 3)	
<bdo> ... </bdo>	Изменяет направление вывода заключенного в нем текста	*
class=*name*	Определяет класс стилей, управляющий внешним видом этого тега	
dir=*dir*	Определяет направление вывода текста либо слева направо (ltr), либо справа налево (rtl)	
id=*name*	Определяет имя для этого тега, уникальное в этом документе	
lang=*language*	Определяет язык, использованный для содержимого этого тега, с применением двухсимвольного названия языка по стандарту ISO	
style=*style*	Определяет встроенный стиль для этого тега	
title=*string*	Определяет название для этого тега	
<bgsound>	Определяет фоновый аудиофрагмент для документа	 *

`loop=`***value***	Устанавливает, сколько раз должен быть проигран аудиофрагмент; *value* может быть целым числом или ключе вым словом `infinite`	
`src=`***url***	Предоставляет URL аудиофайла	
`<big>` **...** `</big>`	Форматирует заключенный в нем текст, используя больший размер шрифта	
`<blink>` **...** `</blink>`	Вызывает мерцание своего содержимого	
`<blockquote>` **...** `</blockquote>`	Заключенный в нем текст является блокомцитатой	
`cite=`***url***	Определяет URL источника цитаты	
`<body>` **...** `</body>`	Ограничивает начало и конец тела документа	
`alink=`***color***	Устанавливает цвет активных ссылок в документе	
`background=`***url***	Определяет URL изображения, использованного в качестве фона документа	
`bgcolor=`***color***	Устанавливает цвет фона документа	
`bgproperties=`***value***	Если *value* это `fixed`, фоновое изображение не будет прокручиваться вместе с содержимым документа	
`leftmargin=`***value***	Устанавливает (в пикселах) размер левого поля в документе	
`link=`***color***	Устанавливает цвет непосещенных гиперссылок в документе	
`onblur=`***applet***	Определяет программный код, который следует исполнить, когда мышь покидает окно документа	
`onfocus=`***applet***	Определяет программный код, который следует исполнить, когда мышь попадает в окно документа	
`onload=`***applet***	Определяет программный код, который следует исполнить, когда документ загружен	
`onunload=`***applet***	Определяет программный код, который следует исполнить, когда документ выгружен	
`text=`***color***	Устанавливает цвет обычного текста в документе	
`topmargin=`***value***	Устанавливает размер (в пикселах) верхнего поля в документе	

vlink=*color*	Устанавливает цвет посещенных гиперссылок в документе	
 	Прерывает поток текста и возобновляет его со следующей строки	*
class=*name*	Определяет класс стилей, управляющий внешним видом этого тега	
clear=*margin*	Прерывает поток текста и спускается вниз до тех пор, пока указанные *поля* (left, right, none или all) не окажутся чистыми	
id=*name*	Определяет имя для этого тега, уникальное в этом документе	
style=*style*	Определяет встроенный стиль для этого тега	
title=*string*	Определяет название для этого тега	
<button>	Создает кнопку в форме	
accesskey=*char*	Определяет горячую клавишу для этой кнопки	
disabled	Выключает кнопку, не позволяя пользователю нажимать на нее	
name=*name*	Определяет имя параметра, который должен передаваться обрабатывающему форму приложению, если эта кнопка выбрана (обязательный)	
onblur=*applet*	Определяет программный код, который следует исполнить, когда мышь уходит с кнопки	
onfocus=*applet*	Определяет программный код, который следует исполнить, когда мышь попадает на кнопку	
tabindex=*n*	Определяет положение этого элемента в табуляционной последовательности	
type=*type*	Определяет тип кнопки: button, submit или reset	
value=*string*	Определяет значение параметра, передаваемого приложению, обрабатывающему форму, если эта кнопка выбрана (обязательный)	
<caption> ... </caption>	Определяет заголовок таблицы	
align=*position*	Устанавливает горизонтальное выравнивание заголовка равным left, center или right	
valign=*position*	Устанавливает положение по вертикали заголовка равным top или bottom	
...	Центрирует заключенный в нем текст	
<cite> ... </cite>	Заключенный в нем текст является цитатой	
<code> ... </code>	Заключенный в нем текст является примером программного кода	

`<col>`	Определяет столбец в `<colgroup>`	
`align=`***position***	Устанавливает выравнивание столбцов равным `left`, `center` или `right`	
`char=`***character***	Определяет символ для выравнивания содержимого ячеек	
`charoff=`***value***	Устанавливает смещение в ячейке, в которой происходит выравнивание по символу	
`span=`***n***	Определяет количество столбцов, на которые действует этот тег `<col>`	
`valign=`***position***	Устанавливает вертикальное выравнивание текста в столбце равным `top`, `middle` или `bottom`	
`width=`***n***	Устанавливает ширину столбцов в пикселах или как процентное значение	
`<colgroup>`	Определяет группу столбцов в таблице	
`align=`***position***	Устанавливает горизонтальное выравнивание текста в столбцах равным `left`, `center` или `right`	
`char=`***character***	Определяет символ для выравнивания содержимого ячеек	
`charoff=`***value***	Устанавливает смещение в ячейке, в которой происходит выравнивание по символу	
`span=`***n***	Определяет число столбцов в группе	
`valign=`***position***	Устанавливает вертикальное выравнивание текста в столбце равным `top`, `middle` или `bottom`	
`width=`***n***	Устанавливает ширину, в пикселах или в виде процентного значения, для всех столбцов в группе	
`<comment>` ... `</comment>`	Помещает в документ комментарии (комментарии будут видимыми во всех других броузерах)	 *
`<dd>` ... `</dd>`	Определяет часть списка определений, относящуюся к описанию значения термина	
`<del>` ... `</del>`	Выделяет удаленный фрагмент документа	
`cite=`***url***	Ссылка на документ, обосновывающий удаление	
`datetime=`***date***	Определяет дату и время удаления	
`<dfn>` ... `</dfn>`	Форматирует заключенный в нем текст как определение	
`<dir>` ... `</dir>`	Создает список-каталог, содержащий теги `<li>`	
`type=`***bullet***	Устанавливает стиль маркера для этого списка равным `circle`, `disc` (по умолчанию) или `square`	
`<div>` ... `</div>`	Создает раздел в документе	

align=*type*	Выравнивает текст в разделе, допустимые значения left, center или right	
nowrap	Подавляет в этом разделе автоматическое разбиение текста на строки	
<dl> ... </dl>	Создает список определений, содержащий теги <dt> и <dd>	
compact	Делает список более компактным, если это возможно	
<dt> ... </dt>	Определяет часть списка определений, относящуюся к термину	
...	Форматирует заключенный в нем текст с дополнительным акцентом	
<embed>	Вкладывает в документ внешнее приложение	 *
align=*position*	Выравнивает область вывода, принимая значения top или bottom по отношению к примыкающему тексту или left или right по отношению к полям страницы, так что последующий текст обтекает область вывода	
border=*n*	Определяет размер (в пикселах) рамки вокруг области вывода приложения	
height=*n*	Определяет высоту области вывода приложения в пикселах	
hidden	Если указан, прячет область вывода приложения	
hspace=*n*	Определяет (в пикселах) добавочное пространство, которое должно быть оставлено слева и справа от области вывода приложения	
name=*name*	Предоставляет имя содержимому тега	
palette=*value*	Для Netscape и Opera значение foreground заставляет приложение использовать палитру переднего плана только в Windows; background использует фоновую палитру. Для Internet Explorer и Firefox указывает цвет переднего плана и фоновый цвет для области вывода приложения, записанные как два цветовых значения, разделенных вертикальной чертой (|)	
src=*url*	Доставляет URL данных для плагина	
type=*type*	Определяет MIME-тип данных для определения плагина	
units=*type*	Устанавливает единицы измерения атрибутов высоты и ширины: пикселы (принимается по умолчанию) или en (половина размера текста в пунктах)	

vspace=*n*	Определяет (в пикселах) добавочное пространство, которое должно быть оставлено над областью вывода и под ней	
width=*n*	Определяет ширину области вывода в пикселах	
<fieldset> ... </fieldset>	Создает в форме группу элементов	
 ... 	Устанавливает размер или цвет заключенного в нем текста (нежелателен)	
class=*name*	Определяет класс стилей, управляющий внешним видом этого тега	
color=*color*	Устанавливает цвет заключенного в теге текста	
dir=*dir*	Определяет направление вывода текста либо слева направо (ltr), либо справа налево (rtl)	
face=*list*	Устанавливает первое доступное начертание заключенного в теге текста в разделенном запятыми *списке* названий шрифтов	
id=*name*	Определяет имя для этого тега, уникальное в этом документе	
lang=*language*	Определяет язык, использованный для содержимого этого тега, с применением двухсимвольного названия языка по стандарту ISO	
size=*value*	Устанавливает размер как абсолютный от 1 до 7 или относительно размера, указанного в <basefont>, используя +n или -n (обязательный)	
style=*style*	Определяет встроенный стиль для этого тега	
title=*string*	Определяет название для этого тега	
<form> ... </form>	Ограничивает форму	
accept-charset=*list*	Определяет список наборов символов, приемлемых для обрабатывающего эту форму сервера	
action=*url*	Определяет URL обрабатывающего эту форму приложения (обязательный)	
enctype=*encoding*	Определяет тип кодирования значений элементов формы	
method=*style*	Определяет стиль передачи параметров: либо get, либо post (обязательный)	
name=*name*	Доставляет имя для этой формы, которое может использоваться JavaScript-кодом	
onreset=*applet*	Определяет программный код, который следует исполнить, когда форма обновляется	
onsubmit=*applet*	Определяет программный код, который следует исполнить, когда форма отправляется	

target=*name*	Определяет имя фрейма или окна, которые должны принять результаты формы после отправки	
<frame> ... </frame>	Определяет фрейм в наборе фреймов	*
bordercolor=*color*	Устанавливает цвет рамки фрейма в *color*	
class=*name*	Определяет класс стилей, управляющий внешним видом этого тега	
frameborder=*n*	Если значение равно 1, включает рамку вокруг фрейма. Если значение равно 0, выключает рамку вокруг фрейма	
id=*name*	Определяет имя для этого тега, уникальное в этом документе	
longdesc=*url*	Предоставляет URL документа, описывающего содержимое фрейма	
marginheight=*n*	Помещает *n* пикселов пространства над и под содержимым фрейма	
marginwidth=*n*	Помещает *n* пикселов пространства слева и справа от содержимого фрейма	
name=*name*	Определяет имя фрейма	
noresize	Запрещает изменение размеров фрейма	
scrolling=*type*	Всегда вставлять линейку прокрутки (yes), никогда не вставлять линейку прокрутки (no) или (только для Netscape) вставлять линейку прокрутки по необходимости (auto)	
src=*url*	Определяет URL документа для этого фрейма	
style=*style*	Определяет встроенный стиль для этого тега	
title=*string*	Определяет название для этого тега	
<frameset> ... </frameset>	Определяет набор фреймов или набор наборов фреймов	*
border=*n*	Устанавливает толщину рамки фрейма в этом наборе фреймов	
bordercolor=*color*	Определяет цвет рамки в этом наборе фреймов	
cols=*list*	Определяет число и ширину фреймов в наборе фреймов	
frameborder=*value*	Если значение равно 1, включает рамки вокруг фреймов. Если значение равно 0, выключает рамки вокруг фреймов	
framespacing=*n*	Определяет толщину рамок фреймов в этом наборе фреймов	

`onblur=`***applet***	Определяет программный код, который следует исполнить, когда мышь покидает этот набор фреймов	
`onfocus=`***applet***	Определяет программный код, который следует исполнить, когда мышь попадает в этот набор фреймов	
`onload=`***applet***	Определяет программный код, который следует исполнить, когда этот набор фреймов загружен	
`onunload=`***applet***	Определяет программный код, который следует исполнить, когда этот набор фреймов удаляется с экрана	
`rows=`***list***	Определяет число и высоту фреймов в наборе фреймов	
`<h`***n***`> … </h`***n***`>`	Заключенный в нем текст является заголовком уровня *n*; *n* может быть от 1 до 6	
`align=`***type***	Определяет выравнивание заголовка как `left` (по умолчанию), `center` или `right`	
`<head> ... </head>`	Ограничивает начало и конец заголовка документа	*
`dir=`***dir***	Определяет направление вывода текста либо слева направо (`ltr`), либо справа налево (`rtl`)	
`lang=`***language***	Определяет язык, использованный для содержимого этого тега, с применением двухсимвольного названия языка по стандарту ISO	
`profile=`***url***	Доставляет URL профиля для этого документа	
`<hr>`	Разрывает поток текста и вставляет горизонтальную линейку	*
`align=`***type***	Определяет для линейки выравнивание как `left`, `center` (по умолчанию) или `right`	
`class=`***name***	Определяет класс стилей, управляющий внешним видом линейки	
`color=`***color***	Определяет цвет линейки	
`dir=`***dir***	Определяет направление вывода текста либо слева направо (`ltr`), либо справа налево (`rtl`)	
`id=`***name***	Определяет имя для этого тега, уникальное в этом документе	
`lang=`***language***	Определяет язык, использованный для содержимого этого тега, с применением двухсимвольного названия языка по стандарту ISO	
`noshade`	Отказывается от применения имитирующих трехмерность теней	

`onclick=`***applet***	Определяет программный код, который должен исполняться, когда кнопкой мыши щелкают на этом теге	
`ondblclick=`***applet***	Определяет программный код, который должен исполняться, когда кнопкой мыши дважды щелкают на этом теге	
`onkeydown=`***applet***	Определяет программный код, который должен исполняться, когда нажимают клавишу в то время, как этот тег активен	
`onkeypress=`***applet***	Определяет программный код, который должен исполняться, когда нажимают клавишу и отпускают в то время, как этот тег активен	
`onkeyup=`***applet***	Определяет программный код, который должен исполняться, когда клавишу отпускают в то время, как этот тег активен	
`onmousedown=`***applet***	Определяет программный код, который должен исполняться, когда на этом теге нажимают кнопку мыши	
`onmousemove=`***applet***	Определяет программный код, который должен исполняться, когда мышью проводят над этим тегом	
`onmouseout=`***applet***	Определяет программный код, который должен исполняться, когда мышь выходит за пределы области отображения этого тега	
`onmouseover=`***applet***	Определяет программный код, который должен исполняться, когда мышь попадает в область отображения этого тега	
`onmouseup=`***applet***	Определяет программный код, который должен исполняться, когда кнопку мыши отпускают над этим тегом	
`size=`***pixels***	Устанавливает толщину линейки равной целому числу пикселов	
`style=`***style***	Определяет встроенный стиль для этого тега	
`title=`***string***	Определяет название для этого тега	
`width=`***value*** или %	Устанавливает ширину линейки равной целому числу пикселов или в процентах от ширины страницы	
`<html>` ... `</html>`	Ограничивает начало и конец всего Hypertext Markup Language (HTML) документа	*
`dir=`***dir***	Определяет направление вывода текста либо слева направо (`ltr`), либо справа налево (`rtl`)	

`lang=`***language***	Определяет язык, использованный для содержимого этого тега, с применением двухсимвольного названия языка по стандарту ISO	
`version=`***string***	Обозначает HTML-версию, использованную при создании этого документа	
`<i>` **...** `</i>`	Форматирует заключенный в нем текст *курсивом*	
`<iframe>` … `</iframe>`	Определяет встроенный фрейм	*
`align=`***position***	Устанавливает положение фрейма выровненным относительно `top`, `center` или `bottom` окружающего текста или сдвинутым к `left`- или `right`-краю, в каковом случае текст обтекает фрейм	
`class=`***name***	Определяет класс стилей, управляющий внешним видом фрейма	
`frameborder=`***value***	Если *value* равно `1`, включает рамку вокруг фрейма. Если *value* равно `0`, выключает рамку вокруг фрейма	
`height=`***n***	Устанавливает высоту фрейма в пикселах	
`id=`***name***	Определяет имя для этого тега, уникальное в этом документе	
`longdesc=`***url***	Доставляет URL документа, описывающего содержимое фрейма	
`marginheight=`***n***	Помещает *n* пикселов пространства над и под содержимым фрейма	
`marginwidth=`***n***	Помещает *n* пикселов пространства слева и справа от содержимого фрейма	
`name=`***name***	Определяет имя фрейма	
`scrolling=`***type***	Всегда вставляет линейку прокрутки (`yes`) или никогда не вставляет линейку прокрутки (`no`)	
`src=`***url***	Определяет URL документа для этого фрейма	
`style=`***style***	Определяет встроенный стиль для этого тега	
`title=`***string***	Определяет название для этого тега	
`width=`***n***	Устанавливает ширину фрейма в пикселах	
`<img>`	Вставляет изображение в поток текста	
`align=`***type***	Выравнивает изображение к `top`, `middle`, `bottom` (по умолчанию), `left`, `right`, `absmiddle`, `baseline` или `absbottom` по отношению к тексту	
`alt=`***text***	Доставляет альтернативный текст для нерафических броузеров	

`border=`***n***	Устанавливает в пикселах толщину рамки вокруг изображений, содержащихся в гиперссылках	
`controls`	Добавляет элементы управления для вложенных видеоклипов	
`dynsrc=`***url***	Определяет URL видеоклипа	
`height=`***n***	Определяет высоту изображения	
`hspace=`***n***	Определяет пространство (в пикселах), которое должно быть добавлено слева и справа от изображения	
`ismap`	Обозначает, что изображение можно выбрать при помощи мыши, при использовании с тегом `<a>`	
`longdesc=`***url***	Доставляет URL документа, описывающего изображение	
`loop=`***value***	Устанавливает число раз, которое следует проиграть видео; *value* может быть целым числом или ключевым словом `infinite`	
`lowsrc=`***url***	Определяет изображение низкого разрешения, которое броузер должен загрузить перед изображением, указанным в атрибуте `src`	
`name=`***name***	Предоставляет имя изображения для использования его из JavaScript	
`onabort=`***applet***	Определяет программный код, который должен быть исполнен, если загрузка изображения прервана	
`onerror=`***applet***	Определяет программный код, который должен быть исполнен, если загрузка изображения произошла неудачно	
`onload=`***applet***	Определяет программный код, который должен быть исполнен, если загрузка изображения произошла успешно	
`src=`***url***	Определяет URL изображения, которое должно быть отображено (обязательный)	
`start=`***start***	Определяет, когда следует начать проигрывать видеоклип: либо `fileopen`, либо `mouseover`	
`usemap=`***url***	Определяет место расположения описания набора областей и ассоциированных с ними гиперссылок для карты	
`vspace=`***n***	Определяет пространство (в пикселах), добавляемое сверху и снизу от изображения	
`width=`***n***	Определяет ширину изображения в пикселах	

`<input type=button>`	Создает обычную кнопку в форме (`<form>`)	
`accesskey=`***char***	Определяет горячую клавишу для этого элемента	
`disabled`	Выключает этот элемент управления, делая его неактивным	
`name=`***name***	Определяет имя параметра, который должен передаваться обрабатывающему форму приложению, если кнопка выбрана (обязательный)	
`notab`	Определяет, что этот элемент не является частью табуляционной последовательности	
`onblur=`***applet***	Определяет программный код, который следует исполнить, когда мышь покидает этот элемент управления	
`onfocus=`***applet***	Определяет программный код, который следует исполнить, когда мышь попадает на этот элемент управления	
`tabindex=`***n***	Определяет положение этого элемента в табуляционной последовательности	
`taborder=`***n***	Определяет положение этого элемента в табуляционной последовательности	
`value=`***string***	Определяет значение параметра, передаваемого обрабатывающему форму приложению, если этот элемент формы выбран (обязательный)	
`<input type=checkbox>`	Создает в *форме* (`<form>`) выключатель	
`accesskey=`***char***	Определяет горячую клавишу для этого элемента	
`checked`	Отмечает элемент как изначально выбранный	
`disabled`	Выключает этот элемент управления, делая его неактивным	
`name=`***string***	Определяет имя параметра, который должен передаваться обрабатывающему форму приложению, если элемент ввода выбран (обязательный)	
`notab`	Определяет, что этот элемент не является частью табуляционной последовательности	
`readonly`	Запрещает изменение пользователем этого элемента	
`tabindex=`***n***	Определяет положение этого элемента в табуляционной последовательности	
`taborder=`***n***	Определяет положение этого элемента в табуляционной последовательности	

value=*string*	Определяет значение параметра, передаваемого обрабатывающему форму приложению, если этот элемент формы выбран (обязательный)	
<input type=file>	Создает элемент выбора файла в форме	
accept=*list*	Определяет список MIME-типов файлов, которые принимаются этим элементом	
accesskey=*char*	Определяет горячую клавишу для этого элемента	
disabled	Выключает этот элемент управления, делая его неактивным	
maxlength=*n*	Определяет максимальное число символов, приемлемое для этого элемента	
name=*name*	Определяет имя параметра, который передается обрабатывающему форму приложению для этого элемента ввода (обязательный)	
notab	Определяет, что этот элемент не является частью табуляционной последовательности	
onblur=*applet*	Определяет программный код, который следует исполнить, когда мышь покидает этот элемент управления	
onchange=*applet*	Определяет программный код, который следует исполнить, когда пользователь изменяет значение этого элемента	
onfocus=*applet*	Определяет программный код, который следует исполнить, когда мышь попадает на этот элемент управления	
readonly	Запрещает изменение пользователем этого элемента	
size=*n*	Определяет число символов, которое следует отобразить в этом элементе	
tabindex=*n*	Определяет положение этого элемента в табуляционной последовательности	
taborder=*n*	Определяет положение этого элемента в табуляционной последовательности	
value=*string*	Определяет значение параметра, передаваемого обрабатывающему форму приложению, если этот элемент формы выбран (обязательный)	
<input type=hidden>	Создает в форме скрытый элемент	
name=*name*	Определяет имя параметра, который передается обрабатывающему форму приложению для этого элемента ввода (обязательный)	

`value=`***string***	Определяет значение этого элемента, который передается обрабатывающему форму приложению	
`<input type=image>`	Создает в форме элемент ввода «изображение»	
`accesskey=`***char***	Определяет горячую клавишу для этого элемента	
`align=`***type***	Выравнивает изображение к `top`, `middle` или `bottom` по отношению к тексту в элементе формы	
`alt=`***string***	Предоставляет альтернативное описание для изображения	
`border=`***n***	Устанавливает в пикселах толщину рамки изображения	
`disabled`	Выключает этот элемент управления, делая его неактивным	
`name=`***name***	Определяет имя параметра, который должен передаваться обрабатывающему форму приложению для этого элемента ввода (обязательный)	
`notab`	Определяет, что этот элемент не является частью табуляционной последовательности	
`src=`***url***	Определяет URL изображения (обязательный)	
`tabindex=`***n***	Определяет положение этого элемента в табуляционной последовательности	
`taborder=`***n***	Определяет положение этого элемента в табуляционной последовательности	
`usemap=`***url***	Определяет URL карты, который должен использоваться для этого изображения	
`<input type=password>`	Создает в форме текстовый элемент ввода с защищенным содержимым	
`accesskey=`***char***	Определяет горячую клавишу для этого элемента	
`disabled`	Выключает этот элемент управления, делая его неактивным	
`maxlength=`***n***	Определяет максимальное число символов, приемлемое для этого элемента	
`name=`***name***	Определяет имя параметра, который должен передаваться обрабатывающему форму приложению для этого элемента ввода (обязательный)	
`notab`	Определяет, что этот элемент не является частью табуляционной последовательности	

`onblur=`***applet***	Определяет программный код, который следует исполнить, когда мышь покидает этот элемент	
`onchange=`***applet***	Определяет программный код, который следует исполнить, когда пользователь изменяет значение этого элемента	
`onfocus=`***applet***	Определяет программный код, который следует исполнить, когда мышь попадает на этот элемент	
`onselect=`***applet***	Определяет программный код, который должен быть исполнен, если пользователь щелкает мышью на этом элементе	
`readonly`	Запрещает изменение пользователем этого элемента	
`size=`***n***	Определяет число символов, которое следует отобразить в этом элементе	
`tabindex=`***n***	Определяет положение этого элемента в табуляционной последовательности	
`taborder=`***n***	Определяет положение этого элемента в табуляционной последовательности	
`value=`***string***	Определяет начальное значение для этого элемента	
`<input type=radio>`	Создает в форме элемент ввода «кнопка переключателя»	
`accesskey=`***char***	Определяет горячую клавишу для этого элемента	
`checked`	Отмечает элемент как выбранный изначально	
`disabled`	Выключает этот элемент управления, делая его неактивным	
`name=`***string***	Определяет имя параметра, который должен передаваться обрабатывающему форму приложению, если элемент ввода выбран (обязательный)	
`notab`	Определяет, что этот элемент не является частью табуляционной последовательности	
`readonly`	Запрещает изменение пользователем этого элемента	
`tabindex=`***n***	Определяет положение этого элемента в табуляционной последовательности	
`taborder=`***n***	Определяет положение этого элемента в табуляционной последовательности	

value=*string*	Определяет значение параметра, передаваемого обрабатывающему форму приложению, если этот элемент формы выбран (обязательный)	
<input type=reset>	Создает в форме кнопку обновления формы	
accesskey=*char*	Определяет горячую клавишу для этого элемента	
disabled	Выключает этот элемент управления, делая его неактивным	
notab	Определяет, что этот элемент не является частью табуляционной последовательности	
tabindex=*n*	Определяет положение этого элемента в табуляционной последовательности	
taborder=*n*	Определяет положение этого элемента в табуляционной последовательности	
value=*string*	Определяет альтернативную надпись для кнопки обновления (по умолчанию «Reset»)	
<input type=submit>	Создает в форме кнопку отправки	
accesskey=*char*	Определяет горячую клавишу для этого элемента	
disabled	Выключает этот элемент управления, делая его неактивным	
name=*name*	Определяет имя параметра, который передается обрабатывающему форму приложению для этого элемента ввода (обязательный)	
notab	Определяет, что этот элемент не является частью табуляционной последовательности	
tabindex=*n*	Определяет положение этого элемента в табуляционной последовательности	
taborder=*n*	Определяет положение этого элемента в табуляционной последовательности	
value=*string*	Определяет альтернативную надпись для кнопки отправки, равно как и значение, передаваемое обрабатывающему форму приложению, когда на эту кнопку нажимают	
<input type=text>	Создает в форме текстовый элемент ввода	
accesskey=*char*	Определяет горячую клавишу для этого элемента	
disabled	Выключает этот элемент управления, делая его неактивным	
maxlength=*n*	Определяет максимальное число символов, приемлемое для этого элемента	

name=*name*	Определяет имя параметра, который передается обрабатывающему форму приложению для этого элемента ввода (обязательный)	
notab	Определяет, что этот элемент не является частью табуляционной последовательности	
onblur=*applet*	Определяет программный код, который следует исполнить, когда мышь покидает этот элемент	
onchange=*applet*	Определяет программный код, который следует исполнить, когда пользователь изменяет значение этого элемента	
onfocus=*applet*	Определяет программный код, который следует исполнить, когда мышь попадает на этот элемент	
onselect=*applet*	Определяет программный код, который должен быть исполнен, если пользователь щелкает мышью на этом элементе	
readonly	Запрещает изменение пользователем этого элемента	
size=*n*	Определяет число символов, которое следует отобразить внутри области ввода этого элемента	
tabindex=*n*	Определяет положение этого элемента в табуляционной последовательности	
taborder=*n*	Определяет положение этого элемента в табуляционной последовательности	
value=*string*	Определяет начальное значение для этого элемента	
<ins> … </ins>	Выделяет вставленный раздел документа	
cite=*url*	Ссылка на документ, объясняющий вставку	
datetime=*date*	Определяет дату и час вставки	
<isindex>	Создает поисковый HTML-документ (признан нежелательным)	 *
action=*url*	Только для Internet Explorer; указывает URL программы, которая должна производить поиск	
class=*name*	Определяет класс стилей, управляющий внешним видом этого тега	
dir=*dir*	Определяет направление вывода текста либо слева направо (ltr), либо справа налево (rtl)	
id=*name*	Определяет имя для этого тега, уникальное в этом документе	

lang=*language*	Определяет язык, использованный для содержимого этого тега, с применением двухсимвольного названия языка по стандарту ISO	
prompt=*string*	*Предоставляет* альтернативную подсказку для элемента ввода	
style=*style*	Определяет встроенный стиль для этого тега	
title=*string*	Определяет название для этого тега	
<kbd> ... </kbd>	Заключенный в нем текст изображает ввод с клавиатуры	
<keygen>	Генерирует ключевую последовательность в форме	*
challenge=*string*	Предоставляет строку запроса, упакованную с помощью ключа	
name=*name*	Задает имя для ключа	
<label> ... </label>	Определяет надпись для элемента формы	
accesskey=*char*	Определяет горячую клавишу для этой надписи	
for=*id*	Определяет элемент формы, ассоциированной с этой надписью	
onblur=*applet*	Определяет программный код, который следует исполнить, когда мышь покидает эту надпись	
onfocus=*applet*	Определяет программный код, который следует исполнить, когда мышь попадает на эту надпись	
<legend> ... </legend>	Определяет легенду для набора полей формы	
accesskey=*char*	Определяет горячую клавишу для этой легенды	
align=*position*	Выравнивает легенду к top, bottom, left или right по отношению к набору полей	
...	Ограничивает элемент списка в упорядоченном ( ) или неупорядоченном ( ) списках	
type=*format*	Устанавливает тип этого элемента списка для надлежащего его форматирования. Для в : A (заглавные буквы), a (строчные буквы), I (заглавные римские цифры), i (строчные римские цифры) или 1 (арабские цифры по умолчанию). для в : circle, disc (по умолчанию) или square	
value=*n*	Устанавливает номер этого элемента списка равным *n*	
<link>	Определяет связь текущего документа с другим документом. Присутствует в заголовке документа (в теге <head>)	

`charset=`***charset***	Определяет набор символов, используемых в документе-цели этой ссылки	
`href=`***url***	Определяет URL документа-цели	
`hreflang=`***language***	Определяет язык, использованный для содержимого этого тега, с применением двухсимвольного названия языка по стандарту ISO	
`media=`***list***	Определяет список видов средств вывода, с помощью которых объект может быть представлен пользователю	
`rel=`***relation***	Обозначает отношение документа к документу-цели	
`rev=`***relation***	Обозначает отношение документа-цели к текущему документу	
`type=`***string***	Определяет MIME-тип для присоединенного документа. Обычно используется для таблиц стилей, тип которых `text/css`	
`<map>` ... `</map>`	Содержит описание набора областей и ассоциированных с ними гиперссылок для карты, обрабатываемой со стороны клиента	
`name=`***name***	Определяет имя этой карты (обязательный)	
`<marquee>` ... `</marquee>`	Создает бегущую строку (только Internet Explorer)	*
`align=`***position***	Выравнивает бегущую строку к `top`, `middle` или `bottom` по отношению к окружающему тексту	
`behavior=`***style***	Определяет стиль бегущей строки как `scroll`, `slide` или `alternate`	
`bgcolor=`***color***	Устанавливает фоновый цвет бегущей строки	
`class=`***name***	Определяет класс стилей, управляющий внешним видом этого тега	
`direction=`***dir***	Определяет направления вывода, `left` или `right`, текста бегущей строки	
`height=`***n***	Определяет высоту (в пикселах) области вывода бегущей строки	
`hspace=`***n***	Определяет пространство (в пикселах) вставляемое слева и справа от бегущей строки	
`loop=`***value***	Устанавливает число пробегов бегущей строки; *value* является или целым числом, или ключевым словом `infinite`	
`scrollamount=`***value***	Устанавливает число пикселов, на которое смещается текст при каждом шаге	

scrolldelay=*value*	Определяет задержку (в миллисекундах) между последовательными смещениями текста бегущей строки	
style=*style*	Определяет встроенный стиль для этого тега	
vspace=*n*	Определяет пространство (в пикселах) оставляемое выше и ниже бегущей строки	
width=*n*	Определяет ширину (в пикселах) области вывода бегущей строки	
<menu> ... </menu>	Определяет меню	
type=*bullet*	Устанавливает стиль маркера для этого списка равным circle, disc (по умолчанию) или square	
<meta>	Предоставляет дополнительную информацию о документе	*
content=*string*	Определяет значение для meta-информации (обязательный)	
dir=*dir*	Определяет направление вывода текста либо слева направо (ltr), либо справа налево (rtl)	
http-equiv=*string*	Определяет имя параметра, который может быть использован HTTP-сервером для формирования HTTP-заголовка этого документа при передаче клиенту	
lang=*language*	Определяет язык, использованный для содержимого этого тега, с применением двухсимвольного названия языка по стандарту ISO	
name=*string*	Определяет имя meta-информации	
scheme=*scheme*	Определяет схему профиля, используемого для интерпретации этого свойства	
<nextid>	Определяет следующий допустимый идентификатор сущности (устарел)	 *
n=*n*	Устанавливает следующий идентификатор	
<nobr> ... </nobr>	Запрещает перенос на новую строку для заключенного в нем текста	 *
<noembed> ... </noembed>	Определяет содержимое, которое должны вывести броузеры, не поддерживающие тега <embed>	 *
<noframes> ... </noframes>	Определяет содержимое, которое должны вывести броузеры, не поддерживающие тега фреймов	
<noscript> ... </noscript>	Определяет содержимое, которое должны вывести броузеры, не поддерживающие тега <script>	

`<object>`	Вставляет объект в документ	
`align=`*position*	Выравнивает объект по отношению к окружающему тексту (`texttop`, `middle`, `textmiddle`, `baseline`, `textbottom` и `center`) или по отношению к полям (`left` и `right`), что заставляет последующий поток текста обтекать объект	
`archive=`*list*	Определяет список URL-архивов ресурсов, используемых этим объектом	
`border=`*n*	Определяет ширину рамки объекта (в пикселах)	
`classid=`*url*	Предоставляет URL объекта	
`codebase=`*url*	Предоставляет базовый URL для кодов объекта	
`codetype=`*type*	Определяет MIME-тип объекта	
`data=`*url*	Предоставляет данные для объекта	
`declare`	Объявляет этот объект, не вставляя его	
`height=`*n*	Определяет высоту объекта в пикселах	
`hspace=`*n*	Предоставляет дополнительное пространство (в пикселах) справа и слева от объекта	
`name=`*name*	Определяет имя этого объекта	
`notab`	Исключает этот объект из табуляционной последовательности	
`shapes`	Определяет, что этот объект имеет чувствительные области с ассоциированными гиперссылками	
`standby=`*string*	Определяет сообщение, которое должно отображаться при загрузке объекта	
`tabindex=`*n*	Определяет положение этого объекта в табуляционной последовательности документа	
`type=`*type*	Определяет MIME-тип для данных объекта	
`usemap=`*url*	Определяет изображение карты для использования с этим объектом	
`vspace=`*n*	Предоставляет дополнительное пространство (в пикселах) выше и ниже объекта	
`width=`*n*	Определяет ширину объекта (в пикселах)	
`<ol>` ... `</ol>`	Определяет упорядоченный список, содержащий пронумерованные (в порядке возрастания) `<li>`-элементы	
`compact`	Представляет список более компактным образом	
`start=`*n*	Начинает нумерацию списка с *n* вместо 1	

type=*format*	Устанавливает стиль нумерации для списка: A (заглавные буквы), a (строчные буквы), I (заглавные римские цифры), i (строчные римские цифры) или 1 (арабские цифры по умолчанию)	
<optgroup> ... </optgroup>	Определяет группу вариантов выбора в элементе <select>	
disabled	Выключает эту группу, делая ее неактивной	
label=*string*	Предоставляет метку для этой группы	
<option> ... </option>	Определяет вариант выбора в элементе <select> в форме	
disabled	Выключает этот пункт, делая его неактивным	
label=*string*	Предоставляет метку для этого варианта выбора	
selected	Делает этот элемент изначально выбранным	
value=*string*	Возвращает указанную строку обрабатывающему форму приложению	
...	Начинает и заканчивает абзац	
align=*type*	Выравнивает текст в абзаце к left, center или right	
<param> ... </param>	Доставляет параметр охватывающему тегу <applet>	*
id=*name*	Определяет уникальный идентификатор для этого параметра	
name=*name*	Определяет имя параметра	
type=*type*	Определяет MIME-тип параметра	
value=*string*	Определяет значение параметра	
valuetype=*type*	Определяет тип значения атрибута либо как data, либо как ref (значение является URL, указывающим на данные), либо как object (значение является именем объекта в этом документе)	
<pre> ... </pre>	Выводит заключенный в нем текст в его оригинальном преформатированном стиле, учитывая разрывы строк и пробельные символы буквально	
width=*n*	Выбирает размер текста таким, чтобы по возможности n символов как раз заполняли окно	
<q> ... </q>	Заключенный в нем текст является встроенной в документ цитатой (не поддерживается броузером Internet Explorer)	

cite=*url*	Определяет URL источника цитаты	
<s> ... </s>	То же, что и тег <strike>, заключенный в нем текст зачеркивается горизонтальной чертой	
<samp> ... </samp>	Заключенный в нем текст является примером	
<script> ... </script>	Определяет сценарий в документе	*
charset=*encoding*	Определяет набор символов, использованных для записи сценария	
defer	Откладывает исполнение этого сценария	
language=*encoding*	Определяет язык, использованный для создания сценария	
src=*url*	Предоставляет URL документа, содержащего сценарии	
type=*encoding*	Определяет MIME-тип сценария	
<select> ... </select>	Создает в форме меню или выпадающий список, содержащий один или несколько тегов <option>	
disabled	Выключает этот элемент управления, делая его неактивным	
multiple	Разрешает пользователю выбор более чем одного <option> в <select>	
name=*name*	Определяет имя для значений, которые передаются обрабатывающему форму приложению при условии сделанного выбора (обязательный)	
onblur=*applet*	Определяет программный код, который следует исполнить, когда мышь покидает этот элемент	
onchange=*applet*	Определяет программный код, который следует исполнить, когда пользователь изменяет значение этого элемента	
onfocus=*applet*	Определяет программный код, который следует исполнить, когда мышь попадает на этот элемент	
size=*n*	Отображает *n* элементов, используя выпадающее меню для size=1 (если не указан multiple) и выпадающий список из *n* элементов в противном случае	
tabindex=*n*	Определяет положение этого элемента в табуляционной последовательности	
...	Форматирует заключенный в нем текст, используя меньший шрифт	
...	Определяет кусок текста для применения стиля	

`<strike> ... </strike>`	Заключенный в нем текст зачеркивается горизонтальной чертой	
`<strong> ... </strong>`	Выражает сильное логическое ударение на заключенном в нем тексте	
`<style> ... </style>`	Определяет один или несколько стилей на уровне документа	*
`dir=`***dir***	Определяет направление вывода для текста названия или слева направо (`ltr`), или справа налево (`rtl`)	
`lang=`***language***	Определяет язык, использованный для содержимого этого тега, с применением двухсимвольного названия языка по стандарту ISO	
`media=`***list***	Определяет список видов посредников, посредством которых объект может быть представлен пользователю	
`title=`***string***	Определяет название для этого тега	
`type=`***type***	Определяет формат стилей (всегда `text/css`)	
`<sub> ... </sub>`	Форматирует заключенный в нем текст как нижний индекс	
`<sup> ... </sup>`	Форматирует заключенный в нем текст как верхний индекс	
`<table> ... </table>`	Определяет таблицу	
`align=`***position***	Выравнивает таблицу по центру. При этом поток текста будет обтекать таблицу	
`background=`***url***	Определяет фоновое изображение для таблицы	
`bgcolor=`***color***	Определяет фоновый цвет для всей таблицы	
`border=`***n***	Создает рамку шириной в *n* пикселов	
`bordercolor=`***color***	Определяет цвет рамки для всей таблицы	
`bordercolordark=`***color***	Определяет цвет для тени выделяющей рамки всей таблицы	
`bordercolorlight=`***color***	Определяет цвет бликов выделяющей рамки всей таблицы	
`cellpadding=`***n***	Помещает *n* пикселов подложки вокруг содержимого каждой ячейки	
`cellspacing=`***n***	Помещает *n* пикселов в промежутке между ячейками	
`cols=`***n***	Определяет число столбцов в этой таблице	
`frame=`***type***	Определяет, где отображаются рамки таблицы: `border` (по умолчанию), `void`, `above`, `below`, `hsides`, `lhs`, `rhs`, `vsides` или `box`	

height=*n*	Определяет высоту таблицы в пикселах	
hspace=*n*	Определяет пространство (в пикселах), добавляемое справа и слева от таблицы	
nowrap	Подавляет автоматическое разбиение на строки текста в ячейках таблицы	
rules=*edges*	Определяет, где проводятся разделяющие ячейки линейки: `all` (по умолчанию), `groups` (только вокруг групп строк и столбцов), `rows`, `cols` или `none`	
summary=*string*	Доставляет сводное описание этой таблицы	
valign=*position*	Выравнивает текст в таблице к `top`, `center`, `bottom` или `baseline`	
vspace=*n*	Определяет пространство (в пикселах), добавляемое сверху и снизу от таблицы	
width=*n*	Устанавливает ширину таблицы в *n* пикселов или в процентах от ширины окна	
`<tbody>` ... `</tbody>`	Создает группу строк в таблице	
align=*position*	Выравнивает содержимое ячеек к `left`, `center` или `right`	
char=*char*	Определяет символ, по которому происходит выравнивание в группе	
charoff=*value*	Определяет смещение положения выравнивания в ячейках	
valign=*position*	Выравнивает по вертикали содержимое в ячейках группы к `top`, `center`, `bottom` или `baseline`	
`<td>` ... `</td>`	Определяет ячейки данных в таблице	
abbr=*string*	Определяет сокращение для содержимого ячеек	
align=*position*	Выравнивает содержимое ячеек к `left`, `center` или `right`	
axis=*string*	Предоставляет имя для группы ячеек	
background=*url*	Определяет фоновое изображение для этой ячейки	
bgcolor=*color*	Определяет фоновый цвет для ячейки	
bordercolor=*color*	Определяет цвет рамки для ячейки	
bordercolordark=*color*	Определяет цвет тени выделяющей рамки для ячейки	
bordercolorlight=*color*	Определяет цвет блика выделяющей рамки для ячейки	
char=*char*	Определяет символ для выравнивания содержимого ячеек	

`charoff=`***value***	Определяет смещение положения выравнивания в ячейке
`colspan=`***n***	Создает ячейку, охватывающую *n* столбцов
`headers=`***list***	Доставляет список идентификаторов заголовочных ячеек, ассоциированных с этой ячейкой
`height=`***n***	Определяет высоту (в пикселах) для этой ячейки
`nowrap`	Отменяет автоматическое разбиение текста на строки текста в этой ячейке 
`rowspan=`***n***	Создает ячейку, охватывающую *n* строк
`scope=`***scope***	Определяет область действия этой заголовочной ячейки: `row`, `col`, `rowgroup` или `colgroup`
`valign=`***position***	Выравнивает по вертикали содержимое этой ячейки к `top`, `center`, `bottom` или `baseline` 
`width=`***n***	Устанавливает ширину этой ячейки равной *n* пикселов или в процентах от ширины таблицы
`<textarea> ... </textarea>`	Определяет многострочную область ввода текста в форме; содержимое тега `<textarea>` является начальным, принимаемым по умолчанию значением
`accesskey=`***char***	Определяет горячую клавишу для этого элемента
`cols=`***n***	Отображает *n* столбцов (символов) текста в области
`disabled`	Выключает этот элемент управления, делая его неактивным
`name=`***string***	Определяет имя для значения области текста, которое передается обрабатывающему форму приложению (обязательный)
`onblur=`***applet***	Определяет программный код, который следует исполнить, когда мышь покидает этот элемент
`onchange=`***applet***	Определяет программный код, который следует исполнить, когда пользователь изменяет значение этого элемента
`onfocus=`***applet***	Определяет программный код, который следует исполнить, когда мышь попадает на этот элемент
`onselect=`***applet***	Определяет программный код, который должен быть исполнен, если пользователь щелкает мышью на этом элементе
`readonly`	Запрещает изменение пользователем этого элемента

rows=*n*	Отображает *n* строк текста в области	
tabindex=*n*	Определяет положение этого элемента в табуляционной последовательности	
<tfoot> ... </tfoot>	Определяет для таблицы нижний колонтитул	
align=*position*	Выравнивает содержимое ячеек нижнего колонтитула к left, center или right	
char=*char*	Определяет символ для выравнивания содержимого ячеек	
charoff=*value*	Определяет смещение положения выравнивания в ячейке	
valign=*position*	Выравнивает по вертикали содержимое ячеек колонтитула к top, center, bottom или baseline	
<th> ... </th>	Определяет заголовочную ячейку таблицы	
abbr=*string*	Определяет сокращение для содержимого ячеек	
align=*position*	Выравнивает содержимое ячеек к left, center или right	
axis=*string*	Устанавливает имя для группы ячеек	
background=*url*	Определяет фоновое изображение для этой ячейки	
bgcolor=*color*	Определяет фоновый цвет для ячейки	
bordercolor=*color*	Определяет цвет рамки для ячейки	
bordercolordark=*color*	Определяет цвет тени выделяющей рамки для ячейки	
bordercolorlight=*color*	Определяет цвет блика выделяющей рамки для ячейки	
char=*char*	Определяет символ для выравнивания содержимого ячеек	
charoff=*value*	Определяет смещение положения выравнивания в ячейке	
colspan=*n*	Создает ячейку, охватывающую *n* столбцов	
headers=*list*	Предоставляет список идентификаторов заголовочных ячеек, ассоциированных с этой ячейкой	
height=*n*	Определяет высоту для этой ячейки (в пикселах)	
nowrap	Отменяет автоматическую расстановку переносов текста в этой ячейке	
rowspan=*n*	Создает ячейку, охватывающую *n* строк	
scope=*scope*	Определяет область действия этой заголовочной ячейки: row, col, rowgroup или colgroup	

`valign=`***position***	Выравнивает по вертикали содержимое этой ячейки к `top`, `center`, `bottom` или `baseline`	
`width=`***n***	Устанавливает ширину этой ячейки равной *n* пикселов или в процентах от ширины таблицы	
`<thead>` **...** `</thead>`	Определяет верхний колонтитул таблицы	
`align=`***position***	Выравнивает содержимое ячеек колонтитула к `left`, `center` или `right`	
`char=`***char***	Определяет символ для выравнивания содержимого ячеек	
`charoff=`***value***	Определяет смещение положения выравнивания в ячейке	
`valign=`***position***	Выравнивает по вертикали содержимое заголовочных ячеек к `top`, `center`, `bottom` или `baseline`	
`<title>` **...** `</title>`	Определяет название HTML-документа	*
`dir=`***dir***	Определяет направление вывода текста либо слева направо (`ltr`), либо справа налево (`rtl`)	
`lang=`***language***	Определяет язык, использованный для содержимого этого тега, с применением двухсимвольного названия языка по стандарту ISO	
`<tr>` **...** `</tr>`	Определяет строку ячеек в таблице	
`align=`***type***	Выравнивает содержимое ячеек в этой строке к `left`, `center` или `right`	
`background=`***url***	Определяет фоновое изображение для этой строки	
`bgcolor=`***color***	Определяет фоновый цвет для этой строки	
`bordercolor=`***color***	Для Internet Explorer определяет цвет рамки для этой строки	
`bordercolordark=`***color***	Для Internet Explorer определяет цвет тени выделяющей рамки ячейки	
`bordercolorlight=`***color***	Для Internet Explorer определяет цвет блика выделяющей рамки ячейки	
`char=`***char***	Определяет символ для выравнивания содержимого ячеек	
`charoff=`***value***	Определяет смещение положения выравнивания в этой строке	
`nowrap`	Отменяет автоматическую расстановку переносов текста в ячейках этой строки	
`valign=`***position***	Выравнивает по вертикали содержимое ячеек в этой строке к `top`, `center`, `bottom` или `baseline`	
`<tt>` **...** `</tt>`	Форматирует заключенный в нем текст в стиле телетайпа (моноширинным шрифтом)	

`<u>` **...** `</u>`	Заключенный в нем текст является подчеркнутым	
`<ul>` **...** `</ul>`	Определяет неупорядоченный список маркированных элементов `<li>`	
`compact`	Отображает список более компактным образом, если возможно	
`type=`***bullet***	Устанавливает стиль маркера для этого списка равным `circle`, `disc` (по умолчанию) или `square`	
`<var>` **...** `</var>`	Заключенный в нем текст является именем переменной	
`<wbr>`	Обозначает место допустимого разрыва строки внутри тега `<nobr>`	 *

C

Краткий справочник по свойствам каскадных таблиц стилей

В следующей таблице в алфавитном порядке перечислены все свойства, определяемые в рекомендуемых спецификациях для каскадных таблиц стилей уровня 2, предложенных World Wide Web Consortium (*http://www.w3.org/pub/WWW/TR/REC-CSS2*). В таблицу включены возможные значения каждого свойства, определенные либо как ключевое слово (набранное моноширинным шрифтом), либо как одно из следующих значений:

angle

Числовое значение, за которым следует `deg`, `grad` или `rad`.

color

Либо название цвета, либо его шестнадцатеричное RGB-значение, как это определяется в приложении G, либо RGB-триплет в следующей форме:

```
rgb(red, green, blue)
```

где *red*, *green*, *blue* – это или числа в интервале от `0` до `255`, или процентные значения, обозначающие яркость соответствующего компонента цвета. Значения `255` или `100%` означают, что соответствующий компонент цвета имеет максимальную яркость, значения `0` или `0%` означают, что соответствующий компонент цвета полностью выключен. К примеру:

```
rgb(27, 119, 207)
rgb(50%, 75%, 0%)
```

являются допустимыми спецификациями цвета.

frequency

Числовое значение, за которым следует `hz` или `khz`, то есть, герцы или килогерцы.

length

Необязательный знак (+ или –), за которым непосредственно следует число (возможно, с десятичной точкой), за которым непосредственно следует двухбуквенное обозначение единиц измерения. Для нулевых значений размерность можно не указывать.

Единицы `em` и `ex` соответствуют полной высоте текущего шрифта и высоте буквы «x» в текущем шрифте, соответственно. Единица `px` равна одному пикселу на отображающем устройстве. Обозначения `in`, `cm`, `mm`, `pt` и `pc` указывают на дюймы, сантиметры, миллиметры, пункты и пики, соответственно. В одном дюйме 72,27 пункта, а в пике 12 пунктов.

number

Необязательный знак (+ или –), за которым непосредственно следует число (возможно, с десятичной точкой).

percent

Необязательный знак (+ или –), за которым непосредственно следует число (возможно, с десятичной точкой), за которым непосредственно следует знак процента. Действительное значение вычисляется относительно другого свойства элемента, обычно относительно размера самого элемента.

shape

Ключевое слово, обозначающее фигуру, за которым следует список параметров, характерных для этой фигуры. Список заключен в скобки, и его элементы разделены запятыми. В настоящее время поддерживается только ключевое слово `rect` (прямоугольник), для которого действительны четыре числовых параметра, определяющих смещение верхней, правой, нижней и левой сторон прямоугольника.

time

Числовое значение, за которым следует `s` или `ms`, то есть, секунды или миллисекунды.

url

Ключевое слово `url`, за которым следуют (без пробелов): левая скобка, URL (его по желанию можно заключить в одиночные или двойные кавычки), закрывающая правая скобка. К примеру:

```
url("http://www.oreilly.com/catalog")
```

является действительным значением URL.

Наконец, некоторые значения свойств являются списками значений, принадлежащих к одному из этих типов, и описываются в таблице как «список» из каких-либо других значений. В этих случаях список со-

стоит из одного или нескольких значений допустимого типа, разделенных запятыми.

Если свойство может иметь несколько различных значений, эти альтернативы разделяются вертикальной чертой (|).

Если стандарт определяет значение, принимаемое по умолчанию, такое значение подчеркивается.

azimuth	*angle* | left-side | far-left | left | center-left | center | center-right | right | far-right | right-side	Определяет позицию кажущегося источника звука по отношению к слушателю	8.4.12.7
background		Композиция свойств background-attachment, background-color, background-image, background-position и background-repeat; его значения – это значения перечисленных свойств в произвольном порядке	8.4.5.6
background-attachment	scroll | fixed	Определяет, прокручивается ли фоновое изображение вместе с документом или занимает фиксированное положение в окне	8.4.5.3
background-color	*color* | transparent	Устанавливает цвет фона элемента	8.4.5.1
background-image	*url* | none	Устанавливает фоновое изображение элемента	8.4.5.2
background-position	*percent* | *length* | top | center | bottom | left | right	Устанавливает, если указан, начальное положение фонового изображения элемента; обычно значения представляют собой пару x, y. По умолчанию принимается 0% 0%	8.4.5.4
background-repeat	repeat | repeat-x | repeat-y | no-repeat	Определяет, как размножается (укладывается) фоновое изображение в элементе	8.4.5.5
border		Устанавливает все четыре стороны рамки элемента; значением является один или несколько цветов, значение для свойства border-width и значение для свойства border-style	8.4.7.6

border-bottom		Устанавливает нижнюю сторону рамки элемента; значением является один или несколько цветов, значение для свойства border-bottom-width и значение для свойства border-style	8.4.7.6
border-bottom-width	*length* | thin | medium | thick	Устанавливает толщину нижней рамки элемента	8.4.7.4
border-collapse	collapse | separate	Указывает алгоритм вывода рамки таблицы	8.4.9.1
border-color	*color*	Устанавливает цвет всех четырех сторон рамки элемента; по умолчанию принимает цвет элемента	8.4.7.3
border-left		Устанавливает левую сторону рамки элемента; значением является один или несколько цветов, значение для свойства border-left-width и значение для свойства border-style	8.4.7.6
border-left-width	*length* | thin | medium | thick	Устанавливает толщину рамки элемента слева	8.4.7.4
border-right		Устанавливает правую сторону рамки элемента; значением является один или несколько цветов, значение для свойства border-right-width и значение для свойства border-style	8.4.7.6
border-right-width	length | thin | medium | thick	Устанавливает толщину рамки элемента справа	8.4.7.4
border-spacing		При наличии отдельных рамок устанавливает расстояние между ними; одно значение устанавливает одинаковое расстояние и по вертикали, и горизонтали; два значения – по вертикали и горизонтали соответственно	8.4.9.1
border-style	dashed | dotted | double | groove | inset | none | outset | ridge | solid	Устанавливает стиль всех четырех сторон рамки элемента	8.4.7.5

border-top		Устанавливает верхнюю сторону рамки элемента; значением является один или несколько цветов, значение для свойства border-top-width и значение для свойства border-style	8.4.7.6
border-top-width	*length* | thin | medium | thick	Устанавливает толщину рамки элемента сверху	8.4.7.4
border-width	*length* | thin | medium | thick	Устанавливает толщину всех четырех сторон рамки элемента	8.4.7.4
bottom	*length* | *percent*	Используется с позиционирующим свойством для расположения нижнего края элемента	8.4.7.14
caption-side	top | bottom | left | right	Устанавливает позицию заголовка таблицы	8.4.9.2
clear	both | left | none | right	Устанавливает, какие стороны элемента не должны соседствовать с плавающим элементом; элемент будет опущен вниз так, чтобы эта сторона была свободной	8.4.7.7
clip	*shape*	Устанавливает маску обрезки элемента	8.4.7.8
color	*color*	Устанавливает цвет элемента	8.4.5.7
content		Располагает сгенерированное содержимое вокруг элемента; подробности см. в тексте	8.4.11.2
counter-increment		Увеличивает счетчик на единицу; значение является списком имен счетчиков, каждое из которых может сопровождаться значением, на которое счетчик увеличивается	8.4.11.4
counter-reset		Сбрасывает счетчик в ноль; значение является списком имен счетчиков, каждое из которых может сопровождаться значением, в которое счетчик сбрасывается	8.4.11.4
cue-after	*url* | none	Воспроизводит указанный звук после проговаривания элемента	8.4.12.5

cue-before	*url* | none	Воспроизводит указанный звук до проговаривания элемента	8.4.12.5
display	block | inline | list-item | marker | none	Управляет тем, как отображается элемент	8.4.10.1
elevation	*angle* | below | level | above | higher | lower	Устанавливает высоту воспроизведения звука	8.4.12.7
empty-cells	hide | show	При раздельных рамках скрывает пустые ячейки таблицы	8.4.9.1
float	left | none | right	Определяет, является ли элемент плавающим и, если да, то к какому краю он будет смещаться, позволяя тексту обтекать себя, к левому или правому. В противном случае (none) элемент отображается в строку	8.4.7.9
font		Устанавливает все атрибуты шрифта для элемента; значением служат перечисленные в том же, что и здесь, порядке значения свойств font-style, font-variant, font-weight, font-size, line-height и font-family	8.4.3.8
font-family	список названий шрифтов	Определяет шрифт для элемента либо в виде конкретного наименования шрифта, либо в виде названия семейства шрифтов serif, sans-serif, cursive, fantasy или monospace	8.4.3.1
font-size	xx-small | x-small | small | medium | large | x-large | xx-large | larger | smaller | ***length*** | ***percent***	Определяет размер шрифта	8.4.3.2
font-size-adjust	none | ***ratio***	Подгоняет аспектное отношение текущего шрифта	8.4.3.4

font-stretch	wider | normal | narrower | ultra-condensed | extra-condensed | condensed | semi-condensed | semi-expanded | expanded | extra-expanded | ultra-expanded	Определяет степень сжатия или расширения текущего шрифта	8.4.3.3
font-style	normal | italic | oblique	Определяет стиль начертания или как нормальный, или как один из вариантов наклонного шрифта	8.4.3.5
font-variant	normal | small-caps	Определяет, что буквы шрифта должны иметь начертание «маленькие заглавные», или капитель	8.4.3.6
font-weight	normal | bold | bolder | lighter | *number*	Определяет жирность шрифта. Если используется *number* (числовое значение), оно должно быть кратным 100 и лежать в интервале от 100 до 900; 400 соответствует normal, 700 – это то же, что bold	8.4.3.7
height	*length* | auto	Определяет высоту элемента	8.4.7.10
left	*length* | *percent*	Используется с позиционирующим свойством для расположения левого края элемента	8.4.7.14
letter-spacing	*length* | normal	Определяет величину дополнительного расстояния между символами	8.4.6.1
line-height	*length* | *number* | *per cent* | normal	Устанавливает расстояние между опорными линиями соседних строк	8.4.6.2
list-style		Определяет стили, относящиеся к спискам, используя значения для свойств list-style-image, list-style-position и list-style-type	8.4.8.4
list-style-ima ge	*url* | none	Определяет изображение, которое должно использоваться в качестве маркера элементов списка вместо значений свойства list-style-type	8.4.8.1

list-style-position	inside | outside	Вдвигает или выдвигает (по умолчанию) маркер элемента списка по отношению к содержимому элемента	8.4.8.2
list-style-type	circle | disc | square | decimal | lower-alpha | lower-roman | none | upper-alpha | upper-roman	Определяет маркер списка как для неупорядоченных списков (circle, disc или square), так и для упорядоченных списков (decimal, lower-alpha, lower-roman, none, upper-alpha или upper-roman)	8.4.8.3
margin	*length* | *percent* | auto	Определяет поля со всех четырех сторон элемента	8.4.7.11
margin-bottom	*length* | *percent* | auto	Определяет поле снизу от элемента; по умолчанию 0	8.4.7.11
margin-left	*length* | *percent* | auto	Определяет поле слева от элемента; по умолчанию 0	8.4.7.11
margin-right	*length* | *percent* | auto	Определяет поле справа от элемента; по умолчанию 0	8.4.7.11
margin-top	*length* | *percent* | auto	Определяет поле сверху от элемента; по умолчанию 0	8.4.7.11
orphans	*number*	Устанавливает минимальное количество строк в висячем фрагменте абзаца	8.4.13.5
overflow	auto | hidden | scroll | visible	Определяет способ вывода переполняющего содержимого	8.4.7.13
padding		Определяет размеры подложки элемента со всех четырех сторон	8.4.7.12
padding-bottom	*length* | *percent*	Определяет размеры подложки элемента снизу; по умолчанию 0	8.4.7.12
padding-left	*length* | *percent*	Определяет размеры подложки элемента слева; по умолчанию 0	8.4.7.12
padding-right	*length* | *percent*	Определяет размеры подложки элемента справа; по умолчанию 0	8.4.7.12
padding-top	*length* | *percent*	Определяет размеры подложки элемента сверху; по умолчанию 0	8.4.7.12
page	*name*	Ассоциирует с элементом именованную компоновку страницы	8.4.13.3

page-break-after	auto | always | avoid | left | right	Принудительно выполняет или подавляет переход на новую страницу после элемента	8.4.13.4
page-break-before	auto | always | avoid | left | right	Принудительно выполняет или подавляет переход на новую страницу перед элементом	8.4.13.4
page-break-inside	auto | avoid	Принудительно выполняет или подавляет переход на новую страницу в пределах элемента	8.4.13.4
pause-after	*percent* | *time*	Делает паузу после проговаривания элемента	8.4.12.4
pause-before	*percent* | *time*	Делает паузу до проговаривания элемента	8.4.12.4
pitch	*frequency* | x-low | low | medium | high | x-high	Устанавливает среднюю высоту для проговариваемого содержимого в элементе	8.4.12.3
pitch-range	*number*	Устанавливает диапазон для высоты от 0 (монотонное проговаривание) до 100 (богатое интонациями); умолчание – 50	8.4.12.3
play-during	*url* | mix | none | repeat	Если указан URL-адрес, звуковой файл воспроизводится как фон для проговариваемого содержимого; repeat зацикливает воспроизведение, а mix микширует звук с другим фоновым звучанием, не замещая его	8.4.12.6
position	absolute | fixed | relative | static	Указывает модель позиционирования элемента	8.4.7.14
quotes	Список строк	Указывает символы, обозначающие цитату	8.4.11.3
richness	*number*	Устанавливает богатство голоса от 0 (монотонный) до 100 (насыщенный обертонами); умолчание – 50	8.4.12.3
right	*length* | *percent*	Используется со свойством позиционирования для расположения правого края элемента	8.4.7.14

speak	normal | none | spell-out	Определяет способ проговаривания содержимого элемента	8.4.12.2
speak-header	always | once	Определяет способ проговаривания табличных заголовков: один раз для каждого столбца или строки, или же для каждой ячейки	8.4.9.3
speak-numeral	continuous | digits	Определяет способ проговаривания чисел	8.4.12.2
speak-punctuation	code | none	Определяет, произносятся ли знаки препинания, или они управляют интонацией	8.4.12.2
speech-rate	*number* | x-slow | slow | medium | fast | x-fast | faster | slower	Устанавливает темп речи (количество слов в минуту)	8.4.12.3
stress	*number*	Устанавливает силу ударения в голосе, от 0 (слабое) до 100 (очень сильное); умолчание – 50	8.4.12.3
table-layout	auto | fixed	Определяет алгоритм вывода таблицы	8.4.9.4
text-align	center | justify | left | right	Устанавливает стиль выравнивания текста в элементе	8.4.6.3
text-decoration	blink | line-through | none | overline | underline	Определяет декорирование текста; значения могут комбинироваться	8.4.6.4
text-indent	*length* | *percent*	Определяет отступы для первой строки элемента; по умолчанию 0	8.4.6.5
text-shadow	Смотрите text	Создает тени разного размера и цвета, отбрасываемые элементом	8.4.6.6
text-transform	capitalize | lowercase | none | uppercase	Преобразует регистр текста элемента	8.4.6.7
top	*length*|*percent*	Используется со свойством позиционирования для расположения верхнего края элемента	8.4.7.14

vertical-align	*percent* | *baseline* | bottom | middle | sub | super | text-bottom | text-top | top	Устанавливает положение элемента по вертикали	8.4.6.8
visibility	collapse | hidden | visible	Определяет, виден ли элемент в документе или таблице	8.4.7.15
voice-family	Список голосов	Выбирает именованное семейство голосов для проговаривания содержимого	8.4.12.3
volume	*number* | percent | silent | x-soft | soft | medium | loud | x-loud	Устанавливает громкость проговаривания содержимого; диапазон значений от 0 до 100	8.4.12.1
white-space	normal | nowrap | pre	Определяет, как следует обращаться с пробельными элементами в данном элементе	8.4.10.2
widows	*number*	Устанавливает минимальное количество висячих строчек во фрагменте абзаца	8.4.13.5
width	*length* | *percent* | auto	Определяет ширину элемента	8.4.7.16
word-spacing	*length* | normal	Увеличивает пробел между словами	8.4.6.9
z-index	*number*	Устанавливает слой вывода на экран для текущего элемента	8.4.7.17

D

HTML 4.01 DTD

Стандарт HTML 4.01 определяется формально в виде трех SGML-определений типа документа (DTD, Document Type Definition) – строгого (Strict) DTD, переходного (Transitional) DTD и фреймового (Frameset) DTD. Строгое DTD определяет только те элементы, которые не признаны нежелательными в стандарте 4.0. В идеале все должны писать документы, соответствующие строгому DTD. Переходное DTD включает в себя все эти нежелательные элементы и более точно отражает современное употребление HTML, в котором все еще широко используются устаревшие элементы. Фреймовое DTD совпадает с переходным за тем исключением, что тело документа `<body>` заменяется на тег `<frameset>`.

Поскольку переходное DTD широчайшим образом покрывает все множество ныне употребляемых элементов HTML, это то DTD, на котором основана эта книга, и именно его мы воспроизводим. Отметьте, что мы перепечатываем DTD дословно и не пытаемся включить в него расширения. Там, где наше описание и DTD расходятся, считайте правильным оригинал DTD.

```
 <!--
This is the HTML 4.01 Transitional DTD, which includes presentation
attributes and elements that W3C expects to phase out as support for style
sheets matures. Authors should use the Strict DTD when possible, but may use
the Transitional DTD when support for presentation attribute and elements is
required.
HTML 4 includes mechanisms for style sheets, scripting, embedding objects,
improved support for right to left and mixed direction text, and enhancements
to forms for improved accessibility for people with disabilities.
           Draft: $Date: 2006/09/27 15:34:23 $
           Authors:
               Dave Raggett <dsr@w3.org>
               Arnaud Le Hors <lehors@w3.org>
               Ian Jacobs <ij@w3.org>
```

```
     Further information about HTML 4.01 is available at:
         http://www.w3.org/TR/1999/REC-html401-19991224
     The HTML 4.01 specification includes additional
     syntactic constraints that cannot be expressed within
     the DTDs.
 -->
 <!ENTITY % HTML.Version "-//W3C//DTD HTML 4.01 Transitional//EN"
   -- Typical usage:
     <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
             "http://www.w3.org/TR/html4/loose.dtd">
     <html>
     <head>
     ...
     </head>
     <body>
     ...
     </body>
     </html>
The URI used as a system identifier with the public identifier allows the
user agent to download the DTD and entity sets as needed.
     The FPI for the Strict HTML 4.01 DTD is:
         "-//W3C//DTD HTML 4.01//EN"
     This version of the strict DTD is:
         http://www.w3.org/TR/1999/REC-html401-19991224/strict.dtd
Authors should use the Strict DTD unless they need the      presentation
control for user agents that don't (adequately)      support style sheets.
If you are writing a document that includes frames, use      the following
FPI:
         "-//W3C//DTD HTML 4.01 Frameset//EN"
     This version of the frameset DTD is:
         http://www.w3.org/TR/1999/REC-html401-19991224/frameset.dtd
     Use the following (relative) URIs to refer to
     the DTDs and entity definitions of this specification:
     "strict.dtd"
     "loose.dtd"
     "frameset.dtd"
     "HTMLlat1.ent"
     "HTMLsymbol.ent"
     "HTMLspecial.ent"
 -->
 <!--============== Imported Names ========================-->
 <!-- Feature Switch for frameset documents -->
 <!ENTITY % HTML.Frameset "IGNORE">
 <!ENTITY % ContentType "CDATA"
     -- media type, as per [RFC2045]
     -->
 <!ENTITY % ContentTypes "CDATA"
     -- comma-separated list of media types, as per [RFC2045]
     -->
 <!ENTITY % Charset "CDATA"
     -- a character encoding, as per [RFC2045]
```

```
    -->
<!ENTITY % Charsets "CDATA"
    -- a space-separated list of character encodings, as per [RFC2045]
    -->
<!ENTITY % LanguageCode "NAME"
    -- a language code, as per [RFC1766]
    -->
<!ENTITY % Character "CDATA"
    -- a single character from [ISO10646]
    -->
<!ENTITY % LinkTypes "CDATA"
    -- space-separated list of link types
    -->
<!ENTITY % MediaDesc "CDATA"
    -- single or comma-separated list of media descriptors
    -->
<!ENTITY % URI "CDATA"
    -- a Uniform Resource Identifier,
       see [URI]
    -->
<!ENTITY % Datetime "CDATA" -- date and time information. ISO date format -->
<!ENTITY % Script "CDATA" -- script expression -->
<!ENTITY % StyleSheet "CDATA" -- style sheet data -->
<!ENTITY % FrameTarget "CDATA" -- render in this frame -->
<!ENTITY % Text "CDATA">
<!-- Parameter Entities -->
<!ENTITY % head.misc "SCRIPT|STYLE|META|LINK|OBJECT" -- repeatable head
elements -->
<!ENTITY % heading "H1|H2|H3|H4|H5|H6">
<!ENTITY % list "UL | OL |  DIR | MENU">
<!ENTITY % preformatted "PRE">
<!ENTITY % Color "CDATA" -- a color using sRGB: #RRGGBB as Hex values -->
<!-- There are also 16 widely known color names with their sRGB values:
    Black  = #000000    Green  = #008000
    Silver = #C0C0C0    Lime   = #00FF00
    Gray   = #808080    Olive  = #808000
    White  = #FFFFFF    Yellow = #FFFF00
    Maroon = #800000    Navy   = #000080
    Red    = #FF0000    Blue   = #0000FF
    Purple = #800080    Teal   = #008080
    Fuchsia= #FF00FF    Aqua   = #00FFFF
 -->
<!ENTITY % bodycolors "
  bgcolor     %Color;        #IMPLIED  -- document background color --
  text        %Color;        #IMPLIED  -- document text color --
  link        %Color;        #IMPLIED  -- color of links --
  vlink       %Color;        #IMPLIED  -- color of visited links --
  alink       %Color;        #IMPLIED  -- color of selected links --
  ">
<!--============= Character mnemonic entities ==============-->
<!ENTITY % HTMLlat1 PUBLIC
```

```
    "-//W3C//ENTITIES Latin1//EN//HTML"
    "HTMLlat1.ent">
 %HTMLlat1;
 <!ENTITY % HTMLsymbol PUBLIC
    "-//W3C//ENTITIES Symbols//EN//HTML"
    "HTMLsymbol.ent">
 %HTMLsymbol;
 <!ENTITY % HTMLspecial PUBLIC
    "-//W3C//ENTITIES Special//EN//HTML"
    "HTMLspecial.ent">
 %HTMLspecial;
 <!--============= Generic Attributes ======================-->
 <!ENTITY % coreattrs
  "id          ID             #IMPLIED  -- document-wide unique id --
   class       CDATA          #IMPLIED  -- space-separated list of classes --
   style       %StyleSheet;   #IMPLIED  -- associated style info --
   title       %Text;         #IMPLIED  -- advisory title --"
   >
 <!ENTITY % i18n
  "lang        %LanguageCode; #IMPLIED  -- language code --
   dir         (ltr|rtl)     #IMPLIED  -- direction for weak/neutral text --"
   >
 <!ENTITY % events
  "onclick     %Script;       #IMPLIED  -- a pointer button was clicked --
   ondblclick  %Script;       #IMPLIED  -- a pointer button was double
clicked--
   onmousedown %Script;       #IMPLIED  -- a pointer button was pressed down --
   onmouseup   %Script;       #IMPLIED  -- a pointer button was released --
   onmouseover %Script;       #IMPLIED  -- a pointer was moved onto --
   onmousemove %Script;       #IMPLIED  -- a pointer was moved within --
   onmouseout  %Script;       #IMPLIED  -- a pointer was moved away --
   onkeypress  %Script;       #IMPLIED  -- a key was pressed and released --
   onkeydown   %Script;       #IMPLIED  -- a key was pressed down --
   onkeyup     %Script;       #IMPLIED  -- a key was released --"
   >
 <!-- Reserved Feature Switch -->
 <!ENTITY % HTML.Reserved "IGNORE">
 <!-- The following attributes are reserved for possible future use -->
 <![ %HTML.Reserved; [
 <!ENTITY % reserved
  "datasrc     %URI;          #IMPLIED  -- a single or tabular Data Source --
   datafld     CDATA          #IMPLIED  -- the property or column name --
   dataformatas (plaintext|html) plaintext -- text or html --"
   >
 ]]>
 <!ENTITY % reserved "">
 <!ENTITY % attrs "%coreattrs; %i18n; %events;">
 <!ENTITY % align "align (left|center|right|justify)  #IMPLIED"
                   -- default is left for ltr paragraphs, right for rtl --
   >
 <!--============= Text Markup ==============================-->
```

```
<!ENTITY % fontstyle
 "TT | I | B | U | S | STRIKE | BIG | SMALL">
<!ENTITY % phrase "EM | STRONG | DFN | CODE |
                   SAMP | KBD | VAR | CITE | ABBR | ACRONYM" >
<!ENTITY % special
   "A | IMG | APPLET | OBJECT | FONT | BASEFONT | BR | SCRIPT |
    MAP | Q | SUB | SUP | SPAN | BDO | IFRAME">
<!ENTITY % formctrl "INPUT | SELECT | TEXTAREA | LABEL | BUTTON">
<!-- %inline; covers inline or "text-level" elements -->
<!ENTITY % inline "#PCDATA | %fontstyle; | %phrase; | %special; |
%formctrl;">
<!ELEMENT (%fontstyle;|%phrase;) - - (%inline;)*>
<!ATTLIST (%fontstyle;|%phrase;)
  %attrs;                               -- %coreattrs, %i18n, %events --
  >
<!ELEMENT (SUB|SUP) - - (%inline;)*     -- subscript, superscript -->
<!ATTLIST (SUB|SUP)
  %attrs;                               -- %coreattrs, %i18n, %events --
  >
<!ELEMENT SPAN - - (%inline;)*        -- generic language/style container -->
<!ATTLIST SPAN
  %attrs;                               -- %coreattrs, %i18n, %events --
  %reserved;                           -- reserved for possible future use --
  >
<!ELEMENT BDO - - (%inline;)*           -- I18N BiDi over-ride -->
<!ATTLIST BDO
  %coreattrs;                           -- id, class, style, title --
  lang        %LanguageCode; #IMPLIED  -- language code --
  dir         (ltr|rtl)      #REQUIRED -- directionality --
  >
<!ELEMENT BASEFONT - O EMPTY            -- base font size -->
<!ATTLIST BASEFONT
  id          ID             #IMPLIED  -- document-wide unique id --
  size        CDATA          #REQUIRED -- base font size for FONT elements --
  color       %Color;        #IMPLIED  -- text color --
  face        CDATA          #IMPLIED  -- comma-separated list of font names --
  >
<!ELEMENT FONT - - (%inline;)*          -- local change to font -->
<!ATTLIST FONT
  %coreattrs;                           -- id, class, style, title --
  %i18n;                                -- lang, dir --
  size        CDATA          #IMPLIED  -- [+|-]nn e.g. size="+1", size="4" --
  color       %Color;        #IMPLIED  -- text color --
  face        CDATA          #IMPLIED  -- comma-separated list of font names --
  >
<!ELEMENT BR - O EMPTY                  -- forced line break -->
<!ATTLIST BR
  %coreattrs;                           -- id, class, style, title --
  clear       (left|all|right|none) none -- control of text flow --
  >
<!--============== HTML content models ====================-->
```

```
<!--
    HTML has two basic content models:
        %inline;     character level elements and text strings
        %block;      block-like elements e.g. paragraphs and lists
-->
<!ENTITY % block
     "P | %heading; | %list; | %preformatted; | DL | DIV | CENTER |
      NOSCRIPT | NOFRAMES | BLOCKQUOTE | FORM | ISINDEX | HR |
      TABLE | FIELDSET | ADDRESS">
<!ENTITY % flow "%block; | %inline;">
<!--================ Document Body =========================-->
<!ELEMENT BODY O O (%flow;)* +(INS|DEL) -- document body -->
<!ATTLIST BODY
  %attrs;                                -- %coreattrs, %i18n, %events --
  onload          %Script;   #IMPLIED  -- the document has been loaded --
  onunload        %Script;   #IMPLIED  -- the document has been removed --
  background      %URI;      #IMPLIED  -- texture tile for document
                                          background --
  %bodycolors;                         -- bgcolor, text, link, vlink, alink --
  >
<!ELEMENT ADDRESS - - ((%inline;)|P)*  -- information on author -->
<!ATTLIST ADDRESS
  %attrs;                               -- %coreattrs, %i18n, %events --
  >
<!ELEMENT DIV - - (%flow;)*           -- generic language/style container -->
<!ATTLIST DIV
  %attrs;                                -- %coreattrs, %i18n, %events --
  %align;                                -- align, text alignment --
  %reserved;                            -- reserved for possible future use --
  >
<!ELEMENT CENTER - - (%flow;)*         -- shorthand for DIV align=center -->
<!ATTLIST CENTER
  %attrs;                                -- %coreattrs, %i18n, %events --
  >
<!--================ The Anchor Element =====================-->
<!ENTITY % Shape "(rect|circle|poly|default)">
<!ENTITY % Coords "CDATA" -- comma-separated list of lengths -->
<!ELEMENT A - - (%inline;)* -(A)       -- anchor -->
<!ATTLIST A
  %attrs;                                -- %coreattrs, %i18n, %events --
  charset     %Charset;      #IMPLIED  -- char encoding of linked resource --
  type        %ContentType;  #IMPLIED  -- advisory content type --
  name        CDATA          #IMPLIED  -- named link end --
  href        %URI;          #IMPLIED  -- URI for linked resource --
  hreflang    %LanguageCode; #IMPLIED  -- language code --
  target      %FrameTarget;  #IMPLIED  -- render in this frame --
  rel         %LinkTypes;    #IMPLIED  -- forward link types --
  rev         %LinkTypes;    #IMPLIED  -- reverse link types --
  accesskey   %Character;    #IMPLIED  -- accessibility key character --
  shape       %Shape;        rect      -- for use with client-side image maps --
  coords      %Coords;       #IMPLIED  -- for use with client-side image maps --
```

```
  tabindex    NUMBER         #IMPLIED  -- position in tabbing order --
  onfocus     %Script;       #IMPLIED  -- the element got the focus --
  onblur      %Script;       #IMPLIED  -- the element lost the focus --
  >
<!--============== Client-side image maps =================-->
<!-- These can be placed in the same document or grouped in a
     separate document although this isn't yet widely supported -->
<!ELEMENT MAP - - ((%block;) | AREA)+ -- client-side image map -->
<!ATTLIST MAP
  %attrs;                              -- %coreattrs, %i18n, %events --
  name        CDATA          #REQUIRED -- for reference by usemap --
  >
<!ELEMENT AREA - O EMPTY               -- client-side image map area -->
<!ATTLIST AREA
  %attrs;                              -- %coreattrs, %i18n, %events --
  shape      %Shape;       rect     -- controls interpretation of coords --
  coords      %Coords;      #IMPLIED  -- comma-separated list of lengths --
  href        %URI;         #IMPLIED  -- URI for linked resource --
  target      %FrameTarget; #IMPLIED  -- render in this frame --
  nohref      (nohref)      #IMPLIED  -- this region has no action --
  alt         %Text;        #REQUIRED -- short description --
  tabindex    NUMBER         #IMPLIED  -- position in tabbing order --
  accesskey   %Character;    #IMPLIED  -- accessibility key character --
  onfocus     %Script;       #IMPLIED  -- the element got the focus --
  onblur      %Script;       #IMPLIED  -- the element lost the focus --
  >
<!--============== The LINK Element ======================-->
<!--
  Relationship values can be used in principle:
   a) for document specific toolbars/menus when used
      with the LINK element in document head e.g.
        start, contents, previous, next, index, end, help
   b) to link to a separate style sheet (rel=stylesheet)
   c) to make a link to a script (rel=script)
   d) by stylesheets to control how collections of
      html nodes are rendered into printed documents
   e) to make a link to a printable version of this document
      e.g. a postscript or pdf version (rel=alternate media=print)
-->
<!ELEMENT LINK - O EMPTY               -- a media-independent link -->
<!ATTLIST LINK
  %attrs;                              -- %coreattrs, %i18n, %events --
  charset     %Charset;      #IMPLIED  -- char encoding of linked resource --
  href        %URI;          #IMPLIED  -- URI for linked resource --
  hreflang    %LanguageCode; #IMPLIED  -- language code --
  type        %ContentType;  #IMPLIED  -- advisory content type --
  rel         %LinkTypes;    #IMPLIED  -- forward link types --
  rev         %LinkTypes;    #IMPLIED  -- reverse link types --
  media       %MediaDesc;    #IMPLIED  -- for rendering on these media --
  target      %FrameTarget;  #IMPLIED  -- render in this frame --
  >
```

```
<!--================ Images ====================================-->
<!-- Length defined in strict DTD for cellpadding/cellspacing -->
<!ENTITY % Length "CDATA" -- nn for pixels or nn% for percentage length -->
<!ENTITY % MultiLength "CDATA" -- pixel, percentage, or relative -->
<![ %HTML.Frameset; [
<!ENTITY % MultiLengths "CDATA" -- comma-separated list of MultiLength -->
]]>
<!ENTITY % Pixels "CDATA" -- integer representing length in pixels -->
<!ENTITY % IAlign "(top|middle|bottom|left|right)" -- center? -->
<!-- To avoid problems with text-only UAs as well as
   to make image content understandable and navigable
   to users of non-visual UAs, you need to provide
   a description with ALT, and avoid server-side image maps -->
<!ELEMENT IMG - O EMPTY                -- Embedded image -->
<!ATTLIST IMG
  %attrs;                              -- %coreattrs, %i18n, %events --
  src         %URI;          #REQUIRED -- URI of image to embed --
  alt         %Text;         #REQUIRED -- short description --
  longdesc    %URI;          #IMPLIED  -- link to long description
                                          (complements alt) --
  name        CDATA          #IMPLIED  -- name of image for scripting --
  height      %Length;       #IMPLIED  -- override height --
  width       %Length;       #IMPLIED  -- override width --
  usemap      %URI;          #IMPLIED  -- use client-side image map --
  ismap       (ismap)        #IMPLIED  -- use server-side image map --
  align      %IAlign;       #IMPLIED  -- vertical or horizontal alignment --
  border      %Pixels;       #IMPLIED  -- link border width --
  hspace      %Pixels;       #IMPLIED  -- horizontal gutter --
  vspace      %Pixels;       #IMPLIED  -- vertical gutter --
  >
<!-- USEMAP points to a MAP element which may be in this document
  or an external document, although the latter is not widely supported -->
<!--================ OBJECT ====================================-->
<!--
  OBJECT is used to embed objects as part of HTML pages
  PARAM elements should precede other content. SGML mixed content
  model technicality precludes specifying this formally ...
-->
<!ELEMENT OBJECT - - (PARAM | %flow;)*
 -- generic embedded object -->
<!ATTLIST OBJECT
  %attrs;                              -- %coreattrs, %i18n, %events --
  declare     (declare)    #IMPLIED  -- declare but don't instantiate flag --
  classid     %URI;       #IMPLIED  -- identifies an implementation --
  codebase    %URI;        #IMPLIED  -- base URI for classid, data, archive--
  data        %URI;          #IMPLIED  -- reference to object's data --
  type        %ContentType;  #IMPLIED  -- content type for data --
  codetype    %ContentType;  #IMPLIED  -- content type for code --
  archive     CDATA         #IMPLIED  -- space-separated list of URIs --
  standby     %Text;         #IMPLIED  -- message to show while loading --
  height      %Length;     #IMPLIED  -- override height --
```

```
  width       %Length;       #IMPLIED  -- override width --
  usemap      %URI;          #IMPLIED  -- use client-side image map --
  name        CDATA          #IMPLIED  -- submit as part of form --
  tabindex    NUMBER         #IMPLIED  -- position in tabbing order --
  align       %IAlign;       #IMPLIED  -- vertical or horizontal alignment --
  border      %Pixels;       #IMPLIED  -- link border width --
  hspace      %Pixels;       #IMPLIED  -- horizontal gutter --
  vspace      %Pixels;       #IMPLIED  -- vertical gutter --
  %reserved;                           -- reserved for possible future use --
  >
<!ELEMENT PARAM - O EMPTY              -- named property value -->
<!ATTLIST PARAM
  id          ID             #IMPLIED  -- document-wide unique id --
  name        CDATA          #REQUIRED -- property name --
  value       CDATA          #IMPLIED  -- property value --
  valuetype   (DATA|REF|OBJECT) DATA   -- How to interpret value --
  type        %ContentType;  #IMPLIED  -- content type for value
                                          when valuetype=ref --
  >
<!--================ Java APPLET ===========================-->
<!--
  One of code or object attributes must be present.
  Place PARAM elements before other content.
-->
<!ELEMENT APPLET - - (PARAM | %flow;)* -- Java applet -->
<!ATTLIST APPLET
  %coreattrs;                          -- id, class, style, title --
  codebase    %URI;          #IMPLIED  -- optional base URI for applet --
  archive     CDATA          #IMPLIED  -- comma-separated archive list --
  code        CDATA          #IMPLIED  -- applet class file --
  object      CDATA          #IMPLIED  -- serialized applet file --
  alt         %Text;         #IMPLIED  -- short description --
  name        CDATA         #IMPLIED  -- allows applets to find each other --
  width       %Length;       #REQUIRED -- initial width --
  height      %Length;       #REQUIRED -- initial height --
  align       %IAlign;       #IMPLIED  -- vertical or horizontal alignment --
  hspace      %Pixels;       #IMPLIED  -- horizontal gutter --
  vspace      %Pixels;       #IMPLIED  -- vertical gutter --
  >
<!--================ Horizontal Rule ======================-->
<!ELEMENT HR - O EMPTY -- horizontal rule -->
<!ATTLIST HR
  %attrs;                              -- %coreattrs, %i18n, %events --
  align       (left|center|right) #IMPLIED
  noshade     (noshade)      #IMPLIED
  size        %Pixels;       #IMPLIED
  width       %Length;       #IMPLIED
  >
<!--================ Paragraphs ===========================-->
<!ELEMENT P - O (%inline;)*            -- paragraph -->
<!ATTLIST P
```

```
  %attrs;                              -- %coreattrs, %i18n, %events --
  %align;                              -- align, text alignment --
  >
<!--================ Headings ===============================-->
<!--
  There are six levels of headings from H1 (the most important)
  to H6 (the least important).
-->
<!ELEMENT (%heading;)  - - (%inline;)* -- heading -->
<!ATTLIST (%heading;)
  %attrs;                              -- %coreattrs, %i18n, %events --
  %align;                              -- align, text alignment --
  >
<!--================ Preformatted Text ======================-->
<!-- excludes markup for images and changes in font size -->
<!ENTITY % pre.exclusion "
IMG|OBJECT|APPLET|BIG|SMALL|SUB|SUP|FONT|BASEFONT">
<!ELEMENT PRE - - (%inline;)* -(%pre.exclusion;) -- preformatted text -->
<!ATTLIST PRE
  %attrs;                              -- %coreattrs, %i18n, %events --
  width       NUMBER         #IMPLIED
  >
<!--================== Inline Quotes ========================-->
<!ELEMENT Q - - (%inline;)*            -- short inline quotation -->
<!ATTLIST Q
  %attrs;                              -- %coreattrs, %i18n, %events --
  cite        %URI;          #IMPLIED  -- URI for source document or msg --
  >
<!--================ Block-like Quotes ======================-->
<!ELEMENT BLOCKQUOTE - - (%flow;)*     -- long quotation -->
<!ATTLIST BLOCKQUOTE
  %attrs;                              -- %coreattrs, %i18n, %events --
  cite        %URI;          #IMPLIED  -- URI for source document or msg --
  >
<!--================ Inserted/Deleted Text ==================-->
<!-- INS/DEL are handled by inclusion on BODY -->
<!ELEMENT (INS|DEL) - - (%flow;)*      -- inserted text, deleted text -->
<!ATTLIST (INS|DEL)
  %attrs;                              -- %coreattrs, %i18n, %events --
  cite        %URI;          #IMPLIED  -- info on reason for change --
  datetime    %Datetime;     #IMPLIED  -- date and time of change --
  >
<!--================ Lists ==================================-->
<!-- definition lists - DT for term, DD for its definition -->
<!ELEMENT DL - - (DT|DD)+              -- definition list -->
<!ATTLIST DL
  %attrs;                              -- %coreattrs, %i18n, %events --
  compact     (compact)      #IMPLIED  -- reduced interitem spacing --
  >
<!ELEMENT DT - O (%inline;)*           -- definition term -->
<!ELEMENT DD - O (%flow;)*             -- definition description -->
```

```
<!ATTLIST (DT|DD)
  %attrs;                              -- %coreattrs, %i18n, %events --
  >
<!-- Ordered lists (OL) Numbering style
    1   arabic numbers     1, 2, 3, ...
    a   lower alpha        a, b, c, ...
    A   upper alpha        A, B, C, ...
    i   lower roman        i, ii, iii, ...
    I   upper roman        I, II, III, ...
    The style is applied to the sequence number which by default
    is reset to 1 for the first list item in an ordered list.
    This can't be expressed directly in SGML due to case folding.
-->
<!ENTITY % OLStyle "CDATA"      -- constrained to: "(1|a|A|i|I)" -->
<!ELEMENT OL - - (LI)+                 -- ordered list -->
<!ATTLIST OL
  %attrs;                              -- %coreattrs, %i18n, %events --
  type        %OLStyle;      #IMPLIED  -- numbering style --
  compact     (compact)      #IMPLIED  -- reduced interitem spacing --
  start       NUMBER         #IMPLIED  -- starting sequence number --
  >
<!-- Unordered Lists (UL) bullet styles -->
<!ENTITY % ULStyle "(disc|square|circle)">
<!ELEMENT UL - - (LI)+                 -- unordered list -->
<!ATTLIST UL
  %attrs;                              -- %coreattrs, %i18n, %events --
  type        %ULStyle;      #IMPLIED  -- bullet style --
  compact     (compact)      #IMPLIED  -- reduced interitem spacing --
  >
<!ELEMENT (DIR|MENU) - - (LI)+ -(%block;) -- directory list, menu list -->
<!ATTLIST DIR
  %attrs;                              -- %coreattrs, %i18n, %events --
  compact     (compact)      #IMPLIED -- reduced interitem spacing --
  >
<!ATTLIST MENU
  %attrs;                              -- %coreattrs, %i18n, %events --
  compact     (compact)      #IMPLIED -- reduced interitem spacing --
  >
<!ENTITY % LIStyle "CDATA" -- constrained to: "(%ULStyle;|%OLStyle;)" -->
<!ELEMENT LI - O (%flow;)*             -- list item -->
<!ATTLIST LI
  %attrs;                              -- %coreattrs, %i18n, %events --
  type        %LIStyle;      #IMPLIED  -- list item style --
  value       NUMBER         #IMPLIED  -- reset sequence number --
  >
<!--============= Forms ===================================-->
<!ELEMENT FORM - - (%flow;)* -(FORM)   -- interactive form -->
<!ATTLIST FORM
  %attrs;                              -- %coreattrs, %i18n, %events --
  action      %URI;          #REQUIRED -- server-side form handler --
  method     (GET|POST)     GET       -- HTTP method used to submit the form--
```

```
  enctype     %ContentType;  "application/x-www-form-urlencoded"
  accept     %ContentTypes; #IMPLIED  -- list of MIME types for file upload --
  name        CDATA          #IMPLIED  -- name of form for scripting --
  onsubmit    %Script;       #IMPLIED  -- the form was submitted --
  onreset     %Script;       #IMPLIED  -- the form was reset --
  target      %FrameTarget;  #IMPLIED  -- render in this frame --
  accept-charset %Charsets;  #IMPLIED  -- list of supported charsets --
  >
<!-- Each label must not contain more than ONE field -->
<!ELEMENT LABEL - - (%inline;)* -(LABEL) -- form field label text -->
<!ATTLIST LABEL
  %attrs;                              -- %coreattrs, %i18n, %events --
  for         IDREF          #IMPLIED  -- matches field ID value --
  accesskey   %Character;    #IMPLIED  -- accessibility key character --
  onfocus     %Script;       #IMPLIED  -- the element got the focus --
  onblur      %Script;       #IMPLIED  -- the element lost the focus --
  >
<!ENTITY % InputType
  "(TEXT | PASSWORD | CHECKBOX |
    RADIO | SUBMIT | RESET |
    FILE | HIDDEN | IMAGE | BUTTON)"
   >
<!-- attribute name required for all but submit and reset -->
<!ELEMENT INPUT - O EMPTY              -- form control -->
<!ATTLIST INPUT
  %attrs;                              -- %coreattrs, %i18n, %events --
  type        %InputType;    TEXT      -- what kind of widget is needed --
  name        CDATA          #IMPLIED  -- submit as part of form --
  value       CDATA          #IMPLIED  -- specify for radio buttons and
                                          checkboxes --
  checked     (checked)      #IMPLIED  -- for radio buttons and check boxes --
  disabled    (disabled)     #IMPLIED  -- unavailable in this context --
  readonly    (readonly)     #IMPLIED  -- for text and passwd --
  size        CDATA          #IMPLIED  -- specific to each type of field --
  maxlength   NUMBER         #IMPLIED  -- max chars for text fields --
  src         %URI;          #IMPLIED  -- for fields with images --
  alt         CDATA          #IMPLIED  -- short description --
  usemap      %URI;          #IMPLIED  -- use client-side image map --
  ismap       (ismap)        #IMPLIED  -- use server-side image map --
  tabindex    NUMBER         #IMPLIED  -- position in tabbing order --
  accesskey   %Character;    #IMPLIED  -- accessibility key character --
  onfocus     %Script;       #IMPLIED  -- the element got the focus --
  onblur      %Script;       #IMPLIED  -- the element lost the focus --
  onselect    %Script;       #IMPLIED  -- some text was selected --
  onchange    %Script;       #IMPLIED  -- the element value was changed --
  accept     %ContentTypes; #IMPLIED  -- list of MIME types for file upload --
  align       %IAlign;       #IMPLIED  -- vertical or horizontal alignment --
  %reserved;                           -- reserved for possible future use --
  >
<!ELEMENT SELECT - - (OPTGROUP|OPTION)+ -- option selector -->
<!ATTLIST SELECT
```

```
  %attrs;                                -- %coreattrs, %i18n, %events --
  name        CDATA          #IMPLIED  -- field name --
  size        NUMBER         #IMPLIED  -- rows visible --
  multiple    (multiple)     #IMPLIED  -- default is single selection --
  disabled    (disabled)     #IMPLIED  -- unavailable in this context --
  tabindex    NUMBER         #IMPLIED  -- position in tabbing order --
  onfocus     %Script;       #IMPLIED  -- the element got the focus --
  onblur      %Script;       #IMPLIED  -- the element lost the focus --
  onchange    %Script;       #IMPLIED  -- the element value was changed --
  %reserved;                           -- reserved for possible future use --
  >
<!ELEMENT OPTGROUP - - (OPTION)+ -- option group -->
<!ATTLIST OPTGROUP
  %attrs;                                -- %coreattrs, %i18n, %events --
  disabled    (disabled)     #IMPLIED  -- unavailable in this context --
  label       %Text;         #REQUIRED -- for use in hierarchical menus --
  >
<!ELEMENT OPTION - O (#PCDATA)         -- selectable choice -->
<!ATTLIST OPTION
  %attrs;                                -- %coreattrs, %i18n, %events --
  selected    (selected)     #IMPLIED
  disabled    (disabled)     #IMPLIED  -- unavailable in this context --
  label       %Text;         #IMPLIED  -- for use in hierarchical menus --
  value       CDATA          #IMPLIED  -- defaults to element content --
  >
<!ELEMENT TEXTAREA - - (#PCDATA)       -- multi-line text field -->
<!ATTLIST TEXTAREA
  %attrs;                                -- %coreattrs, %i18n, %events --
  name        CDATA          #IMPLIED
  rows        NUMBER         #REQUIRED
  cols        NUMBER         #REQUIRED
  disabled    (disabled)     #IMPLIED  -- unavailable in this context --
  readonly    (readonly)     #IMPLIED
  tabindex    NUMBER         #IMPLIED  -- position in tabbing order --
  accesskey   %Character;    #IMPLIED  -- accessibility key character --
  onfocus     %Script;       #IMPLIED  -- the element got the focus --
  onblur      %Script;       #IMPLIED  -- the element lost the focus --
  onselect    %Script;       #IMPLIED  -- some text was selected --
  onchange    %Script;       #IMPLIED  -- the element value was changed --
  %reserved;                           -- reserved for possible future use --
  >
<!--
  #PCDATA is to solve the mixed content problem,
  per specification only whitespace is allowed there!
 -->
<!ELEMENT FIELDSET - - (#PCDATA,LEGEND,(%flow;)*) -- form control group -->
<!ATTLIST FIELDSET
  %attrs;                                -- %coreattrs, %i18n, %events --
  >
<!ELEMENT LEGEND - - (%inline;)*       -- fieldset legend -->
<!ENTITY % LAlign "(top|bottom|left|right)">
```

```
<!ATTLIST LEGEND
  %attrs;                              -- %coreattrs, %i18n, %events --
  accesskey   %Character;    #IMPLIED  -- accessibility key character --
  align       %LAlign;       #IMPLIED  -- relative to fieldset --
  >
<!ELEMENT BUTTON - -
     (%flow;)* -(A|%formctrl;|FORM|ISINDEX|FIELDSET|IFRAME)
     -- push button -->
<!ATTLIST BUTTON
  %attrs;                              -- %coreattrs, %i18n, %events --
  name        CDATA          #IMPLIED
  value       CDATA          #IMPLIED  -- sent to server when submitted --
  type        (button|submit|reset) submit -- for use as form button --
  disabled    (disabled)     #IMPLIED  -- unavailable in this context --
  tabindex    NUMBER         #IMPLIED  -- position in tabbing order --
  accesskey   %Character;    #IMPLIED  -- accessibility key character --
  onfocus     %Script;       #IMPLIED  -- the element got the focus --
  onblur      %Script;       #IMPLIED  -- the element lost the focus --
  %reserved;                           -- reserved for possible future use --
  >
<!--================== Tables ============================-->
<!-- IETF HTML table standard, see [RFC1942] -->
<!--
 The BORDER attribute sets the thickness of the frame around the
 table. The default units are screen pixels.
 The FRAME attribute specifies which parts of the frame around
 the table should be rendered. The values are not the same as
 CALS to avoid a name clash with the VALIGN attribute.
 The value "border" is included for backwards compatibility with
 <TABLE BORDER> which yields frame=border and border=implied
 For <TABLE BORDER=1> you get border=1 and frame=implied. In this
 case, it is appropriate to treat this as frame=border for backwards
 compatibility with deployed browsers.
-->
<!ENTITY % TFrame "(void|above|below|hsides|lhs|rhs|vsides|box|border)">
<!--
 The RULES attribute defines which rules to draw between cells:
 If RULES is absent then assume:
     "none" if BORDER is absent or BORDER=0 otherwise "all"
-->
<!ENTITY % TRules "(none | groups | rows | cols | all)">

<!-- horizontal placement of table relative to document -->
<!ENTITY % TAlign "(left|center|right)">
<!-- horizontal alignment attributes for cell contents -->
<!ENTITY % cellhalign
  "align      (left|center|right|justify|char) #IMPLIED
   char       %Character;    #IMPLIED  -- alignment char, e.g. char=':' --
   charoff    %Length;       #IMPLIED  -- offset for alignment char --"
  >
<!-- vertical alignment attributes for cell contents -->
```

```
<!ENTITY % cellvalign
  "valign     (top|middle|bottom|baseline) #IMPLIED"
  >
<!ELEMENT TABLE - -
     (CAPTION?, (COL*|COLGROUP*), THEAD?, TFOOT?, TBODY+)>
<!ELEMENT CAPTION  - - (%inline;)*     -- table caption -->
<!ELEMENT THEAD    - O (TR)+           -- table header -->
<!ELEMENT TFOOT    - O (TR)+           -- table footer -->
<!ELEMENT TBODY    O O (TR)+           -- table body -->
<!ELEMENT COLGROUP - O (COL)*          -- table column group -->
<!ELEMENT COL      - O EMPTY           -- table column -->
<!ELEMENT TR       - O (TH|TD)+        -- table row -->
<!ELEMENT (TH|TD)  - O (%flow;)*     -- table header cell, table data cell-->
<!ATTLIST TABLE                        -- table element --
  %attrs;                              -- %coreattrs, %i18n, %events --
  summary     %Text;         #IMPLIED  -- purpose/structure for speech
                                          output--
  width       %Length;       #IMPLIED  -- table width --
  border      %Pixels;       #IMPLIED  -- controls frame width around table --
  frame       %TFrame;       #IMPLIED  -- which parts of frame to render --
  rules       %TRules;       #IMPLIED  -- rulings between rows and cols --
  cellspacing %Length;       #IMPLIED  -- spacing between cells --
  cellpadding %Length;       #IMPLIED  -- spacing within cells --
  align       %TAlign;       #IMPLIED  -- table position relative to window --
  bgcolor     %Color;        #IMPLIED  -- background color for cells --
  %reserved;                           -- reserved for possible future use --
  datapagesize CDATA         #IMPLIED  -- reserved for possible future use --
  >
<!ENTITY % CAlign "(top|bottom|left|right)">
<!ATTLIST CAPTION
  %attrs;                              -- %coreattrs, %i18n, %events --
  align       %CAlign;       #IMPLIED  -- relative to table --
  >
<!--
COLGROUP groups a set of COL elements. It allows you to group
several semantically related columns together.
-->
<!ATTLIST COLGROUP
  %attrs;                              -- %coreattrs, %i18n, %events --
  span        NUMBER         1        -- default number of columns in group --
  width       %MultiLength;  #IMPLIED  -- default width for enclosed COLs --
  %cellhalign;                         -- horizontal alignment in cells --
  %cellvalign;                         -- vertical alignment in cells --
  >
<!--
 COL elements define the alignment properties for cells in
 one or more columns.
 The WIDTH attribute specifies the width of the columns, e.g.
     width=64        width in screen pixels
     width=0.5*      relative width of 0.5
 The SPAN attribute causes the attributes of one
```

```
 COL element to apply to more than one column.
-->
<!ATTLIST COL                           -- column groups and properties --
  %attrs;                               -- %coreattrs, %i18n, %events --
  span        NUMBER         1         -- COL attributes affect N columns --
  width       %MultiLength;  #IMPLIED  -- column width specification --
  %cellhalign;                          -- horizontal alignment in cells --
  %cellvalign;                          -- vertical alignment in cells --
  >
<!--
    Use THEAD to duplicate headers when breaking table
    across page boundaries, or for static headers when
    TBODY sections are rendered in scrolling panel.
    Use TFOOT to duplicate footers when breaking table
    across page boundaries, or for static footers when
    TBODY sections are rendered in scrolling panel.
    Use multiple TBODY sections when rules are needed
    between groups of table rows.
-->
<!ATTLIST (THEAD|TBODY|TFOOT)           -- table section --
  %attrs;                               -- %coreattrs, %i18n, %events --
  %cellhalign;                          -- horizontal alignment in cells --
  %cellvalign;                          -- vertical alignment in cells --
  >
<!ATTLIST TR                            -- table row --
  %attrs;                               -- %coreattrs, %i18n, %events --
  %cellhalign;                          -- horizontal alignment in cells --
  %cellvalign;                          -- vertical alignment in cells --
  bgcolor     %Color;        #IMPLIED  -- background color for row --
  >
<!-- Scope is simpler than headers attribute for common tables -->
<!ENTITY % Scope "(row|col|rowgroup|colgroup)">
<!-- TH is for headers, TD for data, but for cells acting as both use TD -->
<!ATTLIST (TH|TD)                       -- header or data cell --
  %attrs;                               -- %coreattrs, %i18n, %events --
  abbr        %Text;         #IMPLIED  -- abbreviation for header cell --
  axis        CDATA          #IMPLIED  -- comma-separated list of related
                                          headers--
  headers     IDREFS         #IMPLIED  -- list of id's for header cells --
  scope       %Scope;        #IMPLIED  -- scope covered by header cells --
  rowspan     NUMBER         1         -- number of rows spanned by cell --
  colspan     NUMBER         1         -- number of cols spanned by cell --
  %cellhalign;                          -- horizontal alignment in cells --
  %cellvalign;                          -- vertical alignment in cells --
  nowrap      (nowrap)       #IMPLIED  -- suppress word wrap --
  bgcolor     %Color;        #IMPLIED  -- cell background color --
  width       %Length;       #IMPLIED  -- width for cell --
  height      %Length;       #IMPLIED  -- height for cell --
  >
<!--============== Document Frames ========================-->
<!--
```

```
   The content model for HTML documents depends on whether the HEAD is
   followed by a FRAMESET or BODY element. The widespread omission of
   the BODY start tag makes it impractical to define the content model
   without the use of a marked section.
 -->
 <![ %HTML.Frameset; [
 <!ELEMENT FRAMESET - - ((FRAMESET|FRAME)+ & NOFRAMES?) -- window
subdivision-->
 <!ATTLIST FRAMESET
   %coreattrs;                          -- id, class, style, title --
   rows        %MultiLengths; #IMPLIED  -- list of lengths,
                                           default: 100% (1 row) --
   cols        %MultiLengths; #IMPLIED  -- list of lengths,
                                           default: 100% (1 col) --
   onload      %Script;       #IMPLIED  -- all the frames have been loaded  --
   onunload    %Script;       #IMPLIED  -- all the frames have been removed --
   >
 ]]>
 <![ %HTML.Frameset; [
 <!-- reserved frame names start with "_" otherwise starts with letter -->
 <!ELEMENT FRAME - O EMPTY              -- subwindow -->
 <!ATTLIST FRAME
   %coreattrs;                          -- id, class, style, title --
   longdesc    %URI;          #IMPLIED  -- link to long description
                                           (complements title) --
   name        CDATA          #IMPLIED  -- name of frame for targetting --
   src         %URI;          #IMPLIED  -- source of frame content --
   frameborder (1|0)          1         -- request frame borders? --
   marginwidth %Pixels;       #IMPLIED  -- margin widths in pixels --
   marginheight %Pixels;      #IMPLIED  -- margin height in pixels --
   noresize    (noresize)     #IMPLIED  -- allow users to resize frames? --
   scrolling   (yes|no|auto)  auto      -- scrollbar or none --
   >
 ]]>
 <!ELEMENT IFRAME - - (%flow;)*         -- inline subwindow -->
 <!ATTLIST IFRAME
   %coreattrs;                          -- id, class, style, title --
   longdesc    %URI;          #IMPLIED  -- link to long description
                                           (complements title) --
   name        CDATA          #IMPLIED  -- name of frame for targetting --
   src         %URI;          #IMPLIED  -- source of frame content --
   frameborder (1|0)          1         -- request frame borders? --
   marginwidth %Pixels;       #IMPLIED  -- margin widths in pixels --
   marginheight %Pixels;      #IMPLIED  -- margin height in pixels --
   scrolling   (yes|no|auto)  auto      -- scrollbar or none --
   align       %IAlign;       #IMPLIED  -- vertical or horizontal alignment --
   height      %Length;       #IMPLIED  -- frame height --
   width       %Length;       #IMPLIED  -- frame width --
   >
 <![ %HTML.Frameset; [
 <!ENTITY % noframes.content "(BODY) -(NOFRAMES)">
```

```
]]>
<!ENTITY % noframes.content "(%flow;)*">
<!ELEMENT NOFRAMES - - %noframes.content;
 -- alternate content container for non frame-based rendering -->
<!ATTLIST NOFRAMES
  %attrs;                              -- %coreattrs, %i18n, %events --
  >
<!--============= Document Head ============================-->
<!-- %head.misc; defined earlier on as "SCRIPT|STYLE|META|LINK|OBJECT" -->
<!ENTITY % head.content "TITLE & ISINDEX? & BASE?">
<!ELEMENT HEAD O O (%head.content;) +(%head.misc;) -- document head -->
<!ATTLIST HEAD
  %i18n;                               -- lang, dir --
  profile     %URI;          #IMPLIED  -- named dictionary of meta info --
  >
<!-- The TITLE element is not considered part of the flow of text.
       It should be displayed, for example as the page header or
       window title. Exactly one title is required per document.
    -->
<!ELEMENT TITLE - - (#PCDATA) -(%head.misc;) -- document title -->
<!ATTLIST TITLE %i18n>
<!ELEMENT ISINDEX - O EMPTY            -- single line prompt -->
<!ATTLIST ISINDEX
  %coreattrs;                          -- id, class, style, title --
  %i18n;                               -- lang, dir --
  prompt      %Text;         #IMPLIED  -- prompt message -->
<!ELEMENT BASE - O EMPTY               -- document base URI -->
<!ATTLIST BASE
  href        %URI;          #IMPLIED  -- URI that acts as base URI --
  target      %FrameTarget;  #IMPLIED  -- render in this frame --
  >
<!ELEMENT META - O EMPTY               -- generic metainformation -->
<!ATTLIST META
  %i18n;                              -- lang, dir, for use with content --
  http-equiv  NAME           #IMPLIED  -- HTTP response header name  --
  name        NAME           #IMPLIED  -- metainformation name --
  content     CDATA          #REQUIRED -- associated information --
  scheme      CDATA          #IMPLIED  -- select form of content --
  >
<!ELEMENT STYLE - - %StyleSheet        -- style info -->
<!ATTLIST STYLE
  %i18n;                               -- lang, dir, for use with title --
  type        %ContentType;  #REQUIRED -- content type of style language --
  media       %MediaDesc;    #IMPLIED  -- designed for use with these media --
  title       %Text;         #IMPLIED  -- advisory title --
  >
<!ELEMENT SCRIPT - - %Script;          -- script statements -->
<!ATTLIST SCRIPT
  charset     %Charset;      #IMPLIED  -- char encoding of linked resource --
  type        %ContentType;  #REQUIRED -- content type of script language --
  language    CDATA          #IMPLIED  -- predefined script language name --
```

```
  src         %URI;          #IMPLIED  -- URI for an external script --
  defer       (defer)        #IMPLIED  -- UA may defer execution of script --
  event       CDATA          #IMPLIED  -- reserved for possible future use --
  for         %URI;          #IMPLIED  -- reserved for possible future use --
  >
<!ELEMENT NOSCRIPT - - (%flow;)*
  -- alternate content container for non script-based rendering -->
<!ATTLIST NOSCRIPT
  %attrs;                              -- %coreattrs, %i18n, %events --
  >
<!--============= Document Structure =======================-->
<!ENTITY % version "version CDATA #FIXED '%HTML.Version;'">
<![ %HTML.Frameset; [
<!ENTITY % html.content "HEAD, FRAMESET">
]]>
<!ENTITY % html.content "HEAD, BODY">
<!ELEMENT HTML O O (%html.content;)    -- document root element -->
<!ATTLIST HTML
  %i18n;                               -- lang, dir --
  %version;
  >
```

E

XHTML 1.0 DTD

Стандарт XHTML 1.0 определяется формально в виде трех XML-определений типа документа (DTD, Document Type Definition) – строгого (Strict) DTD, переходного (Transitional) DTD и фреймового (Frameset) DTD. Эти DTD соответствуют одноименным HTML 4.01 DTD, определяя те же самые теги и атрибуты с применением XML вместо SGML.

Строгое DTD определяет только те элементы, которые не объявлены нежелательными в стандарте HTML 4.01. В идеале все должны писать документы, соответствующие строгому DTD. Переходное DTD включает в себя все эти нежелательные элементы и более точно отражает современное употребление HTML, в котором все еще широко используются устаревшие элементы. Фреймовое DTD совпадает с переходным за тем исключением, что тело документа `<body>` заменяется на тег `<frameset>`.

Поскольку переходное HTML DTD – это то DTD, на котором основана эта книга, мы включаем сюда соответствующее XHTML DTD. Отметьте, что мы перепечатываем DTD дословно и не пытаемся включить в него расширения. Там, где наше описание и DTD расходятся, считайте правильным DTD.

```
<!--
   Extensible HTML version 1.0 Transitional DTD
   This is the same as HTML 4.0 Transitional except for
   changes due to the differences between XML and SGML.
   Namespace = http://www.w3.org/1999/xhtml
   For further information, see: http://www.w3.org/TR/xhtml1
   Copyright (c) 1998-2000 W3C (MIT, INRIA, Keio),
   All Rights Reserved.
   This DTD module is identified by the PUBLIC and SYSTEM identifiers:
   PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   SYSTEM "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
```

```
   $Revision: 1.18 $
   $Date: 2006/09/27 15:34:26 $
-->
<!--================ Character mnemonic entities =========================-->
<!ENTITY % HTMLlat1 PUBLIC
   "-//W3C//ENTITIES Latin 1 for XHTML//EN"
   "xhtml-lat1.ent">
%HTMLlat1;
<!ENTITY % HTMLsymbol PUBLIC
   "-//W3C//ENTITIES Symbols for XHTML//EN"
   "xhtml-symbol.ent">
%HTMLsymbol;
<!ENTITY % HTMLspecial PUBLIC
   "-//W3C//ENTITIES Special for XHTML//EN"
   "xhtml-special.ent">
%HTMLspecial;
<!--================== Imported Names ====================================-->
<!ENTITY % ContentType "CDATA">
    <!-- media type, as per [RFC2045] -->
<!ENTITY % ContentTypes "CDATA">
    <!-- comma-separated list of media types, as per [RFC2045] -->
<!ENTITY % Charset "CDATA">
    <!-- a character encoding, as per [RFC2045] -->
<!ENTITY % Charsets "CDATA">
    <!-- a space separated list of character encodings, as per [RFC2045] -->
<!ENTITY % LanguageCode "NMTOKEN">
    <!-- a language code, as per [RFC1766] -->
<!ENTITY % Character "CDATA">
    <!-- a single character from [ISO10646] -->
<!ENTITY % Number "CDATA">
    <!-- one or more digits -->
<!ENTITY % LinkTypes "CDATA">
    <!-- space-separated list of link types -->
<!ENTITY % MediaDesc "CDATA">
    <!-- single or comma-separated list of media descriptors -->
<!ENTITY % URI "CDATA">
    <!-- a Uniform Resource Identifier, see [RFC2396] -->
<!ENTITY % UriList "CDATA">
    <!-- a space separated list of Uniform Resource Identifiers -->
<!ENTITY % Datetime "CDATA">
    <!-- date and time information. ISO date format -->
<!ENTITY % Script "CDATA">
    <!-- script expression -->
<!ENTITY % StyleSheet "CDATA">
    <!-- style sheet data -->
<!ENTITY % Text "CDATA">
    <!-- used for titles etc. -->
<!ENTITY % FrameTarget "NMTOKEN">
    <!-- render in this frame -->
<!ENTITY % Length "CDATA">
    <!-- nn for pixels or nn% for percentage length -->
```

```
<!ENTITY % MultiLength "CDATA">
    <!-- pixel, percentage, or relative -->
<!ENTITY % MultiLengths "CDATA">
    <!-- comma-separated list of MultiLength -->
<!ENTITY % Pixels "CDATA">
    <!-- integer representing length in pixels -->
<!-- these are used for image maps -->
<!ENTITY % Shape "(rect|circle|poly|default)">
<!ENTITY % Coords "CDATA">
    <!-- comma separated list of lengths -->
<!-- used for object, applet, img, input and iframe -->
<!ENTITY % ImgAlign "(top|middle|bottom|left|right)">
<!-- a color using sRGB: #RRGGBB as Hex values -->
<!ENTITY % Color "CDATA">
<!-- There are also 16 widely known color names with their sRGB values:
    Black  = #000000    Green  = #008000
    Silver = #C0C0C0    Lime   = #00FF00
    Gray   = #808080    Olive  = #808000
    White  = #FFFFFF    Yellow = #FFFF00
    Maroon = #800000    Navy   = #000080
    Red    = #FF0000    Blue   = #0000FF
    Purple = #800080    Teal   = #008080
    Fuchsia= #FF00FF    Aqua   = #00FFFF
-->
<!--===================== Generic Attributes ==============================-->
<!-- core attributes common to most elements
  id       document-wide unique id
  class    space separated list of classes
  style    associated style info
  title    advisory title/amplification
-->
<!ENTITY % coreattrs
 "id          ID             #IMPLIED
  class       CDATA          #IMPLIED
  style       %StyleSheet;   #IMPLIED
  title       %Text;         #IMPLIED"
  >
<!-- internationalization attributes
  lang        language code (backwards compatible)
  xml:lang    language code (as per XML 1.0 spec)
  dir         direction for weak/neutral text
-->
<!ENTITY % i18n
 "lang        %LanguageCode; #IMPLIED
  xml:lang    %LanguageCode; #IMPLIED
  dir         (ltr|rtl)      #IMPLIED"
  >
<!-- attributes for common UI events
  onclick     a pointer button was clicked
  ondblclick  a pointer button was double clicked
  onmousedown a pointer button was pressed down
```

```
  onmouseup   a pointer button was released
  onmousemove a pointer was moved onto the element
  onmouseout  a pointer was moved away from the element
  onkeypress  a key was pressed and released
  onkeydown   a key was pressed down
  onkeyup     a key was released
-->
<!ENTITY % events
 "onclick     %Script;       #IMPLIED
  ondblclick  %Script;       #IMPLIED
  onmousedown %Script;       #IMPLIED
  onmouseup   %Script;       #IMPLIED
  onmouseover %Script;       #IMPLIED
  onmousemove %Script;       #IMPLIED
  onmouseout  %Script;       #IMPLIED
  onkeypress  %Script;       #IMPLIED
  onkeydown   %Script;       #IMPLIED
  onkeyup     %Script;       #IMPLIED"
  >
<!-- attributes for elements that can get the focus
  accesskey   accessibility key character
  tabindex    position in tabbing order
  onfocus     the element got the focus
  onblur      the element lost the focus
-->
<!ENTITY % focus
 "accesskey   %Character;    #IMPLIED
  tabindex    %Number;       #IMPLIED
  onfocus     %Script;       #IMPLIED
  onblur      %Script;       #IMPLIED"
  >
<!ENTITY % attrs "%coreattrs; %i18n; %events;">
<!-- text alignment for p, div, h1-h6. The default is
     align="left" for ltr headings, "right" for rtl -->
<!ENTITY % TextAlign "align (left|center|right) #IMPLIED">
<!--================== Text Elements ====================================-->
<!ENTITY % special
   "br | span | bdo | object | applet | img | map | iframe">
<!ENTITY % fontstyle "tt | i | b | big | small | u
                      | s | strike |font | basefont">
<!ENTITY % phrase "em | strong | dfn | code | q | sub | sup |
                   samp | kbd | var | cite | abbr | acronym">
<!ENTITY % inline.forms "input | select | textarea | label | button">
<!-- these can occur at block or inline level -->
<!ENTITY % misc "ins | del | script | noscript">
<!ENTITY % inline "a | %special; | %fontstyle; | %phrase; | %inline.forms;">
<!-- %Inline; covers inline or "text-level" elements -->
<!ENTITY % Inline "(#PCDATA | %inline; | %misc;)*">
<!--================== Block level elements ============================-->
<!ENTITY % heading "h1|h2|h3|h4|h5|h6">
<!ENTITY % lists "ul | ol | dl | menu | dir">
```

```
<!ENTITY % blocktext "pre | hr | blockquote | address | center | noframes">
<!ENTITY % block
    "p | %heading; | div | %lists; | %blocktext; | isindex |fieldset | table">
<!ENTITY % Block "(%block; | form | %misc;)*">
<!-- %Flow; mixes Block and Inline and is used for list items etc. -->
<!ENTITY % Flow "(#PCDATA | %block; | form | %inline; | %misc;)*">
<!--================== Content models for exclusions ===================-->
<!-- a elements use %Inline; excluding a -->
<!ENTITY % a.content
   "(#PCDATA | %special; | %fontstyle; | %phrase; | %inline.forms; |
%misc;)*">
<!-- pre uses %Inline excluding img, object, applet, big, small,
     sub, sup, font, or basefont -->
<!ENTITY % pre.content
   "(#PCDATA | a | br | span | bdo | map | tt | i | b | u | s |
      %phrase; | %inline.forms;)*">
<!-- form uses %Flow; excluding form -->
<!ENTITY % form.content "(#PCDATA | %block; | %inline; | %misc;)*">
<!-- button uses %Flow; but excludes a, form, form controls, iframe -->
<!ENTITY % button.content
   "(#PCDATA | p | %heading; | div | %lists; | %blocktext; |
      table | br | span | bdo | object | applet | img | map |
      %fontstyle; | %phrase; | %misc;)*">
<!--================ Document Structure ================================-->
<!-- the namespace URI designates the document profile -->
<!ELEMENT html (head, body)>
<!ATTLIST html
  %i18n;
  xmlns       %URI;          #FIXED 'http://www.w3.org/1999/xhtml'
  >
<!--================ Document Head =====================================-->
<!ENTITY % head.misc "(script|style|meta|link|object|isindex)*">
<!-- content model is %head.misc; combined with a single
     title and an optional base element in any order -->
<!ELEMENT head (%head.misc;,
     ((title, %head.misc;, (base, %head.misc;)?) |
      (base, %head.misc;, (title, %head.misc;))))>
<!ATTLIST head
  %i18n;
  profile     %URI;          #IMPLIED
  >
<!-- The title element is not considered part of the flow of text.
       It should be displayed, for example as the page header or
       window title. Exactly one title is required per document.
    -->
<!ELEMENT title (#PCDATA)>
<!ATTLIST title %i18n;>
<!-- document base URI -->
<!ELEMENT base EMPTY>
<!ATTLIST base
  href        %URI;          #IMPLIED
```

```
  target      %FrameTarget;  #IMPLIED
  >
<!-- generic metainformation -->
<!ELEMENT meta EMPTY>
<!ATTLIST meta
  %i18n;
  http-equiv  CDATA          #IMPLIED
  name        CDATA          #IMPLIED
  content     CDATA          #REQUIRED
  scheme      CDATA          #IMPLIED
  >
<!--
  Relationship values can be used in principle:
   a) for document specific toolbars/menus when used
      with the link element in document head e.g.
      start, contents, previous, next, index, end, help
   b) to link to a separate style sheet (rel="stylesheet")
   c) to make a link to a script (rel="script")
   d) by stylesheets to control how collections of
      html nodes are rendered into printed documents
   e) to make a link to a printable version of this document
      e.g. a PostScript or PDF version (rel="alternate" media="print")
-->
<!ELEMENT link EMPTY>
<!ATTLIST link
  %attrs;
  charset     %Charset;      #IMPLIED
  href        %URI;          #IMPLIED
  hreflang    %LanguageCode; #IMPLIED
  type        %ContentType;  #IMPLIED
  rel         %LinkTypes;    #IMPLIED
  rev         %LinkTypes;    #IMPLIED
  media       %MediaDesc;    #IMPLIED
  target      %FrameTarget;  #IMPLIED
  >
<!-- style info, which may include CDATA sections -->
<!ELEMENT style (#PCDATA)>
<!ATTLIST style
  %i18n;
  type        %ContentType;  #REQUIRED
  media       %MediaDesc;    #IMPLIED
  title       %Text;         #IMPLIED
  xml:space   (preserve)     #FIXED 'preserve'
  >
<!-- script statements, which may include CDATA sections -->
<!ELEMENT script (#PCDATA)>
<!ATTLIST script
  charset     %Charset;      #IMPLIED
  type        %ContentType;  #REQUIRED
  language    CDATA          #IMPLIED
  src         %URI;          #IMPLIED
```

```
  defer       (defer)        #IMPLIED
  xml:space   (preserve)     #FIXED 'preserve'
  >
<!-- alternate content container for non script-based rendering -->
<!ELEMENT noscript %Flow;>
<!ATTLIST noscript
  %attrs;
  >
<!--======================== Frames ======================================-->
<!-- inline subwindow -->
<!ELEMENT iframe %Flow;>
<!ATTLIST iframe
  %coreattrs;
  longdesc    %URI;          #IMPLIED
  name        NMTOKEN        #IMPLIED
  src         %URI;          #IMPLIED
  frameborder (1|0)          "1"
  marginwidth %Pixels;       #IMPLIED
  marginheight %Pixels;      #IMPLIED
  scrolling   (yes|no|auto)  "auto"
  align       %ImgAlign;     #IMPLIED
  height      %Length;       #IMPLIED
  width       %Length;       #IMPLIED
  >
<!-- alternate content container for non frame-based rendering -->
<!ELEMENT noframes %Flow;>
<!ATTLIST noframes
  %attrs;
  >
<!--=================== Document Body ====================================-->
<!ELEMENT body %Flow;>
<!ATTLIST body
  %attrs;
  onload      %Script;       #IMPLIED
  onunload    %Script;       #IMPLIED
  background  %URI;          #IMPLIED
  bgcolor     %Color;        #IMPLIED
  text        %Color;        #IMPLIED
  link        %Color;        #IMPLIED
  vlink       %Color;        #IMPLIED
  alink       %Color;        #IMPLIED
  >
<!ELEMENT div %Flow;>  <!-- generic language/style container -->
<!ATTLIST div
  %attrs;
  %TextAlign;
  >
<!--=================== Paragraphs ======================================-->
<!ELEMENT p %Inline;>
<!ATTLIST p
  %attrs;
```

```
  %TextAlign;
  >
<!--==================== Headings ==========================================-->
<!--
  There are six levels of headings from h1 (the most important)
  to h6 (the least important).
-->
<!ELEMENT h1  %Inline;>
<!ATTLIST h1
  %attrs;
  %TextAlign;
  >
<!ELEMENT h2 %Inline;>
<!ATTLIST h2
  %attrs;
  %TextAlign;
  >
<!ELEMENT h3 %Inline;>
<!ATTLIST h3
  %attrs;
  %TextAlign;
  >
<!ELEMENT h4 %Inline;>
<!ATTLIST h4
  %attrs;
  %TextAlign;
  >
<!ELEMENT h5 %Inline;>
<!ATTLIST h5
  %attrs;
  %TextAlign;
  >
<!ELEMENT h6 %Inline;>
<!ATTLIST h6
  %attrs;
  %TextAlign;
  >
<!--==================== Lists =============================================-->
<!-- Unordered list bullet styles -->
<!ENTITY % ULStyle "(disc|square|circle)">
<!-- Unordered list -->
<!ELEMENT ul (li)+>
<!ATTLIST ul
  %attrs;
  type        %ULStyle;     #IMPLIED
  compact     (compact)     #IMPLIED
  >
<!-- Ordered list numbering style
    1   arabic numbers      1, 2, 3, ...
    a   lower alpha         a, b, c, ...
    A   upper alpha         A, B, C, ...
```

```
    i   lower roman         i, ii, iii, ...
    I   upper roman         I, II, III, ...
    The style is applied to the sequence number which by default
    is reset to 1 for the first list item in an ordered list.
-->
<!ENTITY % OLStyle "CDATA">
<!-- Ordered (numbered) list -->
<!ELEMENT ol (li)+>
<!ATTLIST ol
  %attrs;
  type        %OLStyle;      #IMPLIED
  compact     (compact)      #IMPLIED
  start       %Number;       #IMPLIED
  >
<!-- single column list (DEPRECATED) -->
<!ELEMENT menu (li)+>
<!ATTLIST menu
  %attrs;
  compact     (compact)     #IMPLIED
  >
<!-- multiple column list (DEPRECATED) -->
<!ELEMENT dir (li)+>
<!ATTLIST dir
  %attrs;
  compact     (compact)     #IMPLIED
  >
<!-- LIStyle is constrained to: "(%ULStyle;|%OLStyle;)" -->
<!ENTITY % LIStyle "CDATA">
<!-- list item -->
<!ELEMENT li %Flow;>
<!ATTLIST li
  %attrs;
  type        %LIStyle;      #IMPLIED
  value       %Number;       #IMPLIED
  >
<!-- definition lists - dt for term, dd for its definition -->
<!ELEMENT dl (dt|dd)+>
<!ATTLIST dl
  %attrs;
  compact     (compact)      #IMPLIED
  >
<!ELEMENT dt %Inline;>
<!ATTLIST dt
  %attrs;
  >
<!ELEMENT dd %Flow;>
<!ATTLIST dd
  %attrs;
  >
<!--================== Address ===========================================-->
<!-- information on author -->
```

```
<!ELEMENT address %Inline;>
<!ATTLIST address
  %attrs;
  >
<!--==================== Horizontal Rule ================================-->
<!ELEMENT hr EMPTY>
<!ATTLIST hr
  %attrs;
  align       (left|center|right) #IMPLIED
  noshade     (noshade)      #IMPLIED
  size        %Pixels;       #IMPLIED
  width       %Length;       #IMPLIED
  >
<!--==================== Preformatted Text ==============================-->
<!-- content is %Inline; excluding
       "img|object|applet|big|small|sub|sup|font|basefont" -->
<!ELEMENT pre %pre.content;>
<!ATTLIST pre
  %attrs;
  width       %Number;      #IMPLIED
  xml:space   (preserve)    #FIXED 'preserve'
  >
<!--==================== Block-like Quotes ==============================-->
<!ELEMENT blockquote %Flow;>
<!ATTLIST blockquote
  %attrs;
  cite        %URI;          #IMPLIED
  >
<!--==================== Text alignment =================================-->
<!-- center content -->
<!ELEMENT center %Flow;>
<!ATTLIST center
  %attrs;
  >
<!--==================== Inserted/Deleted Text ==========================-->
<!--
  ins/del are allowed in block and inline content, but its
  inappropriate to include block content within an ins element
  occurring in inline content.
-->
<!ELEMENT ins %Flow;>
<!ATTLIST ins
  %attrs;
  cite        %URI;          #IMPLIED
  datetime    %Datetime;     #IMPLIED
  >
<!ELEMENT del %Flow;>
<!ATTLIST del
  %attrs;
  cite        %URI;          #IMPLIED
  datetime    %Datetime;     #IMPLIED
```

```
  >
<!--================== The Anchor Element ==============================-->
<!-- content is %Inline; except that anchors shouldn't be nested -->
<!ELEMENT a %a.content;>
<!ATTLIST a
  %attrs;
  charset     %Charset;      #IMPLIED
  type        %ContentType;  #IMPLIED
  name        NMTOKEN        #IMPLIED
  href        %URI;          #IMPLIED
  hreflang    %LanguageCode; #IMPLIED
  rel         %LinkTypes;    #IMPLIED
  rev         %LinkTypes;    #IMPLIED
  accesskey   %Character;    #IMPLIED
  shape       %Shape;        "rect"
  coords      %Coords;       #IMPLIED
  tabindex    %Number;       #IMPLIED
  onfocus     %Script;       #IMPLIED
  onblur      %Script;       #IMPLIED
  target      %FrameTarget;  #IMPLIED
  >
<!--====================== Inline Elements ================================-->
<!ELEMENT span %Inline;> <!-- generic language/style container -->
<!ATTLIST span
  %attrs;
  >
<!ELEMENT bdo %Inline;>  <!-- I18N BiDi over-ride -->
<!ATTLIST bdo
  %coreattrs;
  %events;
  lang        %LanguageCode; #IMPLIED
  xml:lang    %LanguageCode; #IMPLIED
  dir         (ltr|rtl)      #REQUIRED
  >
<!ELEMENT br EMPTY>   <!-- forced line break -->
<!ATTLIST br
  %coreattrs;
  clear       (left|all|right|none) "none"
  >
<!ELEMENT em %Inline;>   <!-- emphasis -->
<!ATTLIST em %attrs;>
<!ELEMENT strong %Inline;>   <!-- strong emphasis -->
<!ATTLIST strong %attrs;>
<!ELEMENT dfn %Inline;>   <!-- definitional -->
<!ATTLIST dfn %attrs;>
<!ELEMENT code %Inline;>   <!-- program code -->
<!ATTLIST code %attrs;>
<!ELEMENT samp %Inline;>   <!-- sample -->
<!ATTLIST samp %attrs;>
<!ELEMENT kbd %Inline;>  <!-- something user would type -->
<!ATTLIST kbd %attrs;>
```

```
<!ELEMENT var %Inline;>   <!-- variable -->
<!ATTLIST var %attrs;>
<!ELEMENT cite %Inline;>   <!-- citation -->
<!ATTLIST cite %attrs;>
<!ELEMENT abbr %Inline;>   <!-- abbreviation -->
<!ATTLIST abbr %attrs;>
<!ELEMENT acronym %Inline;>   <!-- acronym -->
<!ATTLIST acronym %attrs;>
<!ELEMENT q %Inline;>   <!-- inlined quote -->
<!ATTLIST q
  %attrs;
  cite        %URI;          #IMPLIED
  >
<!ELEMENT sub %Inline;> <!-- subscript -->
<!ATTLIST sub %attrs;>
<!ELEMENT sup %Inline;> <!-- superscript -->
<!ATTLIST sup %attrs;>
<!ELEMENT tt %Inline;>   <!-- fixed pitch font -->
<!ATTLIST tt %attrs;>
<!ELEMENT i %Inline;>   <!-- italic font -->
<!ATTLIST i %attrs;>
<!ELEMENT b %Inline;>   <!-- bold font -->
<!ATTLIST b %attrs;>
<!ELEMENT big %Inline;>   <!-- bigger font -->
<!ATTLIST big %attrs;>
<!ELEMENT small %Inline;>   <!-- smaller font -->
<!ATTLIST small %attrs;>
<!ELEMENT u %Inline;>   <!-- underline -->
<!ATTLIST u %attrs;>
<!ELEMENT s %Inline;>   <!-- strike-through -->
<!ATTLIST s %attrs;>
<!ELEMENT strike %Inline;>   <!-- strike-through -->
<!ATTLIST strike %attrs;>
<!ELEMENT basefont EMPTY>  <!-- base font size -->
<!ATTLIST basefont
  id          ID             #IMPLIED
  size        CDATA          #REQUIRED
  color       %Color;        #IMPLIED
  face        CDATA          #IMPLIED
  >
<!ELEMENT font %Inline;> <!-- local change to font -->
<!ATTLIST font
  %coreattrs;
  %i18n;
  size        CDATA          #IMPLIED
  color       %Color;        #IMPLIED
  face        CDATA          #IMPLIED
  >
<!--===================== Object ======================================-->
<!--
  object is used to embed objects as part of HTML pages.
```

```
  param elements should precede other content. Parameters
  can also be expressed as attribute/value pairs on the
  object element itself when brevity is desired.
-->
<!ELEMENT object (#PCDATA | param | %block; | form | %inline; | %misc;)*>
<!ATTLIST object
  %attrs;
  declare     (declare)      #IMPLIED
  classid     %URI;          #IMPLIED
  codebase    %URI;          #IMPLIED
  data        %URI;          #IMPLIED
  type        %ContentType;  #IMPLIED
  codetype    %ContentType;  #IMPLIED
  archive     %UriList;      #IMPLIED
  standby     %Text;         #IMPLIED
  height      %Length;       #IMPLIED
  width       %Length;       #IMPLIED
  usemap      %URI;          #IMPLIED
  name        NMTOKEN        #IMPLIED
  tabindex    %Number;       #IMPLIED
  align       %ImgAlign;     #IMPLIED
  border      %Pixels;       #IMPLIED
  hspace      %Pixels;       #IMPLIED
  vspace      %Pixels;       #IMPLIED
  >
<!--
  param is used to supply a named property value.
  In XML it would seem natural to follow RDF and support an
  abbreviated syntax where the param elements are replaced
  by attribute value pairs on the object start tag.
-->
<!ELEMENT param EMPTY>
<!ATTLIST param
  id          ID             #IMPLIED
  name        CDATA          #REQUIRED
  value       CDATA          #IMPLIED
  valuetype   (data|ref|object) "data"
  type        %ContentType;  #IMPLIED
  >
<!--================== Java applet =====================================-->
<!--
  One of code or object attributes must be present.
  Place param elements before other content.
-->
<!ELEMENT applet (#PCDATA | param | %block; | form | %inline; | %misc;)*>
<!ATTLIST applet
  %coreattrs;
  codebase    %URI;          #IMPLIED
  archive     CDATA          #IMPLIED
  code        CDATA          #IMPLIED
  object      CDATA          #IMPLIED
```

```
  alt         %Text;         #IMPLIED
  name        NMTOKEN        #IMPLIED
  width       %Length;       #REQUIRED
  height      %Length;       #REQUIRED
  align       %ImgAlign;     #IMPLIED
  hspace      %Pixels;       #IMPLIED
  vspace      %Pixels;       #IMPLIED
  >
<!--=================== Images ============================================-->
<!--
   To avoid accessibility problems for people who aren't
   able to see the image, you should provide a text
   description using the alt and longdesc attributes.
   In addition, avoid the use of server-side image maps.
-->
<!ELEMENT img EMPTY>
<!ATTLIST img
  %attrs;
  src         %URI;          #REQUIRED
  alt         %Text;         #REQUIRED
  name        NMTOKEN        #IMPLIED
  longdesc    %URI;          #IMPLIED
  height      %Length;       #IMPLIED
  width       %Length;       #IMPLIED
  usemap      %URI;          #IMPLIED
  ismap       (ismap)        #IMPLIED
  align       %ImgAlign;     #IMPLIED
  border      %Length;       #IMPLIED
  hspace      %Pixels;       #IMPLIED
  vspace      %Pixels;       #IMPLIED
  >
<!-- usemap points to a map element which may be in this document
  or an external document, although the latter is not widely supported -->
<!--=================== Client-side image maps ============================-->
<!-- These can be placed in the same document or grouped in a
     separate document although this isn't yet widely supported -->
<!ELEMENT map ((%block; | form | %misc;)+ | area+)>
<!ATTLIST map
  %i18n;
  %events;
  id          ID             #REQUIRED
  class       CDATA          #IMPLIED
  style       %StyleSheet;   #IMPLIED
  title       %Text;         #IMPLIED
  name        CDATA          #IMPLIED
  >
<!ELEMENT area EMPTY>
<!ATTLIST area
  %attrs;
  shape       %Shape;        "rect"
  coords      %Coords;       #IMPLIED
```

```
  href        %URI;          #IMPLIED
  nohref      (nohref)       #IMPLIED
  alt         %Text;         #REQUIRED
  tabindex    %Number;       #IMPLIED
  accesskey   %Character;    #IMPLIED
  onfocus     %Script;       #IMPLIED
  onblur      %Script;       #IMPLIED
  target      %FrameTarget;  #IMPLIED
  >
<!--================ Forms ===============================================-->
<!ELEMENT form %form.content;>   <!-- forms shouldn't be nested -->
<!ATTLIST form
  %attrs;
  action      %URI;          #REQUIRED
  method      (get|post)     "get"
  name        NMTOKEN        #IMPLIED
  enctype     %ContentType;  "application/x-www-form-urlencoded"
  onsubmit    %Script;       #IMPLIED
  onreset     %Script;       #IMPLIED
  accept      %ContentTypes; #IMPLIED
  accept-charset %Charsets;  #IMPLIED
  target      %FrameTarget;  #IMPLIED
  >
<!--
  Each label must not contain more than ONE field
  Label elements shouldn't be nested.
-->
<!ELEMENT label %Inline;>
<!ATTLIST label
  %attrs;
  for         IDREF          #IMPLIED
  accesskey   %Character;    #IMPLIED
  onfocus     %Script;       #IMPLIED
  onblur      %Script;       #IMPLIED
  >
<!ENTITY % InputType
  "(text | password | checkbox |
    radio | submit | reset |
    file | hidden | image | button)"
   >
<!-- the name attribute is required for all but submit & reset -->
<!ELEMENT input EMPTY>     <!-- form control -->
<!ATTLIST input
  %attrs;
  type        %InputType;    "text"
  name        CDATA          #IMPLIED
  value       CDATA          #IMPLIED
  checked     (checked)      #IMPLIED
  disabled    (disabled)     #IMPLIED
  readonly    (readonly)     #IMPLIED
  size        CDATA          #IMPLIED
```

```
  maxlength   %Number;       #IMPLIED
  src         %URI;          #IMPLIED
  alt         CDATA          #IMPLIED
  usemap      %URI;          #IMPLIED
  tabindex    %Number;       #IMPLIED
  accesskey   %Character;    #IMPLIED
  onfocus     %Script;       #IMPLIED
  onblur      %Script;       #IMPLIED
  onselect    %Script;       #IMPLIED
  onchange    %Script;       #IMPLIED
  accept      %ContentTypes; #IMPLIED
  align       %ImgAlign;     #IMPLIED
  >
<!ELEMENT select (optgroup|option)+>  <!-- option selector -->
<!ATTLIST select
  %attrs;
  name        CDATA          #IMPLIED
  size        %Number;       #IMPLIED
  multiple    (multiple)     #IMPLIED
  disabled    (disabled)     #IMPLIED
  tabindex    %Number;       #IMPLIED
  onfocus     %Script;       #IMPLIED
  onblur      %Script;       #IMPLIED
  onchange    %Script;       #IMPLIED
  >
<!ELEMENT optgroup (option)+>   <!-- option group -->
<!ATTLIST optgroup
  %attrs;
  disabled    (disabled)     #IMPLIED
  label       %Text;         #REQUIRED
  >
<!ELEMENT option (#PCDATA)>     <!-- selectable choice -->
<!ATTLIST option
  %attrs;
  selected    (selected)     #IMPLIED
  disabled    (disabled)     #IMPLIED
  label       %Text;         #IMPLIED
  value       CDATA          #IMPLIED
  >
<!ELEMENT textarea (#PCDATA)>     <!-- multi-line text field -->
<!ATTLIST textarea
  %attrs;
  name        CDATA          #IMPLIED
  rows        %Number;       #REQUIRED
  cols        %Number;       #REQUIRED
  disabled    (disabled)     #IMPLIED
  readonly    (readonly)     #IMPLIED
  tabindex    %Number;       #IMPLIED
  accesskey   %Character;    #IMPLIED
  onfocus     %Script;       #IMPLIED
  onblur      %Script;       #IMPLIED
```

```
  onselect    %Script;       #IMPLIED
  onchange    %Script;       #IMPLIED
  >
<!--
  The fieldset element is used to group form fields.
  Only one legend element should occur in the content
  and if present should only be preceded by whitespace.
-->
<!ELEMENT fieldset (#PCDATA | legend | %block; | form | %inline; | %misc;)*>
<!ATTLIST fieldset
  %attrs;
  >
<!ENTITY % LAlign "(top|bottom|left|right)">
<!ELEMENT legend %Inline;>     <!-- fieldset label -->
<!ATTLIST legend
  %attrs;
  accesskey   %Character;    #IMPLIED
  align       %LAlign;       #IMPLIED
  >
<!--
 Content is %Flow; excluding a, form, form controls, iframe
-->
<!ELEMENT button %button.content;>  <!-- push button -->
<!ATTLIST button
  %attrs;
  name        CDATA          #IMPLIED
  value       CDATA          #IMPLIED
  type        (button|submit|reset) "submit"
  disabled    (disabled)     #IMPLIED
  tabindex    %Number;       #IMPLIED
  accesskey   %Character;    #IMPLIED
  onfocus     %Script;       #IMPLIED
  onblur      %Script;       #IMPLIED
  >
<!-- single-line text input control (DEPRECATED) -->
<!ELEMENT isindex EMPTY>
<!ATTLIST isindex
  %coreattrs;
  %i18n;
  prompt      %Text;         #IMPLIED
  >
<!--======================= Tables ===================================-->
<!-- Derived from IETF HTML table standard, see [RFC1942] -->
<!--
 The border attribute sets the thickness of the frame around the
 table. The default units are screen pixels.
 The frame attribute specifies which parts of the frame around
 the table should be rendered. The values are not the same as
 CALS to avoid a name clash with the valign attribute.
-->
<!ENTITY % TFrame "(void|above|below|hsides|lhs|rhs|vsides|box|border)">
```

```
<!--
 The rules attribute defines which rules to draw between cells:
 If rules is absent then assume:
     "none" if border is absent or border="0" otherwise "all"
-->
<!ENTITY % TRules "(none | groups | rows | cols | all)">

<!-- horizontal placement of table relative to document -->
<!ENTITY % TAlign "(left|center|right)">
<!-- horizontal alignment attributes for cell contents
  char        alignment char, e.g. char=':'
  charoff     offset for alignment char
-->
<!ENTITY % cellhalign
  "align      (left|center|right|justify|char) #IMPLIED
   char       %Character;    #IMPLIED
   charoff    %Length;       #IMPLIED"
  >
<!-- vertical alignment attributes for cell contents -->
<!ENTITY % cellvalign
  "valign     (top|middle|bottom|baseline) #IMPLIED"
  >
<!ELEMENT table
     (caption?, (col*|colgroup*), thead?, tfoot?, (tbody+|tr+))>
<!ELEMENT caption  %Inline;>
<!ELEMENT thead    (tr)+>
<!ELEMENT tfoot    (tr)+>
<!ELEMENT tbody    (tr)+>
<!ELEMENT colgroup (col)*>
<!ELEMENT col      EMPTY>
<!ELEMENT tr       (th|td)+>
<!ELEMENT th       %Flow;>
<!ELEMENT td       %Flow;>
<!ATTLIST table
  %attrs;
  summary     %Text;         #IMPLIED
  width       %Length;       #IMPLIED
  border      %Pixels;       #IMPLIED
  frame       %TFrame;       #IMPLIED
  rules       %TRules;       #IMPLIED
  cellspacing %Length;       #IMPLIED
  cellpadding %Length;       #IMPLIED
  align       %TAlign;       #IMPLIED
  bgcolor     %Color;        #IMPLIED
  >
<!ENTITY % CAlign "(top|bottom|left|right)">
<!ATTLIST caption
  %attrs;
  align       %CAlign;       #IMPLIED
  >
<!--
```

```
colgroup groups a set of col elements. It allows you to group
several semantically related columns together.
-->
<!ATTLIST colgroup
  %attrs;
  span        %Number;       "1"
  width       %MultiLength;  #IMPLIED
  %cellhalign;
  %cellvalign;
  >
<!--
 col elements define the alignment properties for cells in
 one or more columns.
 The width attribute specifies the width of the columns, e.g.
     width=64        width in screen pixels
     width=0.5*      relative width of 0.5
 The span attribute causes the attributes of one
 col element to apply to more than one column.
-->
<!ATTLIST col
  %attrs;
  span        %Number;       "1"
  width       %MultiLength;  #IMPLIED
  %cellhalign;
  %cellvalign;
  >
<!--
    Use thead to duplicate headers when breaking table
    across page boundaries, or for static headers when
    tbody sections are rendered in scrolling panel.
    Use tfoot to duplicate footers when breaking table
    across page boundaries, or for static footers when
    tbody sections are rendered in scrolling panel.
    Use multiple tbody sections when rules are needed
    between groups of table rows.
-->
<!ATTLIST thead
  %attrs;
  %cellhalign;
  %cellvalign;
  >
<!ATTLIST tfoot
  %attrs;
  %cellhalign;
  %cellvalign;
  >
<!ATTLIST tbody
  %attrs;
  %cellhalign;
  %cellvalign;
  >
```

```
<!ATTLIST tr
  %attrs;
  %cellhalign;
  %cellvalign;
  bgcolor     %Color;        #IMPLIED
  >
<!-- Scope is simpler than headers attribute for common tables -->
<!ENTITY % Scope "(row|col|rowgroup|colgroup)">
<!-- th is for headers, td for data and for cells acting as both -->
<!ATTLIST th
  %attrs;
  abbr        %Text;         #IMPLIED
  axis        CDATA          #IMPLIED
  headers     IDREFS         #IMPLIED
  scope       %Scope;        #IMPLIED
  rowspan     %Number;       "1"
  colspan     %Number;       "1"
  %cellhalign;
  %cellvalign;
  nowrap      (nowrap)       #IMPLIED
  bgcolor     %Color;        #IMPLIED
  width       %Pixels;       #IMPLIED
  height      %Pixels;       #IMPLIED
  >
<!ATTLIST td
  %attrs;
  abbr        %Text;         #IMPLIED
  axis        CDATA          #IMPLIED
  headers     IDREFS         #IMPLIED
  scope       %Scope;        #IMPLIED
  rowspan     %Number;       "1"
  colspan     %Number;       "1"
  %cellhalign;
  %cellvalign;
  nowrap      (nowrap)       #IMPLIED
  bgcolor     %Color;        #IMPLIED
  width       %Pixels;       #IMPLIED
  height      %Pixels;       #IMPLIED
>
```

F

Коды символов

В следующей таблице собраны стандартные, предлагаемые для включения в стандарт и некоторые нестандартные, но обычно поддерживаемые символьные замены для HTML и XHTML.

Именные замены (если они определены) представлены для соответствующих символов и могут использоваться в виде &имя; для определения символов, которые должен отобразить броузер. В противном случае или в качестве альтернативы именной замене используйте трехзначные коды символов в виде &#nnn;[1]. Сами обозначаемые символы, возможно, будут отображены броузером, а возможно, и нет. Это зависит от платформы и от того, какой шрифт выбрал пользователь для отображения документа.

В таблице представлены не все 256 символов набора ISO. Отсутствующие символы не распознаются броузерами ни в виде именных замен, ни в виде числовых кодов.

Чтобы гарантировать полное соответствие ваших документов стандартам HTML 4.0 и XHTML 1.0, используйте коды только тех символов, для которых поле в колонке «Соответствие» пусто. Символы, у которых в последней колонке стоит «!!!», формально не определены стандартом. Используйте их на свой страх и риск.

Числовой код	Именная замена	Символ	Описание	Соответствие
			Горизонтальная табуляция	
		Перевод строки	
		Возврат каретки	

[1] Код символа в стандарте Unicode. – *Примеч. науч. ред.*

Числовой код	Именная замена	Символ	Описание	Соответствие
			Пробел	
!		!	Восклицательный знак	
"	"	"	Кавычка	
#		#	Решетка (диез)	
$		$	Доллар	
%		%	Процент	
&	&	&	Амперсанд	
'		'	Апостроф	
(		(	Левая скобка	
)		)	Правая скобка	
*		*	Звездочка	
+		+	Плюс	
,		,	Запятая	
-		-	Дефис	
.		.	Точка	
/		/	Наклонная черта (слэш)	
0 – 9		0–9	Цифры 0–9	
:		:	Двоеточие	
;		;	Точка с запятой	
<	<	<	Меньше	
=		=	Равно	
>	>	>	Больше	
?		?	Вопросительный знак	
@		@	Коммерческое эт (собака)	
A – Z		A–Z	Буквы A–Z	
[		[	Левая квадратная скобка	
\		\	Обратный слэш	
]		]	Правая квадратная скобка	
^		^	Вставка (карет)	
_		_	Подчеркивание	
`		`	Гравис	

Числовой код	Именная замена	Символ	Описание	Соответствие
a – z		a–z	Латинские буквы a–z	
{		{	Левая фигурная скобка	
			|	Вертикальная черта	
}		}	Правая фигурная скобка	
~		~	Тильда	
‚		‚	Нижняя левая одиночная кавычка	!!!
ƒ		ƒ	Флорин	!!!
„		„	Нижняя левая двойная кавычка	!!!
…		…	Многоточие	!!!
†		†	Крестик	!!!
‡		‡	Двойной крестик	!!!
ˆ		ˆ	Циркумфлекс	!!!
‰		‰	Промилле	!!!
Š		Љ	Заглавная S, карон	!!!
‹		‹	Меньше	!!!
Œ		Њ	Заглавная OE, лигатура	!!!
Ž		Ћ	Заглавная Z, карон	!!!
‘		‘	Левая одиночная кавычка	!!!
’		’	Правая одиночная кавычка	!!!
“		“	Левая двойная кавычка	!!!
”		”	Правая двойная кавычка	!!!
•		•	Буллит	!!!
–		–	Короткое тире	!!!
—		—	Длинное тире	!!!
˜		~	Тильда	!!!
™		™	Торговая марка	!!!
š		љ	Строчная s, карон	!!!
›		›	Больше	!!!
œ		њ	Строчная oe, лигатура	!!!
ž		ћ	Строчная z, карон	!!!
Ÿ		џ	Заглавная Y, умляут	!!!
			Неразрывный пробел	

Числовой код	Именная замена	Символ	Описание	Соответствие
¡	¡	ў	Перевернутый восклицательный знак	
¢	¢	ў	Цент	
£	£	J	Фунт	
¤	¤	¤	Валюта	
¥	¥	Ґ	Йена	
¦	¦	¦	Разорванная вертикальная черта	
§	§	§	Параграф	
¨	¨	Ё	Умляут	
©	©	©	Копирайт	
ª	ª	Є	Антитеза	
«	«	«	Левая угловая кавычка	
¬	¬	¬	Отрицание	
		-	Мягкий перенос	
®	®	®	Зарегистрированная торговая марка	
¯	¯	Ї	Долгая гласная	
°	°	°	Градус	
±	±	±	Плюс или минус	
²	²	²	Верхний индекс 2	
³	³	³	Верхний индекс 3	
´	´	ґ	Акут	
µ	µ	µ	Микро (греческая мю)	
¶	¶	¶	Абзац	
·	·	·	Центральная точка	
¸	¸	¸	Седиль	
¹	¹	¹	Верхний индекс 1	
º	º	є	Теза	
»	»	»	Правая угловая кавычка	
¼	¼	$^1/_4$	Дробь четверть	
½	½	$^1/_2$	Дробь половина	
¾	¾	$^3/_4$	Дробь три четверти	
¿	¿	ї	Перевернутый вопросительный знак	

Числовой код	Именная замена	Символ	Описание	Соответствие
À	À	А	Заглавная A, гравис	
Á	Á	Б	Заглавная A, акут	
Â	Â	В	Заглавная A, циркумфлекс	
Ã	&Атильда;	Г	Заглавная A, тильда	
Ä	Ä	Д	Заглавная A, умляут	
Å	Å	Е	Заглавный ангстрем	
Æ	Æ	Ж	Заглавная AE, лигатура	
Ç	Ç	З	Заглавная C, седиль	
È	È	И	Заглавная E, гравис	
É	É	Й	Заглавная E, акут	
Ê	Ê	К	Заглавная E, циркумфлекс	
Ë	Ë	Л	Заглавная E, умляут	
Ì	Ì	М	Заглавная I, гравис	
Í	Í	Н	Заглавная I, акут	
Î	Î	О	Заглавная I, циркумфлекс	
Ï	Ï	П	Заглавная I, умляут	
Ð	Ð	Ньь	Заглавная исландская eth	
Ñ	Ñ	С	Заглавная N, тильда	
Ò	Ò	Т	Заглавная O, гравис	
Ó	Ó	У	Заглавная O, акут	
Ô	Ô	Ф	Заглавная O, циркумфлекс	
Õ	Õ	Х	Заглавная O, тильда	
Ö	Ö	Ц	Заглавная O, умляут	
×	×	Ч	Умножение	
Ø	Ø	Ш	Заглавная O, перечеркнутая	
Ù	Ù	Щ	Заглавная U, гравис	
Ú	Ú	Ъ	Заглавная U, акут	
Û	Û	Ы	Заглавная U, циркумфлекс	
Ü	Ü	Ь	Заглавная U, умляут	
Ý	Ý	Э	Заглавная Y, акут	
Þ	Þ	Ю	Заглавная исландская торн	
ß	ß	Я	Строчная sz, лигатура, немецкая	
à	à	а	Строчная a, гравис	

Числовой код	Именная замена	Символ	Описание	Соответствие
á	á	б	Строчная a, акут	
â	â	в	Строчная a, циркумфлекс	
ã	&атильда;	г	Строчная a, тильда	
ä	ä	д	Строчная a, умляут	
å	å	е	Строчный ангстрем	
æ	æ	ж	Строчная ae, лигатура	
ç	ç	з	Строчная c, седиль	
è	è	и	Строчная e, гравис	
é	é	й	Строчная e, акут	
ê	ê	к	Строчная e, циркумфлекс	
ë	ë	л	Строчная e, умляут	
ì	ì	м	Строчная i, гравис	
í	í	н	Строчная i, акут	
î	î	о	Строчная i, циркумфлекс	
ï	ï	о	Строчная i, умляут	
ð	ð	р	Строчная исландская eth	
ñ	ñ	с	Строчная n, тильда	
ò	ò	т	Строчная o, гравис	
ó	ó	у	Строчная o, акут	
ô	ô	ф	Строчная o, циркумфлекс	
õ	õ	х	Строчная o, тильда	
ö	ö	ц	Строчная o, умляут	
÷	÷	ч	Деление	
ø	ø	ш	Строчная o, перечеркнутая	
ù	ù	щ	Строчная u, гравис	
ú	ú	ъ	Строчная u, акут	
û	û	ы	Строчная u, циркумфлекс	
ü	ü	ь	Строчная u, умляут	
ý	ý	э	Строчная y, акут	
þ	þ	ю	Строчная исландская торн	
ÿ	ÿ	я	Строчная y, умляут	

G

Названия и коды цветов

Работая с популярными броузерами и в соответствии со стандартом каскадных таблиц стилей (CSS), можно предписать цвет отображения для различных элементов документа. Это делается указанием кода цвета или его стандартного названия. Пользователь может отменить эти определения цветов, используя настройки броузера.

Коды цветов

Во всех случаях можно установить цвет HTML-элементов, таких как текст в теге `<body>`, фон таблицы и т. д., в виде шестизначного шестнадцатеричного числа, представляющего красный, зеленый и синий компоненты цвета. Две первые цифры соответствуют красному компоненту цвета, две следующие – зеленому и две последние – синему. Значение 00 соответствует полностью выключенному компоненту, шестнадцатеричное значение FF (десятичное 255) соответствует полностью включенному компоненту. Таким образом, ярко-красный цвет – это FF0000, ярко-зеленый – 00FF00, ярко-синий – 0000FF. Другие основные цвета являются смесью компонентов, например желтый (yellow) (FFFF00), пурпурный (magenta) (FF00FF) и голубой (cyan) (FF00FF). Нетрудно догадаться, как записываются белый (FFFFFF) и черный (000000) цвета.

Эти значения указываются в тегах, заменяя название цвета знаком решетки (#), за которым следует RGB-триплет. Таким образом, чтобы сделать все использованные гиперссылки пурпурными, напишите:

```
<body vlink="#FF00FF">
```

Названия цветов

Определение RGB-триплета для цветов, отличающихся от простейших (попробуйте закодировать точно такие изысканные цвета, как «рарауа

whip» или «navajo white»), не такое простое дело. Можно сойти с ума, подбирая RGB-триплет для получения нужного оттенка, особенно когда каждый раз нужно загружать документ в броузер, чтобы увидеть результат.

Для облегчения жизни авторов стандарты определяют шестнадцать стандартных названий цветов, которые можно использовать всюду, где могут появляться числовые коды цветов. Например, можно сделать все использованные гиперссылки пурпурными, записав тег `<body>` со следующим значением соответствующего атрибута:

```
<body vlink="magenta">
```

В стандартах HTML 4 и XHTML определены следующие названия цветов, соответствующие указанным в скобках RGB-кодам:

aqua (#00FFFF)	gray (#808080)	navy (#000080)	silver (#C0C0C0)
black (#000000)	green (#008000)	olive (#808000)	teal (#008080)
blue (#0000FF)	lime (#00FF00)	purple (#800080)	yellow (#FFFF00)
fuchsia (#FF00FF)	maroon (#800000)	red (#FF0000)	white (#FFFFFF)

Популярные броузеры выходят далеко за пределы стандартов и поддерживают несколько сотен цветов, определяемых для использования в X Window System. Заметьте, что в этих названиях цветов не должно быть пробелов, а также то, что во всех названиях цветов можно с равным успехом писать «grey» и «gray».

Цвета, помеченные звездочкой (*), на деле представляют семейство цветов, пронумерованных от единицы до четырех. Так, имеется четыре варианта синего с названиями «blue1», «blue2», «blue3» и «blue4» вместе с простым «blue». «blue1» – это самый светлый из четырех, «blue4» – самый темный. Название цвета без номера соответствует первому, таким образом, «blue» и «blue1» совпадают.

Наконец, если всего предыдущего недостаточно, еще имеется сто вариантов серого цвета (grey или gray) с номерами от одного до ста. «gray1» – самый темный, «gray100» – самый светлый, а «gray» очень близок к «gray75».

Вот расширенный набор названий цветов:

aliceblue	darkturquoise	lightseagreen	palevioletred*
antiquewhite*	darkviolet	lightskyblue*	papayawhip
aquamarine*	deeppink*	lightslateblue	peachpuff*
azure*	deepskyblue*	lightslategray	peru
beige	dimgray	lightsteelblue*	pink*
bisque*	dodgerblue*	lightyellow*	plum*
black	firebrick*	limegreen	powderblue

blanchedalmond	floralwhite	linen	purple*
blue*	forestgreen	magenta*	red*
blueviolet	gainsboro	maroon*	rosybrown*
brown*	ghostwhite	mediumaquamarine	royalblue*
burlywood*	gold*	mediumblue	saddlebrown
cadetblue*	goldenrod*	mediumorchid*	salmon*
chartreuse*	gray	mediumpurple*	sandybrown
chocolate*	green*	mediumseagreen	seagreen*
coral*	greenyellow	mediumslateblue	seashell*
cornflowerblue	honeydew*	mediumspringgreen	sienna*
cornsilk*	hotpink*	mediumturquoise	skyblue*
cyan*	indianred*	mediumvioletred	slateblue*
darkblue	ivory*	midnightblue	slategray*
darkcyan	khaki*	mintcream	snow*
darkgoldenrod*	lavender	mistyrose*	springgreen*
darkgray	lavenderblush*	moccasin	steelblue*
darkgreen	lawngreen	navajowhite*	tan*
darkkhaki	lemonchiffon*	navy	thistle*
darkmagenta	lightblue*	navyblue	tomato*
darkolivegreen*	lightcoral	oldlace	turquoise*
darkorange*	lightcyan*	olivedrab*	violet
darkorchid*	lightgoldenrod*	orange*	violetred*
darkred	lightgoldenrodyellow	orangered*	wheat*
darksalmon	lightgray	orchid*	white
darkseagreen*	lightgreen	palegoldenrod	whitesmoke
darkslateblue	lightpink*	palegreen*	yellow*
darkslategray*	lightsalmon*	paleturquoise*	yellowgreen

Стандартная палитра

Поддержка сотен названий цветов и миллионов RGB-триплетов – это хорошо, но реальность состоит в том, что большая (хотя сокращающаяся) часть пользователей имеют системы, отображающие только 256 цветов. Встретившись с цветом, не входящим в этот набор из 256 цветов, броузер может поступить двояко: преобразовать цвет в один из принадлежащих палитре или сымитировать этот цвет путем смешивания цветов палитры.

Преобразовать легко – незнакомый цвет сравнивается со всеми цветами палитры и заменяется самым близким из них. Смешивание сложнее. Используя два или большее количество цветов, броузер приближает «неправильный» цвет сочетанием взятых в некоторой пропорции цветов палитры. При близком рассмотрении вы увидите узор из разноцветных пикселов. На расстоянии пикселы смешиваются, создавая впечатление цвета, близкого к оригиналу.

Вообще, ваши изображения будут выглядеть лучше всего, если вы сумеете избежать как преобразования цветов, так и смешивания. Преобразование приводит к «непопаданию в цвет», а смешивание может сделать изображение размытым. Как этого избежать? Легко – используйте при создании своих изображений только цвета стандартной палитры.

Стандартная палитра содержит на самом деле 216 значений. В ней шесть вариантов красного, шесть – зеленого и шесть – синего, которые комбинируются всеми возможными способами для создания 216 (6×6×6) цветов. Эти варианты имеют яркость, соответствующую десятичным 0, 51, 102, 153, 204 и 255, что соответствует шестнадцатеричным 00, 33, 66, 99, CC и FF. Такие цвета, как 003333 (темно-голубой, dark cyan) и 999999 (средний серый, medium gray), присутствуют в палитре и не нуждаются ни в преобразовании, ни в смешивании цветов.

Имейте в виду, что многие цвета из расширенного набора не входят в стандартную палитру и будут преобразованы или сымитированы путем смешивания, чтобы получился (приходится надеяться) похожий цвет.

При создании изображений старайтесь использовать цвета стандартной палитры. При выборе цвета для текста, гиперссылок или фона проследите, чтобы выбранные цвета принадлежали стандартной палитре. Ваши страницы будут выглядеть лучше, и разные броузеры будут отображать их более похожим образом.

H

Расширенные средства макетирования для Netscape

При возникновении Netscape разработчики этой корпорации находились на переднем крае конструирования броузеров, отвечающих потребностям коммерческих приложений. В течение нескольких лет они последовательно расширяли HTML, предоставляя в каждой следующей версии своего продукта значительно большие возможности для создания интересных макетов страниц, чем могли предложить все остальные броузеры. И им это удавалось. Netscape Navigator доминировал вплоть до начала XXI века, когда появились каскадные таблицы стилей (CSS) и другие стандарты. В конце концов корпорация Microsoft отвоевала свое место на рынке.

В этом приложении мы документируем для истории три функциональных особенности, которыми обладали броузеры Netscape версий по четвертую включительно, причем только они. Это выделение пустого пространства, расположение текста в несколько колонок и наложение слоев. Соответствующие теги соблазняют веб-дизайнеров потрясающими возможностями разметки страниц. Если вам интересно, можете с ними поэкспериментировать, но мы вас предупреждаем: они никогда не будут включены в стандарты HTML/XHTML. Более того, они не поддерживаются последней версией броузера Netscape Navigator.

Выделение пустого пространства

Одним из простейших элементов конструкции любой страницы является пустое пространство, окружающее ее содержимое. Пустое пространство часто не менее важно для внешнего вида страницы, чем области, заполненные текстом и изображением. Называемые здесь *пробельными элементами*, эти пустые области придают форму содержимому вашего документа.

В HTML нет способа для создания пустого пространства на странице, кроме использования тега <pre>, заполненного пустыми строками, и пустых изображений. Броузеры, действуя в соответствии со стандартами HTML/XHTML, удаляют ведущие, закрывающие и все другие лишние пробелы в тексте и игнорируют лишние переводы строки. Netscape исправляет этот недостаток при помощи тега <spacer>. [
, 4.6.1]

Тег <spacer> (устарел)

Используйте тег <spacer> для создания горизонтальных, вертикальных и прямоугольных пропусков в документах, отображаемых броузером Netscape 4.

<spacer>	
Функция:	Определяет в документе пустую область
Атрибуты:	align, height, size, type, width
Завершающий тег:	В HTML отсутствует
Содержит:	Ничего
Может содержаться в:	*тексте*

Создание горизонтального пропуска

Чаще всего тег <spacer> используют для создания отступа в строке текста. Чтобы добиться этого эффекта, необходимо присвоить атрибуту type значение horizontal и использовать атрибут size для определения ширины в пикселах (не в символах) горизонтальной области. Например:

```
<spacer type=horizontal size=100>
```

вставляет 100 пикселов пустого пространства в текущую строку текста. Netscape 4 присоединит последующее содержимое к концу пустого промежутка, если в текущей строке осталось свободное место. В противном случае он поместит следующий элемент на новой строке, следуя обычному форматированию текста.

Если в текущей строке не хватает места для размещения всего пустого промежутка, описанного в теге <spacer>, броузер сокращает его до размеров текущей строки. По существу, размер вставляемого интервала является «мягким», предлагающим броузеру вставить в текущую строку пробел, который заканчивается либо по достижении указанной ширины, либо по окончании строки.

К примеру, если пробельный интервал имеет ширину 100 пикселов, а в текущей строке в окне броузера остается только 75 пикселов, Netscape 4 вставит в эту строку 75 пикселов пробела, а со следующей строки начнет отображение следующего за тегом <spacer> элемента. То есть горизонтальный пробел никогда не разбивается на несколько строк и не переносится на новую строку.

До сих пор горизонтальный пробел используется чаще всего для создания отступа в первой строке абзаца. Чтобы получить такой результат, просто поместите тег горизонтального пробела в начало абзаца:

```
<spacer type=horizontal size=50>
The effects of cooler weather on the kumquat's ripening process
 vary based upon the temperature.  Temperatures above 28&deg; sweeten the
fruit, while four or more hours below 28&deg; will damage the tree.
```

Результат можно увидеть на рис. Н.1.

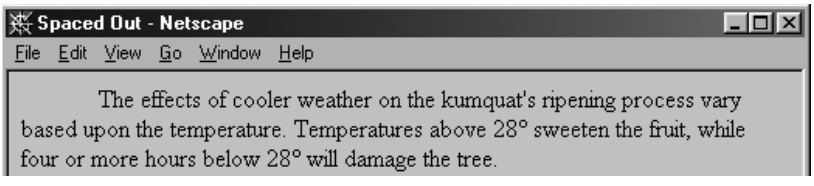

Рис. Н.1. *Горизонтальный отступ в первой строке абзаца (только Netscape 4)*

Можно, разумеется, использовать горизонтальные пропуски, чтобы оставлять дополнительные промежутки между словами и буквами в тексте. Это может оказаться полезным при отображении стихов или при распечатке. Но не используйте этот тег для создания блока текста с равномерным отступом – невозможно заранее предвидеть ширину окна пользовательского броузера, размеры шрифтов и т. д., а следовательно, и того, в каком месте броузер разорвет конкретную строку текста. Вместо этого используйте тег `<blockquote>` или подберите подходящую ширину поля абзаца при помощи стилей.

Создание вертикальных пропусков

Дополнительные интервалы между строками текста или абзацами в документе можно вставить, присваивая атрибуту `type` тега `<spacer>` значение `vertical`. Надо включить в тег и атрибут `size`. Его значение должно быть положительным целым числом, равным ширине интервала в пикселах.

Вертикальный пропуск действует точно так же, как тег `<br>`. Оба тега вызывают немедленный разрыв строки. Разница, разумеется, состоит в том, что вертикальный пропуск позволяет руководить тем, сколько Netscape 4 должен оставить пустого пространства между текущей и следующей строкой. Вертикальный пропуск добавляется к обычному интервалу, оставляемому под текущей строкой текста в соответствии с установками текущего абзаца, и, таким образом, не может быть меньше него.

Поскольку высота HTML-страниц ничем не ограничена, вертикальный пропуск может быть высотой в любое число пикселов. Было бы неразумно, конечно, допускать здесь излишества (если хотите, попробуйте написать `size=1.000.000.000`). Большинство современных мони-

торов имеют вертикальное разрешение не больше, чем 1024 линии. Так что высота пропуска 1025 пикселов гарантирует, что следующая строка текста будет начинаться на следующей странице, если вы именно этого добиваетесь.

Вертикальные пропуски совсем не так популярны, как горизонтальные, но они все же могут быть полезны. В нижеследующем тексте мы использовали вертикальный пропуск, чтобы оставить несколько большее расстояние между заголовками и основным текстом:

```
<h1 align=right>Temperature Effects</h1>
<spacer type=vertical size=50>
The effects of cooler weather on the kumquat's ripening process vary based
upon the temperature.  Temperatures above 28&deg; sweeten the fruit, while
four or more hours below 28&deg; will damage the tree.
```

Результат можно увидеть на рис. H.2.

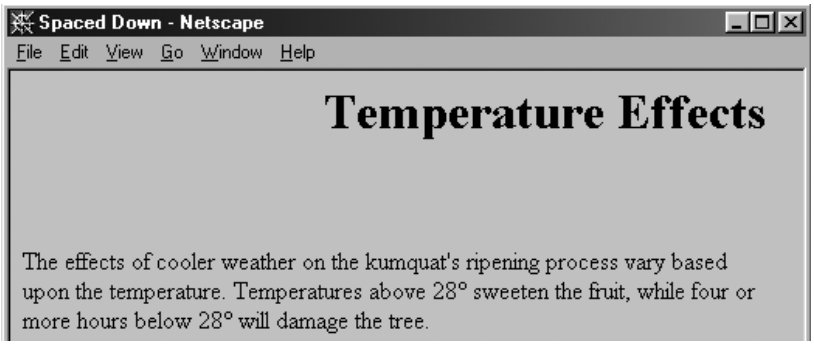

Рис. H.2. *Использование вертикального пропуска для отделения заголовка от текста (только Netscape 4)*

Создание пустых блоков

Пропуски третьего типа представляют собой прямоугольные блоки пустого пространства, очень похожие на пустые изображения. Чтобы полностью определить пропуск такого типа, присвойте атрибуту `type` значение `block` и включите в тег три других атрибута: `width`, `height` и `align`.

Атрибуты `width` и `height` определяют размеры прямоугольника в пикселах или процентах от размеров объемлющего его элемента. Эти атрибуты используются, только если значение атрибута `type` — это `block`, в противном случае они игнорируются. Если размеры указываются в пикселах, нужно присвоить каждому из атрибутов `width` и `height` целое положительное значение. По умолчанию размеры устанавливаются равными нулю.

Третий обязательный атрибут пустого блока, `align`, управляет тем, как Netscape 4 размещает пустой блок относительно окружающего текста. Значения этого атрибута совпадают со значениями его тезки для атри-

бута <img>. Используйте значения top, texttop, middle, absmiddle, baseline, bottom и absbottom, чтобы добиться нужного вертикального выравнивания блока. Используйте значения left и right, чтобы сдвинуть блок к соответствующему краю окружающего текста и заставить окружающий текст обтекать блок. По умолчанию принимается значение bottom. Полное описание атрибута align и его значений можно найти в разделе 4.1.1.1.

Следующий HTML-фрагмент расставляет название сторон света вокруг пустого прямоугольника:

```
<center>
North
<br>
West
<spacer type=block width=50 height=50 align=absmiddle>
East
<br>
South
</center>
```

Результат показан на рис. Н.3.

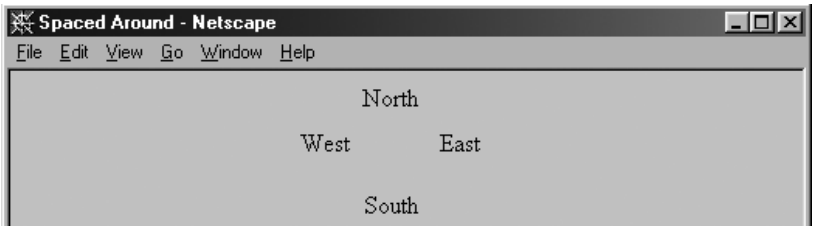

Рис. Н.3. *Использование пустого блока в документе (только Netscape 4)*

Имитация тега <spacer>

Поскольку тег <spacer> поддерживается только броузером Netscape версии 4 и более ранних, другие броузеры его игнорируют, разрушая заботливо выстроенный макет. Мы настоятельно рекомендуем вам использовать для этих целей свойство text-indent, определенное в стандарте CSS. Но возможна полная эмуляция действия тега <spacer> при помощи тега <img> и специального маленького изображения. Таким образом вы сможете добиться <spacer>-подобных эффектов даже в броузерах, не поддерживающих CSS.

Для эмуляции действия тега <spacer> потребуется маленькое совершенно прозрачное GIF-изображение. Поскольку его никто и никогда не увидит, можно сделать его сколь угодно малым. Мы рекомендуем использовать изображение GIF размером 1×1 пиксел. В следующих примерах наше крохотное прозрачное изображение называется *small.gif*.

Чтобы изобразить такой горизонтальный пропуск:

```
<spacer type=horizontal size=n>
```

используйте следующий тег <img>:

```
<img src=small.gif width=n height=1>
```

Замените, разумеется, *n* нужной шириной в пикселах. Имейте в виду, однако, что ширина тега <img> является фиксированной и, возможно, она не интегрируется в поток текста так, как это сделал бы тег <spacer>, особенно если тег <img> окажется рядом с концом текстовой строки.

Чтобы изобразить вертикальный пропуск, записанный так:

```
<spacer type=vertical size=n>
```

используйте следующий HTML-фрагмент:

```
<br>
<img src=small.gif width=1 height=n>
<br>
```

Теги
 в примере нужны, чтобы сымитировать разрыв строки, осуществляемый вертикальным тегом <spacer>. Здесь снова нужно заменить *n* – нужной высотой.[1]

Для имитации блока, записанного как:

```
<spacer type=block width=w height=h align=a>
```

используйте тег <img>:

```
<img src=small.gif width=w height=h align=a>
```

Замените *w*, *h* и *a* нужными шириной, высотой и значением выравнивания, соответственно.

Вывод текста в несколько колонок

Разбивка текста на несколько столбцов – это самое обычное средство в настольных издательских системах. Вывод текста в несколько колонок не только позволяет создавать привлекательные, по-разному от-

[1] Имейте в виду, что некоторые броузеры интерпретируют любую последовательность пробельных символов (включая символ перевода строки, табуляцию и т. д.) как один символ. Соответственно в этом примере между тегами
 по краям нашего прозрачного изображения появятся два пробельных текстовых символа. Поэтому таким HTML-кодом нельзя добавить вертикальный пропуск, величина которого меньше высоты строки текущего шрифта. Чтобы гарантированно установить вертикальный промежуток меньше высоты строки, следует убрать переводы строки между тегами
, записав тот же самый код в одну строку без пробелов между тегами. – *Примеч. науч. ред.*

форматированные страницы, но и облегчает чтение текста, делая строки короче. Конструкторы HTML-страниц жаждали иметь возможность с легкостью размещать несколько колонок текста на одной странице, но были вынуждены использовать всевозможные трюки, такие как таблицы с несколькими столбцами (см. главу 17).

Netscape 4 изящно разрешил эту проблему, введя свойственный только ему тег <multicol>. Хотя с применением этого тега невозможно создавать причудливые не сбалансированные и «шатающиеся» колонки, как это можно сделать с применением таблиц, с помощью тега <multicol> легко создаются обычные сбалансированные колонки текста. И хотя такая возможность реализуется только с Netscape 4, другие броузеры удовлетворительно отображают его содержимое.

Тег <multicol> (устарел)

Тег <multicol> выводит текст в несколько колонок и позволяет вам регулировать их число и размер.

	<multicol> (устарел)
Функция:	Выводит содержимое в несколько колонок
Атрибуты:	class, cols, gutter, style, width
Завершающий тег:	</multicol>; всегда используется
Содержит:	*body_content* (содержимое тела)
Может содержаться в:	*блоке*

Тег <multicol> может заключать в себе любое другое HTML-содержимое совершенно так же, как и тег . Все содержимое тега <multicol> отображается точно так же, как обычно, за исключением того, что Netscape 4 размещает это содержимое в нескольких колонках, а не в одной.

Тег <multicol> производит разрыв в потоке текста и перед выводом своего содержимого в несколько колонок вставляет пустую строку. По завершении тега вставляется еще одна пустая строка и возобновляется поток текста, подчиняющийся предыдущему форматированию и макетированию.

Netscape 4 автоматически балансирует колонки, делая каждую из них приблизительно одной и той же высоты. По возможности броузер перемещает текст из колонки в колонку, чтобы добиться балансировки. В некоторых случаях колонки невозможно совершенно сбалансировать из-за вложенных изображений, таблиц и других больших элементов.

Можно вкладывать теги <multicol> друг в друга, помещая одно множество колонок в другое. Хотя Netscape поддерживает вложения любой глубины, вложения третьего уровня уже приводят к непривлекательным результатам.

Атрибут cols

Атрибут `cols` используется в теге `<multicol>` для определения числа колонок. Если он опущен, Netscape 4 создаст одну колонку, как если бы тега `<multicol>` не было совсем. Можно создать любое число колонок, но на практике текст, расположенный в более чем трех или четырех колонках, на большинстве дисплеев будет неудобно читать.

Следующий пример создает макет с тремя колонками:

```
<h1 align=right>Temperature Effects</h1>
<multicol cols=3>
The effects of cooler weather on the kumquat's ripening process vary based
upon the temperature. Temperatures above 28&deg;sweeten the fruit, while four
or more hours below 28&deg; will damage the tree. The savvy quat farmer will
carefully monitor the temperature, especially in the predawn hours when
the mercury dips to its lowest point. Smudge pots and grove heaters may be
required to keep the trees warm; many growers will spray the trees with
water to create an insulating layer of ice over the fruit and leaves.
<p>
If a disastrous frost is predicted, below 20&deg;, the only recourse
may be to harvest the fruit early to save it from an assured disaster.
Kumquats may subsequently be ripened using any of the popular
methane and cyanoacrylate injection systems used for other citrus
fruits. Used correctly, these systems will produce fruit whose taste
is indistinguishable from tree-ripened kumquats.
</multicol>
```

Результат показан на рис. H.4.

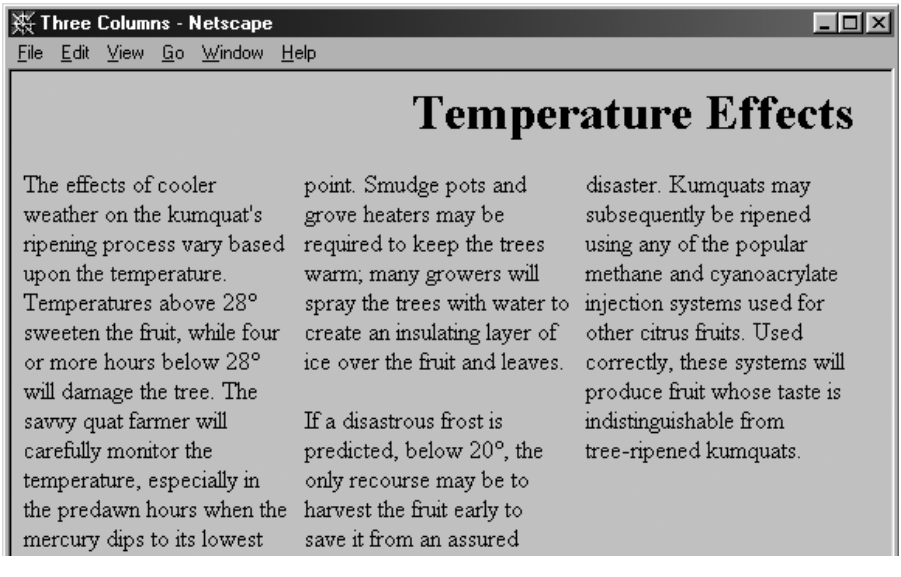

Рис. H.4. *Часть документа, выведенная в три колонки при помощи тега <multicol> (только Netscape 4)*

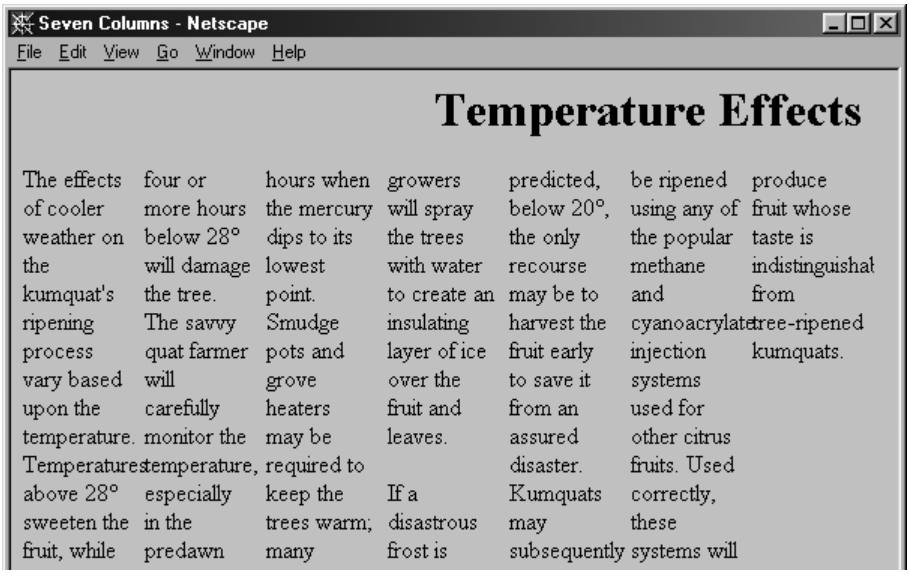

Рис. Н.5. *Текст невозможно читать – слишком много колонок (только Netscape 4)*

На рис. Н.4 можно увидеть, как Netscape 4 балансирует колонки, чтобы все они имели приблизительно одну длину. Также показано, что некоторые строки в колонках оказываются заметно короче других из-за того, что длинные слова перенесены на следующие строки. Эти растрепанные правые края колонок неизбежны и еще раз показывают, что не следует разбивать текст более чем на четыре или пять колонок. Наш пример еще можно будет прочитать, хоть и с трудом, если его вывести в пять колонок, но он разрушится совсем и даже вызовет ошибки вывода, если присвоить `cols` значение 7. Это можно увидеть на рис. Н.5.

Атрибут gutter

Промежуток между колонками называют *средником* (*gutter*). По умолчанию Netscape создает средники шириной 10 пикселов. Чтобы их увеличить, присвойте атрибуту `gutter` значение, равное нужной ширине средника в пикселах. Netscape 4 оставит между колонками этот увеличенный промежуток. Оставшееся место будет занято самими колонками.

Рисунок Н.6 иллюстрирует производимое этим действие на колонки. Мы переформатировали текст из нашего примера, применив тег `<multicol cols=3 gutter=50>`. Сравните его с рис. Н.4, на котором средники имеют принятую по умолчанию ширину 10 пикселов.

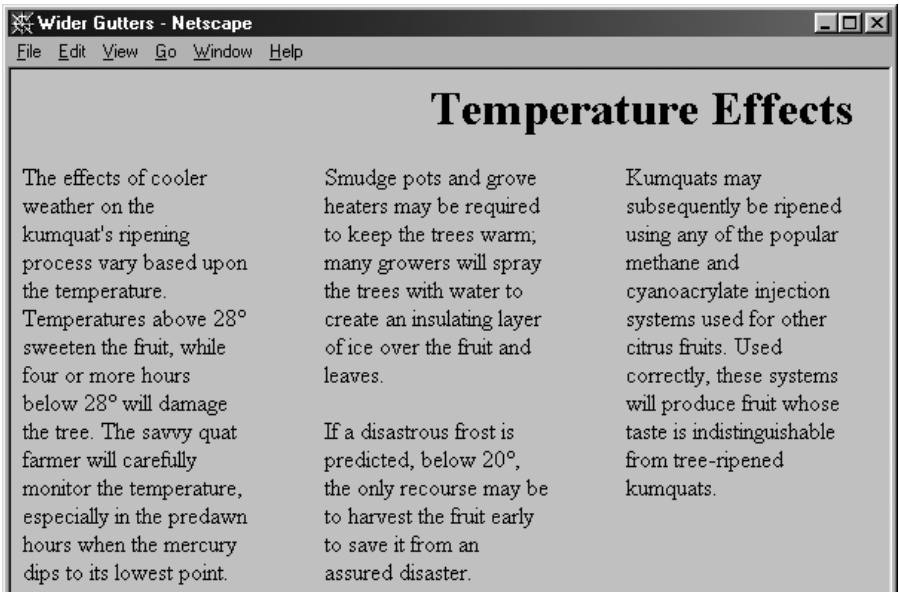

Рис. H.6. *Изменение межстолбцового промежутка при помощи атрибута gutter тега <multicol> (только Netscape 4)*

Атрибут width

Обычно содержимое тега `<multicol>` заполняет все окно броузера. Чтобы колонки располагались в более узкой полосе или чтобы распространить их за пределы окна, используйте атрибут `width`, определяющий совокупную ширину, на которой должно расположиться содержимое тега `<multicol>`. Размеры колонок изменятся так, чтобы колонки вместе со средниками заполнили предоставленное им пространство.[1] Ширина должна указываться как абсолютное число пикселов или как процентное отношение к текущей ширине потока текста.

Рис. H.7 показывает результат вставки `width="75%"` в тег `<multicol>` из нашего примера, в котором средник имеет принятую по умолчанию ширину 10 пикселов.

Если колонки в вашем документе содержат изображения и другие элементы фиксированной ширины, осторожнее уменьшайте их размер. Netscape 4 не будет обходить при выводе текста изображения, вылезающие из своей колонки. Такое изображение просто накроет текст в соседней колонке, испортив документ. Позаботьтесь о том, чтобы

[1] В точности ширина каждой колонки будет $(w - g(n - 1))/n$ пикселов, где w – это значение `width` в теге `<multicol>`, g – это средник (gutter), а n – это число колонок. Таким образом, тег `<multicol cols=3 gutter=10 width=500>` создает колонки шириной 160 пикселов.

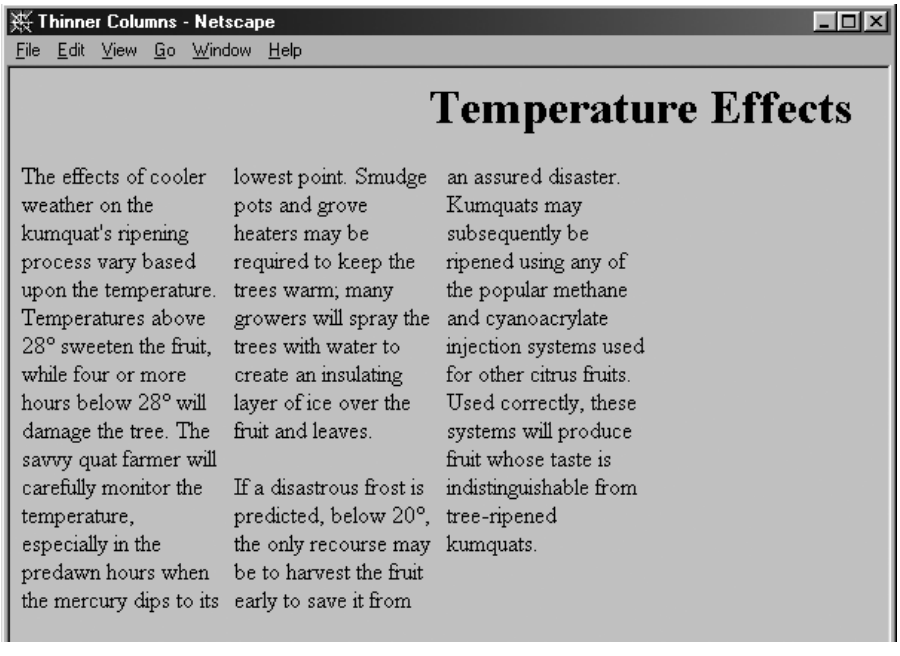

Рис. Н.7. *Изменение ширины колонок в теге <multicol> (только Netscape 4)*

вложенные элементы в колонках были достаточно маленькими, чтобы помещаться в колонку на весьма маленьких дисплеях.

Атрибуты style и class

Атрибут `style` с тегом `<multicol>` используется для встроенного определения стиля содержимого тега. Атрибут `class` позволяет применять к содержимому тега стиль, заранее определенный для данного класса. Значение атрибута `class` – это имя стиля, определенного в таблице стилей на уровне документа или внешним образом. [встроенные стили: атрибут style, 8.1.1] [стилевые классы, 8.3]

Вывод в несколько колонок другими броузерами

Как мы уже отмечали, тег `<multicol>` поддерживает только Netscape версии 4 и более ранних. К счастью, когда другие броузеры встречают тег `<multicol>`, они игнорируют его и выводят содержащийся в нем текст как часть обычного текстового потока, как правило, с небольшим повреждением последующего содержимого.

Единственной проблемой может быть то, что содержимое тега `<multicol>` может слиться с предыдущим потоком текста из-за пропавшего разрыва. Поэтому можно попробовать вставлять перед каждым тегом `<multicol>` тег `<p>`. Netscape 4 не будет возражать, а другие броузеры хотя бы вставят разрыв абзаца перед тем, как вывести ваш текст в одну колонку.

Возможна эмуляция тега <multicol> с использованием таблиц, но результат будет грубым и его будет трудно подстроить под разные броузеры. Чтобы ее осуществить, создайте таблицу с одной строкой и ячейками для каждой колонки. Поместите в каждую ячейку подходящее количество содержимого, чтобы получить сбалансированные колонки. Трудность заключается, разумеется, в этом «подходящем количестве», которое резко варьируется от броузера к броузеру, делая практически невозможным создание колонок, которые бы выглядели привлекательно на различных броузерах.

Если вывод в несколько колонок абсолютно необходим и вы спокойно относитесь к тому, что на не поддерживающих вывод в несколько колонок броузерах текст будет выведен в одну колонку, тогда можно рекомендовать использовать тег <multicol>.

Эффективное использование вывода в несколько колонок

Мы давали вам советы по поводу колонок на протяжении этого раздела. Здесь мы быстро повторим их:

- Используйте небольшое число колонок.
- Не делайте средники слишком широкими.
- Убедитесь, что вложенные элементы, такие как изображения и таблицы, помещаются в колонки на большинстве дисплеев.
- Поставьте перед каждым тегом <multicol> тег <p>, чтобы улучшить вывод ваших документов на других броузерах.
- Избегайте вложения тега <multicol> на глубину больше двух уровней.

Слои

Пропуски и вывод в несколько колонок, помещаемые в обычный поток текста, – это естественное расширение HTML. В Netscape версии 4 были введены слои – новое измерение HTML. Слои (layer) трансформируют поэлементную модель документа в модель, в которой при формировании окончательного документа участвуют элементы, расположенные в разных слоях. К сожалению, слои не поддерживаются броузером Netscape 6, а также ни одной из версий Internet Explorer.[1]

[1] Не последнюю роль в исчезновении слоев, по всей видимости, сыграло то обстоятельство, что вся функциональность слоев реализуется элементом в сочетании с соответствующими установками стиля его отображения. Слои для Netscape Navigator версии 4.x (несовместимость которых с другими броузерами попортила немало крови авторам страниц) приходилось использовать в основном из-за того, что в указанных версиях броузеров была весьма ограниченная поддержка CSS. И когда, наконец, Netscape 6 стал в полной мере поддерживать CSS2, необходимость в слоях отпала как таковая. – *Примеч. науч. ред.*

Слои снабжают дизайнера самым важным из отсутствующих в HTML средством – возможностью абсолютного позиционирования содержимого в окне броузера. Слои позволяют вам создавать самодостаточную единицу содержимого HTML, которую можно позиционировать где угодно в окне броузера, располагая ее под или над другими слоями, и заставить ее появляться или исчезать по вашему желанию. С помощью слоев с легкостью получаются макеты документов, недостижимые при использовании традиционного HTML.

Если представить себе документ как лист бумаги, то о слоях можно думать, как о листах прозрачного пластика, наложенных на документ. Для каждого слоя можно определить содержимое слоя, его положение относительно основного документа и порядок, в котором слои должны накладываться на документ. Слои могут быть прозрачными и непрозрачными, видимыми и скрытыми, представляя бесконечные комбинации вариантов макета.

Тег <layer> (устарел)

Каждый слой содержимого в HTML-документе определяется при помощи тега `<layer>`. О слое можно думать как о миниатюрном HTML-документе, содержимое которого находится между тегами `<layer>` и `</layer>`. Альтернативным образом содержимое слоя может быть получено из другого HTML-документа с применением атрибута `src` тега `<layer>`.

<layer> (устарел)	
Функция:	Определяет в документе слой
Атрибуты:	`above`, `background`, `below`, `bgcolor`, `class`, `clip`, `left`, `name`, `src`, `style`, `top`, `visibility`, `width`, `z-index`
Завершающий тег:	`</layer>`; всегда присутствует
Содержит:	*body_content* (содержимое тела)
Может содержаться в:	*блоке*

Независимо от его происхождения, Netscape 4 форматирует содержание слоя точно так же, как традиционный документ, за исключением того, что результаты этого форматирования содержатся в слое, отдельном от остальной части документа. Расположением этого слоя и его видимостью можно управлять при помощи атрибутов тега `<layer>`.

Слои можно также вкладывать друг в друга. Вложенные слои двигаются вместе с содержащим их слоем и видимы, только если виден их родительский слой.

Атрибут name

Если вы собираетесь создать слой и никогда на него не ссылаться, то нет необходимости называть его. Если же вы собираетесь размещать слои относительно текущего слоя, как мы это делаем далее в этом при-

ложении, или намерены его модифицировать при помощи JavaScript, то необходимо присвоить слою имя при помощи атрибута name. Значением атрибута name должна быть строка текста, начинающаяся с буквы, но не с цифры или другого символа.

После того как слой назван, на него можно ссылаться из любого места документа и изменять его в процессе взаимодействия пользователя с документом. Вот, например, следующий кусочек HTML:

```
<layer name="warning" visibility=hide>
Внимание! Введенный вами параметр имеет недопустимое значение!
</layer>
```

создает слой под именем warning, который изначально скрыт. Если в процессе проверки заполнения формы с применением программы JavaScript будет найдена ошибка и вы захотите вывести предупреждение, то можно использовать оператор:

```
warning.visibility = "show";
```

Netscape 4 тогда сделает слой видимым для пользователя.

Атрибуты left и top

Слой без атрибутов помещается в окне документа так, как если бы он был частью обычного потока содержимого документа. Слои, находящиеся в самом начале документа, помещаются вверху окна Netscape 4, слои, содержащиеся между другими элементами традиционного содержимого документа, помещаются в строку с этим содержимым.

Сила слоев, однако, заключается в том, что можно поместить их в любое место документа. Используйте атрибуты top и left тега <layer> для определения абсолютного положения слоя в образе документа.

Оба атрибута принимают целые значения, равные смещению слоя от левого верхнего края области отображения документа (с координатами (0,0)), измеренному в пикселах. В случае вложенного слоя смещение отсчитывается от расположения родительского слоя. Как и в случае других элементов, размер или положение которых выходят за пределы окна броузера, Netscape предоставляет пользователю полосу прокрутки, позволяющую добраться до элементов слоя, расположенных за пределами текущей области видимости.

Ниже следует простой пример слоев, который расставляет три слоя по диагонали, – это уже то, что трудно вообще сделать и нельзя сделать с такой же точностью, используя традиционный HTML.

```
<layer left=10 top=10>
  Upper left!
</layer>
<layer left=50 top=50>
  Middle!
</layer>
<layer left=90 top=90>
```

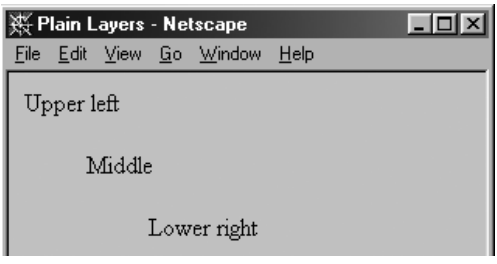

Рис. Н.8. *Простое позиционирование текста при помощи тега <layer>*

```
    Lower right!
  </layer>
```

Результат показан на рис. Н.8.

Можно признать, что этот пример глуповат. Вот пример получше, он создает тень, отбрасываемую заголовком:

```
<layer>
 <layer left=2 top=2>
  <h1><font color=gray>Introduction to Kumquat Lore</font></h1>
 </layer>
 <layer left=0 top=0>
   <h1>Introduction to Kumquat Lore</h1>
 </layer>
</layer>
<h1> </h1>
Early in the history of man, the kumquat played a vital role in
the formation of religious beliefs. Central to annual harvest celebrations
 was the day upon which kumquats ripened. Likened to the sun (<i>sol</i>),
the golden fruit was taken (<i>stisus</i>) from the trees on the day
the sun stood highest in the sky. We carry this day forward even today,
as our summer <i>solstice</i>.
```

На рис. Н.9 показан результат. Рис. Н.10 демонстрирует, что получается со слоями, когда их рассматривают на броузере, отличном от Netscape 4.

Мы использовали несколько трюков при создании эффекта отбрасываемой тени заголовка. Во-первых, Netscape 4 покрывает слои, появившиеся в документе раньше, последующими слоями. Поэтому мы сначала нарисовали серую тень, за которой уже идет настоящий заголовок, так что он оказывается сверху, над тенью. Мы также заключили оба эти слоя в отдельный охватывающий их слой. Теперь положения тени и заголовка определяются относительно контейнера, а не относительно самого документа. Родительский слой, положение которого не указано явно, помещается в поток обычного содержимого, как если бы принадлежал ему, и будет прокручиваться вместе с содержимым документа, так что заголовок будет оставаться на своем месте.

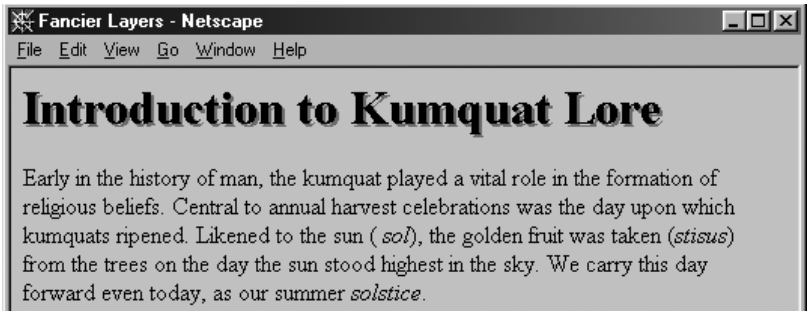

Рис. H.9. *Создание эффекта отбрасываемой тени при помощи нескольких слоев (только Netscape 4)*

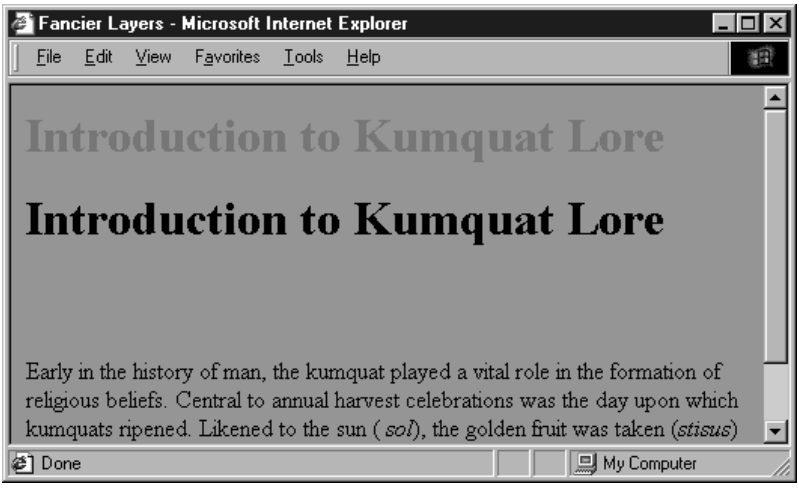

Рис. H.10. *Internet Explorer не поддерживает слои (только Netscape 4)*

Обычное содержимое, однако, по-прежнему будет начинаться с самого верха документа и могло бы скрыться под затейливым заголовком из нашего примера. Чтобы вытолкнуть содержимое из-под нашего заголовка, мы вставили пустой заголовок (содержащий неразрывный пробел – ` `) перед основным текстом нашего документа.

Это заслуживает повторения: обычное содержимое документа, следующее за тегом `<layer>`, размещается прямо под слоем, за которым оно следует в исходном тексте. Этот эффект можно обойти с применением встроенных слоев, описанных в разделе «Тег `<ilayer>` (устарел)» далее в этом приложении.

Атрибуты above, below и z-index

Слои располагаются в трех измерениях, они покрывают некую область страницы и накладываются друг поверх друга и поверх традиционного

содержимого документа. Как мы уже отмечали, обычно слои накладываются друг на друга в порядке их появления в документе – первые слои накрываются следующими в общей для них области отображения.

Можно управлять тем, в каком порядке слои накладываются друг на друга, при помощи атрибутов `above`, `below` и `z-index` тега `<layer>`. Эти атрибуты взаимно исключают друг друга – для каждого слоя можно определить только какой-нибудь один из них.

Значением атрибутов `above` и `below` служит имя другого слоя в этом же документе. Разумеется, можно сослаться только на тот слой, у которого имеется атрибут `name`, значение которого и есть то имя, которое присваивается атрибутам `above` и `below` ссылающегося на него тега `<layer>`. При этом необходимо создать слой прежде, чем ссылаться на него, – нельзя ссылаться на слой, описываемый ниже.

В прямом противоречии с тем, что можно было бы ожидать, Netscape 4 помещает текущий слой под слоем, упоминаемым в атрибуте `above` (над), и над слоем, на который указывает атрибут `below`[1] (под). Ну и ладно. Имейте в виду, что слои должны занимать общее пространство отображения, чтобы можно было заметить какой-нибудь эффект.

Давайте возьмем снова наш пример с отбрасываемой тенью, чтобы проиллюстрировать действия атрибута `above`:

```
<layer>
  <layer name=text left=0 top=0>
    <h1>Introduction to Kumquat Lore</h1>
  </layer>
  <layer name=shadow above=text left=2 top=2>
    <h1><font color=gray>Introduction to Kumquat Lore
    </font></h1>
  </layer>
</layer>
```

Атрибут `above` в слое по имени `shadow` говорит Netscape, что надо поместить слой с тенью так, чтобы слой по имени `text` находился над ним. Результат будет таким, как на рис. Н.9.

В атрибутах `above` и `below` легко запутаться, когда нужно накладывать несколько слоев. Мы считаем, что при помощи атрибута `z-index` легче проследить за тем, какой слой на каком лежит. При помощи атрибута `z-index` определяется порядок, в котором Netscape будет накладывать слои, – слои с большими значениями `z-index` лежат выше слоев с меньшими значениями `z-index`.

Например, чтобы создать наш заголовок с тенью при помощи атрибута `z-index`, можно написать:

```
<layer>
```

[1] От этого не станет легче, но можно воображать, что атрибуты `above` и `below` реализовывали в спешке.

```
    <layer left=0 top=0 z-index=2>
      <h1>Introduction to Kumquat Lore</h1>
    </layer>
    <layer left=2 top=2 z-index=1>
      <h1><font color=gray>Introduction to Kumquat Lore
      </font></h1>
    </layer>
  </layer>
```

И снова результат будет тем же, что и на рис. H.9. По умолчанию Netscape расположил бы второй слой – серый в нашем случае – поверх первого слоя. Но поскольку мы присвоили атрибуту `z-index` серого слоя меньшее значение, он будет находиться под первым слоем.

Значения атрибута `z-index` не обязаны быть последовательными, хотя они должны быть целыми, чтобы мы могли использовать 99 и 2 вместо 2 и 1 соответственно и получить тот же результат. Не обязательно также определять `z-index` для всех слоев, занимающих общую область отображения. Достаточно сделать это только для тех, которые нужно поднять или опустить относительно других слоев. Имейте в виду, однако, что порядок следования слоев может оказаться запутанным, если все слои не были проиндексированы.

Какой, например, порядок следования цветов вы предскажете, если Netscape 4 выведет следующую последовательность слоев:

```
<layer left=0 top=0 z-index=3>
  <h1><font color=red>Introduction to Kumquat Lore
  </font></h1>
</layer>
<layer left=4 top=4>
  <h1><font color=green>Introduction to Kumquat Lore
  </font></h1>
</layer>
<layer left=8 top=8 z-index=2>
  <h1><font color=blue>Introduction to Kumquat Lore
  </font></h1>
</layer>
```

Дайте себе орден, если вы сказали, что зеленый (green) заголовок будет находиться над красным (red), который будет находиться над синим (blue). Почему? Потому что красный заголовок имеет меньший приоритет по отношению к зеленому, в силу порядка появления в записи, а мы подложили синий слой под красный, дав ему меньшее значение атрибута `z-index`. Netscape 4 отображает слои, для которых определен `z-index`, в соответствии с определенным им порядком, а слои без атрибута `z-index` – в порядке их появления в документе. Ранжирование, основанное на порядке появления, применяется также к слоям, имеющим равные значения `z-index`. Если вы вкладываете слои, все слои одного уровня вложения упорядочиваются в соответствии со значением атрибута `z-index`. Затем эта группа как один слой находит свое

место на следующем уровне вложенности. Короче говоря, слои, вложенные в один слой, не могут перемежаться слоями другого уровня.

Для примера рассмотрим нижеследующее описание вложенных слоев, в которых для краткости опущены содержимое и закрывающие теги (отступы обозначают вложенность тегов):

```
<layer name=a z-index=20>
  <layer name=a1 z-index=5>
  <layer name=a2 z-index=15>
<layer name=b z-index=30>
  <layer name=b1 z-index=10>
  <layer name=b2 z-index=25>
  <layer name=b3 z-index=20>
<layer name=c z-index=10>
```

Слои a, b и c находятся на одном уровне, слои a1 и a2 вложены в a, а слои b1, b2 и b3 вложены в b. Хотя значения z-index, как может показаться, предписывают Netscape 4 перемешать между собой слои, вложенные в разные слои верхнего уровня, на деле порядок слоев снизу вверх будет таким: c, a, a1, a2, b, b1, b3, b2.

Если два слоя вложены в один общий слой и значения атрибута z-index у них равны, слой, определенный позже, будет находиться выше слоя, определенного раньше.[1]

Атрибуты background и bgcolor

Подобно соответствующим атрибутам тега <body>, атрибуты bgcolor и background позволяют определить для Netscape 4 цвет фона и фоновое изображение слоя соответственно.[2] По умолчанию фон слоя прозрачный, позволяющий другим слоям просвечивать сквозь него.

Атрибут bgcolor принимает в качестве значения название цвета или RGB-триплет, описываемые в приложении G. Если этот атрибут определен, Netscape делает весь слой непрозрачным, закрашивая его фон указанным цветом. Этот атрибут удобен для создания цветных прямоугольников за текстом с целью подчеркнуть его значение или привлечь к нему внимание. Такой слой, однако, скроет все нижележащие слои, включая обычное HTML-содержимое.[3]

---

[1] Это относится, конечно, только к слоям, лежащим внутри одного родительского тега.

[2] Отметьте, что можно управлять цветом фона, как и многими другими характеристиками отображения не только одного тега, но и всех тегов <layer> в документе, используя таблицы стилей. См. раздел 14.3.1.9.

[3] Под «обычным HTML-содержимым» здесь подразумевается то, что не включено ни в один слой. Вообще говоря, модель слоев Netscape подразумевает, что обычное содержимое тела документа – это тоже слой, который автоматически создается броузером при отображении содержимого тела документа. – *Примеч. науч. ред.*

Атрибут background принимает в качестве значения URL изображения. Изображение размножается вправо и вниз, и его экземпляры покрывают в совокупности всю область, занимаемую слоем. Если некоторые части изображения прозрачны, эти части слоя тоже будут прозрачны и нижележащие слои будут сквозь них видны.

Если определить оба атрибута, то цвет фона будет виден через прозрачные участки фонового изображения. А слой в целом будет непрозрачен.

Атрибут background полезен, чтобы задать текстуру в качестве фона для текста, но он никуда не годится, если нужно поместить текст на фоне определенного изображения. Поскольку размер слоя определяется его содержимым, а не фоновым изображением, использование изображения в качестве фона ведет к тому, что оно будет или обрезано, или размножено, в зависимости от размера текста. Чтобы поместить текст поверх изображения, используйте вложение одного слоя в другой:

```
<layer>
  <img src="sunset.gif">
  <p>
  <layer top=75>
    <h2 align=center>And they lived happily ever after...</h2>
  </layer>
</layer>
```

Netscape 4 выделит пространство для целого изображения на внешнем слое. Вложенный слой занимает то же пространство, за исключением того, что мы сдвинули его на 75 пикселов, чтобы лучше выровнять текст по отношению к изображению. Результат показан на рис. H.11.

Рис. H.11. *Размещение текста поверх изображения с применением слоев (только Netscape 4)*

Атрибут visibility

По умолчанию слои обычно видимы (хотя и не слышимы). Это можно изменить, присвоив атрибуту visibility одно из значений: show, hide или inherit. Как и следовало ожидать, значение show делает слой видимым, hide скрывает его, а inherit явным образом объявляет, что слою следует унаследовать родительскую видимость. По умолчанию значе-

нием этого атрибута является inherit. Слои, не вложенные в другие слои, считаются детьми основного документа, который всегда является видимым. Таким образом, не вложенные слои без атрибута visibility в начале являются видимыми.

Мало смысла в том, чтобы прятать слои, если только вы не хотите показать их впоследствии. В общем случае этот атрибут используется только тогда, когда в документ включаются некие JavaScript-программы, которые открывают изначально скрытые слои в результате взаимодействия с пользователем. [обработчики событий JavaScript, 12.3.3]

Скрытые слои не загораживают нижележащих слоев. Наоборот, о скрытом слое лучше всего думать как о прозрачном слое. Единственный способ скрыть содержимое основного документа – это положить на него непрозрачный слой. Чтобы показать скрытое содержимое, скройте непрозрачный слой, открыв лежащее под ним содержимое.

Атрибут width

Слои имеют ровно такую величину, чтобы вместить свое содержимое. Начальная ширина слоя полагается равной расстоянию от точки, в которой слой создается в потоке текста, до правого края. Netscape 4 затем форматирует содержимое слоя по этой ширине и делает слой достаточно высоким, чтобы в нем поместилось все его содержимое. Если содержимое слоя оказывается меньше, чем начальная ширина, то ширина слоя уменьшается до этого меньшего значения.

Можно явным образом установить ширину слоя при помощи атрибута width. Значение этого атрибута устанавливает ширину слоя в пикселах или в процентах от охватывающего слоя. Как можно было ожидать, Netscape 4 затем устанавливает высоту слоя, основываясь на размере его содержимого, учитывая автопереносы текста в пределах указанной ширины. Если элементы в слое, такие как изображения, нельзя разбить на несколько строк и они выступают за правый край слоя, только часть элемента будет показана. Остаток будет обрезан по краю слоя и не будет виден. Это похоже на поведение изображения в окне основного документа. Если изображение продолжается за пределами окна броузера, отображается только часть изображения. В отличие от окна броузера, однако, слои не могут выставить полос прокрутки, позволяющих пользователю заглянуть за край слоя.

Атрибут src

Содержимое слоя не ограничивается тем, что можно набрать между тегами <layer> и </layer>. Можно также указать и автоматически загрузить в слой содержимое другого документа при помощи атрибута src. Значением атрибута src является URL документа, заключающего в себе содержимое слоя.

Отметьте, что документ, на который ссылается слой в своем атрибуте src, *не* может быть полноценным HTML-документом. В частности, он

не может содержать тег <head> или <body>, хотя другое HTML-содержимое допустимо.

Можно сочетать содержимое тега <layer> с содержимым, полученным из другого файла при помощи атрибута src. В этом случае содержимое файла помещается в слой первым, и за ним уже следует содержимое тега. Если атрибут src используется без дополнительного встроенного содержимого, все равно нужно написать закрывающий тег </layer>, чтобы закончить определение слоя.

Атрибут src впервые предоставил возможность включения содержимого одного HTML-документа в другой. Прежде, чтобы вставить содержимое одного документа в другой, нужно было воспользоваться возможностью, реализуемой со стороны сервера, прочитать другой файл и вставить его в правильном месте.[1] Поскольку слои позиционируются по умолчанию в той точке потока содержимого, в которой они определяются, – включение в документ другого файла – это очень простое дело:

```
...предыдущее содержимое
<layer src="boilerplate">...</layer> ...последующее содержимое...
```

Поскольку слой выводится как отдельный HTML-документ, содержимое включаемого файла не вольется в окружающий текст, и все будет выглядеть так, как если бы вы вставили этот текст, заключив его в тег или другой блочный элемент HTML.

Атрибут clip

Обычно пользователи видят слой целиком, если только его не загораживает накрывающий слой. При помощи атрибута clip (обрезать) вы можете замаскировать некоторые части слоя, оставив для обозрения только прямоугольную область внутри слоя. Часть слоя за пределами вырезанной области делается прозрачной, позволяя видеть все, что под ней.

Значениями атрибута clip являются пара или четверка целых чисел, разделенных запятыми и определяющих смещения относительно текущего слоя, соответствующие по порядку левой, верхней, правой и нижней границам обрезаемой области. Если в качестве значения атрибута предлагается пара чисел, они принимаются соответствующими правому и нижнему краям видимой области, и Netscape предполагает, что верхнее и левое значения равны нулю. То есть clip="75,100" эквивалентно clip="0,0,75,100".

Атрибут clip удобен, чтобы скрыть часть слоя или для реализации динамических эффектов с применением функций JavaScript.

---

[1] Эта технология называется SSI (Server Side Include – подключение файлов на сервере). Существенным плюсом SSI является то, что такой способ не зависит от типа броузера, поскольку формирование составных страниц происходит на сервере. Поскольку детали реализации SSI зависят от сервера, за более подробной информацией обращайтесь к документации по вашему веб-серверу. – *Примеч. науч. ред.*

Атрибуты style и class

Используйте атрибут `style` с тегом `<layer>` для встроенного в тег определения стиля содержимого слоя. Атрибут `class` позволяет применять к содержимому тега стиль, заранее определенный для данного класса слоев. Значение атрибута `class` – это имя стиля, определенного в таблице стилей на уровне документа или внешним образом. Соответственно можно принять решение использовать таблицы стилей вместо частых и избыточных атрибутов `bgcolor` для определения цвета фона для всех слоев в документе или для определенного класса слоев. [встроенные стили: атрибут style, 8.1.1] [стилевые классы, 8.3]

Тег <ilayer> (устарел)

Хотя управлять положением слоя можно при помощи атрибутов `top` и `left`, смещающих слой по отношению ко всей области отображения документа, Netscape 4 предоставляет еще один тег `<ilayer>`, позволяющий позиционировать отдельные слои по отношению к потоку содержимого совершенно так же, как встроенные изображения.

<ilayer> (устарел)	
Функция:	Определяет в потоке текста встроенный слой
Атрибуты:	`above`, `background`, `below`, `bgcolor`, `class`, `clip`, `left`, `name`, `src`, `style`, `top`, `visibility`, `width`, `z-index`
Завершающий тег:	`</ilayer>`; присутствует всегда
Содержит:	*body_content* (содержимое тела)
Может содержаться в:	*тексте*

Тег `<ilayer>` создает слой, который занимает пространство в содержащем его потоке текста. Последующее содержимое располагается вслед за занятым тегом `<ilayer>` пространством. Это контрастирует с поведением тега `<layer>`, который создает слой поверх содержащего его текста, позволяя последующему содержимому располагаться под только что созданным слоем.

Тег `<ilayer>` устраняет необходимость в охватывающем безатрибутном теге `<layer>`, который служит для того, чтобы размещать в нем слои, позиционируемые относительно предыдущего содержимого документа, как мы это делали в большинстве примеров в предыдущих разделах этого приложения. Атрибуты тега `<ilayer>` такие же, как у тега `<layer>`.

Атрибуты top и left

Единственными атрибутами, которые отличают поведение тега `<ilayer>` от поведения его товарища `<layer>`, являются атрибуты `left` и `top` – Netscape 4 выводит содержимое тега `<ilayer>` прямо в окружающем потоке текста, смещая его на значения атрибутов `top` и `left` от текущей

позиции, а не от верхнего левого угла всего документа, как в случае тега <layer>. Netscape 4 принимает также отрицательные значения этих атрибутов, позволяя вам сдвигать содержимое вверх и налево по отношению к потоку содержимого.

К примеру, чтобы поднять, опустить или сдвинуть в сторону слова в текущей строке, можно написать:

```
This <ilayer top=4>word</ilayer> is shifted down, while
this <ilayer left=10>one</ilayer>
is shifted over. With a negative
 value, words can be moved <ilayer top=-4>up</ilayer> and to
 the
<ilayer left=-10>left</ilayer>.
```

Результат показан на рис. H.12. Отметьте, как сдвинутые слова наслаиваются и загораживают окружающий текст. Netscape 4 не предпринимает никаких мер, чтобы выделить пространство для сдвинутых элементов, они просто перемещаются на новое место на странице.

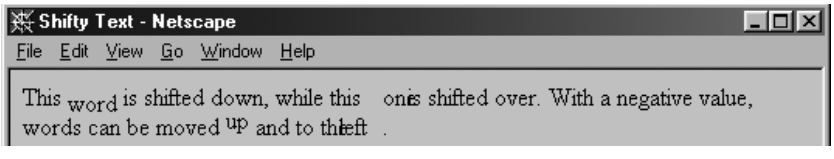

Рис. H.12. *Перемещение встроенных слоев относительно окружающего текста (только Netscape 4)*

Сравнение тегов <layer> и <ilayer>

Все, что можно создать при помощи обычного слоя, может использоваться внутри встроенного слоя. Имейте только всегда в виду, что смещения в атрибутах top и left будут отсчитываться относительно области, выделяемой для содержимого тега <ilayer>, а не относительно области отображения документа. В соответствии с этим используйте тег <ilayer> для позиционирования содержимого по отношению к основному потоку HTML в документе, а тег <layer> – для точного размещения элементов и содержимого по отношению к области отображения документа.

Также (по счастью) Netscape 4 не различает теги <ilayer> и <layer> в том, что касается порядка их наложения. Можно объявить, что слой <ilayer> лежит под слоем <layer>, используя атрибуты name и above:

```
<layer name=me>Я сверху</layer>
<ilayer above=me>Я снизу</ilayer>
```

Подобным образом можно переупорядочить наложение абсолютных и встроенных слоев (там, где они перекрываются), присваивая различным элементам разные значения атрибута z-index. Применимы также и правила вложения.

Алфавитный указатель

Специальные символы

A

B

C

D

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

А

Б

В

Г

Д

З

И

К

Л

М

Н

О

П

P

C

T