БОЛЬШАЯ КНИГА

CSS

Дэвид Макфарланд

2-Е ИЗДАНИЕ



O'REILLY®

ПИТЕР®



CSS

Second Edition



David Sawyer McFarland

POGUE PRESS™
O'REILLY®

Beijing • Cambridge • Farnham • Köln • Paris • Sebastopol • Taipei • Tokyo

БОЛЬШАЯ КНИГА

CSS



Дэвид Макфарланд

ПИТЕР®

Москва · Санкт-Петербург · Нижний Новгород · Воронеж
Ростов-на-Дону · Екатеринбург · Самара · Новосибирск
Киев · Харьков · Минск
2012

ББК 32.988.02
УДК 004.738.5
М17

Макфарланд Д.
М17 Большая книга CSS. 2-е изд. — СПб.: Питер, 2012. — 560 с.: ил. — (Серия «Бестселлеры O'Reilly»).

ISBN 978-5-459-01560-7

Современные технологии веб-дизайна активно развиваются. Если раньше процесс графического оформления сайта представлял собой скрупулезную работу в HTML, то сегодня CSS позволяет без лишних усилий создавать действительно уникальные, удобные и функциональные сайты. CSS (каскадные таблицы стилей) — это технология описания внешнего вида документа с помощью языка разметки, позволяющая легко и быстро задавать параметры графического отображения веб-страницы. Это не просто полезный инструмент для «украшения»: используя CSS, можно в полном объеме управлять внешним видом сайтов (от шрифта и цвета до макета страницы). Данная книга доступно и подробно объясняет основы этого мощного инструмента веб-дизайна и помогает научиться как созданию новых, так и апгрейду уже существующих сайтов. Во втором издании книги материал в значительной степени переработан с учетом самых современных интернет-технологий, а также актуальных и анонсируемых обновлений CSS.

ББК 32.988.02
УДК 004.738.5

Все права защищены. Никакая часть данной книги не может быть воспроизведена в какой бы то ни было форме без письменного разрешения владельцев авторских прав.

Информация, содержащаяся в данной книге, получена из источников, рассматриваемых издательством как надежные. Тем не менее, имея в виду возможные человеческие или технические ошибки, издательство не может гарантировать абсолютную точность и полноту приводимых сведений и не несет ответственности за возможные ошибки, связанные с использованием книги.

ISBN 978-5-459-01560-7
ISBN 978-0596802448 (англ.)

© 2009 David Sawyer McFarland
© Перевод на русский язык ООО Издательство «Питер», 2012
© Издание на русском языке, оформление
ООО Издательство «Питер», 2012

Краткое содержание

Об авторе 12
О творческой команде 13
Благодарности 14
Введение 15
От издательства 28

Часть 1. Основы CSS

Глава 1. CSS — новый подход к HTML 30
Глава 2. Создание стилей и таблиц стилей 44
Глава 3. Селекторы: определение элементов стилизации 63
Глава 4. Механизм наследования стилей 96
Глава 5. Управление сложной структурой стилей: каскадность 106

Часть 2. Применение CSS

Глава 6. Форматирование текста 126
Глава 7. Поля, отступы, границы 165
Глава 8. Добавление графики на веб-страницы 203
Глава 9. Приводим в порядок навигацию сайта 243
Глава 10. Форматирование таблиц и форм 290

Часть 3. Макет страницы

Глава 11. Введение в разметку CSS 318
Глава 12. Разметка страницы на основе плавающих элементов 329
Глава 13. Позиционирование элементов на веб-странице 376
Глава 14. CSS для распечатываемых веб-страниц 417
Глава 15. Совершенствуем навыки в CSS 437
Глава 16. CSS 3 — на гребне волны 460

Приложения

Приложение 1. Справочник свойств CSS 482
Приложение 2. CSS в Dreamweaver CS4 510
Приложение 3. Ресурсы по CSS 541

Алфавитный указатель 549

Оглавление

Об авторе 12

О творческой команде 13

Благодарности 14

Введение 15

Как работает CSS 15
Преимущества CSS 16
Что необходимо знать 16
HTML: структура языка 17
Как работают HTML-теги 17
XHTML: современный HTML 19
HTML 5 — новый виток 20
Программное обеспечение для CSS 21
Об этой книге 22
Основные разделы книги 24
Примеры программного кода на CSS 24
О сайте MissingManuals.com 25
Основная терминология 25
О стрелках 26
Соглашения, использованные в данной книге 26
Технология Safari® 27

От издательства 28

Часть 1. Основы CSS

Глава 1. CSS — новый подход к HTML 30
HTML: прошлое и настоящее 30
Написание HTML-кода для CSS 33
Важность doctype 40
Долгожданный Internet Explorer 8 42

Глава 2. Создание стилей и таблиц стилей 44
Что такое стиль 45
Понимание таблиц стилей 47
Внутренние таблицы стилей 48
Внешние таблицы стилей 49
Обучающий урок: создание первых стилей 52

Глава 3. Селекторы: определение элементов стилизации 63
Селекторы типов: дизайн страницы 63
Селекторы классов: точное управление 65
ID-селекторы: определение элементов веб-страниц 67
Стилизация групп тегов 70
Стилизация вложенных тегов 71
Псевдоклассы и псевдоэлементы 75
Другие селекторы 80
Обучающий урок: примеры использования селекторов 85

Глава 4. Механизм наследования стилей 96
Что такое наследование? 96
Упрощение таблиц стилей через наследование 98
Ограничения наследования 98
Обучающий урок: наследование 100

Глава 5. Управление сложной структурой стилей: каскадность 106
Каскадность стилей 106
Особенности механизма каскадности: какие стили имеют преимущество . . 111
Управление каскадностью 114
Начинаем с чистого листа 117
Обучающий урок: механизм каскадности в действии 119

Часть 2. Применение CSS

Глава 6. Форматирование текста 126
Стилизация текста 126
Установка размера шрифта 133
Форматирование символов и слов 138
Форматирование абзацев текста 142
Стилизация списков 149
Обучающий урок: форматирование текста в действии 154

Глава 7. Поля, отступы, границы 165
Понятие блочной модели 165
Управление размерами полей и отступов 167
Добавление границ 175
Установка цвета фона 179
Определение параметров высоты и ширины 180
Управление обтеканием содержимого плавающих элементов 185
Обучающий урок: поля, фоновые параметры, границы 191
Двигаемся дальше 202

Глава 8. Добавление графики на веб-страницы 203
CSS и тег <img> 203
Фоновые изображения 204
Управление повтором фоновых изображений 209
Позиционирование фоновых изображений 210
Сокращенный вариант свойства background 216
Обучающий урок 1: совершенствуем изображения 218
Обучающий урок 2: создание фотогалереи 224
Обучающий урок 3: использование фоновых изображений 231
Двигаемся дальше 241

Глава 9. Приводим в порядок навигацию сайта 243
Выборка стилизуемых ссылок 243
Стилизация ссылок 246
Создание панелей навигации 253
Современные методы стилизации ссылок 263
Обучающий урок 1: стилизация ссылок 272
Обучающий урок 2: создание панели навигации 278

Глава 10. Форматирование таблиц и форм 290
Правильное использование таблиц 290
Создание стилей для таблиц 293
Создание стилей для форм 299
Обучающий урок 1: создание стилей для таблиц 304
Обучающий урок 2: создание стилей для форм 311

Часть 3. Макет страницы

Глава 11. Введение в разметку CSS 318
Типы разметок веб-страницы 318
Как работает CSS-разметка 320
Стратегии разметки 323

Глава 12. Разметка страницы на основе плавающих элементов 329
Использование плавающих элементов в разметках 332
Преодоление проблем перемещения 342
Обработка ошибок в Internet Explorer 6 353
Обучающий урок 1: разметки с множеством столбцов 360
Обучающий урок 2: разметка с отрицательными полями 368

Глава 13. Позиционирование элементов на веб-странице 376
Как работают свойства позиционирования 376
Мощные стратегии позиционирования 390
Обучающий урок: позиционирование элементов страницы 403

Глава 14. CSS для распечатываемых веб-страниц 417
Как работают аппаратно-зависимые таблицы стилей 417
Как добавлять аппаратно-зависимые таблицы стилей 420
Создание таблиц стилей для печати 421
Обучающий урок: создание таблицы стилей для печати 429

Глава 15. Совершенствуем навыки в CSS 437
Добавление комментариев 437
Организация стилей и таблиц стилей 438

Устранение столкновений стилей в браузере 446
Использование селекторов потомков 450
Управление браузером Internet Explorer 455

Глава 16. CSS 3 — на гребне волны 460
Обзор CSS 3 461
Селекторы в CSS 3 462
Прозрачность 467
RGBA-цвет 468
Тень для текста 472
Свобода для шрифтов 474
Генерируемое содержимое страницы 477

Приложения

Приложение 1. Справочник свойств CSS 482

Приложение 2. CSS в Dreamweaver CS4 510

Приложение 3. Ресурсы по CSS 541

Алфавитный указатель 549

Об авторе



Дэвид Сойер Макфарланд (David Sawyer McFarland) является президентом Sawyer McFarland Media, Inc., компании по обучению и разработке интернет-приложений в Портленде, штат Орегон. Он создает сайты с 1995 года: именно тогда Дэвид разработал свой первый проект — онлайн-журнал для специалистов в области коммуникаций. Он работал веб-мастером в Калифорнийском университете в Беркли и в Центре мультимедийных исследований Беркли (Berkeley Multimedia Research Center), а также участвовал в проектировании и создании огромного количества сайтов для всевозможных клиентов, включая Macworld.com.

Кроме всего прочего, Дэвид является писателем, тренером и инструктором. Он преподавал веб-дизайн в Высшей школе журналистики в Беркли, Центре электронного искусства (Electronic Art), Колледже искусств, Центре новой прессы и Государственном университете Портленда. Им написаны статьи о Сети для журналов *Practical Web Design*, *MX Developer's Journal* и *Macworld*, а также для портала CreativePro.com.

Дэвид также является автором книг *Dreamweaver: The Missing Manual* и *JavaScript: The Missing Manual*.

С автором вы можете связаться по электронной почте: missing@sawmac.com (если же вам нужна техническая помощь, то обращайтесь к ресурсам, указанным в приложении 3).

О творческой команде

Нэн Барбер (Nan Barber) (редактор) начала работать с серией Missing Manual еще в прошлом тысячелетии. Живет в Массачусетсе вместе со своим мужем и является владелицей компьютера Macintosh G4. Ее электронный адрес: nanbarber@oreilly.com.

Нелли Маккессон (Nellie McKesson) (выпускающий редактор) — проживает в Бригтоне. Коротает свободное время, играя в музыкальной группе Dr. & Mrs. Van der Trampp (http://mysapce.com/drmrsvandertrampp) и делая футболки для своих друзей (http://mattsaundersbynellie.etsy.com). Ее электронный адрес: nellie@oreilly.com.

Марсия Симмонс (Marcia Simmons) (литературный редактор) — писатель и редактор. Проживает в Сан-Франциско. В собственном блоге Мария рассказывает о технологических новинках и делится рецептами коктейлей (www.smartkitty.org).

Анджела Говард (Angela Howard) (составитель алфавитного указателя) — занимается составлением указателей более 10 лет, по большей части для компьютерной литературы, но время от времени интересуется и другими темами: путешествиями, альтернативной медициной и леопардовыми ящерицами. Живет в Калифорнии вместе с мужем, дочерью и двумя кошками.

Тони Раско (Tony Ruscoe) (технический редактор) — веб-разработчик. Живет в Шеффилде, Англия. Первая его программа была написана для компьютера ZX Spectrum на языке BASIC еще в середине 1980-х годов. С 1997 года он занимается разработкой сайтов и других веб-приложений, используя в ходе работы различные технологии. Сейчас он ведет свой персональный сайт http://ruscoe.net/ и сайт, посвященный исследованиям: http://ruscoe.name/.

Кристофер Шмидт (Christopher Schmitt) (технический редактор) — автор множества книг по дизайну и цифровым изображениям, включая книгу *CSS Cookbook*. Пишет статьи для журнала *New Architect* и сайтов A List Apart, Digital Web и Web Reference. Кристофер является основателем небольшой издательской компании, которая также занимается дизайном и была удостоена за это награды. Кроме того, Крис соведущий Adobe Task Force — проекта по стандартизации Сети (WaSP, Web Standards Project). Его сайт: http://www.cristopherschmitt.com/.

Благодарности

Большое спасибо всем, кто помогал в работе над этой книгой, включая моих студентов, которые оценивали ее с позиции новичков. Благодарю своих технических редакторов, Кристофера Шмидта и Тони Раско, которые предостерегли меня от сбивающих с толку ошибок, а также Зое Гилленуотер (Zoe Gillenwater) — за ее бесценный вклад в первое издание книги. Кроме того, мы все в долгу перед веб-дизайнерами, которые стали новаторами, используя CSS с творческим подходом, и поделились своими наработками в сообществе, посвященном веб-дизайну.

Наконец, спасибо Дэвиду Погу (David Pogue), чей неувядающий энтузиазм и выносливость ободряли меня; Нэн Барбер — за очистку моих рукописей и прочую помощь в написании этой книги; моей жене Сколле (Scholle) — за любовь и поддержку; моему сыну Грэхему (Graham), который убеждал меня в том, что я закончу эту книгу гораздо быстрее, если в каждой главе буду писать «Тра-ля-ля!»; моей чудесной дочке Кейт (Kate), чья улыбка всегда меня воодушевляла, и всей моей семье: маме, Дугу (Doug), Мари (Mary), Дэвиду (David), Марисе (Marisa), Тессе (Tessa), Филис (Phyllis), Лес (Les), Дел (Del), Патриции (Patricia) и Майку (Mike).

Введение

Каскадные таблицы стилей, или Cascading Style Sheets (CSS), обеспечивают творческую свободу в разметке и дизайне веб-страниц. Пользуясь ими, вы сможете украсить текст страниц привлекательными заголовками, буквицами, рамками, как в красочных глянцевых журналах.

Можно точно разместить и позиционировать изображения, создать столбцы и баннеры, выделить текстовые ссылки динамическими эффектами.

Вы думаете, что все это довольно сложно? Напротив! Цель CSS как раз и состоит в том, чтобы упростить процесс моделирования. На страницах этой книги вы изучите основы CSS.

Как работает CSS

Если вы пользовались стилями в программах обработки текста, например Microsoft Word, или разметки веб-страниц, например Adobe InDesign, то язык CSS покажется знакомым. *Стиль* — это правило, описывающее форматирование отдельного фрагмента веб-страницы. *Таблица стиля* — ряд определений стилей.

CSS *отличается* от HTML тем, что это совершенно другой язык. HTML структурирует документ, упорядочивая информацию в заголовки, абзацы, маркированные списки и т. д., в то время как CSS тесно взаимодействует с браузером, чтобы оформление HTML-документа имело совершенный вид. Другими словами, язык HTML отвечает за *содержание*, а CSS — за *внешний вид* веб-страниц.

Например, вы могли бы использовать HTML, чтобы превратить фразу в заголовок, отделяя его от содержимого страницы, но лучше использовать CSS для форматирования заголовка, скажем, большим полужирным красным шрифтом с позиционированием на расстоянии 50 пикселов от левого края окна. CSS требуется повсеместно для изменения и улучшения отображения HTML.

Можно также создавать стили специально для работы с изображениями. Например, с помощью стилей можно выровнять изображение по правому краю веб-страницы, поместить его в цветную рамку, отделить от окружающего текста промежутком 50 пикселов.

Создав стиль один раз, можно применять его к текстовым фрагментам, изображениям, заголовкам и любым другим элементам страницы сколько угодно. Например, вы можете выделить абзац текста и применить стиль, изменяющий размер, цвет и шрифт текста.

Можно также создать стили для определенных HTML-тегов так, чтобы, например, все заголовки первого уровня (теги `<h1>`) на вашем сайте были отображены в одинаковом стиле, независимо от того, где они размещены.

Преимущества CSS

До появления CSS дизайнеры веб-страниц были ограничены возможностями разметки и оформления языка HTML. И если вы занимались серфингом в Интернете в 1995 году, то уясните разницу в возможностях CSS и HTML. Читая дальше эту книгу, вы поймете, что HTML все еще является основой создания страниц в Сети, но это отнюдь не средство формирования их дизайна. Безусловно, HTML обеспечивает простейшее форматирование текста, изображений, таблиц и других элементов страниц и оформитель может придать им прекрасный внешний вид. Но, как правило, в результате веб-страницы получаются громоздкими и медленно загружаются.

Таблицы стилей CSS, напротив, имеют следующие преимущества.

- Больше возможностей форматирования, нежели в HTML. В CSS вы можете форматировать абзацы по мере их появления в тексте (например, с абзацным отступом и с произвольным интервалом между абзацами) и изменять межстрочный интервал (расстояние между двумя соседними строками текста в абзаце).
- При использовании CSS вы решаете, каким образом добавить фоновое изображение на страницу. Оно может отображаться в виде неперекрывающейся мозаики, повторяться и т. д.
- Еще более значимо то, что стили CSS занимают намного меньше места, чем форматирующие команды HTML. Например, тег `<font>`. Применяя CSS, вы можете уменьшить размер веб-страниц. В результате они будут стильно выглядеть и быстрее загружаться.
- Стилевые таблицы также облегчают обновление сайта. Можно собрать все стили в единственный внешний файл и связать его со всеми страницами сайта. Когда вы редактируете стиль, изменения моментально затрагивают все элементы страниц сайта, где есть ссылка на описанный в таблице стиль. Вы можете полностью изменить внешний вид путем редактирования единственного файла таблицы стилей.

ПРИМЕЧАНИЕ

HTML уже настолько умудрен опытом и преуспел в пренебрежении требованиями Консорциума Всемирной паутины — World Wide Web Consortium (W3C), ответственного за определение стандартов WWW, что большое количество тегов, используемых исключительно для форматирования внешнего вида HTML (тег <font>, например), уже устарело.

Полный список устаревших тегов смотрите в Интернете по адресу www.codehelp.co.uk/html/deprecated.html.

Что необходимо знать

Эта книга предполагает, что вы уже знакомы с языком HTML (и, возможно, имеете небольшой опыт работы с CSS). Подразумевается, что вы создали пару сайтов (или по крайней мере несколько веб-страниц) и знакомы с основными тегами, такими как `<html>`, `<p>`, `<h1>`, `<table>` и т. д., составляющими основу языка гипертекстовой разметки документов. CSS бесполезен без HTML, поэтому, чтобы продол-

жать изучение CSS, вы должны знать, как создать простейшую веб-страницу с использованием основных HTML-тегов.

Если вы раньше создавали веб-страницы на HTML, но чувствуете, что знания требуется освежить, вам поможет следующий раздел книги.

ПРИМЕЧАНИЕ

Если вы только знакомитесь с HTML и возможностями применения его на практике, то посетите сайты бесплатного обучения: HTML Dog (www.htmldog.com/guides/htmlbeginner/) и W3Schools (www.w3schools.com/html/). Если вам больше нравится читать книги, обратитесь к руководствам по созданию сайтов: Creating Web Sites: The Missing Manual Мэтью Макдональда (Matthew MacDonald) или Head First HTML with CSS & XHTML Элизабет Фриман (Elisabeth Freeman) и Эрика Фримана (Eric Freeman) издательства O'Reilly.

HTML: структура языка

В языке гипертекстовой разметки HTML используются простые команды, именуемые *тегами*, для определения различных частей — фрагментов. Ниже приведен код HTML простой веб-страницы:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/
html4/strict.dtd">
<html>
  <head>
    <title>Это заголовок веб-страницы</title>
  </head>
  <body>
    <p>А это содержимое страницы</p>
  </body>
</html>
```

Конечно, пример очень простой, но демонстрирует все основные элементы, необходимые обычной веб-странице. В нем вы заметите то, что называется *объявлением типа документа*, — DOCTYPE, за ним следует тег `html` (в угловых скобках — < и >), потом `head` («голова», заголовок), следом `body` («тело», тело документа), а в нем — непосредственно содержимое веб-страницы. Все это завершается закрывающим тегом `</html>`.

Как работают HTML-теги

В приведенном выше примере, как и в HTML-коде любой веб-страницы, большинство команд-тегов используются парами, начиная и завершая какой-то фрагмент — блок текста или другие команды.

Будучи заключенными в скобки, эти теги представляют собой команды, которые говорят браузеру, каким образом отображать веб-страницу.

Открывающий тег каждой пары показывает браузеру, где команда начинается, а заканчивающий — где заканчивается. Закрывающий тег всегда предваряется прямым слэшем (/) после первого символа скобки (<).

Для функционирования веб-страницы необходимо наличие как минимум четырех элементов.

- Самая первая строка примера содержит объявление типа документа — DOCTYPE. Это объявление — не совсем настоящий HTML-тег: такая строка сообщает браузеру о том, с каким именно типом HTML-страницы ему придется встретиться. Подобных типов существует не так много. Один из них XHTML — веб-страница, основанная на языке разметки XML. Об XHTML вы узнаете в следующем разделе. Конечно, можно и не использовать объявление типа документа, но это считается дурным тоном. Кроме того, далее вы узнаете, что DOCTYPE — это один из основных элементов, необходимых для того, чтобы быть уверенным, что ваше CSS-оформление будет работать во всех браузерах.
- Тег `<html>` требуется в начале веб-страницы и (с добавленным слешем) в конце. Этот тег говорит браузеру, что документ является программным кодом на языке HTML.

  Все содержимое страницы, включая остальные теги, находится между открывающим и закрывающим `<html>`.

  Если представить веб-страницу в виде дерева, то `<html>` будет его стволом.

  Две основные части любой веб-страницы — заголовок и тело — представляют собой ветви.
- Заголовок веб-страницы, заключенный в теги `<head>`, содержит название. Здесь также может содержаться другая информация, невидимая при просмотре веб-страницы (ключевые слова поиска и т. д.), которая предназначена для браузеров и поисковых машин.

  Кроме того, заголовок может содержать информацию, используемую браузером для правильного отображения веб-страницы и для придания ей интерактивности. В заголовке документа можно разместить, например, таблицы стилей, а также сценарии JavaScript, функции, определения переменных.
- Тело веб-страницы, следующее непосредственно за заголовком и заключенное в теги `<body>`, содержит все, что должно появиться в окне браузера: заголовки, текст, изображения и т. д.

  Внутри `<body>`, как правило, можно найти следующие теги:

  - `<p>` — открывающий тег начинает абзац, а закрывающий `</p>` — заканчивает;
  - `<strong>` — выделяет текст полужирным шрифтом; например, фрагмент `<strong>Warning!</strong>` сообщит браузеру о том, что слово Warning! должно быть выделено;
  - `<a>`, или тег привязки (якорный), — создает гиперссылку, при щелчке на которой можно переместиться в другое место веб-страницы или на другой сайт (нужно указать браузеру эту ссылку путем размещения ее внутри `<a>`, например, можно набрать `<a href="http://www.missingmanuals.com/">Нажмите здесь!</a>`).

Браузер знает, что при щелчке кнопкой мыши на ссылке со словами **Нажмите здесь!** посетитель вашей страницы должен перейти на сайт с адресом **http://www.missingmanuals.com**. Часть тега `a` — слово `href` — называют *атрибутом*, а URL

(унифицированный указатель информационного ресурса, или адрес URL) является его *значением*. В этом примере `http://www.missingmanuals.com` — значение атрибута `href`.

XHTML: современный HTML

Множество языков продолжают сражаться за титул лучшего языка для Всемирной паутины. Язык HTML 4.01, созданный еще в прошлом тысячелетии, хотя и всего 10 лет назад, уже, можно считать, повержен своим преемником. HTML всегда был непривередлив: строчные или прописные буквы в тегах или вообще их смесь (например, теги `<body>`, `<BODY>`, `<BoDy>` абсолютно идентичны), наличие возможности опускать закрывающие теги (например, можно создать абзац, используя только открывающий тег `<p>`, а закрывающий `</p>` даже и не указывать). С одной стороны, подобная гибкость позволяет создавать сайты гораздо быстрее, но с другой — она доставляет массу проблем для различных браузеров, карманных устройств и других мест, где у вас есть возможность использовать свои веб-страницы.

Появление XHTML 1.0 — это новая жизнь HTML, но уже для повсеместного использования. Если вам знаком HTML, не волнуйтесь: XHTML — это не революционно новый язык, на изучение которого уйдут годы. Это обычный HTML, но основанный на возможностях XML. Как и HTML язык XML использует теги, которые позволяют просто и понятно собирать информацию воедино так, что различные компьютеры, операционные системы и программы смогут беспрепятственно ею обмениваться. Однако в отличие от HTML язык XHTML не ограничен конкретным набором тегов. На самом деле XML позволяет вам задавать свои собственные теги, используя для этого определенные правила. XML вообще дает возможность делать очень многое, начиная от создания RSS-потоков новостей и заканчивая созданием списков воспроизведения для iTunes.

Прекрасный способ узнать, что лучше — HTML 4.01 или XHTML 1.0, — вести горячие дискуссии в Сети. Вам даже может показаться, что это два абсолютно разных языка, но это далеко не так. С помощью HTML 4.01 вы можете сделать прекрасный и функциональный сайт, который будет работать еще очень и очень долго.

Что ж, если вы хотите продолжать использовать HTML, то внимательно следуйте рекомендациям, предложенным в гл. 1. Например, вы обязательно должны объявлять тип документа для своих HTML-страниц, иначе ваши CSS-стили могут перестать работать в некоторых браузерах. Кроме того, вам необходимо *валидировать* вашу страницу (то есть проверить ее на соответствие всем нормам и правилам для данного типа HTML), чтобы быть уверенными в том, что в ней нет никаких типологических или других ошибок, которые могут повлиять на ее внешний вид. То же самое вам нужно проделывать и с XHTML-страницами, правила валидации которых значительно строже. К примеру, если для HTML-документов указывать DOCTYPE не обязательно, то для XHTML это непременное условие.

ПРИМЕЧАНИЕ

Если вы действительно хотите разобраться в структуре XHTML, зайдите на обучающий сайт W3Schools XHTML по адресу www.w3schools.com/xhtml/default.asp.

Код страницы HTML, показанный в начале книги, в XHTML будет выглядеть так:

```
<!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/1999/xhtml">
  <head>
    <title>Это название веб-страницы.</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  </head>
  <body>
    <p>Это содержимое веб-страницы.</p>
  </body>
</html>
```

Как видите, этот код очень походит на HTML-код. Чтобы XHTML-файл соответствовал стандартам XML, нужно придерживаться некоторых правил.

- **Страница должна начинаться с объявления типа документа (DOCTYPE).** Это две самые первые строки кода. Можно подумать, что они ничем не отличаются от тех, что представлены в примере ранее: здесь объявляется тип XHTML 1.0 Transitional для XHTML-документа. Чуть больше об этом, а также о важности этих типов для CSS вы узнаете из гл. 1.
- **Теги и их атрибуты должны быть написаны в нижнем регистре.** Это же касается и <body>.
- **Для атрибутов тегов требуются кавычки.** Например, ссылка <a href=http://www.missingmanuals.com> правильна в HTML, но не будет работать в XHTML. Вы должны заключить значение атрибута href в кавычки: <a href="http://www.missingmanuals.com">.
- **Все теги (даже пустые) должны быть закрыты.** Например, абзац в XHTML должен начинаться с и заканчиваться . Проблема в том, что некоторые теги не имеют пар (пустые), то есть у них нет завершающего тега. Таков, например, тег разрыва строки. Чтобы закрыть его, нужно вписать в конце тега, перед закрывающей скобкой, символ обратного слэша: <br/>.

HTML 5 — новый виток

Всемирная паутина не ограничивается языками XML или XHTML. В 2006 году, на момент выхода первого издания этой книги, Консорциум W3C (Консорциум Всемирной паутины, World Wide Web Consortium) работал над языком XHTML 2 — новой, более мощной версией XHTML, которая должна была навсегда изменить представление дизайнеров о том, как следует создавать веб-страницы. Однако мешала одна маленькая проблема: все выглядело так сложно, что, казалось, без специального образования в сфере компьютерных наук сделать веб-страницу просто невозможно. После того как большинство создателей браузеров, включая таких, как Mozilla (Firefox) и Apple (Safari), отказались делать браузеры, поддерживающие XHTML 2, W3C сменил направление деятельности и сформировал группу разработчиков для очередного нового стандарта, которым стал HTML 5.

Итак, HTML собирается вновь быть у руля. Но, по правде говоря, случится это не раньше 2022 года. Иными словами, не беспокойтесь пока о том, что вам изучать: HTML или XHTML.

В данный момент вы можете свободно использовать как HTML 4.01, так и XHTML 1.0. Все браузеры понимают эти языки. В гл. 1 я вас познакомлю со способами, которые укрепят связку HTML (или XHTML) и CSS.

ПРИМЕЧАНИЕ

Если вы не очень хорошо знакомы с созданием веб-страниц, то многое покажется сложным и не совсем понятным. Не волнуйтесь, в гл. 1 мы поговорим о таком современном инструменте, как W3C Validator, который служит для того, чтобы проверить правильность кода XHTML. Кроме того, он проверяет правильность вашей страницы и сообщает, все ли в порядке.

Программное обеспечение для CSS

Чтобы создавать веб-страницы на языках HTML и CSS, вполне достаточно обычного текстового редактора, такого как Блокнот в Windows или Text Edit в Macintosh. После набора нескольких сотен строк кода HTML или CSS вы, наверное, захотите пользоваться программой, более подходящей для работы с веб-страницами. В этом разделе перечислены некоторые из них. Одни бесплатные, другие придется приобрести.

ПРИМЕЧАНИЕ

Существуют буквально сотни программ, которые могут помочь вам в создании веб-страниц, поэтому здесь приводится неполный список. Все же это самые популярные программы, которыми пользуются любители CSS на сегодняшний день.

Бесплатные программы

Есть много бесплатных программ для редактирования веб-страниц и таблиц стилей.

Если вы все еще пользуетесь обычным текстовым редактором, то имеет смысл попробовать одну из нижеприведенных программ.

- **jEdit** (Windows, Mac, Linux, http://jedit.org/). Бесплатный текстовый редактор, использующий Java, работает практически на всех компьютерах. В нем вы найдете большинство тех функций, которые доступны в коммерческих программах, включая подсветку синтаксиса для CSS.
- **Notepad++** (Windows, http://notepad-plus.sourceforge.net/). Множество людей просто преклоняются перед этим текстовым редактором. Естественно, в нем есть встроенная функциональность, которая идеально подходит для написания HTML- и CSS-кода, включая подсветку синтаксиса, — теги и другие ключевые слова имеют собственные цвета, что значительно облегчает их поиск среди других элементов HTML и CSS.
- **HTML-Kit** (Windows, www.chami.com/html-kit/). Этот мощный редактор HTML/XHTML предоставляет множество полезных средств для создания веб-страниц. Среди них: предварительный просмотр веб-страницы непосредственно в программе (поэтому вам не придется переключаться между браузером и редактором), ярлыки для добавления HTML-тегов и т. д.

- **TextWrangler** (Macintosh, www.barebones.com/products/textwrangler/). Эта бесплатная программа — практически облегченная версия текстового редактора BBEdit для Macintosh. У TextWrangler нет встроенных средств, облегчающих редактирование HTML, но он обеспечивает цветовую подсветку синтаксиса (имеется в виду, что теги и атрибуты просто подсвечиваются разными цветами для удобного просмотра страницы и определения ее фрагментов), имеется поддержка протокола FTP (вы можете загружать файлы на сервер) и др.

Коммерческое программное обеспечение

Существует множество коммерческих программ для создания сайтов: от недорогих тектовых редакторов до мощных конструкторов.

- **EditPlus** (Windows, www.editplus.com) — недорогой текстовый редактор, который включает подсветку синтаксиса, поддержку FTP, автозавершение ввода и другие функции.
- **skEdit** (Macintosh, www.skti.org) — дешевый редактор веб-страниц, полная поддержка FTP/SFTP, подсказка команд и другие полезные функции.
- **Coda** (Mac, www.panic.com/coda/). Многофункциональное средство для создания веб-страниц. Включает в себя текстовый редактор, средство предварительного просмотра страниц, FTP- и SFTP-клиент и графический интерфейс для создания CSS-стилей.
- **Dreamweaver** (Macintosh и Windows, www.macromedia.com/software/dreamweaver) — визуальный редактор веб-страниц. Он позволяет видеть, как ваша страница выглядит в браузере. Программа также включает мощный текстовый редактор и превосходные инструменты создания кода CSS. Смотрите полную документацию (например, мою книгу *Dreamweaver: The Missing Manual*, O'Reilly) для эффективного использования этой мощной программы.
- **Expression Web 2** (Windows, www.microsoft.com/expression) — новая программа Microsoft для веб-дизайна. Она заменяет FrontPage и включает много профессиональных инструментов веб-дизайна, а также хорошую поддержку создания кода CSS.

ПРИМЕЧАНИЕ

Здесь речь идет о программах, которые позволяют редактировать код, написанный на HTML/XHTML и CSS. Вам достаточно изучить всего одну из них для создания веб-страниц. Если у вас уже есть любимый редактор HTML/XHTML, которым постоянно пользуетесь, но средства которого не позволяют редактировать код CSS, то можете найти CSS-ориентированный редактор. Их список приведен в приложении 3.

Об этой книге

Интернет — действительно очень удобное изобретение. К сожалению, правила, по которым работает Сеть, не так просты для понимания. Программисты и технические специалисты, которые пишут официальную документацию, поясняющую основные понятия ее функционирования, не ориентируются на среднестатистического пользователя. Зайдите на сайт www.w3.org/TR/CSS21/, чтобы осознать, может ли это быть понятно обычному человеку.

Не существует никакого официального руководства для изучения CSS. Люди, только начинающие заниматься этим, как правило, не знают, с чего начать. Цель этой книги — служить руководством для обучения. На ее страницах вы найдете пошаговые инструкции для создания красивых веб-страниц с использованием языка CSS.

Книга написана так, чтобы помочь читателям любого уровня. Для извлечения максимальной пользы из данного материала вы обязательно должны учиться на приведенных примерах HTML и CSS. Если же вы никогда раньше не создавали веб-страницы, то обратитесь к обучающему курсу, который приводится в конце каждой главы.

Материал, содержащийся в этих главах, написан для продвинутых новичков или тех, кто имеет средний уровень знаний в этой области. Если же вы плохо знакомы с созданием веб-страниц, то для лучшего понимания освещаемой темы должны ознакомиться с текстом врезки «Информация для новичков». С другой стороны, если у вас имеется большой опыт создания веб-страниц, то не обращайте внимания на эти врезки. Они содержат подсказки, приемы и методы для компьютерных пользователей, но не для опытных программистов.

ИНФОРМАЦИЯ ДЛЯ НОВИЧКОВ

Разновидности CSS

Как производители операционных систем и устройств iPods, так и создатели CSS постоянно выпускают новые версии своих программных продуктов. CSS 1, появившийся в 1996 году, стал основой для дальнейшего совершенствования языка каскадных таблиц стилей. В этой самой первой версии языка CSS были введены основные понятия и команды: структура таблиц стилей, понятие идентификатора элемента (см. гл. 3) и большая часть свойств CSS.

В версии CSS 2 были добавлены новые возможности, включая совместимость с различными принтерами, мониторами и другими устройствами. Здесь также добавились новые определения идентификаторов элементов и способность точно позиционировать элементы на веб-страницах.

Данная книга полностью описывает версию языка CSS 2.1, которая на сегодняшний день является стандартом языка каскадных таблиц стилей. Она унаследовала все возможности и особенности CSS 1. Сюда добавлены некоторые новые свойства, а также исправлены неточности предыдущих версий.

И все-таки CSS 2.1 мало чем отличается от CSS 2, и большинство браузеров давно адаптированы под эту версию. Исключение составляет Internet Explorer 6, вследствие чего вы найдете в книге много примеров обходных путей правильного отображения веб-страниц с использованием CSS 2.1 в различных версиях браузера. К счастью, в версии браузера Internet Explorer 7 исправлено большинство недостатков предыдущих версий, а версия Internet Explorer 8 наконец обрабатывает практически все правила CSS 2.1 верно.

Версия CSS 3 готовится к выпуску. Хотя Консорциум World Wide Web (W3C) еще не утвердил этот стандарт, производители некоторых браузеров уже адаптируют свои программные продукты с учетом рекомендаций и особенностей нового стандарта. Технология Safari нового CSS 3 предоставляет возможность добавлять несколько изображений в фон одного элемента. А вообще, очень много нововведений CSS 3 уже работают в современных браузерах, так что вся глава 16 посвящена именно им. Лучшим ресурсом, внимательно следящим за развитием нового стандарта CSS 3, является сайт CSS3.info (http://www.css3.info).

ПРИМЕЧАНИЕ

В этой книге я периодически рекомендую пользоваться другой литературой по CSS, если темы недостаточно развернуто освещены в данном руководстве.

Основные разделы книги

Книга разделена на четыре части, каждая из которых содержит несколько глав.

- **Часть 1. Основы CSS.** Здесь описано создание каскадных таблиц стилей в целом и дан краткий обзор ключевых понятий, таких как наследование, идентификаторы элементов, свойство каскадности таблиц стилей. Попутно с изучением CSS вы получите основные навыки написания кода HTML/XHTML. Четыре обучающие программы закрепят полученные знания и дадут почувствовать мощь каскадных таблиц стилей.
- **Часть 2. Применение CSS.** Перенесет вас в реальный мир веб-дизайна. Вы изучите наиболее важные свойства CSS и их использование для форматирования текста, попрактикуетесь в создании полезных инструментов навигации и сможете улучшить внешний вид своих экспериментальных веб-страниц, добавив графику. Эта часть также содержит рекомендации, как улучшить вид распечатываемых веб-страниц и как создавать красивые таблицы и формы.
- **Часть 3. Макет страницы.** Поможет вам разобраться с самым запутанным, но очень полезным аспектом CSS: управлением размещения элементов на странице. Вы познакомитесь со схемами дизайна (размещение разделов в два и три столбца) и узнаете, как добавить боковые меню. Вы также узнаете о двух основных методах позиционирования элементов на странице: абсолютном и относительном.

  Кроме того, главы этой части научат вас создавать совершенные веб-страницы, которые выглядят одинаково хорошо как на экране монитора, так и в распечатанном виде. Описаны также методы еще более эффективного применения CSS. Ну и, конечно же, именно здесь вы узнаете о будущем CSS, а также научитесь использовать некоторые новейшие возможности CSS 3, работающие уже сегодня.
- **Приложения.** Справочник свойств CSS описывает каждое свойство в отдельности в простой и доступной для понимания форме, служит для быстрого поиска нужных элементов, облегчения повторения пройденного курса обучения. По сути, это систематизированный справочник CSS. В остальных двух приложениях находится описание инструментов и средств для создания и применения каскадных таблиц стилей: от приемов создания кода CSS в программе Dreamweaver до перечня полезных сайтов и книг.

Примеры программного кода на CSS

Эта книга создана для того, чтобы организовать быструю и профессиональную работу с вашими веб-проектами.

По ходу чтения книги вы будете находить примеры *пошаговых обучающих программ*, которые сможете выполнять, используя исходные данные (графические образы и незаконченные веб-страницы), загруженные с сайта www.sawmac.com/css2e/. Вы немногому научитесь, если просто прочтете эти уроки, отвлекаясь на еще что-нибудь. Однако если не будете торопиться, а станете терпеливо работать над

примерами на компьютере, то это даст реальное понимание того, как проектировщики-профессионалы создают веб-страницы.

В обучающих уроках вы также найдете URL законченных страниц, чтобы сравнить свою работу с требуемым конечным результатом. Другими словами, вы не только увидите, как веб-страницы должны выглядеть внешне, но и найдете их в Интернете.

О сайте MissingManuals.com

На сайте www.missingmanuals.com вы можете найти статьи, подсказки, обновления к данной книге. Я охотно прислушиваюсь к вашим замечаниям, дополнениям, исправлениям, касающимся содержания, и учитываю их. Книга претерпевает изменения всякий раз, когда мы выпускаем очередной ее тираж, вносим уточняющие исправления, выбранные из тех, которые вы предлагаете на сайте.

На сайте также можно подписаться на бесплатную рассылку, которая уведомит вас о запуске в печать новых книг.

Основная терминология

При чтении данной книги и выполнении примеров на компьютере вы должны быть знакомы с некоторыми терминами и понятиями.

- **Щелчок кнопкой мыши.** Пользуясь манипулятором типа «мышь» вашего компьютера, вы можете выполнить три действия. Щелчок кнопкой мыши означает, что нужно навести указатель мыши на какой-либо объект на экране монитора, а затем, не перемещая указатель, нажать и отпустить левую кнопку мыши. Двойной щелчок кнопкой мыши означает, соответственно, быстрое нажатие кнопки мыши дважды, опять-таки не перемещая указатель. Перетаскивание мышью означает перемещение указателя при нажатой и удерживаемой кнопке мыши.

  Если говорится, что нужно выполнить щелчок кнопкой мыши на Macintosh или Ctrl-щелчок на PC, то вы просто щёлкаете кнопкой мыши, предварительно нажав клавишу ⌘ или Ctrl (две клавиши Ctrl находятся по обе стороны от клавиши Пробел).

- **Меню.** Это строка с кнопками, находящаяся вверху вашего экрана или окна: File (Файл), Edit (Редактирование) и т. д. Чтобы появился список команд, соответствующих каждому конкретному пункту меню, нужно просто щелкнуть кнопкой мыши на нужном слове (пункте меню), в результате раскроется перечень команд меню.

- **Операционные системы.** В книге подразумевается, что вы знаете, как открывать программы, просматривать страницы в Интернете, загружать файлы. Вы должны знать, как пользоваться меню Пуск в Windows или -меню в Macintosh, а также Панелью управления в Windows или Системными настройками в Macintosh OS.

  Если вы владеете всей этой информацией, то у вас есть необходимая техническая основа для того, чтобы попробовать себя в программировании и насладиться возможностями языка CSS, читая это руководство.

О стрелках

В этой книге вы будете находить текст такого рода: «Откройте папку System ▸ Library ▸ Fonts (Система ▸ Библиотека ▸ Шрифты)». Это сокращение ряда цепочки команд, которые вы должны выполнить, чтобы открыть последовательно три вложенные папки. Это означает: «На жестком диске найдите папку под названием System (Система). Откройте ее. В окне системной папки есть папка под названием Library (Библиотека); дважды щелкните кнопкой мыши, чтобы открыть ее. В этой папке находится папка с названием Fonts (Шрифты). Дважды щелкните кнопкой мыши и также откройте ее».

Точно такие же сокращенные команды помогут вам открыть нужный пункт меню, как показано на рис. 0.1.

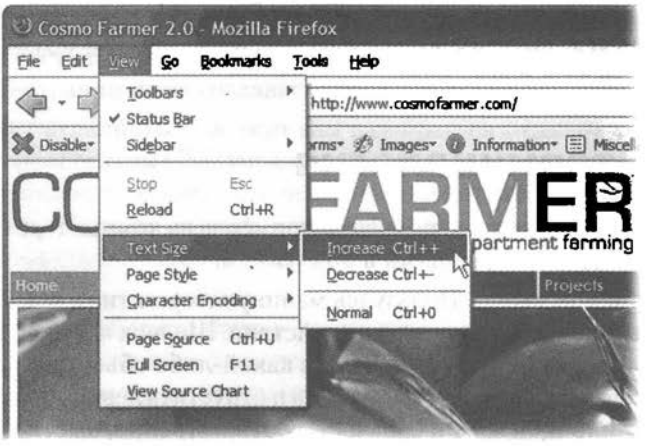


Рис. 0.1. Такой способ перехода по стрелкам упрощает выполнение команд меню

Например, выбор меню View ▸ Text Size ▸ Increase (Вид ▸ Размер шрифта ▸ Увеличить) — более компактный способ сказать: «В пункте меню View (Вид) выберите подпункт Text Size (Размер шрифта), а затем выберите подпункт Increase (Увеличить)».

Соглашения, использованные в данной книге

Здесь приводится список соглашений, использованных в данной книге.

Шрифт для названий

Применяется для отображения URL, а также названий папок и выводимой на экран информации.

`Шрифт для команд`

Используется для имен файлов, названий путей, имен переменных и команд. Например, путь будет выглядеть так: `/Developer/Applications`.

`Шрифт с постоянной шириной`

Применяется для отображения примеров исходного кода и содержимого файлов.

`Полужирный шрифт с постоянной шириной`

Используется для выделения кода, добавленного в старый код.

Вам следует обращать особое внимание на специальные заметки, отделенные от основного текста.

ПРИМЕЧАНИЕ

Это подсказка, пожелание, заметка общего типа. Содержит полезную прикладную информацию по рассматриваемой теме.

ВНИМАНИЕ

Это предостережение или указание, говорящее о том, что вам необходимо быть внимательным. Оно часто указывает на то, что ваши деньги или ваша частная информация могут оказаться под угрозой.

СОВЕТ

Это совет. Советы содержат полезную информацию по рассматриваемой теме, зачастую выделяя важные концепции или лучшие практические решения.

Технология Safari®



Когда вы видите рисунок Safari® Enabled на обложке своей любимой технической книги, это значит, что вы можете получить доступ к этой книге в Интернете с помощью электронной библиотеки O'Reilly Safari.

Safari предлагает более продвинутую систему, чем электронные книги. Это виртуальная библиотека, которая позволяет проводить поиск по тысячам лучших технических книг, копировать фрагменты примеров исходного кода, загружать целые главы, а также находить быстрые ответы на вопросы, когда вам нужна наиболее точная и современная информация по данной теме. Воспользуйтесь бесплатно этой библиотекой по адресу http://my.safaribooksonline.com.

От издательства

Ваши замечания, предложения и вопросы отправляйте по адресу электронной почты halickaya@minsk.piter.com (издательство «Питер», компьютерная редакция).

Мы будем рады узнать ваше мнение!

На сайте издательства http://www.piter.com вы найдете подробную информацию о наших книгах.

ЧАСТЬ 1

Основы CSS

Глава 1. CSS — новый подход к HTML
Глава 2. Создание стилей и таблиц стилей
Глава 3. Селекторы: определение элементов стилизации
Глава 4. Механизм наследования стилей
Глава 5. Управление сложной структурой стилей: каскадность

1 CSS — новый подход к HTML

Ваша веб-страница должна иметь грамотно написанный HTML-код, чтобы можно было сказать, что вы достигли наилучших результатов, применяя CSS. Из этой главы вы узнаете, как писать HTML-код, хорошо адаптированный под нужды CSS. Если вы используете на своих веб-страницах CSS, то фактически становится *легче* писать HTML-код. Больше не нужно применять средства HTML для создания и улучшения дизайна страниц (собственно, HTML для этого никогда и не предназначался). Все, что связано с графическим дизайном страниц, обеспечивается с помощью CSS. Соответственно, работа над основным кодом веб-страниц на языке HTML упрощается, поскольку HTML-страницы, написанные для совместной работы с CSS, имеют менее громоздкий, более понятный и прозрачный код. При этом, естественно, страницы загружаются быстрее, что немаловажно для посетителей вашего сайта.

Посмотрите на рис. 1.1. Дизайн обеих страниц одинаков, однако верхняя создана лишь с использованием CSS, а нижняя — только с помощью HTML. Размер HTML-файла верхней страницы всего 4 Кбайт, в то время как его размер для нижней страницы, полностью написанной на HTML, — почти в четыре раза больше (14 Кбайт). Подход к написанию веб-страниц с применением исключительно HTML требует намного большего объема кода, чтобы достичь практически тех же визуальных эффектов: 213 строк HTML-кода по сравнению с 71 строкой для версии с применением CSS.

HTML: прошлое и настоящее

Как говорилось во введении, HTML или XHTML на сегодняшний день являются основой для написания любой веб-страницы во Всемирной паутине. При использовании CSS написание кода на языке HTML коренным образом меняется. Забудьте про HTML-теги, которые совсем не подходят для создания дизайна страниц и достижения визуальных эффектов. Особенно это касается HTML-тегов типа `<font>`. В следующих разделах я объясню почему.

ПРИМЕЧАНИЕ

В этой главе все, что говорится про HTML, касается и XHTML. Существует множество разновидностей HTML и XHTML. В итоге нужно выбрать свой вариант и убедиться, что в вашей веб-странице определены тип и версия языка разметки, иначе браузеры посетителей не смогут корректно отобразить содержимое веб-страниц. О том, как сообщить CSS о версии HTML/XHTML, на котором написаны ваши веб-страницы, я расскажу чуть ниже.



Рис. 1.1. Веб-дизайн с применением CSS делает написание HTML-кода проще

Прошлое HTML: все выглядело прилично

Когда группа ученых создала Интернет, который первоначально предназначался для совместного использования и поиска технической документации, никто и не задумывался о том, что документы могут содержать графическую информацию. HTML нужен был только для структурирования информации и представления

документов в простом и понятном виде. Возьмем для примера тег <h1>. Он выделяет в тексте важный заголовок, а тег <h2> создает подзаголовок с меньшим размером шрифта. Другой широко используемый тег — (ordered list — «упорядоченный список») — создает нумерованный список для перечислений типа: «Десять причин не играть с медузой».

Когда язык HTML стали применять не только ученые, но и обычные пользователи, они захотели, чтобы их веб-страницы выглядели красиво. С тех пор дизайнеры веб-страниц начали использовать теги для стилизации страниц в дополнение к их основному назначению — структурированию данных. Например, вы можете применить тег <blockquote> (предназначен для цитирования материала из другого источника) к любому тексту, который нужно выделить небольшим отступом вправо. Вы можете пользоваться тегами заголовков, чтобы выделить любой текст полужирным шрифтом большего размера, независимо от того, заголовок это или нет.

Достаточно сложный способ применения тега <table> был придуман веб-дизайнерами, чтобы создать колонки текста, а также точно позиционировать изображения и текст на странице.

Поскольку тег первоначально предназначался для отображения таких таблиц, как результаты исследовательских данных, расписания поездов и т. д., проектировщикам пришлось упражняться в применении тега <table> самыми необычными способами, создавая неоднократно вложенные таблицы, чтобы содержимое веб-страниц смотрелось красиво.

Тем временем производители браузеров также прилагали усилия в совершенствовании языка разметки, вводя новые теги и атрибуты для улучшения дизайна веб-страниц. Например, тег <font> позволяет определять цвет, начертания и один из семи размеров шрифта (это приблизительно в 100 раз меньше типоразмеров шрифта, чем предлагает редактор Microsoft Word).

Наконец, когда проектировщики не могли добиться нужных результатов, они часто применяли графику. Например, использовали очень большой рисунок в качестве фона для веб-страницы или нарезали его на маленькие графические файлы и собирали их воедино в таблицах, чтобы воссоздать оригинальное изображение.

По мере использования всех этих премудростей и применения атрибутов тегов, широкого употребления изображений и т. д. для изменения дизайна страниц HTML-код разрастался до неузнаваемости.

Настоящее HTML: фундамент для CSS

Независимо от того, что представляет собой веб-страница — календарь промыслового сезона, схему проезда к ближайшему супермаркету или фотоальбом прошлого дня рождения вашего ребенка, — именно ее дизайн создает имидж, заставляя сайт выглядеть профессионально. Хороший дизайн страниц сайта помогает донести послание его посетителям, которые должны легко найти там именно то, что ищут.

Таким образом, чтобы веб-страницы на языке HTML выглядели привлекательно, дизайнерам приходилось усиленно трудиться. Дизайн с помощью CSS позволяет HTML вернуться к исполнению своей прямой обязанности — созданию структуры документа. Использование HTML для дизайна веб-страниц на сегодняшний день является признаком дурного тона. Поэтому не волнуйтесь по поводу того, что

у тега <h1> слишком большой размер шрифта или отступы упорядоченного списка очень велики. Вы сможете изменить их с помощью таблиц стилей CSS. Во время написания кода думайте о HTML как о средстве структурирования. Используйте его, чтобы упорядочить содержимое страницы, а CSS — чтобы сделать это содержимое привлекательным.

Написание HTML-кода для CSS

Для тех, кто не очень хорошо знаком с веб-дизайном, возможно, будут полезны некоторые подсказки относительно языка HTML. Если раньше вы уже создавали веб-страницы, то сможете избавиться от стиля написания HTML-кода, который лучше быстрее забыть. Сейчас речь пойдет о стиле написания HTML-кода для его совместного использования с кодом CSS.

Думайте о структуре

HTML придает особый вид тексту путем деления его на логические блоки и их определения на веб-странице: например, основное назначение тега <h1> — создать заголовок-введение, предшествующий основному содержимому страницы. Заголовки второго, третьего уровней и т. д. — подзаголовки — позволяют делить содержимое страниц на менее важные, но связанные разделы. У веб-страницы, как у книги, которую вы держите в руках, должна быть логическая структура. У каждой главы этой книги есть заголовок (отформатированный, например, тегом <h1>), а также несколько разделов и, соответственно, подзаголовков (например, с тегом <h2>), которые, в свою очередь, содержат подразделы с заголовками более низкого уровня. Представьте, насколько сложнее было бы читать эту книгу, если бы весь текст состоял из одного длинного абзаца, без деления на разделы, подразделы, пункты, без выделения примечаний, гиперссылок и т. д.

ПРИМЕЧАНИЕ

Дополнительную литературу по языкам разметки HTML/XHTML вы сможете найти в руководстве HTML & XHTML: The Definitive Guide Чака Мусциано (Chuck Musciano) и Билла Кеннеди (Bill Kennedy), издательство O'Reilly, а обучающие программы и примеры — на сайте www.w3schools.com. Полный перечень всех доступных тегов языков разметки HTML/XHTML приведен на сайте www.w3schools.com/tags/.

Помимо тегов заголовков, в HTML есть множество других тегов для *разметки* содержимого веб-страницы, а также для определения назначения ее каждого логического фрагмента. Наиболее часто применяют следующие теги: <p> — для создания абзацев текста, <ul> — для создания маркированных (ненумерованных) списков. Далее по степени применения идут теги, отображающие специфичное содержимое, например <abbr> — сокращения, аббревиатуры, <code> — программный код.

При написании HTML-кода для CSS, выбирая теги, ориентируйтесь на роль, которую играет фрагмент текста на веб-странице, а не на внешний вид, который текст приобретает благодаря этому тегу (рис. 1.2). Например, перечень ссылок на панели управления — это и не заголовок, и не абзац текста. Больше всего это

похоже на маркированный список параметров. Таким образом, выбираем тег <ul>. Вы можете сказать, что элементы маркированного списка расположены вертикально один за другим, а нам требуется горизонтальная панель управления, в которой все ссылки располагаются горизонтально. Об этом можно не беспокоиться. Средствами CSS очень просто преобразовать вертикальный список ссылок в элегантную горизонтальную панель управления, о чем вы сможете прочитать в гл. 9.



Рис. 1.2. Пример изменения внешнего вида текста на странице благодаря использованию CSS

До появления CSS дизайнеры прибегали к применению тега <font> и других средств HTML для достижения определенных визуальных эффектов:

```
<p>
  <strong>
    <font color="#0066FF" size="5" face="Verdana,
      Arial, Helvetica, sans-serif">Urban Agrarian
      Lifestyle</font></strong>
    <br />
    <font color="#FF3300" size="4" face="Georgia,
      Times New Roman, Times, serif">
    <em>
  <strong>A Revolution in Indoor Agriculture
    <br /></strong></em></font>
      Lorem ipsum dolor sit amet...
</p>
```

Используя CSS, можно добиться тех же результатов (и даже лучше) с гораздо меньшим объемом HTML-кода (см. рис. 1.2):

```
<h1>The Urban Agrarian Lifestyle</h1>
<h2>A Revolution in Indoor Agriculture</h2>
<p>Lorem ipsum dolor sit amet...</p>
```

Кроме того, применяя CSS для дизайна веб-страницы, вы используете HTML по его прямому назначению, то есть именно для разметки веб-страницы на логические фрагменты, не заботясь о форматировании и внешнем виде страницы.

Два новых HTML-тега

Даже весь спектр HTML-тегов не может удовлетворить потребностей веб-дизайнера в разметке разнообразного содержимого веб-страниц. Конечно, тег <code> удо-

бен для обозначения и выделения программного кода, но кто-то скажет, что тег <recipe> («набор команд») подошел бы лучше. Но, увы, такого тега не существует. К счастью, HTML предлагает два тега для группировки, объединения и разбивки — в общем, для определения логического фрагмента веб-страницы. Они позволяют точно и просто задать любой фрагмент содержимого веб-страницы и впоследствии, определив обработчик для данного элемента, придать ему необходимый вид с помощью стилей CSS.

БРИЛЛИАНТ БЕЗ ОГРАНКИ

Простой HTML — помощь поисковым серверам

Научившись писать HTML-код так, чтобы использовать его исключительно для структурирования содержимого документов, и применяя CSS как инструмент для создания дизайна, стилизации, придания страницам законченного внешнего вида, вы обнаружите дополнительные преимущества простого и понятного HTML. С одной стороны, вы сможете поднять позиции своих веб-страниц в поисковых системах, заняв верхние строки в списках таких поисковиков, как Google, Yahoo!, MSN и др. Когда поисковые серверы сканируют просторы Интернета в поисках новых сайтов и новой информации, они систематизируют данные, просматривая HTML-коды веб-страниц в поисках реального содержимого. Старый способ написания HTML-кода с применением форматирующих тегов (например <font>) и множества таблиц для создания дизайна страницы влияет на работу поискового сервера. Некоторые поисковые системы прекращают чтение HTML-кода страницы после просмотра определенного количества символов. Если у вас на страницах имеется большой объем HTML-кода для создания дизайна, то поисковый сервер может упустить важное содержимое страниц или некоторые страницы вообще не будут систематизированы. Простой и структурированный HTML, напротив, будет легко просматриваться и систематизироваться поисковыми системами. Использование тега <h1> для выделения важных тем страниц (в противоположность форматированию текста большим полужирным шрифтом) — грамотный подход, поскольку поисковые серверы придают большое значение содержимому этого тега во время сканирования страниц.

Рекомендации Google по созданию веб-страниц для правильного восприятия их поисковыми системами смотрите на сайте www.google.com/webmasters/guidelines.html.

О хитростях написания HTML-кода, оптимизированного специально для поисковых машин, вы прочтете по адресу www.digital-web.com/articles/seo_and_your_web_site/.

Теги и похожи на пустые сосуды, которые вы сами и заполняете. Поскольку у них нет никаких свойств для визуализации, вы можете применять к ним CSS-стили, чтобы фрагменты внутри этих тегов выглядели так, как вам хочется. Тег (предназначен для *деления на фрагменты*) определяет любой отдельный блок содержимого, например абзац или заголовок. Однако вы также можете логически объединить любой набор таких элементов, как заголовок, несколько абзацев, маркированный список и т. д., в единственном блоке . Тег — замечательное средство разбивки веб-страницы на такие логические фрагменты, как баннер, нижний колонтитул, боковое меню и т. д. Впоследствии, используя CSS, вы сможете позиционировать любой из этих фрагментов в выбранное место веб-страницы, создавая сложную схему разметки (см. часть 3).

Тег применим к *внутренним* (inline) элементам страницы, то есть к словам, фразам, которые находятся в пределах абзаца текста или оглавления. Его можно использовать точно так же, как и другие внутренние HTML-теги, например как <a> (чтобы добавить ссылку к фрагменту текста) или (чтобы выделить слово в абзаце полужирным шрифтом). Можно применять тег , чтобы указать название компании, и затем использовать CSS, чтобы выделить это название другим шрифтом, цветом и т. д. Рассмотрим пример этих тегов в работе. К ним добавлены атрибуты id и class, часто используемые для применения стилей к фрагментам страницы.

```
<div id="footer">
  <p>Copyright 2006, <span class="bizName">CosmoFarmer.com</span></p>
  <p>Call customer service at 555-555-5501 for more information</p>
</div>
```

Это было краткое введение в теги для веб-страниц. В разд. «Селекторы классов: точное управление» гл. 3 рассказывается, как пользоваться такими тегами в сочетании с CSS, чтобы получить полный творческий контроль над дизайном веб-страниц.

Что нужно забыть об HTML

CSS позволяет писать более простой и понятный HTML-код. Вам больше не нужно пользоваться тегами и атрибутами HTML для создания и улучшения дизайна веб-страниц. Тег <font> — наглядный тому пример. Единственная его цель состоит в том, чтобы изменить цвет, размер и начертание шрифта текста страницы. Он не выполняет никакого структурирования и не делает страницу логически более понятной.

Привожу список рекомендаций, тегов и атрибутов HTML, которые вы можете легко, без ущерба для внешнего вида страниц заменить CSS-стилями.

- Избавьтесь от тега <font> для управления отображением текста. CSS выполнит это гораздо лучше (о форматировании текста читайте в гл. 6).
- Забудьте о тегах <b> и <i>, используемых для выделения текста курсивом или полужирным шрифтом. CSS в состоянии сделать полужирным или курсивным любой текст, и нет никакой необходимости пользоваться специфическими форматирующими тегами языка HTML. Чтобы *выделить* слово или фразу, пользуйтесь тегом (браузеры в любом случае отображают текст, выделенный этим тегом, как полужирный). Если вы хотите придать тексту чуть меньший акцент, то пользуйтесь для его выделения тегом (браузеры выделяют содержимое этого тега курсивом).

СОВЕТ

Чтобы выделить заголовок статьи курсивом, пользуйтесь тегом <cite> — таким образом вы убьете сразу двух зайцев. Он одновременно выделяет заголовок курсивом и помечает его как цитату для поисковых серверов. Запомните это, пожалуйста.

- Не пользуйтесь тегом <table> для разметки страницы. Применяйте его только с целью отображения табличной информации (например, электронных таблиц,

списков, диаграмм). В части 3 этой книги вы узнаете, что CSS позволяет создать всю необходимую разметку веб-страницы гораздо быстрее и с меньшим объемом кода, нежели с помощью тега <table>.

- Уберите атрибуты тега <body>, которые всего лишь улучшают внешний вид содержимого веб-страницы: background, bgcolor, text, link, alink и vlink устанавливают цвет текста, фона страницы, гиперссылок, добавляют фоновое изображение на страницу. CSS справляется с этим гораздо лучше (см. в гл. 7 и 8 CSS-аналоги этих атрибутов). Избавьтесь также от браузер-зависимых атрибутов, устанавливающих границы страницы: leftmargin, topmargin, marginwidth, marginheight. CSS выполнит и эту работу (см. гл. 7).
- Не злоупотребляйте тегом <br />. Если вы привыкли пользоваться им для вставки разрывов строк, не создавая новый абзац, то можете попасть в затруднительное положение (иногда браузеры автоматически добавляют пустой промежуток между абзацами, а также между заголовками и тегами <p>; раньше разработчики шли сложными обходными путями, чтобы избежать этого интервала между абзацами, заменяя единственный тег <p> несколькими тегами разрывов строк плюс тегом <font>, чтобы первая строка абзаца была *похожа* на заголовок). Использование свойства margin в CSS дает свободу определения таких параметров, и вы с легкостью можете установить интервал между абзацами, заголовками и другими блочными элементами (см. разд. «Управление размерами полей и отступов» гл. 7).

ПРИМЕЧАНИЕ

В следующей главе вы узнаете о технике, называемой «сброс стилей». Она устраняет проблему лишних разрывов строк, вставляемых между абзацами и другими тегами.

В целом добавление в теги атрибутов, управляющих цветом, границами, рамками, фоновым изображением, выравниванием текста, форматированием таблиц, — устаревший стиль написания HTML-кода. Сюда также входят атрибуты позиционирования изображений, текста в абзацах и ячейках таблицы. Вместо этого обратитесь к средствам CSS, которые обеспечат и выравнивание текста (см. подраздел «Выравнивание текста» разд. «Форматирование абзацев текста» гл. 6), и установку границ (см. разд. «Добавление границ» гл. 7), и фоновые параметры (см. разд. «Установка цвета фона» гл. 7), и позиционирование изображений (см. разд. «Фоновые изображения» гл. 8).

Подсказки: выбираем правильную дорогу

Всегда неплохо иметь подробное пошаговое руководство к действию. Если вы все еще не уверены в том, как именно пользоваться HTML для создания хорошо структурированных веб-страниц, даю несколько подсказок.

- Применяйте на странице только один тег <h1>, чтобы определить ее главную тему. Представьте, что это заголовок главы: ведь название главы дается один раз. Корректное использование тега <h1> имеет дополнительное преимущество — правильную систематизацию страниц поисковыми серверами (см. врезку «Бриллиант без огранки» выше).

- Используйте заголовки, чтобы указать относительную важность текста. Если два заголовка имеют одинаковую степень важности в теме вашей страницы, то применяйте тег заголовка одного уровня. Если один из заголовков имеет меньшую значимость, то есть является подтемой другого, — заголовок на уровень ниже, подзаголовок. Например, от заголовка уровня <h2> переходите к подзаголовку уровня <h3> (рис. 1.3). Лучше использовать заголовки по порядку и стараться не пропускать их номера, например, никогда не переходите от тега <h2> к тегу <h5>.

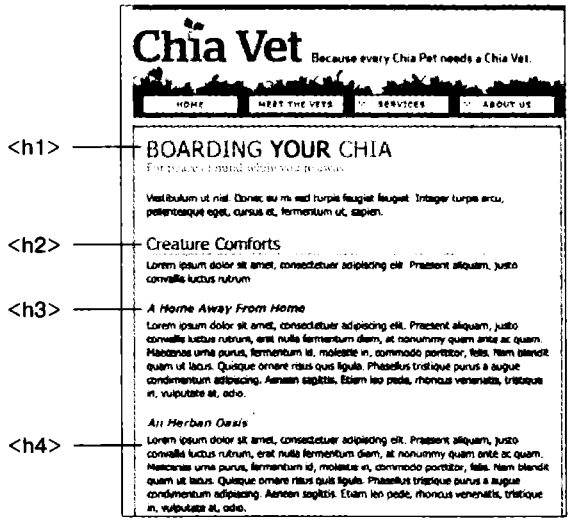


Рис. 1.3. Используйте теги заголовков: расположите их в тексте в порядке важности

- Используйте тег <p> для абзацев текста.
- Применяйте маркированные списки, если у вас есть перечень связанных элементов, таких как ссылки навигации, заголовки, подсказки и др.
- Пользуйтесь нумерованными списками, чтобы определить ряд последовательно выполняемых операций или порядок набора элементов. В обучающих примерах этой книги (см. разд. «Обучающий урок: форматирование текста в действии» гл. 6) есть наглядный пример такого списка.
- Чтобы создать словарь терминов и их определений, пользуйтесь тегом <dl> (список определений терминов) в сочетании с тегами <dt> (название термина) и <dd> (определение термина). Просмотреть пример использования этой комбинированной конструкции тегов можно по адресу www.w3schools.com/tags/tryit.asp?filename=tryhtml_list_definition.
- Если вы хотите включить цитату в виде отрывка текста с другого сайта, чьего-либо высказывания и др., используйте тег <blockquote> для длинного контекста (высказываний), а тег <q> — для коротких цитат.
- Применяйте такие малоизвестные теги, как <cite>, чтобы сослаться на книжный заголовок, газетную статью или сайт, <address> — для указания контактной информации автора страницы (удобно применять для обозначения авторских прав).

- Не используйте теги или их атрибуты для изменения внешнего вида текста, изображений. Все это выполнит CSS.
- Если нет HTML-тега, соответствующего элементу или набору элементов, которые вы хотите выделить на странице для придания им определенного внешнего вида, то пользуйтесь тегами и . Об их использовании будет рассказано в следующих главах (например, в разд. «Селекторы классов: точное управление» гл. 3).
- Не злоупотребляйте тегом . Некоторые дизайнеры полагают, что — это все, что им нужно, и при этом игнорируют теги, которые могут быть более уместны. Возьмем пример создания навигационной панели. Можно добавить блок на страницу и заполнить его кучей ссылок. Но ведь гораздо лучше использовать маркированный список: в конце концов, навигационная панель и сама является списком ссылок.
- Никогда не забывайте указывать закрывающие теги. Открывающий тег требует соответствующего ему закрывающего тега , как и любые другие теги, за исключением одиночных, например
 и <img/>.
- Проверяйте синтаксис своих веб-страниц с помощью W3C-валидатора (см. врезку «Информация для новичков» и рис. 1.4 ниже). Плохо написанный HTML-код, как и код с опечатками, вызовет множество непредсказуемых ошибок браузера.

ИНФОРМАЦИЯ ДЛЯ НОВИЧКОВ

Проверяйте правильность кода веб-страниц

В HTML существуют определенные правила: например, тег <html> охватывает все содержимое веб-страницы, а тег <title> должен находиться внутри <head>. В XHTML синтаксис еще более строгий. Если забыть об этих правилах или просто сделать при наборе опечатку, то некорректный HTML-код станет причиной некоторых проблем (например, веб-страница отобразится в различных браузерах по-разному). Более того, вряд ли вы напишете правильный CSS-код вместе с ошибочным HTML. К счастью, существуют программные средства для проверки правильности HTML-кода.

Самый легкий способ проверить HTML-код — выполнить синтаксический контроль (валидацию) на сайте W3C по адресу http://validator.w3.org/ (см. рис. 1.4). А еще вы можете воспользоваться расширением для браузера Firefox, которое называется Web Developer и размещается по адресу http://chrispederick.com/work/web-developer/. Это еще один простой и быстрый способ протестировать страницу W3C-валидатором.

Можно либо указать валидатору адрес существующей страницы в Интернете, либо загрузить файл с HTML-кодом с вашего компьютера, либо вставить HTML-код веб-страницы в окно формы на сайте и нажать кнопку запуска проверки.

Консорциум Всемирной паутины — World Wide Web Consortium (W3C) — организация, ответственная за определение стандартов веб-технологий и языков программирования, включая HTML, XHTML, XML.

При нахождении ошибок на ваших веб-страницах W3C-валидатор сообщит о них. При использовании браузера Firefox вы можете загрузить и установить надстройку, которая позволит выполнять синтаксический контроль веб-страниц непосредственно в этом браузере, не посещая сайт W3C. Эта надстройка даже попытается исправить обнаруженные ошибки. Загрузить надстройку можно по адресу http://users.skynet.be/mgueury/mozilla/. Подобный инструмент для браузера Safari доступен на сайте www.zappatic.net/safaritidy/.

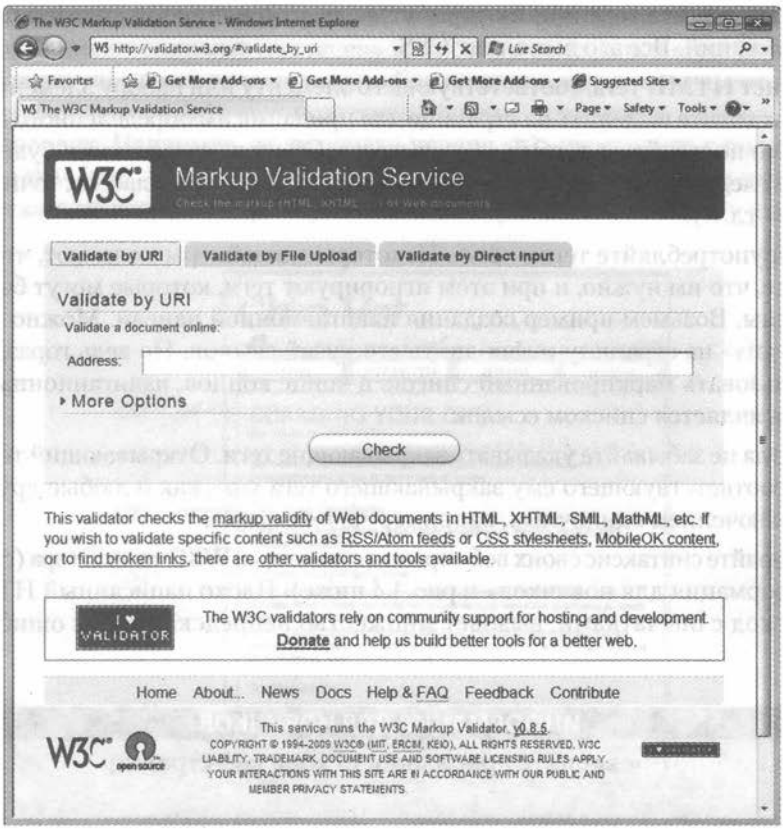


Рис. 1.4. HTML-валидатор от W3C позволит быстро проверить правильность кода веб-страницы

Важность doctype

HTML следует некоторым правилам, содержащимся в файле объявления типа документа (*Document Type Definition* (DTD)). DTD представляет собой XML-документ, определяющий, какие теги, атрибуты и их значения действительны для конкретного типа HTML. Для каждой версии HTML есть свой DTD.

Какое отношение все это имеет к CSS? Самое прямое. Если хотите, чтобы веб-страницы корректно и согласованно отображались браузерами, то вы сами должны сообщить браузеру, какой версией HTML или XHTML пользуетесь, указывая этот самый тип документа — DOCTYPE — в начале веб-страницы. Но DOCTYPE определяет не только версию HTML (например HTML 4.01 Transitional), а и соответствующий DTD-файл в Интернете. Если вы укажете DOCTYPE с ошибкой, то браузер переключится в альтернативный (случайный) режим работы.

Такое переключение — попытка производителей браузеров заставить свое программное обеспечение вести себя подобно устаревшим браузерам, выпускавшимся до 1999 года (Netscape 4, Internet Explorer 5). Если современный браузер сталкивается со страницей, в которой отсутствует правильный DOCTYPE, то он «думает», что

эта страница была написана в текстовом редакторе HTML давным-давно, и отображает ее так, как это сделал бы старый браузер. Именно поэтому без правильного DOCTYPE ваши CSS-стилизованные веб-страницы, возможно, не будут смотреться так, как они должны выглядеть в соответствии с текущими стандартами. Если, проверяя веб-страницу в браузере, вы невольно просматриваете ее в альтернативном режиме, то можете не ломать себе голову, пытаясь исправить проблемы отображения. Они связаны с неправильным DOCTYPE, а не с ошибочным HTML или CSS.

ПРИМЕЧАНИЕ

Для более подробного ознакомления с техническими особенностями альтернативного режима работы браузеров посетите сайты www.quirksmode.org/index.html?/css/quirksmode.html и http://hsivonen.iki.fi/doctype/.

Указать правильный DOCTYPE достаточно просто. Нужно только знать используемую версию HTML. По всей вероятности, вы создаете веб-страницы с применением HTML 4. Возможно, даже начали пользоваться XHTML для сайтов (см. введение).

Самые популярные версии HTML и XHTML на сегодня — HTML 4.01 Transitional и XHTML 1.0 Transitional. Эти типы языка разметки все еще позволяют использовать такие теги, как <font>, обеспечивая переход от старого HTML к новым типам HTML и XHTML, более строгим по синтаксису.

Лучше, конечно, вообще не применять эти теги, хотя они все еще работают в переходных (transitional) версиях языка. Можно постепенно исключить их из кода своих веб-страниц. В синтаксически строгих версиях HTML и XHTML некоторые устаревшие теги не работают вообще.

ПРИМЕЧАНИЕ

В синтаксически строгих версиях HTML и XHTML запрещены теги и атрибуты, предназначенные для стилизации веб-страниц, например тег <font> и атрибут центрирования абзаца. В этих версиях также не используются некоторые другие некогда популярные атрибуты (например, открываемая в новом окне гиперссылка).

Если у вас версия HTML 4.01 Transitional, то добавляйте приведенный ниже doctype в самом начале веб-страниц:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
```

Определение doctype для XHTML 1.0 Transitional похоже, но указывает на другой DTD. Кроме того, необходимо добавить кое-что еще в открывающий тег <html>. Эта строча определяет тип используемого XML языка, в нашем случае это XHTML.

```
<!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/1999/xhtml" xml:lang="en" lang="en">
```

ПРИМЕЧАНИЕ

Если вы используете на своих веб-страницах фреймы, то должны указать doctype, предназначенный специально для их применения. Список соответствий типов doctype смотрите на сайте W3C по адресу www.w3.org/QA/2002/04/valid-dtd-list.html.

Если вы не хотите вникать в такие подробности, то просто убедитесь, что правильно определили doctype из вышеприведенного списка, и всегда добавляйте его первой строкой в HTML-файлах (перед открывающим тегом <html>). В этом увесистом куске кода можно сделать описку или незаметную ошибку, так что лучше всего проверьте валидность страницы, чтобы быть уверенными в правильности. Очень удобно иметь на компьютере шаблон-копию пустой HTML-страницы с правильным doctype. Это дает возможность копировать ее каждый раз при создании новой веб-страницы. Общие шаблоны для написания HTML/XHTML веб-страниц можно найти на сайте книги по адресу http://www.sawmac.com/missing/css2e/.

ПРИМЕЧАНИЕ

Большинство визуальных программных пакетов для создания веб-страниц (Dreamweaver, Expression Web) автоматически добавляют определение doctype каждый раз, когда вы создаете новую веб-страницу. В хороших текстовых редакторах HTML-кода имеются также ярлыки для облегчения добавления doctype.

Долгожданный Internet Explorer 8

Благодаря автоматическому обновлению продуктов Microsoft новый Internet Explorer 8 сумел занять свою нишу на рынке браузеров. К великому счастью дизайнеров, IE 8 считается версией, наиболее соответствующей стандартам. В ней исправлены все ошибки, рожденные браузерами IE 6 и IE 7, а также достигнута практически идеальная совместимость с CSS 2.1. Это значит, что хорошо сделанные сайты будут выглядеть во всех браузерах (IE 8, Firefox, Safari и Opera) практически одинаково. Читая дальше, вы поймете, что это не касается браузеров IE 6 и IE 7, где потребуется создание особого, специфичного для каждого браузера кода, чтобы все выглядело одинаково.

Тем не менее IE 8 можно назвать хамелеоном. Он может принимать вид других версий. Поэтому, если вы не позаботитесь о своем коде, ваша страница может принять совершенно не тот вид, которого вы хотите добиться. Очень важно, например, указывать правильный тип документа (DOCTYPE). Вы уже знаете, что без указания DOCTYPE браузер начинает работать в режиме совместимости. А Internet Explorer 8, работая в этом режиме, постарается представить страницу как Internet Explorer 5 (!!!).

Но это еще не все! IE 8 может притвориться IE 7. Если кто-то, рассматривая ваш сайт в IE 8, нажмет кнопку **Режим совместимости**, то IE 8 станет IE 7 и покажет страницу, отключив все свойства CSS 2.1. То же самое может произойти, если компания Microsoft поместит ваш сайт в список совместимых страниц (Compatibility View List). Это сайты, которые с точки зрения Microsoft выглядят лучше в IE 7, а не в IE 8. Вам не потребуется этот режим, если при создании сайта вы будете учитывать все рекомендации, предложенные в этой книге.

К счастью, у нас есть возможность сказать IE 8 быть просто IE 8. Для этого нужно добавить один meta-тег, чтобы IE 8 начал игнорировать режим совместимости и список совместимых страниц и стал отображать вашу страницу в режиме наибольшей совместимости со стандартами:

```
<meta http-equiv="X-UA-Compatible" content="IE=edge">
```

Поместите эту строку в разделе <head> вашей страницы, сразу за тегом <title>. Эта конструкция будет работать и для последующих версий Internet Explorer. Часть тега "IE=edge" сообщит будущим версиям браузера, что они должны использовать свои стандартные режимы, отображая сайты. Да, и не забудьте помещать эту строку на каждой веб-странице!

ИНФОРМАЦИЯ ДЛЯ НОВИЧКОВ

Кроссбраузерное тестирование

Как известно, существует множество различных браузеров. Если вы являетесь пользователем Windows, то по умолчанию вам доступен Internet Explorer, а еще вы можете установить дополнительные браузеры, например Firefox, Safari, Opera или Google Chrome. Пользователи Apple Mac могут остановить свой выбор на предустановленном Safari или же воспользоваться браузерами Firefox 3 либо Google Chrome 1. Несмотря на то что все последние браузеры лучше всего приспособлены для работы с веб-страницами, основанными на CSS, этого нельзя сказать об Internet Explorer 6, который все еще является самым распространенным браузером в Сети. Даже IE 7 содержит те же огрехи.

Чтобы знать наверняка, что ваши сайты работают у самой широкой аудитории пользователей, вам потребуется протестировать их дизайн на как можно большем количестве браузеров. Предлагаю несколько способов.

Пользователи Windows. Обычно компьютеры под управлением Windows могут запускать только одну версию браузера IE, то есть по умолчанию у вас нет никакой возможности использовать IE 6, IE 7 или IE 8 одновременно на одном компьютере. Что ж, по умолчанию вы этого не можете, а вот благодаря маленькой, но очень полезной программе IETester есть возможность узнать, как выглядят ваши страницы в IE 5.5, IE 6, IE 7 и IE 8 одновременно. Скачать ее можно по адресу www.my-debugbar.com/wiki/IETester/HomePage.

Да, установите и другие браузеры на ваш компьютер: Firefox, Safari, Opera и Chrome. К счастью, вам не понадобится Mac, поскольку Apple выпускает браузер Safari и для Windows-машин.

Пользователи Mac OS. Для таких пользователей тестирование немного затруднено. Вы просто должны проверять сайты в Internet Explorer — это самый распространенный браузер. Даже если ваша искусная работа прекрасно выглядит на вашем компьютере, известные проблемы с отображением страниц в IE 6 или IE 7 могут все испортить. Итак, у вас три варианта. Первый — купить (или одолжить) Windows-компьютер. Второй — установить Windows на собственный компьютер с помощью программы Boot Camp (www.apple.com/macosx/features/bootcamp.html), но у вас должна быть система Mac OS на основе процессора Intel. Третий — воспользоваться программами виртуализации VMWare Fusion или Parallels Desktop. Они позволяют запускать виртуальную машину Windows одновременно с Mac OS так, что появляется возможность переходить с Windows на Mac OS и обратно, тестируя сайты в обеих операционных системах. Этот способ лучший из всех предложенных. Как программы виртуализации, так и Boot Camp требуют копии Windows.

Для всех. Существует еще один способ, доступный для пользователей и Windows, и Mac OS, а также не требующий установки какого-либо дополнительного программного обеспечения. Нужно воспользоваться интернет-сервисами для кроссбраузерного тестирования. Они позволяют видеть то, как выглядят страницы в разных операционных системах и браузерах. Многие из этих сервисов коммерческие, а потому требуют денег за использование. Litmus (http://litmusapp.com/) делает скриншоты ваших страниц во многих браузерах. CrossBrowserTesting.com (www.crossbrowsertesting.com) позволяет удаленно работать с другим компьютером. Иначе говоря, вы можете работать с Windows-машиной, где установлены все версии IE, Firefox и Safari. Такой способ позволяет проверять не только внешний вид страниц, но и взаимодействие с ней, например то, как работают программы, написанные на JavaScript в различных браузерах. Browsercam (www.browsercam.com) — еще один сервис, который позволяет создавать скриншоты страниц, как это делает Litmus, или же воспользоваться удаленным компьютером так, как это возможно в CrossBrowserTesting.

2 Создание стилей и таблиц стилей

Даже самые сложные и красивые сайты (в том числе и тот, который представлен на рис. 2.1) когда-то начинались с определения единственного CSS-стиля. Постепенно, по мере добавления все новых таблиц стилей, можно разработать полностью законченный дизайн сайта. Независимо от того, являетесь вы новичком в изучении CSS или профессиональным веб-дизайнером, всегда следует помнить несколько основных правил создания таблиц стилей. В этой главе мы начнем изучение CSS с основных понятий, которыми нужно руководствоваться в процессе создания и использования таблиц стилей.

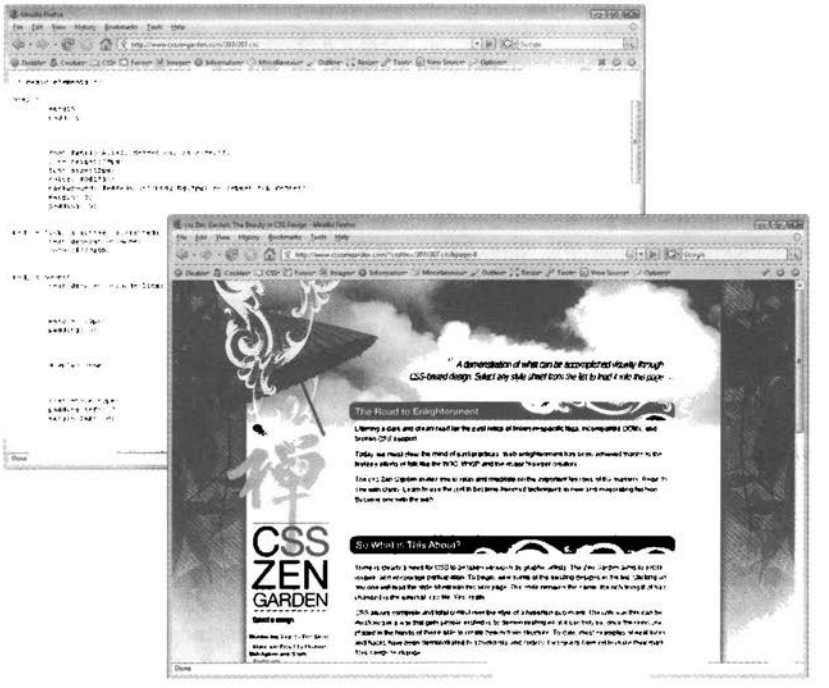


Рис. 2.1. Любая CSS-стилизованная веб-страница состоит из отдельных определений CSS-стилей

В левой части рис. 2.1 представлен код таблицы стилей CSS, определяющей стиль основного содержимого веб-страницы, показанной справа.

ПРИМЕЧАНИЕ

Многие люди лучше воспринимают материал, обучаясь сразу на конкретных примерах, предпочитая их чтению книг и руководств. Если вы сначала хотите попробовать свои силы в создании таблиц стилей, а затем вернуться к этому теоретическому материалу, чтобы прочитать о том, что только что сами сделали, перейдите к обучающему уроку данной главы.

Что такое стиль

Определение стиля в CSS, устанавливающего внешний вид какого-либо элемента (фрагмента) веб-страницы, — это всего лишь правило, которое сообщает браузеру, что и каким образом форматировать: изменить цвет шрифта заголовка на синий, выделить фото красной рамкой, создать меню шириной 150 пикселов для списка гиперссылок. Если бы стиль мог говорить, он сказал бы: «Браузер, сделай, чтобы вот это выглядело так-то». Фактически определение стиля состоит из двух основных элементов: это сам элемент веб-страницы, который непосредственно подлежит форматированию браузером, — *селектор*, а также форматирующие команды — *блок объявления*. Селекторами могут быть заголовок, абзац текста, изображение и т. д. Блоки объявления могут, например, окрасить текст в синий цвет, добавить красную рамку (границу) вокруг абзаца, установить фотографию в центре страницы — возможности форматирования бесконечны.

ПРИМЕЧАНИЕ

Как и разработчики W3C, веб-дизайнеры часто называют CSS-стили правилами. В этой книге оба термина взаимозаменяемы.

Разумеется, CSS-стили не могут быть написаны на обычном языке, как, например, предыдущий абзац. У них есть свой собственный язык. В частности, чтобы выбрать стандартный цвет и размер шрифта для всех абзацев на веб-странице, нужно написать следующее:

```
p { color: red; font-size: 1.5em; }
```

Этот стиль как бы говорит браузеру: «Сделай текст всех абзацев веб-страницы, помеченных тегом <p>, красным и установи размер шрифта равным полуторакратной высоте латинской буквы em (em — буква m, напечатанная шрифтом Cicero, — стандартная единица измерения в полиграфии, обычный размер шрифта текста в браузере, см. гл. 6). Любой стиль, даже самый простой, содержит несколько элементов (рис. 2.2). Он состоит из селектора, который сообщает браузеру, что именно форматировать, и блока объявления, в котором перечислены форматирующие команды, используемые браузером для стилизации фрагмента, определенного селектором.

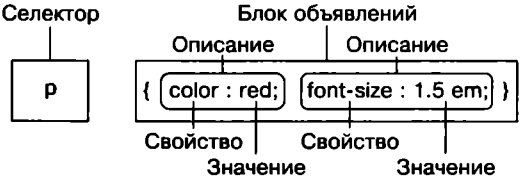


Рис. 2.2. Пример описания стиля (или правила)

- **Селектор.** Как уже было отмечено, селектор сообщает браузеру, к какому элементу или элементам веб-страницы применяется стиль: к заголовку, абзацу, изображению или гиперссылке. На рис. 2.2 селектор `p` обращается к тегу `<p>`, передавая браузеру, что все теги `<p>` нужно форматировать, используя объявления, указанные в данном стиле. Благодаря большому разнообразию селекторов, предлагаемых языком CSS, и небольшому творческому потенциалу вы сможете мастерски форматировать веб-страницы (в следующей главе селекторы описаны более подробно).
- **Блок объявления стиля.** Код, расположенный сразу за селектором, содержит все форматирующие команды, которые можно применить к этому селектору. Блок начинается с открывающей ({) и заканчивается закрывающей фигурной скобкой (}).
- **Объявление свойства.** Между открывающей и закрывающей фигурными скобками блока объявления можно добавить одно или несколько определений или форматирующих команд. Каждое объявление имеет две части — *свойство* и *значение* свойства, заканчивающиеся точкой с запятой.
- **Свойство.** CSS предлагает большой выбор команд форматирования, называемых свойствами. Свойство представляет собой слово или несколько написанных через дефис слов, определяющих конкретный стиль или стилевой эффект. У большинства свойств есть соответствующие, простые для понимания названия, такие как `font-size`, `margin-top`, `background-color` и т. д. (в переводе с английского: размер шрифта, верхний отступ, цвет фона). В следующих главах будет описано множество полезных свойств CSS.

ПРИМЕЧАНИЕ

В приложении 1 вы сможете найти удобный глоссарий свойств CSS.

- **Значение.** Наконец настал тот момент, когда вы можете задействовать свой творческий потенциал, присваивая значения CSS-свойствам: определяя фоновый цвет, например, синим, красным, фиолетовым, салатовым и т. д. Как будет описано в других главах, различные CSS-свойства требуют определенных типов значений — цвет (`red` или `#FF0000`), длина (`18px`, `200%` или `5em`), URL (`images/background.gif`), а также определенных ключевых слов (`top`, `center`, `bottom`).

Вам не обязательно описывать стиль в одной строке, как изображено на рис. 2.2. У стилей может быть множество форматирующих свойств, и есть возможность облегчить просмотр таблицы стилей путем разбивки объявлений на строки. Например, поместите селектор и открывающую скобку в одной строке, каждое объявление — далее в отдельных строках, а закрывающую фигурную скобку — в последней строке стиля. Это будет выглядеть следующим образом:

```
p {
  color: red;
  font-size: 1.5em;
}
```

вместо

```
p { color:red; font-size:1.5em; }
```

Любой браузер игнорирует символы пробела и табуляции, так что вы можете спокойно добавлять их, создавая хорошо читабельные стили CSS. Так, полезно сделать отступ при перечислении свойств табуляцией или несколькими пробелами для явного отделения селектора от блока объявления. И к тому же один пробел между двоеточием и значением свойства, конечно, необязателен, но он обеспечивает дополнительную удобочитаемость стилей. Фактически можно добавить любое количество пробелов там, где вам захочется. Например, `color:red`, `color: red` и `color:  red` — все варианты будут правильно работать.

ПРИМЕЧАНИЕ

Не забывайте ставить в конце каждой пары «свойство — значение» точку с запятой:

color: red;

Пропуская эту точку с запятой, вы собьете с толку браузер, в результате чего таблица стилей будет нарушена и веб-страница отобразится некорректно.

Однако не стоит переживать — описанная ошибка достаточно распространена, поэтому просто убедитесь в том, что используете CSS-валидатор, о котором будет рассказано через пару страниц.

Понимание таблиц стилей

Конечно, один стиль не превратит веб-страницу в произведение искусства. Он может выделить абзацы красным цветом, но, чтобы придать сайту красивый и стильный внешний вид, вам придется определить множество различных стилей. Весь набор определяемых CSS-стилей включается в *таблицу стилей*. Таблицы стилей бывают двух видов — *внутренние* и *внешние*, — в зависимости от того, где определена стилевая информация: непосредственно в самой веб-странице или в отдельном файле, связанном с веб-страницей.

Как выбрать внутренние или внешние таблицы стилей? Еще с момента изобретения CSS внешние таблицы стилей были лучшим способом создания дизайна веб-страниц. Они делают создание сайтов проще, а обновления быстрее. Внешняя таблица стилей сосредотачивает всю информацию о стилях в едином файле, который вы затем присоединяете к странице, написав для этого всего строку кода. Вы можете присоединить одну и ту же внешнюю таблицу стилей к каждой странице сайта, создавая, таким образом, единый дизайн. А обновление внешнего вида всего сайта будет заключаться лишь в редактировании одного-единственного текстового файла — внешней таблицы стилей.

Внешние таблицы стилей помогают веб-страницам открываться быстрее. Когда вы используете внешние таблицы стилей, веб-страницы содержат только сам HTML-код, без кода громоздких вложенных таблиц для стилизации, без тегов `<font>`, без кода встроенных стилей CSS. Кроме того, когда браузер загрузит внешнюю таблицу стилей, он сохранит этот файл на клиентском компьютере посетителя веб-страницы (в специальной системной папке, называемой *кэшем*) для быстрого доступа к нему. Когда посетитель веб-страницы переходит к другим страницам сайта, которые используют ту же самую внешнюю таблицу стилей, браузеру нет никакой необходимости снова загружать таблицу стилей. Он просто загружает запрашиваемый HTML-файл и достает внешнюю таблицу стилей из своего кэша, что дает существенный выигрыш во время загрузки страниц.

ПРИМЕЧАНИЕ

Когда вы работаете над своим сайтом и пользуетесь предварительным просмотром в браузере, кэш работает без всякой пользы для вас. Подробнее об этом рассказано в следующей врезке.

ОБХОДНОЙ ПРИЕМ

Не попадитесь с кэшированием

Кэш браузера обеспечивает значительное увеличение скорости просмотра веб-страниц.

Всякий раз, когда браузер открывает веб-страницу, он помещает загруженную информацию в кэш. Сюда также попадают такие часто используемые файлы, как внешние таблицы стилей CSS или изображения, что позволяет сэкономить время загрузки веб-страниц. Вместо того чтобы в следующий раз (при просмотре других страниц этого же сайта или повторного просмотра этих же страниц в будущем) повторно загружать те же самые файлы, браузер может достать их прямо из кэша, будь то просмотренная ранее веб-страница или рисунок. Однако не всегда то, что хорошо для посетителей сайта, удобно для вас.

Поскольку браузер кэширует и повторно обращается к файлам внешних таблиц стилей CSS, это часто сбивает с толку, например во время работы над дизайном сайта. Допустим, работая над стилизацией веб-страницы, которая использует внешние таблицы стилей, вы просматриваете ее в браузере, дабы убедиться, что добились именно того, чего хотели. Но не все выглядит так, как было задумано, хотя вроде бы в CSS-коде ошибок нет. Вы возвращаетесь в программу-редактор и вносите изменения в файл внешних таблиц стилей CSS. Снова вернувшись в браузер и перезагрузив страницу для просмотра результатов только что внесенных изменений, вы видите, что никаких изменений не произошло! Вы только что попали в ловушку, связанную с особенностями кэша. Дело в том, что при перезагрузке веб-страницы браузер не всегда перезагружает данные, уже находящиеся в кэше, в том числе внешние таблицы стилей. Таким образом, невозможно увидеть, как выглядит веб-страница, стилизованная с помощью только что отредактированного CSS-кода из внешней таблицы стилей.

Есть два пути решения этой проблемы: выключить кэширование или заставить браузер перезагрузить все содержимое веб-страницы.

Чтобы обойти эту путаницу, можно выполнить принудительную перезагрузку страницы (вместе с перезагрузкой всех связанных файлов), нажав клавишу Ctrl, а затем, удерживая ее, кнопку Reload (Обновить) браузера. В Internet Explorer для этой цели предназначено сочетание клавиш Ctrl+F5, а в Firefox — Ctrl+Shift+R.

Внутренние таблицы стилей

Внутренняя таблица стилей — это набор стилей, часть кода веб-страницы, которая всегда должна находиться между открывающим и закрывающим тегами `<style>` и `</style>` HTML-кода, в теле тега `<head>` веб-страницы. Например:

```
<style type="text/css">
h1 {
  color: #FF7643;
  font-family: Arial;
}
p {
  color: red;
  font-size: 1.5em;
```

```
}
</style>
</head>
<body>
/* Далее следует остальная часть вашей веб-страницы... */
```

ПРИМЕЧАНИЕ

Вы можете поместить тег <style> и все его стили после <title>, но веб-дизайнеры обычно размещают их прямо перед закрывающим тегом </head>, как показано в примере выше. Если вы также используете код JavaScript на страницах, добавляйте его после таблиц стилей. Часто программы JavaScript полагаются на CSS, поэтому, добавляя таблицы стилей первыми, вы гарантируете, что код JavaScript будет располагать всей необходимой для своего выполнения информацией.

Тег `<style>` — тег HTML, а не CSS, но именно он сообщает браузеру, что данные, содержащиеся внутри, являются кодом CSS, а не HTML. Создание внутренней таблицы стилей идентично созданию внешней, с той лишь разницей, что перечень стилей не выносится в отдельный файл, а заключается между тегами `<style>`, как описано выше, в самой веб-странице.

Внутренние таблицы стилей можно легко добавить в веб-страницу, и так же просто перейти к редактированию HTML-кода этой же веб-страницы. Однако эти таблицы стилей являются отнюдь не самым эффективным способом для проектирования дизайна сайта, состоящего из множества страниц. Во-первых, вам придется копировать и вставлять код внутренней таблицы стилей в каждую страницу сайта, а это не только трудоемкая, но еще и глупая работа. Этот код делает каждую страницу вашего сайта громоздкой. К тому же такая страница медленно загружается. Во-вторых, внутренние таблицы стилей доставляют много трудностей при обновлении дизайна сайта.

Например, нужно изменить представление заголовков первого уровня (заключенных в тег `<h1>`), которые первоначально отображались крупным, полужирным шрифтом зеленого цвета. Теперь вы хотите, чтобы заголовки были написаны маленьким шрифтом Courier синего цвета. Используя внутренние таблицы стилей, пришлось бы редактировать каждую страницу сайта. У кого-нибудь найдется столько времени? К счастью, для устранения данной проблемы нашлось простое решение — использование внешних таблиц стилей.

ПРИМЕЧАНИЕ

Иногда можно прибегнуть к способу добавления стилевой информации непосредственно к каждому конкретному HTML-тегу без применения таблицы стилей. В обучающем уроке этой главы будет показано, как выполнить такой маневр, используя встроенные стили.

Внешние таблицы стилей

Внешняя таблица стилей — это не что иное, как текстовый файл, содержащий весь набор стилей CSS. Он не должен содержать HTML-кода, поэтому никогда не включайте сюда тег `<style>`. Вдобавок название этого файла всегда должно заканчиваться расширением CSS. Можно давать какое угодно имя этому файлу, но лучше, чтобы оно было наглядным. Назовите файл внешней таблицы стилей, например,

global.css, site.css или main.css, если это общая таблица стилей, связанная со всеми страницами вашего сайта, или form.css, если он содержит описания для стилизации заполняемых форм сайта.

ИНФОРМАЦИЯ ДЛЯ НОВИЧКОВ

Проверяйте правильность CSS-кода

Точно так же, как вы должны были удостовериться в правильности написания HTML-кода веб-страниц, используя валидатор HTML от W3C (см. врезку «Информация для новичков» в разд. «Долгожданный Internet Explorer 8» гл. 1), проверьте CSS-код на наличие ошибок. На сайте W3C имеется инструмент для проверки синтаксиса кода CSS: http://jigsaw.w3.org/css-validator/.

Он работает, как и HTML-валидатор: можно ввести URL-адрес веб-страницы (как вариант — только адрес к внешнему CSS-файлу), загрузить или скопировать CSS-файл, вставить код в форму и просто нажать кнопку запуска проверки.

При наборе CSS-кода очень просто сделать опечатку или ошибку, которой вполне достаточно для того, чтобы изменить до неузнаваемости весь тщательно продуманный дизайн страниц сайта. Если веб-страница, содержащая CSS-код, выглядит не так, как вы ожидали, то причиной тому может быть небольшая ошибка в коде.

CSS-валидатор от W3C — первое средство поиска проблем с дизайном веб-страниц.

Можно также выполнить быструю проверку синтаксиса, используя браузер Firefox. Загрузите страницу с CSS-кодом, которую хотите проверить, и выберите в меню Tools (Инструменты) пункт Error Console (Консоль ошибок). Нажмите кнопку Warnings (Предупреждения), и вы увидите список ошибок или опечаток, которые не прошли проверку Firefox.

СОВЕТ

Если есть веб-страница с внутренней таблицей стилей, а вы хотите использовать внешнюю таблицу стилей, то всего лишь вырежьте фрагмент описания стилей, расположенный между тегами <style> (без самих тегов). Потом создайте новый текстовый файл и вставьте в него CSS-код. Сохраните файл с расширением CSS, например global.css, и свяжите его с вашей веб-страницей, используя одну из методик, описанных далее.

Создав внешнюю таблицу стилей, вы должны подключить ее к веб-странице, которую нужно отформатировать. Можно прикрепить таблицу стилей к веб-странице с помощью HTML-тега <link> или встроенного в CSS правила @import, которое делает то же самое. Все существующие браузеры обрабатывают эти команды одинаково. Обе команды позволяют прикрепить таблицы стилей к веб-странице, так что выбор того или иного способа — лишь вопрос предпочтения.

ПРИМЕЧАНИЕ

Правило @import может сделать одну вещь, с которой тег <link> не справится, — присоединить одну внешнюю таблицу стилей к другой. Этот способ описан в подразделе «Используйте несколько таблиц стилей» разд. «Организация стилей и таблиц стилей» гл. 15.

Связывание таблиц стилей с HTML-кодом

Наиболее распространенный метод добавления внешней таблицы стилей в веб-страницы — использование HTML-тега <link>. Синтаксис применения этого тега

немного различается в зависимости от того, чем вы пользуетесь — HTML или XHTML.

Пример для HTML:

```
<link rel="stylesheet" type="text/css" href="css/global.CSS">
```

Код XHTML:

```
<link rel="stylesheet" type="text/css" href="css/global.css" />
```

Единственное отличие состоит в том, как тег `<link>` заканчивается. В HTML тег связывания — пустой элемент, поэтому у него есть только открывающий тег и нет соответствующего закрывающего тега `</link>`. В XHTML для закрытия тега необходимо добавить слэш (/).

В остальном теги связывания в HTML и XHTML идентичны и требуют наличия трех атрибутов.

- `rel="stylesheet"` — указывает тип ссылки; в данном случае это ссылка на таблицу стилей.
- `type="text/css"` — сообщает браузеру, данные какого типа ему ожидать; в рассматриваемом случае — текстовый файл, содержащий CSS-код.
- `href` — указывает местонахождение внешнего CSS-файла на сайте. Значение этого атрибута — URL-адрес, который будет отличаться в зависимости от того, где вы храните CSS-файл. Он работает так же, как атрибут `src` при добавлении изображения на страницу или атрибут `href` гиперссылки, указывающей на другую веб-страницу.

СОВЕТ

К веб-странице можно присоединить множество таблиц стилей, добавляя несколько тегов <link>, каждый из которых указывает на свой файл таблицы стилей. Эта методика — отличный способ организовать CSS-стили ваших веб-страниц. Подробнее об этом читайте в разд. «Организация стилей и таблиц стилей» гл. 15.

Прикрепление таблиц стилей с использованием CSS

CSS имеет встроенный способ привязки внешних таблиц стилей к коду HTML — правило `@import`. Его нужно добавить в HTML-тег `<style>`. Например:

```
<style type="text/css">
  @import url(css/global.css);
</style>
```

В отличие от HTML-тега `<link>` правило `@import` — языковая конструкция CSS, обладающая некоторыми несвойственными HTML качествами.

- Чтобы выполнить привязку к внешнему файлу CSS, нужно использовать `url` вместо `href` и заключать путь к CSS-файлу в круглые скобки. Так, в рассмотренном выше примере `css/global.css` — путь к внешнему CSS-файлу. Кавычки, в которые заключен URL, необязательны. Таким образом, `url(css/global.css)` и `url("css/global.css")` будут работать одинаково.

- Посредством нескольких правил `@import`, как и с помощью нескольких тегов `<link>`, можно присоединить любое количество внешних таблиц стилей:

```
<style type="text/css">
  @import url(css/global.css);
  @import url(css/forms.css);
</style>
```

- После правила `@import` можно добавлять обычные CSS-стили, если, например, вы хотите создать стиль, который нужно применить только к одной веб-странице, используя для форматирования остального содержимого единый дизайн, уже описанный во внешней таблице стилей.

ПРИМЕЧАНИЕ

О стилях, их приоритетах и взаимодействии, а также о том, как создать стиль, который отменяет другие стили на веб-странице, читайте в разд. «Управление каскадностью» гл. 5. Вы можете даже создать внешний CSS-файл, который содержит только правила @import, выполняющие привязку других файлов внешних таблиц стилей. Такая методика часто применяется в целях упорядочения и систематизации стилей сайта (см. подраздел «Используйте несколько таблиц стилей» разд. «Организация стилей и таблиц стилей» гл. 15).

Например:

```
<style type="text/css">
  @import url(css/global.css);
  @import url(css/forms.css);
  p { color:red; }
</style>
```

С точки зрения синтаксиса нужно поместить все правила `@import` перед CSS-стилями, как показано в примере, но, если вы забудете об этом, ничего страшного не случится. Браузер должен проигнорировать любые таблицы стилей, импортируемые после CSS-стилей. Однако все существующие браузеры пренебрегают этим правилом-ограничением.

Так какой метод лучше использовать? Хоть оба они работают, использование тега `<link>` встречается чаще. В некоторых случаях правило `@import` может замедлять загрузку ваших таблиц стилей (чтобы узнать, когда и почему это происходит, посетите страницу www.stevesouders.com/blog/2009/04/09/dont-use-import). Поэтому, если у вас нет явных предпочтений одному из методов, просто используйте тег `<link>`, подробно описанный в этой главе.

Обучающий урок: создание первых стилей

В этом разделе речь пойдет об основных приемах создания CSS-стилей, в том числе встроенных, а также внутренних и внешних таблиц стилей. По мере прочтения книги на практических примерах вы научитесь создавать различные CSS-стили, от простых элементов дизайна до полноценных CSS-ориентированных макетов веб-страниц.

Перед началом урока нужно загрузить файлы с обучающим материалом, расположенные по адресу www.sawmac.com/css2e/. Перейдите по ссылке и загрузите ZIP-архив с файлами (подробное описание процесса разархивации файлов содержится на указанном сайте). Файлы каждой главы находятся в отдельных папках, названных 02 (для второй главы), 03 (для третьей) и т. д.

ВНИМАНИЕ

В описанном архиве содержатся папки с примерами для всех обучающих уроков, приведенных в этой книге. Скачайте его, чтобы в дальнейшем выполнять задания, предлагаемые в конце каждой главы.

Затем следует запустить программу редактирования веб-страниц, которой вы пользуетесь, будь то простой текстовый редактор (Блокнот или Text Edit) или программный комплекс для визуального проектирования (Dreamweaver или Microsoft Expression Web Designer) (перечень программного обеспечения приведен во введении).

ПРИМЕЧАНИЕ

Если вы пользуетесь Dreamweaver, то вам пригодится описание применения этой программы для создания стилей и таблиц стилей, приведенное в приложении 2. Dreamweaver, как и другие программы для редактирования HTML, позволяет работать с HTML-кодом в чистом виде, переключившись в режим его просмотра. Именно это и потребуется в ходе обучающего урока.

Создание встроенного стиля

Размещая стилевые команды CSS (см. разд. «Что такое стиль» этой главы) непосредственно в HTML-коде страницы, вы создаете *встроенный* стиль. Встроенные стили не обеспечивают экономии ни времени загрузки веб-страниц, ни трафика, поэтому нет никаких положительных доводов для их использования.

В крайнем случае, если обязательно нужно изменить стиль единственного элемента одной веб-страницы, можно прибегнуть к встроенным стилям. (Например, создавая HTML-форматированные электронные письма, лучше всего использовать внутренние стили. Это, к слову, единственная возможность заставить CSS работать в Gmail.) Если вы все-таки применяете этот метод и хотите, чтобы стиль работал должным образом, то уделите особое внимание размещению стилевых команд внутри тегов, которые следует отформатировать. Рассмотрим пример, наглядно демонстрирующий, как это делается.

1. В своей программе редактирования веб-страниц откройте файл `02\basic.HTML`.

   Этот простой и изящно написанный XHTML-файл содержит несколько заголовков, абзац, маркированный список и информацию об авторском праве в теге `<address>`. Начнем с создания встроенного стиля для тега `<h1>`.

2. В открывающем теге `<h1>` укажите свойство стиля `style="color: #C7AA8D;"`.

   Тег должен выглядеть следующим образом:

   ```
   <h1 style="color: #C7AA8D;">
   ```

Атрибут `style` относится к HTML, а не к CSS, поэтому после него идет знак =, а значение атрибута — весь код CSS — заключено в кавычки. Код CSS — это только та часть, что находится в кавычках. В данном случае вы добавили свойство `color`, которое воздействует на цвет текста, и установили его значение равным `#C7AA8D`, то есть шестнадцатеричному коду, определяющему серовато-коричневый цвет (об изменении цвета текста читайте в подразделе «Придание тексту цветового оформления» разд. «Стилизация текста» гл. 6). Двоеточие отделяет название свойства от значения, которое вы хотите установить для данного свойства. Далее проверим результат в браузере.

3. Откройте страницу `basic.HTML` в браузере.

   Запустите любимый браузер и выберите пункт меню **File ▸ Open** (Файл ▸ Открыть) или нажмите **Ctrl+O** и выберите файл `basic.html` в папке **02** с обучающими материалами на своем компьютере (можете просто перетащить этот файл с **Рабочего стола** — или из другого места, где вы сохраняли обучающие материалы, — в пустое окно браузера). Во многих HTML-редакторах имеется функция **Preview in Browser** (Предварительный просмотр в браузере), которая с помощью определенного сочетания клавиш или пункта меню открывает страницу для просмотра в браузере. Посмотрите руководство программы редактирования HTML: возможно, в ней есть команда, которая сэкономит ваше время. Открыв страницу в браузере, вы видите, что заголовок стал серовато-коричневым. Встроенные стили могут содержать более одного свойства CSS. Добавим в тег еще одно свойство.

4. Вернитесь в свой HTML-редактор. После точки с запятой за кодом `#C7AA8D` наберите `font-size: 3em;`.

   Точка с запятой отделяет два различных свойства. Тег `<h1>` должен выглядеть следующим образом:

   ```
   <h1 style="color: red;font-size: 3em;">
   ```

5. Посмотрите на страницу в браузере. Нажмите кнопку **Reload** (Обновить), но сначала удостоверьтесь, что сохранили XHTML-файл.

   Теперь заголовок имеет внушительный вид. Вы почувствовали, как непросто добавлять встроенные стили? Придание соответствующего вида всем заголовкам `<h1>` веб-страницы требует выполнения тех же действий над каждым из них. На это могут уйти часы и даже целые дни набора и добавления HTML-кода.

6. Вновь перейдите в редактор веб-страниц и удалите из тега `<h1>` весь код стилевого атрибута, вернув его к нормальному виду.

Далее мы создадим таблицу стилей внутри веб-страницы (окончательная версия этой части обучающего примера представлена файлом `inline.html` в папке **02_finished**).

Создание внутренней таблицы стилей

Вместо встроенных стилей лучше использовать таблицу стилей, содержащую множество стилей CSS, каждый из которых придает внешний вид своему элементу страницы. Прочитав этот раздел, вы научитесь создавать стиль, который изменяет

внешний вид всех заголовков первого уровня за один прием. Этот единственный стиль автоматически отформатирует все теги <h1> веб-страницы.

1. В файле basic.html, открытом в текстовом редакторе, установите курсор сразу после закрывающего тега </title>, нажмите клавишу **Enter** и наберите <style type="text/css">.

   Теперь HTML-код должен выглядеть следующим образом (текст, который нужно добавить, выделен полужирным шрифтом):

   ```
   <title>Большая книга CSS - Глава 2</title>
   <style type="text/css">
   </head>
   ```

   Открывающий тег <style> отмечает начало таблицы стилей. Желательно сразу же, как только вы набираете открывающий тег, закрывать его, поскольку об этом очень легко забыть, переключая внимание на написание CSS. В данном случае вы закроете тег <style> до того, как станете добавлять стили CSS.

2. Нажмите **Enter** дважды и наберите </style>.

   Теперь вы добавите селектор CSS, обозначающий начало вашего первого стиля.

3. Щелкните кнопкой мыши между открывающим и закрывающим тегами <style> и введите h1 {.

   Элемент h1 определяет сам тег, к которому браузер должен применить последующий стиль.

   Фигурную скобку после h1 называют открывающей скобкой. Она обозначает начало определения CSS-свойств для данного стиля. Иначе говоря, за ней начинается самое интересное. Надо отметить, что, как и в случае с закрывающими тегами, желательно ставить закрывающую скобку стиля до непосредственного добавления каких-либо свойств этого стиля.

4. Нажмите **Enter** дважды и поставьте одну закрывающую скобку }.

   Ответная закрывающая скобка, соответствующая введенной в предыдущем шаге открывающей, должна сообщить браузеру, что этот CSS-стиль здесь заканчивается.

5. Перейдите к пустой строке между двумя скобками, нажмите клавишу **Tab** и введите color: #C7AA8D;.

   Это текст того же свойства стиля, что и во встроенном варианте, — свойство color со значением #C7AA8D. Точка с запятой обозначает окончание объявления свойства.

ПРИМЕЧАНИЕ

Исходя из синтаксиса языка CSS, не обязательно размещать каждое свойство стиля в отдельной строке, но это в ваших же интересах. Построчный набор свойств облегчает просмотр таблицы стилей и позволяет визуально выделить каждое свойство. Кроме того, есть еще правило оформления, рекомендуемое для лучшего представления структуры CSS-кода, — это использование табуляции (вместо нее можно также добавлять несколько пробелов). Такой сдвиг текста обеспечивает быстрый и понятный просмотр таблиц стилей, выстраивая селекторы (в данном примере в качестве селектора выступает h1) в одну линию вдоль левого края страницы и располагая свойства на одинаковом уровне со смещением вправо.

6. Нажмите Enter снова и добавьте дополнительно три свойства, как показано ниже:

```
font-size: 3em;
font-family: "Arial Black", Arial, sans-serif;
margin: 0;
```

Убедитесь в том, что вы не забыли поставить точку с запятой в конце каждой строки, иначе CSS некорректно отобразится в браузере.

Каждое из этих свойств придает заголовку определенный визуальный эффект. Первые два свойства назначают размер и шрифт текста, в то время как третье удаляет отступы (пустое пространство) вокруг заголовка. Более подробно об этих свойствах читайте в части 2 книги.

Поздравляю — вы только что создали внутреннюю таблицу стилей. Код, который вы добавили на веб-страницу, должен выглядеть так, как выделенный ниже полужирным шрифтом:

```
<title>Простейшая веб-страница</title>
<style type="text/css">
  h1 {
    color: #C7AA8D;
    font-size: 3em;
    font-family: "Arial Black", Arial, sans-serif;
    margin: 0;
  }
</style>
</head>
```

7. Сохраните страницу и просмотрите ее в браузере.

Вы можете сделать это так, как описано в третьем шаге, или, если страница все еще открыта в окне браузера с предыдущего раза, просто нажмите кнопку перезагрузки страницы (Reload (Обновить)).

Теперь мы добавим другой стиль.

ПРИМЕЧАНИЕ

Никогда не забывайте добавлять закрывающий тег </style> в конце внутренней таблицы стилей. Если вы не сделаете этого, браузер отобразит на экране код стилей CSS, за которым последует сама веб-страница без всякого форматирования, а может быть и такое, что браузер вообще не покажет содержимого веб-страницы.

8. Переключитесь обратно в программу редактирования, установите курсор после закрывающей фигурной скобки стиля h1, который вы только что создали, нажмите Enter и добавьте следующий стиль:

```
p {
  color: #616161;
  line-height: 150%;
  margin-top: 10px;
  margin-left: 80px;
}
```

Этот стиль форматирует все абзацы веб-страницы. Не переживайте, что пока не знаете, что делает каждое из описываемых свойств CSS. В последующих главах эти свойства будут описаны подробно. А пока просто потренируйтесь правильно набирать код и прочувствуйте, каково это — добавлять CSS на страницу.

9. Просмотрите страницу в браузере.

   Страница начинает меняться и выглядит так, как показано на рис. 2.3. Вы видите, как изменяется стилистическая направленность абзаца под первым заголовком? Можете посмотреть окончательную версию этой части примера, открыв файл `internal.html` из папки **02_finished**.

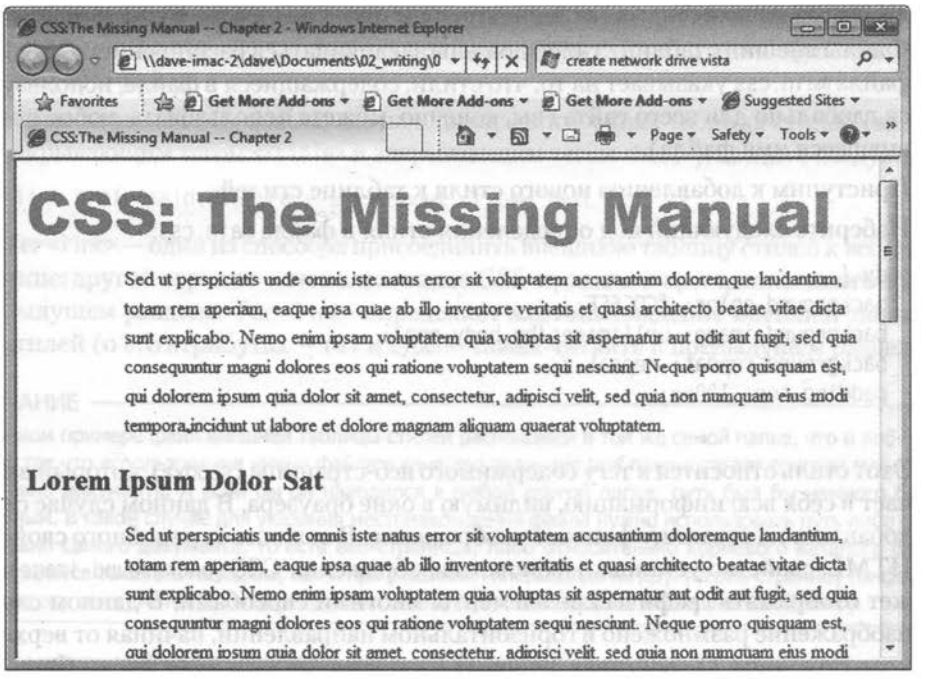


Рис. 2.3. CSS легко и творчески справляется с форматированием текста, позволяя изменять начертание, размер, цвет шрифтов текста и даже добавляя декоративные рамки и подчеркивание

Все то, чем вы занимались в обучающем уроке, можно назвать «CSS в двух словах»: начать с HTML-страницы, добавить таблицу стилей, создать прочие CSS-стили, чтобы заставить страницу прилично выглядеть. В следующей части этой обучающей программы вы увидите, как можно более эффективно работать, используя внешние таблицы стилей.

Создание внешней таблицы стилей

Поскольку во внутренних таблицах стилей все стили сгруппированы в начале веб-страницы, создавать и редактировать их намного проще и удобнее, чем встроенные

стили, с которыми вы имели дело до этого. Кроме того, внутренние таблицы стилей позволяют форматировать любое количество экземпляров тегов веб-страницы одновременно (как в примере с тегом <h1>), создав один простой стиль (правило). Внешние таблицы стилей не только наследуют преимущества внутренних таблиц, но и имеют дополнительные плюсы: в них можно хранить все стили для *всех страниц* сайта. Редактирование одного стиля во внешней таблице обновляет стиль целого сайта. В этом разделе вы возьмете стили, созданные в предыдущем уроке, и поместите их во внешнюю таблицу стилей.

1. В своей программе редактирования создайте новый файл и сохраните его под именем `main.css` в той же самой папке, где находится веб-страница, над которой вы сейчас работаете.

   Файлы внешних таблиц стилей должны заканчиваться расширением CSS. Имя файла `main.css` указывает на то, что стили, содержащиеся в файле, используются глобально для всего сайта (вы, конечно, можете использовать любое понравившееся имя файла).

   Приступим к добавлению нового стиля к таблице стилей.

2. Наберите следующий код определения стиля в файле `main.css`:

   ```
   body {
     background-color: #CDE6FF;
     background-image: url(images/bg_body.png);
     background-repeat: repeat-x;
     padding-top: 100px;
   }
   ```

   Этот стиль относится к тегу содержимого веб-страницы (<body>), который включает в себя всю информацию, видимую в окне браузера. В данном случае стиль добавляет на страницу фоновое изображение. В отличие от подобного свойства HTML, свойство установки фонового изображения CSS (`background-image`) может отображать графические элементы многими способами. В данном случае изображение размножено в горизонтальном направлении, начиная от верхнего края страницы. О свойствах фоновых изображений читайте в разд. «Фоновые изображения» гл. 8.

   Этот стиль также добавляет цвет фона страницы и отодвигает ее содержимое вниз на 100 пикселов от верхнего края окна браузера. Когда вы просматриваете страницу, вы видите, что это дополнительное пространство отодвигает заголовок от фонового рисунка.

3. Вместо повторного набора стилей, созданных в предыдущем уроке, просто скопируем стили, определенные во внутренней таблице стилей, и вставим их в эту внешнюю таблицу стилей. Откройте страницу `basic.html`, над которой работали, и скопируйте весь текст, содержащийся внутри тегов <style> (не копируйте сами теги <style>).

   Скопируйте стилевую информацию тем же самым способом, которым скопировали бы любой текст. Например, с помощью меню **Edit ▸ Copy** (Правка ▸ Копировать) или нажатием сочетания клавиш **Ctrl+C**.

4. В файл `main.css` вставьте этот код стилей либо посредством меню **Edit ▸ Paste** (Правка ▸ Вставить), либо с помощью сочетания клавиш **Ctrl+V**.

   Внешняя таблица стилей никогда не должна содержать HTML-код, именно поэтому вы и не копировали теги `<style>`.

5. Сохраните файл `main.css`.

   Теперь нужно очистить старый HTML-файл от CSS-кода и связать новую таблицу стилей с этим файлом.

6. Вернитесь к файлу `basic.html` в своем текстовом редакторе и удалите теги `<style>` и все CSS-стили, определенные в этом файле ранее.

   Вам больше не нужны эти стили внутренней таблицы, поскольку они перенесены во внешнюю таблицу стилей, которую сейчас нужно присоединить к HTML-файлу.

7. В том месте HTML-файла, где находилась встроенная таблица стилей (между закрывающим тегом `</title>` и закрывающим тегом `<head>`), введите следующее:

   ```
   <link href="main.CSS" rel="stylesheet" type="text/css" />
   ```

   Тег `<link>` — один из способов присоединить внешнюю таблицу стилей к веб-странице; другой вариант — использование CSS-правила `@import`, описанного в предыдущем разделе. Тег `<link>` определяет местонахождение внешней таблицы стилей (о его атрибутах — `rel` и `type` — также читайте в предыдущем разделе).

ПРИМЕЧАНИЕ

В данном примере файл внешней таблицы стилей расположен в той же самой папке, что и веб-страница, так что использование имени файла в качестве значения href предполагает простой путь относительно документа. А если бы он находился в любой другой папке, путь был бы немного более сложным. В таком случае для указания местонахождения файла нужно использовать путь либо относительно самого документа, то есть веб-страницы, либо относительно корневого каталога сайта. Применяется такая же методика, как и при указании гиперссылки на другую веб-страницу (информацию на эту тему смотрите по адресу www.communitymx.com/content/article.cfm?cid=230AD).

8. Сохраните файл и просмотрите его в браузере.

   Веб-страница должна выглядеть так, как описано в шаге 9 предыдущего раздела обучающего урока, но с добавлением синего фона и картинки с травой и цветами вдоль верхнего края страницы (благодаря CSS-коду, введенному в шаге 2). CSS-стили в этой внешней таблице стилей — те же самые, что были во внутренней таблице стилей, только они расположены в другом месте. Чтобы продемонстрировать, насколько удобнее и лучше вариант хранения CSS-стилей в отдельном файле внешней таблицы стилей, нужно присоединить созданную внешнюю таблицу стилей к другой веб-странице.

ПРИМЕЧАНИЕ

Если на получившейся веб-странице отсутствует форматирование (к примеру, заголовок маленький и никак не выделен), то, вероятно, вы набрали код в шаге 6 с ошибкой или сохранили файл main.css в папке, отличной от той, в которой находится файл basic.html. В этом случае просто переместите файл main.css в ту же самую папку.

9. Откройте файл 02\another_page.html.

   Эта веб-страница содержит те же самые HTML-элементы: H1, H2, P и др., с которыми вы уже работали.

10. Установите курсор после закрывающего тега </title> и нажмите клавишу Enter.

    Сейчас нужно присоединить к этой веб-странице уже созданную внешнюю таблицу стилей.

11. Наберите тот тег <link>, который применялся в седьмом шаге этого примера.

    Код веб-страницы должен выглядеть следующим образом (код, который вы только что набрали, отмечен полужирным шрифтом):

```
<title>Другая страница</title>
<link href="main.CSS" rel="stylesheet" type="text/css" />
</head>
```

12. Сохраните страницу и просмотрите ее в браузере.

    Достаточно всего одной строки кода, добавленной в веб-страницу, чтобы мгновенно преобразить ее внешний вид. Чтобы показать, насколько просто обновить стиль, описанный во внешней таблице стилей, попробуйте отредактировать один из стилей или добавить другие.

13. Откройте файл main.css и добавьте определение CSS-свойства — font-family: "Palatino Linotype", Baskerville, serif; — в начале стиля элемента P.

    Код должен выглядеть следующим образом (добавленный код выделен полужирным шрифтом):

```
p {
font-family: "Palatino Linotype", Baskerville, serif;
color: #616161;
line-height: 150%;
margin-top: 10px;
margin-left: 80px;
}
```

    Напоследок создайте новый стиль для элемента H2.

14. Установите курсор после заключительной фигурной скобки } стиля элемента P, нажмите Enter и добавьте следующий стиль:

```
h2 {
  color: #B1967C;
  font-weight: normal;
  font-family: "Palatino Linotype", Baskerville, serif;
  font-size: 2.2em;
  border-bottom: 2px white solid;
  background: url(images/bullet_flower.png) no-repeat;
  padding: 0 0 2px 80px;
  margin: 0;
}
```

С большинством этих CSS-свойств вы уже знакомы, однако некоторые из них новые для вас, например `border-bottom`, используемое для добавления линии под заголовком. Свойство `background` вообще предоставляет возможность для комбинирования различных свойств — в данном случае это `background-image` и `background-repeat` — в одно. Не беспокойтесь о назначении этих свойств, вы изучите их подробнейшим образом в последующих главах. О свойствах шрифта читайте в гл. 6, о свойствах, устанавливающих отступы и границы, — в гл. 7, а о свойствах, устанавливающих параметры фона, — в гл. 8.

Стили, которые вы создавали до сих пор, работают с основными элементами `H1`, `H2` и `P`, изменяя облик каждого их экземпляра. Другими словами, стиль `P`, который вы создали, форматирует каждый абзац на странице. Но если вы хотите воздействовать на какой-то конкретный абзац, нужно использовать другой вид стиля.

15. Перейдите к концу стиля `H2`, нажмите **Enter** после закрывающей скобки } и наберите следующий код:

```
.intro {
  color: #6A94CC;
  font-family: Arial, Helvetica, sans-serif;
  font-size: 1.2em;
  margin-left: 0;
  margin-bottom: 25px;
}
```

Если вы просмотрите файл `basic.html` в браузере, то пока не заметите никаких изменений. Этот тип стиля, называемый *селектором класса*, форматирует только отдельные теги, к которым вы примените класс. Для того чтобы этот новый стиль работал, придется немного отредактировать HTML-код.

16. Сохраните файл `main.css` и перейдите к файлу `basic.html` в своем текстовом редакторе. Найдите открывающий тег `<p>`, следующий за тегом `<h1>`, и добавьте `class="intro"`. Открывающий тег должен выглядеть следующим образом:

```
<p class="intro">
```

Вам не нужно добавлять точку перед словом `intro`, как вы это делали, создавая стиль в шаге 15 (почему так, вы узнаете в следующей главе). Этот дополнительный код HTML применяет стиль к первому абзацу (и только к первому).

Повторите этот шаг для файла `another_page.html` — добавьте `class="intro"` к первому тегу `<p>` на этой странице.

17. Сохраните все файлы и просмотрите страницы `basic.html` и `another_page.htm` в браузере. На рис. 2.4 представлен внешний вид страницы `another_page.htm` в окончательном виде.

Обратите внимание, что внешний вид обеих веб-страниц определяется единственным CSS-файлом с внешней таблицей стилей. Простым редактированием файла `global.css` вы можете изменить стиль сразу обеих веб-страниц. Закройте глаза и представьте, что ваш сайт содержит тысячи страниц. Мощные средства изменения дизайна, не правда ли?

Окончательную версию этой части обучающего примера вы найдете в папке 02_finished.

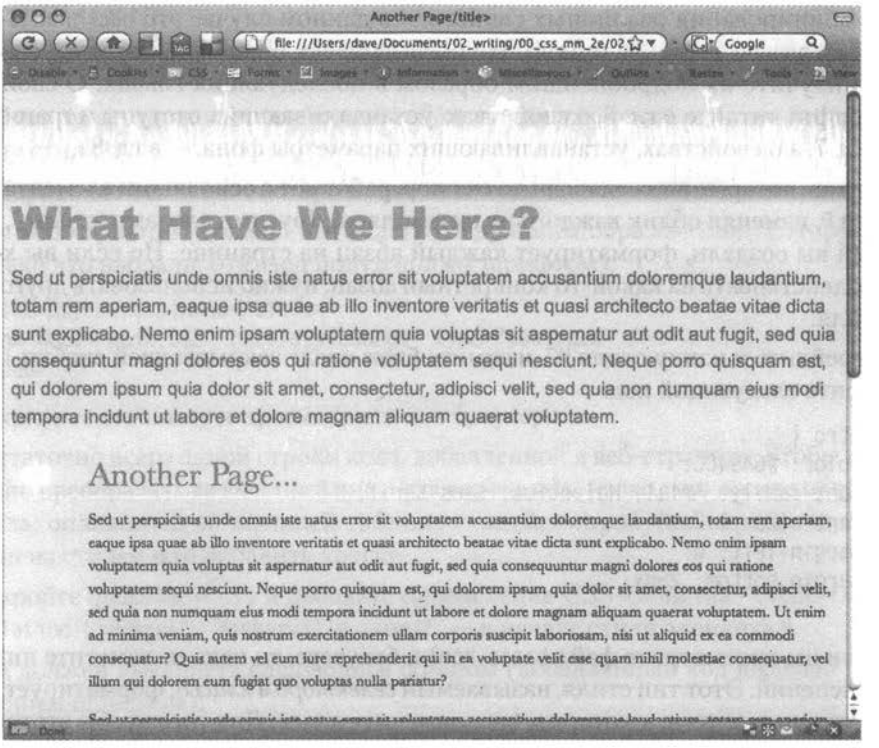


Рис. 2.4. Применение внешних таблиц стилей дает возможность обновления дизайна всех страниц сайта в один прием посредством редактирования единственного CSS-файла

Полное исключение CSS-кода из HTML-документа и вынесение его в отдельный файл позволяет значительно уменьшить размер самих веб-страниц, следовательно, они будут загружаться намного быстрее.

3 Селекторы: определение элементов стилизации

Каждый CSS-стиль имеет две основные части: селектор и блок объявления (о нем говорилось в предыдущей главе), который, в свою очередь, содержит форматирующие свойства — цвет, размер шрифта текста и т. д. Они относятся лишь к оформлению. Волшебство CSS заключается как раз в самых первых символах, начинающих определение любого стиля, — селекторах. Например:

```
h1 {
  font-family: Arial, sans-serif;
  color: #CCCCFF;
}
```

Здесь первая часть стиля — селектор — определяет элементы, подлежащие форматированию. В данном случае `h1` означает «все заголовки первого уровня (тег <h1>) веб-страницы».

ПРИМЕЧАНИЕ

Как видно из примера, в имена не входят символы < и >, в которые заключены соответствующие HTML-теги. Поэтому, например, когда вы пишете стиль для тега <p>, набирайте только название — p.

Именно селектор контролирует дизайн веб-страницы, определяя элемент, который вы хотите изменить. Другими словами, именно он используется для форматирования множества элементов одновременно. Если дать более подробное описание, то селекторы позволяют выбрать один или несколько схожих элементов для их последующего изменения с помощью свойств в блоке объявления. Селекторы CSS — большой потенциал для создания дизайна веб-страниц.

ПРИМЕЧАНИЕ

Если вы хотите немного попрактиковаться на примерах, прежде чем изучать теоретический материал по данной теме селекторов, обратитесь к обучающему уроку этой главы.

Селекторы типов: дизайн страницы

Селекторы типов (иногда называют *селекторами тегов*) представляют весьма эффективное средство проектирования дизайна веб-страниц, поскольку определяют стиль всех экземпляров конкретного HTML-элемента.

С их помощью можно быстро изменять дизайн веб-страницы с небольшими усилиями. Например, если надо отформатировать все абзацы текста, используя шрифт одного начертания, размера, а также цвета, то вам просто нужно создать стиль с селектором P (применительно к тегу <p>). Он переопределяет, каким образом браузер отобразит отдельно взятый тег (в данном случае <p>).

Еще до появления CSS, чтобы отформатировать текст, вам пришлось бы заключить его по всем абзацам в тег <font> многократно. Этот процесс занял бы много времени, не говоря о том, что код HTML-страниц при этом увеличится в объеме до неимоверных размеров, страницы будут загружаться медленнее, их обновление будет занимать много времени. С селекторами типов вам фактически ничего не нужно делать с HTML-кодом, просто создайте CSS-стили и позвольте браузеру сделать все остальное.

Селекторы исключительно просто определить в CSS-стилях, так как они наследуют название форматируемых элементов — P, H1, TABLE, IMG и т. д. Например, страница, представленная на рис. 3.1, имеет три тега <h2> (помечены на левой границе окна браузера). Единственный CSS-стиль с селектором h2, приведенным ниже, определяет представление каждого заголовка второго уровня страницы.

```
h2 {
  font-family:"Century Gothic", "Gill Sans", sans-serif;
  color:#000000;
  margin-bottom:0;
}
```



Рис. 3.1. Селектор тега воздействует на все экземпляры указанного элемента веб-страницы

Однако и здесь имеются свои недостатки. Как сделать, чтобы не все абзацы веб-страницы выглядели одинаково? Простыми селекторами типов этого добиться не

удастся, потому что они не предоставляют достаточную информацию браузеру. Например, нужно задать различия между элементом P, выделенным определенным цветом и кеглем, и элементом P, который вы хотите оставить написанным шрифтом черного цвета. CSS предоставляет сразу несколько способов решения данной проблемы, самый простой из которых — селекторы классов.

Селекторы классов: точное управление

Если вы хотите, чтобы какие-то элементы выглядели не так, как другие родственные им элементы из той же веб-страницы, например, хотите задать для одного или двух рисунков красную рамку, оставив все остальные изображения нестилизованными, то можете использовать *селектор классов*. Если вы привыкли работать со стилями в программах для редактирования текста, таких как Microsoft Word, то селекторы классов покажутся вам хорошо знакомыми. Вы создаете селектор, назначая ему имя, а затем применяете его лишь к тем тегам HTML, которые хотите отформатировать.

Например, вы можете создать класс `.copyright` и с его помощью выделить абзац, содержащий информацию об авторских правах, не затрагивая остальные абзацы.

Селекторы классов позволяют указать конкретный элемент веб-страницы, независимо от тегов. Предположим, вы хотите отформатировать одно или несколько слов абзаца. В данном случае применяется форматирование не ко всему тегу <p>, а лишь к фрагменту абзаца. Таким образом, вам нужно использовать селектор класса для выделения определенного текста. Можно применить изменения к множеству элементов, входящих в различные HTML-теги.

Например, вы можете придать какому-то абзацу и заголовку второго уровня (тег <h2>) одинаковый стиль, как показано на рис. 3.2. В отличие от селекторов типов, которые ограничивают вас существующими на веб-странице HTML-тегами, с помощью этого метода вы можете создать любое количество селекторов классов и поместить их в выбранное место.

ПРИМЕЧАНИЕ

Вы можете стилизовать один экземпляр заголовка <h2> («Wet Sod is Heavy Sod»). Селектор класса .special сообщает браузеру о необходимости применения стиля к единственному тегу <h2>. Создав его однажды, вы можете пользоваться им и в дальнейшем применительно к любым тегам. В примере (см. рис. 3.2) такой стиль применен к верхнему абзацу.

Если вы хотите применить селектор класса всего к нескольким словам текста, содержащегося в произвольном теге HTML-кода (подобно среднему абзацу на рис. 3.2), то используйте тег . Для более детального ознакомления смотрите ниже врезку «Бриллиант без огранки».

Вы, вероятно, обратили внимание на точку, с которой начинается каждое название селектора класса: `.copyright`, `.special`. Это одно из нескольких правил, которые необходимо иметь в виду, именуя классы.

- Все названия селекторов классов должны начинаться с точки. С ее помощью браузеры находят селекторы классов в таблице стилей CSS.
- При именовании стилевых классов разрешается использование только букв алфавита, чисел, дефисов, знаков подчеркивания.

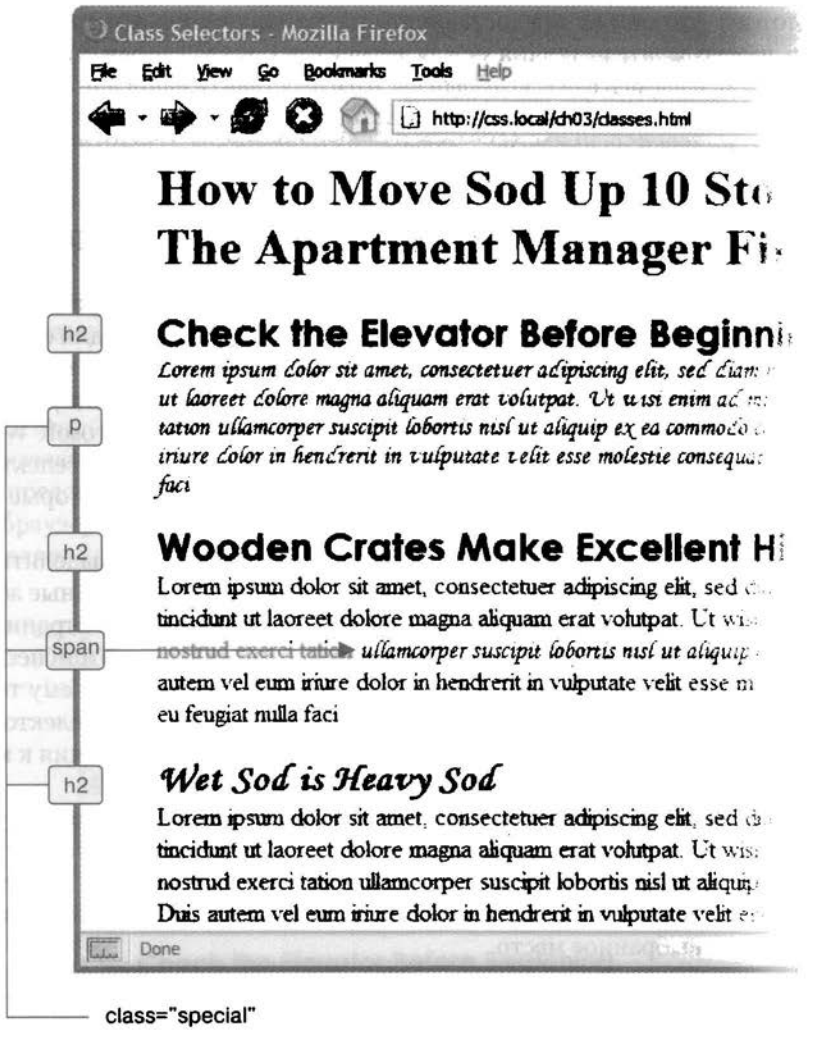


Рис. 3.2. Селекторы классов позволяют целенаправленно изменять дизайн фрагментов веб-страниц

- Название после точки всегда должно начинаться с символа — буквы алфавита. Например, `.9lives` — неправильное имя класса, а `.crazy8` — правильное. Можно называть классы, например, именами `.copy-right` и `.banner_image`, но не `.-bad` или `_as_bad`.
- Имена стилевых классов чувствительны к регистру. Например, `.SIDEBAR` и `.sidebar` рассматривается языком CSS по-разному, как различные классы.

```
.special {
  color:#FF0000;
  font-family:"Monotype Corsiva";
}
```

Классы описываются так же, как стили тегов. После названия идет блок объявления, содержащий все необходимые свойства:

```
.special {
  color:#FF0000;
  font-family:"Monotype Corsiva";
}
```

Поскольку стили тегов распространяются на все типы веб-страницы, их достаточно определить в заголовке, и они начинают работать. Форматируемые HTML-теги уже вписаны в веб-страницу. Чтобы воспользоваться преимуществами, которые обеспечивают стилевые классы, требуется выполнить еще несколько действий: использование селекторов классов — двухступенчатый процесс. После того как вы создали класс, надо указать, *где* вы хотите его применить. Для этого добавьте атрибут `class` к HTML-тегу, который нужно стилизовать.

Допустим, вы создали класс `.special` с целью выделения определенных элементов веб-страницы. Чтобы применить этот стиль к абзацу, добавьте атрибут `class` к тегу `<p>`, как показано ниже:

```
<p class="special">
```

ПРИМЕЧАНИЕ

Вы не должны набирать точку перед именем класса в коде HTML (в атрибуте class). Она требуется только в названии селектора таблицы стилей.

Таким образом, когда браузер сталкивается с этим тегом, он знает, что правила форматирования, содержащиеся в стиле `.special`, необходимо применить к данному тексту. Вы можете также использовать класс только в части абзаца или заголовка, добавив `<span>`, как описано ниже.

Создав класс, можно применить его практически к любому тегу веб-страницы. Вообще, вы можете применять один и тот же класс к разным тегам, создав, к примеру, стиль `.special` с особым шрифтом и цветом для тегов `<h2>`, `<p>` и `<ul>`. Однако, хотя классы и предоставляют почти безграничные возможности форматирования, все-таки они не всегда являются верным инструментом CSS для создания дизайна веб-страницы. Иногда лучше пользоваться ID-селектором, который позволяет определить форматирующие правила для их однократного применения на странице.

ID-селекторы: определение элементов веб-страниц

В языке CSS ID-селектор предназначен для *идентификации* уникальных частей веб-страниц, таких как шапка, панель навигации, основная информационная область содержимого и т. д. Как и при использовании селектора класса, вы создаете идентификатор (ID), придумав ему название, а затем применяете его к HTML-коду своей веб-страницы. Так в чем же различие? Как описано во врезке «Бриллиант без огранки», ID-селекторы имеют дополнительное специфическое применение.

Например, в веб-страницах с использованием кода JavaScript или в очень объемных страницах.

Другими словами, существует несколько вынужденных причин для использования ID-селекторов.

Чтобы точно решить, что использовать — классы или идентификаторы, — следуйте основным правилам.

- Для стиля, используемого неоднократно, применяют классы. Если на веб-странице есть несколько рисунков, то необходимо применить селектор класса (например, задающий такой стилевой эффект, как рамка) к каждому из тегов `<img>`, которые вы хотите отформатировать.
- Пользуйтесь идентификаторами, когда необходимо определить разделы веб-страницы, которые встречаются один раз. Если их дизайн основан на CSS, то часто используются ID-селекторы, чтобы определить такие уникальные части, как боковые меню, низ страницы. В части 3 настоящей книги описывается использование этого метода.
- Примите во внимание то, что решить конфликты стилей позволяет использование ID-селекторов, которым браузеры придают больший приоритет, чем классовым. Например, когда попадаются два различных стиля, относящихся к одному и тому же тегу, но определяющих, допустим, различный цвет фона. В этом случае применяется свойство стиля с ID-селектором, как имеющее более высокий приоритет (см. гл. 5).

ПРИМЕЧАНИЕ

Каждый идентификатор стиля должен относиться к определенному HTML-тегу. Хотя, если будет применен один и тот же идентификатор для двух или более тегов на одной веб-странице, это не будет причиной аварийного завершения работы. Большинство браузеров правильно обрабатывает CSS-код стиля с таким ID-селектором. Однако HTML-код вашей веб-страницы не пройдет синтаксическую проверку правильности (валидацию), и дизайнеры по проектированию не смогут понять, что от них требуется.

БРИЛЛИАНТ БЕЗ ОГРАНКИ

**HTML-теги и **

В гл. 1 были кратко описаны `<div>` и `<span>` — два универсальных HTML-тега, которые однозначно указывают на применение класса или стиля с ID-селектором.

Логическое деление страницы на такие фрагменты, как шапка, панель навигации, боковые меню, низ страницы, обеспечивает тег `<div>`. Вы можете его также использовать, чтобы охватить любой фрагмент, включающий ряд последовательных элементов веб-страницы, в том числе заголовки, маркированные списки, абзацы (программисты называют эти элементы блочными, потому что они формируют логически законченный блок информационного содержимого с разрывами строк до и после такого блока). Тег `<div>` функционирует как тег абзаца: набираем открывающий `<div>`, далее добавляем некоторый текст, рисунок или какое-то другое информационное содержимое, а затем заканчиваем их закрывающим `</div>`.

Тег `<div>` имеет уникальную способность включать в себя несколько блочных элементов, являясь средством группировки (логотип и панель навигации или блок новостей, формирующий боковой столбец).

БРИЛЛИАНТ БЕЗ ОГРАНКИ

После группировки содержимого веб-страницы таким образом вы можете применить определенное форматирование исключительно к тегам, находящимся внутри этого фрагмента , или переместить (позиционировать) весь отмеченный фрагмент содержимого в конкретное место веб-страницы, например в правый столбец. Об управлении разделами в CSS этим способом рассказывается в части 3 настоящей книги.

Например, вы добавили на веб-страницу фотографию, у которой есть сопроводительное описание. Их можно заключить в (с применением к тегу класса), чтобы объединить в группу оба элемента:

```
<div class="photo">
  <img src="holidays.jpg"
    alt="Penguins getting frisky"/>
  <p>Mom, dad and me on our yearly trip
    to Antarctica.</p>
</div>
```

В зависимости от того, как вы опишете стиль, класс .photo может добавить декоративную рамку, цветной фон и т. д. сразу ко всему блоку, включающему и фотографию, и ее описание. В части 3 книги описываются еще более мощные способы применения , включая вложенные конструкции из этих тегов.

С другой стороны, позволяет применять классы и ID-селекторы к фрагменту — части тега. С его помощью вы можете выхватить из абзаца отдельные слова и фразы (которые часто называются встроенными (inline) элементами), чтобы форматировать их отдельно друг от друга.

В данном примере стилевой класс, который назван .companyName, стилизует встроенные элементы CosmoFarmer.com, Disney и ESPN:

```
<p>Welcome to <span class="companyName">
  CosmoFarmer.com</span>, the parent
  company of such well-known corporations
  as <span class="companyName">Disney
  </span> and <span class="companyName">
  ESPN</span>...well, not really.
</p>
```

Использование ID-селекторов не представляет никаких сложностей. Если в селекторах классов перед названием ставится точка (.), то тут вначале должен быть указан символ решетки (#). Во всем остальном руководствуйтесь теми же правилами, что и для классов (см. выше).

Следующий пример устанавливает цвет фона, ширину и высоту элемента:

```
#banner {
  background: #CC0000;
  height: 300px;
  width: 720px;
}
```

Применение идентификаторов в HTML схоже с использованием классов, но требует другого атрибута с соответствующим названием — id. Для указания, что последний абзац страницы — единственный с информацией об авторских правах, вы можете создать ID-стиль под названием #copyright и применить его к тегу этого абзаца:

```
<p id="copyright">
```

ПРИМЕЧАНИЕ

Символ решетки (#) используется только при описании стиля в таблице. При вставке же имени идентификатора в HTML-теги # указывать не нужно: .

ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

Специальное применение идентификаторов

У ID-селекторов имеется несколько преимуществ перед селекторами классов. Эти особенности фактически не имеют никакого отношения к CSS, поэтому могут вам и не понадобиться.

В JavaScript-программировании ID-селекторы используются для определения местонахождения и управления фрагментами веб-страниц. Специалисты часто применяют идентификаторы к заполняемым элементам форм (например, текстовые поля) для получения имени посетителя сайта и т. д. Это позволяет JavaScript получить доступ к полям форм, проверить их содержимое.

Идентификаторы также позволяют делать ссылки на определенные части веб-страниц, при этом быстро перемещаясь по ним. Если у вас есть алфавитный словарь терминов, можно использовать ID-селектор для создания указателя по буквам алфавита. При щелчке кнопкой мыши на букве R посетители сразу же переходят к словам, начинающимся с этой буквы. Для этого вам не придется использовать CSS — достаточно и простого HTML. Сначала добавьте атрибут `id` в то место на странице, на которое вы хотите ссылаться. Например, в указателе вы можете добавить тег `<h2>` с очередной буквой из алфавита. Добавьте соответствующий `id` к каждому тегу `<h2>`: `<h2 id="R">R</h2>`. Чтобы создать ссылку в HTML, добавьте символ решетки и имя идентификатора в конец адреса URL — `index.html#R`. Эта ссылка указывает непосредственно на элемент с `#R` страницы `index.html` (использование идентификатора таким способом оказывает действие, аналогичное применению якорного тега `<a>` в языке HTML: `<a name="R">R</a>`).

Стилизация групп тегов

Иногда необходимо быстро применить одинаковое форматирование сразу к нескольким различным элементам веб-страницы. Например, нужно, чтобы все заголовки имели один и тот же цвет и шрифт. Создание отдельного стиля для каждого заголовка определенного уровня — слишком объемная работа. А если вы потом захотите изменить цвет всех заголовков одновременно, то придется все обновлять. Для решения этой задачи лучше использовать групповые селекторы, которые позволяют применить стиль, описав его лишь один раз, одновременно ко всем элементам.

Создание групповых селекторов

Для работы с группой селекторов создайте список, в котором один селектор отделен от другого запятыми. Таким образом, получаем:

```
h1, h2, h3, h4, h5, h6 { color: #F1CD33; }
```

Здесь приведены только селекторы типов, но можно использовать любые виды (или их сочетание). Например, стиль с групповым селектором, который определяет одинаковый цвет шрифта для `<h1>`, `<p>` и любых других, принадлежащих классу `.copyright`, а также тегу с идентификатором `#banner`.

```
h1, p, .copyright, #banner { color: #F1CD33; }
```

ПРИМЕЧАНИЕ

Если вам нужно применить к нескольким элементам веб-страницы одинаковые и в то же время несколько различные форматирующие свойства, то можете создать групповой селектор с общими командами и отдельные стили с уникальным форматированием для каждого индивидуального элемента. Другими словами, два (или больше) стиля могут форматировать один и тот же тег. Это и является мощной особенностью языка CSS (информацию по этой теме см. в гл. 5).

Универсальный селектор

Групповые селекторы можно рассматривать как подручное средство для применения одинаковых свойств различных элементов. CSS предоставляет *универсальный* селектор * для выборки *всех* тегов веб-страницы. Например, если вы хотите, чтобы все отображалось полужирным шрифтом, нужно добавить следующий код:

```
a, p, img, h1, h2, h3, h4, h5 ..... { font-weight: bold; }
```

Использование символа * — более быстрый способ сообщить CSS о выборке *всех* HTML-тегов веб-страницы:

```
* { font-weight: bold; }
```

Кроме того, вы можете использовать универсальный селектор в составе селектора потомков: применяете стиль ко всем тегам-потомкам, подчиненным определенному элементу веб-страницы. Например, #banner * выбирает все теги внутри элемента, имеющего идентификатор #banner (более подробно о селекторах потомков вы узнаете чуть позже).

Универсальный селектор не определен как тип тегов, поэтому трудно описать его воздействие на HTML-теги сайта. По этой причине дизайнеры со стажем используют для форматирования различных элементов такую особенность языка CSS, как *наследование*. Она описана в следующей главе.

Стилизация вложенных тегов

Выбор типа селекторов обусловлен в каждом случае конкретной целью. Селекторы типов можно быстро и просто создать, но они придают всем экземплярам стилизуемого элемента одинаковый внешний вид. Это бывает необходимо, например, если нужно, чтобы все заголовки второго уровня вашей веб-страницы выглядели одинаково. Классовые и ID-селекторы обеспечивают большую гибкость независимой стилизации отдельно взятых элементов страницы. Однако создание нового CSS-стиля всякий раз, когда требуется изменить шрифт лишь одного заголовка, отнимает гораздо больше времени, требует большего объема работы и применения кода CSS и HTML. Все, что нужно в этом случае, — объединить простоту использования селекторов типов с точностью селекторов классов и идентификаторов.

В CSS можно определять *наследуемые селекторы*. Ими пользуются для того, чтобы единообразно отформатировать целый набор тегов, но только когда они расположены в определенном контексте — фрагменте веб-страницы. Их работу

можно описать так: «Нужно отформатировать все элементы A, находящиеся на панели управления. С остальными такими элементами ничего не нужно делать».

Наследуемые селекторы позволяют форматировать теги в зависимости от их связей с другими тегами. Чтобы разобраться, как они работают, придется немного покопаться в языке HTML.

Понимание работы наследуемых селекторов поможет получить представление о функционировании других типов, работа которых описана далее.

ПРИМЕЧАНИЕ

Наследуемые селекторы могут показаться немного запутанными на первый взгляд, однако это одна из наиболее важных техник для эффективного и точного применения CSS. Запаситесь терпением, и вы скоро освоите их.

Дерево HTML

Код HTML, на котором написана любая веб-страница, похож на генеалогическое дерево. Первый используемый HTML-тег — <html> — похож на главного прародителя всех остальных. Из него выходят <head> и <body>, следовательно, <html> является *предком* (прародителем) названных тегов.

Тег, расположенный внутри другого, — его *потомок*. Тег <title> в следующем примере — потомок <head>:

```
<html>
  <head>
    <title>Простой документ</title>
  </head>
  <body>
    <h1>Заголовок</h1>
    <p>Абзац с <strong>важным </strong>текстом.</p>
  </body>
</html>
```

Представленный выше HTML-код можно изобразить в виде схемы (рис. 3.3). Здесь показаны отношения между тегами веб-страницы. Сначала идет <html>; от него ответвляются два раздела, представленные <head> и <body>. Они содержат и другие, которые, в свою очередь, могут включать еще теги. Таким образом, рассматривая, какие из них являются вложенными, можно схематически изобразить любую веб-страницу.

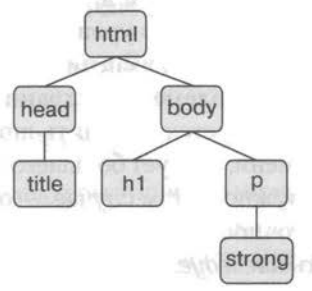


Рис. 3.3. HTML состоит из вложенных тегов, которые представляют собой своего рода генеалогическое дерево

Схемы в форме дерева помогают выяснить и проследить, как CSS «видит» взаимодействие элементов веб-страницы. Функционирование многих селекторов, описанных в настоящей главе, включая наследуемые, основывается на их родственных отношениях. Рассмотрим самые важные из них.

- **Предок.** Как описано в начале данного раздела, HTML-элемент, который заключает в себе другие элементы, — это предок. На рис. 3.3 это <html>, в то время как <body> — предок для всех содержащихся в нем тегов: <h1>, <p>, .
- **Потомок.** Элемент, расположенный внутри одного или более типов, — потомок. На рис. 3.3 <body> — потомок <html>, в то время как <p> — потомок *одновременно* и для <body>, и для <html>.
- **Родительский элемент.** Он связан с другими элементами более низкого уровня и находится выше на дереве. На рис. 3.3 <html> является родительским только для <head> и <body>. Тег <p> — родительский по отношению к .
- **Дочерний элемент.** Элемент, непосредственно подчиненный другому элементу более высокого уровня, является дочерним. На рис. 3.3 оба тега — <h1> и <p> — дочерние по отношению к <body>, но не является дочерним для <body>, так как он расположен непосредственно внутри <p>.
- **Сестринский элемент.** Элементы, являющиеся дочерними для одного и того же родительского тега, называются *сестринскими* или элементами одного уровня. На рис. 3.3 они расположены рядом друг с другом. Теги <head> и <body> — элементы одного уровня, как и <h1> и <p>.

На этом в CSS «родственные отношения» заканчиваются.

Создание селекторов потомков

Селекторы потомков позволяют использовать дерево HTML, форматируя теги по-разному, в зависимости от того, где они расположены. Например, на веб-странице имеется <h1> и вы хотите выделить слово внутри этого заголовка с помощью тега . Проблема в том, что большинство браузеров отобразит все это полужирным шрифтом, поэтому всякий, кто просматривает веб-страницу, не сможет увидеть разницы между выделенным словом и другими частями заголовка. Создание селектора тега — не очень хорошее решение: так вы измените цвет всех вхождений тега веб-страницы. Селектор же потомков позволит вам изменить цвет тега только в том случае, если он расположен внутри заголовка первого уровня.

Решение проблемы выглядит следующим образом:

```
h1 strong { color: red; }
```

В данном случае любой текст тега , находящегося внутри тега <h1>, будет выделен красным цветом, но на другие экземпляры веб-страницы этот стиль не повлияет. Вы могли добиться того же результата, создав класс стиля, например .StrongHeader. Но в таком случае понадобилось бы вносить изменения в HTML, добавляя новый класс к тегу внутри заголовка. Подход, основанный на наследуемых селекторах, позволяет обойтись без лишней работы при создании стилей.

Селекторы потомков стилизуют вложенные элементы веб-страницы, следуя тем же правилам, которым подчиняются теги-предки и теги-потомки в дереве HTML.

Вы создаете селектор потомков, объединяя селекторы вместе (согласно ветви дерева, которую нужно отформатировать), помещая самого старшего предка слева, а форматируемый тег — справа. На рис. 3.4 обратите внимание на три ссылки (<a>) — элементы маркированного списка, и еще одну, расположенную внутри абзаца. Чтобы отформатировать их иначе, вы можете создать стиль с селектором потомков:

```
li a { font-family: Arial; }
```

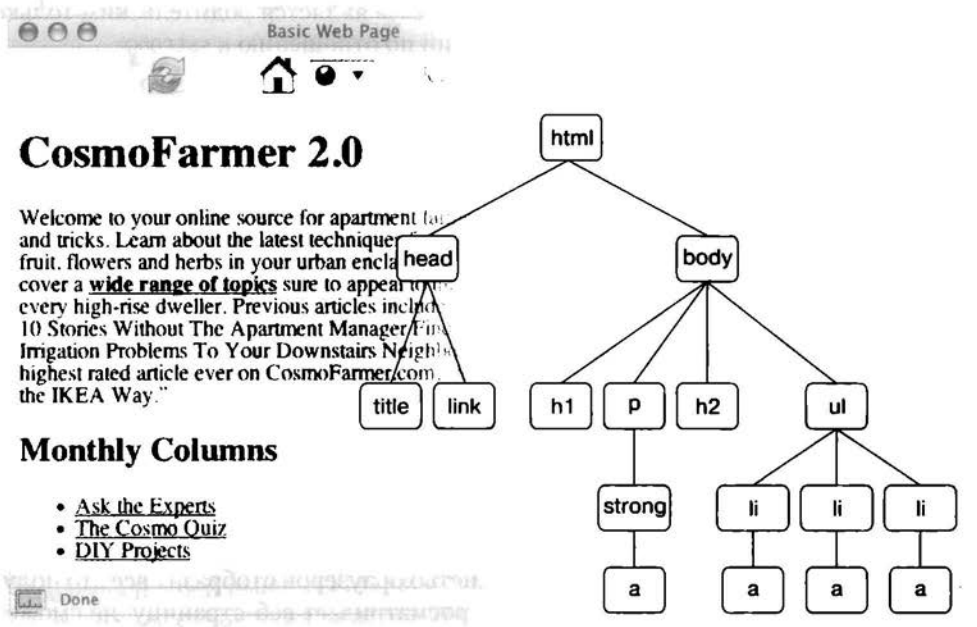


Рис. 3.4. Простейшая диаграмма в виде дерева, представляющая структуру веб-страницы

На рисунке все теги веб-страницы — потомки (производные) <html>. Тег может происходить от множества других. Например, первый тег <a> диаграммы является потомком , <p>, <body> и <html>.

Этот стиль говорит: «Нужно отформатировать все ссылки (a), которые расположены в элементах списка (li), используя шрифт Arial». Вообще, наследуемые селекторы могут включать более двух элементов. Ниже представлены правильные определения для тегов <a>, находящихся в маркированных списках (см. рис. 3.4):

```
ul li a
body li a
html li a
html body ul li a
```

ПРИМЕЧАНИЕ

Несмотря на то что лучше стараться использовать как можно более короткие селекторы потомков, одна из причин, по которой удобно использовать дополнительные селекторы потомков, — возможность замены нескольких различных стилей, одновременно форматирующих один и тот же тег. Команды могут переопределить стили классов или тегов. Подробнее об этом читайте в следующей главе.

Все четыре селектора, представленные выше, имеют один и тот же эффект, что лишний раз демонстрирует отсутствие необходимости полностью описывать всех предков тега, который вы хотите отформатировать. Например, во втором примере (`body li a`) определение `ul` не требуется. Этот селектор выполняет свои функции, если есть `<a>`, являющийся потомком `<li>`. Его одинаково просто применить как к `<a>`, расположенному внутри `<em>`, так и к `<a>` в `<strong>` или `<li>` и т. д.

В любом случае вы не ограничены только селекторами типов. Можно комбинировать различные типы, чтобы ссылки были окрашены в желтый цвет, когда они находятся в тексте введения (которое вы определили стилевым классом `intro`). Этого можно добиться с помощью следующего селектора:

```
p.intro a { color: yellow: }
```

ПРИМЕЧАНИЕ

Нужно применить данный стиль ко всем ссылкам (a) — потомкам абзаца (p), относящегося к классу с именем intro. Обратите внимание, что между p и intro нет пробелов. Это говорит CSS о том, что класс intro должен сочетаться конкретно с тегом `<p>`.

Если вы добавите пробел, получится совсем другой эффект:

```
p .intro a { color: yellow: }
```

Данная корректировка селектора, на первый взгляд весьма незначительная, сильно повлияет на область действия стиля. Будут выбраны `<a>`, расположенные внутри любого тега класса `.intro`, который сам должен являться потомком `<p>`.

Если из названия селектора убрать имя тега-предка (в данном случае `p`), то получится более гибкий стиль:

```
.intro a {color: yellow: }
```

Этот селектор указывает на любой тег `<a>`, расположенный внутри других тегов — `<div>`, `<h1>`, `<table>` и т. д., обозначенных классом `.intro`.

Селекторы потомков — очень мощное оружие в арсенале CSS. Периодически в книге будут описаны другие способы их использования, а в гл. 15 вы найдете раздел, посвященный эффективным методикам их применения.

Псевдоклассы и псевдоэлементы

Иногда требуется выбрать фрагмент веб-страницы, в котором вообще нет тегов, но в то же время его достаточно просто идентифицировать. Это может быть первая строка абзаца или ссылка, на которую наведен указатель мыши. CSS предоставляет для этих целей *псевдоклассы* и *псевдоэлементы*.

Стилизация ссылок

Существует четыре псевдокласса, которые позволяют форматировать ссылки, в зависимости от того, какое действие над ними выполняет посетитель веб-страницы.

- `a:link` — обозначает любую ссылку, по которой посетитель веб-страницы еще не переходил, даже если на нее не наведен указатель мыши. Это обычный стиль непосещенных гиперссылок.
- `a:visited` — является ссылкой, по которой посетитель веб-страницы уже переходил. Она сохраняется в истории браузера. Можно разработать для этого типа стиль, отличный от обычного, чтобы сказать посетителю, что он уже посещал эту страницу.
- `a:hover` — позволяет изменять вид ссылки, на которую посетитель навел указатель мыши. Вы можете добавить визуальные эффекты трансформации (переходов), которые служат для улучшения визуального восприятия, например, кнопки панели навигации.
- Кроме того, можно использовать псевдокласс `:hover` для применения стилей к элементам веб-страниц, отличным от ссылок. Например, вы можете использовать его для выделения фрагмента текста, заключенного в теги `<p>` или `<div>`, каким-либо стилевым эффектом в тот момент, когда посетитель веб-страницы перемещает указатель мыши над этим фрагментом. В данном случае вместо использования `a:hover` для добавления эффекта наведения указателя на ссылку вы можете создать стиль под названием `p:hover`, добавляющий эффект наведения указателя на абзац. А если вы хотите добавить стиль к тегам, применяя к ним специальный класс `highlight`, создайте стиль под названием `highlight:hover`.

ПРИМЕЧАНИЕ

В Internet Explorer 6 и более ранних версиях браузера псевдокласс :hover работает только со ссылками. Как решить эту проблему с помощью JavaScript-кода, читайте чуть ниже, во врезке «Информация для опытных пользователей». (Если при использовании IE 7 страница не располагает правильным DOCTYPE, :hover также не будет работать ни с чем, кроме ссылок.)

- `a:active` — позволяет определить, как будет выглядеть ссылка во время выбора ее посетителем веб-страницы. Другими словами, это стиль во время щелчка кнопкой мыши.

В гл. 9 описывается, как спроектировать дизайн ссылок при использовании селекторов для хорошего восприятия посетителями сайта (например, элементов навигации для перехода по разделам сайта и т. д.).

ПРИМЕЧАНИЕ

Если вы не собираетесь заниматься профессиональным веб-дизайном, то не утруждайтесь чтением последующих разделов о селекторах. Многие специалисты вообще никогда ими не пользуются. Все, что вы уже изучили, — селекторы типов, классов, потомков и т. д. — позволит создавать красивые, функциональные, легко обновляемые, профессиональные с точки зрения дизайна сайты. Если вы готовы к практической части, то переходите сразу к обучающему уроку данной главы. Вы можете закончить изучение теоретического материала позже, в любое удобное для вас время.

Стилизация фрагментов абзаца

На этапе становления Интернета не стоял вопрос дизайна страниц и сайтов, никто и не думал о том, что они должны выглядеть красиво. Теперь же это важно. В CSS имеется два псевдоэлемента — `:first-letter` и `:first-line`. Их использование обеспечит вашим веб-страницам изящное оформление, которым печатные издания обладают уже на протяжении многих столетий.

Псевдоэлемент `:first-letter` позволяет создавать буквицу — начальный символ абзаца, который выделяется из остального текста, как в начале книжной главы.

Стилизация первой строки с помощью псевдоэлемента `:first-line` отличным от основного абзаца цветом притягивает посетителей веб-страницы изяществом оформления и легкостью визуального восприятия содержимого сайта (в гл. 6 читайте все о форматировании текста, там же вы найдете подробное описание этих двух псевдоэлементов).

Дополнительные псевдоклассы и псевдоэлементы

В руководстве по языку CSS определены еще несколько мощных псевдоклассов, псевдоэлементов и селекторов. К сожалению, все еще популярный на сегодняшний день браузер — Internet Explorer 6 для Windows — не распознает их. По этой причине большинство серверов Интернета не смогут оценить дизайн веб-страниц с использованием этих элементов (по крайней мере, пока их Internet Explorer не обновится до версии 8 или не будет использоваться браузер Firefox или Safari). Вы можете обойти данную проблему, используя код JavaScript, как описано во врезке «Информация для опытных пользователей» чуть ниже.

:before. Псевдоэлемент `:before` выполняет такую функцию, которая не присуща ни одному селектору: он позволяет добавлять сообщение, предшествующее определенному элементу веб-страницы. Допустим, вы хотите поместить слово ПОДСКАЗКА! перед абзацами, чтобы визуально выделить их. Вместо многократного набора текста в HTML-коде вашей веб-страницы вы можете сделать это с помощью псевдоэлемента-селектора `:before`. Кроме того, такое решение позволит уменьшить объем кода веб-страницы. Так, если вы решите изменить сообщение с ПОДСКАЗКА! на ЭТО НУЖНО ЗНАТЬ, можно быстро отформатировать его на всех страницах сайта путем изменения текста один раз в таблице стилей. Однако негативная сторона состоит в том, что сообщение невидимо для браузеров, которые не понимают CSS или псевдоэлемента `:before`.

Для работы псевдоэлемента нужно создать класс (скажем, `.tip`) и применить его к абзацам, которым должно предшествовать данное сообщение, например `<p class="tip">`. Добавьте текст сообщения в таблицу стилей:

```
p.tip:before {content: "ПОДСКАЗКА!" }
```

Всякий раз, когда браузеру встречается класс `.tip` внутри тега `<p>`, он будет добавлять перед абзацем сообщение ПОДСКАЗКА!.

Текст, который добавляется этим псевдоэлементом-селектором, еще называют сгенерированным содержимым, поскольку браузер создает его моментально. В исходном программном коде HTML-страницы этой информации не содержится.

Браузеры все время генерируют свое информационное содержимое, например маркеры в маркированных списках или цифры в нумерованных. Чтобы понять, как браузер отображает все это, можно даже использовать селектор `:before`.

Ни IE 6, ни IE 7 не понимают CSS-свойство `content`, так что вы, возможно, не увидите повсеместного использования псевдоэлементов `:before` и `:after`, описанного далее. Однако Internet Explorer 8, как и другие распространенные браузеры, поддерживает это свойство, поэтому в гл. 16 приведены инструкции по его использованию.

ЧАСТО ЗАДАВАЕМЫЕ ВОПРОСЫ

Должен ли я обращать внимание на проблемы с IE 6?

Если вы работаете веб-дизайнером, то на вашем компьютере, вероятно, установлена последняя обновленная версия Firefox, Opera, Safari или IE. Но большинство пользователей все еще работают с Internet Explorer 6, нередко называемым бедой для веб-дизайна. В соответствии с информацией от компании Net Applications (на май 2009 года), порядка 17 % пользователей заходили на сайты с браузера IE 6. И хотя эта доля продолжает уменьшаться, IE 6 все еще будет востребованным в ближайшее время.

Некоторые люди не желают обновлять программное обеспечение и продолжают пользоваться IE 6 даже при наличии новых версий. Другие же ограничены программным обеспечением, установленным на их рабочем месте, и не имеют возможности заменить его.

Таким образом, проектируя сайт для широкой аудитории, вы должны учитывать этот фактор, только если сайтом не будут пользоваться исключительно продвинутые люди, часто обновляющие версии своих браузеров.

Есть несколько проблем IE 6, которые способны целиком испортить вид веб-страницы при просмотре в этом браузере, — в некоторых случаях содержимого страниц просто не видно или оно непригодно для чтения. Конечно, вы захотите устранить эти проблемы, и в книге будет описано, как это сделать.

Ваш сайт не обязательно должен выглядеть одинаково в IE 6 и других браузерах. Из-за незначительных различий браузеров вы часто можете увидеть, что визуально одна и та же страница выглядит по-разному в Firefox, Safari или IE.

Вашей главной целью должна быть гарантия того, что у каждого пользователя будет доступ к содержимому сайта. Если на него можно зайти с браузера IE 6, нормально читать, скачивать материалы без каких-либо проблем, значит, вы справились со своей работой. И уже после этого вы можете побеспокоиться о том, чтобы сайт выглядел максимально одинаково во всех браузерах.

С помощью CSS вы можете делать много интересных вещей, которые не будут работать в IE 6. Например, селектор `:focus` позволяет изменить стиль текстового поля в форме, когда пользователь переходит к этому полю. Это отличный способ выделить поле для ввода данных. Однако IE 6 и 7 не понимают этот селектор. Вы можете использовать JavaScript, чтобы устранить эту проблему, или просто ничего не делать, проигнорировав эти различия. Ведь если кто-то посетит ваш сайт с браузера IE 6, он все равно сможет воспользоваться текстовым полем.

Свободно применяйте другие новшества CSS, работающие в современных версиях браузеров. Другими словами, если ваш сайт работает в IE 6, но выглядит еще лучше в IE 8, Firefox, Safari и Opera, то все нормально.

:after. Псевдоэлемент-селектор `:before` добавляет сгенерированное содержимое перед определенным элементом, а `:after` — после. Например, им вы можете пользоваться для добавления заключительных кавычек после процитированного материала.

ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

Пользуйтесь обновленной версией браузера

Internet Explorer 6 уже устарел и не распознает всех новшеств CSS, причем даже не самых последних. Но вам не придется отказываться от всех интересных селекторов, описанных на этих страницах, — `:before`, `:after`, `:hover`. Всего лишь с небольшой помощью JavaScript и парочкой нововведений вы сможете написать сценарии, которые научат IE обрабатывать эти селекторы.

Например, CSSHOVER научит, как обращаться с селекторами `:focus` и `:hover` (применительно к остальным элементам веб-страниц, отличным от ссылок). Вы можете прочитать об этом по адресу www.xs4all.nl/~peterned/csshover.html. Там же вы найдете небольшой сценарий, упрощающий создание основанных на CSS раскрывающихся навигационных меню, о которых вы можете прочитать на сайте http://sperling.com/examples/menuh.

Вдобавок, вы можете использовать библиотеку JavaScript jQuery (www.jquery.com), чтобы, по существу, заставить IE 6 понимать любой селектор. Техника заключается в создании двух стилей: первого для IE 7, Firefox и других браузеров со встроенной поддержкой; и второго — класса стилей — с теми же свойствами CSS. Вы можете использовать jQuery для динамического применения класса в IE 6. Одно из существенных достоинств jQuery заключается в возможности выбора любого элемента на странице посредством использования селектора CSS и применения класса к этому элементу.

Преимущество использования усовершенствованных селекторов, например селекторов атрибутов, состоит в отсутствии необходимости вносить какие-либо изменения в HTML, чтобы все работало корректно. Так, вам не нужно добавлять класс ко всей связке тегов HTML. Благодаря jQuery и небольшому куску кода вы добьетесь той же легкости использования и для IE 6. Чтобы узнать больше об этой технике, зайдите на сайт http://somedirection.com/2007/06/10/using-jquery-to-avoid-classitis-in-ie6.

:first-child. Псевдоэлемент `:first-child` позволяет вам выбрать и отформатировать первый из множества дочерних элементов, подчиненных родительскому.

Например, у тега `<ul>`, создающего маркированный список, может быть сколько угодно дочерних элементов (на рис. 3.4 он имеет три подчиненных элемента списка `<li>`). Чтобы отформатировать только первый из них полужирным шрифтом, можно создать следующий стиль:

```
li:first-child { font-weight: bold; }
```

Поскольку селектор `:first-child` описывает лишь название дочернего элемента, этот стиль будет применен к любому тегу `<li>`, который является первым подчиненным элементом любого другого тега, а не только `<ul>`. Мы знаем, что элементы списка всегда расположены внутри его. Селектор `li:first-child` воздействует на все типы списков веб-страницы: маркированные и нумерованные. С остальными он ведет себя по-другому. В примере на рис. 3.4 `p:first-child` никак не будет работать: `<p>` — не первый дочерний элемент `<body>`.

«Родственные» отношения между родительскими и дочерними HTML-элементами могут изменяться всякий раз, когда вы редактируете веб-страницу, поэтому трудно предугадать, как поведет себя селектор `:first-child`. Кроме того, он вообще не работает в Internet Explorer 6 или более ранних версиях браузера — еще одна причина, по которой не рекомендуется пользоваться этим селектором.

:focus. Псевдокласс `:focus` функционирует подобно `:hover`, с той лишь разницей, что `:hover` применяется, когда посетитель перемещает указатель мыши над ссылкой,

a :focus — когда нажимает клавишу табуляции или щелкает кнопкой мыши на текстовом поле (то есть требуется акцентировать внимание посетителя на конкретном (текущем) элементе веб-страницы). Щелчок на заполняемой форме программисты называют *фокусировкой*. Это единственный способ для дизайнера веб-страницы узнать, на чем сосредоточено внимание посетителя, с каким элементом страницы он имеет дело.

Селектор :focus полезен в основном для обеспечения обратной связи с посетителями сайта. Например, для смены цвета фона заполняемого поля формы, чтобы указать, где именно нужно вводить данные (однострочные текстовые поля, поля ввода пароля, многострочные поля <textarea> — везде можно использовать :focus). Этот стиль задает светло-желтый цвет любому текстовому полю, на котором посетитель щелкает кнопкой мыши или в которое переходит с помощью табуляции:

```
input:focus { background-color: #FFFFCC; }
```

Селектор :focus задает стилевой эффект только на время, пока элемент находится в фокусе. Когда посетитель переходит к другому полю или в другое место веб-страницы, фокус, как и весь стилевой эффект CSS, исчезает. К сожалению, ни IE 6, ни IE 7 не поддерживают этот селектор, однако вы все же можете использовать его для создания визуальных эффектов в других браузерах (IE 6 и IE 7 просто не узнают об этом).

ПРИМЕЧАНИЕ

Изучение правильного написания селекторов иногда похоже на изучение иероглифов. Чтобы перевести селекторы на общепонятный язык, посетите веб-страницу http://gallery.theopalgroup.com/selectoracle/. Этот информационный ресурс научит правильно определять селекторы и понимать, к каким элементам веб-страниц они применяются.

Другие селекторы

В руководстве языка CSS 2 описывается несколько мощных селекторов, которые обеспечивают еще более тонкое управление дизайном веб-страниц по сравнению с уже изученными. Они также не работают в браузере Internet Explorer 6 и его более ранних версиях (но вы можете воспользоваться языком JavaScript, как это было описано во врезке «Информация для опытных пользователей»).

Селекторы дочерних элементов

Подобно применению селекторов потомков, описанных ранее, в CSS можно форматировать вложенные элементы других тегов с помощью *селектора дочерних элементов*, который использует дополнительный символ — угловую скобку (>) — для указания отношения между двумя элементами. Например, body > h1 выбирает любой тег <h1>, дочерний по отношению к <body>.

В отличие от селектора потомков, который применяется ко всем потомкам (то есть вложенным элементам), селектор дочерних элементов позволяет определить конкретные дочерний и родительский элементы. На рис. 3.5 вы можете видеть

два тега <h2>. Использование body h2 привело бы к выбору обоих, так как они являются вложенными по отношению к <body>. Только второй тег — дочерний элемент <body>. Первый <h2> непосредственно вложен в , который и является родительским.

Поскольку у <h2> различные родительские теги, то для того, чтобы добраться до каждого из них отдельно, можно использовать селектор дочерних элементов. Для выбора только второго тега <h2> используем body > h2. Если первый тег <h2>, то div > h2.

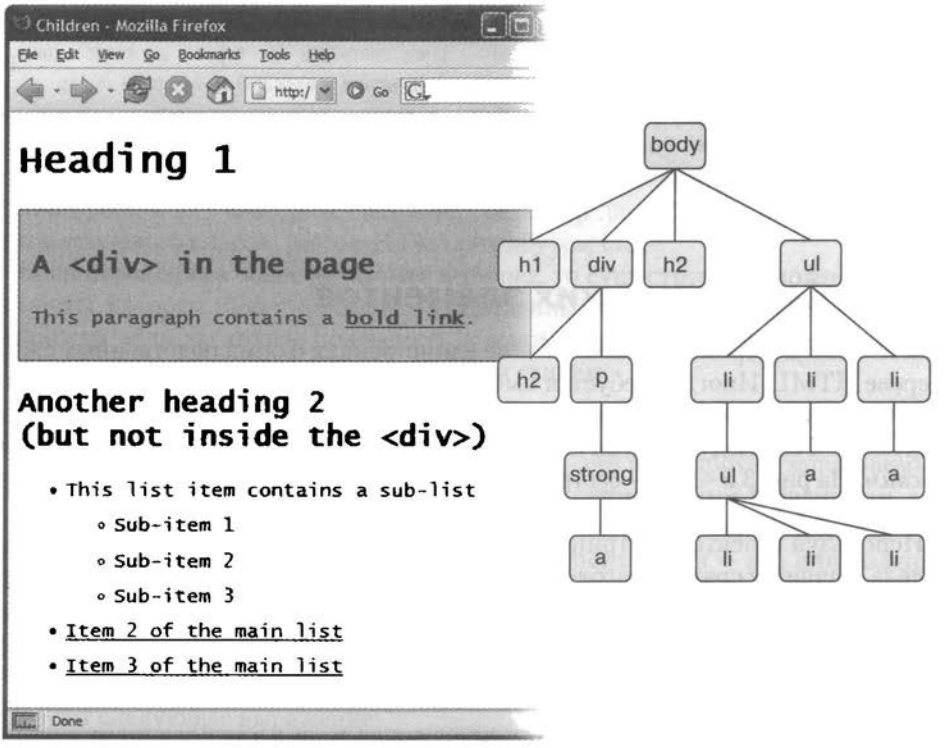


Рис. 3.5. Диаграмма в виде дерева *(справа)* показывает отношения между HTML-тегами *(слева)*

ЧАСТО ЗАДАВАЕМЫЕ ВОПРОСЫ

Создание списков

Когда нужно использовать селекторы дочерних элементов? Вы можете сказать, что уже знаете достаточное количество селекторов для выборки практически любого элемента веб-страницы. Так для чего нужны остальные селекторы?

На самом деле существуют некоторые сложности дизайна веб-страниц, где селекторы дочерних элементов просто незаменимы и никакие другие не смогут с этим справиться. Такая ситуация встречается на большинстве сайтов. В любом маркированном списке присутствует один или несколько пунктов — элементов списка (см. рис. 3.5). Здесь можно использовать селекторы дочерних элементов, чтобы визуально упорядочить данные в категориях (пунктах) и подкатегориях (подпунктах). Вы форматируете элементы первого уровня списка одним способом,

ЧАСТО ЗАДАВАЕМЫЕ ВОПРОСЫ

а второго — другим. Содержимое, представленное таким способом, выглядит четко, профессионально и читаемо.

Сначала создайте класс для самого верхнего — внешнего — уровня вложенности элементов списка и назовите его, скажем, `.mainList`. Для первого уровня вы можете использовать шрифт sans-serif, имеющий немного больший размер по сравнению с основным текстом веб-страницы, возможно, в другом цвете. Последующие категории могут быть представлены Times для лучшего восприятия. При большом объеме текста стилизация каждого уровня подкатегорий с небольшим отличием позволяет посетителям веб-страниц визуально ориентироваться в материале.

Теперь примените стилевой класс `.mainList` к первому тегу `<ul>`: `<ul class="mainList">`. Затем используйте селектор дочерних элементов (`ul.mainList > li`) для выбора элементов списка только первого уровня и придания пунктам необходимого форматирования. Данный стиль будет применен только к тегу `<li>`, являющемуся дочерним по отношению к `<ul>` и принадлежащему классу `.mainList`. Все остальные подкатегории `<li>` не будут затронуты. Например, чтобы стилизовать теги `<li>` первого вложенного списка, используйте следующий селектор: `ul.mainList > li > ul > li` (он осуществит выборку маркированных элементов списков во всех категориях и подкатегориях).

Селекторы сестринских элементов

Родительско-дочерние отношения — не единственная форма родственных связей в дереве HTML. Иногда требуется выбрать тег, относящийся к группе элементов одного уровня, находящихся в непосредственной близости друг от друга, с общим родителем. Тег, следующий за несколькими другими, в HTML называется *сестринским*. На рис. 3.5 `<div>` — сестринский по отношению к `<h1>`, а `<p>` — по отношению к `<h2>` и т. д.

Используя селектор сестринских элементов, можно придать первому абзацу раздела, идущему сразу за заголовком, форматирование, отличное от следующего далее по тексту. Предположим, вы хотите удалить отступ, который автоматически появляется перед абзацем `<p>`, чтобы между заголовком и тегом не было никакого промежутка. Или хотите придать абзацу, как небольшому вводному описанию, другой цвет и размер шрифта.

Селектор сестринских элементов использует знак `+` для соединения одного элемента с другим. Так, чтобы выбрать все первые абзацы, следующие за любым `<h2>`, используйте `h2 + p` (пробелы необязательны, так что `h2+p` также будет прекрасно работать). Последний элемент в селекторе (в данном случае — `p`) — тег, который нужно отформатировать, но только при условии, что он соседствует с `<h2>`.

Селекторы атрибутов

CSS обеспечивает возможность форматирования тегов на основе выборки любых содержащихся в них атрибутов. Например, вы хотите оформить в рамку некоторые важные изображения веб-страницы, при этом не форматируя логотип, кнопки и другие небольшие изображения, в которых также есть тег `<img>`. Вы должны понимать, что если у всех рисунков есть описание с атрибутом `title`, то это способ-

ствует использованию селектора для выделения из массы изображений только нужных.

ПРИМЕЧАНИЕ

Применение атрибутов title в HTML — отличный способ добавить подсказки (всплывающие сообщения) к ссылкам и изображениям веб-страницы. Это также является одним из способов краткого описания содержания для поисковых серверов. Узнать об этом больше вы можете по адресу http://webdesign.about.com/od/htmltags/a/aa101005.htm.

С помощью *селекторов атрибутов* вы можете выбрать теги с конкретными элементами. Вот пример, в котором выделены все теги <img> с атрибутами title:

```
img [title]
```

Первая часть селектора — название (img); атрибут идет далее в квадратных скобках: [title].

CSS не ограничивает селекторы атрибутов названиями тегов: вы можете также комбинировать их с классами. Например, селектор .photo[title] выбирает все элементы стилевого класса .photo с HTML-атрибутом title.

Если необходима более детальная выборка, то существует возможность найти элементы, которые имеют не только определенный атрибут, но и точное значение. Например, если вы хотите подсветить ссылки, указывающие на специфический URL, создайте селектор атрибута с броским стилем:

```
a[href="http://www.cosmofarmer.com"]{ color:red; font-weight:bold; }
```

Уточнение конкретным значением очень полезно при работе с заполняемыми формами. Многие элементы форм имеют теги с одинаковыми названиями, даже если они выглядят и функционируют по-разному. Будь то флажок, текстовое поле, кнопка отправки данных или какие-то другие поля форм — все они содержат <input>. Тип атрибута — вот что придает определенную форму с соответствующими функциональными возможностями. Например, <input type="text"> создает текстовое поле, а <input type="checkbox"> — флажок.

Например, чтобы выбрать только текстовые поля в форме веб-страницы, используют следующее выражение:

```
input[type="text"]
```

Селектор атрибута — очень разносторонний элемент. Он не только позволяет находить теги с определенным значением атрибута (например, все поля формы с типом checkbox), но и выбирать элементы со значением атрибута, начинающимся с какого-либо значения, заканчивающимся им или содержащим его. Хоть эта возможность сразу может показаться лишней, на самом деле она достаточно полезна.

Представьте, что вы хотите создать стиль, который бы выделял внешние ссылки (ведущие за пределы вашего сайта), говоря пользователю: «Ты покинешь этот сайт, если выберешь ссылку». Принимая во внимание то, что абсолютные ссылки внутри собственного сайта не используются, мы устанавливаем, что любая внешняя ссылка будет начинаться с http:// — первой части любой абсолютной ссылки.

Тогда селектор будет выглядеть следующим образом:

```
a[href^="http://"]
```

Символы ^= означают «начинается на», так что вы можете использовать этот селектор для форматирования любой ссылки, начинающейся на http://. С его помощью вы легко стилизуете ссылку, указывающую на http://www.google.com либо http://www.sawmac.com — любую внешнюю ссылку.

ПРИМЕЧАНИЕ

Этот селектор не будет работать в случае защищенного SSL-соединения, то есть когда ссылка начинается на https://. Чтобы создать стиль, учитывающий данную проблему, вам понадобится групповой селектор:

```
a[href^="http://"], a[href^="https://"]
```

Встречаются также ситуации, когда вам необходимо выбрать элемент с атрибутом, заканчивающимся определенным значением. И снова ссылки пригодны для этой задачи. Скажем, вы хотите добавить небольшой значок после ссылок, указывающих на PDF-файлы. Поскольку они имеют разрешение PDF, вы знаете, что ссылка на эти документы будет заканчиваться также этими символами, например <a href = "download_instructions">. Значит, чтобы выбрать только такие ссылки, нужен следующий селектор:

```
a[href$=".pdf"]
```

А стиль целиком будет выглядеть так:

```
a[href$=".pdf"] {
  background-image: url(doc_icon.png) no-repeat;
  padding-left: 15px;
};
```

Не беспокойтесь о том, что не знаете конкретные свойства из этого стиля, — вы познакомитесь с ними далее в книге. Только обратите внимание на один интересный селектор: $ означает «заканчивается на». Вы можете использовать его для форматирования ссылок на документы Word ([a href$=".doc"]), фильмы ([a href$=".mov"]) и т. д.

И наконец, вы можете выбирать элементы с атрибутами, содержащими конкретное значение. Например, вам нужно выделить фотографии сотрудников повсюду на сайте. Вы хотите, чтобы у всех фотографий был общий стиль, допустим, зеленая рамка и серый фон. Один из способов сделать это — создать класс стиля, например .headshot, и добавлять вручную атрибут класса к соответствующим тегам <img>. Однако если вы задали для фотографий последовательные названия, это не самый быстрый метод.

Например, каждую из фотографий вы называли, используя при этом слово headshot, — mcfarland_headshot, mccord_headshot и т. д. В каждом файле встречается слово headshot, поэтому и атрибут src тега <img> содержит это слово. Вы можете создать селектор специально для этих изображений:

```
img[src*="headshot"]
```

Выражение переводится как «выберите все изображения, атрибут src которых содержит в любом месте слово headshot». Это простой и изящный способ форматирования фотографий сотрудников.

Internet Explorer 6, как и полагается, не поддерживает селекторы атрибутов, поэтому используйте их, только если отсутствующий стиль не навредит отображению страницы в IE 6. Во многих случаях вы можете использовать селекторы атрибутов просто для того, чтобы добавить какую-нибудь оригинальную идею на страницу для современных браузеров.

ПРИМЕЧАНИЕ

В версии CSS 3 разработчики обещают большее разнообразие селекторов атрибутов. Из гл. 16 вы узнаете о самых многообещающих селекторах (и о тех, которые можно найти в реальных браузерах).

Обучающий урок: примеры использования селекторов

В оставшейся части этой главы вы потренируетесь в создании разных селекторов и увидите, как они влияют на дизайн веб-страницы. Обучающий урок начнем с представления основных типов, а затем перейдем к более современным стилям.

Чтобы начать обучение, вы должны иметь в распоряжении файлы с учебным материалом. Файлы текущей обучающей программы находятся в папке с названием **03**, расположенной в архиве, работа с которым описана в конце гл. 2.

ЧАСТО ЗАДАВАЕМЫЕ ВОПРОСЫ

Разрабатывая дизайн одиночных веб-страниц, используйте внутренние таблицы стилей

Вы можете спросить: почему в этой обучающей программе мы пользуемся внутренними таблицами стилей? Ведь в гл. 2 книги рекомендуется применять внешние.

Да, они используются для создания «быстрых» сайтов. Однако внутренние таблицы стилей облегчают проектирование одиночных веб-страниц, таких как в этой обучающей программе. В данном случае гораздо удобнее работать с одним файлом вместо того, чтобы переключаться между файлом внешней таблицы стилей и веб-страницей. Вы можете пользоваться предварительным просмотром результатов своей работы без постоянного обновления кэша браузера.

Многие знатоки начинают разработку дизайна с внутренней таблицы стилей, поскольку так намного быстрее. Это исключает все прочие проблемы, в том числе с кэшем. Вот их методика: как только они добьются требуемых результатов по созданию стиля отдельно взятой веб-страницы, они просто копируют код внутренней таблицы стилей и вставляют его во внешнюю, а затем связывают ее со страницами сайта, как описано в гл. 2.

1. Откройте файл selector_basics.html в программе редактирования.

   Страница состоит из основных HTML-тегов. Самая интересная вещь на ней — это графический баннер (рис. 3.6). В этой обучающей программе мы попытаемся

оживить элементы веб-страницы, придать им изящный внешний вид. С добавлением CSS-стилей мы превратим «серенькие» веб-страницы в страницы с потрясающим дизайном. Начнем с добавления к этому файлу внутренней таблицы стилей.

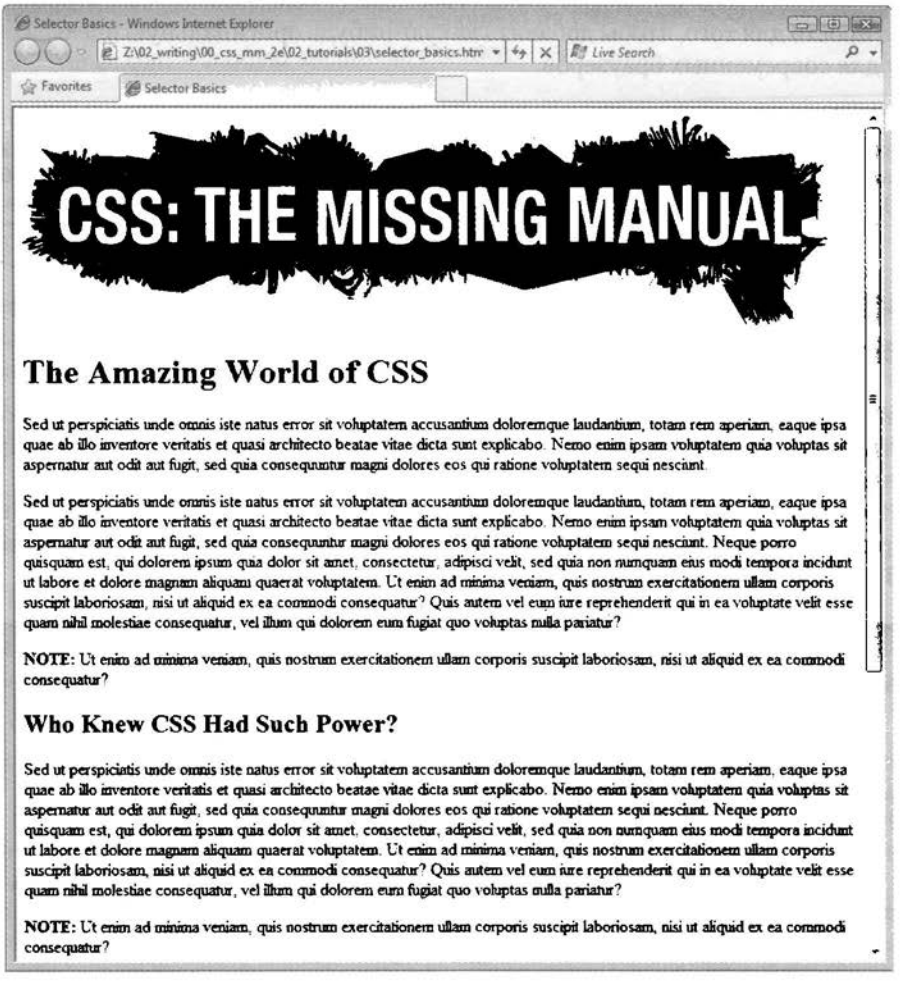


Рис. 3.6. Простой HTML-текст смотрится в браузере чересчур холодно и монотонно

2. Щелкните кнопкой мыши сразу после закрывающего тега `</title>`, нажмите клавишу Enter для добавления новой строки и наберите `<style type="text/css">`. Дважды нажмите клавишу Enter и наберите `</style>`.

   Это открывающий и закрывающий теги внутренней таблицы стилей. Очень полезно набирать их вместе одновременно, чтобы случайно не забыть о закрывающем. Так, они вместе сообщают браузеру, что между ними находятся коман-

ды языка CSS. HTML-код теперь должен выглядеть следующим образом (код, который вы только что добавили, выделен полужирным шрифтом):

```
<title>Selector Basics</title>
<style type="text/css">

</style>
</head>
```

Теперь нужно ввести селектор типа, который вы собираетесь создать, — из набора основных типов (если вы дошли до конца обучающей программы в прошлой главе, то уже умеете это делать).

3. Перейдите к строке между открывающим и закрывающим тегами <style>, а затем введите p {. Дважды нажмите Enter и введите }.

   Как я уже отмечал, желательно ставить закрывающую скобку сразу же, как только вы вводите открывающую. Чтобы создать селектор типа, просто используйте название HTML-тега, которому желаете придать стиль. Он будет применен ко всем абзацам текста, заключенным в <p>.

4. Перейдите к строке между открывающей и закрывающей скобками, добавьте четыре свойства CSS, чтобы обеспечить форматирующий стиль — цвет, размер, начертание шрифта, отступ:

```
p {
  color: #505050;
  font-size: 1em;
  font-family: "Helvetica Neue", Arial, Helvetica, sans-serif;
  margin-left: 100px;
}
```

5. Нажмите клавишу Enter, чтобы поместить каждое CSS-свойство в отдельной строке. Кроме того, полезно сделать отступы клавишей табуляции, чтобы улучшить визуальное восприятие кода CSS.

ПРИМЕЧАНИЕ

Если вы новичок в веб-дизайне, то названия свойств и их значения вам пока незнакомы. Так что просто набирайте их в том виде, как показано в тексте. Что означает 1em и 100px и что такое текстовые свойства, вы узнаете из гл. 6.

Работа над вашей таблицей стилей закончена.

6. Откройте страницу в браузере, чтобы осуществить предварительный просмотр.

   Если вы не изменяли стандартные параметры настроек свойств браузера, то большинство из них отображает текст веб-страницы черным шрифтом с засечками (serif) — Times. Если ваш CSS-стиль функционирует должным образом, то теперь вы увидите семь абзацев, для которых задан темно-серый цвет текста, шрифт sans-serif и отступы в начале.

Создание групповых селекторов

Очень часто несколько различных элементов веб-страницы должны иметь одинаковый внешний вид. Вероятно, и вы хотите, чтобы все заголовки отображались шрифтом одного начертания и цвета. Вместо того чтобы создавать отдельные стили и дублировать одни и те же атрибуты и параметры для каждого тега <h1>, <h2> и т. д., вы можете собрать и сгруппировать несколько тегов в единственный селектор.

1. Вернитесь к своему HTML-редактору с файлом `selector_basics.html`.

   Сейчас мы добавим новый стиль сразу после только что созданного стиля тега <p>.

2. Щелкните кнопкой мыши после закрывающей фигурной скобки селектора <p>, нажмите клавишу **Enter** для начала новой строки и наберите `h1, h2, h3 {`.

   Как описано в этой главе ранее, групповой селектор — это просто список. Данный стиль создает одинаковое форматирование тегов <h1>,<h2> и <h3> веб-страницы.

3. Нажмите клавишу **Enter** и добавьте пять CSS-свойств:

```
color: #BD8100;
font-family: Baskerville, "Palatino Linotype", Times, serif;
border-top: 2px solid #86A100;
padding-top: 7px;
padding-left: 100px;
```

   Здесь вы задаете цвет и тип шрифта для заголовков, добавляете линию границы над заголовками, устанавливаете отступы слева и сверху, используя свойство `padding`. Если говорить кратко, то это свойство добавляет дополнительное пространство от краев элемента без воздействия на фон или рамку, то есть вы убираете текст заголовка от левого и верхнего краев, не затрагивая линию рамки, которая охватывает всю страницу.

4. Теперь нажмите клавишу **Enter** и введите закрывающую фигурную скобку, чтобы завершить данный стиль. Все описание должно выглядеть следующим образом:

```
h1, h2, h3 {
  color: #BD8100;
  font-family: Baskerville, "Palatino Linotype", Times, serif;
  border-top: 2px solid #86A100;
  padding-top: 7px;
  padding-left: 100px;
}
```

5. Сохраните файл и просмотрите его работу в браузере.

   Заголовок <h1> в начале веб-страницы и заголовки <h2> и <h3> ниже на странице имеют одинаковое начертание и цвет шрифта, а также зеленую рамку вверху (рис. 3.7).

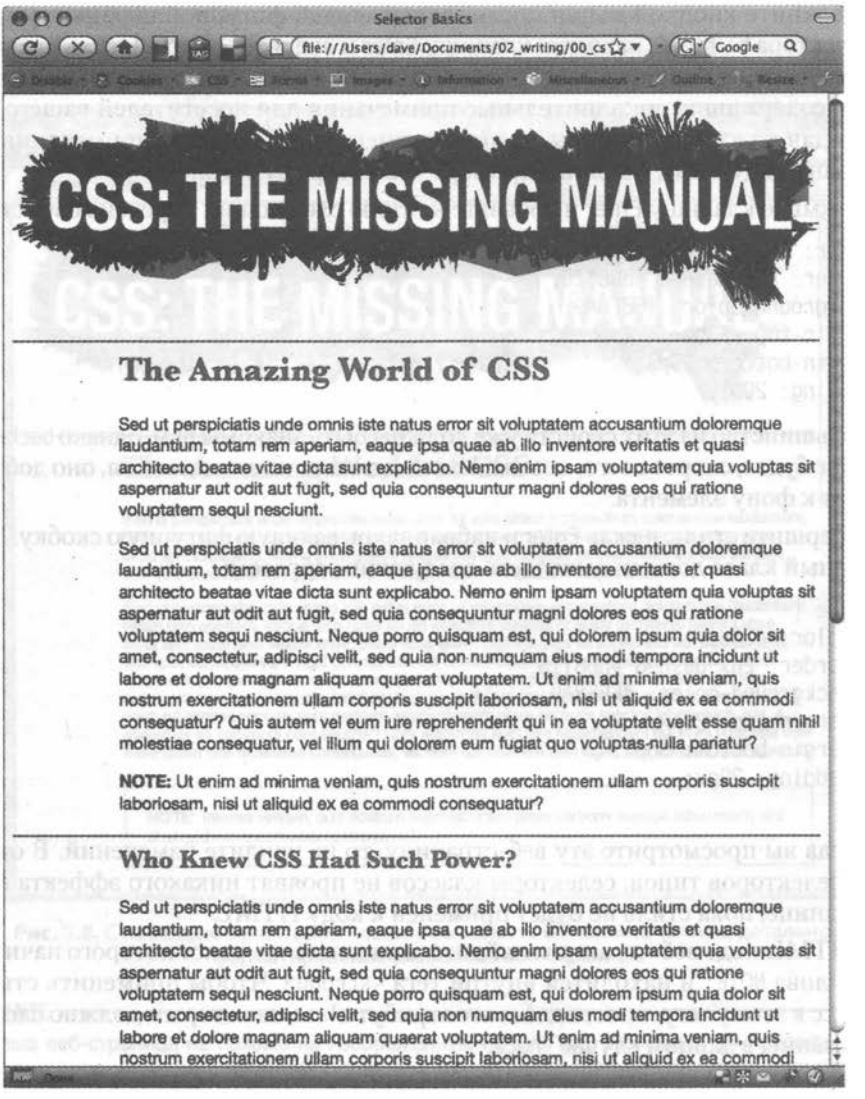


Рис. 3.7. Простой селектор типа может коренным образом преобразовать внешний вид всех вхождений форматируемого тега, быстро выполнив стилизацию текста веб-страницы

Создание селекторов классов

Селекторы типов выполняют свои функции быстро и эффективно, но они, можно сказать, совсем неразборчивы в деталях. Что же делать, если вы хотите отформатировать один тег <p> веб-страницы иным способом, чем все остальные такие теги? Решение проблемы — использование классов.

1. Вернитесь к HTML-редактору с файлом `selector_basics.html`.

   Добавьте вслед за последним созданным стилем еще один.

2. Щелкните кнопкой мыши после закрывающей фигурной скобки группового селектора h1, h2, h3, нажмите клавишу **Enter** и введите `.note {`. Стиль назван `note` (примечание) не случайно. Он соответствует своим функциям: выделяет абзацы, содержащие дополнительные примечания для посетителей вашего сайта. Создав класс один раз, вы можете применить его ко всем примечаниям веб-страницы (сайта), в данном примере — ко второму абзацу.
3. Нажмите клавишу **Enter** и добавьте к стилю следующий перечень свойств:

```
color: #333;
border: 2px dashed #BD8110;
background-color: #FBF8A9;
margin-top: 25px;
margin-bottom: 35px;
padding: 20px;
```

Большинство из этих свойств уже должны быть знакомы вам, однако `background-color` будет, скорее всего, новым. Как и вытекает из его названия, оно добавляет цвет к фону элемента.

4. Завершите стиль, нажав **Enter** и набрав закрывающую фигурную скобку. Законченный класс должен выглядеть следующим образом:

```
.note {
  color: #333;
  border: 2px dashed #BD8110;
  background-color: #FBF8A9;
  margin-top: 25px;
  margin-bottom: 35px;
  padding: 20px;
}
```

Когда вы просмотрите эту веб-страницу, то не увидите изменений. В отличие от селекторов типов, селекторы классов не проявят никакого эффекта на веб-странице, пока стиль не будет применен к коду HTML.

5. В HTML-коде веб-страницы найдите вхождение `<p>`, текст которого начинается со слова `NOTE:` и находится внутри тега `<strong>`. Чтобы применить стилевой класс к этому тегу, просто добавьте атрибут `class`, за которым должно следовать название, в данном случае `.note`.
6. Щелкните кнопкой мыши сразу за буквой «р» тега `<p>`, нажмите **Пробел**, а дальше введите `class="note"`. HTML-код теперь должен иметь такой вид (только что набранный код отмечен полужирным шрифтом):

```
<p class="note"><strong>NOTE:</strong> Ut enim ad
```

Убедитесь, что не ввели атрибут так: `class=".note"`. Точка требуется только во время определения названия стилевого класса в таблице стилей; в HTML она не нужна. Повторите этот шаг для второго абзаца (он идет сразу над тегом `<h3>` с текстом «Not Me!»).

ПРИМЕЧАНИЕ

Несмотря на название, которое вы дали классу, можете применить его к любым другим тегам, а не только к <p>. Если данное форматирование относится, например, к <h2>, то HTML-код должен выглядеть следующим образом: <h2 class="note">.

7. Сохраните файл и просмотрите веб-страницу в браузере.
   Абзац с примечанием красиво подсвечен на странице (рис. 3.8).

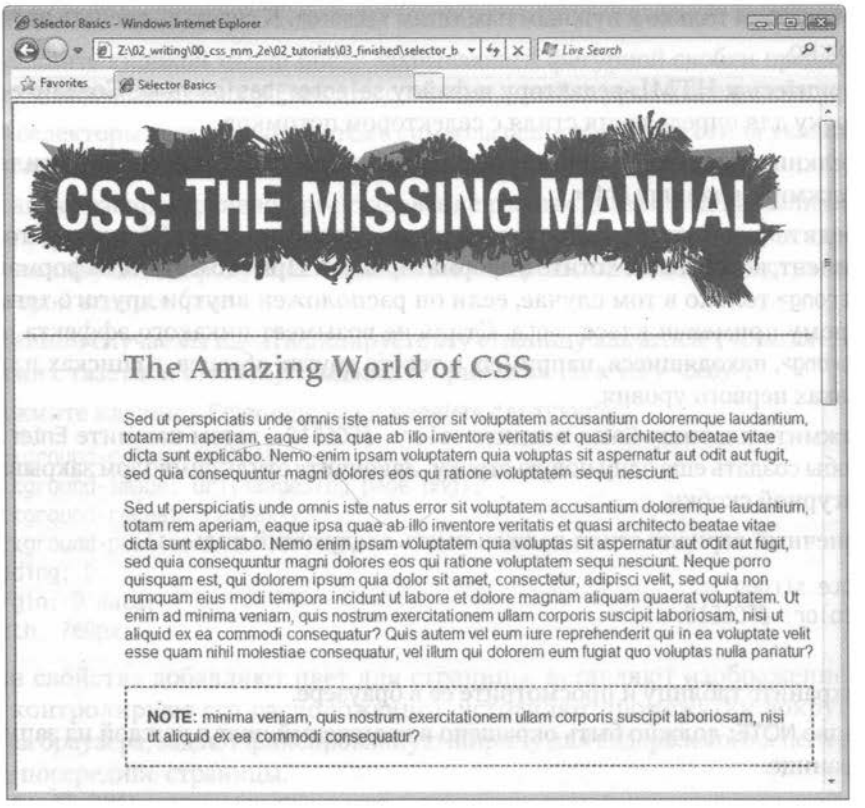


Рис. 3.8. С помощью селекторов классов вы можете выполнить точное, детальное форматирование элементов веб-страницы

ПРИМЕЧАНИЕ

Если ваша веб-страница не похожа на изображенную на рис. 3.8, возможно, вы набрали какое-то свойство или его значение с ошибкой. Проверьте код по шагам. Кроме того, удостоверьтесь в том, чтобы каждая пара «свойство — значение» была завершена точкой с запятой и в самом конце определения стиля присутствовала закрывающая фигурная скобка. Если ваш стиль не работает должным образом, то, скорее всего, в нем не хватает именно этих символов. Это самая частая ошибка.

Создание селекторов потомков

На странице `selectors_basics.html` вы применили класс `note` к двум абзацам. Каждый из них начинается словом **NOTE:**, выделенным жирным шрифтом (на самом деле это слово находится внутри тега `<strong>`). Но что делать, если вы хотите отформатировать эти слова еще и ярко-оранжевым цветом? Вы могли бы создать стиль для тега `<strong>`, но он затронет все теги `<strong>` на странице, в то время как вы хотите изменить только те, которые находятся внутри этих записей. Одним из решений было бы создание класса, например `.noteText`, и применение его к каждому из тегов

 внутри записей. Но вы наверняка забудете применить класс, если у вас много таких страниц с записями.

Лучший способ решить эту проблему — создать наследуемый селектор, который относится только к нужным нам тегам . К счастью, сделать это совсем не сложно.

1. Вернитесь к HTML-редактору и файлу `selector_basics.html`. Создайте новую строку для определения стиля с селектором потомков.

   Щелкните кнопкой мыши после закрывающей фигурной скобки стиля `.note` и нажмите клавишу **Enter**.

2. Введите `.note strong {`. Последнее слово селектора — тег — это и есть элемент, который вы хотите отформатировать. При этом стиль отформатирует только в том случае, если он расположен внутри другого тега, к которому применен класс `.note`. Стиль не возымеет никакого эффекта на теги , находящиеся, например, в тексте других абзацев, в списках или заголовках первого уровня.

3. Нажмите клавишу **Enter**, введите `color: #FC6512;`, затем нажмите **Enter** снова, чтобы создать еще одну новую строку. Закончите стиль символом закрывающей фигурной скобки.

   Конечный вариант стиля должен иметь следующий вид:

```
.note strong {
  color: #FC6512;
}
```

4. Сохраните таблицу и просмотрите ее в браузере.

   Слово **NOTE:** должно быть окрашено в оранжевый цвет в каждой из записей на странице.

Селекторы потомков — одно из самых мощных средств языка CSS. Профессиональные веб-дизайнеры используют их достаточно интенсивно для целенаправленной стилизации отдельных тегов, при этом не засоряя HTML-код классами. В книге они используются повсеместно, а более подробно селекторы потомков мы рассмотрим в гл. 15.

Создание ID-селекторов

Вы можете применить селекторы классов ко многим элементам веб-страницы. Например, ранее вы создали класс `.note` и применили его к двум абзацам, хотя могли бы применить и к большему числу абзацев или даже других тегов.

ID-селекторы выглядят и функционируют точно так же, как селекторы классов, с тем исключением, что их можно применить всего один раз. Дизайнеры часто используют их для того, чтобы указать и отделить друг от друга уникальные разделы веб-страницы, как описано в разд. «ID-селекторы: определение элементов веб-страниц» этой главы.

В данном примере мы создадим стиль, который устанавливает определенную ширину для основного содержимого веб-страницы, располагает его посередине

окна браузера и добавляет декоративное фоновое изображение. Вы примените ID-селектор к тегу <body> для создания уникального дизайна страницы.

1. Вернитесь к HTML-редактору с файлом selector_basics.html.

   Добавим за последним созданным классом .note strong новый стиль.

2. Щелкните кнопкой мыши после закрывающей фигурной скобки предыдущего стиля, нажмите клавишу **Enter** для создания новой строки и введите #article {.

   ID-селекторы всегда начинаются с символа решетки #. Имя стиля указывает на тип веб-страницы.

   В сайтах распространено проектирование разных дизайнов для различных типов страниц. Например, домашняя страница выглядит не так, как страница, рекламирующая продукцию, а та, в свою очередь, отличается от страницы, на которой ведется блог.

   В данном случае вы идентифицируете эту страницу как article («статья», по аналогии с газетной статьей), создавая и применяя ID к тегу <body>.

3. Нажмите клавишу **Enter** еще раз и введите следующее:

```
background-color: #FDF8AB;
background-image: url(images/bg_page.png);
background-repeat: repeat-y;
background-position: center top;
padding: 0;
margin: 0 auto;
width: 760px;
```

   Эти свойства добавляют цвет для страницы, вставляют изображение в фон (и контролируют его расположение), устраняют промежутки вокруг краев окна браузера, задают фиксированную ширину для содержимого и центрируют все посередине страницы.

4. Завершите определение стиля, набрав закрывающую фигурную скобку. Весь код стиля должен выглядеть так:

```
#article {
  background-color: #FDF8AB;
  background-image: url(images/bg_page.png);
  background-repeat: repeat-y;
  background-position: center top;
  padding: 0;
  margin: 0 auto;
  width: 760px;
}
```

   Как и в примере с классом, данный стиль не будет эффективен, пока вы не примените его к веб-странице. Таким образом, нужно добавить атрибут id к HTML-коду веб-страницы, обозначая фрагмент, к которому вы хотите его отнести.

5. Найдите на веб-странице открывающий тег <body> и добавьте id="article", чтобы он выглядел следующим образом (изменения выделены полужирным):

```
<body id="article">
```

Теперь тег <body> отражает форматирование, определенное в стиле #article. Как это часто случается при работе с CSS, есть много способов добиться одного и того же результата. Вы могли бы использовать класс стиля и применить его к тегу <body> при условии, что делаете это не более одного раза на странице. Вы даже могли бы просто создать стиль для элемента body с теми же свойствами форматирования, если они подойдут и для всех остальных страниц вашего сайта. Но в данном случае вы используете селектор ID, поскольку назначение стиля — идентификация типа страницы — соответствует идее селекторов ID.

6. Сохраните страницу и просмотрите ее в браузере.

   Все содержимое веб-страницы — логотип и текстовые данные — теперь имеют фиксированную ширину и расположены в центре относительно окна браузера. Даже если вы измените размеры окна браузера (попробуйте!), информационное содержимое останется центрированным. Вдобавок на каждой стороне содержимого появляется тень, что происходит благодаря полезному свойству background-image (более подробно о нем вы узнаете далее в этой книге).

Последние штрихи

Для развлечения добавим один усовершенствованный стиль — смежный сестринский одноуровневый селектор — для форматирования абзаца, следующего сразу за первым заголовком на странице (того же самого результата вы можете добиться, создавая класс стиля и применяя его к абзацу, но смежный сестринский селектор освобождает от внесения правок в код HTML).

1. Вернитесь к HTML-редактору с файлом selector_basics.html. Создайте новую строку для нового стиля.

   Если вы только что завершили предыдущие шаги, щелкните кнопкой мыши сразу за закрывающей скобкой стиля #article и затем нажмите клавишу **Enter**.

2. Введите h1+p {.

   Этот стиль будет применяться к любому абзацу, следующему за тегом <h1>, то есть за главным заголовком на странице. Ко второму и последующим абзацам он применяться не будет. Селектор предоставляет легкий способ создать уникальный внешний вид для вступительного абзаца, его визуального выделения и обозначения начала статьи.

3. Нажмите **Enter** и добавьте к стилю следующие свойства:

```
color: #FF6600;
font-size: 1.2em;
line-height: 140%;
margin-top: 20px;
```

   Здесь вы изменяете цвет и размер шрифта, а также добавляете немного свободного пространства над абзацем. Свойство line-height (подробно о нем вы узнаете далее в этой книге) управляет промежутком между строками в абзаце (это свойство также известно как leading).

4. Наконец, завершите стиль, нажав **Enter** и поставив закрывающую скобку. Стиль должен выглядеть так:

```
h1+p {
  color: #FF6600;
  font-size: 1.2em;
  line-height: 140%;
  margin-top: 20px;
}
```

Если вы просмотрите страницу в браузере сейчас, то увидите, что верхний абзац стал оранжевым, текст выглядит более крупным, а между строками расстояние больше (рис. 3.9). Если вы удаляли этот абзац в HTML, то заметите, что оставшийся абзац стал оранжевым и имеет более крупный текст, поскольку он теперь новый смежный сестринский тег для <h1>.

ПРИМЕЧАНИЕ

Internet Explorer 6 не поддерживает смежные сестринские селекторы, поэтому в этом браузере первый абзац будет выглядеть аналогично всем остальным.

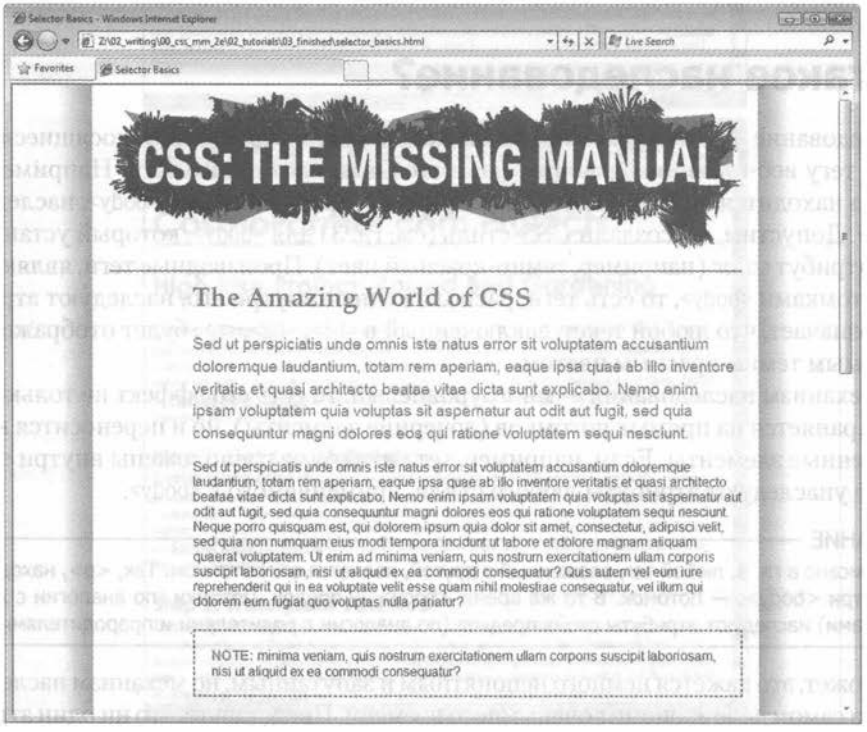


Рис. 3.9. Теперь веб-страница действительно имеет законченный вид

ПРИМЕЧАНИЕ

Окончательную версию созданной в этой главе экспериментальной веб-страницы вы можете найти в папке 03_finished загруженного вами архива с учебным материалом.

Итак, мы ознакомились с различными типами селекторов. Более подробно вы изучите их (и не только их) в обучающих примерах в следующих главах этой книги.

4 Механизм наследования стилей

Дети наследуют некоторые черты своих родителей: цвет глаз, рост. Как вы убедились в предыдущей главе, модель семейных отношений применима и к структуре языка HTML. И точно так же, как люди, теги могут унаследовать CSS-свойства от своих предков.

Что такое наследование?

Наследование — это прием, с помощью которого CSS-свойства, относящиеся к одному тегу веб-страницы, распространяются и на вложенные теги. Например, <p> всегда находится внутри <body>. Так, атрибуты, применяемые к <body>, наследуются <p>. Допустим, вы создали CSS-стиль (см. гл. 3) для <body>, который устанавливает атрибут color (например, темно-красный цвет). Производные теги, являющиеся потомками <body>, то есть теги, расположенные внутри его, наследуют атрибут. Это означает, что любой текст, заключенный в <h1>, <h2>, <p>, будет отображен тем же самым темно-красным цветом.

Механизм наследования — многоуровневый, то есть его эффект не только распространяется на прямых потомков (дочерние элементы), но и переносится на все вложенные элементы. Если, например, и расположены внутри <p>, то также унаследуют атрибуты любого стиля, применяемого к <body>.

ПРИМЕЧАНИЕ

Как описано в гл. 3, любой тег, вложенный в другой, является его потомком. Так, <p>, находящийся внутри <body>, — потомок. В то же время <body> — предок. Потомки (по аналогии с детьми и внуками) наследуют атрибуты своих предков (по аналогии с родителями и прародителями).

Может, это кажется немного непонятным и запутанным, но механизм наследования на самом деле экономит очень много времени. Представьте, что ни один атрибут не наследуется вложенными тегами и у вас есть абзац текста, который содержит, например, тег , выделяющий фрагмент текста, или <a>, добавляющий гиперссылку. Если вы создали стиль, форматирующий данный абзац шрифтом Arial размером 24 пиксела фиолетового цвета, было бы странно, если бы внутри отобразился прежний стиль. Вам пришлось бы создавать для форматирования еще один стиль.

На странице, изображенной вверху рис. 4.1, тег абзаца устанавливает определенное начертание, размер, цвет шрифта. Абзацы наследуют эти свойства и имеют единообразный стиль.



Рис. 4.1. Наследование позволяет вложенным тегам копировать атрибуты окружающих их родительских тегов

Если бы наследования не существовало, то веб-страница выглядела бы так, как показано в нижней части рисунка. Обратите внимание, что `<strong>`, `<em>` и все вложенные теги сохранили обычное начертание, размер и цвет шрифта, определенные браузером стандартно. Чтобы отформатировать их подобно остальной части абзаца, вам пришлось бы создавать дополнительные стили.

Наследование работает не только со стилями тегов, но и с любыми другими типами. Когда вы применяете стилевой класс (см. разд. «Селекторы типов: дизайн страницы» гл. 3) к какому-нибудь тегу, то вложенные в него теги наследуют стилевые атрибуты. То же самое справедливо и для всех типов селекторов, рассмотренных в гл. 3.

Упрощение таблиц стилей через наследование

Вы можете использовать преимущества механизма наследования в своих интересах для того, чтобы упростить и ускорить написание таблиц стилей. Предположим, вы хотите отобразить весь текст веб-страницы одинаковым шрифтом. Вместо того чтобы создавать стили для каждого тега, просто создайте один для <body> (или создайте класс и примените его). Определите нужный шрифт, и все теги веб-страницы унаследуют его:

```
body { font-family: Arial, Helvetica, sans-serif; }
```

Вы также можете использовать наследование для применения стилевых атрибутов к целому разделу веб-страницы. Например, вы можете применять, как и большинство дизайнеров, тег (см. разд. «Селекторы классов: точное управление» гл. 3) для определения таких фрагментов, как шапка, навигационное меню, нижняя часть страницы. Применяя стиль к , вы выделяете специфические CSS-свойства для всех вложенных тегов, находящихся внутри данного раздела веб-страницы. Чтобы весь текст в навигационном меню был отображен тем же самым цветом, можно создать стиль и применить его к . Все теги, заключенные внутри, в том числе <p>, <h1> и т. д., унаследуют этот цвет шрифта.

ПРИМЕЧАНИЕ

В части 3 этой книги вы узнаете о неисчерпаемых возможностях применения при проектировании дизайна веб-страниц с использованием CSS.

Ограничения наследования

Механизм наследования неидеален. Многие CSS-свойства вообще не наследуются, например border (позволяющий оформить в рамку элемент веб-страницы). Однако если бы наследование применялось к этому свойству, то все теги, вложенные в элемент, были бы одинаковы. Например, если бы вы добавили рамку к <body>, то она была бы во всех маркированных списках (в каждом пункте, подпункте и т. д.) (рис. 4.2).

ПРИМЕЧАНИЕ

Полный список CSS-свойств, включая их подробное описание, параметры наследования и т. д., приведен в приложении 1.



Рис. 4.2. Рамка, относящаяся к абзацу *(вверху)*, не наследуется тегами, находящимися внутри

Ниже приведены конкретные случаи, когда наследование точно не применяется.

- Как правило, свойства, которые затрагивают размещение элементов на странице (отступы (поля), границы (рамки) элементов), не наследуются.
- Браузеры используют свои собственные встроенные стили для форматирования различных тегов. Заголовки обычно отображаются крупным полужирным шрифтом, ссылки — синим цветом и т. д. Даже если определен конкретный размер кегля для текстового содержимого веб-страницы и применен к <body>, заголовки

будут отображены бóльшим по размеру шрифтом. Теги <h1> будут крупнее <h2>. Точно так же, когда вы устанавливаете цвет применительно к <body>, гиперссылки веб-страницы все равно будут отображены синим цветом с подчеркиванием.

ПРИМЕЧАНИЕ

Будет очень полезным устранять встроенные стили браузеров — это упростит создание сайтов, совместимых с различными типами браузеров. В следующей главе вы узнаете, как добиться этого.

Если возникает конфликт, то побеждает более явно определенный стиль. Другими словами, когда вы применяете CSS-свойство к элементу веб-страницы (например, устанавливаете размер шрифта для маркированного списка) и оно конфликтует с наследуемым (например, размером шрифта <body>), то браузер использует явно описанное свойство, более близко относящееся к стилизуемому элементу (в данном случае применяется размер шрифта, определенный для <ul>).

ПРИМЕЧАНИЕ

Такие типы конфликтов между стилями встречаются очень часто, и правила, определяющие, как должен вести себя браузер, называются каскадностью и выявляются приоритетом. Подробнее об этом вы узнаете в следующей главе.

Обучающий урок: наследование

В этом обучающем уроке, состоящем из трех частей, вы увидите, как функционирует механизм наследования. Сначала создадим простой селектор типа и понаблюдаем, каким образом он передает свои стилевые параметры. Создадим класс, который использует наследование для изменения форматирования всей веб-страницы. И наконец, рассмотрим случаи отступления от правил, на которые следует обратить внимание.

Файлы текущей обучающей программы находятся в папке с названием **04** архива, описанного в конце гл. 2.

Одноуровневое наследование

Для того чтобы увидеть и понять, как работает механизм наследования, добавим единственный теговый стиль и посмотрим, как он воздействует на вложенные теги. Все три части этого обучающего урока взаимосвязаны, поэтому сохраняйте свой экспериментальный файл в конце каждого урока для его последующего использования.

1. Откройте файл `inheritance.html` в HTML-редакторе.

   Сейчас добавим в него внутреннюю таблицу стилей.

ПРИМЕЧАНИЕ

Вообще, в сайтах лучше использовать внешние таблицы стилей по причинам, описанным в гл. 2. Иногда проще начать разработку CSS-дизайна отдельных веб-страниц, используя внутреннюю таблицу, как в этом примере, а уже потом преобразовать ее во внешнюю.

2. Щелкните кнопкой мыши сразу после закрывающего тега `</title>`. Нажмите клавишу **Enter** и введите `<style type="text/css">`. Теперь дважды нажмите **Enter**

и создайте закрывающий тег `</style>`, обозначающий конец таблицы стилей. Эти теги определяют область, в которой будут находиться команды языка CSS.

Теперь нужно создать стиль, который будет применен ко всем тегам абзацев <p>.

3. Перейдите к пустой строке между открывающим и закрывающим тегами `<style>` и введите `p {`. Дважды нажмите клавишу **Enter** и поставьте закрывающую скобку `}`. Мы создали селектор <p>, который будет применен ко всем тегам абзаца настоящей веб-страницы.
4. В строке между двумя скобками введите `color: #FF6600`. Стиль теперь должен выглядеть следующим образом:

```
p {
  color: #FF6600;
}
```

С этим свойством мы имели дело в предыдущем обучающем уроке. Оно устанавливает цвет текста. А ваша таблица стилей уже готова.

5. Чтобы посмотреть на результат работы, откройте страницу в браузере.

Цвет всех четырех абзацев страницы изменился с черного на оранжевый (рис. 4.3).

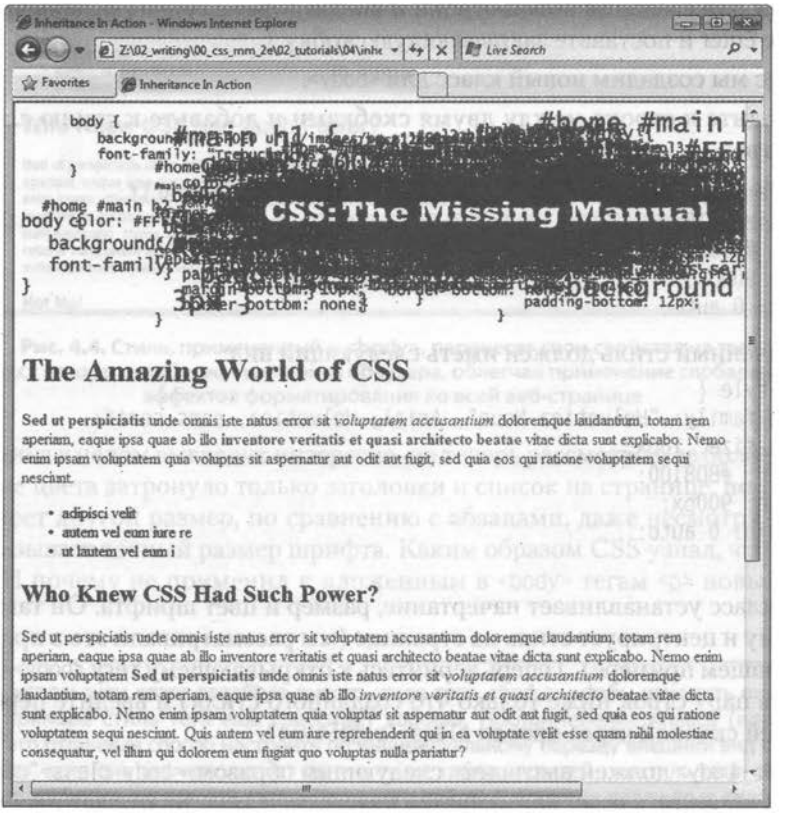


Рис. 4.3. Текст, выделенный полужирным шрифтом, приобрел тот же цвет, что и абзац <p>, окружающий его

Обратите внимание на то, как этот стиль <p> воздействует на другие теги. Они вложены в <p> и также меняют цвет. Например, текст, заключенный в и внутри каждого абзаца, также изменяется на оранжевый, при этом сохраняется выделение полужирным шрифтом. В конечном счете устанавливается цвет текста абзаца, который вы хотели, независимо от любых других тегов.

Без наследования создание таблиц стилей было бы очень трудоемкой задачей: теги , <a> и не унаследовали бы цветового атрибута <p>. Следовательно, пришлось бы создавать дополнительные стили, скорее всего, с селектором `p em` и `p strong`, чтобы правильно отформатировать текст.

Наследование для стилизации веб-страницы

Наследование работает и с классами. Любой тег с примененным к нему стилем переносит CSS-свойства и на производных потомков. Учитывая это, можно пользоваться наследованием для быстрого изменения дизайна всей веб-страницы.

1. Вернитесь к HTML-редактору с файлом `inheritance.html`. Сейчас мы добавим новый стиль вслед за только что созданным стилем тега <p>.
2. Щелкните кнопкой мыши сразу за закрывающей скобкой селектора `p`. Нажмите **Enter** для создания новой строки и введите `.pageStyle {`. Теперь дважды нажмите **Enter** и поставьте закрывающую скобку `}`.

   Сейчас мы создадим новый класс для <body>.
3. Перейдите к строке между двумя скобками и добавьте к стилю следующие свойства:

```
font-family: "Helvetica Neue", Arial, Helvetica, sans-serif;
font-size: 18px;
color: #BD8100;
width: 900px;
margin: 0 auto;
```

   Законченный стиль должен иметь следующий вид:

```
.pageStyle {
  font-family: "Helvetica Neue", Arial, Helvetica, sans-serif;
  font-size: 18px;
  color: #BD8100;
  width: 900px;
  margin: 0 auto;
}
```

4. Этот класс устанавливает начертание, размер и цвет шрифта. Он также задает ширину и центрирует стиль на странице (мы рассматривали это в предыдущем обучающем примере). Теперь вернитесь к открывающему тегу <body> (расположен на пару строк ниже только что созданного стиля) и введите перед закрывающей скобкой через пробел `class="pageStyle"`.

   Сейчас <body> должен выглядеть следующим образом: `<body class="pageStyle">`. Таким образом, к нему применяется класс `.pageStyle`. Благодаря наследованию все теги, заключенные внутри <body> (текст которых отображен в окне браузера), наследуют стилевые свойства и, следовательно, используют тот же шрифт.

5. Сохраните и просмотрите веб-страницу в браузере.

   Как видно из рис. 4.4, наш стилевой класс обеспечил всему тексту веб-страницы согласованный внешний вид, без резких переходов, плавно сочетающий все фрагменты содержимого. И заголовки, и абзацы, заключенные в <body>, — все элементы веб-страницы за счет изменения свойств шрифта приобрели новый стиль.

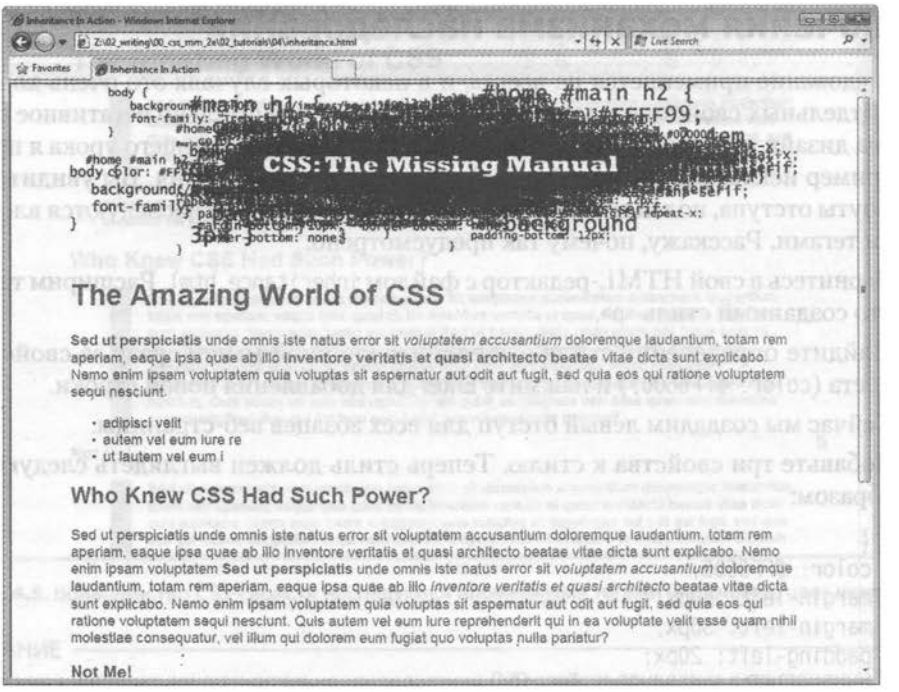


Рис. 4.4. Стиль, примененный к <body>, перенесет свои свойства на теги, текст которых отображается в окне браузера, облегчая применение глобальных эффектов форматирования ко всей веб-странице

Страница в целом выглядит интересно, но теперь рассмотрим ее более детально: изменение цвета затронуло только заголовки и список на странице, но текст заголовка имеет другой размер, по сравнению с абзацами, даже несмотря на то, что стиль указывает точный размер шрифта. Каким образом CSS узнал, что не нужно делать? И почему не применил к вложенным в <body> тегам <p> новый атрибут цвета?

ПРИМЕЧАНИЕ

Почему мы используем класс pageStyle вместо стиля тега <body>, чтобы изменить вид страницы? В данном примере стиль тега еще сработает хорошо. Но применение класса (или ID) к тегу <body> — это отличный способ настроить по индивидуальному образцу внешний вид страниц сайта. Например, если они все используют одну и ту же таблицу стилей, то стиль тега <body> будет применяться к <body> на каждой странице вашего сайта. А создавая различные классы (или ID), вы можете создавать различные стили для тега <body> для разных разделов сайта или разных типов страниц.

Вы видите, как оказывает влияние свойство каскадных таблиц стилей? В этом примере для <p> образовался конфликт стилей, в частности двух одинаковых атрибутов цвета, — тегового, созданного выше, и стилевого класса <body>. Если произошла такая ситуация, то браузер должен выбрать один из стилей. Используется более близкий к элементу стиль — то есть цвет, который вы явно назначили <p>. О правилах каскадности будет рассказано в следующей главе.

Исключения механизма наследования

Наследование применяется не всегда, и в некоторых случаях это очень хорошо. Для отдельных свойств наследование имело бы исключительно негативное влияние на дизайн веб-страницы. В заключительной части обучающего урока я приведу пример исключения (бездействия) механизма наследования. Вы увидите, что атрибуты отступа, полей и границ (среди других свойств) не наследуются вложенными тегами. Расскажу, почему так предусмотрено.

1. Вернитесь в свой HTML-редактор с файлом `inheritance.html`. Расширим только что созданный стиль <p>.
2. Найдите определение стиля `p`, щелкните кнопкой мыши сразу за свойством цвета (`color: #FF6600;`) и нажмите **Enter** для добавления новой строки.

   Сейчас мы создадим левый отступ для всех абзацев веб-страницы.
3. Добавьте три свойства к стилю. Теперь стиль должен выглядеть следующим образом:

```
p {
  color: #FF6600;
  margin-left: 50px;
  margin-left: 50px;
  padding-left: 20px;
  border-left: solid 25px #BD8100;
}
```

   Свойство `margin-left` создает левый отступ размером 50 пикселов, свойство `padding` формирует отступ от границы размером 20 пикселов.
4. Сохраните файл и просмотрите его в браузере.

   Обратите внимание на то, что отступ в 50 пикселов от левого края окна и жирную коричневую границу имеют все абзацы, представленные <p>. Однако у тегов, вложенных в любой из <p> (например, у) нет такого отступа или границы (рис. 4.5). Это поведение браузера оправданно: веб-страница выглядела бы непонятно, если бы каждый тег и в абзаце имел дополнительный левый отступ размером 50 пикселов!

   Если вы хотите увидеть, что бы случилось, если бы эти свойства наследовались, отредактируйте селектор `p` следующим образом: `p, p *`, что сделает его групповым. Первая часть — это тот селектор `p`, который вы только что создали. А вторая часть — `p *` — буквально означает следующее: «выберите все теги внутри <p> и примените стиль к ним» (*, или универсальный селектор, был описан в предыдущей главе).

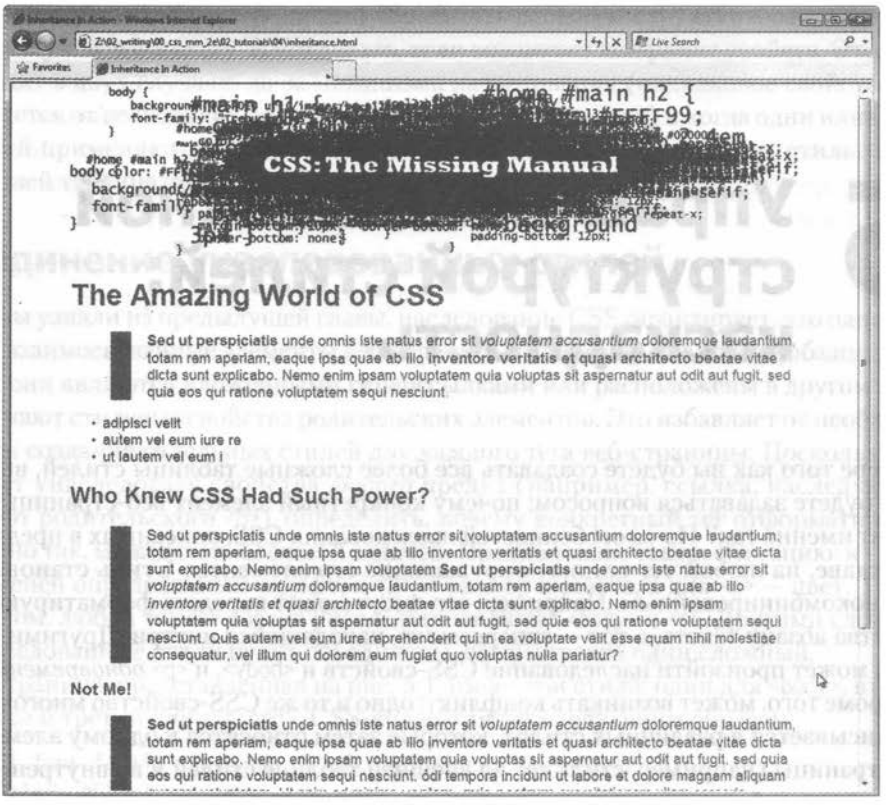


Рис. 4.5. Большинство CSS-свойств наследуются вложенными тегами (например, цвет шрифта)

ПРИМЕЧАНИЕ

Справочная информация относительно наследования CSS-свойств приведена в приложении 1.

ПРИМЕЧАНИЕ

Конечный вариант веб-страницы, которую вы только что создали в этом обучающем уроке, можно найти в папке 04_finished учебного материала.

5 Управление сложной структурой стилей: каскадность

По мере того как вы будете создавать все более сложные таблицы стилей, вы все чаще будете задаваться вопросом: почему конкретный элемент веб-страницы выглядит именно так? Из-за особенностей наследования CSS, описанных в предыдущей главе, на любой тег влияют окружающие его элементы. Стиль становится сложнокомбинированным. Например, тег <body> может передать форматирующие свойства абзацу текста, а тот — гиперссылке, находящейся внутри. Другими словами, может произойти наследование CSS-свойств и <body>, и <p> *одновременно.*

Кроме того, может возникать конфликт: одно и то же CSS-свойство многократно описывается в различных стилях, которые затем относятся к одному элементу веб-страницы (например, стиль <p> во внешней таблице стилей и во внутренней). Вы можете наблюдать такую ситуацию, когда текст отображается ярко-синим цветом, несмотря на то, что установлен красный. Существует особенная система, которая управляет взаимодействием стилей и определяет их приоритет в случае конфликта. Этот механизм называется правилами каскадности, или просто *каскадностью.*

ПРИМЕЧАНИЕ

В данной главе описываются проблемы, возникающие при построении сложных таблиц стилей, работа которых основана на принципах наследования и использовании более сложных типов производных селекторов (см. разд. «ID-селекторы: определение элементов веб-страниц» гл. 3). Все правила достаточно логичны и понятны для опытного веб-дизайнера, но у начинающего все-таки может возникнуть множество сложностей. Можно сравнить обучение этому языку с овладением тонкостями Налогового кодекса. Если у вас нет желания углубленно изучать все особенности языка CSS, просто пропускайте теоретический материал и переходите к выполнению обучающего урока. Вы всегда сможете вернуться к этой главе после того, как овладеете основами CSS.

Каскадность стилей

Каскадность — ряд правил, определяющих, какие именно стилевые свойства необходимы элементам веб-страницы, то есть задающих последовательность применения многократно определенных стилей. Другими словами, каскадность определя-

ет, каким образом браузер должен обработать сложную структуру, относящуюся к одному и тому же тегу, и что делать, если возникает конфликт свойств. Это происходит в двух случаях: из-за механизма наследования (одинаковое свойство наследуется от нескольких родительских элементов-предков) и когда один или более стилей применяются к одному элементу веб-страницы (например, стиль <p> во внешней таблице стилей и <p> во внутренней).

Объединение унаследованных стилей

Как вы узнали из предыдущей главы, наследование CSS гарантирует, что однородные, взаимосвязанные элементы веб-страницы (например, все слова в абзаце, даже если они являются вложенными гиперссылками или расположены в другом теге) получают стилевые свойства родительских элементов. Это избавляет от необходимости создания отдельных стилей для каждого тега веб-страницы. Поскольку тег может унаследовать свойства *любого* предка (например, ссылка, наследующая шрифт родительского <p>), определить, почему конкретный тег отформатирован именно так, может быть сложной задачей. Предположим такую ситуацию: к <body> применен определенный шрифт, к <p> — размер шрифта, а для <a> — цвет. Таким образом, любой тег <a> абзаца унаследует свойства <body> и <p>. Другими словами, унаследованные стили будут объединены, сформировав один сложный.

Страница, представленная на рис. 5.1, имеет три стиля: один для <body>, второй для <p> и третий для . CSS-код выглядит следующим образом:

```
body { font-family: Verdana, Arial, Helvetica, sans-serif; }
p { color: #F30;}
strong { font-size: 24px; }
```

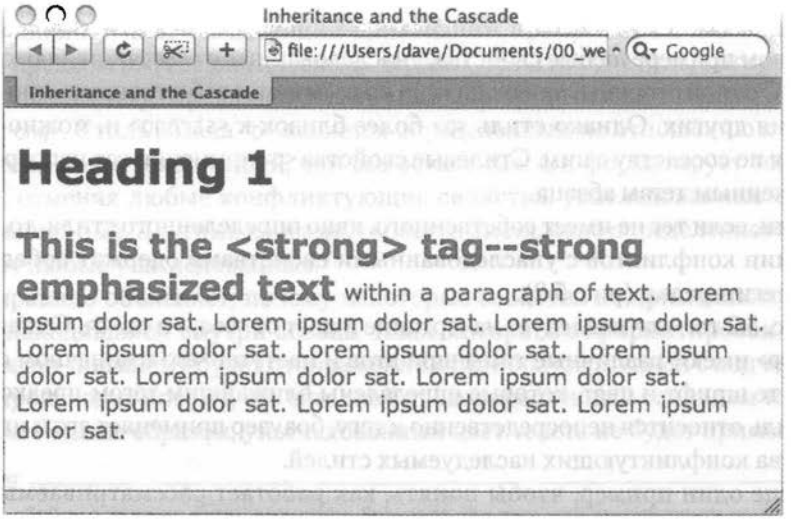


Рис. 5.1. Благодаря наследованию можно воспроизводить один тег несколькими стилями

Тег вложен в абзац, который, в свою очередь, является вложенным в <body>. Стилевые свойства наследует у всех элементов-предков, получая начертания шрифта font-family от <body> и цвет color от родительского абзаца. Кроме того, имеет собственное CSS-свойство, устанавливающее размер шрифта 24 пиксела. Конечный внешний вид тега определяется сочетанием всех трех стилей. Другими словами, выглядит так, будто для него создали следующий стиль:

```
strong {
  font-family: Verdana, Arial, Helvetica, sans-serif;
  color: color: #F30;
  font-size: 24px;
}
```

Превосходство близкого родительского элемента-предка

Как видно из предыдущего примера, комбинирование стилей различных тегов с применением наследования создает полный стилевой пакет. Но что произойдет в случае конфликта унаследованных свойств CSS? Вспомните веб-страницу, в которой вы устанавливали цвет шрифта для <body> и <p>.

На рис. 5.1 отформатирован шрифтом определенного начертания, размера и цвета, несмотря на то что для него в стиле явно определено только одно свойство. Два других унаследованы от тегов-предков <body> и <p>.

Теперь предположим, что внутри абзаца имеется тег . Какого цвета в нем будет текст? Тега <body> или <p>? Ответ прост: цвета абзаца. Браузер применит стиль, который является самым близким по отношению к рассматриваемому тегу.

В данном примере любые свойства, унаследованные от <body>, являются скорее общими. С одной стороны, они относятся ко всем тегам веб-страницы, при условии отсутствия других. Однако стиль <p> более близок к и, можно сказать, находится по соседству с ним. Стилевые свойства <p> применяются непосредственно к вложенным тегам абзаца.

По сути, если тег не имеет собственного, явно определенного стиля, то при возникновении конфликтов с унаследованными свойствами одержат победу самые близкие теги-предки (рис. 5.2).

На рис. 5.2 тег наследует начертание и цвет от <body> и от <p>. Однако стили <body> и <p> имеют различные типы шрифтов и цвет, тег в конечном счете использует те шрифт и цвет, которые определены ближайшим тегом-предком — <p>. Когда стиль относится непосредственно к тегу, браузер применяет его и игнорирует свойства конфликтующих наследуемых стилей.

Это еще один пример, чтобы понять, как работает рассматриваемый механизм. Если имеется CSS-стиль, определяющий цвет текста для таблицы <table>, и еще один, определяющий *другой* цвет для <td>, то теги, заключенные в ячейки

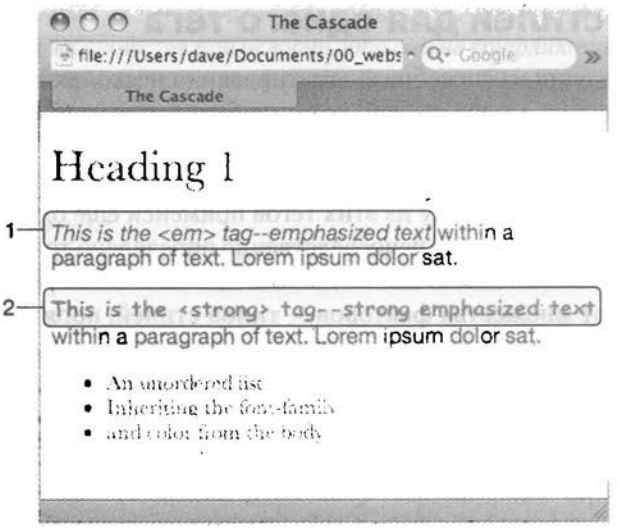


Рис. 5.2. Браузеры определяют, какие свойства применять в случае возникновения конфликтов с наследуемыми стилями

данной таблицы (<td>), — теги абзаца, заголовка, маркированного списка — унаследуют цвет стиля <td>, поскольку он является самым близким родительским тегом-предком.

Преимущества непосредственно примененного стиля

Единственный стиль, обладающий наивысшим приоритетом, является самым близким предком в «генеалогическом» дереве CSS. Это тот стиль, который напрямую применен к тегу. Предположим, что цвет шрифта устанавливается для <body>, <p> и . Стиль абзаца <p> является определенным по отношению к <body>, но еще более конкретный, чем все остальные. Он форматирует только теги , отменяя любые конфликтующие свойства, унаследованные от других тегов (см. рис. 5.2). Другими словами, свойства стиля, явно определенного для тега, отменяют любые унаследованные.

Это правило объясняет, почему некоторые свойства не применяются. Гиперссылка, находящаяся внутри абзаца, текст которого отформатирован красным шрифтом, по-прежнему будет отображена в стиле браузера синей с подчеркиванием. Это происходит потому, что есть свой предопределенный стиль для тега гиперссылки <a>. Таким образом, унаследованный цвет текста не будет применен.

ПРИМЕЧАНИЕ

О том, как обойти встроенные теговые стили браузера для тега <a> и установить желаемый стиль для гиперссылок, я расскажу в гл. 9.

Множество стилей для одного тега

Наследование — один из способов форматирования несколькими стилями. Однако можно определить много стилей, которые будут применены *непосредственно*. Рассмотрим такой случай: у нас имеется прикрепленная к веб-странице внешняя таблица стилей с тегом <p>. В ней есть внутренняя таблица, которая *также* содержит определение <p>. К одному из этих тегов применен еще один стилевой класс. Так, для одного тега абзаца <p> непосредственно определено три различных явных форматирующих стиля. Какой из них должен использовать браузер?

Все зависит от множества факторов — типов стилей, порядка, в котором они были созданы. Браузер сам выбирает: применить один или несколько одновременно. Рассмотрим ситуации, когда множественные стили могут применяться к одному и тому же тегу.

- **К тегу одновременно применен стиль с селектором и стилевой класс.** Например, для тега <h2> это `.leadHeadline`. HTML-код имеет вид: `<h2 class="leadHeadline">Your Future Revealed!</h2>`. К <h2> будут применены оба стиля.

ПРИМЕЧАНИЕ

Описание того, что произойдет в случае конфликта стилей, следует далее.

- **Одинаковое название стиля встречается в таблице несколько раз.** Это может быть групповой селектор (см. разд. «Стилизация групп тегов» гл. 3). Например, `.leadHeadline, .secondaryHeadline, .newsHeadline` и стилевой класс `.leadHeadline` в той же таблице. Форматирование элемента `leadHeadline` будет определяться обоими стилями.
- **К тегу одновременно применены стилевой класс и ID-стиль.** Это может быть ID-селектор `#banner` и класс `.news`. HTML-код имеет вид: `<div id="banner" class="news">`. К будут применены оба стиля.
- **С веб-страницей связано несколько таблиц, и в каждой из них содержится одинаковое название стиля.** Такие имена могут быть присоединены к веб-странице правилом `@import` посредством связывания с внешней таблицей или путем непосредственного включения внутренней.
- **Единственный тег веб-страницы может быть объектом воздействия сложных селекторов.** Это обычная ситуация, когда вы используете производные селекторы (см. разд. «Селекторы классов: точное управление» гл. 3). Допустим, на веб-странице имеется тег (например: `<div id="mainContent">`) и внутри его заключен абзац со стилевым классом: `<p class="byline">`. Здесь будут применены следующие селекторы:
  - `#mainContent p;`
  - `#mainContent .byline;`
  - `p.byline;`
  - `.byline.`

Если к конкретному элементу веб-страницы применено несколько стилей, то браузер объединяет их свойства *при условии, что они не конфликтуют между собой.* Приведенный ниже пример разъясняет этот принцип. Предположим, есть абзац, в котором указаны имя автора веб-страницы и ссылка на адрес его электронной почты. HTML-код может выглядеть следующим образом:

```
<p class="byline">Written by <a href="mailto:jean@cosmofarmer.com">Jean
  Graine de Pomme</a></p>
```

Между тем в таблице стилей веб-страницы есть три стиля для форматирования гиперссылки:

```
a { color: #6378df; }
p a { font-weight: bold }
.byline a { text-decoration: none; }
```

Первый стиль окрашивает элемент тега <a> в зеленовато-голубой цвет; второй стиль форматирует все <a>, находящиеся в <p>, полужирным шрифтом; а третий стиль убирает подчеркивание ссылок, вложенных в элементы, принадлежащие стилевому классу `byline`.

Воздействие всех трех стилей распространяется на такой часто используемый тег, как <a>. Ни одно из свойств этих стилей не конфликтует с остальными. Ситуация похожа на случай, рассмотренный выше, в подразделе «Объединение унаследованных стилей»: стили объединяются между собой и образуют один комбинированный суперстиль, содержащий все три свойства. Таким образом, данная гиперссылка отображается зеленовато-голубым полужирным шрифтом без подчеркивания.

ПРИМЕЧАНИЕ

Имейте в виду, что на стиль форматирования этой ссылки также могут влиять переданные свойства. Например, может быть унаследовано начертание шрифта абзаца. Лучше понять работу механизма каскадности вам помогут несколько инструментов, описанных во врезке «Часто задаваемые вопросы» ниже.

Особенности механизма каскадности: какие стили имеют преимущество

Предыдущий пример слишком прост. Но что получится, если каждый из трех стилей, приведенных выше, имеет свое определение начертания в свойстве `font-family`? Какой из них выберет браузер?

Если вы внимательно читали книгу, то помните, что механизм каскадности устанавливает несколько правил. *Побеждают (имеют преимущество) свойства самого близкого по отношению к стилизуемому элементу, самого явно определенного стиля.* Однако, как и в примере со стилями, не совсем понятно, какой из них является наиболее определенным. К счастью, CSS предлагает метод определения *приоритетов.* Он основан на присвоении значений в условных единицах (пунктах)

каждому типу стилевых селекторов: тегов, классов, ID-селекторам и т. д. Система работает так.

- Селектор тегов имеет значимость 1 пункт.
- Селектор классов — 10 пунктов.
- ID-селектор — 100.
- Встроенный (inline) стиль — 1000.

ПРИМЕЧАНИЕ

Математические расчеты, используемые для определения приоритетов на самом деле немного более сложные. Но эта формула работает во всех случаях, кроме самых странных и запутанных. Чтобы узнать о том, как браузеры рассчитывают приоритеты, посетите страницу www.w3.org/TR/CSS21/cascade.html#specificity.

Чем больше числовое значение, тем выше значимость данного типа селектора. Предположим, вы создали три стиля:

- теговый стиль для `<img>` (значимость = 1);
- стилевой класс `.highlight` (значимость = 10);
- ID-стиль `#logo` (значимость = 100).

Веб-страница содержит следующий HTML-код: `<img id="logo" class="highlight" src="logo.gif" />`. Если определить одинаковый атрибут во всех трех стилях (например, рамка `border`), то будет применено значение атрибута ID-стиля (`#logo`), как наиболее значимого.

ПРИМЕЧАНИЕ

Псевдоэлемент (например, :first-child) обрабатывается браузером как теговый селектор и имеет значимость 1 пункт. Псевдокласс (например, :link) рассматривается как класс и имеет значимость 10 пунктов (см. разд. «Псевдоклассы и псевдоэлементы» гл. 3).

Поскольку производные селекторы состоят из нескольких простых — например, `content p` или `h2 strong`, — определить их значимость сложнее: необходимо найти суммарное значение их приоритетов (табл. 5.1).

Таблица 5.1. Когда к единственному тегу применяется несколько стилей, браузер должен определить, какой из них будет применен в случае возникновения конфликта

Селектор	Идентификатор	Класс	Тег	Итого
p	0	0	1	1
.byline	0	1	0	10
p.byline	0	1	1	11
#banner	1	0	0	100
#banner p	1	0	1	101
#banner .byline	1	1	0	110
a:link	0	1	1	11
p:first-line	0	0	2	2
h2 strong	0	0	2	2
#wrapper #content .byline a:hover	2	2	1	221

ПРИМЕЧАНИЕ

Наследуемые свойства вообще лишены такого показателя, как значимость. Так, даже если тег наследует, например, селектор #banner, то эти свойства в любом случае будут заменены теми, что непосредственно относятся к этому тегу.

Разрешение конфликтов: побеждает последний стиль. Два стиля могут иметь одинаковый приоритет. Конфликт значимостей может произойти в случае двойного определения одинаковых селекторов. У вас может быть селектор тега <p> во внутренней таблице и такой же во внешней. Или два различных стиля могут иметь одинаковый приоритет. В таком случае более значимым будет последний определенный стиль таблицы.

Вот пример HTML-кода.

```
<p class="byline">Written by <a class="email"
 href="mailto:jean@cosmofarmer.com">Jean Graine de Pomme</a></p>
```

В таблице для веб-страницы, содержащей вышеприведенные абзац и гиперссылку, у вас будет два стиля:

```
.email { color: blue; }
.byline a { color: red; }
```

Оба стиля имеют значимость, равную 11 (10 — для названия класса и 1 — для селектора тега), и относятся к <a>. Конфликт этих стилей очевиден. Какой цвет выберет браузер для окрашивания гиперссылки в приведенном абзаце? Красный, так как он последний в таблице стилей.

ЧАСТО ЗАДАВАЕМЫЕ ВОПРОСЫ

Инструменты для облегчения работы

Может возникнуть вопрос: не существует ли какого-нибудь вспомогательного средства, чтобы представить в понятной форме воздействие, оказываемое механизмом каскадности на конечный дизайн веб-страницы?

Есть несколько инструментов, которые могут наглядно изобразить каскадность. Программный пакет Dreamweaver (www.adobe.com) имеет удобную панель CSS. Одного взгляда достаточно, чтобы увидеть эффект воздействия данного механизма на произвольно выбранный элемент веб-страницы. Другими словами, вы моментально получаете подробнейший список примененных к конкретному элементу свойств стилей. Для этих целей также подойдет бесплатное расширение браузера Firefox — View Formatted Source (https://addons.mozilla.org/extensions/moreinfo.php? application=firefox&id=697). Оно позволяет просмотреть, какие стили применены к определенному элементу веб-страницы (однако не показывает наследуемые стили).

И наконец, есть браузер от Apple — Safari, который используют как для Mac, так и для Windows. Встроенное средство под названием Web Inspector (Анализатор) предоставляет полнейшую информацию о веб-странице, ее CSS-стилях, влиянии механизма каскадности на теги. Вам нужно лишь установить флажок Show Develop Menu (Показывать развернутое меню) на вкладке Advanced (Дополнительно) окна Preferences (Параметры). Более подробную информацию об использовании Web Inspector можно найти по адресу http://tinyurl.com/web-inspector.

Теперь представьте, что таблица стилей имеет следующий вид:

```
p .email { color: blue; }
.byline a { color: red; }
```

В данном случае гиперссылка была бы красного цвета. Стиль с селектором `.byline a` расположен в таблице после `p .email`, поэтому его свойства имеют преимущество.

Теперь рассмотрим, что произойдет, если имеются конфликтующие стили (или их свойства) во внешней и во внутренней таблицах стилей. В этом случае важна последовательность размещения на веб-странице (в HTML-файле). Если вы сначала добавляете внутреннюю таблицу, используя `<style>` (см. гл. 2), а только затем присоединяете внешнюю далее по тексту HTML, используя `<link>`, то будет применен стиль последней (в сущности, это принцип, который только что был описан: значение имеет последний из конфликтующих стилей). Вывод: будьте последовательны в размещении на веб-странице внешней таблицы стилей. Сначала ее нужно присоединить, а только затем включать внутренние таблицы.

СОВЕТ

Любые внешние таблицы стилей, присоединяемые директивой @import, должны находиться до внутренних, заключенных в тег <style>. Для получения дополнительной информации о внешних и внутренних таблицах стилей см. гл. 2.

БРИЛЛИАНТ БЕЗ ОГРАНКИ

Отмена правил значимости

CSS предоставляет возможность полностью отменить значимость стилей. Вы можете использовать этот прием, чтобы никакой другой более значимый стиль не отменил конкретное свойство элемента веб-страницы. Просто вставьте после него значение `!important`.

Рассмотрим пример. У нас есть два стиля:

```
#nav a { color: red; }
a { color: teal !important; }
```

При обычном раскладе ссылка, вложенная в элемент с идентификатором `#nav`, была бы окрашена в красный цвет, так как стиль, определенный селектором `#nav a`, более значимый, чем `<a>`. Однако добавление `!important` подразумевает, что данное свойство всегда будет иметь больший приоритет. Так, в вышеприведенном примере все ссылки веб-страницы, в том числе вложенные с идентификатором `#nav`, будут отображены зеленовато-голубым цветом.

Обратите внимание, что вы применяете `!important` к отдельному свойству, а не ко всему стилю.

В заключение нужно сказать: когда для двух одинаковых свойств различных стилей указано `!important`, опять вступает в силу правило значимости и приоритет имеет более значимый атрибут из отмеченных.

Браузер Internet Explorer 6 не всегда правильно обрабатывает свойства с `!important`, а иногда полностью их игнорирует.

Управление каскадностью

Как вы могли заметить, чем больше CSS-стилей создано, тем больше вероятность запутаться в них. Например, можно создать стилевой класс, устанавливающий для

шрифта определенное начертание и размер, но применение его к абзацу ни к чему не приводит. Эта проблема обычно связана с механизмом каскадности. Даже когда вы абсолютно уверены в конечном результате, все равно может существовать стиль с большей значимостью.

Есть несколько вариантов решения этой проблемы. Во-первых, можно использовать `!important` (как описано в выделенном блоке выше), чтобы гарантировать применение конкретного свойства. Этот способ не совсем удобен, так как трудно предугадать, что вы не захотите отменить такую значимость свойства. Рассмотрим два других метода управления каскадностью.

Изменение значимости

На рис. 5.3 приведен пример, когда определенный стиль тега проигрывает в каскадной игре. Здесь два стиля форматируют первый абзац. Стилевой класс `.intro` не столь значимый, как `#sidebar p`. Таким образом, свойства `.intro` не будут применены к абзацу. Чтобы увеличить значимость, добавьте к стилю идентификатор `#sidebar .intro`.

На рис. 5.3 абзац размещен внутри тега `<div>` с идентификатором `#sidebar`. Таким образом, производный селектор `#sidebar p` является более значимым по сравнению со стилевым классом `.intro` (соотношение значимостей — 101 к 10). Вывод: необходимо придать `.intro` бóльшую значимость, добавив перед ним идентификатор, как на рис. 5.4: `#sidebar p.intro`.

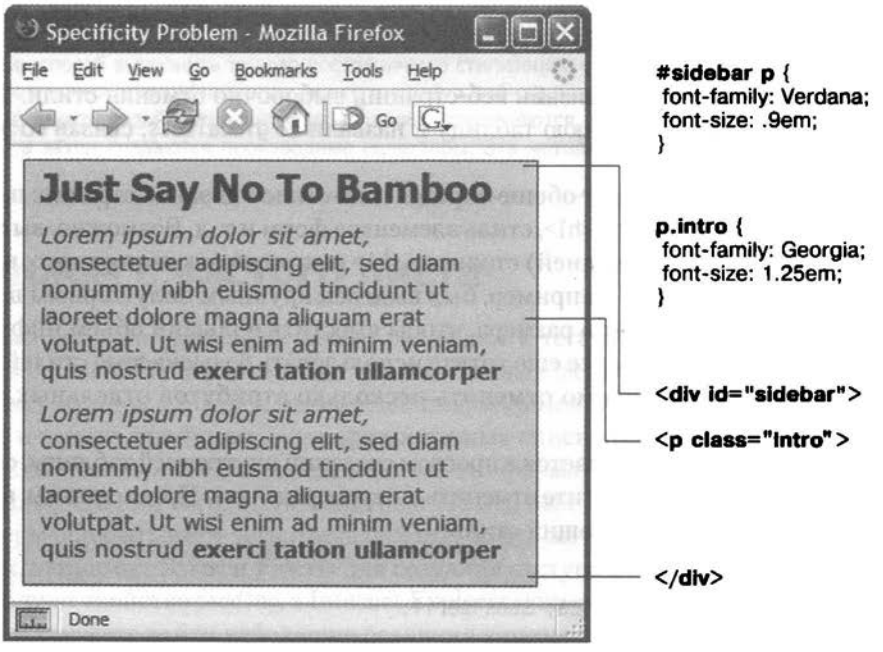


Рис. 5.3. Даже если к определенному тегу применен стилевой класс, его свойства не всегда имеют эффект

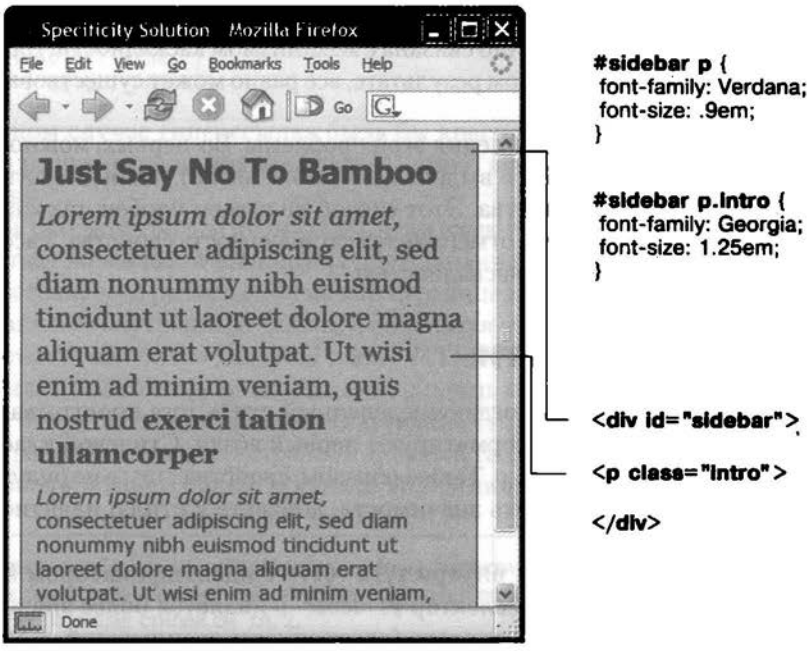


Рис. 5.4. Можно придать бóльшую значимость стилевому классу, добавив перед ним идентификатор

Выборочная отмена значимости

Можно точно проработать дизайн веб-страниц, выборочно отменяя стили. Предположим, вы создали внешнюю таблицу и назвали ее `global.css`, связав со всеми страницами сайта.

Этот CSS-файл содержит общие определения стилей дизайна страниц: шрифт и цвет для тегов заголовков `<h1>`, стиль элементов форм и т. д. Возможно, вы хотите, чтобы на главной (домашней) странице `<h1>` выглядел отлично от того, как он отображен на остальных. Например, был выделен крупным полужирным шрифтом или шрифтом меньшего размера, чтобы вместить больший объем информации. Другими словами, вы все еще хотите использовать *большинство* стилей файла `global.css`, но необходимо отменить несколько атрибутов отдельных тегов (`<h1>`, `<p>` и т. д.).

Один из способов заключается в простом создании внутренней таблицы, содержащей стили, которые вы хотите отменить и переопределить. Предположим, в файле `global.css` имеется следующий стиль:

```
h1 {
  font-family: Arial, Helvetica, sans-serif;
  font-size: 24px;
  color: #000;
}
```

Вы хотите, чтобы заголовок <h1> главной веб-страницы был отображен крупным шрифтом красного цвета. Просто добавьте во внутреннюю таблицу следующий стиль:

```
h1 {
  font-size: 36px;
  color: red;
```

В данном случае к тегу <h1> главной страницы сайта будет применен шрифт Arial (из внешней таблицы стилей), но в то же время он будет окрашен в красный цвет размером 36 пикселов (определенные во внутренней).

СОВЕТ

Убедитесь, что вы присоединяете внешнюю таблицу стилей перед внутренней в разделе HTML-заголовка <head>. Это гарантирует, что нужные стили будут иметь преимущество в тех случаях, когда значимость одинаковая.

Другой метод заключается в создании еще одной внешней таблицы. Например, таблицы `home.css`, которую нужно будет присоединить к главной веб-странице в дополнение к `global.css`. Файл `home.css` будет содержать те стили `global.css`, которые вы хотите переопределить. Для правильной работы файл `home.css` должен быть присоединен *после* `global.css` в HTML-коде главной веб-страницы:

```
<link rel="stylesheet" type="text/css" href="css/global.css">
<link rel="stylesheet" type="text/css" href="css/home.css">
```

СОВЕТ

Еще один способ выполнить точную постраничную стилизацию веб-страниц основан на использовании различных идентификаторов для тега <body> веб-страниц разного содержания. Например, чтобы изменить дизайн отдельных веб-страниц, применяются идентификаторы #review, #story, #home, а затем создаются производные селекторы. Эта методика описана в обучающем уроке гл. 9.

Начинаем с чистого листа

Как было сказано ранее, браузеры применяют к тегам свои собственные стили: например, теги <h1> больше тегов <h2>, они оба выделены полужирным, в то время как текст абзацев меньше и не выделен полужирным шрифтом; ссылки подчеркнутые и имеют синий цвет, а у маркированных списков есть отступ. В стандарте HTML нет ничего, что бы определяло все это форматирование: браузеры просто добавляют его для того, чтобы обычный HTML был более читабельным. Разные браузеры обрабатывают теги очень похоже, но все же неодинаково.

Так, например, Safari и Firefox для создания отступа в маркированных списках используют свойство `padding`, а Internet Explorer применяет свойство `margin`. Кроме того, вы сможете найти небольшие различия в размерах тегов в разных браузерах и обнаружить вовсе вводящее в заблуждение использование отступов самыми распространенными на сегодняшний день браузерами. Из-за этих несоответствий вы

столкнетесь с проблемами, когда, например, Firefox добавит отступ от верхнего края, а Internet Explorer этого не сделает. Такого рода проблемы не ваша вина — они вытекают из различий во встроенных в браузер стилях.

Для предотвращения кроссбраузерного несоответствия лучше всего начинать таблицу стилей с чистого листа. Другими словами, удалить встроенное в браузер форматирование и добавить свое собственное. Концепция устранения стилей браузера называется *сбросом стандартных стилей (CSS-сбросом)*. В этом разделе я введу вас в суть дела.

В частности, есть базовый набор стилей, который вы должны включить в верхнюю часть своей таблицы стилей. Они устанавливают базовые значения для свойств, которые обычно по-разному обрабатываются во всех браузерах.

Рассмотрим шаблон сброса стандартных стилей:

```
html, body, h1, h2, h3, h4, h5, h6, p, ol, ul, li, pre, code, address,
variable, form, fieldset, blockquote {
  padding: 0;
  margin: 0;
  font-size: 100%;
  font-weight: normal;
}
ol {
  margin-left: 1.4em;
  list-style: decimal;
}
ul {
  margin-left: 1.4em;
  list-style:square;
}
img {
  border: 0;
}
```

Первый стиль — очень длинный групповой селектор, затрагивающий наиболее распространенные теги и обнуляющий их. Он удаляет поля и отступы, устанавливая 100%-ный размер шрифта и убирая выделения полужирным. Благодаря этому шагу ваши теги смотрятся практически одинаково (рис. 5.5). Но так и нужно — ведь вы хотите начать с чистого листа, а затем добавить собственное форматирование, чтобы все браузеры согласованно отображали ваш HTML-код.

ПРИМЕЧАНИЕ

Вам не нужно вводить этот код самостоятельно. Вы найдете файл с именем reset.css в папке 05 обучающих материалов с сайта www.sawmac.com/css2e, где есть базовый файл для сброса стандартных стилей. Просто скопируйте стили из этого файла и вставьте их в свои собственные таблицы стилей. Более комплексный сброс стандартных стилей CSS (будет обсуждаться далее в этой книге) доступен внутри папки 15 обучающих материалов для гл. 15.

Второй и третий стили (для тегов `<ol>` и `<ul>`) устанавливают согласованные отступы от левого края и определенное форматирование (далее в книге будет представлена стилизация списков), а последний стиль устраняет границу, которую отдельные браузеры добавляют к изображениям, являющимся ссылками.

Обучающий урок: механизм каскадности в действии

В этой обучающей программе вы увидите, как элементы взаимодействуют между собой и конфликтуют, что приводит к неожиданным результатам. Для начала посмотрите на базовую страницу, упомянутую выше, на которой были сброшены стандартные стили. Кроме того, есть несколько других стилей, обеспечивающих простую разметку. Затем мы создадим два стиля и понаблюдаем за действием механизма каскадности. Мы также рассмотрим, как наследование влияет на теги веб-страницы и как браузер решает конфликты CSS-стилей. Наконец, вы узнаете, как решаются проблемы механизма каскадности.

Чтобы начать обучающий урок, вы должны загрузить файлы с учебным материалом. Как это сделать, описывается в конце гл. 2. Файлы текущей обучающей программы находятся в папке 05.

Сброс стандартных стилей и создание стилей с чистого листа

Для начала посмотрите на ту страницу, с которой будете работать.

1. В браузере откройте страницу `cascade.html` из папки 05 (см. рис. 5.5).

   Страница выглядит очень просто: два столбца (один из них имеет синий фон) и много однотипного текста. К этому файлу уже применялись некоторые стили, поэтому откройте CSS-код в текстовом редакторе и просмотрите его.

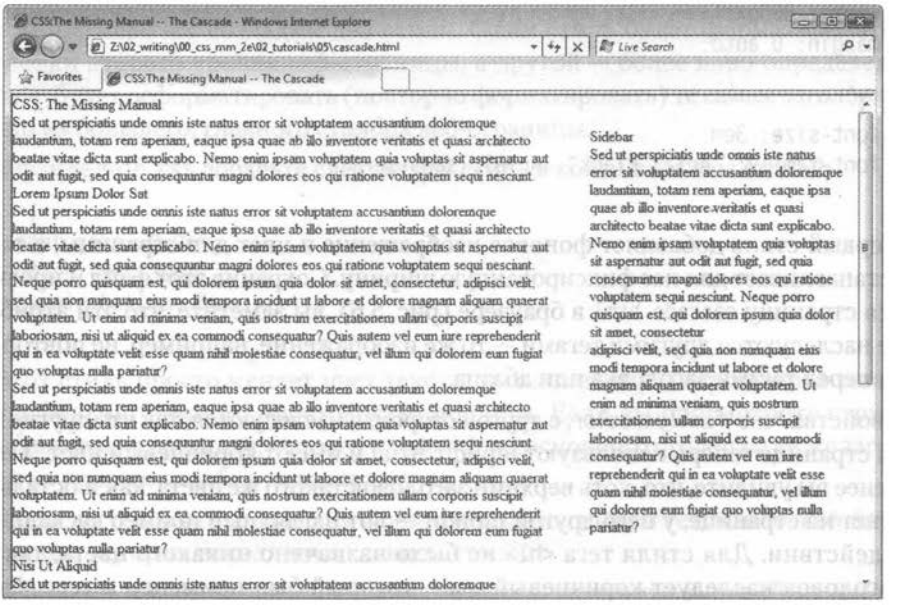


Рис. 5.5. Базовый сброс стандартных стилей на этой странице устраняет небольшие различия в том, как разные браузеры отображают основные теги HTML

2. Откройте файл main.css из папки 05 в текстовом редакторе либо редакторе веб-страниц.

   Это внешняя таблица стилей, которую использует файл cascade.html. В ней уже есть шесть стилей: первые четыре сбрасывают стандартные стили, что мы обсуждали на предыдущей странице. Они устраняют основные стили браузера, поэтому весь текст пока выглядит одинаково. Скоро вы будете создавать свои собственные стили, чтобы эта страница смотрелась более эффектно.

3. Последние два стиля — ID-стили #main и #sidebar — создают два столбца, которые вы видели на рис. 5.5. HTML разделен на два тега , каждый из которых имеет свой собственный идентификатор. ID-стили здесь, по существу, размещают два тега так, чтобы они отображались бок о бок в виде столбцов (как именно работают ID-стили, вы узнаете, когда ближе познакомитесь с разметкой CSS, начиная с гл. 10.)

   Сначала вы добавите пару стилей, чтобы улучшить общий вид страницы и ее верхний заголовок.

4. В файле main.css добавьте два этих стиля в нижней части таблицы стилей за последней скобкой } стиля #sidebar:

```
body {
  color: #B1967C;
  font-family: "Palatino Linotype", Baskerville, serif;
  padding-top: 100px;
  background: #CDE6FF url(images/bg_body.png) repeat-x;
  width: 800px;
  margin: 0 auto;
}
h1 {
  font-size: 3em;
  font-family: "Arial Black", Arial, sans-serif;
}
```

   Первый стиль добавляет фоновое изображение и цвет для страницы, а также устанавливает для нее фиксированную ширину. Сохранив этот файл и просмотрев страницу cascade.html в браузере (рис. 5.6), вы заметите, что эти атрибуты не наследуются другими тегами — то же изображение, например, не повторяется перед тегами заголовка или абзаца.

   Свойства font-family и color, с другой стороны, наследуются, так что другие теги на странице теперь используют шрифт Arial и имеют коричневый цвет. Тем не менее вы увидите, что, хоть верхний заголовок такого же цвета, как и остальной текст на странице, у него другой шрифт — вот наглядный пример каскадности в действии. Для стиля тега <h1> не было назначено никакого цвета, так что заголовок наследует коричневый цвет, который был применен к тегу <body>. Но, поскольку тег <h1> определяет начертание, он замещает унаследованный шрифт от стиля тега <body>.

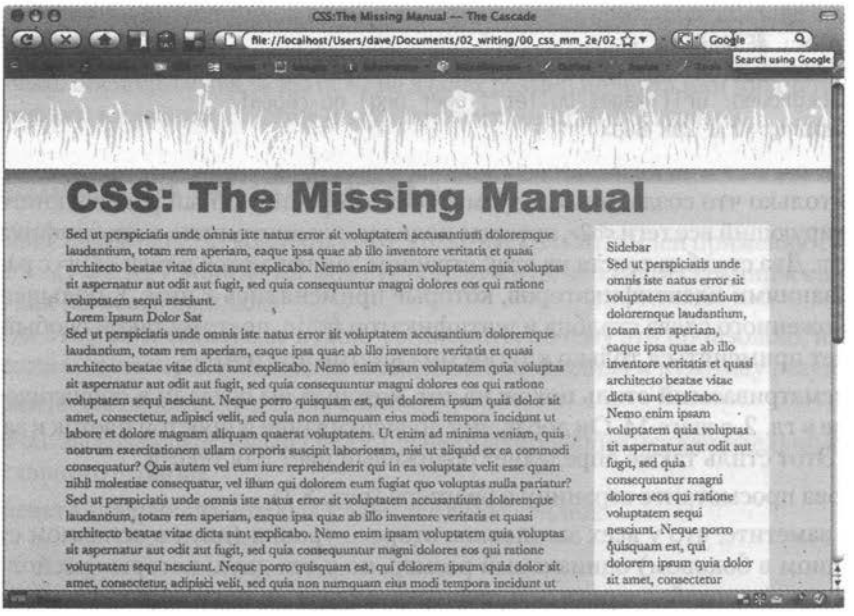


Рис. 5.6. Наследование и каскадность в действии: тег <h1> вверху этой страницы наследует ее шрифт из стиля тега <body>, но получает размер и начертание шрифта из специального стиля тега <h1>

Создание комбинированных стилей

В этом примере мы создадим два стиля. Один стиль будет форматировать все заголовки второго уровня веб-страницы, а другой — более явно определенный стиль — будет реформатировать (повторно форматировать) те самые заголовки, но только из большего, главного столбца веб-страницы.

1. В файле `main.css` добавьте следующий стиль в конец таблицы стилей:

```
h2 {
  font-size: 2.2em;
  color: #AFC3D6;
  margin-bottom: 5px;
}
```

Этот стиль просто меняет цвет текста, увеличивает размер шрифта тега <h2> и добавляет немного пространства под ним. Если вы просмотрите страницу в браузере, то увидите, что заголовки <h2> из основного столбца и те же заголовки из правого столбца похожи друг на друга.

Далее вы создадите стиль для форматирования только тех заголовков второго уровня, которые находятся в главном столбце.

2. Вернитесь в свой редактор к файлу `main.css`, щелкните кнопкой мыши сразу после окончания стиля тега <h2> и нажмите клавишу Enter, чтобы создать пустую строку. Добавьте следующий стиль:

```
#main h2 {
  color: #E8A064;
  border-bottom: 2px white solid;
  background: url(images/bullet_flower.png) no-repeat;
  padding: 0 0 2px 80px;
}
```

Вы только что создали наследуемый селектор, описанный ранее в книге, форматирующий все теги <h2>, которые появляются внутри тега с идентификатором #main. Два столбца текста на этой странице заключаются в теги с разными названиями идентификаторов, которые применялись к ним. У большего, расположенного слева столбца идентификатор #main, поэтому такой особый стиль будет применяться только к тегам <h2> внутри этого .

Рассматриваемый стиль похож на тот, который вы создали в обучающем примере в гл. 2, в шаге 14. Он добавляет подчеркивание и простой цветок к заголовку. Этот стиль также определяет оранжевый цвет для текста.

3. Снова просмотрите страницу в браузере (рис. 5.7).

   Вы заметите, что у всех заголовков второго уровня (двух в основном столбце и одном в боковом) одинаковый размер, но у тех двух, которые расположены в основном столбце, также есть подчеркивающая линия и изображение цветка.

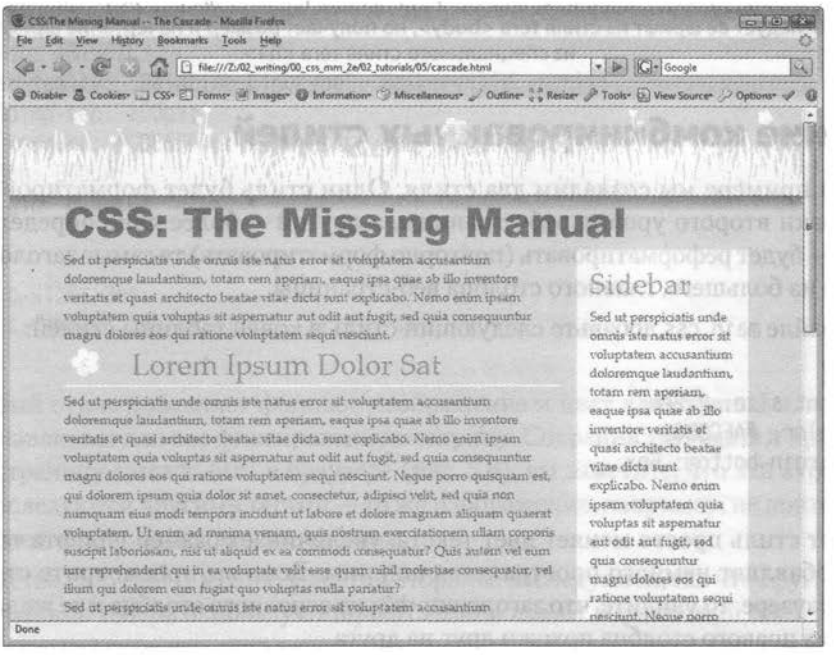


Рис. 5.7. Стили h2 и #main h2 применяются к заголовкам второго уровня из левого столбца этой страницы, причем стиль #main h2 — только к ним

Поскольку стиль #main h2 является более специфичным по сравнению с простым стилем h2, при возникновении каких-либо конфликтов между ними (в данном случае в свойстве color) свойства #main h2 побеждают. Таким образом, хотя заго-

ловки второго уровня в основном столбце получают синий цвет текста от стиля h2, оранжевый цвет от более специфичного стиля #main h2 побеждает.

Однако, поскольку стиль #main h2 не задает размер шрифта или нижнего отступа, заголовки в главном столбце получают эти свойства от стиля h2.

Преодолеваем конфликты

Поскольку свойства CSS конфликтуют, когда несколько стилей применяются к одному и тому же тегу, иногда вы можете обнаружить, что ваши страницы выглядят не так, как вы этого ожидали.

Когда это происходит, вам нужно установить, почему это произошло, и внести изменения в селекторы CSS, чтобы каскадность приводила к нужному результату.

1. Вернитесь к редактору и файлу main.css.

   Сейчас вы создадите новый стиль для форматирования исключительно абзацев из главного столбца.

2. Добавьте следующий стиль в конец таблицы стилей:

```
#main p {
  color: #616161;
  font-family: "Palatino Linotype", Baskerville, serif;
  font-size: 1.1em;
  line-height: 150%;
  margin-bottom: 10px;
  margin-left: 80px;
}
```

   Этот стиль изменяет цвет, размер и шрифт текста, растягивает строки (свойство line-height) и регулирует нижние и левые отступы абзацев.

   Страница будет выглядеть лучше, если вы выделите абзац, следующий сразу за заголовком. Сделав этот абзац крупнее, вы создадите сообщение, намного лучше концентрирующее внимание. Самый простой способ стилизовать один-единственный абзац — создать стилевой класс и применить его к этому абзацу.

3. Добавьте последний стиль к концу таблицы стилей:

```
.intro {
  color: #6A94CC;
  font-family: Arial, Helvetica, sans-serif;
  font-size: 1.2em;
  margin-left: 0;
  margin-bottom: 25px;
}
```

   Этот стиль изменяет цвет, шрифт и размер, а также немного регулирует отступы. Все, что вы должны сделать, — применить класс к HTML-коду.

4. Откройте файл cascade.html в редакторе веб-страниц. Найдите тег <p>, который идет после тега <h1>CSS: The Missing Manual</h1> прямо под , и затем добавьте следующий атрибут класса:

```
<p class="intro">
```

5. Просмотрите страницу в браузере.

   Абзац никак не изменился. Согласно правилам каскадности, `.intro` — базовый селектор класса, а `#main p` является наследуемым селектором и состоит из идентификатора и имени тега. Они добавлены для создания более специфичного стиля, поэтому его свойства стиля решают любой конфликт между ним и стилем `.intro`.

   Для того чтобы заработал стиль `.intro`, необходимо немного «укрепить» его, наделив его селектор большей значимостью.

6. Вернемся к файлу `main.css` в редакторе и изменим название стиля с `.intro` на `#main .intro`.

   Теперь у вас есть наследуемый селектор, составленный из идентификатора и класса. Этот стиль является более специфичным, чем `#main p`, и его свойства замещают аналогичные из более общего стиля.

7. Просмотрите страницу в браузере (рис. 5.8).

   Вуаля! Абзац изменил свой цвет на голубой, его текст стал крупнее, шрифт поменялся, и отсутствует левый отступ. Если бы у вас не было четкого понимания каскадности, вы бы ломали голову над тем, почему этот стилевой класс не работал в первый раз.

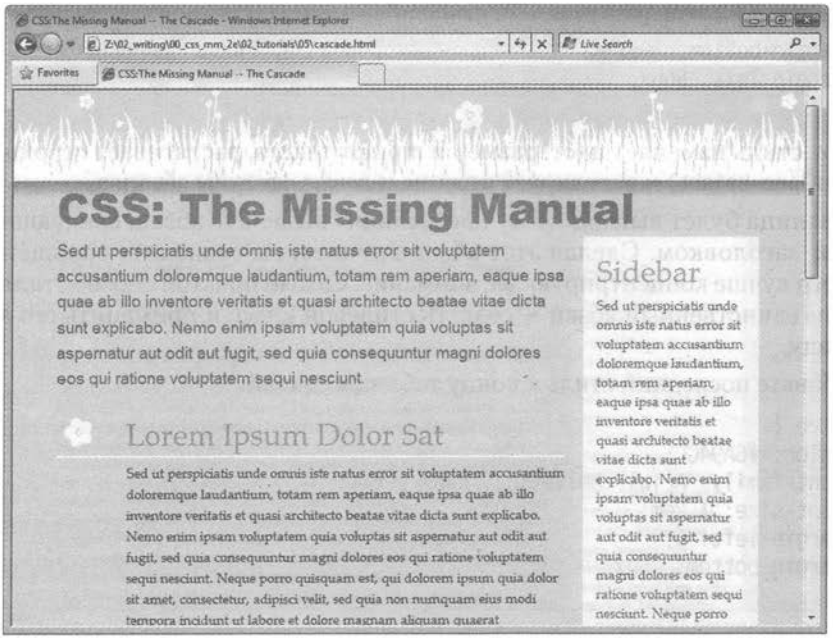


Рис. 5.8. Даже в простой странице с небольшим набором стилей внешний вид тегов зачастую является комбинацией свойств различных стилей

В этой и четырех предыдущих главах мы рассмотрели основы CSS. Теперь, в части 2, пришло время применить полученные знания для создания настоящих полноценных дизайнов.

ЧАСТЬ 2

Применение CSS

Глава 6. Форматирование текста
Глава 7. Поля, отступы, границы
Глава 8. Добавление графики на веб-страницы
Глава 9. Приводим в порядок навигацию сайта
Глава 10. Форматирование таблиц и форм

6 Форматирование текста

Большинство сайтов в Интернете по-прежнему построены исключительно на текстовом содержимом. Несомненно, людям нравится видеть фотографии, видеоклипы, Flash-анимацию, но все-таки именно содержательный текстовый материал заставляет их возвращаться на определенные сайты вновь и вновь. Люди жаждут обновлений в Facebook, новостей, статей с практическими рекомендациями, рецептов, ответов на актуальные вопросы, полезной информации и новых записей в микроблогах Twitter. С помощью языка CSS вы можете придать такой вид заголовкам и текстовому содержимому веб-страниц, что они привлекут внимание посетителей не хуже других фотографий.

CSS предлагает для этих целей обширный набор мощных команд форматирования, которые позволяют назначать шрифты, цвет, размеры, межстрочный интервал и другие свойства и атрибуты, которые оказывают влияние на визуальное восприятие как отдельных элементов веб-страницы (заголовков, маркированных списков, обычных абзацев текста), так и всей веб-страницы, сайта в целом (рис. 6.1). Настоящая глава описывает и показывает все многообразие свойств форматирования текстового содержимого веб-страниц и заканчивается обучающим уроком, позволяющим попрактиковаться в создании таблиц стилей CSS для текста и применении их к реальным веб-страницам.

Стилизация текста

Первое, что вы можете сделать для большей привлекательности сайта, — применить различные шрифты к заголовкам, абзацам и другим элементам. Для выбора начертания в CSS-стиле используется свойство `font-family:`.

```
font-family: Arial;
```

ПРИМЕЧАНИЕ

На практике для применения определенного CSS-свойства вы должны добавить все остальные обязательные детали блока описания стиля и таблицы (см. гл. 2). Например, p { font-family: Arial; }. Если есть такие примеры определений, помните, что это всего лишь отдельное свойство CSS-стиля для сокращения и упрощения написания книги и удобства чтения.

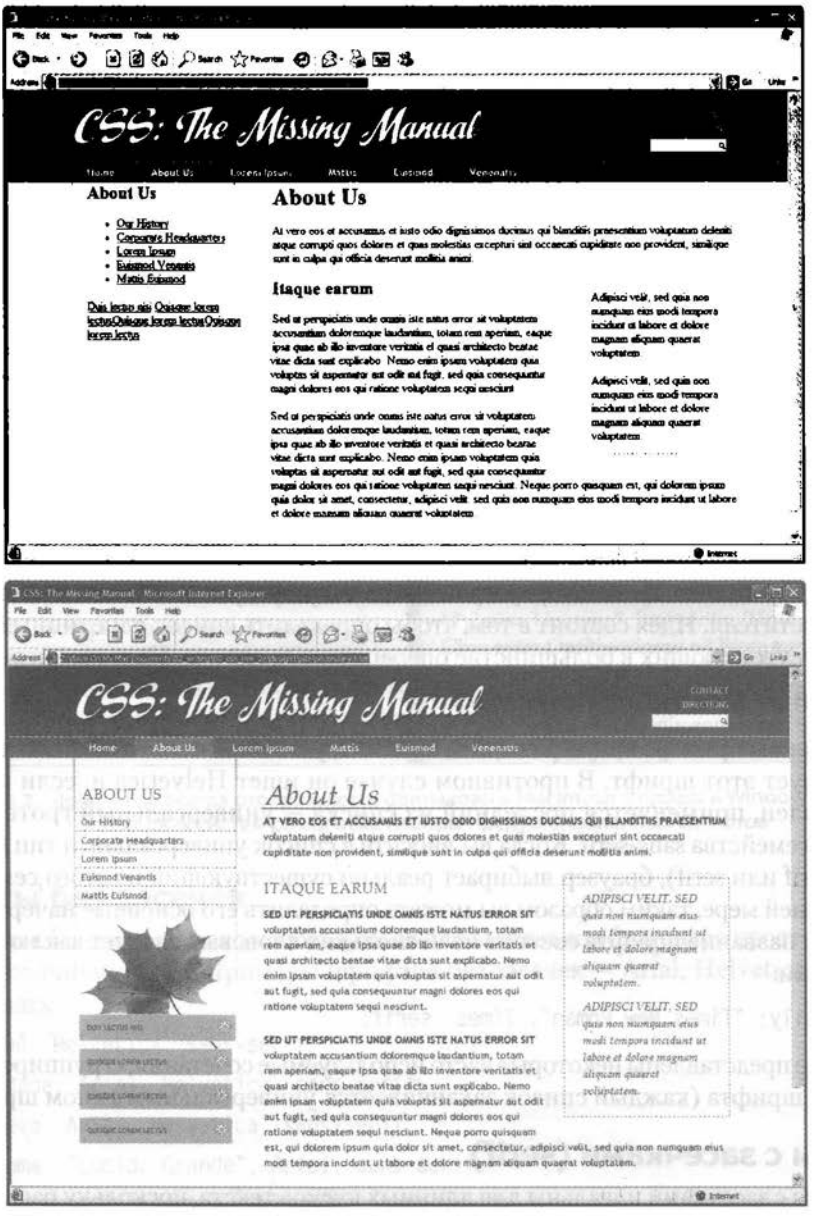


Рис. 6.1. Зачем соглашаться со скучным и заурядным оформлением текста веб-страниц *(вверху)*, если можно с легкостью сделать и заголовки, и текст выразительными и притягательными благодаря нескольким простым свойствам CSS *(внизу)*

Выбор шрифта

Рекомендуется выбирать шрифты, которые делают текст привлекательным для глаза (особенно если это заголовок) и читабельным (если это основной текст

содержимого веб-страницы). К сожалению, вы не сможете воспользоваться абсолютно любым шрифтом. Вернее, фактически *можно* использовать все, что хотите, но текст веб-страницы не будет отображен на экране компьютера в браузере посетителя, пока он не установит в системе такой же шрифт. По этой причине такое написание будет бесполезно для применения. В противном случае браузеры посетителей вашего сайта отобразят текст шрифтом по умолчанию (предопределенным). Обычно для текста используется семейство Times.

СОВЕТ

Если вы все-таки хотите отобразить текст произвольным шрифтом, для этого есть метод Cufon (http://wiki.github.com/sorccu/cufon/about). Этот метод, основанный на использовании JavaScript, позволяет преобразовывать один из ваших шрифтов в файл, который затем можно использовать для замещения текста HTML. Хоть сама технология работы Cufon довольно сложная, применить ее не составит большого труда.

Одно из решений данной проблемы заключается в определении необходимого шрифта, а также нескольких резервных на случай его отсутствия. Если на компьютере посетителя сайта установлен первый определенный вами предпочитаемый шрифт, то браузер его использует. Если же нет, то список будет просматриваться по порядку до тех пор, пока браузер не найдет шрифт, установленный на компьютере посетителя. Идея состоит в том, чтобы определить список однотипных шрифтов, присутствующих в большинстве операционных систем. Например:

```
font-family: Arial, Helvetica, sans-serif;
```

В данном примере браузер сначала выяснит, установлен ли Arial. Если да, то он использует этот шрифт. В противном случае он ищет Helvetica и, если и тот не установлен, применяется последний из списка — универсальный гротесковый шрифт семейства sans-serif. Когда вы вносите в список универсальный тип шрифта (sans-serif или serif), браузер выбирает реально существующий из этого семейства. По крайней мере, таким образом вы можете определить его основные начертания.

Если название шрифта состоит из нескольких слов, вам следует заключать его в кавычки:

```
font-family: "Times New Roman", Times, serif;
```

Ниже представлены некоторые часто используемые сочетания, сгруппированные по типу шрифта (каждый список заканчивается универсальным типом шрифта).

Шрифты с засечками (serif)

Шрифты с засечками идеальны для длинных кусков текста, поскольку распространено мнение, что засечки — маленькие росчерки на концах основных штрихов — хорошо для глаз связывают одну букву с другой, делая текст более читабельным. Примерами шрифтов с засечками являются Times, Times New Roman, Georgia:

- `"Times New Roman", Times, serif;`
- `Georgia, "Times New Roman", Times, serif;`
- `Baskerville, "Palatino Linotype", Times, serif;`
- `"Hoefler Text", Garamond, Times, serif.`

Примеры данных шрифтов приведены на рис. 6.2.

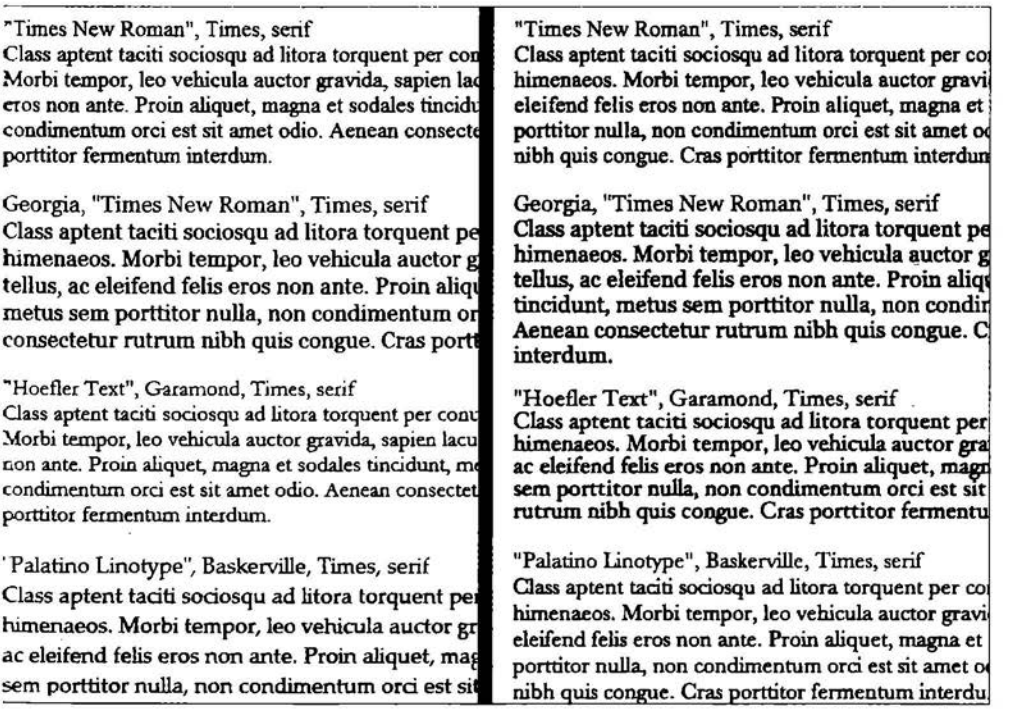


Рис. 6.2. Шрифты не всегда отображаются одинаково в Macintosh *(справа)* и Windows *(слева)*. У этих двух систем различные наборы предустановленных шрифтов

Шрифты без засечек

Шрифты без засечек часто используются для заголовков благодаря простому и четкому внешнему виду. Примеры шрифтов без засечек — Arial, Helvetica, Verdana и Formata:

- `Arial, Helvetica, sans-serif;`
- `Verdana, Arial, Helvetica, sans-serif;`
- `Geneva, Arial, Helvetica, sans-serif;`
- `Tahoma, "Lucida Grande", Arial, sans-serif;`
- `"Trebuchet MS", Arial, Helvetica, sans-serif;`
- `"Century Gothic", "Gill Sans", Arial, sans-serif.`

Примеры данных шрифтов приведены на рис. 6.3.

Некоторые люди верят в то, что на веб-страницах стоит использовать лишь шрифты без засечек, потому что декоративные штрихи шрифтов с засечками отображаются не очень хорошо на экранах с низким разрешением. В конце концов, ваш вкус — лучший ориентир. Выбирайте те типы шрифтов, которые считаете нужными.

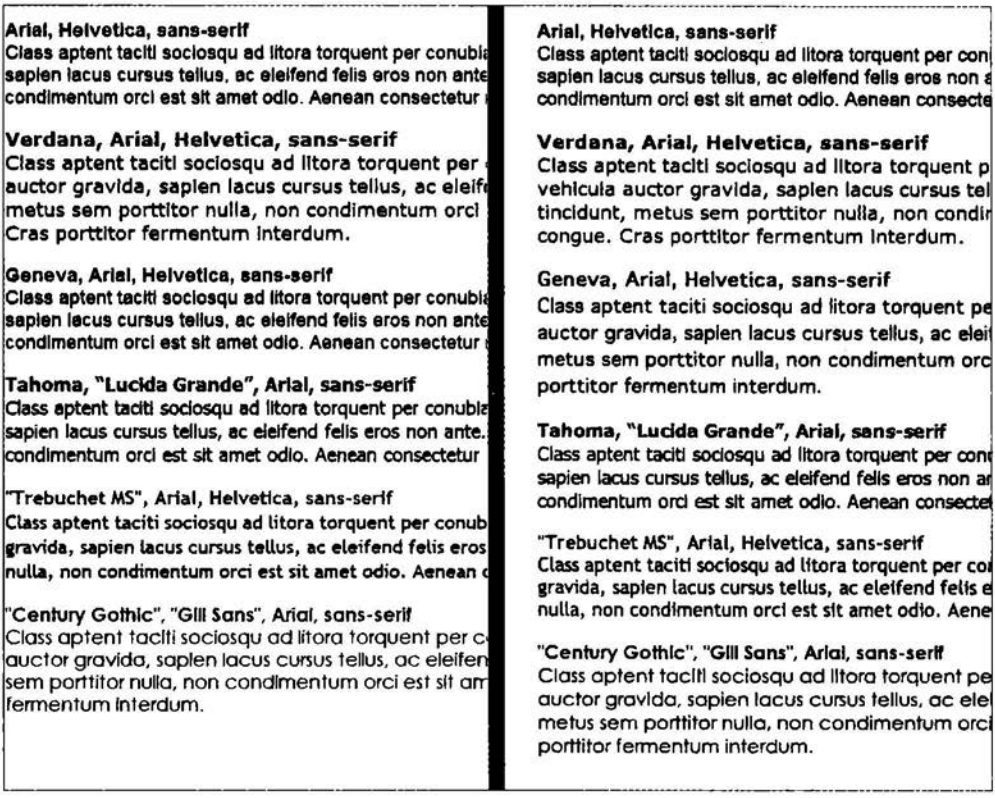


Рис. 6.3. Текст в Windows *(слева)* и Macintosh *(справа)*

Моноширинные шрифты

Моноширинный шрифт часто используется для отображения компьютерного кода (например, фрагментов кода повсюду в этой книге). Каждая буква в моноширинном шрифте имеет одинаковую ширину (они традиционно использовались в печатных машинках, для того чтобы выравнивать данные в аккуратные колонки).

- `"Courier New", Courier, monospace.`
- `"Lucida Console", Monaco, monospace.`
- `"Copperplate Light", "Copperplate Gothic Light", serif.`
- `"Marker Felt", "Comic Sans MS", fantasy.`

С примерами этих списков шрифтов можете ознакомиться на рис. 6.4.

Дополнительные шрифты

На самом деле существуют буквально тысячи шрифтов, и каждая операционная система поставляется со многими из тех шрифтов, которые не перечислены здесь. Тем не менее приведу несколько шрифтов, которые очень часто встречаются на



Рис. 6.4. Текст в Windows *(слева)* и Macintosh *(справа)*. Courier самый распространенный моноширинный шрифт, но ограничиваться только им не обязательно. Lucida Console очень популярен в Windows и Macintosh, а Monaco установлен на каждом компьютере Macintosh

персональных компьютерах и компьютерах Macintosh. Возможно, вы захотите выбрать какие-то из них:

- Arial Black;
- Arial Narrow;
- Impact.

Будьте осторожны с Arial Black и Impact: у них есть только одна плотность и они не поддерживают курсивный вариант. Соответственно, если вы используете эти шрифты, не забудьте установить для `font-weight` и `font-style` значение `normal`. В противном случае, если шрифт будет полужирным или курсивным, браузер сделает все от него зависящее, чтобы текст выглядел ужасно.

СОВЕТ

Вы найдете намного больше информации о том, какие шрифты в каких системах установлены (в том числе Macintosh, Windows и Linux), на сайте www.codestyle.org/css/font-family. Еще один хороший ресурс, позволяющий выйти за рамки стандартного набора веб-шрифтов, расположен по адресу http://unitinteractive.com/blog/2008/06/26/better-css-font-stacks.

Придание тексту цветового оформления

Черно-белые цвета хорошо смотрятся в фильмах Вуди Аллена, но когда мы имеем дело с текстом, шрифт небесно-синего цвета будет выглядеть намного более четко, изящно и стильно по сравнению с простым черным. Окрашивание текста веб-страниц средствами CSS не представляет трудностей. Фактически мы уже использовали

свойство color в предыдущих обучающих уроках. Существует несколько способов определения конкретного цвета, но все они базируются на одном принципе: нужно указать само свойство color и затем его значение:

```
color: #3E8988;
```

В этом примере значение цвета представлено шестнадцатеричным числом, определяющим слабый оттенок зеленовато-голубого цвета (о шестнадцатеричных числах читайте далее).

СОВЕТ

Если вам требуется помощь по цветовому оформлению, можете найти множество согласованных коллекций цветов, а также полезных, связанных с ними ресурсов в Интернете по адресу www.colourlovers.com.

Шестнадцатеричное представление цвета

Чаще всего дизайнеры используют кодирование цвета элементов веб-страниц в шестнадцатеричной системе счисления. Значение, например #6600FF, фактически содержит три шестнадцатеричных числа. В данном примере это 66, 00 и FF. Каждое число определяет уровень красного, зеленого и синего цветов соответственно (как и в цветовой системе RGB, описываемой далее). Окончательное значение цвета получается при смешивании этих трех составляющих.

СОВЕТ

Можете сокращать шестнадцатеричные числа до трех символов, если каждое из трех двухзначных чисел содержит по два одинаковых символа. Например, можно привести #6600FF к виду #60F или #FFFFFF к #FFF.

RGB

Можете также пользоваться методом кодирования RGB (red — красный, green — зеленый, blue — синий), с которым вы, возможно, знакомы из программ компьютерной графики. Значение цвета кодируется тремя числами, представляющими процентные соотношения (0–100 %) или числа в диапазоне 0–255 для формирования каждого оттенка (красный, зеленый, синий). Если вы хотите установить белый цвет текста (возможно, чтобы контрастно выделить его на темном фоне веб-страницы), можно использовать следующее свойство.

```
color: rgb(100%,100%,100%);
```

или

```
color: rgb(255,255,255);
```

СОВЕТ

Вы всегда можете вернуться к классическим цветам HTML. Однако это будет минус в новизне и оригинальности вашего сайта. Существует 17 ключевых определений цветов: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, orange, purple, red, silver, teal, white, yellow. В CSS их добавляют в стили следующим образом: color: fuchsia; и пр.

Установка размера шрифта

Установка определеннного размера шрифта текста веб-страницы — хороший способ визуально придать тексту значимость и привлечь внимание посетителей к наиболее важным фрагментам страницы. Заголовки, отображенные шрифтом большего размера, привлекают внимание. В то же время информация, отображенная маленьким, возможно, подстрочным шрифтом, позволяет этому блоку не выделяться на фоне общего текста веб-страницы, создавая ненавязчивый комментарий.

Свойство `font-size` устанавливает размер шрифта текста. За значением всегда должна следовать единица измерения величины. Например, так:

```
font-size: 1em;
```

Сочетание значения свойства и единицы измерения, которые вы указываете для установки размера шрифта (в данном примере `1em`), определяют размер текста. CSS предлагает большой выбор единиц измерения: предопределенные ключевые слова, em (стандартная единица измерения в типографской системе: размер буквы M, напечатанной шрифтом Cicero), exs (то же самое, только берется размер буквы x), пикселы, проценты, пики, пункты, дюймы, сантиметры и миллиметры.

Единицы измерения, обычно используемые в печати (пики, пункты, дюймы и т. д.), не очень удобно применять к веб-страницам, так как невозможно предугадать их вид на разных мониторах. Однако пункты удобно использовать при создании таблиц стилей для веб-страниц, ориентированных для печати на принтере. Только небольшую часть всех существующих единиц измерения — пикселы, ключевые слова, em, проценты — можно использовать для определения размеров шрифтов текста веб-страниц, отображаемых браузером на экране монитора. В следующей части книги рассказывается, как они работают.

Пикселы

Переменные для значений пикселов являются самыми легкими для понимания, поскольку не зависят от параметров настроек браузера. Когда вы определяете, например, размер шрифта равным 36 пикселам для заголовка `<h1>`, браузер отображает текст высотой 36 пикселов. Дизайнеры выбирают эти единицы измерения потому, что они обеспечивают постоянные совместимые параметры размеров текста на различных типах компьютеров и браузеров (см. врезку «Часто задаваемые вопросы» ниже, где описано ограничение для размеров, устанавливаемых в пикселах).

Чтобы установить для свойства `font-size` значение в пикселах, введите число с сокращением данной единицы измерения `px`:

```
font-size: 36px;
```

ПРИМЕЧАНИЕ

Не добавляйте пробел между значением свойства и единицей измерения. Например, правильно 36px, но не 36 px.

ЧАСТО ЗАДАВАЕМЫЕ ВОПРОСЫ

Проблема с пикселами

Кажется, что пиксела вполне достаточно для конкретизации значений свойств и обеспечения полной свободы действий в установке размеров шрифтов текста веб-страниц. Так зачем все-таки нужно использовать другие единицы измерения, определяя параметры текста?

К сожалению, в браузере Internet Explorer 6 (и его более ранних версиях) имеет место одно серьезное ограничение, проявляющееся при установке параметров текста в пикселах: программа просмотра не может управлять размерами шрифтов. Некоторые люди, особенно с плохим зрением, пользуются возможностью браузера изменять размеры шрифтов текста при просмотре веб-страниц командой View ▸ Text Size (Вид ▸ Размер шрифта), что облегчает его восприятие. Однако Internet Explorer не будет ничего изменять, если для него в качестве единиц измерения установлены пикселы. Браузер придерживается пожеланий дизайнера веб-страницы: оставляет внешний вид текста при просмотре неизменным.

Использовать в качестве единиц измерения пикселы или нет — актуальный вопрос веб-дизайнеров. Многие разработчики полагают, что, если размер шрифтов текста устанавливается в пикселах, это создает проблемы при просмотре: препятствует доступу к сайту людей с ограниченными физическими возможностями.

Версии Internet Explorer, начиная с седьмой, как и большинство других известных браузеров, реально позволяют изменять размеры текста, определенного в пикселах. Эти браузеры следуют тенденции масштабирования страниц, и вместо простого увеличения текста вы можете масштабировать страницу, увеличивая ее всю, включая изображения, текст и т. д. Поскольку Internet Explorer 8 становится все более популярным, в нем вы сможете использовать на веб-страницах CSS-свойства со значениями в пикселах, не заботясь об ограничениях, присутствующих в Internet Explorer 6.

Я рекомендую прислушиваться к мнению целевой аудитории. Если вы создаете сайт, ориентированный на старших людей или школьников, то применяйте для текстовых данных одну из следующих единиц измерения: ключевые слова, em, процентные значения. Как бы то ни было, посетителями ваших веб-страниц могут оказаться люди с ограниченными физическими возможностями, или, вероятно, они просто будут просматривать сайт на компьютерах с устаревшим программным обеспечением.

Ключевые слова, проценты и единица измерения em

Все нижеперечисленные способы установки размеров текста средствами CSS с помощью ключевых слов, процентных значений и единицы em увеличивают или уменьшают размеры шрифтов текста, отображаемого в окне браузера в соответствии с определенными правилами. Другими словами, если вы не определите размер средствами CSS, браузер применит к тексту свои предопределенные параметры. В большинстве случаев обычный текст, находящийся вне тегов заголовков, отобразится высотой 16 пикселов — это *основной (базовый) размер шрифта текста.*

Серверы могут корректировать настройки браузеров, увеличивая или уменьшая базовый размер шрифта. В Internet Explorer, например, чтобы установить требуемый параметр, нужно выбрать пункт меню View ▸ Text Size ▸ Larger или Smaller (Вид ▸ Размер шрифта ▸ Крупный или Мелкий); в Firefox 2 это будет пункт меню View ▸ Text Size ▸ Increase или Decrease (Вид ▸ Размер шрифта ▸ Увеличить или Уменьшить), в Firefox 3 — View ▸ Zoom (Вид ▸ Масштаб), а в Safari — View ▸ Make Text

Smaller (Вид ▸ Сделать текст меньше) или View ▸ Make Text Bigger (Вид ▸ Сделать текст больше).

Когда вы изменяете текст с помощью CSS, браузер берет базовый размер шрифта текста (будь то первоначальный высотой 16 пикселов или другой) и корректирует его в большую или меньшую сторону в соответствии со значением ключевого слова, единицей измерения em, процентным отношением.

Ключевые слова

CSS предлагает семь ключевых слов, которые позволяют назначить размер шрифта относительно базового: `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large` и `xx-large`. CSS-код будет выглядеть следующим образом:

```
font-size: large;
```

Среднее значение `medium` — базовый размер шрифта браузера. Остальные значения уменьшают или увеличивают размер шрифта с различными коэффициентами. Другими словами, несмотря на то, что, казалось бы, каждое изменение размера должно быть последовательным увеличением или уменьшением предыдущего значения, это не так. Обычно `xx-small` является эквивалентом 9 пикселов (принимая во внимание, что вы не изменяли базовый размер шрифта в своем браузере), `x-small` соответствует 10 пикселам, `small` — 13, `large` — 18, `x-large` — 24, а `xx-large` — 32 пикселам.

Как вы заметили, имеется ограниченный набор ключевых слов — всего семь вариантов. Если требуется бóльшая свобода действий по масштабированию, надо обратиться к другим средствам и единицам измерения, описываемым ниже.

Процентные значения

Подобно ключевым словам, процентные значения изменяют размер текста относительно шрифта, определенного браузером. Они обеспечивают более гибкое и точное управление, чем ключевые `large`, `x-large` и т. д. Любой браузер имеет предопределенный базовый размер шрифта. У большинства он составляет 16 пикселов. Можно подкорректировать этот размер с помощью настроек браузера. Независимо от выбора установки, основной размер шрифта эквивалентен 100 %. Другими словами, это равнозначно установке шрифта высотой 16 пикселов.

Допустим, вы хотите отобразить определенный заголовок в два раза больше, чем средний шрифт веб-страницы. Просто установите размер, равный 200 %:

```
font-size: 200%;
```

Или, если хотите, чтобы шрифт был немного меньше по умолчанию, укажите значение 90 %.

Приведенные примеры являются самыми простыми, но на практике встречаются более сложные ситуации. Например, относительный размер 100 % может изменяться, если наследуется значение тега-предка.

На веб-странице в левой нижней части рис. 6.5 есть текст в теге `<div>`, шрифт которого установлен размером 200 %. Это в два раза больше базового размера и составляет 32 пиксела. Все вложенные теги наследуют его и используют для вычисления своих размеров шрифтов. Другими словами, для тегов, вложенных в `<div>`, размер 100 % равен 32 пикселам. Так, текст заголовка `<h1>`, находящегося внутри

тега , составляет 100 % и отображается в два раза больше базового для этой веб-страницы, то есть высотой 32 пиксела.

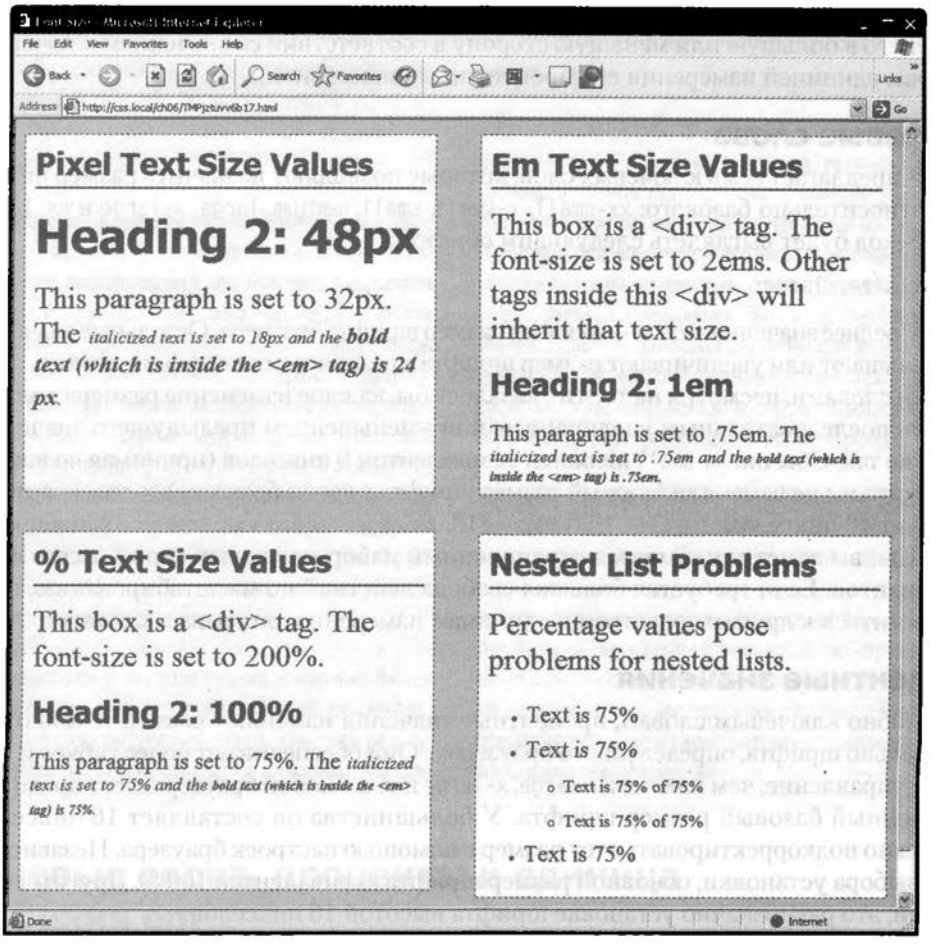


Рис. 6.5. Тремя самыми распространенными и наиболее часто применяемыми единицами измерения для установки размеров шрифтов являются пикселы, em и проценты

Будьте внимательны при использовании em и процентных значений, когда имеют место наследуемые свойства размеров шрифтов. Если вы заметили, что какой-то текст выглядит не так, как предусмотрено, необычно большим или маленьким, убедитесь в том, что он не наследует параметры.

Единица измерения em

Если вы поняли, как работают процентные отношения, то легко поймете и единицу em. Они функционируют практически одинаково, но большинство веб-дизайнеров все же используют эту единицу измерения из-за применения ее в книгопечатании.

Слово «em» позаимствовано из типографского дела. Оно имеет отношение к размеру заглавной буквы M определенного шрифта. В мире веб-дизайна это слово приобрело собственное значение. В CSS понятие относится к базовому размеру шрифта текста. Иными словами, значение размера шрифта `1em` означает то же самое, что и 100 %, как описано в предыдущем разделе. Иначе говоря, процентное значение эквивалентно em, умноженному на 100: `.5em` — 50 % и т. д.

Например, этот CSS-код вызывает тот же эффект, что и `font-size: 200%;`.

```
font-size: 2em;
```

ПРИМЕЧАНИЕ

Как и при использовании пикселов, вы не должны указывать пробел между числом и единицей измерения em. Кроме того, даже если значение свойства больше единицы, совсем не нужно добавлять s в конце: правильно 2.5em, но не 2.5ems.

Принцип работы механизма наследования с em точно такой же, как и с процентами. Взгляните, например, на верхнюю правую часть рис. 6.5. Шрифт нижнего абзаца определен равным `.75em`, а поскольку тег абзаца `<p>` наследует размер шрифта `2em` (32 пиксела) от `<div>`, в результате получается размер шрифта: .75 × 32 = 24 пиксела. Внутри `<p>` есть два других тега с размером шрифта `.75em`. Размер шрифта тега наиболее глубокой вложенности `<strong>` установлен равным `.75em`, или, по сути, 75 % от унаследованного. Довольно сложный расчет: 32 пиксела (размер, унаследованный от `<div>`) × 75 (размер от `<p>`) × .75 (размер `<em>`) × 75 (собственный размер `<strong>`). В результате вычислений получается размер, равный приблизительно 14 пикселам.

ПРИМЕЧАНИЕ

Браузер Internet Explorer 6 и его более ранние версии иногда некорректно отображают текст вебстраницы при использовании исключительно единиц измерения em. Есть два способа решения этой проблемы: применять процентные значения или установить основной размер шрифта для `<body>` в процентах, а затем использовать em для каких-либо изменений. Это помогает устранить ошибку Internet Explorer 6.

ОБХОДНОЙ ПРИЕМ

Разбор вложенных конструкций

Унаследованные значения размеров шрифтов могут вызвать проблемы для вложенных (многоуровневых) списков (см. рис. 6.5). Например, у вас имеется стиль `ul { font-size: 75% }`, а затем вы создаете вложенный список, то есть внутри `<ul>` расположены другие теги. Получается, что для вложенного `<ul>` должен быть установлен размер шрифта, равный 75 % от внешнего `<ul>`. Следовательно, подпункты списков будут отображены меньшим шрифтом, чем пункты и т. д. применительно к подпунктам следующего подуровня.

Чтобы обойти эту проблему, создайте дополнительный стиль с селектором потомков (см. разд. «Стилизация вложенных тегов» гл. 3): `ul ul {font-size: 100%}`. Этот стиль устанавливает размер шрифта любого `<ul>`, вложенного в другой `<ul>`, равным 100 %. Другими словами, 100 % от размера шрифта родительского элемента `ul`, в который вложен другой элемент. В данном примере это обеспечивает сохранение размера шрифта вложенных подпунктов списков равным 75 %.

СОВЕТ

Текст веб-страницы выделяют самыми различными способами. Этого можно добиться посредством изменения размера шрифта определенного фрагмента текста, отдельных слов или с помощью контрастности. Окрашивание текста в темный, светлый или более яркий оттенок визуально выделяет его. Применение контрастности — одно из наиболее важных правил хорошего графического дизайна. Контрастность может помочь выделить важные сообщения, фрагменты текста, привлечь внимание. Краткий обзор особенностей типографского контраста описан в Интернете по адресу: http://www.creativepro.com/story/feature/19877.html.

Форматирование символов и слов

Вам потребуется немало времени для точной и тонкой настройки параметров текста: цвета, размеров шрифтов и т. д. Однако CSS предоставляет также множество других средств для форматирования текста. Из самых распространенных свойств можно выделить, например, полужирное и курсивное начертания, а из менее широко используемых — создание малых прописных букв, буквиц, изменение межсимвольного интервала и т. д.

СОВЕТ

Средства языка CSS позволяют комбинировать множество свойств форматирования текста вместе. Слишком «тяжеловесное» оформление делает страницу сложной для чтения и понимания. Хуже всего, если из-за такой стилизации остается негативное впечатление от посещения сайта.

Курсив и полужирный шрифт

Браузеры отображают текст, заключенный внутри тегов и <i>, *курсивом* (шрифтом с наклонным начертанием), а текст, находящийся в тегах , <b>, <th> (table header — «заголовок таблицы»), тегах заголовков (<h1>, <h2> и т. д.), — **полужирным**. Вы можете управлять этими параметрами. Если хотите, то можете отменить полужирное начертание для заголовков или выделить курсивом обычный текст, используя свойства `font-style` и `font-weight`.

Чтобы выделить текст курсивом, добавьте к стилю следующий код:

```
font-style: italic;
```

Или, наоборот, можете установить для текста обычный, не курсивый шрифт:

```
font-style: normal;
```

ПРИМЕЧАНИЕ

Свойство стиля шрифта font-style на самом деле содержит третью команду — oblique, которая создает тот же эффект, что italic.

Свойство толщины шрифта `font-weight` позволяет делать текст полужирным или обычным. Фактически, согласно правилам CSS, можно указать девять числовых значений (100–900) для определения точных, едва различимых градаций плотности (от «суперплотного» (900) до «крайне легкого, почти невидимого» (100)) шрифта. Естественно, чтобы эти команды работали, сами используемые шрифты должны

иметь для них девять различных значений толщины, обеспечивая тем самым заметный визуальный эффект для посетителей сайта. Но не волнуйтесь — на сегодняшний день не существует шрифтов, которые бы работали вместе с браузером подобным образом. Реально применимых команд для этого свойства гораздо меньше.

Чтобы сделать шрифт текста полужирным, наберите:

```
font-weight: bold;
```

Обычный шрифт устанавливается следующим образом:

```
font-weight: normal;
```

СОВЕТ

Поскольку для заголовков уже предопределен стиль с полужирным начертанием, возможно, потребуется другой способ для выделения, которое вы хотите придать некоторым частям текста внутри заголовка. Вот один из способов:

```
h1, strong {color: #3399FF;}
```

Этот стиль с производным селектором меняет цвет всех элементов (обычно отображаемых полужирным шрифтом), заключенных внутри <h1>.

Прописные буквы

Набрать текст прописными буквами очень просто: нажмите клавишу фиксации регистра заглавных букв Caps Lock и вводите текст. Что же делать, если нужно, чтобы прописными буквами были написаны все уже набранные заголовки веб-страницы или текст, который скопировали и вставили из Microsoft Word? Вместо того чтобы повторно вводить заголовки, обратитесь к CSS-свойству `text-transform`. С его помощью можно преобразовать любой текст в верхний или нижний регистры или даже превратить первые буквы каждого слова в прописные (для названий и заголовков). Вот пример преобразования в прописные буквы:

```
text-transform: uppercase;
```

Сделать буквы строчными можно с помощью значения `lowercase`, а превратить первые буквы слов в прописные — с помощью `capitalize`.

Поскольку это свойство наследуемо, любые теги, вложенные в `text-transform`, приобретают то же форматирование: верхний регистр, нижний, первые прописные буквы слов. Чтобы указать CSS *не* изменять регистр текста, используйте значение свойства `none`:

```
text-transform: none;
```

Малые прописные буквы. Для придания тексту типографской изысканности можно также использовать свойство `font-variant`, которое позволяет преобразовать текст таким образом, что все буквы будут малыми прописными. В этом стиле строчные буквы выглядят как немного уменьшенные заглавные. Большие фрагменты такого текста становятся трудночитаемыми, но этот стиль придает веб-странице «старосветскую», книжную многозначительность. Для создания стиля с малыми прописными буквами используйте следующий код:

```
font-variant: small-caps;
```

Украшение текста

CSS предлагает также свойство `text-decoration` для улучшения внешнего вида текста путем добавления различных дополнительных элементов. С его помощью можно добавить линии подчеркивания, надчеркивания, перечеркнуть текст (рис. 6.6) или сделать его мигающим. Свойство `text-decoration` применяется в сочетании со следующими ключевыми словами: `underline`, `overline`, `line-through` или `blink`. Например, чтобы подчеркнуть фрагмент, наберите код:

```
text-decoration: underline;
```

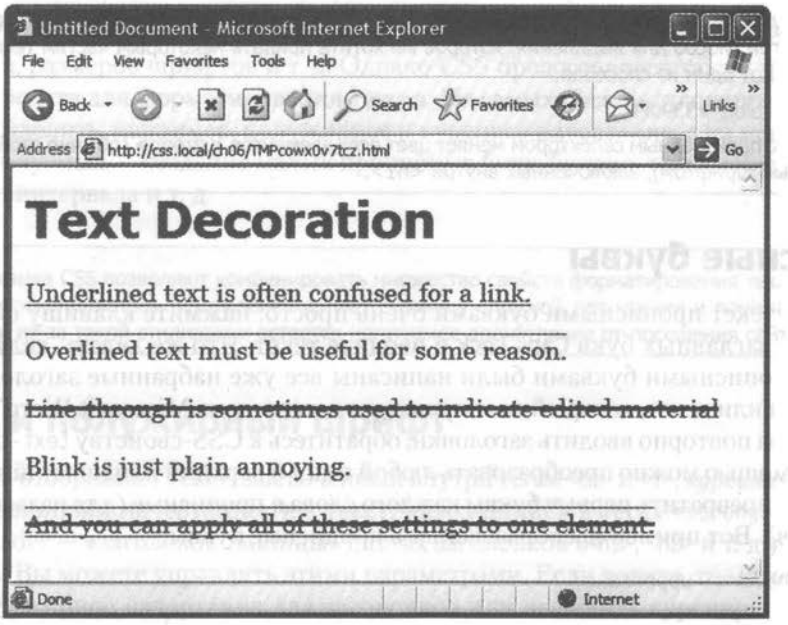


Рис. 6.6. Здесь показан результат применения свойства text-decoration

Вы можете также объединять несколько ключевых слов вместе для комбинирования эффектов. Добавить к тексту линии подчеркивания и надчеркивания одновременно можно следующим способом:

```
text-decoration: underline overline;
```

Однако не стоит изощряться в таком «вычурном» форматировании только потому, что есть такая возможность. Во-первых, у всех, кто пользуется Интернетом на протяжении долгого времени, любой подчеркнутый текст инстинктивно ассоциируется с гиперссылкой, возникает желание непременно щелкнуть на ней кнопкой мыши. Таким образом, подчеркивать слова, не являющиеся частью гиперссылки, будет не очень хорошей идеей. Значение же мерцания `blink` на самом деле больше походит на неоновую вывеску типа «Для любителей!» (к тому же большинство браузеров не сделают текст мерцающим, даже если вы укажете им это).

СОВЕТ

Вы можете применить эффект, подобный подчеркиванию и надчеркиванию, если добавите к элементу — в данном случае к тексту — линию рамки border сверху или снизу (см. разд. «Добавление границ» гл. 7). Преимущество состоит в том, что с помощью свойства border вы можете управлять одновременно многими параметрами: размещением, размером, цветом линий-рамок — и в конечном счете придать тексту более привлекательный вид, не делая его похожим на гиперссылки.

Значение `overline` свойства `text-decoration` надчеркивает, а `line-through` — перечеркивает текст. Некоторые веб-дизайнеры применяют последний эффект, чтобы показать читателю, что фрагмент был удален из первоначального варианта документа. В сочетании с селектором `a:visited` можно создать красивый эффект.

Отменить все элементы украшений можно путем применения ключевого слова `none`:

```
text-decoration: none;
```

Зачем вам может понадобиться отмена декорирования? Самый распространенный пример — удаление стандартной предопределенной подчеркивающей линии гиперссылки (см. разд. «Стилизация ссылок» гл. 9).

Межсимвольный и межсловный интервал

Другой способ выделения текстового фрагмента состоит в регулировании интервала, с которым отображаются отдельные буквы или слова (рис. 6.7). Уменьшение межсимвольного интервала `letter-spacing` поможет сжать текст заголовков. Они будут казаться еще более плотными и тяжеловесными, а заодно вмещать больше текста на единственной строке заголовка. И наоборот, увеличение интервала может придать заголовкам более спокойный, величественный вид. Если хотите уменьшить расстояние между буквами, используйте отрицательные значения свойства:

```
letter-spacing: -1px;
```

Положительные значения свойства делают промежуток между буквами больше:

```
letter-spacing: 10px;
```

Аналогично можно изменить межсловный интервал, используя свойство `word-spacing`. Он делает промежуток между словами шире или уже, не затрагивая самих слов:

```
word-spacing: 2px;
```

С этими свойствами можно использовать любые единицы измерения, применимые к тексту: пикселы, em, проценты (с положительными или отрицательными значениями).

Если вы хотите, чтобы текст хорошо читался, применяйте небольшие значения интервалов. В противном случае буквы и слова будут перекрываться, накладываться друг на друга. Чтобы обеспечить представление содержимого сайта четким и понятным, аккуратно используйте межсимвольные и межсловные интервалы.

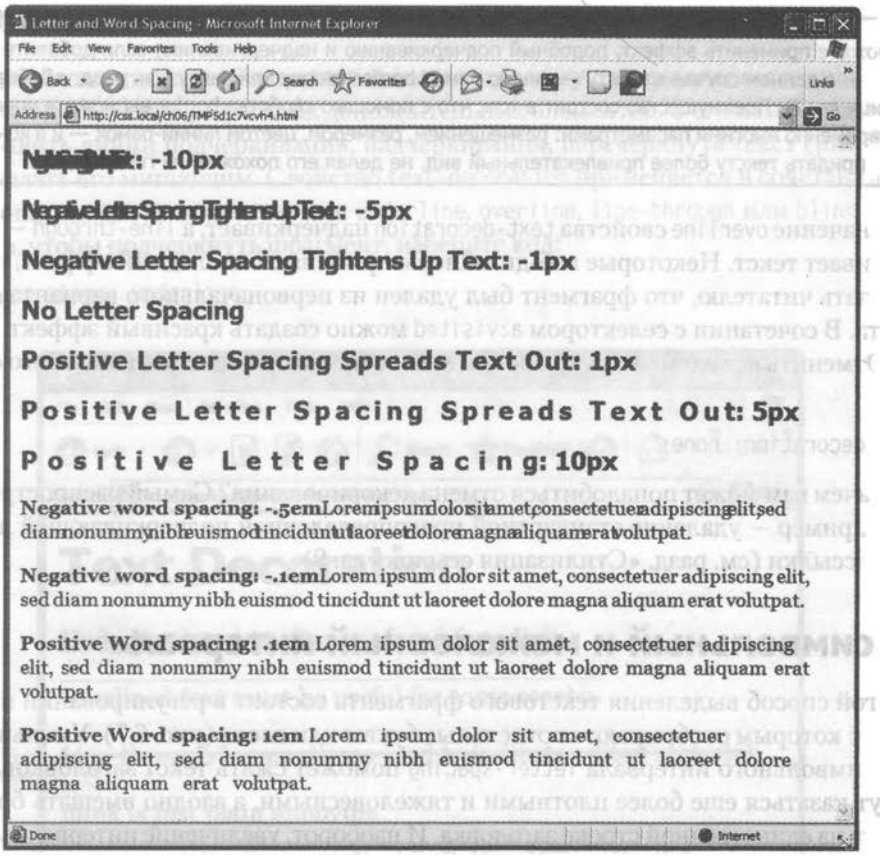


Рис. 6.7. Слишком большое или слишком маленькое значение межсимвольного или межсловного интервала может сделать текст трудночитаемым или вообще нечитаемым

Форматирование абзацев текста

В CSS есть свойства, которые используются для форматирования не отдельно взятых слов, а фрагментов, блоков текста. Иначе говоря, эти свойства можно применять к целым абзацам, заголовкам и т. д.

Установка межстрочного интервала

В дополнение к настройке интервала между словами и буквами (межсловный и межсимвольный интервалы) CSS позволяет устанавливать межстрочный интервал (интерлиньяж) — расстояние, промежуток между базовыми линиями двух соседних строк текста, — используя свойство `line-height`. Чем больше межстрочный интервал, тем больше промежуток между отдельными строками (рис. 6.8).

Стандартный межстрочный интервал эквивалентен 120 %. Таким образом, на рис. 6.8, *вверху*, мы видим, что меньшее процентное значение сжимает строки, а большее значение распределяет их более свободно (см. рис. 6.8, *внизу*).

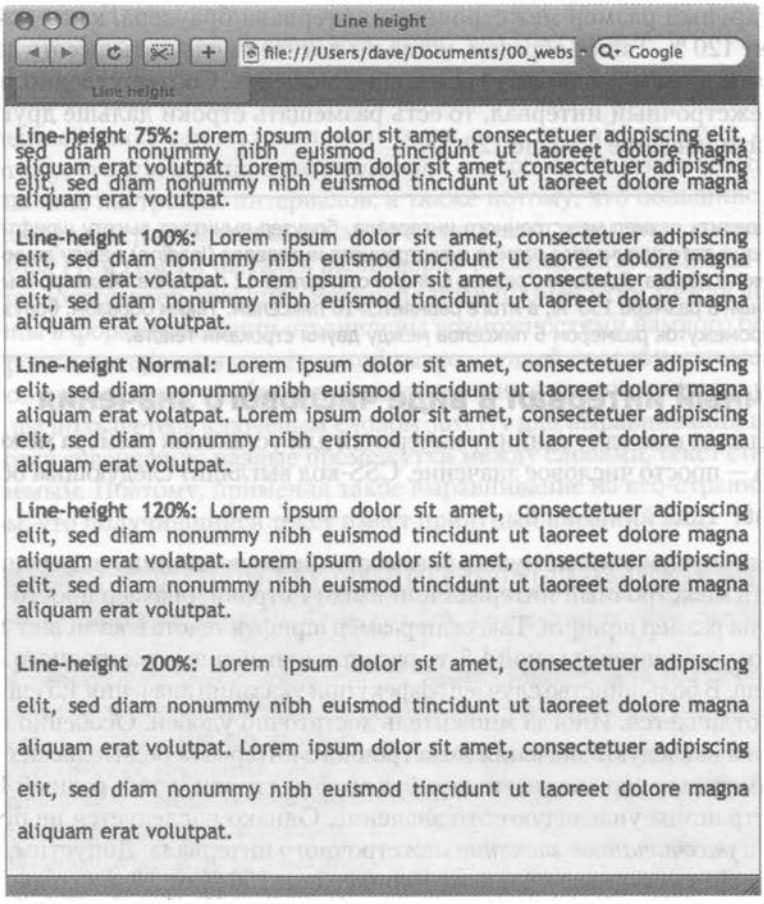


Рис. 6.8. Свойство установки межстрочного интервала line-height позволяет растянуть строки абзаца *(внизу)* или расположить их ближе друг к другу, с меньшим промежутком *(вверху)*

Межстрочный интервал в пикселах, em, процентах

Как и в свойстве размера шрифта `font-size`, вы можете использовать пикселы, em или проценты для установки размера межстрочного интервала с помощью свойства `line-height`:

```
line-height: 150%;
```

Вообще, процентные значения или em лучше, нежели пикселы: размер, установленный в этих единицах измерения, напрямую зависит от параметров шрифта и автоматически корректируется пропорционально изменению свойства `font-size`. Так, если вы установите межстрочный интервал равным 10 пикселам и затем измените на гораздо больший (например, на 36 пикселов), то высота останется равной 10, а строки будут накладываться друг на друга. Однако при использовании значения размера в процентах (скажем, 150 %) межстрочный интервал корректируется пропорционально всякий раз, когда вы изменяете значение размера шрифта соответствующего свойства `font-size`.

Стандартный размер межстрочного интервала браузера, как я уже сказал, составляет 120 %. Таким образом, когда вы хотите уменьшить высоту строк, промежуток между ними, используйте меньшее значение. Соответственно, чтобы увеличить межстрочный интервал, то есть размещать строки дальше друг от друга, используйте значение больше 120 %.

ПРИМЕЧАНИЕ

Чтобы определить размер межстрочного интервала, браузер вычитает высоту шрифта из высоты строки. В результате получается размер межстрочного интервала (leading) между двумя соседними строками текста абзаца. Допустим, размер шрифта составляет 12 пикселов. Межстрочный интервал, установленный в размере 150 %, в итоге равняется 18 пикселам. Таким образом, браузер добавляет пустой промежуток размером 6 пикселов между двумя строками текста.

Межстрочный интервал в виде числового значения

CSS предлагает еще одну единицу измерения для установки размера межстрочного интервала — просто числовое значение. CSS-код выглядит следующим образом:

```
line-height: 1.5;
```

После этого значения не нужно указывать никакую единицу измерения. Чтобы определить межстрочный интервал или высоту строки, браузер просто умножает это число на размер шрифта. Так, если размер шрифта текста составляет 1 em, а высота строки установлена равной 1,5, то расчетное значение межстрочного интервала равно 1,5 em. В большинстве случаев эффект при указании значения 1,5 em или 150 % ничем не отличается. Иногда множитель достаточно удобен. Особенно когда вложенные теги наследуют значения межстрочного интервала родительских тегов.

Например, вы устанавливаете атрибут высоты строки <body>, равный 150 %. Все теги веб-страницы унаследуют это значение. Однако наследуется не процентное значение, а *рассчитанное значение* межстрочного интервала. Допустим, основной размер шрифта установлен равным 10 пикселам. 150 % от 10 пикселов составляет 15. Все теги унаследуют межстрочный интервал размером 15 пикселов. Так, если на веб-странице есть абзац текста со шрифтом высотой 36 пикселов, его межстрочный интервал размером 15 пикселов будет намного меньше самого текста, а строки сольются вместе.

В данном примере вместо высоты строки 150 % для тега <body> лучше установить общий для всех тегов пропорциональный базовый межстрочный интервал в размере 1,5. Любой тег, вместо того чтобы наследовать точное абсолютное значение высоты строки в пикселах, просто умножает размер своего шрифта на этот коэффициент. Так, в вышеупомянутом примере, где абзац текста отображен размером шрифта 36 пикселов, межстрочный интервал составит: 1,5 × 36 = 54 пиксела.

Выравнивание текста

Одним из самых быстрых способов изменить внешний вид веб-страницы является выравнивание текста. Используя свойство `text-align`, вы можете разместить абзац в центре веб-страницы (горизонтально), расположить текст вдоль левого или правого края или выровнять по ширине (подобно абзацам настоящей книги). Обычно текст на странице выровнен по левому полю, но вам может понадобиться центри-

ровать заголовки. Чтобы устанавливать выравнивание для текста, пользуйтесь одним из следующих ключевых слов: `left`, `right`, `justify`, `center`.

```
text-align: center;
```

Выровненный текст замечательно выглядит на печатной странице, главным образом потому, что при большой разрешающей способности печати учитываются даже мельчайшие настройки интервалов, а также потому, что большинство программ, используемых для разметки печатного материала, могут писать через дефис длинные слова (тем самым пытаясь равномерно распределить символы по строкам). Это предотвращает большие, неприглядные промежутки между абзацами. Веб-страницы в форматировании ограничены возможностями намного более грубой регулировки интервалов из-за низкой разрешающей способности мониторов компьютеров и из-за того, что браузеры не умеют писать длинные слова через дефис. Когда вы пользуетесь ключевым словом `justify` для выравнивания по ширине, получаются совершенно разные промежутки между словами, текст становится трудночитаемым. Поэтому, применяя такое выравнивание на веб-страницах, убедитесь в том, что получившийся текст имеет приятный внешний вид.

ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

Метод стенографии для набора стилей, форматирования текста

Многократный повторный набор стилевых свойств — достаточно утомительное занятие, особенно если нужно использовать несколько различных текстовых свойств сразу. К счастью, в CSS есть свойство `font`, облегчающее написание стилей. Оно позволяет объединять несколько свойств в одну строку: `font-style`, `font-variant`, `font-weight`, `font-size`, `line-height` и `font-family`. Например, объявление `font: italic bold small-caps 18px/150% Arial, Helvetica, sans-serif;` создает полужирный, курсивый шрифт с малыми прописными буквами размером 18 пикселов, семейства Arial (или Helvetica, или sans-serif) с межстрочным интервалом 150 %. Имейте в виду следующие правила:

- не обязательно применять все эти свойства, но нужно включить размер шрифта и начертание (семейства шрифтов): `font: 1.5em Georgia, Times, serif;`;
- используйте один пробел между каждым значением свойства, а запятую только для того, чтобы отделить семейства шрифтов в списке: `Arial, Helvetica, sans-serif`;
- определяя межстрочный интервал, добавляйте слеш после размера шрифта, за которым должно следовать значение самого межстрочного интервала: `% 1.5em/150` или `24px/37px`;
- последние два свойства должны быть комбинацией: размер шрифта/межстрочный интервал (или `font-size/line-height`); все остальные могут быть перечислены в любом порядке. Например, оба объявления: `font: italic bold small-caps 1.5em Arial` и `font: bold small-caps italic 1.5em Arial;` — равнозначны и к ним применен одинаковый эффект.

Наконец, исключение значения из списка означает то же, что и установка его по умолчанию. Допустим, вы создали стиль `<p>`, который форматирует все абзацы полужирным курсивным шрифтом с малыми прописными буквами и межстрочным интервалом 2000 % (совсем не обязательно это повторять). Затем создали стилевой класс под названием, скажем, `.specialParagraph` с таким объявлением стиля шрифта: `font: 1.5em Arial;` — и применили его к какому-то абзацу текста. В результате данный абзац не унаследует полужирное курсивое начертание с малыми прописными буквами и межстрочный интервал. Исключение этих четырех значений из стиля `.specialParagraph` можно приравнять к написанию следующего: `font: normal normal normal 1.5em/normal Arial;`.

Отступ первой строки абзаца и удаление полей абзацев

В большинстве печатных изданий первая строка каждого абзаца имеет так называемый *абзацный отступ*. Он выделяет начало каждого абзаца текста, если нет дополнительных промежутков или увеличенных межстрочных интервалов, визуально разделяющих абзацы. В веб-страницах в Интернете нет отступов, но применяется увеличенный межстрочный интервал, как в книге.

Если у вас есть желание придать веб-страницам индивидуальность, сделать их непохожими на другие, то воспользуйтесь преимуществами таких CSS-свойств, как `text-indent` и `margin`. С их помощью вы можете создать отступ первой строки абзацев и удалить (или увеличить) поля.

Абзацный отступ

Для установки отступа первой строки абзаца можно использовать такие единицы измерения, как пиксел и em:

```
text-indent: 25px;
```

или

```
text-indent: 5em;
```

Значения в пикселах — абсолютные значения, точное число, в то время как em определяет размер отступа в количестве символов (базируется на текущем размере шрифта).

СОВЕТ

В свойстве абзацного отступа text-indent вы можете использовать отрицательные значения для создания выступа, то есть абзаца с выступающей (смещенной, «свисающей») влево первой строкой.

Вы можете также использовать процентные значения, но со свойством `text-indent` эти единицы измерения приобретают другое значение. В данном случае размер выступа, установленный в процентах, связан не со шрифтом текста, а с шириной элемента, в который заключен абзац. Например, если текстовый отступ установлен равным 50 % и абзац охватывает всю ширину окна браузера, то первая строка будет начинаться посередине экрана. Если вы меняете размеры окна, то изменяется ширина абзаца и, соответственно, отступ (о значениях свойств, устанавливаемых в процентах, и о том, как они взаимодействуют с шириной элементов веб-страницы, читайте в разд. «Определение параметров высоты и ширины» гл. 7).

Управление полями (границами) абзацев

Многие веб-дизайнеры не любят промежутки или дополнительные интервалы, которые любой браузер добавляет между абзацами. До появления CSS с этим приходилось мириться. К счастью, теперь можно воспользоваться свойствами

margin-top и margin-bottom для удаления (или увеличения) этих промежутков. Чтобы полностью избавиться от верхнего и нижнего полей, введите следующее:

```
margin-top: 0;
margin-bottom: 0;
```

Чтобы удалить поля между *всеми* абзацами веб-страницы, создайте такой стиль:

```
p {
  margin-top: 0;
  margin-bottom: 0;
}
```

Для установки значений абзацных полей, как и для отступов, вы можете применять такие единицы измерения, как пикселы или em. Можно также использовать проценты, но, как и в случае с абзацными отступами, процентные значения относятся к *ширине* элемента, в который заключен абзац. Во избежание путаницы, связанной с вычислением верхнего и нижнего полей, расчет которых базируется на ширине абзацев, проще применять значения в em или пикселах.

ПРИМЕЧАНИЕ

Поскольку не все браузеры обрабатывают верхнее и нижнее поля заголовков и абзацев согласованно, рекомендую просто обнулить (то есть удалить) все поля в начале таблицы стилей. Посмотреть на то, как это работает, можно в разд. «Начинаем с чистого листа» гл. 5.

Для создания специальных эффектов можно назначить *отрицательное* значение верхнему или нижнему абзацному полю. Например, верхняя граница, установленная равной 10 пикселам, приподнимает абзац выше на 10 пикселов, возможно даже визуально накладывая его на вышестоящий элемент веб-страницы.

Форматирование первой буквы, первой строки абзаца

CSS дает возможность форматирования абзаца с использованием псевдоэлементов :first-line и :first-letter (рис. 6.9). С технической точки зрения они являются селекторами, определяющими фрагмент, к которому применены CSS-свойства. С помощью псевдоэлемента :first-letter можно создать начальную заглавную букву или буквицу, имитируя рукописный стиль текста. Например, чтобы сделать первый символ каждого абзаца полужирным и выделить его красным цветом, необходимо написать следующее:

```
p:first-letter {
  font-weight: bold;
  color: red;
```

Для избирательной стилизации, скажем форматирования только первого символа определенного абзаца, можно применить стилевой класс, например, .intro:

```
<p class="intro">Text for the introductory paragaph goes here...</p>
```

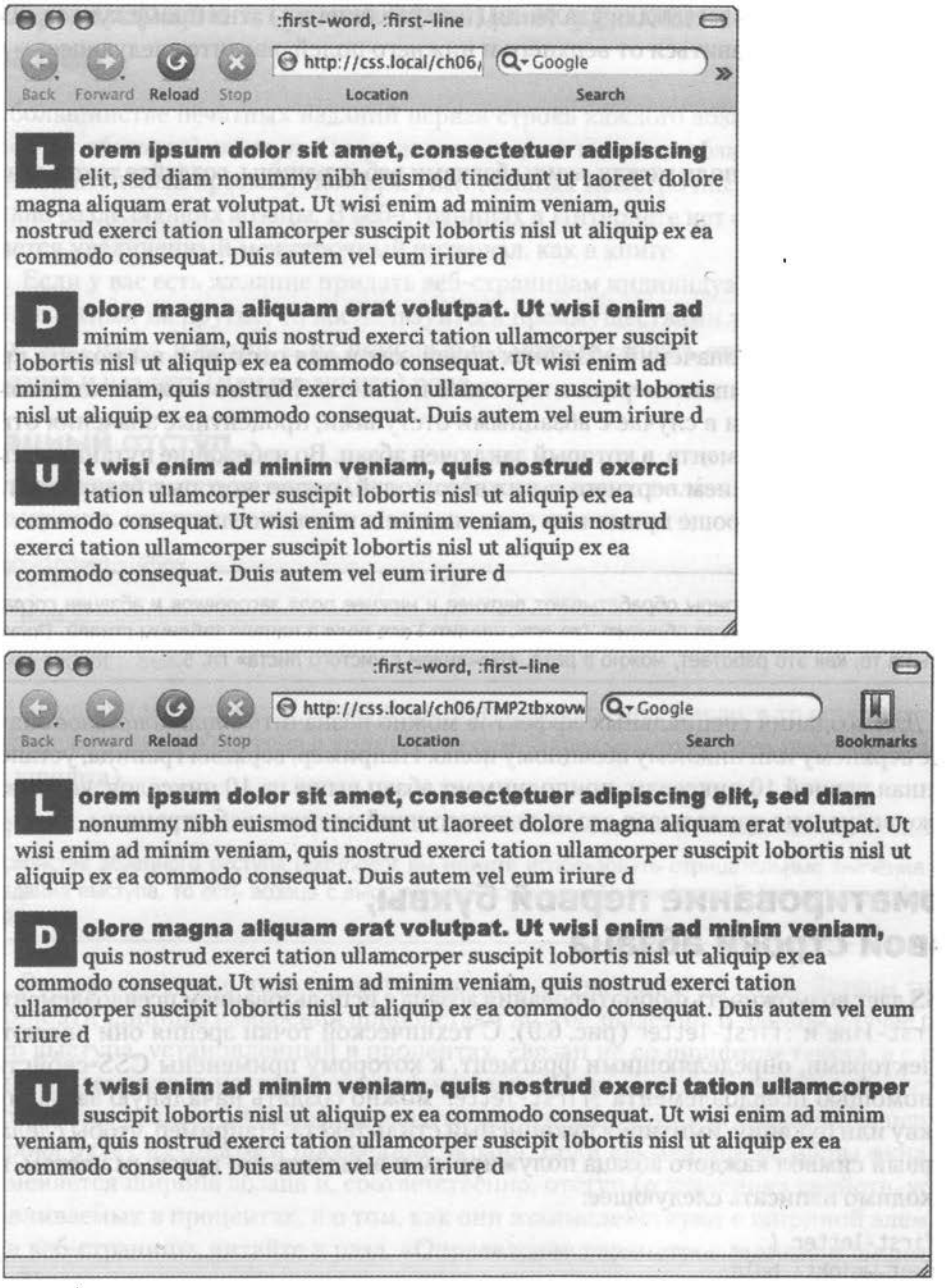


Рис. 6.9. Псевдоэлемент :first-letter форматирует первый символ стилизуемого элемента — заглавные буквы абзацев

Затем вы можете создать стиль с таким названием, как `.intro:first-letter`.

Псевдоэлемент `:first-line` форматирует первую строку абзаца. Вы можете применить его к любому блоку текста, будь то заголовок (`h2:first-line`) или абзац

(p:first-line). Как и в примере с :first-line, можно применить стилевой класс к единственному абзацу и отформатировать только его первую строку. Допустим, вы хотите отобразить прописными буквами все символы страницы. Примените стилевой класс к HTML-коду первого абзаца: <p class = "intro"> и создайте следующее:

```
.intro:first-line { text-transform: uppercase: }
```

ПРИМЕЧАНИЕ

По какой-то странной причине браузер Safari не распознает свойство text-transform, когда оно используется с псевдоэлементом :first-line. Другими словами, вы не можете использовать в Safari CSS для преобразования букв первой строки абзаца в прописные.

Стилизация списков

Теги <ul> и <ol> создают маркированные и нумерованные списки: взаимосвязанных элементов, пунктов или пронумерованных шагов, последовательности действий. Однако, как вы, наверное, заметили, не всегда подходит предопределенный браузером способ форматирования. Возможно, вы захотите заменить в маркированных списках стандартный маркер собственным, более красивым, использовать в нумерованных списках буквы вместо чисел и т. д.

Типы списков

Большинство браузеров отображают маркированные списки (теги <ul>), используя маркеры в виде окружности, а нумерованные списки (<ol>) — предваряя пункты числами. В CSS вы можете выбрать три типа маркеров: *диск* (круглый маркер с заполнением), *окружность* (круг, полый круглый маркер), *квадрат* (квадрат с зарисовкой). Для нумерованных списков предусмотрено шесть вариантов-схем нумерации: decimal, decimal-leading-zero, upper-alpha, lower-alpha, upper-roman, lower-roman (рис. 6.10). Все эти варианты можно выбрать, используя CSS-свойство list-style-type:

```
list-style-type: square;
```

или

```
list-style-type: upper-alpha;
```

Firefox и другие браузеры, построенные на основе Mozilla, например Camino (изображенный на рис. 6.10), корректно отображают списки с применением decimal-leading-zero, добавляя 0 перед однозначными числами, например 01. Internet Explorer 6 и 7 не распознают списки с применением decimal-leading-zero или lower-greek.

СОВЕТ

Можно также заменить числа нумерованных списков греческими символами (α, β, γ), используя ключевое слово lower-greek.

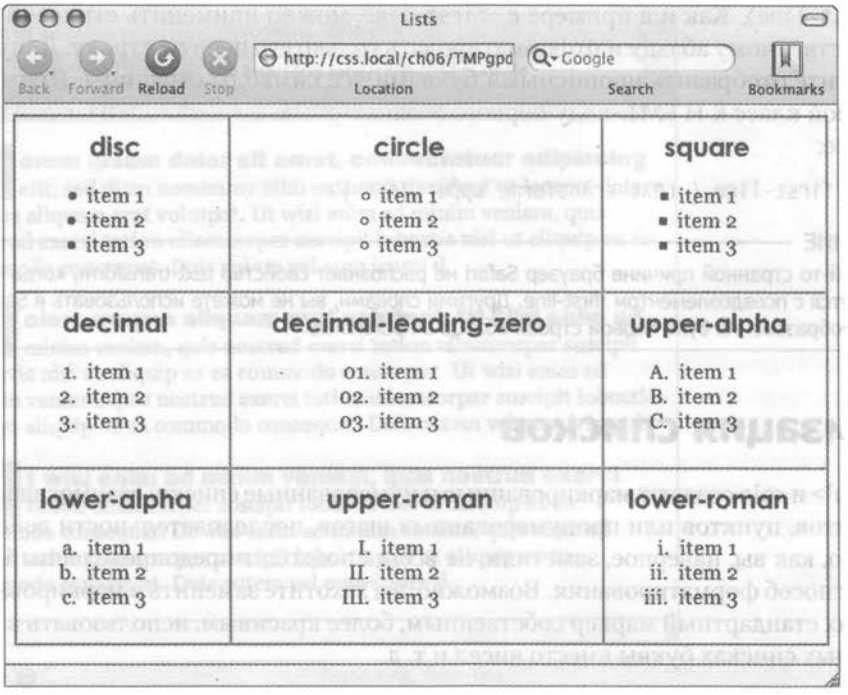


Рис. 6.10. Большинство браузеров отображают варианты decimal и decimal-leading-zero одинаково

Чаще всего это свойство используют при определении стилей, форматирующих <ol> или <ul>. Типичные примеры — включение свойства в стили `ol` или `ul`: `ul {list-style-type: square;}` либо в стилевой класс, применяемый к одному из этих тегов. Тем не менее вы можете применить это свойство и к отдельно взятому элементу списка (<li>). Вы даже можете применить несколько стилей с различными маркерами к отдельным пунктам-элементам одного и того же списка. Например, можете создать стиль тега <ul>, устанавливающий маркеры в виде квадратов, а затем создать стилевой класс `.circle`, который изменяет тип маркера на окружность:

```
.circle { list-style-type: circle; }
```

Теперь примените класс к элементам списка через один для чередования квадратных и круглых маркеров:

```
<ul>
  <li>Item K/li>
  <li class="circle">Item 2</li>
  <li>Item 3</li>
  <li class="circle">Item 4</li>
</ul>
```

Иногда может понадобиться вообще скрыть маркеры, в том числе тогда, когда вы захотите использовать свои собственные графические маркеры (см. обучающий урок этой главы). Кроме того, когда панель навигации сайта представляет собой

список ссылок, вы также можете использовать список <ul>, скрыв его маркеры (см. подраздел «Использование маркированных списков» разд. «Создание панелей навигации» гл. 9). Чтобы отключить отображение маркеров, используйте ключевое слово none:

```
list-style-type: none;
```

Позиционирование маркеров и нумерации списков

Как правило, браузеры отображают маркеры или числа списков слева от текста пунктов — элементов списка (рис. 6.11, *слева*). Ключевое слово outside визуально выделяет список и каждый его элемент из всего текста. С помощью CSS вы можете управлять размещением маркеров, используя свойство list-style-position. Определить местоположение можно, как говорилось выше, слева от текста пунктов списка, выделяя их обособленно (стандартный способ отображения браузерами), или *внутри* текстовых блоков элементов списка (см. рис. 6.11, *справа*). Значение inside обеспечит списку максимальную ширину на странице:

```
list-style-position: outside;
```

или

```
list-style-position: inside;
```

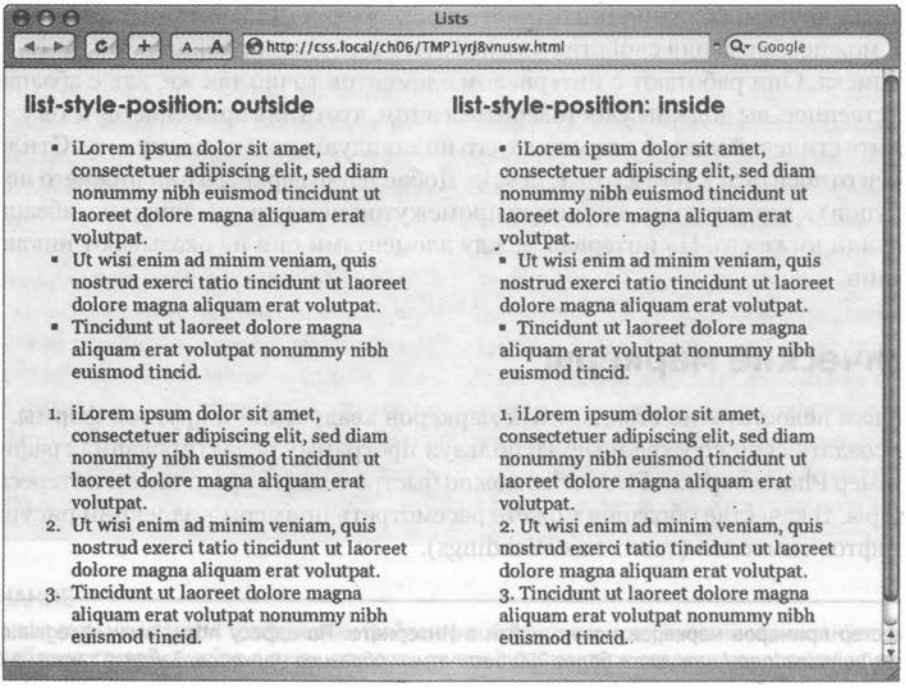


Рис. 6.11. Используя свойство list-style-position, вы можете управлять позиционированием маркеров и чисел в списках

СОВЕТ

Вы можете изменить промежуток между маркером и текстом путем увеличения или уменьшения значения свойства padding-left (см. разд. «Управление размерами полей и отступов» гл. 7). Чтобы использовать это свойство, вам нужно создать стиль для тега элементов списка <li>. Этот способ работает только в том случае, если свойство list-style-position определено со значением outside (или вообще отсутствует).

Кроме того, если вам не нравится, что браузеры делают для списков отступ, смещая все содержимое вправо, можете переопределить стиль, установив значения свойств `margin-left` и `padding-left` равными 0. Чтобы удалить отступ, можно создать такой групповой селектор:

```
ul, ol {
  padding-left: 0;
  margin-left: 0;
}
```

Вы можете создать стилевой класс с такими свойствами и применить его к конкретным тегам <ul> или <ol>. Рекомендую установить собственные значения свойств `padding` и `margin` по той причине, что одни браузеры для управления отступом используют свойство `padding` (Firefox, Mozilla, Safari), а другие — `margin` (Internet Explorer). Подробно о свойствах `padding` и `margin` вы можете прочесть в следующей главе.

В обычном порядке браузеры отображают пункты маркированных списков один над другим, без дополнительного промежутка. Добавить интервал между ними можно, применяя свойства `margin-top` и `margin-bottom` к конкретным элементам списка. Они работают с интервалом элементов точно так же, как с абзацами. Единственное, вы должны удостовериться в том, что стиль применяется к тегу <li>: создайте стилевой класс и примените его индивидуально к каждому тегу. Стиль не должен относиться к тегам <ul> или <ol>. Добавление верхнего или нижнего полей (отступов) к этим тегам увеличивает промежуток между всем списком и абзацами выше или ниже его. На интервал между элементами они не оказывают никакого влияния.

Графические маркеры

Если вам недостаточно стандартных маркеров квадратной и круглой формы, можете создать свои собственные. Используя программу редактирования графики, например Photoshop или Fireworks, можно быстро создать красочные и интересные маркеры. В качестве образцов можете рассмотреть примеры коллекции рисунков и шрифтов символов (таких как Webdings).

СОВЕТ

Множество примеров маркеров можно найти в Интернете. По адресу http://www.stylegala.com/features/bulletmadness/ находится более 200 бесплатных образцов маркеров. Зайдите также на сайт www.cssjuice.com/38-free-icon-checkpoints, где собраны ресурсы с коллекциями значков и маркеров.

CSS-свойство `list-style-image` позволяет определить путь к графическому символу на сайте таким же образом, как вы указываете местонахождение файла с изображением, используя атрибут `src` HTML-тега `<img>`. Синтаксис команды следующий:

```
list-style-image: url(images/bullet.gif);
```

Термин `url` и круглые скобки обязательны. Часть, заключенная в круглые скобки, — в данном примере `images/bullet.gif` — это и есть путь к графическому символу. Обратите также внимание на то, что, в отличие от HTML, не нужно заключать путь в кавычки.

ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

Настройка параметров маркеров и чисел в списках

Возможно, вы хотите стилизовать числа нумерованных списков таким образом, чтобы они отображались полужирным шрифтом красного цвета вместо давно надоевшего черного. Как можно настроить внешний вид маркеров и чисел в списках?

CSS предлагает несколько способов настройки параметров маркеров, предваряющих пункты — элементы списка. Вы можете использовать собственные графические символы, как описано выше. Но имеется еще два метода. Один из них достаточно трудоемкий, однако работает с большинством браузеров, другой — самый современный, но не работает в Internet Explorer 7 или его более ранних версиях.

Рассмотрим первый метод. Предположим, вы хотите, чтобы числа нумерованного списка были отображены полужирным шрифтом красного цвета, а текст пунктов — обычным черным. Создайте стилевой класс, как делали это применительно к тегам `<ol>` и `<li>`. На данном этапе все содержимое списка должно отображаться полужирным шрифтом красного цвета.

Затем создайте стилевой класс, например `.regularList`, который устанавливает черный цвет шрифта и нормальную плотность (то есть не полужирную). Теперь заключите каждый элемент списка в тег `<span>` и примените к ним стилевой класс. Например: `<li><span class="regularL-ist">Item 1</span></li>`. Сейчас маркеры отображаются полужирным красным шрифтом, а текст — обычным черным цветом. К сожалению, придется заключать в тег каждый элемент списка.

Продвинутый способ, экономящий объем CSS-кода, состоит в том, чтобы использовать так называемое *сгенерированное содержимое*. По сути, это все то, что не набирается вручную в виде кода веб-страницы, а автоматически добавляется браузером при отображении страницы. Хороший пример — сами маркеры. Вы не вводите знаки маркера при создании списка — они добавляются на веб-страницу сами. С помощью CSS можно сообщить браузеру, чтобы он генерировал, добавлял такое содержимое и даже отформатировал должным образом все, что находится перед текстом пунктов списка, — `<li>. http:///`. О генерируемом содержимом вы узнаете в разд. «Генерируемое содержимое страницы» гл. 16.

ПРИМЕЧАНИЕ

Указывая путь или адрес к графическим файлам (изображениям, графическим символам) во внешней таблице стилей, имейте в виду, что путь должен указываться относительно таблицы стилей, а не веб-страницы. Более подробно об этом вы прочтете в разд. «Фоновые изображения» гл. 8, когда мы начнем использовать графику.

Свойство list-style-image позволяет использовать графические символы в качестве маркеров. Однако оно не обеспечивает управления его размещением. Маркер может оказаться слишком высоко или низко расположенным относительно пункта списка. Придется переделывать сам графический символ маркера, пока он не будет сочетаться. О более грамотном подходе вы прочтете в гл. 8. Он основан на использовании свойства background-image. Это свойство позволяет точно позиционировать графические элементы, в том числе маркеры в списках.

СОВЕТ

Как и в случае со свойством font (см. врезку «Информация для опытных пользователей» в разд. «Форматирование абзацев текста» этой главы), здесь применяется метод упрощенного набора атрибутов списков. В свойстве list-style могут быть перечислены все параметры форматирования списков, в том числе list-style-image, list-style-position и list-style-type. Например, стиль ul { list-style: circle inside; } отобразит маркированные списки с маркерами в виде окружностей без заполнения, не выделяя их из текста пунктов — элементов списка. Когда вы описываете стиль списка с одновременным указанием типа маркера и графического символа — list-style: circle url(images/bullet.gif) inside; — и графический символ не может быть найден по заданному пути, браузер будет использовать предопределенный маркер, в данном случае — окружность.

Обучающий урок: форматирование текста в действии

В этом обучающем уроке мы попрактикуемся в стилизации таких элементов веб-страниц, как заголовки, списки, абзацы текста, используя мощные средства форматирования CSS.

Чтобы начать обучающий урок, вы должны загрузить файлы, содержащие учебный материал. Как это сделать, рассказывается в конце гл. 2. Файлы текущей обучающей программы находятся в папке с названием 06.

Настройка параметров страницы

Начнем с таблицы стилей и добавим стили для форматирования основного текста страницы.

1. Запустите браузер и откройте файл text.html (рис. 6.12).

   Здесь пока особо не на что смотреть — есть только коллекция заголовков, абзацы и один маркированный список. Но скоро вы заметно преобразите эту страницу.

2. Откройте файл text.html в HTML-редакторе.

   Начнем с добавления к этому файлу внутренней таблицы стилей.

3. В заголовке <head> веб-страницы щелкните кнопкой мыши сразу после закрывающей скобки тега <title>. Нажмите Enter и наберите <style type="text/css">. Дважды нажмите Enter и наберите </style>.

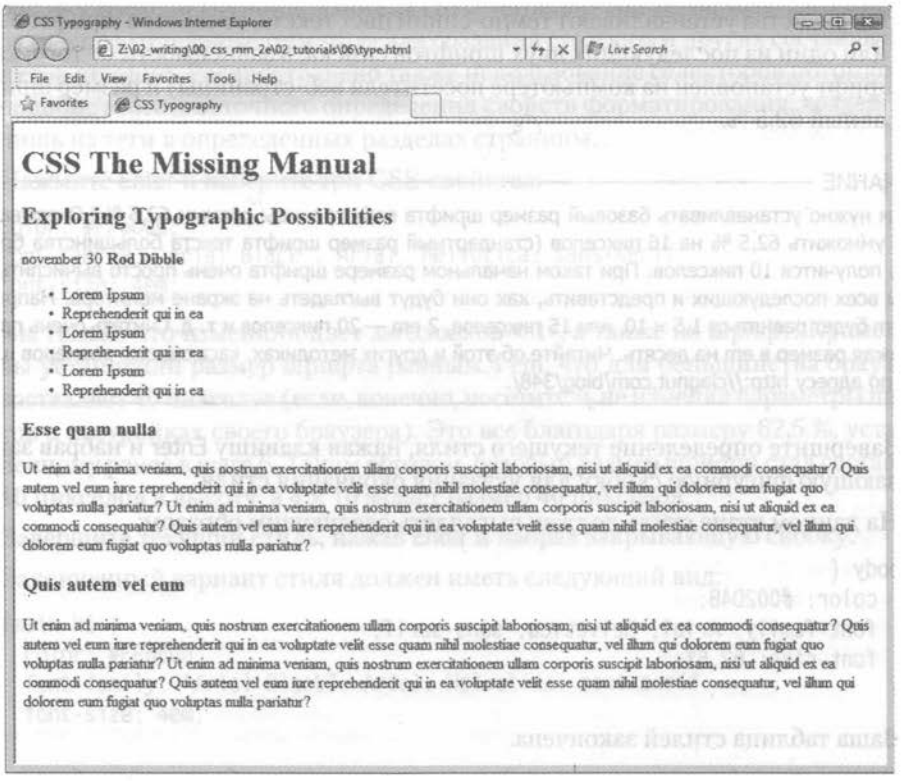


Рис. 6.12. Работа над любой веб-страницей начинается с обычного HTML-содержимого

Теперь, когда основные теги стилей на месте, вы добавите сброс стандартных настроек CSS. Вместо того чтобы набирать вручную весь код, просто скопируйте и вставьте CSS из внешней таблицы стилей.

4. Откройте файл `reset.css`. Скопируйте весь код оттуда и вставьте его между открывающим и закрывающим тегами `<style>`, добавленными в предыдущем шаге.

   Если вы просмотрите файл `text.html` в браузере, то увидите, что текст выглядит примерно одинаково, то есть все стандартное форматирование браузера было устранено и теперь можно начинать с чистого листа.

5. Нажмите **Enter** и введите `body {`.

   Это базовый селектор, который применяется к тегу `<body>`. Как обсуждалось в гл. 4, другие теги наследуют свойства этого тега. Вы можете настроить некоторые основные параметры текста, например шрифт, цвет, размер шрифта для последующих тегов, и использовать их в качестве начальных настроек.

6. Снова нажмите **Enter** и добавьте следующие три свойства:

```
color: #002D4B;
font-family: Arial, Helvetica, sans-serif;
font-size: 62.5%;
```

Эти свойства устанавливают темно-синий цвет текста, семейство шрифтов Arial (или один из последующих двух шрифтов списка, в зависимости от того, какой шрифт установлен на компьютере посетителя веб-страницы) и размер шрифта, равный 62,5 %.

ПРИМЕЧАНИЕ

Зачем нужно устанавливать базовый размер шрифта веб-страницы равным 62,5 %? Оказывается, если умножить 62,5 % на 16 пикселов (стандартный размер шрифта текста большинства браузеров), получится 10 пикселов. При таком начальном размере шрифта очень просто вычислить размеры всех последующих и представить, как они будут выглядеть на экране монитора. Например, 1,5 em будет равняться 1,5 × 10, или 15 пикселов, 2 em — 20 пикселов и т. д. Считать очень просто, умножая размер в em на десять. Читайте об этой и других методиках, касающихся размеров шрифтов, по адресу http://clagnut.com/blog/348/.

7. Завершите определение текущего стиля, нажав клавишу Enter и набрав закрывающую фигурную скобку для указания окончания стиля.

   На данном этапе стиль должен выглядеть следующим образом:

```
body {
  color: #002D4B;
  font-family: Arial, Helvetica, sans-serif;
  font-size: 62.5%;
}
```

   Ваша таблица стилей закончена.

8. Сохраните веб-страницу и откройте ее для просмотра в браузере, чтобы увидеть результат работы.

   Текст на странице изменил свой цвет и шрифт. Он стал действительно маленьким. Не волнуйтесь, это из-за размера 62,5 %, который вы установили в шаге 6. Это начальный параметр для всего текста, и вы сможете спокойно увеличить текст, определяя размеры em для других тегов.

Форматирование заголовков и абзацев

Теперь, когда основное форматирование текста выполнено, пришло время улучшить представление заголовков и абзацев.

1. Вернитесь к файлу `text.html` в HTML-редакторе. Установите курсор за закрывающую скобку селектора тега `<body>` во внутренней таблице стилей, нажмите Enter для создания новой строки кода и наберите `#main h1 {`.

   Это селектор потомка, более специфичный по сравнению с базовым селектором тега HTML. В данном случае селектор указывает браузеру применить следующее форматирование к любому тегу `<h1>`, находящемуся внутри другого тега с идентификатором `main`. Если вы просмотрите код HTML веб-страницы, то увидите, что там есть тег `<div>` с идентификатором `main` (`<div id="main">`). Как вы узнаете позже, при создании дизайнов, основанных на CSS, достаточно

распространено группирование HTML-тегов внутри тегов . Вы сможете размещать отдельные теги для создания столбцов и других составных разметок страниц. Распространено также использование селекторов потомков наподобие этого для точного определения свойств форматирования, воздействуя лишь на теги в определенных разделах страницы.

2. Нажмите Enter и наберите три CSS-свойства:

```
color: #FF6600;
font-family: "Arial Black", Arial, Helvetica, sans-serif;
font-size: 4em;
```

Вы только что изменили цвет заголовков <h1>, а также их шрифт. Кроме того, вы установили размер шрифта равным 4 em, что для большинства браузеров составляет 40 пикселов (если, конечно, посетитель не изменял параметры шрифтов в настройках своего браузера). Это все благодаря размеру 62,5 %, установленному ранее в шаге 6. Таким образом, базовый размер шрифта стал составлять 10 пикселов в высоту, а 4 × 10 задает размер 40 пикселов.

3. Завершите текущий стиль, нажав Enter и набрав закрывающую скобку.

Законченный вариант стиля должен иметь следующий вид:

```
#main h1 {
  color: #FF6600;
  font-family: "Arial Black", Arial, Helvetica, sans-serif;
  font-size: 4em;
}
```

4. Сохраните файл и воспользуйтесь предварительным просмотром в браузере.

Теперь изменим внешний вид остальных заголовков и абзацев.

5. Вернитесь к файлу text.html в HTML-редакторе. Переведите указатель за закрывающую скобку стиля <h1>, нажмите Enter и добавьте следующие стили:

```
#main h2 {
  font: bold 3.5em "Hoefler Text", Garamond, Times, serif;
  border-bottom: 1px solid #002D4B;
  margin-top: 25px;
}
```

Здесь идет еще один наследуемый селектор, который относится только к тегам <h2>, находящимся внутри другого тега с идентификатором main. Свойство font сокращает количество кода, объединяя в себе font-weight, font-size и font-family. Другими словами, одна строка делает заголовок полужирным, устанавливает для него высоту 3,5 em и определяет шрифт.

Кроме того, этот стиль добавляет декоративную границу под заголовком и немного пространства между заголовком и тегом над ним (то есть пространство добавляется между заголовками «CSS The Missing Manual» и «Exploring Typographic Possibilities»).

Пришло время взяться за другие заголовки.

6. Добавьте новый стиль под тем, который вы создавали в предыдущем шаге:

```
#main h3 {
  color: #F60;
  font-size: 1.9em;
  font-weight: bold;
  text-transform: uppercase;
  margin-top: 25px;
  margin-bottom: 10px;
}
```

Этот стиль устраняет некоторые свойства обычного форматирования: цвет, размер шрифта, жирность, а также использует свойство `text-transform`, чтобы все буквы текста внутри заголовка `<h3>` стали прописными. Наконец, он добавляет немного пространства сверху и снизу заголовка с помощью свойства `margin`.

Далее вы улучшите внешний вид абзацев.

7. Добавьте новый стиль на страницу:

```
#main p {
  font-size: 1.5em;
  line-height: 150%;
  margin-left: 150px;
  margin-right: 50px;
  margin-bottom: 10px;
}
```

Этот стиль содержит свойство `line-height`, которое определяет расстояние между строками. Значение 150 % добавляет немного больше пространства между строками абзаца, чем вы обычно видите в браузере. Благодаря этому дополнительному пространству текст чувствует себя свободнее, а предложения становится легче читать (но только если вы используете латиницу).

Этот стиль также увеличивает размер шрифта до 1,5 em (15 пикселов для большинства браузеров) и отодвигает абзац от левого и правого краев страницы. Вы можете заметить, что для свойства `margin` приходится набирать слишком много кода. Но, к счастью, как вы узнаете в следующей главе, есть сокращенная запись этого свойства.

Теперь попробуем более продвинутый тип селектора.

8. Добавьте следующий стиль к таблице стилей:

```
#main p:first-line {
  font-weight: bold;
  color: #999;
}
```

Псевдоэлемент `:first-line` воздействует только на первую строку абзаца. В этом случае именно первая строка текста в каждом абзаце внутри основного `<div>` будет окрашена в серый цвет и выделена полужирным.

9. Сохраните страницу и откройте ее для просмотра результатов в браузере.

На текущий момент ваша веб-страница должна быть похожа на ту, что показана на рис. 6.13.

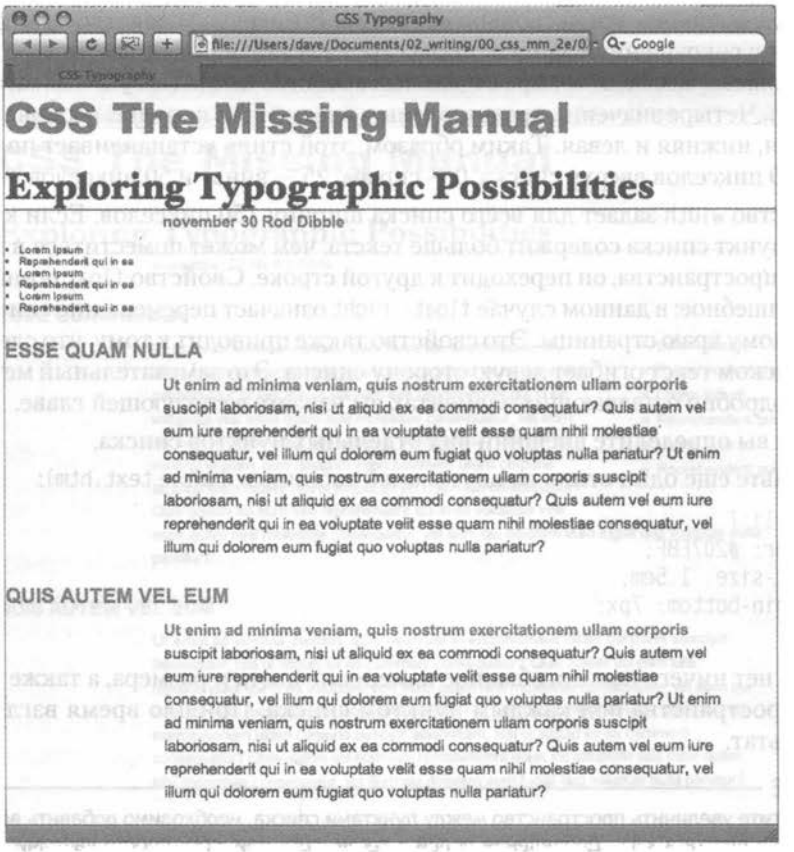


Рис. 6.13. Внешний вид веб-страницы преображается: параметры заголовков, абзацев и основного текста приведены в порядок

Форматирование списков

На этой странице есть один маркированный список. Вы переместите его вверх к правому краю страницы и сделаете так, чтобы текст, идущий за этим списком, обтекая его. С помощью CSS сделать это достаточно легко.

1. Вернитесь к файлу `text.html` в HTML-редакторе. Добавьте следующий стиль в конце внутренней таблицы стилей:

```
#main ul {
  margin: 50px 0 25px 50px;
  width: 150px;
  float: right;
}
```

При форматировании списков вы обычно создаете стили для двух разных элементов: непосредственно для самого списка (тега `<ul>` для маркированных либо тега `<ol>` для нумерованных списков) и отдельных элементов списка (тег `<li>`). Этот стиль управляет всем списком.

В стиле выполняется несколько действий. Во-первых, свойство `margin` используется в сокращенной записи. Одна строка устанавливает все четыре поля вокруг списка, замещая четыре отдельных свойства `margin` (`margin-top`, `margin-right` и т. д.). Четыре значения представлены в следующем порядке: верхняя сторона, правая, нижняя и левая. Таким образом, этой стиль устанавливает поле шириной 50 пикселов вверху списка, 0 — справа, 25 — внизу и 50 пикселов — слева.

Свойство `width` задает для всего списка ширину 150 пикселов. Если какой-нибудь пункт списка содержит больше текста, чем может поместиться в пределах этого пространства, он переходит к другой строке. Свойство `float` по-настоящему волшебное: в данном случае `float: right` означает перемещение списка вверх к правому краю страницы. Это свойство также приводит к тому, что следующий за списком текст огибает левую сторону списка. Это замечательный метод, а более подробно о плавающих элементах вы узнаете в следующей главе.

Далее вы определите внешний вид отдельных пунктов списка.

2. Добавьте еще один стиль во внутренней таблице в файле `text.html`:

```
#main li {
  color: #207EBF;
  font-size: 1.5em;
  margin-bottom: 7px;
}
```

Здесь нет ничего нового: обычное изменение цвета и размера, а также добавление пространства под каждым пунктом списка. Пришло время взглянуть на результат.

ПРИМЕЧАНИЕ

Если вы хотите увеличить пространство между пунктами списка, необходимо добавить верхние или нижние поля для тега <li>. Добавление полей к тегам <ul> или <ol> просто увеличит пространство вокруг всего списка.

3. Сохраните страницу и воспользуйтесь предварительным просмотром в браузере.

Теперь страница должна быть похожа на ту, что представлена на рис. 6.14.

Точная настройка с классами

Иногда появляется желание иметь еще больше контроля над тем, как применяется стиль. Например, вы хотите видеть большинство абзацев в каком-либо разделе страницы одинаковыми, но, вероятно, пожелаете определить уникальный вид для одного или двух из них. В этом обучающем примере абзац рядом с верхней частью страницы — «November 30 Rod Dibble» — содержит особую информацию (о дате публикации и об авторе). Сделаем так, чтобы этот абзац выделялся среди других, добавив класс к HTML и создав для него стиль.

1. Найдите вышеупомянутый абзац в коде HTML (`<p>november 30 <strong>Rod Dibble</strong></p>`) и добавьте `class="byline"` к открывающему тегу `<p>`. Код HTML должен выглядеть так:

```
<p class="byline">november 30 <strong>Rod Dibble</strong></p>
```

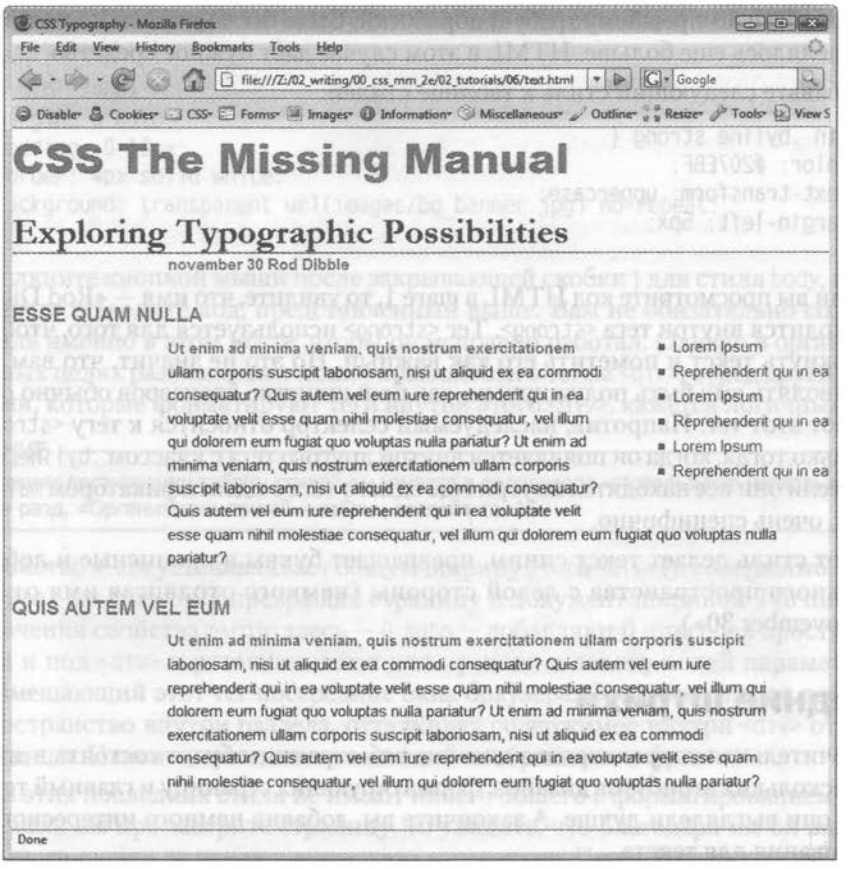


Рис. 6.14. Свойство float дает интересные возможности при проектировании дизайна. В данном случае маркированный список перемещается к правому краю страницы

Теперь осталось создать стилевой класс, переопределяющий общие свойства форматирования абзацев на этой странице.

2. Во внутренней таблице стилей рядом с верхней частью страницы добавьте стиль для этого абзаца:

```
#main .byline {
  color: #999999;
  font-size: 1.6em;
  margin: 5px 0 25px 50px;
}
```

Этот стиль настраивает цвет, размер и расположение только одного абзаца. Заметьте, что, если бы вы назвали этот стиль просто `.byline` (базовый селектор класса), он бы не работал. Благодаря правилам каскадности, описанным в предыдущей главе, `.byline` является менее значимым, чем стиль `#main p`, созданный в шаге 7 пару страниц назад, поэтому он будет не способен переопределить цвет, размер и поля, указанные в `#main p`. А `#main .byline`, в свою очередь, является более значимым и успешно форматирует верхний абзац.

Этот абзац по-прежнему требует доработки. Было бы замечательно, если бы имя выделялось еще больше. HTML в этом случае дает нужное средство.

3. Добавьте следующий стиль к таблице стилей:

```
#main .byline strong {
  color: #207EBF;
  text-transform: uppercase;
  margin-left: 5px;
}
```

Если вы просмотрите код HTML в шаге 1, то увидите, что имя — «Rod Dibble» — находится внутри тега . Тег используется для того, чтобы подчеркнуть текст и пометить его как важный. Но это не значит, что вам нужно позволять ему быть полужирным, как большинство браузеров обычно отображают этот тег. Напротив, наследуемый селектор относится к тегу , но только тогда, когда он появляется внутри другого тега с классом .byline, и только если они все находятся внутри еще одного тега с идентификатором main — вот так, очень специфично.

Этот стиль делает текст синим, превращает буквы в прописные и добавляет немного пространства с левой стороны (немного отодвигая имя от текста «November 30»).

Последние штрихи

Заключительная корректировка дизайна веб-страницы будет состоять в добавлении нескольких атрибутов дизайна, форматирующих страницу и главный тег , чтобы они выглядели лучше. А закончите вы, добавив немного интересного форматирования для текста.

1. Вернитесь к файлу text.html в HTML-редакторе.

   Сначала зададим фоновый цвет и изображение для страницы.

2. Найдите стиль body в верхней части внутренней таблицы стилей и добавьте одно новое свойство (изменения выделены полужирным):

```
body {
  font-size: 62.5%;
  font-family: Arial, Helvetica, sans-serif;
  color: #002D4B;
  background: #E1EEFD url(images/bg_body.png) repeat-x;
}
```

Свойство background представляет собой мощный инструмент для любого веб-дизайнера. Вы уже использовали его пару раз в предыдущих обучающих примерах; оно позволяет добавлять цвет, а также вставлять изображения и управлять их расположением в фоне какого-либо тега. Вы узнаете все тонкости этого свойства в следующей главе, а сейчас просто отметим, что введенная строка изменяет цвет фона страницы на светло-голубой и добавляет темно-синюю полосу в верхнюю часть страницы.

Далее мы преобразим основной тег .

3. Добавьте еще один стиль между стилем body и стилем #main h1:

```
#main {
  width: 740px;
  margin: 0 auto;
  padding: 0 10px;
  border: 4px solid white;
  background: transparent url(images/bg_banner.jpg) no-repeat;
}
```

Щелкните кнопкой мыши после закрывающей скобки } для стиля body, нажмите Enter и введите код, представленный выше. Вам не обязательно создавать стиль именно в этом месте, чтобы он исправно работал. Однако в организационных целях размещение стиля, управляющего тегом , перед другими стилями, которые форматируют теги внутри этого , кажется логичным.

ПРИМЕЧАНИЕ

О стратегиях организации таблиц стилей вы узнаете в подразделе «Используйте несколько таблиц стилей» разд. «Организация стилей и таблиц стилей» гл. 15.

Свойство width устанавливает общую ширину этого (и содержимого внутри его), по существу превращая страницу в документ шириной 740 пикселов. Значения свойства margin здесь — 0 auto — добавляют 0 пикселов пространства над и под и устанавливают для правого и левого полей параметр auto, размещающий этот тег посередине окна браузера. Свойство padding добавляет пространство внутри раздела, отталкивая содержимое внутри от линии границы. Наконец, мы также поместили изображение в фоне .

Два этих последних стиля не имеют ничего общего с форматированием текста, но, если вы просмотрите страницу, то увидите, что благодаря им он выглядит намного лучше, за исключением двух заголовков. Первый из них недостаточно жирный, а второй должен появляться под недавно добавленным рисунком.

4. Добавьте один последний стиль сразу после стиля #main h1:

```
#main h1 strong {
  font-size: 150px;
  color: white;
  line-height: 1em;
  margin-right: -1.25em;
}
```

HTML для заголовка выглядит следующим образом:

```
<h1><strong>CSS</strong> The Missing Manual</h1>.
```

«CSS» заключено внутри тега , так что данный наследуемый селектор форматирует только этот текст (в этом смысле он подобен стилю, который вы добавили в шаге 3 ранее). Размер шрифта был увеличен, его цвет изменился, а высота строки задана так, что теперь вписывается в верхнюю часть страницы. Вы заметите, что высота строки равна 1 em, а как вы читали в начале этой книги, em основывается на текущем размере шрифта элемента, поэтому в данном случае высота строки будет переведена в 150 пикселов — таков размер шрифта данного стиля.

Один интересный прием позволяет осуществить свойство `margin-right`, для которого устанавлено отрицательное значение: `-1.25em`. Поскольку положительные поля отодвигают элементы, отрицательные, в свою очередь, присоединяют элементы друг к другу. Так, в данном случае остальной текст заголовка («The Missing Manual») располагается поверх «CSS».

ПРИМЕЧАНИЕ

Использование отрицательных полей разрешено в CSS (хотя это и сложная работа). Их даже применяют для некоторых достаточно продвинутых разметок CSS.

5. Сохраните файл и просмотрите его в браузере.

   Веб-страница должна иметь вид, представленный на рис. 6.15. Теперь вы можете сравнить получившийся файл с законченным вариантом `text.html`, который находится в папке **06_finished** учебного материала.

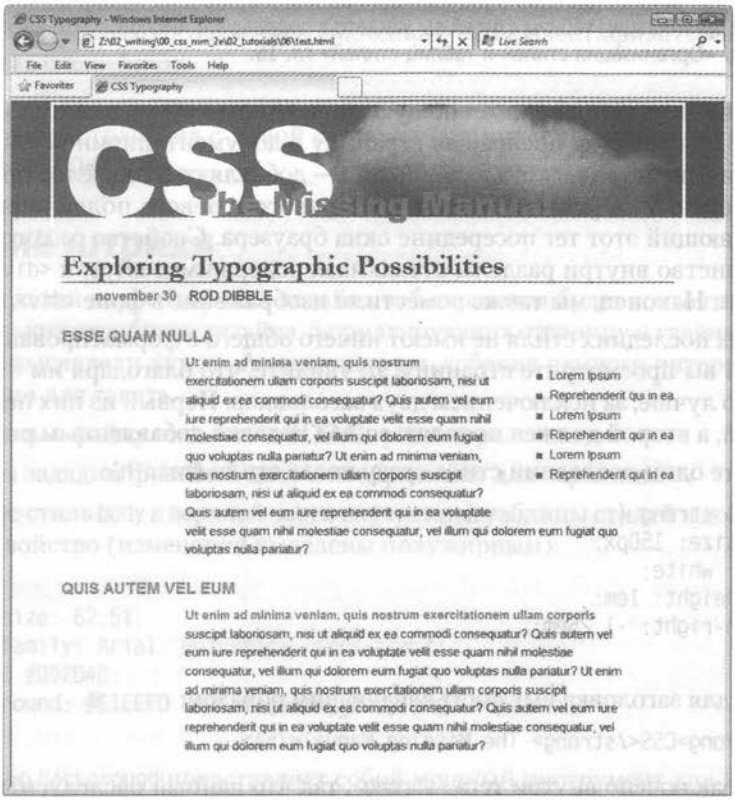


Рис. 6.15. С небольшой помощью CSS можно превратить простой текст в документ с профессиональным дизайном

Вы изучили основные свойства форматирования текста, предлагаемые CSS, и превратили малопривлекательный текст на языке HTML в страницу с отличным дизайном. В следующей главе мы рассмотрим применение на веб-страницах графики, рамок, полей и прочих мощных команд форматирования, предлагаемых CSS.

7 Поля, отступы, границы

На любой HTML-тег воздействует множество CSS-свойств, определяющих, каким образом он будет отображен браузером. Некоторые из них, например границы (рамки) и фоновый цвет, непосредственно видимы для глаза. Другие нельзя определить явно (например, отступы и поля), но они также обеспечивают форматирование. Понимая, как эти атрибуты работают, вы можете создать привлекательные столбцы, меню, элементы навигации, а также управлять пространством вокруг них (веб-дизайнеры называют его *свободным, незаполненным пространством*). Это делается, чтобы ваши веб-страницы не казались беспорядочными и нечитаемыми и вообще выглядели профессионально.

Все свойства, описываемые в данной главе, составляют основу *блочной модели CSS*, которая представляет одну из важнейших составляющих этого языка.

Понятие блочной модели

Когда вы думаете об абзаце текста или заголовке, то представляете буквы, слова, предложения. Фотографии, логотипы и другие изображения должны ассоциироваться с тегом `<img>`. Браузер обрабатывает все теги как небольшие *блоки*. Для него любой тег — контейнер с содержимым: текстом, изображением или другими тегами (рис. 7.1). Область в пределах границ, которая включает содержимое и отступы, может также иметь свой цвет фона. Фактически он является своеобразной подложкой, то есть расположен поверх фона. Таким образом, когда вы назначаете границы в виде штриховой линии, цвет проступает в промежутках между точками или штрихами линий границ.

Все то разнообразие свойств, в которое заключено содержимое, образует блок, или контейнер.

- `Padding` — *отступ, «заполнение»* — промежуток между содержимым и его границей. Отступ отделяет фотографию от окаймляющей ее рамки.
- `Border` — *граница, рамка* — линия, черта, контур с любой стороны элемента. Граница может быть со всех четырех сторон или с одной стороны в любой их комбинации.
- `Background-color` — *цвет фона* — заполняет пространство внутри границы, включая область отступа `padding`.

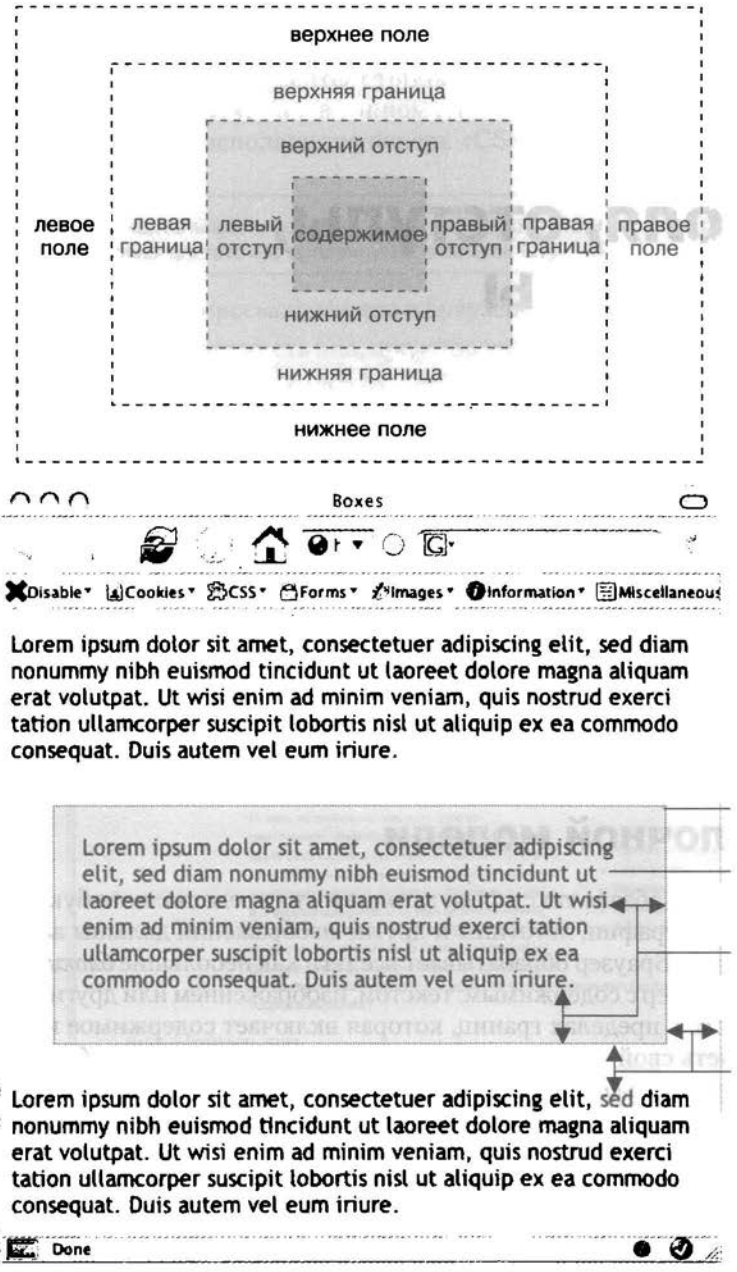


Рис. 7.1. Блочную структуру элемента образует содержимое тега (например, несколько предложений текста), а также отступы, границы и поля

- `Margin` — *поле* — это то пустое пространство, которое отделяет один тег от другого. Полем, например, является промежуток, который находится между нижним краем одного абзаца текста и верхним последующего.

Для заданного тега можно применить любые комбинации. Вы можете установить только поле или добавить границы и отступы и т. д. Если вы сами явно не устанавливаете какие-то из этих свойств, то браузер предопределит их в своих настройках по умолчанию. Как правило, к тегам веб-страниц не добавляют ни отступов, ни границ. В то же время теги заголовков и абзацев по умолчанию форматируются браузерами с предопределенными значениями верхнего и нижнего полей.

ПРИМЕЧАНИЕ

Поскольку разные браузеры применяют разное количество отступов и полей к одним и тем же тегам, лучше всего обнулять значения этих свойств для всех тегов. Другими словами, используйте набор простых стилей, названный сбросом значений CSS, для удаления отступов и полей из тегов HTML. Потом, когда вы будете создавать дополнительные стили, добавляющие поля и отступы, вы сможете быть уверены в том, что у вас будет согласованный вид элементов в разных браузерах.

Управление размерами полей и отступов

Как поля, так и отступы добавляют промежутки вокруг содержимого тегов. Свойства `margin` и `padding` используются для отделения одного элемента веб-страницы от другого. Можно использовать их, например, чтобы добавить пустое пространство между навигационным меню слева и основным содержимым главного раздела веб-страницы справа. Возможно, вы захотите отодвинуть границу от края фотографии (рис. 7.2).

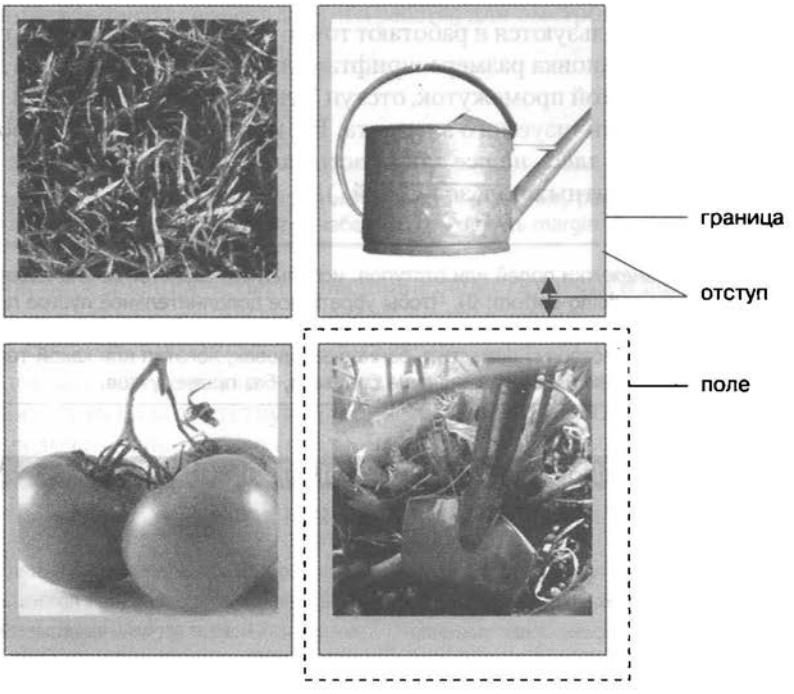


Рис. 7.2. Каждая фотография этой веб-страницы имеет поле размером 10 пикселов, то есть промежуток, отделяющий две соседние фотографии, составляет 20 пикселов

На рис. 7.2 благодаря отступам изображения отделены друг от друга и для них установлен серый цвет фона. Вы можете устанавливать границы, поля для каждой стороны изображения независимо друг от друга. Обратите внимание, что для нижних краев (оснований) фотографий установлены бóльшие по размеру отступы, чем для остальных.

Свойства `padding` и `margin` производят одинаковый визуальный эффект, и, пока вы не примените границу или цвет фона, вы не сможете сказать наверняка, каким свойством определен этот промежуток. Но если элемент имеет обрамляющую границу или цветной фон (подложку), вы заметите существенное отличие этих свойств. Отступ добавляет промежуток между содержимым и границей элемента и предотвращает эффект заключения содержимого элемента в рамку, в то время как поля добавляют так называемые межстолбцовые промежутки, которые придают веб-странице более «воздушный» внешний вид.

Вы можете управлять полями или отступами каждого отдельного элемента независимо. Четыре свойства управляют соответствующими полями с каждой стороны элемента: `margin-top`, `margin-right`, `margin-bottom` и `margin-left`. Аналогично с отступами: `padding-top`, `padding-right`, `padding-bottom` и `padding-left`. Вы можете использовать любые единицы измерения CSS для определения размеров полей и отступов, например:

```
margin-right: 20px;
padding-top: 3em;
margin-left: 10%;
```

Пикселы и em используются и работают точно так же, как при форматировании текста (см. разд. «Установка размера шрифта» гл. 6). Поле 20 пикселов добавляет соответствующий пустой промежуток, отступ 3 em — промежуток, в три раза больше размера шрифта стилизуемого элемента. Вы можете также использовать процентные значения, но здесь не все так просто (комментарии смотрите во врезке «Информация для опытных пользователей»).

СОВЕТ

Чтобы удалить все промежутки полей или отступов, используйте свойства со значением 0 (например, margin-top: 0 или padding-bottom: 0). Чтобы убрать все дополнительное пустое пространство с четырех сторон окна браузера, нужно присвоить свойствам margin и padding нулевые значения: margin: 0; padding 0;. Это позволит поместить баннер-заголовок, логотип или какой-то другой элемент веб-страницы вплотную у самого края окна браузера, без промежутков.

ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

Поля, отступы в процентах

При использовании в качестве единиц измерения процентов браузеры вычисляют размер промежутков полей и отступов на основе *ширины самого элемента-контейнера*, в который заключены форматируемые элементы. Рассмотрим самый простой случай веб-страницы, когда таким элементом-контейнером является `<body>`, который имеет ширину всего окна браузера. В данном случае значение в процентах в каждый конкретный момент времени вычисляется на основании текущей ширины окна. Допустим, оно составляет 760 пикселов. В этом случае левое поле, равное 10 %, добавит промежуток 76 пикселов с левого края стили-

ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

зуемого элемента. Но если вы измените размеры окна браузера, размер промежутка левого поля тоже изменится. Уменьшение до 600 пикселов изменит размер на 60 пикселов (10 % от 600 пикселов).

Однако элемент-контейнер не всегда равен ширине окна браузера. В последующих главах книги, когда мы будем создавать более сложный дизайн веб-страниц, вы увидите, что для разработки комплексного дизайна придется добавлять дополнительные элементы.

Возможно, вы захотите добавить в веб-страницу тег `<div>` для группировки, отделения содержимого бокового навигационного меню. Допустим, оно имеет ширину 300 пикселов. Тег `<div>` будет элементом-контейнером для всех остальных вложенных в него тегов. Таким образом, размер правого поля любого элемента, вложенного в `<div>` бокового меню и установленного в размере 10 %, будет равен 30 пикселам.

При установке процентных значений верхнего и нижнего полей элементов ситуация еще более запутанная: эти значения вычисляются на основании ширины элемента-контейнера, а не его высоты. Таким образом, 20%-ное верхнее поле составит 20 % от ширины стилизуемого тега.

Сокращенный набор свойств margin и padding

Нередко требуется одновременно установить одинаковые значения полей или отступов для всех четырех сторон стилизуемого элемента. Но последовательно набирать четыре различных свойства стиля (`margin-right`, `margin-left` и т. д.) утомительно и отнимает лишнее время. Не пугайтесь: здесь вы также можете использовать сокращенные варианты свойств `margin` и `padding` для быстрой установки всех четырех параметров одновременно:

```
margin: 0 10px 10px 20px;
padding: 10px 5px 5px 10px;
```

СОВЕТ

Если при описании стиля для CSS-свойства используется значение, равное 0, то совсем не нужно указывать единицу измерения. Например, наберите всего лишь margin: 0; вместо margin: 0px;.

Порядок определения четырех значений свойств `margin` и `padding` важен. Они должны идти в такой последовательности: *верхнее значение*, *правое значение*, *нижнее* и *левое*. Без этого у вас могут быть проблемы с форматированием. Самый легкий способ запомнить очередность — по буквам английского слова **TRouBLe** (проблемы, неприятности), соответствующим первым буквам английских слов, отражающих последовательность: **T**op (верх), **R**ight (право), **B**ottom (низ), **L**eft (лево).

Если вы хотите применить одинаковое значение свойства для всех четырех сторон, нет ничего проще, — используйте единственное значение. Чтобы удалить все поля из заголовка `<h1>`, напишите такой стиль:

```
h1 {
  margin: 0;
}
```

Кроме того, пользуйтесь кратким вариантом набора для добавления отступов — промежутков между содержимым и его границами:

```
padding: 10px;
```

ПРИМЕЧАНИЕ

Если нужно применить одинаковое значение свойства поля или отступа сверху и снизу элемента и одно и то же значение для левого и правого края, можете указывать два значения. Так, объявление margin: 0 2em; удаляет верхнее и нижнее поля, а левое и правое поля устанавливаются равными 2 em.

Конфликты полей

В CSS не всегда справедливы математические расчеты. Вы сами можете в этом убедиться, когда нижнее поле одного элемента веб-страницы касается верхнего поля другого элемента. Вместо того чтобы объединить эти поля вместе, браузер использует большее из них (рис. 7.3, *вверху*). Предположим, значение нижнего поля маркированного списка установлено равным 30 пикселам, а значение верхнего поля следующего за ним абзаца составляет 20 пикселов. Вместо того чтобы сложить два значения, получив общий промежуток размером 50 пикселов между списком и абзацем, браузер применяет *наибольшее из двух* значений — в данном случае 30 пикселов. Если вас это не устраивает, используйте вместо полей верхний или нижний отступ (см. рис. 7.3, *внизу*).

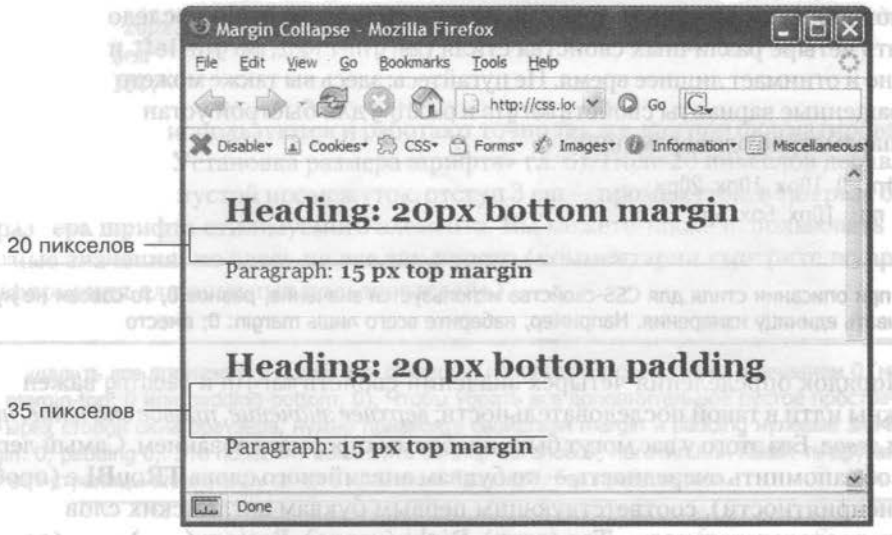


Рис. 7.3. В случае соприкосновения двух вертикальных полей меньшее из них игнорируется

На рис. 7.3, несмотря на то что и верхний заголовок имеет нижнее поле размером 20 пикселов, а у расположенного ниже абзаца текста верхнее составляет 15 пикселов, браузер добавляет промежуток между ними, равный всего 20 пикселам. Чтобы получить тот промежуток, который вы хотели (35 пикселов), используйте вместо полей отступы, как показано в нижней части рисунка. Здесь для заголовка установлен нижний отступ, равный 20 пикселам. Он складывается с верхним полем абзаца, равным 15 пикселам, и получается общий промежуток размером 35 пикселов.

Ситуация еще более усугубляется, когда один элемент веб-страницы *вложен* в другой. Это может привести к затиранию отдельных частей. Допустим, вы добавляете на веб-страницу «предупреждение» (заключенное в тег). Верхнее и нижнее поля устанавливаются размером 20 пикселов, чтобы отделить сообщение от заголовка сверху и от абзаца текста снизу. Пока все выглядит неплохо.

Но, предположим, вы вставляете заголовок в сообщение с предупреждением и, чтобы добавить небольшие промежутки между сообщением и верхним и нижним краем блока , устанавливаете для заголовка поле размером 10 пикселов. Вы, наверное, думаете, что добавили 10-пиксельный промежуток между заголовком и верхним и нижним краем блока , но вы не правы (рис. 7.4, *вверху*). Вместо этого поле появляется *над* блоком . В данном случае не имеет значения, какого размера поле применяется к заголовку, – поле все равно окажется над .

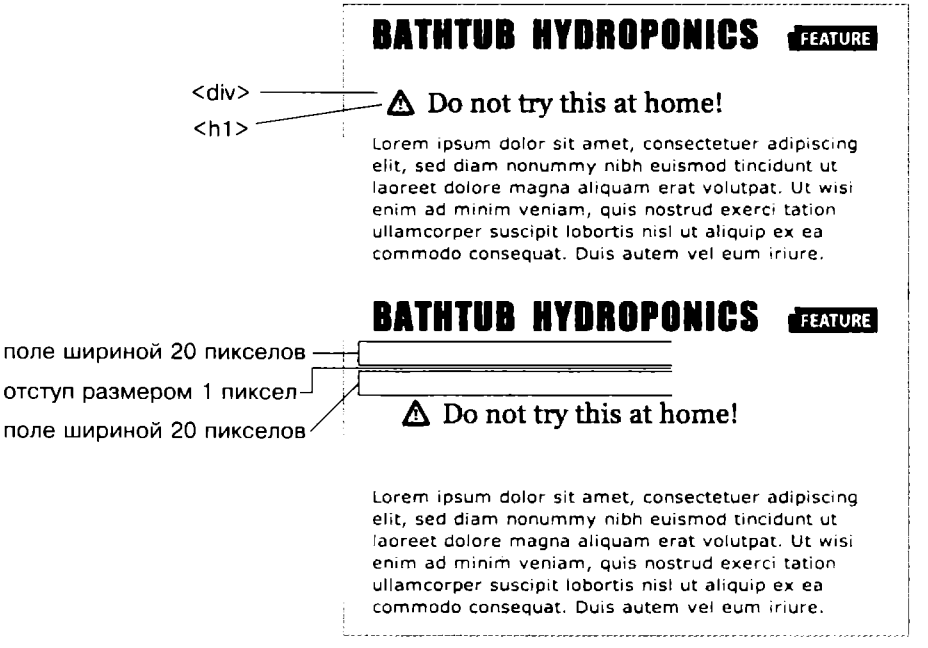


Рис. 7.4. Как бы ни соприкасались вертикальные поля, в любом случае произойдет конфликт

Для того чтобы решить эту проблему, добавьте небольшой отступ или границу вокруг элемента-контейнера (в данном случае 1 пиксел нижнего отступа для тега).

ПРИМЕЧАНИЕ

На профессиональном жаргоне CSS это явление называется «конфликтом полей». Оно означает, что два поля фактически превращаются в одно.

Есть два пути решения этой проблемы: добавить либо небольшой отступ, либо границу вокруг . Поскольку *между* этими двумя полями располагаются

граница и отступ, поля больше не соприкасаются и заголовок имеет небольшой отделяющий промежуток (см. рис. 7.4, *внизу*).

ПРИМЕЧАНИЕ

Горизонтальные (левые и правые) поля и поля между плавающими элементами не конфликтуют. Между абсолютно и относительно позиционируемыми элементами, о которых вы узнаете из гл. 13, также нет конфликта.

Удаление пустых полей с помощью отрицательных значений

Большинство значений свойств в CSS должны быть положительными. Что же произойдет, если указать для размера шрифта текста *отрицательное значение*, например минус 20 пикселов? Вообще, отступы должны иметь положительные значения. Однако CSS допускает использование отрицательных значений для создания определенных визуальных эффектов. Отрицательные поля вместо добавления пустого пространства между тегом и соседними элементами, наоборот, вызывают *удаление* этих промежутков. В результате может получиться абзац, накладывающийся на заголовок, выступающий из своего элемента-контейнера (бокового меню или другого элемента `<div>`) или даже совсем исчезающий за пределами окна браузера. Таким образом, можно с уверенностью сказать, что применение отрицательных значений полей дает немалую пользу.

Даже когда вы устанавливаете значения полей, равные 0, между двумя заголовками, все равно остается небольшой промежуток (благодаря межстрочному интервалу, как описано в подразделе «Установка межстрочного интервала» разд. «Форматирование абзацев текста» гл. 6). На самом деле это не так уж плохо, поскольку трудно читать предложения без дополнительных интервалов, сливающиеся друг с другом. Тем не менее разумное, умеренное использование небольших промежутков между заголовками текста поможет создать интересные визуальные эффекты. Второй заголовок на рис. 7.5 (который начинается со слов «Raise Tuna») имеет верхнее поле, равное 10 пикселам. Оно поднимает заголовок вверх, что обеспечивает небольшое наложение текста на пространство вышестоящего заголовка. Кроме того, левые и правые границы заголовка, начинающегося со слов «Extra! Extra!», теперь соприкасаются с буквами большего заголовка, создавая эффект единой надписи.

Можно также использовать отрицательные значения полей для того, чтобы имитировать отрицательный отступ. В третьем заголовке на рис. 7.5, который начинается со слов «The Extraordinary Technique», линия подчеркивания отображена прямо под текстом. Она на самом деле представляет собой *верхнюю* границу следующего абзаца (о том, как к тексту добавить границы, вы узнаете в следующем разделе). Но поскольку здесь определена верхняя граница с отрицательным значением, она располагается чуть выше текста абзаца и фактически находится под верхним заголовком. Обратите внимание на то, что хвост буквы Q заголовка буквально свисает под линией-границей. Поскольку отступ между содержимым (то есть Q) и границей не может быть отрицательным, вам не удастся поднять ниж-

нюю границу так, чтобы она находилась ближе к тексту или любому другому содержимому, не говоря уже о наложении. И все же есть возможность добиться этого эффекта, применив границу с отрицательным значением к последующему, нижестоящему элементу веб-страницы.



Рис. 7.5. Чтобы превратить верхнюю границу последнего абзаца в нижнюю вышестоящего заголовка, вернее, имитировать такой эффект, добавим небольшой отступ

СОВЕТ

Можете использовать либо верхнее поле абзаца с отрицательным значением, либо отрицательное нижнее поле заголовка. Оба варианта приведут к одному и тому же визуальному эффекту поднятия абзаца выше, ближе к тексту заголовка.

Отображение встроенных и блочных элементов

Хотя браузеры и обрабатывают любой тег веб-страницы подобно блочному элементу, на самом деле они не все одинаковы. В CSS есть два различных типа элементов: *блочные (прямоугольные)* и *встроенные (inline, поточные)*, которым соответствуют два одноименных типа тегов.

В *блочных* элементах создается разрыв строки перед тегом и после него. Например, тег `<p>` создает блок, отделенный от тегов, расположенных выше и ниже его. Другими примерами являются заголовки, теги `<div>`, таблицы и списки.

Встроенные (inline) элементы не создают разрывов строк ни до, ни после самих тегов. Они отображаются на одной строке с содержимым рядом стоящих тегов. Тег `<strong>` — встроенный. Слово, отформатированное с его помощью, будет расположено на одной строке с текстом, заключенным в другие встроенные теги, например `<em>`. Было бы очень странно, если бы одно слово в середине абзаца, выделенное `<strong>`, вдруг появилось на отдельной строке.

Другими примерами inline-тегов являются `<img>` — для добавления изображений, `<a>` — для создания ссылок, различные теги для полей форм и т. д.

В большинстве случаев CSS работает со встроенными и блочными элементами одинаково. Можно применять шрифты, цвет, фоновые параметры, границы к обоим типам элементов. Однако поля и отступы встроенных элементов браузеры обрабатывают по-другому. Если добавить поля или отступы слева или справа от встроенного элемента, то не удастся увеличить высоту встроенного элемента посредством установки верхнего или нижнего отступа или поля. В верхнем абзаце на рис. 7.6 к встроенному элементу применено форматирование, включающее границы, фоновый цвет и поля по 20 пикселов со всех четырех сторон. Но браузер добавляет пустые промежутки только с левой и правой стороны элемента.

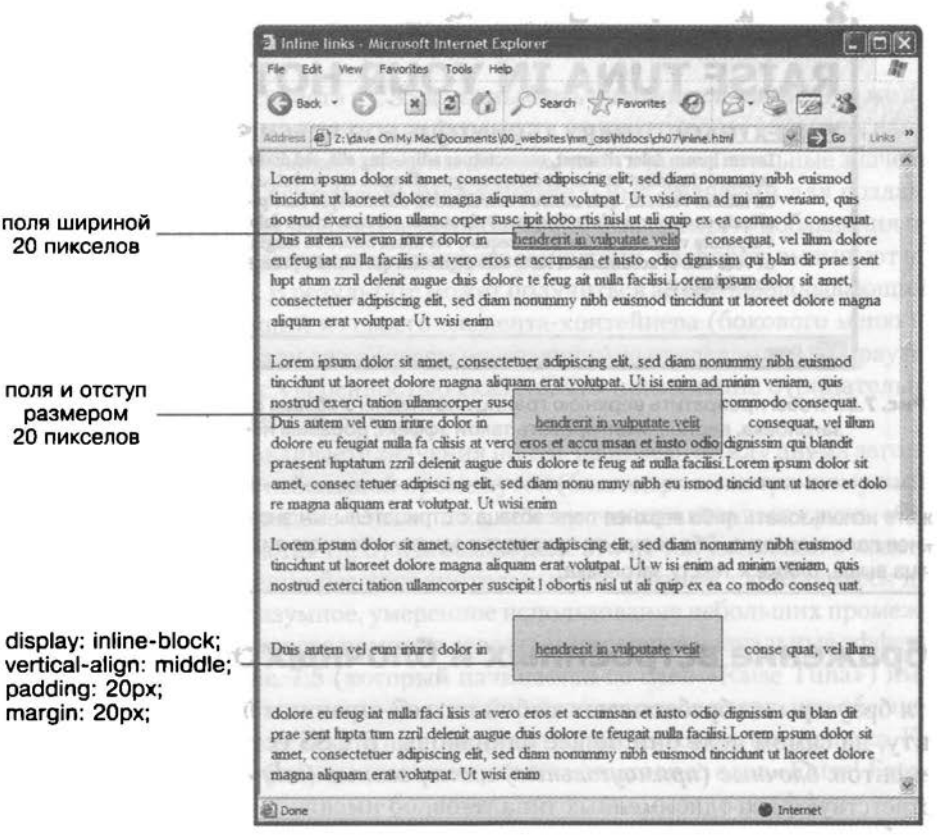


Рис. 7.6. Добавление к встроенному элементу верхнего, нижнего поля или отступа не изменит высоту элемента: форматирование будет не таким, как вы ожидаете

На рис. 7.6 в среднем абзаце фон и границы ссылки накладываются на текст, находящийся выше и ниже стилизуемого. Цвет фона встроенного элемента отображается поверх вышестоящей строки текста, но под следующей. Браузер обрабатывает каждую строку так, как будто она расположена в стеке, наверху по отношению к предыдущей. Как правило, это не представляет проблемы, так как строки текста обычно не накладываются. Если вы хотите, чтобы верхние и нижние поля работали для встроенного элемента, используйте инструкцию `display: inline-block` (см. рис. 7.6, *внизу*). Она оставит элемент встроенным, но он будет восприниматься как блочный,

поэтому отступы, поля, границы, ширина и высота будут к нему применяться. Это работает даже в таких браузерах, как Internet Explorer 6 и 7, но только для нормальных встроенных элементов, например ссылок, тегов , и . Кроме того, вам следует добавить инструкцию vertical-align:middle, чтобы IE 6 и 7 отображал встроенный блок таким же образом, как и остальные браузеры.

ПРИМЕЧАНИЕ

Есть одно исключение из этого правила: если поля или отступы применяются к встроенным элементам <img>, элементы не изменяют своей высоты. Браузеры корректно изменяют высоту контейнера элемента-изображения, чтобы подогнать ваши отступы и поля.

Иногда требуется, чтобы встроенные элементы вели себя так же, как блочные, или наоборот. Маркированные списки рассматривают элемент в виде отдельного блока, то есть каждый из списка располагается в стеке поверх следующего. Но что делать, если вы хотите изменить поведение пунктов списка таким образом, чтобы все они располагались рядом друг с другом, на одной строке? Или, возможно, вы захотите, чтобы встроенный элемент обрабатывался как блочный, например, изображение, встроенное в абзац текста, было расположено на отдельной строке, с верхним и нижним промежутками-интервалами.

К счастью, в CSS есть команда, которая позволяет вам это сделать, — это свойство display. С его помощью можно заставить блочный элемент работать как встроенный:

```
display: inline;
```

Или, наоборот, вы можете сделать так, чтобы встроенные элементы, например изображение или ссылка, вели себя как блочные:

```
display: block;
```

ПРИМЕЧАНИЕ

У свойства display есть большое количество параметров, многие из которых не работают во всех браузерах. Значение inline-block работает во всех современных браузерах (см. рис. 7.6). Другое значение — none — обрабатывается в большинстве браузеров и имеет множество вариантов использования. Это значение выполняет одну простую функцию — полностью скрывает стилизуемый элемент, чтобы он не отображался в окне браузера.

Используя программный код JavaScript, вы можете скрыть элементы, чтобы они стали видимыми после изменения значения свойства display на inline или block. Сделать их видимыми можно и средствами CSS: в книге вы увидите такой пример. И, наконец, несколько других значений для свойства display распознаются в IE 8, Firefox, Safari и Opera и предоставляют способ для создания основанной на CSS разметки. Эта усовершенствованная методика будет рассмотрена далее в книге.

Добавление границ

Граница представляет собой обычную линию, которая очерчивает стилизуемый элемент. Как показано на рис. 7.1, она располагается между отступом и полем элемента. С помощью границ, добавленных со всех сторон, можно заключить изображение в рамку или выделить баннер и т. д. Но совсем не обязательно применять границы, создающие очертания со всего содержимого элемента. Точно так же, как вы добавляете границы с четырех сторон элемента, можно добавить требуемую границу только к какой-либо комбинации. Эта гибкость обеспечивает добавление произвольных

элементов дизайна. Например, добавьте границу слева от элемента, придайте ей толщину около 1 em, и она станет похожа на маркер в виде квадрата. Единственная граница с нижнего края абзаца производит тот же визуальный эффект, что и элемент <hr> (горизонтальная линия), выступая разделителем частей веб-страницы.

Вы можете управлять тремя различными свойствами любой из границ: `color` (цвет), `width` (ширина, толщина) и `style` (стиль). Цвет `color` может быть представлен шестнадцатеричным числом, ключевым словом или значением в системе RGB, как и при форматировании текста (см. разд. «Стилизация текста» гл. 6). Ширина границы `width` — толщина линии, используемой для очерчивания. Вы можете использовать любые единицы измерения CSS (кроме процентов) или ключевые слова `thin` (тонкая линия), `medium` (средняя) и `thick` (толстая). Самые распространенные и понятные единицы измерения для данного свойства — пикселы.

И наконец, свойство `style` управляет типом линии границы. Существует множество различных стилей. Примеры приведены на рис. 7.7. Вы можете также определить стиль с помощью ключевых слов. Например, `solid` рисует сплошную линию, а `dashed` — ***штриховую (пунктирную)***. В CSS для границ имеются следующие стили: `solid`, `dotted`, `dashed`, `double`, `groove`, `ridge`, `inset`, `outset`, `none` и `hidden`. Ключевые слова `none` и `hidden` работают одинаково: они полностью удаляют границы. Но значение `none` удобно использовать для удаления границы с одной стороны элемента (см. разд. «Форматирование абзацев текста» гл. 6).

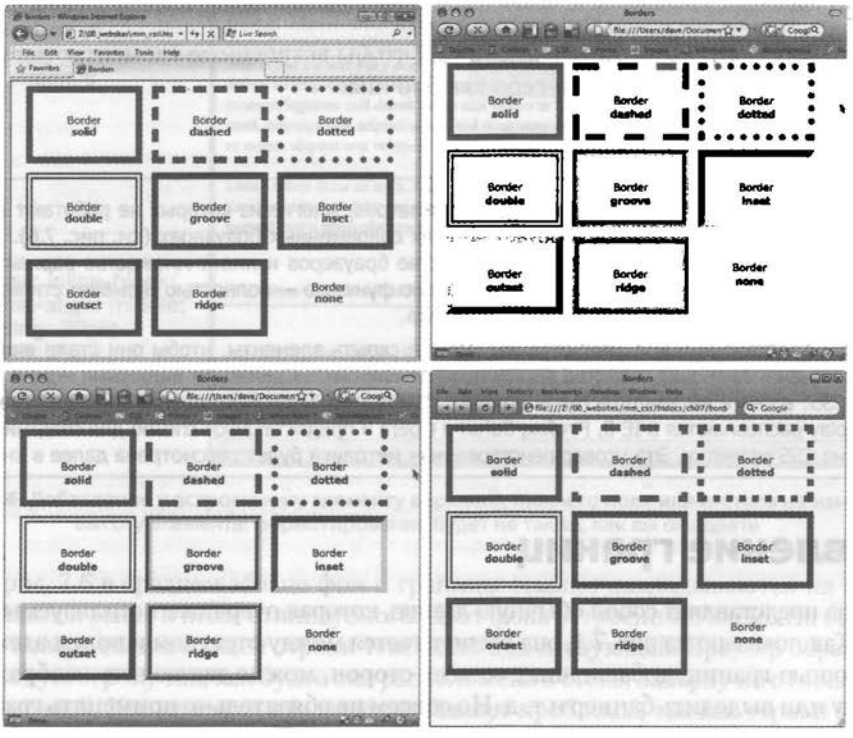


Рис. 7.7. Внешний вид стилей границ в следующих браузерах: Internet Explorer 8 *(вверху слева)*, Firefox 3 *(вверху справа)*, Opera 9 *(внизу слева)* и Safari 3 для Macintosh *(внизу справа)*. Internet Explorer 6 и 7 отображают границы так же, как и IE 8

ПРИМЕЧАНИЕ

В браузере Internet Explorer 6 для Windows и его более ранних версиях однопиксельная точечная граница выглядит точно так же, как однопиксельная пунктирная линия.

Сокращенный набор свойства border

Если вы посмотрите на полный список свойств, доступных в CSS, то подумаете, что границы действительно сложны. Вообще, есть 20 разновидностей свойств границ, с которыми вы столкнетесь в следующих разделах книги, а также несколько, относящихся к таблицам. Но все это — лишь варианты, обеспечивающие различные способы управления одними и теми же тремя свойствами: цветом, шириной (толщиной) и стилем для каждой из четырех границ. Наиболее простое и понятное свойство — `border`, которое просто добавляет границы с заданными параметрами:

```
border: 4px solid #F00;
```

Данный стиль создает сплошную красную границу с толщиной линии 4 пиксела. Вы можете использовать это свойство для создания простейшей рамки, окаймляющей изображение, панель навигации или любой другой элемент, которые надо выделить в отдельный блок.

ПРИМЕЧАНИЕ

Последовательность указания параметров свойства не имеет значения: border: 4px solid #F00; работает так же, как border: #F00 solid 4px;.

Форматирование отдельных границ

Вы можете управлять границей с каждой стороны элемента отдельно, используя соответствующее свойство: `border-top`, `border-bottom`, `border-left` или `border-right`. Они работают точно так же, как стандартное `border`, с тем исключением, что управляют границей только с одной стороны стилизуемого элемента. Добавить красную пунктирную линию снизу можно, используя следующее объявление свойства:

```
border-bottom: 2px dashed red;
```

Вы можете объединять общее свойство `border` со свойствами отдельных границ. Например, использовать `border-left`, чтобы определить основной, общий стиль, а затем выборочно настроить одну или несколько границ. Допустим, вы хотите, чтобы верхняя, левая и правая стороны абзаца имели одинаковый тип границы, а нижняя выглядела по-другому. Можно написать следующие четыре строки CSS-кода:

```
border-top: 2px solid black;
border-left: 2px solid black;
border-right: 2px solid black;
border-bottom: 4px dashed #333;
```

Такого же эффекта можно достигнуть всего двумя строками CSS-кода:

```
border: 2px solid black;
border-bottom: 4px dashed #333;
```

Первая строка кода определяет общий вид всех четырех границ, а вторая переопределяет вид нижней границы. Преимущество не только в том, что легче написать две строки CSS-кода вместо четырех, но и в том, что изменить стиль будет проще. Если вы захотите сделать цвет верхней, левой и правой границ красным, то необходимо отредактировать единственную строку кода вместо трех:

```
border: 2px solid red;
border-bottom: 4px dashed #333;
```

При использовании этого сокращенного метода установки границ определяется общий вид всех четырех границ. Затем вид одной из границ переопределяется с помощью свойства конкретной границы, например `border-left`. Очень важно, чтобы CSS-свойства были написаны в определенной последовательности. В общем случае глобальные установки границ должны быть на первом месте, а установки отдельной границы — на втором:

```
border: 2px solid black;
border-bottom: 4px dashed #333;
```

Поскольку свойство нижней границы `border-bottom` указано вторым, оно частично переопределяет общие установки свойства `border`. Если бы `border-bottom` было расположено первым, то `border` было бы отменено и все четыре границы стали бы одинаковыми. Последнее явное свойство может переопределить любые аналогичные свойства, определенные в CSS-коде выше. Это пример работы механизма каскадности CSS, который мы рассматривали в гл. 5.

Вы также можете использовать этот сокращенный метод установки границ, чтобы выключить отображение посредством ключевого слова `none`. Предположим, вы хотите установить границы только с трех сторон элемента (сверху, слева, снизу). Всего две строки кода обеспечат такое форматирование:

```
border: 2px inset #FFCC33;
border-right: none;
```

Возможность тонкой настройки границ каждой стороны стилизуемого элемента является причиной большого количества разновидностей свойств границ. Остальные 15 свойств позволяют определять индивидуальные цвета, стили, толщину линий границ для каждой стороны. Например, можно переписать определение `border: 2px double #FFCC33;` в следующем виде:

```
border-width: 2px;
border-style: double;
border-color: #FFCC33;
```

В этом варианте используются три строки кода вместо одной, поэтому вы, наверное, будете избегать такого способа. Однако каждая сторона имеет свой собственный набор из трех свойств, которые удобно использовать для отмены одного. Правая граница: `border-right-width`, `border-right-style` и `border-right-color`. Левая, верхняя и нижняя границы имеют похожие свойства: `border-left-width`, `border-left-style` и т. д.

Вы можете изменить ширину единственной стороны границы так: `border-right-width: 4px;`. При таком подходе хорошо то, что, когда вы позже решите изменить границу на сплошную, нужно будет отредактировать только групповое свойство границы, изменив `dashed` на `solid`.

Кроме того, вы можете задать собственные значения для каждой стороны границы, используя свойства `border-width`, `border-style` и `border-color`. Например, `border-width: 10px 5px 15px 13px;` применит четыре различных значения ширины для каждой из сторон (верхней, правой, нижней и левой).

Допустим, вы хотите установить все четыре границы элемента в виде пунктирной линии толщиной 2 пиксела, но при этом нужно, чтобы каждая граница имела свой цвет (возможно, вы создаете сайт для детей). Вот способ быстро сделать это:

```
border: 2px dashed green;
border-color: green yellow red blue;
```

Этот набор правил создает границы в виде двухпиксельной пунктирной линии со всех четырех сторон элемента, при этом верхняя граница будет иметь зеленый цвет, правая — желтый, нижняя — красный, а левая — синий.

СОВЕТ

Как правило, при использовании границ требуется добавлять отступы. Они обеспечивают промежутки между границами и содержимым элементов: текстом, изображениями, прочими тегами. Обычно границы отображаются слишком близко к содержимому элементов, только если вы не захотите разместить их вокруг изображения.

Установка цвета фона

В CSS имеются средства для добавления фона как для всей веб-страницы, так и для отдельного заголовка или любого другого элемента страницы. Используйте свойство `background-color` в сочетании с любым из действительных определений цветов, которые описаны в разд. «Стилизация текста» гл. 6. При желании вы можете окрасить фон веб-страницы в яркий зеленый цвет, указав следующий код:

```
body { background-color: #6DDA3F; }
```

Или можете создать стилевой класс, например, `.review` со свойством желаемого цвета фона, а затем применить его к тегу `<body>` HTML-кода таким образом: `<body class="review">`.

ПРИМЕЧАНИЕ

Вы можете также поместить изображение на заднем плане в качестве фона веб-страницы и управлять его размещением различными способами. Мы рассмотрим это в следующей главе.

Фоновый цвет удобно применять для создания множества различных визуальных эффектов. Вы можете придать заголовку контрастность, рельефность, установив темный цвет для фона и светлый — для текста. Цвет фона также является отличным средством для выделения таких обособленных частей веб-страницы, как панель навигации, баннер или боковое меню.

СОВЕТ

Когда вы пользуетесь одновременно фоновым цветом и границами, помните: если стиль границы — точечная или пунктирная линия (см. рис. 7.7), то фоновый цвет проступает в промежутках между точками или штрихами линий границ. Другими словами, браузеры размещают линию границ поверх цвета фона.

Определение параметров высоты и ширины

Рассмотрим еще два CSS-свойства, являющихся частью блочной модели CSS. Они предназначены для установки размеров объектов, таких как таблица, столбец, колонка, баннер, боковое меню. Свойства `height` и `width` назначают высоту и ширину области стилизуемого элемента. Мы будем часто пользоваться ими при создании разметки, макета веб-страниц, как описано в части 3. Они также применяются для разработки базового дизайна: назначения ширины таблиц, создания простейших боковых меню или галерей эскизов (примеры приведены в обучающем уроке 1 гл. 8).

Разработка стилей с этими свойствами не составляет сложностей. Просто наберите их со значением в любых единицах измерения CSS, которые мы изучили. Например:

```
width: 300px;
width: 30%;
height: 20em;
```

Пикселы как единицы измерения просты в использовании, понятны и удобны, обеспечивают точные ширину или высоту. Единица измерения em — это примерно то же самое, что и размер шрифта текста, но в условных единицах. Допустим, вы устанавливаете размер шрифта 24 пиксела; единица em для этого стилизуемого элемента будет равна 24 пикселам, а если вы установите ширину равной 2 em, она составит 2 × 24, или 48 пикселов. Если вы не определите в стиле размер шрифта текста, то он будет взят из унаследованных параметров (см. разд. «Установка размера шрифта» гл. 6).

Процентные значения свойства ширины `width` рассчитываются на основании ширины элемента-контейнера. Если вы установите ширину заголовка равной 75 % и этот заголовок не вложен ни в какие другие элементы веб-страницы с явно определенной шириной, то ширина текста заголовка составит 75 % от ширины окна браузера. Если посетитель изменит размер окна браузера, то ширина заголовка тоже изменится. Однако если заголовок заключен в блок `<div>` шириной 200 пикселов (возможно, для создания столбца), то ширина данного заголовка составит 150 пикселов. Процентные значения в свойстве высоты `height` работают точно так же, но расчет базируется на высоте элемента-контейнера, а не на его ширине.

БРИЛЛИАНТ БЕЗ ОГРАНКИ

Минимум и максимум

CSS также поддерживает еще несколько свойств, связанных с установкой высоты и ширины элементов веб-страниц: `min-height`, `min-width`, `max-height` и `max-width`. Эти свойства позволяют устанавливать минимальное и максимальное значения ширины или высоты элемента.

Свойства удобно использовать при гибкой разметке веб-страниц. Такой метод обеспечивает сохранение дизайна (макета и разметки) веб-страницы в нормальных размерах и пропорциях для очень маленьких или очень больших мониторов. К сожалению, браузер Internet Explorer 6 и его более ранние версии не понимают эти свойства.

Вычисление фактических размеров блочных элементов

Свойства `width` и `height` на первый взгляд кажутся довольно простыми и понятными, однако есть несколько нюансов, вводящих начинающих веб-дизайнеров в заблуждение. Прежде всего, существует различие между значениями ширины и высоты, которые вы явно указываете при написании стилей, и размером пространства, которое браузер фактически выделяет и использует для отображения элементов блочной модели CSS. Свойства `width` и `height` устанавливают ширину и высоту *области содержимого* стилизуемого элемента — пространства, в котором заключены текст, изображения или другие вложенные теги (см. рис. 7.1, чтобы вспомнить о том, где именно в блочной конструкции элементов CSS находится область содержимого). Фактическая ширина элемента веб-страницы представляет собой область экрана (окна браузера), выделяемую для отображения. Она состоит из ширины полей, границ, отступов и явно указанного значения ширины в свойстве стиля, как показано на рис. 7.8.

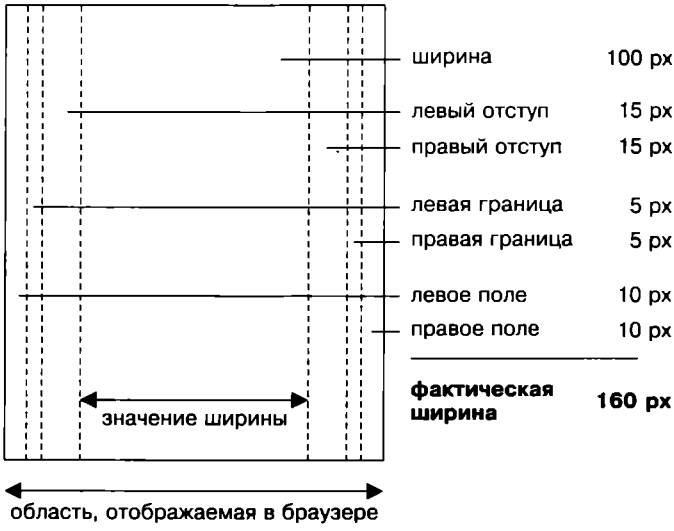


Рис. 7.8. Вычисление фактической ширины блока стилизуемого элемента выполняется путем сложения ее составляющих

Допустим, вы назначили следующие свойства:

```
margin: 10px;
border-width: 5px;
padding: 15px;
width: 100px;
```

Если определено свойство `width`, то вы всегда точно знаете, сколько места займет содержимое элемента — текст и изображения, заполняющие основное пространство элемента, — независимо от любых других установленных свойств. Вам не нужно

ничего вычислять, потому что значение width и есть размер основного пространства для размещения содержимого (в представленном выше примере ширина равна 100 пикселам). Конечно, придется выполнить несложные арифметические операции, чтобы выяснить общий точный размер. В представленном выше примере для размещения стилизуемого элемента браузером отводится пространство шириной 160 пикселов: 20 пикселов для левого и правого полей, 10 для левой и правой границ, 30 для левого и правого отступа и 100 пикселов в качестве ширины основного содержимого. Версии браузера Internet Explorer старше шестой неправильно работают с этими параметрами, поэтому для них требуется сделать кое-какие доработки веб-страницы.

Общее правило по регулированию высоты элементов на странице гласит: не делайте этого! Многие подающие надежды дизайнеры CSS пытаются задать высоту абсолютно для всего, желая получить полный контроль над пикселами. Но если вы не уверены полностью в точных размерах содержимого внутри тега, то можете столкнуться с некоторыми нежелательными результатами (рис. 7.9). В этом примере блок с цитатой, который используется для того, чтобы акцентировать внимание на интересном отрывке из статьи, имеет ширину и высоту по 100 пикселов. Когда в блок добавляется больше текста, чем может уместиться в такую высоту, его содержимое (во всех браузерах, кроме IE 6) выходит за пределы. Даже если вы уверены, что текст, который вы разместили в блоке с фиксированной высотой, соответствует его размерам, посетитель может увеличить размер шрифта в своем браузере, и высота текста, соответственно, может стать больше по сравнению с высотой блока.



Рис. 7.9. Отображение содержимого стилизуемого элемента в браузерах Internet Explorer 6 *(слева)* и Firefox *(справа)*

Другими словами, свойство height полезно для контроля высоты элемента , содержащего, например, изображения, потому что в таком случае вы можете правильно определить его высоту. Однако если вы используете это свойство для элементов, содержащих текст, не забудьте не только протестировать свои страницы

в основных браузерах, но и проверить их при различных установленных размерах шрифта, увеличивая его в браузерах.

ПРИМЕЧАНИЕ

Internet Explorer 5 обрабатывает неправильно все, что связано с шириной. Он использует CSS-свойство width для определения общей ширины, включая поля, отступы и границы, в результате чего элементы страницы получаются гораздо более тонкими по сравнению с аналогичными в других браузерах. Эта проблема также обнаруживается в IE 6 и 7 в причудливом режиме. Поскольку IE 5 уже давно не используется, вы, вероятно, не будете беспокоиться об этом. Но если у вас есть машина времени и вы планируете вернуться на 10 лет назад, чтобы поразить мир футуристическим дизайном CSS, ознакомьтесь с «проблемой блочной модели» в IE 5 на сайте http://reference.sitepoint.com/css/ie5boxmodel.

Управление поведением блочных элементов с помощью свойства overflow

Когда содержимое стилизуемого тега имеет размеры больше определенных свойствами `width` и `height`, происходят странные вещи. На рис. 7.9 показано, как браузер Internet Explorer 6 и его более ранние версии расширяют размеры блочного элемента таким образом, чтобы вместить это содержимое. В других же браузерах содержимое отображается за пределами (выступает наружу) границ элемента, часто накладываясь на него.

Браузер использует в этой ситуации свойство `overflow`. В качестве значения можно указать одно из четырех ключевых слов, определяющих, как должно отображаться содержимое, выходящее за пределы блочного элемента.

- `Visible` — это значение, принимаемое браузером по умолчанию. Указание этого ключевого слова имеет тот же эффект, что отсутствие установки свойства `overflow` (рис. 7.10, *вверху*).
- `Scroll` — позволяет добавить полосы прокрутки (рис. 7.10, *посередине*). Параметр создает своего рода окно мини-браузера внутри веб-страницы, которое выглядит подобно HTML-рамкам (фреймам). Вы можете использовать ключевое слово `scroll`, чтобы вместить объемное содержимое в ограниченной области. К сожалению, при таком варианте полосы прокрутки отображаются *всегда*, даже если содержимое по размерам помещается внутри блока.
- `Auto` — чтобы сделать полосы прокрутки необязательными, пользуйтесь данным значением. Оно выполняет ту же функцию, что и `scroll`, с одним исключением — полосы прокрутки добавляются только при необходимости.
- `Hidden` — скрывает любое содержимое, выходящее за пределы блочного элемента (рис. 7.10, *внизу*). Это значение небезопасно, поскольку может привести к тому, что часть содержимого будет не видна. Но иногда оно оказывается полезным для разрешения некоторых ошибок браузера Internet Explorer (см. разд. «Обработка ошибок в Internet Explorer 6» гл. 12), а также используется при создании плавающих разметок.



Рис. 7.10. Свойство overflow предоставляет три простых способа отображения текста, размеры которого не позволяют браузеру отобразить его внутри блочного элемента

ОШИБКИ БРАУЗЕРОВ

Специальные рекомендации по обеспечению совместимости с браузером Internet Explorer 6

То, что различные браузеры могут отобразить одну и ту же веб-страницу по-разному и самыми удивительными способами, портит жизнь любому веб-дизайнеру. Страница, которая выглядит великолепно в Internet Explorer, может иметь совершенно другой вид или вообще не отображаться в Firefox, и наоборот. В книге вы найдете множество подсказок, хитростей, обходных путей по управлению браузером. Кроме того, будут описаны способы преодоления возникающих трудностей, устранения распространенных ошибок. Неудивительно, что браузер Internet Explorer 6, который в ходу уже более половины десятилетия, имеет множество слабых мест, проявляющихся при отображении веб-страниц. Так, в нем есть проблемы с отображением плавающих разметок, о чем вы узнаете в гл. 12.

Чтобы исключить эти ошибки, часто придется адресовать свойства и их значения для применения исключительно в Internet Explorer, поскольку для получения того же эффекта в других браузерах нужно использовать другие значения, а порой и свойства. Специально для этой цели существует простой способ создания стилей CSS, предназначенных для применения определенным классом браузеров: Internet Explorer 6 и его ранними версиями. Это использование конструкции `* html`. При этом способе стиль должен начинаться с `* html`. Если вы хотите создать стиль тега `<h1>`, ориентированный на применение только Internet Explorer 6 и его ранними версиями, назовите его `* html h1`. Все остальные браузеры будут рассматривать его как производный селектор, не имеющий никакого смысла. Они просто проигнорируют его.

Мы будем часто использовать `* html`, чтобы переопределить или избирательно изменить некоторые параметры для стилей, правильно отображающихся браузерами, отличными от Internet Explorer. В этом случае нужно применить конструкцию `* html` после правильного стиля, скорректировав параметр исключительно для Internet Explorer. Допустим, вы создаете стилевой класс `.sidebar`, который образует симпатичный блок бокового меню для новостей и ссылок навигации сайта. Из-за ошибки в Internet Explorer боковое меню может отображаться со смещением на три пиксела влево или вправо (см. разд. «Обработка ошибок в Internet Explorer 6» гл. 12). Чтобы исключить эту недоработку, вы можете добавить после правильного стиля `.sidebar` специально для Internet Explorer следующую доработку:

```
* html .sidebar { margin-left: -3px }
```

С конструкцией `* html` в этой книге вы встретитесь неоднократно (например, в обучающем уроке данной главы). Другие методики управления браузером Internet Explorer мы изучим в гл. 15.

Браузеры Internet Explorer 7 и 8 не понимают конструкции `* html` и, значит, просто проигнорируют эти определения стилей. К счастью, в этих браузерах исправлены многие ошибки, встречавшиеся в более ранних версиях, поэтому для них использование `* html` неактуально. Несмотря на это, в данной книге вы узнаете о методе, который позволит передавать специфичные стили любой версии Internet Explorer.

Управление обтеканием содержимого плавающих элементов

HTML в обычном порядке отображается, начиная с верхнего края окна браузера по направлению к нижнему: один заголовок, абзац, элемент блочной структуры поверх другого. Такой способ представления неинтересен, но благодаря CSS предоставляется возможность изменить дизайн в лучшую сторону. В части 3 мы рассмотрим много новых методов компоновки, размещения, упорядочения элементов, но уже сейчас вы можете добавить привлекательности своим веб-страницам посредством единственного CSS-свойства `float`.

Свойство `float` перемещает любой элемент в нужное место, выравнивая его по левому или правому краю веб-страницы. В процессе перемещения содержимое, находящееся ниже позиционируемого плавающего элемента, смещается вверх и плавно обтекает сам плавающий элемент (рис. 7.11, *внизу*). Плавающие (или перемещаемые) элементы — идеальное средство для того, чтобы выделить дополнительную информацию из основного текста. Изображения могут смещаться к любому краю веб-страницы, обеспечивая перенос рядом стоящего текстового содержимого и его изящное обтекание вокруг изображений. Точно так же вы можете передвинуть боковое меню с относящейся к тексту информацией и управляющие ссылки к одной из сторон веб-страницы.

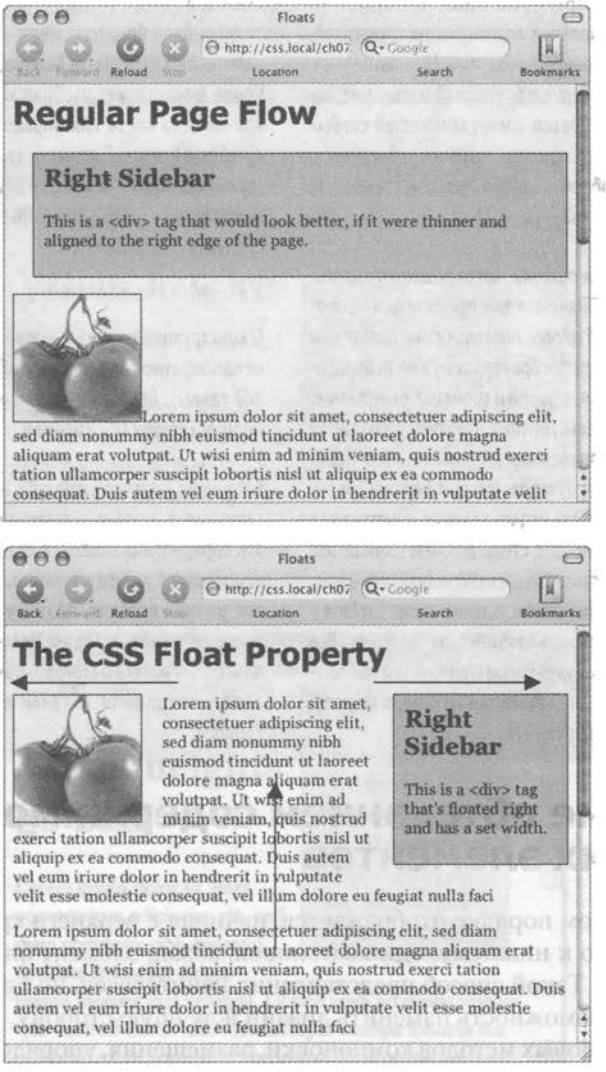


Рис. 7.11. Стандартная последовательность отображения HTML *(вверху)* и вид страницы при использовании свойства float *(внизу)*

Позволяя избавиться от однообразия, свойство float является одним из самых мощных и полезных средств, предлагаемых языком CSS. Широкий диапазон его применения включает не только простое позиционирование, перемещение изображений к одной из сторон абзаца, но и обеспечивает полный контроль по управлению размещением баннеров, меню, панелей навигации и других элементов веб-страниц.

Несмотря на то что свойство float может быть использовано для сложного, комплексного форматирования (об этом вы прочитаете в гл. 12), применение этого простого атрибута не представляет никаких сложностей. В качестве значения указывается одно из трех ключевых слов — left, right или none:

- left — перемещает стилизуемый элемент влево, при этом содержимое, находящееся ниже его, обтекает правый край элемента;
- right — перемещает элемент вправо;
- none — отменяет обтекание и возвращает объект в его обычную позицию.

Например:

```
float: left;
```

ПРИМЕЧАНИЕ

Позиционирование изображений посредством float подобно применению свойства выравнивания по левому или правому краю <img>. Способ выравнивания средствами HTML немного устарел, так что лучше пользуйтесь CSS-свойством float.

Плавающие элементы могут быть расположены у левого или правого края окна содержащего их элемента-контейнера. В некоторых случаях это просто означает, что элемент перемещается к левому или правому краю окна браузера. Однако если вы перемещаете элемент, находящийся внутри другого тега, для которого установлены значения ширины или позиции на веб-странице, то перемещение будет осуществлено к левому или правому краю этого тега, который является контейнером для плавающего элемента. Например, на веб-странице может быть блочный элемент шириной 300 пикселов, который прилегает к правому краю окна браузера. Внутри может располагаться изображение, которое выровнено по левому краю. Иными словами, изображение примыкает к краю этого блока шириной 300 пикселов, а не к окну браузера.

Вы можете применять свойство float к встроенным элементам, например <img>. Надо сказать, использование выравнивания по левому или правому краю для фотографий — самый обычный, даже наиболее распространенный способ применения float. Браузер обрабатывает встроенные элементы точно так же, как блочные. Поэтому, используя отступы и поля во встроенных элементах, вы избежите проблем, которые обычно возникают в такой ситуации.

Вы можете также использовать float для блочных элементов, таких как заголовок или абзац текста. Общая методика применения свойства к тегу , содержащему другие HTML-теги, заключается в создании своего рода блока-контейнера. Таким способом вы можете создавать боковые меню, плавающие цитаты и другие обособленные элементы веб-страниц (пример приведен в обучающем уроке этой главы). При использовании свойства float для блочных элементов рекомендуется установить свойство width для них (на самом деле правила CSS требуют установки ширины плавающим элементам для всех тегов, кроме изображений). Таким образом вы можете управлять размером блока по горизонтали, оставив промежуток для

текстового содержимого, расположенного после этого блочного элемента, и, соответственно, задать его поведение — перенос, обтекание.

ПРИМЕЧАНИЕ

Порядок написания HTML-кода оказывает большое влияние на отображение плавающих элементов веб-страницы. HTML-код должен находиться до любого HTML-содержимого, которое обтекает плавающий элемент.

Допустим, вы создали веб-страницу, в которой имеется тег заголовка <h1>, а за ним идет тег абзаца <p>. Внутри <p> вы вставили фотографию с помощью тега <img>. Теперь, если вы установите выравнивание фотографии по правому краю, заголовок <h1> и часть содержимого абзаца <p> будут по-прежнему отображены над фотографией. Только содержимое, следующее за тегом <img>, будет обтекать изображение с левой стороны.

Фон, границы и плавающие элементы

К недовольству многих веб-дизайнеров, фон и границы не переносятся таким же образом, как другие элементы веб-страницы. Допустим, у вас есть боковое меню, примыкающее к правому краю веб-страницы. Содержимое страницы, расположенное под ним, как ему и положено, приподнимается выше и обтекает меню.

Однако если у этого содержимого есть фоновые параметры или границы, они отображаются, фактически проступая под боковым меню (рис. 7.12, *слева*). По сути, браузер окружает плавающий элемент только текстом, но не границами или фоном. Как это ни удивительно, но именно так работает механизм обтекания. Возможно, вы не будете следовать этим правилам, а захотите, чтобы границы, фоновый цвет или рисунок не отображались при достижении плавающего элемента (см. рис. 7.12, *справа*). С помощью CSS вы можете сделать и это.



Рис. 7.12. Добавление свойства overflow: hidden; в стиль тега <h1> препятствует тому, чтобы атрибуты заголовка повлияли на отображение фона и границ под плавающим элементом

Первый метод заключается в добавлении еще одного свойства в стиль, устанавливающий параметры фона и границ и оказывающий воздействие на плавающий элемент. Нужно добавить в данный стиль следующий код: `overflow: hidden;`. Свойство `overflow` (описано в предыдущем разделе) отменяет отображение продолжающихся фона или границ под плавающими элементами.

ПРИМЕЧАНИЕ

Этот метод работает не во всех браузерах. Смотрите врезку ниже.

Другой подход состоит в добавлении границ вокруг плавающего элемента. Если вы создаете достаточно толстую границу цвета фона страницы, то не будете даже догадываться о существовании лишних границ, находящихся под плавающим элементом, так как они располагаются поверх и скрывают аналогичные лишние атрибуты.

Отмена правил обтекания

Иногда требуется отобразить содержимое тега таким образом, чтобы на него не влияла способность обтекания плавающего элемента. Веб-страница может содержать примечание с текстом авторских прав, которое в любом случае должно отображаться в самой нижней части окна браузера. Если у вас имеется *высокое* боковое меню, примыкающее к левому краю веб-страницы, то примечание с авторскими правами может подняться вверх и отобразиться с обтеканием вокруг плавающего элемента. Таким образом, вместо того, чтобы быть внизу страницы, текст появится гораздо выше, на одном уровне с боковым меню. Вам же нужно, чтобы для примечания с авторскими правами было отменено обтекание, установленное свойством плавающего бокового меню.

Сложности другого рода возникают при наличии нескольких плавающих элементов, расположенных рядом. Когда они имеют небольшую ширину, то поднимаются вверх, располагаясь на одном уровне. В противном случае элементы могут образовать некрасивую «пустоту» (рис. 7.13, *вверху*). В данном случае плавающие элементы не должны отображаться рядом друг с другом. Для решения этой проблемы в CSS предусмотрено свойство `clear`.

ОШИБКИ БРАУЗЕРА

Когда overflow: hidden не работает

Свойство `overflow: hidden` предотвращает отображение фона или границ под плавающими элементами (см. рис. 7.13). Но в мире браузеров не все так просто. Какая-то строка кода может работать в Internet Explorer 7, Firefox, Camino и Safari, но ее работа не гарантируется в Opera (до версии 9), а в Internet Explorer 5 и 6 она может просто игнорироваться.

Увы, но для Opera версии 8 и ниже (в версии 9 все работает хорошо) нет конкретного решения этой проблемы, тем не менее в Internet Explorer 5 и 6 она решаема. Для этого нужно добавить одно дополнительное свойство: `zoom: 1;`.

Это свойство, придуманное Microsoft (и работающее, естественно, только в Internet Explorer), позволяет вам увеличивать размеры элементов веб-страницы. Однако это всего лишь способ избежать отображения фона и границ под плавающими элементами в Internet Explorer 5 и 6 (о том, зачем изменять масштаб, а также более подробное описание этого метода читайте во врезке «В помощь продвинутому пользователю» в разд. «Обработка ошибок в Internet Explorer 6» гл. 12).

Возможно, вы захотите поместить специфическое свойство для Internet Explorer в отдельную таблицу стилей. Вы также можете вставить его в отдельную внешнюю таблицу стилей, ориентированную конкретно для Internet Explorer.

Пример решения данной проблемы приведен в обучающем уроке этой главы.

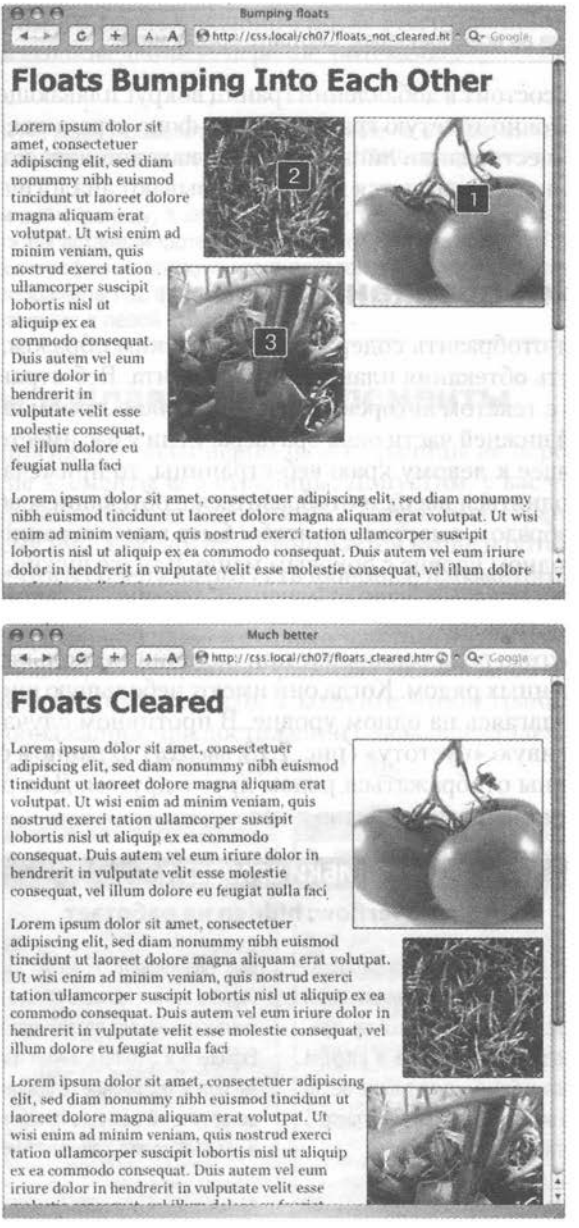


Рис. 7.13. Если не хотите, чтобы элемент обтекал плавающий объект *(вверху)*, вам поможет свойство clear *(внизу)*, предотвращающее размещение изображений на одном уровне

Свойство `clear` говорит браузеру о том, что стилизуемый элемент не должен обтекать плавающий элемент. Очищение параметра приводит к тому, что он смещается вниз по тексту веб-страницы, располагаясь ниже плавающего элемента. Кроме того, вы можете управлять очисткой с любой стороны элемента (слева или справа) или просто полностью удалить все параметры обтекания.

Свойство clear допускает следующие значения:

- left — стилизуемый элемент смещается вниз относительно плавающего с установленным левым обтеканием, но правое обтекание остается в силе;
- right — стилизуемый элемент смещается вниз относительно плавающего с установленным правым обтеканием, параметр левого обтекания остается в силе;
- both — вызывает перемещение стилизуемого элемента вниз относительно плавающего с установленными левым и правым обтеканием;
- none — полностью отменяет очистку обтекания — говорит браузеру, что стилизуемый элемент должен вести себя стандартным образом, как предусмотрено, то есть он может обтекать плавающие элементы как с левой, так и с правой стороны.

В нашем примере текст примечания с авторским правом должен всегда отображаться внизу веб-страницы и не должен обтекать другое содержимое. Ниже приведен стилевой класс, который выполняет это:

```
.copyright { clear: both; }
```

На рис. 7.13 показано, как свойство clear может препятствовать беспорядочному отображению плавающих элементов различной высоты. Для всех трех представленных на рисунке фотографий установлено обтекание по правому краю. На верхней части рисунка фото помидоров *(1)* — первое изображение на веб-странице, оно расположено в верхнем правом углу. На отображение второй фотографии *(2)* оказывает влияние параметр обтекания первого изображения, и оно обтекает слева первое фото. Последнее изображение *(3)* слишком широкое, чтобы отобразиться рядом (на одном уровне) со вторым *(2)*, но все-таки оно пытается расположиться слева по отношению как к первому *(1)*, так и ко второму *(2)* изображению. Однако ширина рисунка *(3)* не позволяет этого сделать.

Применение здесь свойства clear: right; препятствует размещению фотографий на одном уровне (см. рис. 7.13, *внизу*). Очистка обтекания для второй фотографии не допускает ее отображения рядом с первой, а очистка обтекания для третьей предотвращает ее отображение на одном уровне со второй.

ПРИМЕЧАНИЕ

Применение параметров левого и правого обтеканий, а также очистка обтекания кажется достаточно сложным механизмом. Вообще, так оно и есть. В этом разделе я привел вам основные понятия. Мы снова вернемся к этой теме в гл. 12, там вы познакомитесь с использованием обтекания в более сложных вариантах.

Обучающий урок: поля, фоновые параметры, границы

В этом обучающем разделе мы исследуем элементы блочной модели CSS, потренируемся в настройке параметров интервалов, промежутков объектов веб-страниц, применим к элементам красочные границы-рамки, поуправляем размерами и обтеканием элементов веб-страниц.

Чтобы начать обучающий урок, вы должны загрузить файлы, содержащие учебный материал. Как это сделать, рассказано в конце гл. 2. Файлы текущей обучающей программы находятся в папке с названием 07.

Управление полями страницы

Начнем с простого HTML-файла, содержащего внутреннюю таблицу стилей со сбросом стандартных настроек стилей CSS. Пока на странице нет ничего интересного (рис. 7.14).

СОВЕТ

Увидеть конечный результат данного обучающего урока можно на рис. 7.17.

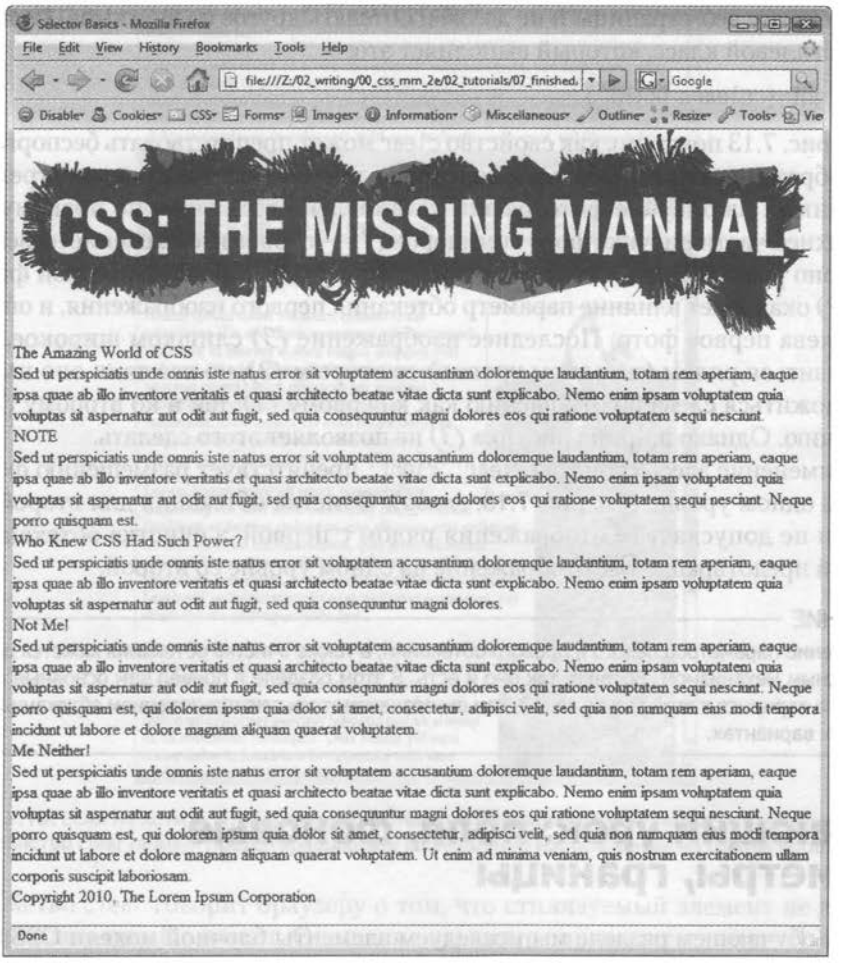


Рис. 7.14. Эта страница представляет собой основу HTML и содержит единственный стиль, устраняющий множество встроенных настроек форматирования браузера. Она будет выглядеть намного лучше после изменения с использованием блочной модели

1. Откройте в HTML-редакторе страницу `sidebar.html` из папки **07**.

   Здесь уже есть внутренняя таблица стилей, содержащая один групповой селектор. Этот селектор является самым важным стилем в таблице стилей, устраняющей стандартные настройки CSS. Он в основном удаляет все поля, отступы, установленные значения для размера шрифта, начертания шрифта из наиболее значимых элементов уровня блока и устраняет множество кроссбраузерных проблем с отображением элементов, с которыми вы могли столкнуться из-за этих свойств.

   Как минимум вы должны всегда включать этот стиль во все создаваемые таблицы стилей. Наиболее важными свойствами в нем являются `margin` и `padding`. Существует достаточно кроссбраузерных странностей, связанных с этими двумя свойствами, поэтому вы должны всегда обнулять их и начинать с чистого листа. Сначала мы определим что-нибудь простое, например цвет фона.

2. Во внутренней таблице стилей щелкните кнопкой мыши сразу за комментарием CSS `/* end reset styles */` и добавьте стиль селектора тега:

   ```
   html {
     background-color: #FDF8AB;
   }
   ```

   Этот стиль устанавливает светло-желтый фон для страницы. Если вы хотите задать цвет для фона веб-страницы, можно добавить свойство `background-color` либо к тегу `<html>`, либо к тегу `<body>`. Далее мы добавим поля, границы и другие свойства тега `<body>`.

3. Добавьте другой стиль во внутренней таблице стилей.

   ```
   body {
     background-color: #FFF;
     border: 3px solid #85A110;
   }
   ```

   Этот стиль добавляет белый цвет фона для тега `<body>` и зеленую трехпиксельную границу. Поскольку тег `<body>` расположен внутри тега `<html>`, браузер считает, что он находится «поверх» тега `<html>` и белый фон покроет фон желтого цвета, установленный в предыдущем шаге. Далее мы зададим ширину для тега тела и настроим его отступы и поля.

СОВЕТ

Обычно, если вы добавляете свойство background-color к тегу <body>, цвет заполняет все окно браузера. Однако если вы также добавите цвет фона для тега <html>, фон <body> будет заполнять только область с содержимым. Чтобы увидеть это в действии, просмотрите веб-страницу после шага 3, затем удалите стиль для тега <html> и просмотрите страницу снова. Странная, но полезная мелочь CSS.

4. Измените стиль тела (`<body>`), который вы только что создали, добавив пять новых свойств (изменения выделены полужирным шрифтом):

   ```
   body {
     background-color: #FFF;
     border: 3px solid #85A110;
   ```

```
  width: 760px;
  margin-top: 20px;
  margin-left: auto;
  margin-right: auto;
  padding: 15px;
}
```

Свойство `width` ограничивает тело страницы 760 пикселами по ширине: если окно браузера посетителя окажется шире, чем 760 пикселов, он увидит фоновый цвет стиля `<html>` и блок шириной 760 пикселов с белым фоном тега `<body>`.

Свойство `margin-top` добавляет 20 пикселов пространства от верхнего края окна браузера, отталкивая содержимое тега `<body>` чуть-чуть вниз, а левое и правое поля центрируют его, размещая посередине окна браузера. Значение `auto` — это просто еще один способ сказать браузеру: «Разбирайся в этом сам», и, поскольку данное значение применяется как к левому, так и к правому полям, браузер создает одинаковые промежутки слева и справа.

ПРИМЕЧАНИЕ

Вы можете также использовать сокращенное свойство margin, чтобы сжать эти три строки, устанавливающие настройки для полей, в одну:

```
margin: 20px auto 0 auto;
```

Наконец, для того чтобы предотвратить прикосновение содержимого тега `<body>` к линии границы, к внутренней части содержимого с помощью свойства `padding` добавлен отступ шириной 15 пикселов. Другими словами, для изображения и текста был задан отступ 15 пикселов от всех четырех краев.

В вашей таблице стилей уже много всего, поэтому пришло время проверить, как выглядит страница.

5. Сохраните файл и просмотрите веб-страницу в браузере.

   Вы должны увидеть белый блок с изображением, кусок текста и зеленый контур, плавающие на желтом фоне (рис. 7.15). Теперь следует уделить внимание тексту, чем мы и займемся далее.

Настройка интервалов между тегами

Поскольку стили, сбрасывающие стандартные настройки CSS, «оголили» текст на странице, вам необходимо создать новые стили, которые преобразили бы заголовки и абзацы. Начнем с тега `<h1>` наверху страницы.

1. Вернитесь к файлу `sidebar.html` в HTML-редакторе. Переведите курсор за закрывающую скобку селектора `body`, нажмите **Enter** и добавьте следующий стиль:

```
h1 {
  font-size: 44px;
  font-family: Georgia, "Times New Roman", Times, serif;
  letter-spacing: 1px;
  color: #85A110;
  text-transform: uppercase;
}
```

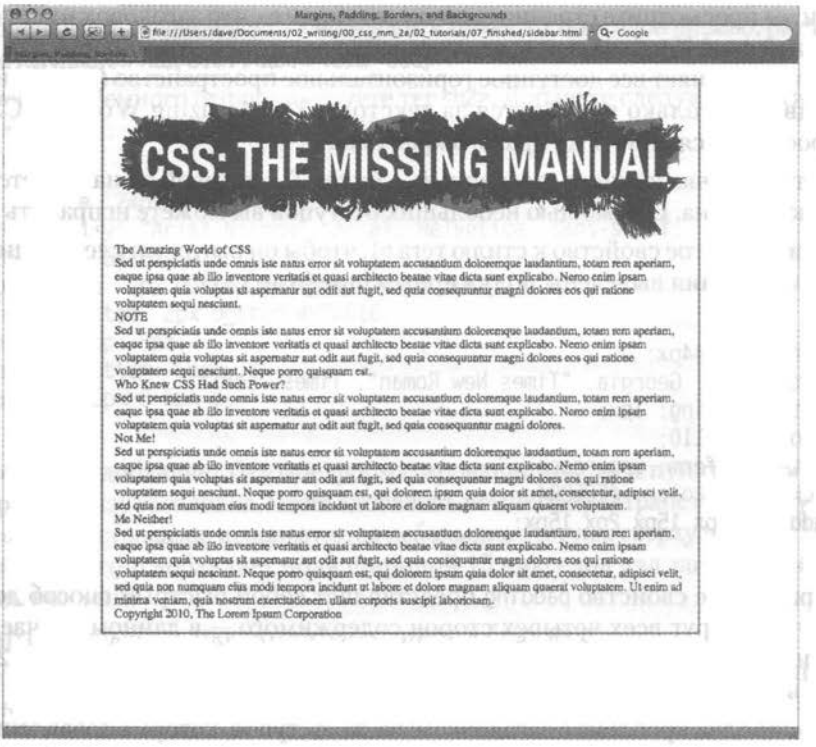


Рис. 7.15. Устанавливая значение auto для левого и правого полей любого элемента, имеющего заданную ширину, вы центрируете этот элемент

Этот стиль использует множество свойств для форматирования текста, которые мы обсуждали в предыдущей главе. Верхний заголовок имеет высоту 44 пиксела и набран прописными буквами. К нему применяется шрифт Georgia зеленого цвета, есть небольшой промежуток между буквами. Очень интересным получается добавление фонового цвета для выделения заголовка.

СОВЕТ

Сохраняйте страницу и просматривайте ее в браузере после выполнения каждого шага из этого примера. Таким образом, вы получите хорошее понимание того, как приведенные здесь свойства CSS воздействуют на элементы, которые они форматируют.

2. Добавьте одно новое свойство к стилевому тегу h1, чтобы он выглядел следующим образом (изменения выделены полужирным шрифтом):

```
h1 {
  font-size: 44px;
  font-family: Georgia, "Times New Roman", Times, serif;
  letter-spacing: 1px;
  color: #85A110;
  text-transform: uppercase;
  background-color: #E2EBB4;
}
```

Если вы просмотрите страницу сейчас, то увидите, что заголовок имеет светло-зеленый фон. Когда фон применяется к блочным элементам, таким как заголовок, он заполняет все доступное горизонтальное пространство (другими словами, цвет не только появляется за текстом «The Amazing World of CSS», но и простирается вплоть до правого края окна).

Текст заголовка немного стесненный — буква T, с которой он начинается, касается края фона. С помощью небольших отступов вы можете исправить это.

3. Добавьте другое свойство к стилю тега h1, чтобы он выглядел следующим образом (изменения выделены полужирным шрифтом):

```
h1 {
  font-size: 44px;
  font-family: Georgia, "Times New Roman", Times, serif;
  letter-spacing: 1px;
  color: #85A110;
  text-transform: uppercase;
  background-color: #E2EBB4;
  padding: 5px 15px 2px 15px;
}
```

Сокращенное свойство padding представляет собой быстрый способ добавить отступы вокруг всех четырех сторон содержимого — в данном случае отступ шириной 5 пикселов добавляется над текстом, 15 пикселов — справа, 2 пиксела — внизу и 15 пикселов — слева.

Есть еще одна проблема с заголовком: из-за отступов, которые добавлены к тегу <body> (см. шаг 4 предыдущего задания), заголовок отодвинется на 15 пикселов от левого и правого краев зеленого контура, окружающего содержимое. Заголовок станет выглядеть лучше, если будет касаться этого контура. Это не проблема: на помощь приходят отрицательные поля.

4. Добавьте последнее свойство к стилю тега h1, чтобы он выглядел следующим образом (изменения выделены полужирным шрифтом):

```
h1 {
  font-size: 44px;
  font-family: Georgia, "Times New Roman", Times, serif;
  letter-spacing: 1px;
  color: #85A110;
  text-transform: uppercase;
  background-color: #E2EBB4;
  padding: 5px 15px 2px 15px;
  margin: 0 -15px 20px -15px;
}
```

Здесь сокращенное свойство margin устанавливает 0 пикселов для верхнего поля, −15 пикселов для правого, 20 пикселов для нижнего и −15 пикселов для левого. Нижнее поле просто добавляет немного пространства между заголовком и абзацем, который следует за ним. Следующий прием заключается в использовании отрицательных значений для левого и правого полей. Как отмечалось ранее, у нас есть возможность назначить отрицательные поля для какого-либо элемента. Это свойство «тянет» элемент по направлению поля — в данном случае

заголовок продлевается на 15 пикселов влево и 15 пикселов вправо, расширяясь и вытягиваясь над отступами тега <body>.

5. Теперь вы немного отформатируете тег <h2>. Добавьте следующий стиль после стиля для h1:

```
h2 {
  font-size: 24px ;
  font-family: "Arial Narrow", Arial, Helvetica, sans-serif;
  color: #F96B18;
  border-top: 2px dotted #8DA516;
  border-bottom: 2px dotted #8DA516;
  padding-top: 5px;
  padding-bottom: 5px;
  margin: 15px 0 5px 0;
}
```

Этот стиль добавляет базовое форматирование текста и пунктирные границы сверху и снизу заголовка. Чтобы добавить немного пространства между текстом заголовка и строками, используются небольшие отступы сверху и снизу. Наконец, свойство margin добавляет поля шириной 15 пикселов над заголовком и 5 пикселов под ним.

6. Сохраните файл и просмотрите страницу в браузере.

Заголовки выглядят хорошо (рис. 7.16). Далее вы создадите боковую панель на правой стороне страницы.

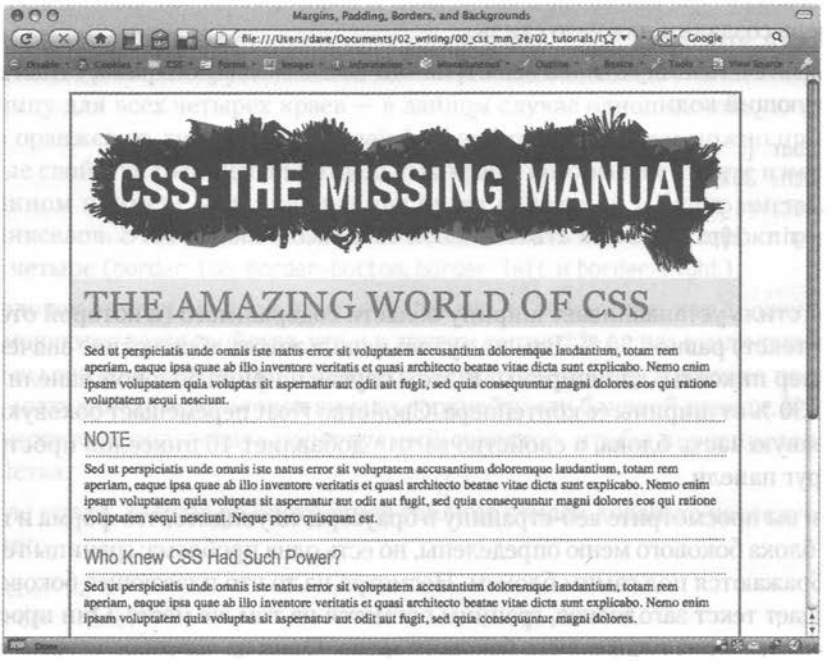


Рис. 7.16. С помощью нескольких стилей вы можете изменить фоновый цвет веб-страницы, добавить поля, регулировать интервалы между заголовками и абзацами

Создание бокового меню

Боковые меню — обычные элементы, применяемые в большинстве печатных изданий: журналах, книгах, газетах. Они отделяют и подсвечивают небольшие блоки информации, например перечень ресурсов, тематический анекдот. Но для того, чтобы боковые меню были достаточно эффективными и полезными, они не должны прерывать потока основного содержимого. Даже название «боковое меню» говорит о том, что этот блок должен быть расположен обособленно и примыкать к краю веб-страницы, что можно легко сделать средствами CSS.

1. Вернитесь к файлу `sidebar.html` в HTML-редакторе.

   Сначала нужно отделить область веб-страницы, составляющую боковое меню. Для этого прекрасно подходит тег `<div>`. С его помощью можно заключить любой объем HTML-кода в свой собственный отдельный блок.

2. Щелкните кнопкой мыши перед первым тегом `<h2>` (с заголовком «NOTE»). Наберите `<div class="sidebar">` и нажмите **Enter**.

   Этот HTML-код отмечает начало блока бокового меню и применяет к нему стилевой класс. Мы создадим сам стилевой класс `.sidebar`, но сначала нужно определить завершение блока бокового меню, закрыв `<div>`.

3. Перейдите к закрывающему тегу `</p>`, который следует сразу за тегом `<h2>` (это тот `</p>`, который появляется прямо перед `<h2>Who Knew CSS Had Such Power?</h2>`). Нажмите после него **Enter** и введите `</div>`.

   Мы только что заключили заголовок и маркированный список в тег `<div>`. Теперь создадим для него стиль.

4. Добавьте в таблицу стилей веб-страницы после созданного ранее стиля для `<h2>` следующий код:

```
.sidebar {
  width: 30%;
  float: right;
  margin: 10px;
}
```

   Этот стиль устанавливает ширину области содержимого (в которой отображается текст) равной 30 %. Вам не нужно использовать абсолютное значение, например пикселы, для ширины. В этом случае ширина боковой панели составляет 30 % от ширины ее контейнера. Свойство `float` перемещает боковую панель в правую часть блока, а свойство `margin` добавляет 10 пикселов пространства вокруг панели.

   Если вы просмотрите веб-страницу в браузере, то увидите, что форма и положение блока бокового меню определены, но есть одна проблема: границы тегов `<h2>` отображаются под самим блоком. Несмотря на то что плавающее боковое меню смещает текст заголовков, границы остаются на том же месте. Они проступают в качестве фона плавающего бокового меню. Один из способов решить эту проблему — просто добавить фоновый цвет для боковой панели, чтобы не видеть границ `h2` (но есть и другой способ, который вы будете использовать в шаге 8).

5. Добавьте другое свойство к стилю .sidebar, чтобы он выглядел следующим образом (изменения выделены полужирным шрифтом):

```
.sidebar {
  width: 30%;
  float: right;
  margin: 10px;
  background-color: #FAEBC7;
  padding: 10px 20px;
}
```

Это свойство добавляет светло-оранжевый цвет к боковому меню и оттесняет текст от границ бокового меню, чтобы он не касался границ, которые вы собираетесь добавить.

6. Наконец, добавьте еще два свойства к стилю .sidebar, чтобы он выглядел следующим образом (изменения выделены полужирным шрифтом):

```
.sidebar {
  width: 30%;
  float: right;
  margin: 10px;
  background-color: #FAEBC7;
  padding: 10px 20px;
  border: 1px dotted #FC6512;
  border-top: 20px solid #FC6512;
}
```

Это пример удобной методики, описанной ранее. Если вы хотите, чтобы большая часть границ вокруг элемента была одинаковой, можно сначала определить границу для всех четырех краев — в данном случае однопиксельную пунктирную оранжевую линию вокруг всей боковой панели. Затем можно применить новые свойства границы для отдельных краев, которые вы хотите изменить, — в данном примере верхняя граница будет сплошной и будет иметь высоту 20 пикселов. Этот способ позволяет использовать всего две строки кода CSS, а не четыре (border-top, border-bottom, border-left и border-right).

Заголовок внутри боковой панели выглядит не совсем так, как должен. К нему применяются те же свойства, что и к другим тегам <h2> (из-за стиля тега h2, который вы создали в шаге 5). Границы отвлекают внимание, а верхнее поле слишком «отталкивает» заголовок вниз от верхней части боковой панели. К счастью, вы можете использовать наследуемый селектор, чтобы переопределить эти свойства.

7. После стиля .sidebar во внутренней таблице стилей добавьте наследуемый селектор:

```
.sidebar h2 {
  border: none;
  margin-top: 0;
  padding: 0;
}
```

Из-за .sidebar этот стиль является доминирующим, то есть имеет большую значимость по сравнению с обычным стилем h2. Он стирает границу, полученную от оригинального стиля тега <h2>, вместе с верхним полем и всеми отступами. Тем не менее, поскольку в этом стиле не определены размер, цвет и начертание шрифта, эти свойства по-прежнему передаются от стиля h2 — каскадность в действии!

Страница стала хорошо выглядеть, но границы тегов <h2> все еще появляются под боковой панелью. На это не очень приятно смотреть, но все можно легко исправить.

8. Найдите стиль h2 и добавьте свойство overflow:

```
h2 {
  font-size: 24px ;
  font-family: "Arial Narrow", Arial, Helvetica, sans-serif;
  color: #F96B18;
  border-top: 2px dotted #8DA516;
  border-bottom: 2px dotted #8DA516;
  padding-top: 5px;
  padding-bottom: 5px;
  margin: 15px 0 5px 0;
  overflow: hidden;
}
```

Установив для свойства overflow значение hidden, вы спрячете границы, которые проходят за пределами текста заголовка и под плавающим элементом (к сожалению, Internet Explorer 6 не понимает этого и по-прежнему отображает границы под боковой панелью, но вы узнаете, как исправить проблему, в следующем разделе).

9. Сохраните файл и просмотрите веб-страницу в браузере. Она должна иметь вид, представленный на рис. 7.17.

К сожалению, в Internet Explorer 5 или 6 ваша страница не будет похожа на ту, что показана на рис. 7.17. В этих версиях браузера есть несколько ошибок, отрицательно влияющих на внешний вид веб-страницы. Дальше они будут описаны.

Устранение ошибок браузера

В Internet Explorer 7 и 8 веб-страница sidebar.html выглядит замечательно, но более ранние версии этого браузера не могут отобразить ее правильно. Если у вас есть Internet Explorer 6, попробуйте, и вы увидите, о чем идет речь.

С одной стороны, граница обоих заголовков <h2> выступает в качестве фона под боковым меню. В шаге 8 (выше) использовалось свойство overflow, позволяющее решить эту проблему в большинстве браузеров, но в браузере Internet Explorer 6 нужно применять какой-то другой способ. Кроме того, правое поле бокового меню заметно больше остальных.

ПРИМЕЧАНИЕ

Если у вас нет возможности убедиться в справедливости моих утверждений о существовании проблем с браузерами, просто поверьте на слово и используйте этот раздел в интересах посетителей вашего сайта, которые все еще пользуются Internet Explorer 6.

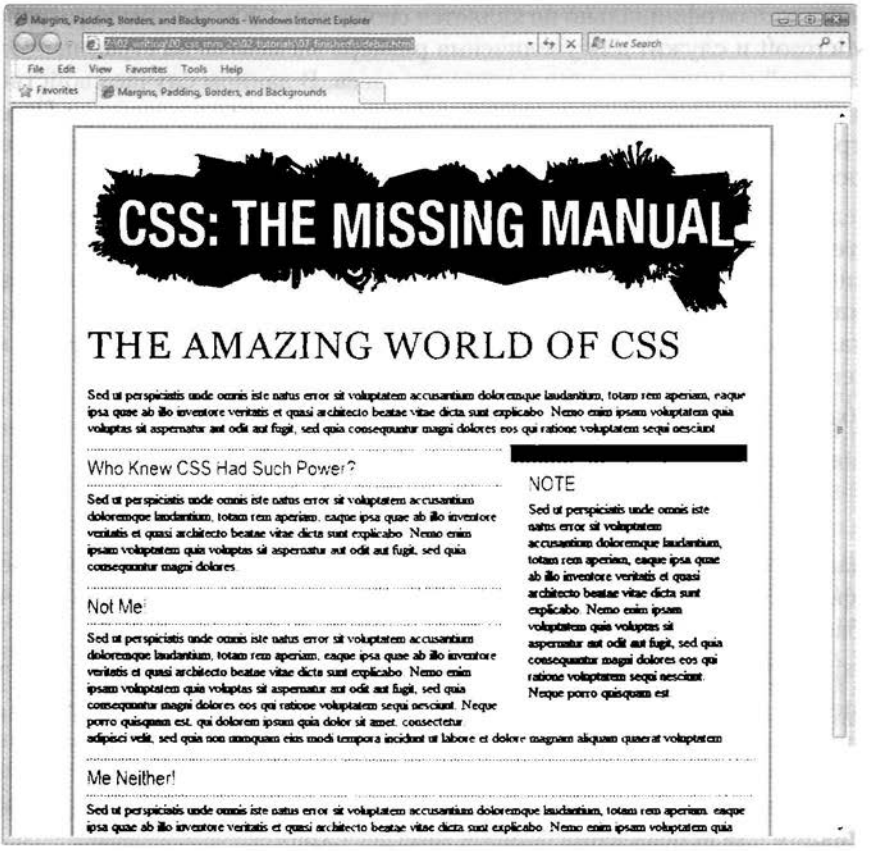


Рис. 7.17. Небольшой набор CSS-стилей придает непривлекательному HTML-коду элегантный дизайн

Первая ошибка, которой мы займемся, — это выступающие границы.

1. Вернитесь к файлу `sidebar.html` в HTML-редакторе.

   Вам нужно добавить всего одно свойство к стилю тега <h2>. Оно дает эффект только в Internet Explorer 6.

2. Добавьте в конце таблицы стилей веб-страницы `sidebar.html` следующий стиль:

```
h2 {
  font-size: 24px ;
  font-family: "Arial Narrow", Arial, Helvetica, sans-serif;
  color: #F96B18;
  border-top: 2px dotted #8DA516;
  border-bottom: 2px dotted #8DA516;
  padding-top: 5px;
  padding-bottom: 5px;
  margin: 15px 0 5px 0;
  overflow: hidden;
  zoom: 1;
}
```

Свойство zoom официально не является свойством CSS. Оно было разработано в Microsoft и служит для увеличения размера элементов на странице. Однако не по этой причине вы используете его здесь. В данном случае свойство zoom препятствует расширению границы под плавающим элементом в IE 6. Оно устраняет ошибку, связанную с границей, хоть и совершенно мистическим способом.

Следующая проблема: ошибка двойного поля. Она вызывает появление дополнительного пространства с правой стороны бокового меню. Поскольку эта ошибка затрагивает браузер Internet Explorer 6, создадим дополнительный стиль с селектором * html для бокового меню (конструкция * html скрывает остальную часть селектора — .sidebar — от обработки любым браузером, кроме Internet Explorer 6 и его более ранними версиями).

3. Добавьте в таблицу стилей следующий стиль:

```
* html .sidebar {
  display: inline;
}
```

В данном случае использование свойства display дает определенный полезный эффект, хотя и не имеет смысла в CSS. Это способ обмануть браузер, чтобы избавиться от лишнего правого поля.

4. Сохраните файл и просмотрите веб-страницу в браузере Internet Explorer 6.

Теперь она должна быть похожа на страницу, представленную на рис. 7.17, даже в браузере Internet Explorer 6. Всем веб-дизайнерам, хотят они того или нет, приходится иметь дело с этими ошибками браузеров, которые часто описываются в данной книге. Кроме того, в гл. 15 вы познакомитесь с несколькими методиками для исправления ошибок Internet Explorer.

Решение задания вы можете найти в файле sidebar_finished.html из папки 07_finished.

Двигаемся дальше

Чтобы проверить новые знания и навыки, попробуйте выполнить следующее практическое задание самостоятельно. Создайте для тега <p> стиль, который бы смог приукрасить абзац: попробуйте добавить поля, указать цвет шрифта и т. д. Затем создайте класс для стилизации примечания с информацией об авторском праве, которое должно отображаться в нижней части страницы sidebar.html (например, с именем .copyright). Добавьте в этот стиль верхнюю границу (над текстом примечания), измените цвет текста, уменьшите размер шрифта и измените регистр букв на прописные (используйте для этого свойство text-transform, описанное в разд. «Форматирование символов и слов» гл. 6). После создания стиля добавьте соответствующий атрибут класса к тегу <p> в HTML-коде.

8 Добавление графики на веб-страницы

Как бы вы ни украшали текст веб-страниц с помощью границ и полей, ничто так не повлияет на внешний вид сайта, как изображения. CSS предлагает вам исчерпывающий набор средств управления ими. Здесь можно работать с графическими элементами двумя способами: посредством тега `<img>` и свойства `background-image` (которое позволяет применить изображение к фону любого тега на странице).

В этой главе углубленно рассмотрены некоторые оригинальные методы применения изображений средствами языка CSS. На самом деле лучший способ узнать о возможностях графики и научиться ее использовать — наглядно увидеть результаты практических примеров, поэтому в конце главы есть три обучающих урока. В них мы создадим веб-страницу фотогалереи и попрактикуемся в быстрой и профессиональной разработке глобального дизайна с применением изображений.

CSS и тег <img>

Тег `<img>` является средством добавления изображений в веб-страницы фотогалерей еще со времен появления Интернета. Даже сайты без фотографий используют этот тег для добавления логотипов, управляющих элементов (кнопок) и иллюстраций. В CSS нет свойств, специально предназначенных для форматирования изображений. Однако вы можете использовать для стилизации графических элементов общие CSS-свойства, с которыми познакомились в предыдущих главах книги. Например, свойство `border` обеспечивает простой и быстрый способ заключить изображение в рамку или унифицировать вид галереи. Ниже представлен список CSS-свойств, наиболее часто используемых в сочетании с изображениями.

- `Border` — можно использовать одно из множества свойств границ (см. разд. «Добавление границ» гл. 7) для обрамления изображений. Пример приведен в обучающем уроке 1 этой главы. Поскольку каждая сторона изображения может иметь свои параметры границ: цвет, стиль, толщину линии — ваши творческие возможности расширяются.
- `Padding` — добавляет пустой промежуток между границей и изображением. Отделение рамки от фотографии небольшим промежутком имитирует подложку, которая традиционно используется для обрамления рисунков с целью отделения их друг от друга. Устанавливая цвет фона, вы даже можете изменить цвет самой подложки.

- `Float` — выравнивание изображения с помощью `float` перемещает его к левому или правому краю веб-страницы. Или, если изображение расположено внутри другого блочного элемента, например бокового меню, — к левой или правой стороне этого элемента. Таким образом, текст и другие элементы веб-страницы обтекают изображение с другой свободной стороны и отображаются далее под ним. Вы можете создать многократное обтекание рисунков для отображения фотогалереи в несколько строк и столбцов.
- `Margin` — чтобы добавить пустой промежуток между изображением и остальным содержимым веб-страницы, используйте свойство `margin`. Когда вы устанавливаете для изображения обтекание, текст обычно располагается слишком близко. Добавление левого поля (с выравниванием по правому краю) или правого поля (с выравниванием по левому краю) создает пустой отделяющий промежуток между текстом и графическим элементом.

В большинстве случаев вам не потребуется создавать стиль для самого тега `<img>`. Его форматирование затрагивает слишком большой диапазон элементов веб-страницы и изменит все изображения, включая элементы, обеспечивающие совершенно разные функции. Так, наряду с изображениями одинаковое форматирование будут иметь логотип, навигационные кнопки, фотографии и даже рекламные баннеры. Наверное, вы не хотели бы видеть одну и ту же черную рамку, обрамляющую все изображения. Вместо этого следует использовать стилевые классы, например `.galleryImage` или `.logo`, для применения стилей к нужным элементам выборочно.

Другой подход заключается в использовании селекторов потомков для целевого форматирования изображений, сгруппированных вместе в одном фрагменте веб-страницы. Если у вас есть галерея фотографий, можно заключить все внутри тега `<div>` с идентификатором `gallery` и затем создать стиль только для изображений, расположенных внутри этого блока, так: `#gallery img`.

Фоновые изображения

Свойство `background-image` — ключ к созданию визуально ошеломляющих сайтов. Стоит только научиться применять его и родственные свойства, и вы сможете заставить сайт выделяться среди прочих. Чтобы убедиться в эффективности фоновых изображений, зайдите по адресу **http://www.csszengarden.com**. HTML-код обеих веб-страниц, показанных на рис. 8.1, совершенно одинаков; визуальные различия достигаются благодаря использованию фоновых изображений.

Весь секрет в том, чтобы сделать любую веб-страницу уникальной в плане дизайна, и для достижения этой цели широкое распространение получили фоновые изображения (на самом деле, если вы посмотрите на HTML-код этих веб-страниц, вы увидите, что в них нет ни единого тега `<img>`).

Если вы уже создавали сайты раньше, то, наверное, использовали изображения в качестве фоновых рисунков веб-страниц. Скорее всего, они были небольших размеров, повторяющиеся на заднем плане окна браузера с едва различимым узором. Этот проверенный временем метод языка HTML использовал атрибут `background` тега `<body>`. CSS сделает все то же самое, но гораздо лучше.

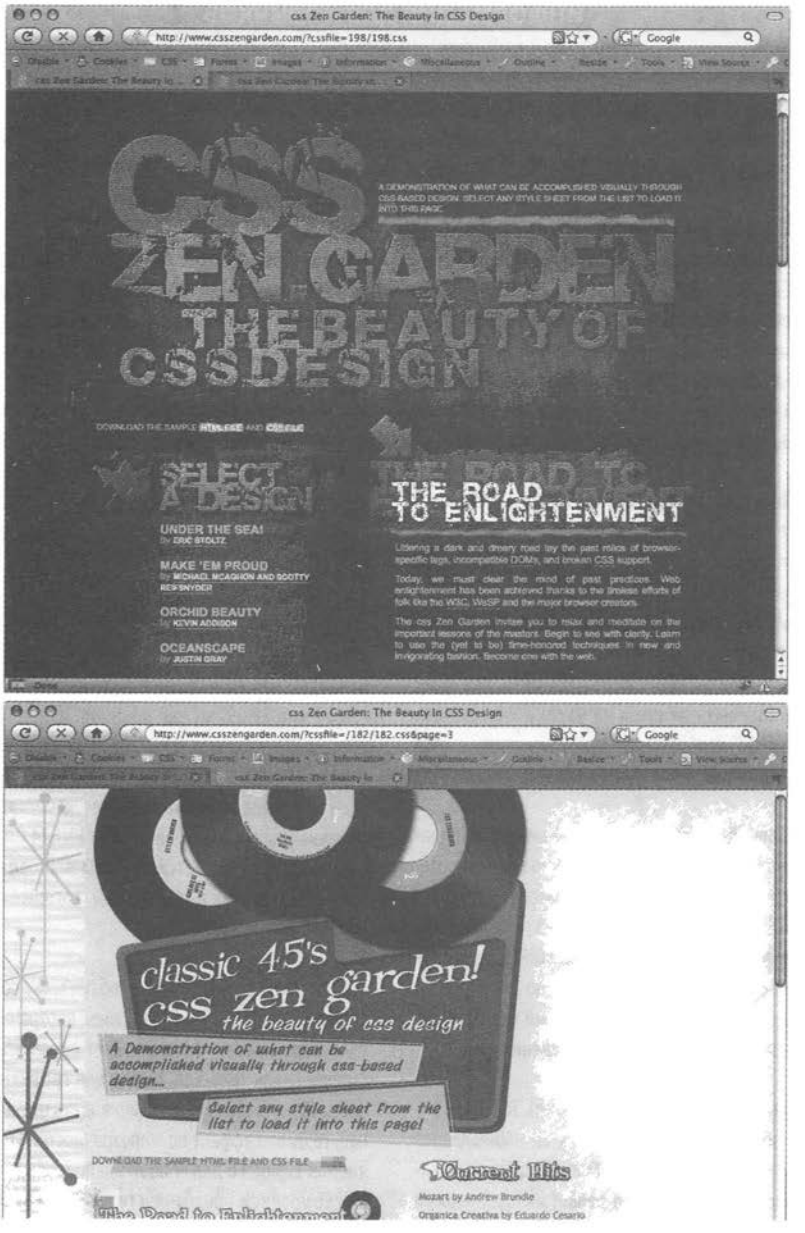


Рис. 8.1. Сайт csszengarden.com демонстрирует мощность каскадных таблиц стилей, показывая, как с помощью CSS вы можете превратить один и тот же HTML-файл в две совершенно разные веб-страницы

ПРИМЕЧАНИЕ

Далее в книге вы найдете описание трех свойств фоновых изображений и изучите CSS-код каждого из них. Позже в этой главе вы познакомитесь с сокращенным методом набора — вариантами свойств, которые сэкономят немало времени.

ИНФОРМАЦИЯ ДЛЯ НОВИЧКОВ

GIF-, JPEG- и PNG-файлы: веб-графика

Компьютерная графика представлена сотнями различных форматов файлов с такими замысловатыми аббревиатурами, как JPEG, GIF, TIFF, PICT, BMP, EPS и т. д.

К счастью, веб-графика немного проще. Современные браузеры работают только с тремя графическими форматами: GIF, JPEG и PNG. Каждый из них обеспечивает хорошее сжатие. Благодаря возможностям компьютера сжатие снижает размер графического файла, и тот может «путешествовать» по Интернету быстрее. Какой из этих форматов лучше выбрать, зависит от изображения, которое вы хотели бы добавить на свою страницу.

GIF (Graphics Interchange Format). Такие файлы обеспечивают хорошее сжатие для изображений, в которых есть области со сплошным цветом: логотипы, текст, простые баннеры и т. д. GIF-файлы также предоставляют возможность установить один из цветов прозрачным, а это означает, что вы можете сделать так, чтобы какой-нибудь цвет исчез, позволяя фону веб-страницы быть видным на части изображения. Кроме того, GIF-изображения могут включать в себя простую анимацию.

Изображения GIF содержат максимум лишь 256 оттенков, вследствие чего фотографии обычно выглядят пастеризованными (пятнистыми и нереалистичного цвета, как плакат). Другими словами, фото заката, которые вы сделали цифровой камерой, в формате GIF будет выглядеть не очень хорошо. Если вам не нужно анимировать изображение, то формат PNG8, который мы обсудим ниже, будет лучшим выбором, по сравнению с GIF.

JPEG (Joint Photographic Experts Group). Такая графика имеет сильные стороны там, где у GIF наблюдаются недостатки. Изображения JPEG могут содержать миллионы различных цветов, что делает их идеальными для фотографий. Однако файлы JPEG имеют преимущества не только в плане фотографий. Они способны сжимать изображения с множеством различных цветов гораздо лучше, чем GIF, потому что алгоритм сжатия JPEG предусматривает то, как человеческий глаз воспринимает смежные значения цветов. Когда графическое приложение сохраняет файл JPEG, происходит комплексный анализ цвета с целью снижения объема данных, необходимых для точного представления образа. С другой стороны, сжатие JPEG приводит к тому, что текст и большие площади со сплошным цветом покрываются пятнами.

Наконец, формат **PNG (Portable Network Graphics)** включает в себя лучшие черты GIF- и JPEG-форматов, но вы должны узнать, какую именно версию PNG использовать в конкретной ситуации. PNG8 в основном заменяет GIF. Как и GIF, он предлагает 256 цветов и базовую возможность сделать один цвет прозрачным. Тем не менее PNG8 обычно сжимает изображения и делает их размер несколько меньшим, чем GIF. По этой причине изображения PNG8 загружаются чуть быстрее, чем такие же изображения, сохраненные в формате GIF.

PNG24 и PNG32 (также известный как *PNG24 с альфа-прозрачностью*) предлагают расширенную цветовую палитру JPEG-изображений без потери качества. Это означает, что фотографии сохраняются в форматах PNG24 или PNG32 и, как правило, имеют более высокое качество, чем JPEG. Но прежде, чем перейти к формату PNG, следует помнить, что изображения JPEG предлагают очень хорошее качество и гораздо меньший размер файла, чем PNG24 либо PNG32. В общем, JPEG является лучшим выбором для фотографий и других изображений, которые состоят из множества цветов.

Однако PNG32 имеет одну особенность, которой нет у других форматов: это 256 уровней прозрачности (*альфа-прозрачность*). Она позволяет оформить фон веб-страницы как тень или задать для графики прозрачность 50 %, чтобы можно было видеть что-либо сквозь нее, создавая эффект полупрозрачности. К сожалению, Internet Explorer 6 для Windows неправильно отображает 256 уровней прозрачности файлов PNG32: вместо того чтобы сделать прозрачные области сквозными, IE 6 заменяет их отвратительным синим фоном (есть несколько методик, использующих JavaScript (см., например, http:// 24ways.org/2007/supersleight-transparent-png-in-ie6), которые могут помочь корректно отобразить в IE 6 прозрачные файлы PNG). К счастью, Internet Explorer 7 и 8 обрабатывают прозрачность PNG, как это делают Firefox, Safari и Opera.

Свойство background-image добавляет рисунок в качестве фона элемента. Чтобы поместить его на заднем плане всей веб-страницы, можно создать стиль для тега <body>:

```
body {
  background-image: url(images/bg.gif);
}
```

Свойство принимает единственное значение: ключевое слово url, за которым следует путь к файлу рисунка, заключенный в круглые скобки. Вы можете использовать абсолютный URL, например, так:

```
url(http://www.cosmofarmer.com/image/bg.gif)
```

Или относительный путь от документа или корневого каталога сайта:

```
url(../images/bg.gif) /* относительный путь от документа */
url(/images/bg.gif) /* относительный путь от корневого каталога */
```

Как описывается во врезке «Информация для новичков» (см. далее), относительный путь от документа указывает адрес файла таблицы стилей, но не стилизуемой HTML-страницы. Конечно, они будут совпадать при использовании внутренней таблицы, но вы должны помнить об этом, применяя *внешнюю* таблицу стилей. Предположим, у вас имеются папка **styles** (содержащая таблицы стилей сайта) и папка **images** (с изображениями сайта). Обе расположены в главном (корневом) каталоге вместе с начальной (домашней) страницей сайта (рис. 8.2). Когда посетитель просматривает ее, загружается также внешняя таблица стилей (шаг 1 на рис. 8.2).

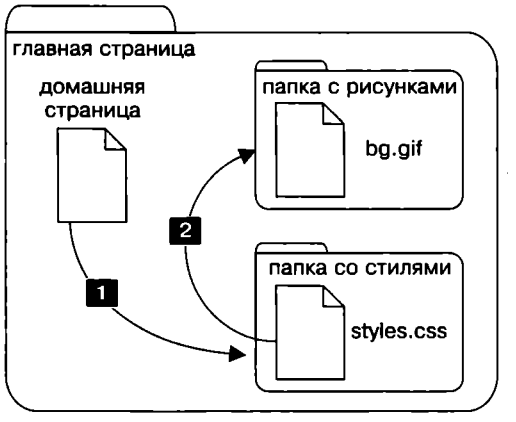


Рис. 8.2. Относительный путь от документа вычисляется применительно к файлу таблицы стилей, но не к стилизуемой веб-странице

Теперь допустим, что таблица включает стиль <body> с атрибутом фонового изображения, в котором в качестве рисунка выбран файл bg.gif из папки **images**. Относительный путь по отношению к документу приведет от таблицы стилей к рисунку (шаг 2 на рис. 8.2). Это будет выглядеть следующим образом:

```
body {
  background-image: url(../images/bg.gif);
}
```

Этот путь интерпретируется так:

- `../` означает «подняться вверх на один уровень», то есть к корневому каталогу, к папке, содержащей папку **styles**;
- `images/` означает «перейти к папке с изображениями **images**»;
- `bg.gif` определяет сам файл изображения.

В приведенных примерах путь не заключен в кавычки, как требует HTML; они необязательны, хотя и могут указываться. В CSS все три следующие строки кода идентичны:

```
background-image: url(images/bg.gif);
background-image: url("images/bg.gif");
background-image: url('images/bg.gif');
```

ИНФОРМАЦИЯ ДЛЯ НОВИЧКОВ

Типы URL

В CSS при добавлении фонового изображения или присоединении внешней таблицы стилей директивой `@import` вы должны определить URL: Uniform Resource Locator (унифицированный указатель ресурса, URL-адрес) — путь к файлу, расположенному в Интернете. Существует три типа путей: абсолютный, относительный от корневого каталога и относительный от документа. Все три варианта просто указывают, где браузер может найти определенный файл (например, другую веб-страницу, рисунок или внешнюю таблицу стилей).

Абсолютный путь похож на почтовый адрес — он содержит всю информацию, необходимую браузеру для нахождения файла. Абсолютный путь включает `http://`, имя компьютера (хоста) в сети, папку и название самого файла. Например: `http://www.cosmofarmer.com/images/bluegrass.jpg`.

Корневой относительный путь (относительный путь от корневого каталога) указывает, где находится файл самого верхнего уровня — корневой папки сайта. Он не содержит ни `http://`, ни доменного имени. Начинается с символа / (слеш), указывающего на корневой каталог сайта (папка, в которой располагается главная страница). Например, `/images/bluegrass.jpg` обозначает, что файл `bluegrass.jpg` находится в папке images, которая расположена в корневом каталоге. Самый простой способ определить корневой относительный путь — взять абсолютный и отбросить `http://` и имя хоста.

Относительный от документа определяет путь к файлу от текущего документа. Когда он указан в таблице стилей, он определяет путь до указанного файла от таблицы стилей, а не от текущей веб-страницы.

Вот несколько подсказок относительно того, какой тип URL нужно использовать в каждом конкретном случае.

- Если вы обращаетесь к файлу, который находится на сервере, отличном от того, где размещена ваша таблица стилей, то должны использовать абсолютный путь. Это единственный тип URL, который может указывать на другой сайт.

- Корневой относительный путь хорош для доступа к изображениям, хранящимся на вашем собственном сайте. Поскольку отсчет всегда начинается с корневого каталога, вы можете перемещать таблицу стилей в любое другое место, не затрагивая при этом пути от корневого каталога до изображений сайта. Однако этим способом затруднительно пользоваться, если вы только учитесь веб-дизайну: вы не сможете воспользоваться предварительным просмотром в браузере, пока не будете просматривать страницы через сервер в Интернете или установленный на компьютере для тестирования и отладочных целей. Другими словами, если вы попытаетесь просто открыть веб-страницу в своем компьютере с помощью меню File ▸ Open

ИНФОРМАЦИЯ ДЛЯ НОВИЧКОВ

(Файл ▸ Открыть), то вообще не увидите никаких изображений, которые помещены на веб-страницу с применением относительного пути от корневого каталога.

- Использование относительного пути от документа — лучший способ разработки веб-дизайна на собственном компьютере, без помощи сервера в Интернете. Вы можете создать CSS-файлы и затем просмотреть их в браузере, просто открывая страницу, сохраненную на жестком диске компьютера. Они будут работать и тогда, когда вы загрузите их на рабочий сайт, но вам придется переписать URL к изображениям, если вы захотите переместить таблицу стилей в другое место на сервере.

Управление повтором фоновых изображений

Устаревший HTML-атрибут `background` имеет одну проблему: фоновый рисунок многократно повторяется в виде мозаики, заполняя всю веб-страницу (в современном стандарте HTML эта недоработка исключена, и не только она). К счастью, CSS предлагает намного более богатые возможности по управлению фоновыми параметрами. Используя свойство `background-repeat`, вы можете определить, каким образом будет повторяться фоновый рисунок (и будет ли он повторяться вообще):

```
background-repeat: no-repeat;
```

Свойство может принимать четыре значения. Рассмотрим их по порядку.

- `repeat` — параметр по умолчанию, обеспечивает повторное отображение фонового рисунка слева направо сверху вниз, до полного заполнения всего пространства веб-страницы (рис. 8.3).
- `no-repeat` — отображает фоновый рисунок один раз, без повторения и перекрытия. Этот параметр используется на практике довольно часто, и мы также будем применять его для установки фоновых изображений тегов, отличных от `<body>`. Вы можете пользоваться им, чтобы поместить графический логотип вверху веб-страницы или создать собственные маркеры в списках (пример такого маркера приведен в обучающем уроке 2 этой главы). Другой пример использования этого параметра — применение его к верхней стороне блока тега `<div>` для создания закругленного края блочного элемента.
- `repeat-x` — вызывает повторение фонового изображения горизонтально вдоль оси *X* (по всей ширине веб-страницы). Это замечательное средство для добавления графического баннера (заголовка) сверху веб-страницы (рис. 8.4, *слева*), декоративной границы сверху или снизу заголовка.
- `repeat-y` — повторяет фоновое изображение вертикально вдоль оси *Y* (по всей высоте веб-страницы). Вы можете использовать этот параметр, чтобы добавить слева или справа веб-страницы графическое навигационное меню (см. рис. 8.4, *справа*) или создать теневой эффект с любой стороны элемента веб-страницы (возможно, того же бокового меню).

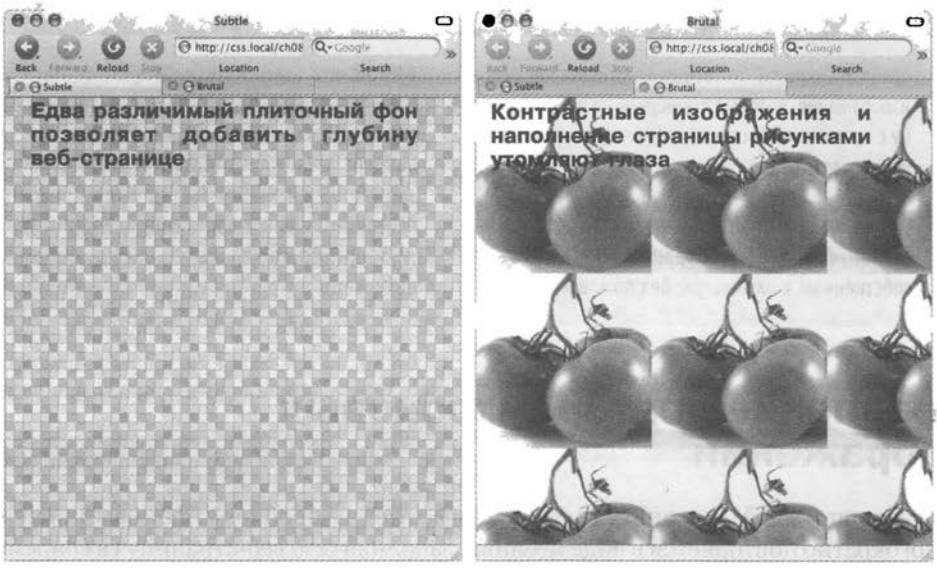


Рис. 8.3. Применяйте повторное отображение фоновых рисунков с осторожностью: выбирайте не слишком контрастные, плавно стыкующиеся изображения *(слева)*; четкие, яркие, высококонтрастные изображения *(справа)* делают основной текст веб-страниц нечитаемым

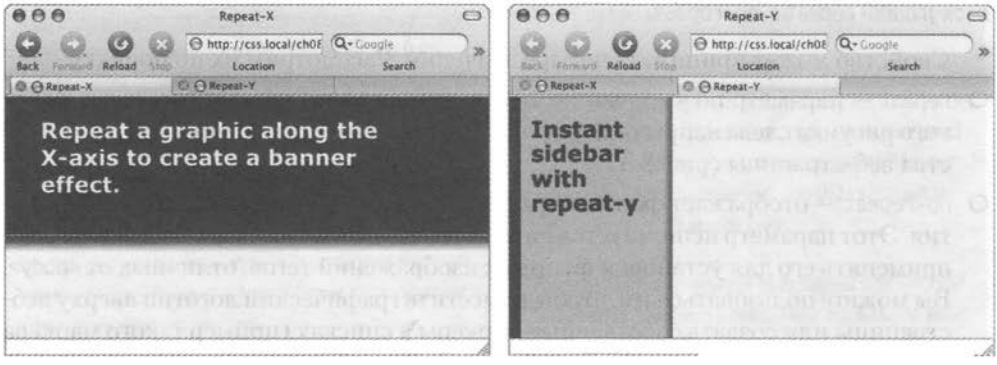


Рис. 8.4. Пользуясь возможностью управления повтором фоновых изображений, можно создать графический фон для заголовков *(слева)* и боковых меню *(справа)* веб-страницы

Позиционирование фоновых изображений

Размещение фоновых изображений и управление их повтором — это только половина дела. CSS-свойство позиционирования фонового изображения `background-position` позволяет управлять точным расположением несколькими способами. Вы можете определить начальную позицию фонового изображения в виде горизонтальной и вертикальной координат посредством трех ключевых слов, точных абсолютных и процентных значений.

Ключевые слова

Вы можете работать с двумя наборами ключевых слов: один из них имеет три параметра управления горизонтальным позиционированием: `left`, `center`, `right`, а другой — три параметра вертикального: `top`, `center`, `bottom` (рис. 8.5). Предположим, вы хотите поместить рисунок прямо в центре веб-страницы. Для этого можно создать следующий стиль:

```
body {
  background-image: url(bg_page.jpg);
  background-repeat: no-repeat;
  background-position: center center;
}
```

Чтобы переместить его в верхний правый угол веб-страницы, просто измените параметр позиции фонового рисунка:

```
background-position: right top;
```

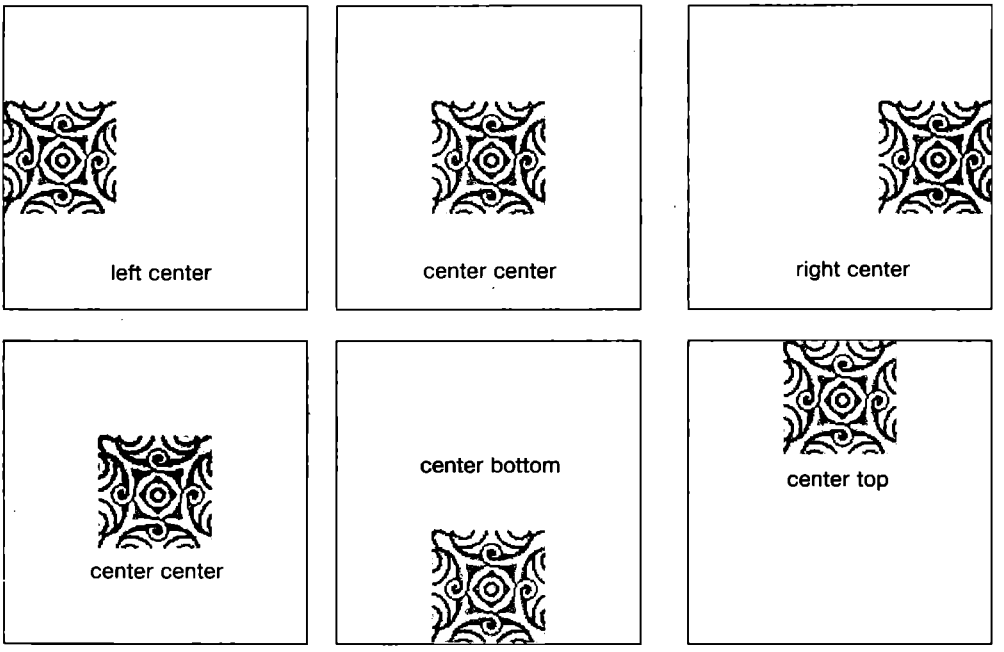


Рис. 8.5. Вы можете использовать для позиционирования фоновых изображений ключевые слова, последовательность написания которых не имеет значения

ПРИМЕЧАНИЕ

Если вы решили применить повторяющееся фоновое изображение (установив для свойства background-repeat одно из значений, описанных ранее), то начальную точку или координату первого отображаемого фонового рисунка определит свойство background-position. Например, если вы примените параметр repeat, то весь фон веб-страницы будет заполнен фоновым рисунком. Но именно background-position укажет, с какой позиции нужно начать повторное отображение.

Ключевые слова полезны, когда необходимо создать вертикальные или горизонтальные баннеры-заголовки. Если вы хотите, чтобы фоновый рисунок был горизонтально центрирован и повторялся от верхнего до нижнего края веб-страницы, создавая фон для всего содержимого (рис. 8.6, *слева*), то можете создать следующий стиль:

```
body {
  background-image: url(background.jpg);
  background-repeat: repeat-y;
  background-position: center top;
}
```

Аналогично, применяя ключевые слова `bottom`, `center` и `top`, вы можете установить горизонтальное повторение фонового изображения в определенном месте веб-страницы (или внутри стилизуемого элемента) и при использовании параметра `repeat-x` (см. рис. 8.4, *слева*). Чтобы поместить разделительную линию под заголовками в обучающем уроке 2, пользуйтесь методикой, представленной на рис. 8.6, *справа*.

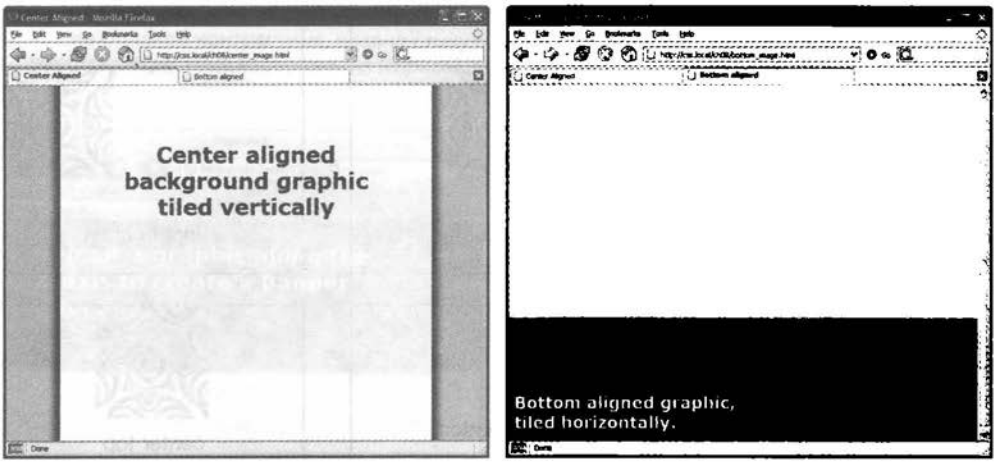


Рис. 8.6. При повторении фонового изображения вертикально *(слева)* или горизонтально *(справа)* пользуйтесь свойством background-position

На рис. 8.6, *слева*, фоновое изображение представляет собой широкий белый блок с тенями с левой и правой сторон. Цвет фона веб-страницы — серый, и текст выглядит так, как будто он написан на белом листе бумаги, лежащем на экране компьютера.

СОВЕТ

На самом деле вы можете добавить фоновый рисунок к обоим тегам: <html> и <body>. Если вы повторяете оба изображения по горизонтали и располагаете рисунок тега <body> вверху, а изображение тега <html> внизу, то можете достичь эффекта двух полосок, идущих через верхнюю и нижнюю части страницы, вне зависимости от высоты страницы. Это работает во всех современных браузерах, даже в IE 6!

ОШИБКИ БРАУЗЕРОВ

Проблема с отображением фоновых рисунков в нижней части окна браузера Firefox

Отображая рисунок в качестве фона всей веб-страницы, Firefox не всегда правильно устанавливает вертикальное позиционирование изображения. Например, если вы используете вертикальную позицию `bottom`, изображение не всегда появляется в нижней части окна браузера. Это случается, когда основное содержимое веб-страницы меньше по высоте.

Если веб-страница имеет всего несколько абзацев текста и отображается на большом мониторе, то Firefox принимает за нижний край окна браузера `bottom` нижний край последнего абзаца текста веб-страницы. Если вы столкнетесь с такой ситуацией, то просто добавьте следующий стиль: `html { height: 100 %; }` (в Internet Explorer 7 и ниже нет этой проблемы).

Точные значения

Позиционировать фоновые изображения можно, используя точные значения в пикселах или em. При этом нужно указать два значения: одно из них определяет расстояние между левой стороной изображения и левым краем элемента-контейнера, а другое — расстояние между верхней стороной изображения и верхним краем элемента-контейнера (другими словами, первое значение указывает горизонтальную координату, а второе — вертикальную).

Допустим, вы хотите установить собственные маркеры для списка. При добавлении в тег `<li>` фонового рисунка маркеры не всегда будут отображаться правильно центрированными на строке (рис. 8.7, *вверху*). Придется выровнять маркеры списка посредством свойства позиционирования фонового рисунка (см. рис. 8.7, *внизу*). В нашем случае элементы списка будут выглядеть гораздо симпатичнее, если расположить маркеры на 5 пикселов правее и на 8 пикселов ниже. Для этого добавим в стиль фонового изображения следующее свойство:

```
background-position:5px 8px;
```

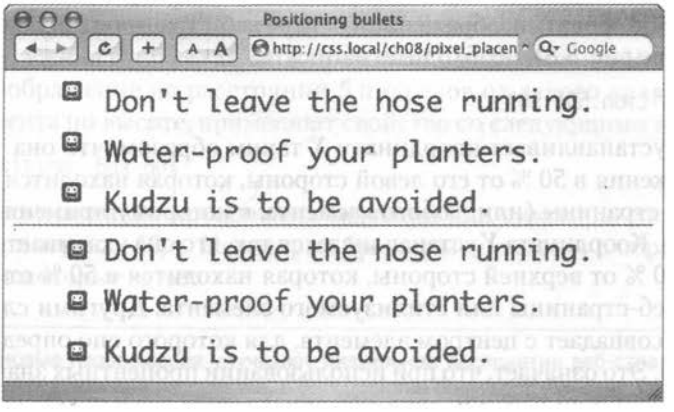


Рис. 8.7. Использование собственных изображений маркеров для списков иногда требует их точного позиционирования так, чтобы они имели соответствующее центрирование и правильно отображались относительно текста элементов списка

Нельзя указывать расстояние относительно нижнего (bottom) или правого (right) края веб-страницы или стилизуемого элемента в пикселах или em, поэтому, если вы хотите быть уверенными в том, что изображение помещено точно в правый нижний угол, используйте в качестве единиц измерения либо ключевые слова (bottom right), либо процентные значения, как описывается далее. Однако вы можете использовать отрицательные значения, чтобы сместить изображение относительно правого края или поднять относительно верхнего, скрывая часть. Возможно, вы захотите применить значения для подрезки части изображения. Или, если у фонового изображения имеется большое пустое поле с левой или верхней стороны, можно использовать отрицательные значения для его устранения.

Процентные значения

Для позиционирования фоновых изображений вы также можете применять процентные значения после ключевых слов или абсолютных значений, описанных выше. Вы добьетесь интересного эффекта. В данном случае можно позиционировать элемент на расстояние, пропорциональное его ширине. Например, когда вам нужно переместить рисунок на расстояние в три четверти размера элемента от заголовка, но вы не знаете ширины.

ПРИМЕЧАНИЕ

Процентными значениями часто пользуются в сочетании с плавающими элементами для установки фонового цвета или рисунка для левого и (или) правого навигационных меню, которые занимают всю высоту веб-страницы.

Как и в случае с пикселами или значениями в em, вы должны указать два процентных значения: одно представляет собой горизонтальную координату, а второе — вертикальную. Относительно чего рассчитывается значение, указанное в процентах? Если объяснить в двух словах, то оно приравнивается к процентному значению стилизуемого элемента.

Лучший способ понять — рассмотреть несколько практических примеров. Чтобы позиционировать изображение в центре веб-страницы (аналогично изображению в центре рис. 8.8), необходимо написать:

```
background-position:50% 50%;
```

Свойство устанавливает координату *X* таким образом, что она указывает на точку изображения в 50 % от его левой стороны, которая находится в 50 % от левого края веб-страницы (или любого элемента, к которому применяется фоновое изображение). Координата *Y* устанавливается так, что она указывает на точку изображения в 50 % от верхней стороны, которая находится в 50 % от соответствующего края веб-страницы или стилизуемого элемента. Другими словами, центр изображения совпадает с центром элемента, для которого оно определено в качестве фонового. Это означает, что при использовании процентных значений точные координаты позиционирования динамически изменяются и похожи на «движущуюся мишень» (поэтому координаты позиционирования стилизуемого элемента в процентах могут изменяться, если посетители веб-страницы меняют размеры окна браузера).

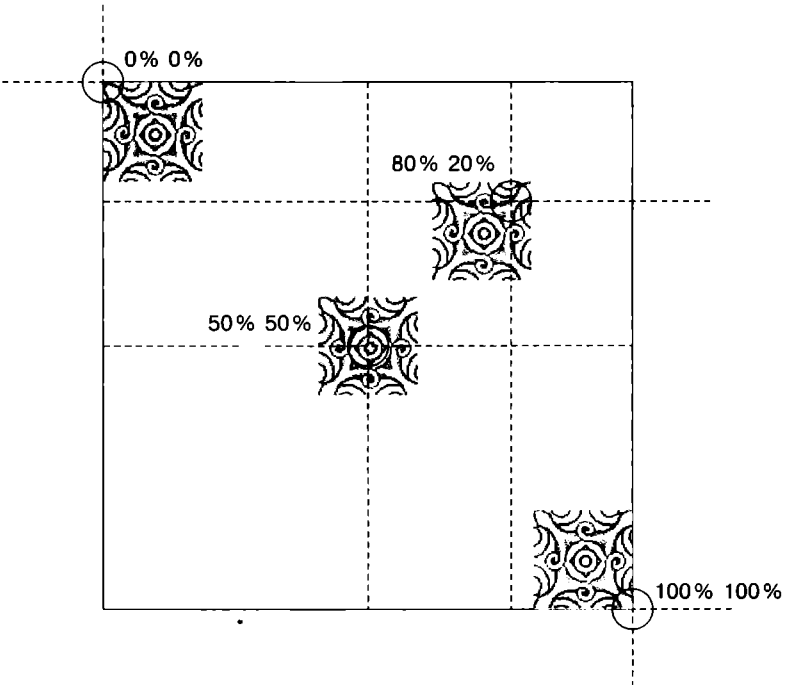


Рис. 8.8. Используя проценты, сначала определите начальную «якорную» точку (координату) в изображении

ПРИМЕЧАНИЕ

Размещение изображения по вертикали в фоне страницы с помощью процентов не обязательно поместит его в правильном месте, если содержимое страницы не заполняет всю высоту окна браузера.

Как и в случае с пикселами и em, можно назначать отрицательные процентные значения, но результаты трудно предугадать. Вы можете также комбинировать и подбирать значения в пикселах, em и процентах одновременно. Например, чтобы поместить изображение на расстоянии 5 пикселов от левого края элемента, но в центре элемента по высоте, применяют свойство со следующими параметрами:

```
background-position: 5px 50%;
```

Избегайте смешивания процентных значений или пикселов/em с ключевыми словами, например `top 50px`. Некоторые браузеры справятся с обработкой этого сочетания, а некоторые — нет.

ПРИМЕЧАНИЕ

Безусловно, фоновые изображения улучшают визуальное восприятие веб-страниц, но они, как правило, не отображаются при распечатке. В большинстве браузеров фоновые изображения могут распечатываться, но обычно это требует дополнительных действий. Если вы хотите предоставить возможность посетителям сайта печатать веб-страницы с рисунками в таком виде, как они отображаются в браузере, то вместо фоновых изображений по-прежнему используйте тег <img> для вставки таких важных фрагментов, как логотип сайта или схема проезда к интернет-магазину.

Фиксация изображения на месте

Когда посетитель прокручивает содержимое веб-страницы так, чтобы увидеть остальную ее часть, фоновое изображение прокручивается вместе с содержимым. В результате кажется, что рисунок на заднем плане перемещается вместе с текстом. Кроме того, если он не повторяется, то в процессе прокрутки исчезнет из виду. Когда вы устанавливаете в качестве фонового изображения веб-страницы логотип сайта или водяной знак, вы, вероятно, хотите, чтобы он всегда оставался в пределах видимости во время просмотра.

В CSS такая проблема решается посредством свойства `background-attachment`, которое может принимать два значения: `scroll` и `fixed`. Значение по умолчанию `scroll` определяет такое поведение браузера, при котором фоновое изображение прокручивается вместе с текстом и другим содержимым. Значение `fixed` предотвращает перемещение, жестко фиксируя его на заднем плане (рис. 8.9). Так, если вы хотите поместить логотип компании в левом верхнем углу веб-страницы и зафиксировать его там даже в случае прокрутки посетителями содержимого вниз, можете создать следующий стиль:

```
body {
  background-image: url(images/logo.gif);
  background-repeat: no-repeat;
  background-attachment: fixed;
}
```

Значение `fixed` также удобно использовать с повторяющимся мозаичным фоновым изображением. Когда вам требуется выполнить прокрутку, текст основного содержимого плавно перемещается (буквально плывет) и изящно исчезает вверху веб-страницы, а фоновое изображение остается на месте, создавая красивый эффект.

ПРИМЕЧАНИЕ

CSS позволяет «прикрепить» фоновое изображение к стилю любого элемента веб-страницы, а не только тега <body>. Однако браузер Internet Explorer 6 и его более ранние версии понимают свойство background-attachment только в случае применения стиля к <body>.

Сокращенный вариант свойства background

Как вы видите из примеров предыдущих разделов, чтобы воспользоваться всеми преимуществами фоновых изображений, необходимо применять свойства, управляющие параметрами фоновых изображений. Но многократный повторный набор таких длинных свойств, как `background-image`, `background-attachment` и т. д., отнимает много времени. Существует более простой метод — применение сокращенного свойства `background`.

Фактически вы можете объединить и перечислить все фоновые свойства (включая `background-color`, с которым мы познакомились в предыдущей главе) в единственной строке лаконичного кода CSS.

Просто набирайте свойство `background`, за которым должны следовать значения для `background-color`, `background-image`, `background-attachment`, `background-position`.

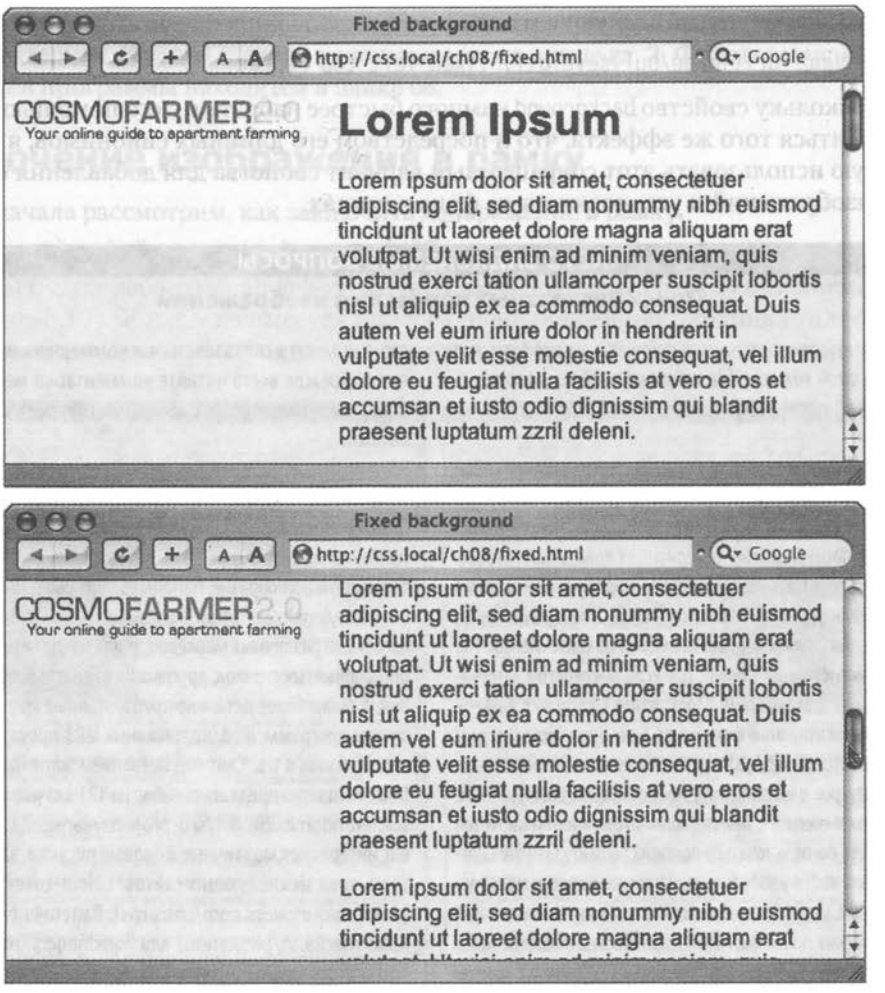


Рис. 8.9. Для закрепления логотипа используйте свойство background-attachment с параметром fixed

Следующий стиль устанавливает фон белого цвета с едва заметным неповторяющимся отпечатком, закрепленным по центру веб-страницы:

```
body {
  background: #FFF url(bullseye.gif) scroll center center no-repeat;
}
```

Вовсе не обязательно указывать абсолютно все параметры свойства. Вы можете использовать один из них или любое сочетание. Например, `background: yellow` равнозначно `background-color: yellow`. Все те параметры свойства, которые вы не определите сами, будут иметь стандартные значения по умолчанию. Допустим, вы определили только само фоновое изображение:

```
background: url(image/bullseye.gif)
```

Это эквивалентно следующему стилю:

```
background: url(image/bullseye.gif) fixed left top repeat;
```

Поскольку свойство background намного быстрее набрать и с его помощью можно добиться того же эффекта, что и посредством его длинных синонимов, я рекомендую использовать этот сокращенный вариант свойства для добавления фоновых изображений и установки цвета фона в стилях.

ЧАСТО ЗАДАВАЕМЫЕ ВОПРОСЫ

Поиск бесплатных коллекций изображений

Вы не художник, не умеете рисовать, и у вас даже нет цифровой камеры. Где же можно найти иллюстративный материал для сайта?

Нужно отдать должное Сети. Она представляет собой универсальное средство поиска, благодаря которому можно выйти из любой ситуации. Существует множество платных сайтов, содержащих коллекции готовых фотографий и иллюстраций, но наряду с ними также имеется довольно много бесплатного материала. Образцы фотографий можно найти на сайте Morgue File (www.morguefile.com), где есть множество замечательных фотографий. Stock.xchng (www.sxc.hu) — еще один отличный фоторесурс. Сайт Open Photo (http://openphoto.net/gallery/browse.html) предлагает коллекции изображений, предоставляя возможность их применения с некоторыми ограничениями, и вы можете воспользоваться поиском по сайту Creative Commons и найти изображения (а также видео и музыку), которые могут быть задействованы в личных или коммерческих проектах: http://search.creativecommons.org. Кроме того, вы можете искать картинки с лицензией Creative Commons через Flickr (www.flickr.com/creativecommons) и Picasa Web Albums (http://picasaweb.google.com) (надо отметить, что за фотографии на бесплатных сайтах не нужно платить, но не все они могут использоваться в коммерческих проектах; прежде всего читайте комментарии мелким шрифтом ко всем фото, с которыми собираетесь работать).

Если вы ищете образцы маркеров для списков, значки для панели управления, рисунки фоновых узоров для заполнения экрана, то у вас есть возможность выбора из большого многообразия. Например, сайт Bullet Madness (www.stylegala.com/features/bulletmadness/) предлагает 200 различных маркеров, в том числе варианты общепринятых стрелок, круглых и квадратных маркеров, а также более детально проработанные маркеры: значки программ, iPod (портативных MP3-проигрывателей), папок и т. д. Сайт под названием Some Random Dude бесплатно предлагает набор из 121 значка: www.somerandomdude.net/srd-projects/sanscons. Если вас интересуют мозаичные фоновые рисунки, зайдите на один из следующих сайтов: Colour-Lovers.com (www.colourlovers.com/patterns), Pattern4u (www.kollermedia.at/pattern4u) или Squidfingers (http://squidfingers.com/patterns). Можете также создать свой собственный черепичный фон с помощью следующих инструментов: BgPatterns (http://bgpatterns.com), Stripe Generator 2.0 (www.stripegenerator.com) или PatternCooler (www.patterncooler.com).

Обучающий урок 1: совершенствуем изображения

Фотогалерея — отличный пример привлекательной веб-страницы. Этот обучающий урок воедино собирает и наглядно демонстрирует различные способы стилизации изображений. Мы попрактикуемся в форматировании изображений с добавлением рамок и надписей, создадим универсальную фотогалерею, легко адаптируемую для просмотра в браузерах с различными размерами окна, применим фоновые рисунки для создания профессиональных визуальных теневых эффектов.

Чтобы начать обучающий урок, вы должны загрузить файлы, содержащие учебный материал. Как это сделать, рассказывается в конце гл. 2. Файлы текущей обучающей программы находятся в папке 08.

Заключение изображения в рамку

Для начала рассмотрим, как заключить изображение в рамку.

1. Откройте файл `image.html` из папки **08\image_ex** в браузере.

   Мы будем работать над веб-страницей вымышленного сайта CosmoFarmer.com (рис. 8.10). У нас уже имеется присоединенная внешняя таблица стилей, создающая модель веб-страницы и обеспечивающая ей простейший дизайн.

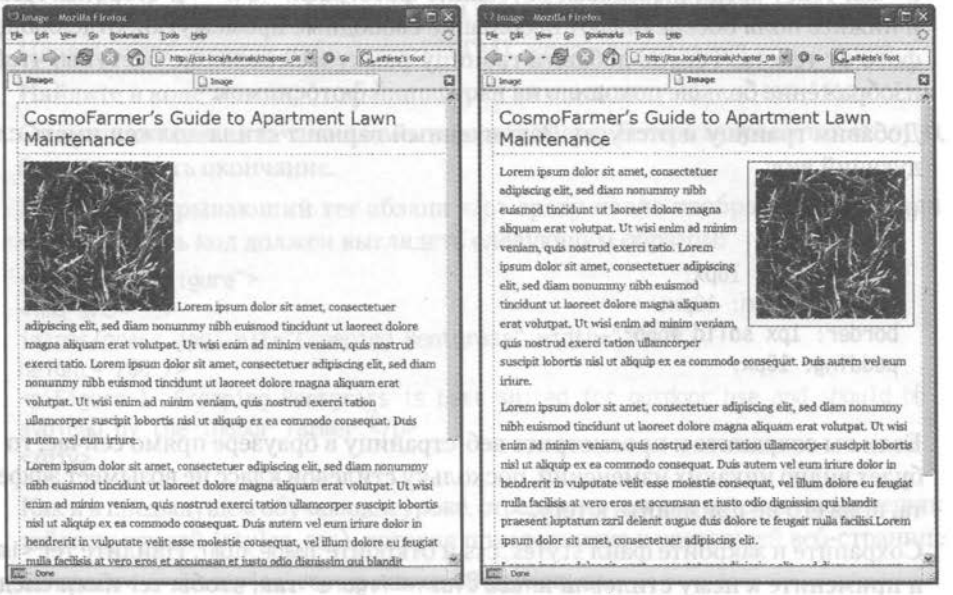


Рис. 8.10. Два варианта веб-страницы: до *(слева)* и после *(справа)* CSS-стилизации

На рис. 8.10 форматирование веб-страницы выполнено исключительно средствами языка HTML. Имеются недостатки. В частности, изображения занимают на странице слишком много места (см. рис. 8.10, *слева*). С помощью кода CSS (см. рис. 8.10, *справа*) вы легко можете заключить изображение в симпатичную рамку и визуально выделить его из основного текстового содержимого.

2. Откройте файл `styles.css` из папки **image_ex** в текстовом редакторе.

   Этот файл представляет собой внешнюю таблицу стилей, которая используется в `image.html`. Вы начнете с добавления класса стиля к этой таблице, а затем примените класс к тегу `<img>` в файле HTML.

3. Перейдите к концу файла и наберите следующее:

```
img.figure {

}
```

Селектор `img.figure` воздействует на любой тег `<img>`, к которому применен класс `figure`. Вы будете использовать его для выборочного форматирования некоторых изображений (вы могли бы назвать стиль просто `.figure`, но в таком случае он применялся бы к любому тегу с этим классом, а не только к изображениям).

4. Добавим в только что созданный стиль свойства `float` и `margin`:

```
float: right;
margin-left: 10px;
margin-bottom: 10px;
```

Свойство `float` смещает изображение к правой стороне веб-страницы, приподнимая текст вверх и создавая обтекание фотографии с левой стороны. Левое и нижнее поля обеспечивают небольшие свободные промежутки, отделяющие фото от текста. Теперь добавим границу-рамку и небольшие отступы, чтобы изображение больше походило на настоящий фотоснимок.

5. Добавим границу и отступы. Законченный вариант стиля должен иметь следующий вид:

```
img.figure {
  float: right;
  margin-left: 10px;
  margin-bottom: 10px;
  border: 1px solid #666;
  padding: 10px;
}
```

Если вы сохраните и просмотрите веб-страницу в браузере прямо сейчас, то не будет видно никаких изменений, поскольку стилевой класс не возымеет эффекта, пока его не применить к тегу.

6. Сохраните и закройте файл `styles.css` и откройте `image.html`. Найдите тег `<img>` и примените к нему стилевой класс `class="figure"` так, чтобы тег имел следующий вид:

```
<img src="../
images/grass.jpg" alt="Apartment Grass" width="200" height="200"
class="figure" />
```

Теперь изображение приобретет все форматирующие параметры, определенные для стилевого класса `.figure`.

7. Просмотрите веб-страницу в браузере. Она должна выглядеть так, как страница, представленная на рис. 8.10, *справа*.

Конечную версию веб-страницы, получившейся в данном обучающем уроке, вы сможете найти в папке **08_finished\image_ex**, файл `image.html`.

Изображение может заменить тысячу слов, но иногда все-таки требуется небольшой комментарий, поясняющий рисунок. В следующей части настоящей обучающей программы мы добавим под фотографией текстовую подпись.

Добавление подписей к изображениям

Нередко требуется добавить к изображению текстовую подпись с подробной информацией об объекте, месте съемки и т. д. При этом вы наверняка захотите, чтобы комментарий также был неделим с изображением, чтобы рядом стоящий текст обтекал надпись так же, как и само изображение. Лучший способ — заключить изображение и текст в контейнер — тег , — для которого будет определено единое форматирование. При этом методе фотография и связанный с ней текст обрабатываются как цельный блочный элемент. Если вы позже решите изменить их размещение на веб-странице и, возможно, установить обтекание с выравниванием по левому краю страницы — никаких проблем: вам потребуется просто изменить форматирование для всего элемента-контейнера.

1. Откройте файл caption.html из папки **08\caption_ex** в HTML-редакторе.

   Начнем с добавления небольшого HTML-кода для создания контейнера.

2. Найдите в коде тег <img>, добавьте перед ним команду .

   Она устанавливает начало элемента-контейнера. Теперь закроем тег , чтобы определить окончание.

3. Найдите закрывающий тег абзаца </p> сразу после изображения и введите . Теперь код должен выглядеть следующим образом:

```
<div class="figure">
<img src="../
images/grass.jpg" alt="Creeping Bentgrass" width="200"
height="200"/>
<p>Figure 1: Creeping Bentgrass is best suited for outdoor use and should be
avoided by the indoor farmer.</p>
</div>
```

   Как и в предыдущем обучающем уроке, отредактируем уже имеющуюся внешнюю таблицу стилей (styles.css), которая присоединена к настоящей веб-странице.

4. Откройте файл styles.css из папки **08\caption_ex**.

   Обратите внимание, что, поскольку это внешняя таблица стилей, в ней нет тега <style>. Он нужен только для внутренних таблиц стилей.

5. Перейдите в конец файла и добавьте следующий стиль:

```
.figure img {
  border: 1px solid #666;
  padding: 10px;
}
```

   Этот производный селектор воздействует на любой тег <img>, вложенный в другой тег, к которому применен стилевой класс figure, — в данном случае на только что добавленный . Поскольку мы используем производный селектор здесь (и в шаге 7), нам не нужно присваивать стилевой класс <img>. В результате не нужно набирать лишний код, то есть мы сокращаем HTML-код, обеспечивая посетителям сайта более быструю загрузку веб-страницы.

Теперь отформатируем элемент таким образом, чтобы основной текст веб-страницы обтекал фотографию и надпись (подпись) к ней, выровненные по правому краю.

6. Добавьте в файл styles.css следующий стиль:

```
.figure {
  float: right;
  width: 222px;
  margin: 15px 10px 5px 10px;
}
```

Мы уже использовали свойство float: right в настоящем обучающем уроке, а свойство margin добавляло небольшие промежутки со всех четырех сторон элемента . Но вы можете спросить, какова же ширина этих полей? Хотя для фотографии ширина установлена (200 пикселов; см. шаг 3 урока), текст подписи определенной ширины не имеет. Если вы не установите ширину для текста абзаца, то длинный текст вызовет увеличение ширины блока , который станет больше самой фотографии. Нам же нужно, чтобы подпись к рисунку имела размер, равный фотографии и ее рамке.

С помощью несложных арифметических вычислений подсчитаем полную ширину блока, занимаемого рисунком на веб-странице: сама фотография имеет ширину 200 пикселов плюс по 10 пикселов на левый и правый отступы, а также по 1 пикселу на левую и правую границы — итого общая ширина фотографии от ее левой до правой границы (рамки) равна 222 пикселам.

Следующим шагом будет улучшение вида текстовой подписи к рисунку.

7. Добавьте в таблицу стилей styles.css следующий стиль:

```
.figure p {
  font: bold 1em/normal Verdana, Arial, Helvetica, sans-serif;
  color: #333;
  text-align: center;
}
```

Этот стиль использует несколько свойств, рассмотренных в гл. 6, для создания выровненной по центру подписи, отображаемой полужирным шрифтом Verdana серого цвета. К счастью, сокращенное свойство font в первой строке стиля позволяет нам заменить четыре различных свойства одним.

Здесь мы опять пользуемся преимуществами производных селекторов (.figure p) для выборки только текста абзаца подписи рисунка. Чтобы заголовок еще лучше выделялся, добавим фоновый цвет и границу-рамку.

8. Добавьте в стиль .figure p еще три свойства:

```
.figure p {
  font: bold 1em/normal Verdana, Arial, Helvetica, sans-serif;
  color: #333;
  text-align: center;
  background-color: #e6f3ff;
  border: 1px dashed #666;
  padding: 5px;
}
```

Назначение свойств background-color, border и padding должно быть понятно — они создают цветной блок (прямоугольник) для подписи. Настало время посмотреть на результаты работы в браузере.

9. Сохраните файлы caption.html и styles.css и воспользуйтесь предварительным просмотром веб-страницы caption.html в браузере.

   Теперь вы понимаете единственную причину, по которой проще разрабатывать дизайн веб-страницы с применением внутренней таблицы стилей — вам нужно работать с одним файлом вместо двух.

   Веб-страница выглядит замечательно: фотография и заголовок выровнены по правому краю окна браузера, а подпись отчетливо выделяется. Однако есть одна небольшая проблема: если вы внимательно посмотрите на текст подписи, то обнаружите, что абзац имеет небольшие отступы слева и справа и текст по ширине занимает пространство немного меньшее, чем фотография. Это и есть пример одной из многочисленных проблемных ситуаций, которые неминуемо возникают при работе с CSS.

   В данном случае вы столкнулись с проблемой каскадности. Текст подписи размещен внутри тега абзаца <p>, и, кроме того, в таблице стилей styles.css имеется стиль для <p>. Рассмотрев его, мы видим, что он устанавливает верхнее и нижнее поля равными 10 пикселам, а также левое и правое поля — 8. Нам нужно переопределить их, например указав новые значения полей в более явно определенном стиле (описание приоритетов стилей и каскадности см. в разд. «Особенности механизма каскадности: какие стили имеют преимущество» гл. 5). К счастью, у вас уже есть более явно определенный стиль — .figure p. Таким образом, чтобы переопределить параметры полей стиля абзаца p с меньшим приоритетом, вы должны всего лишь добавить новые параметры полей в стиль .figure p.

10. Добавьте свойство margin в стиль .figure p:

```
.figure p {
  font: bold 1em/normal Verdana, Arial, Helvetica, sans-serif;
  color: #333;
  text-align: center;
  background-color: #e6f3ff;
  border: 1px dashed #666;
  padding: 5px;
  margin: 10px 0 0 0;
}
```

   Оно удаляет поля со всех сторон подписи за исключением верхнего края, устанавливая промежуток высотой 10 пикселов между подписью и вышестоящей фотографией.

11. Сохраните файлы caption.html и styles.css. Просмотрите веб-страницу caption.html в браузере.

   Теперь веб-страница должна выглядеть подобно странице, представленной на рис. 8.11 (законченную версию вы можете найти в папке 08_finished\caption_ex учебного материала).



Рис. 8.11. Применив тег элемента-контейнера , обтекание с выравниванием по правому краю и некоторое дополнительное форматирование, очень просто добавить подписи к фотографиям

Обучающий урок 2: создание фотогалереи

Раньше веб-дизайнеры создавали фотогалереи на основе HTML-тега таблицы `<table>`, помещая изображения в ячейки, образуемые строками и столбцами. Но сейчас вы можете достигнуть того же самого эффекта с помощью короткого CSS-кода в сочетании с применением гораздо менее объемного HTML-кода.

1. Откройте файл `gallery.html` из папки **08\gallery_ex**. Во-первых, посмотрите на HTML-код, который использован для создания фотогалереи. Веб-страница содержит девять фотографий и подписей к ним. Каждая располагается в отдельном блоке-контейнере `<div>`, к которому применен стилевой класс `figure`. Этот `<div>` функционирует точно так же, как и элемент `<div>`, использованный в предыдущей части урока для добавления подписи к рисунку. Сама фотография заключена в другой блок `<div>`, к которому применен стилевой класс `photo`:

```
<div class="figure">
  <div class="photo">
    <img src="../images/dandelion.jpg" alt="Dandelion"
    width="200" height="200" />
  </div>
<p>Figure 6: The dandelion: scourge of the apartment farmer. </p> </div>
```

ПРИМЕЧАНИЕ

Второй элемент-контейнер пригодится в следующей части урока, когда мы будем учиться добавлять к фотографиям теневые эффекты.

2. Поместите курсор сразу за тегом <link>, находящимся почти в начале файла, и нажмите Enter для создания новой строки.

   Тег <link> присоединяет внешнюю таблицу стилей, содержащую базовое форматирование веб-страницы.

3. Добавьте внутреннюю таблицу стилей, а затем два новых стиля следующим образом:

```
<style type="text/css">
.photo img {
  border: 1px solid #666;
  background-color: #FFF;
  padding: 4px; }

.figure p {
  font: 1.1em/normal Arial, Helvetica, sans-serif;
  text-align: center;
  margin: 10px 0 0 0;
}
</style>
```

   Эти два стиля добавляют границы-рамки ко всем изображениям галереи и устанавливают шрифт, выравнивание и поля для подписей изображений. Они используют производные селекторы для выборки изображений и текстовых абзацев подписей, являющихся элементами фотогалереи.

   Все изображения и подписи к ним заключены в один тег-контейнер с идентификатором gallery, включение же группы фотографий в другой обеспечивает более гибкое форматирование. Вы можете установить конкретную ширину фотогалереи в окне браузера или добавить ограничивающую рамку вокруг. В то же время заключительный тег также позволяет производить выборку фотографий и абзацев подписей, используя производные селекторы. В нашем случае, например, #gallery img и #gallery p — тоже действительные производные селекторы. Главное различие в этих двух подходах — приоритет стилей. Поскольку стиль #gallery img имеет большее значение по сравнению с .photo img, его команды форматирования отменяют и переопределяют параметры стиля .photo img.

   Теперь поместим фотографии рядом друг с другом.

ПРИМЕЧАНИЕ

При вставке внутренней таблицы стилей убедитесь в том, что вы поместили ее в заголовок веб-страницы, между тегом <link> и закрывающим </head>.

4. Добавьте в только что созданную таблицу стилей следующий стиль:

```
.figure {
  float: left;
  width: 210px;
  margin: 0 10px 10px 10px;
}
```

Он создает такое обтекание, при котором все пары «фотография/заголовок» выравниваются по левому краю окна браузера. На самом же деле браузер размещает фотографии на одном уровне, рядом друг с другом, пока не закончится свободное место в строке. Затем браузер переносит следующие изображения на строку ниже, пока не отобразит все, и т. д. Общая ширина складывается из ширины самой фотографии плюс отступы и границы-рамки. В данном примере имеем: 200 пикселов на фотографию, 8 пикселов на левый и правый отступы и 2 пиксела на левую и правую границы-рамки.

СОВЕТ

В нашей экспериментальной фотогалерее все изображения имеют одинаковую ширину. На практике же размеры различаются. Во врезке ниже описан способ компоновки изображений различных размеров. Изображения с различной высотой, конечно же, не будут отображаться правильно (вы увидите это в шаге 5). Если у вас есть изображения различной высоты, используйте метод с применением таблиц HTML (или можете использовать продвинутую, но работающую не во всех браузерах методику, описанную в следующем совете).

ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

Если ширина одного изображения отличается от остальных

Создать фотогалерею, которая похожа на галерею от CosmoFarmer, очень просто. Здесь все фотографии имеют одинаковую ширину. Но как быть, если у вас есть фото различных размеров? Одно из решений — создать стиль для каждой отдельно взятой ширины и применить его к элементу `<div>` со стилевым классом `figure` (это немалая работа, с таким же успехом можно отредактировать фотографии, приведя их ширину к одному размеру).

Другой способ состоит в использовании возможности CSS применить два стилевых класса к одному тегу: `<div class="figure w300">`. Этот тег `<div>` принадлежит как стилевому классу `figure`, так и классу `w300`.

Теперь создайте стилевой класс, например `.w300`, и определите ширину изображения (в данном случае 300) плюс 10 на отступы и границы: `.w300 { width: 310 }`. Чтобы этот прием работал, вы должны либо удалить параметр ширины в стиле `.figure`, либо добавить стиль `.w300` после `.figure` в таблице стилей. И вот почему: эти два значения ширины конфликтуют между собой (210 и 300) и браузер должен разрешить конфликт, используя правило каскадности (см. гл. 5). Поскольку и стилевой класс `.figure`, и класс `.w300` имеют одинаковый приоритет, будет применен тот стиль, который последним определен в таблице стилей.

5. Сохраните файл и просмотрите веб-страницу `gallery.html` в браузере. Она должна быть похожа на страницу, представленную на рис. 8.12, *слева*.
На рис. 8.12 метод работает неправильно, если элементы имеют различную высоту (см. рис. 8.12, *слева*). Здесь вам поможет свойство `height` для принудительной установки одинаковой высоты всех элементов и гарантии того, что все они выстроятся и отобразятся правильно (см. рис. 8.12, *справа*).

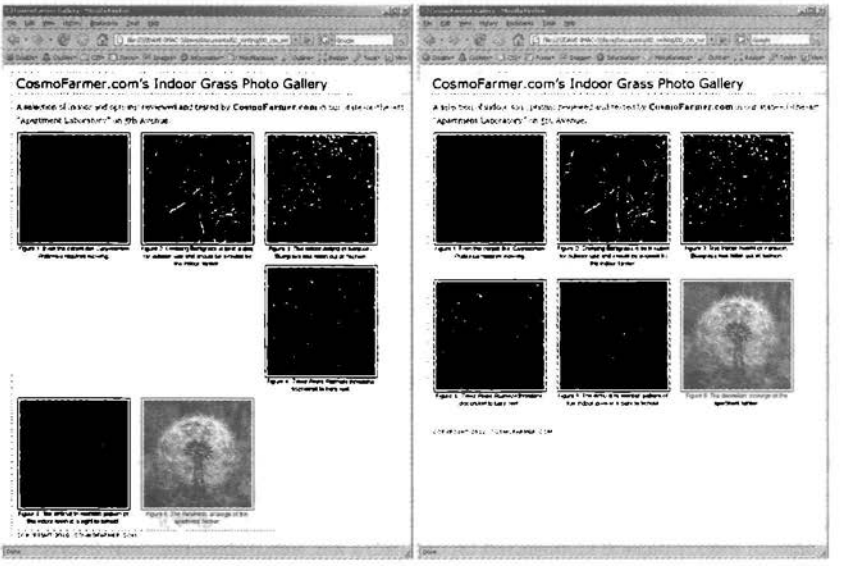


Рис. 8.12. Плавающие элементы, расположенные рядом друг с другом, — один из способов имитации столбцов и строк таблицы

Измените ширину окна браузера так, чтобы оно стало у́же или шире, и понаблюдайте, как перераспределяются изображения. Вы увидите, что не совсем все правильно. Во второй строке изображений имеется два пустых места, где должны располагаться фотографии. Эта проблема возникает по той причине, что подпись ко второму изображению на первой строке более высокая, чем другие на этой же строке. Изображения, которые переносятся на эту подпись, не могут разместиться рядом (об этой путанице, имеющей место при использовании свойства `float`, читайте в разд. «Управление обтеканием содержимого плавающих элементов» гл. 7). К счастью, существует простое решение этой проблемы.

СОВЕТ

Существует способ избежать этой проблемы со странным отображением, о которой говорилось в шаге 5, а также создать галерею, способную обрабатывать различные по высоте изображения. Вместо того чтобы использовать свойство float, вы можете указать display: inline-block в стиле .figure. Таким образом, каждая пара «изображение/надпись» будет обрабатываться как блок (область, имеющая высоту и ширину), а также в качестве встроенного элемента (поэтому блоки смогут располагаться бок о бок). Кроме того, вы можете использовать свойство vertical-align, чтобы выровнять рисунок вертикально. Стиль .figure из шага 4 можно переписать так:

```
.figure {
  display: inline-block;
  vertical-align: top;
  width: 210px;
  margin: 0 10px 10px 10px;
}
```

Недостатком этого очень простого и полезного приема является то, что он не будет работать в IE 6, IE 7 и Firefox 2. Зато он работает во всех остальных браузерах, в том числе и в IE 8!

6. Вернитесь к файлу gallery.html в HTML-редакторе. Найдите стиль .figure p и добавьте в него свойство height. В итоге все должно выглядеть следующим образом:

```
.figure p {
  font: 1.1em/normal Arial, Helvetica, sans-serif;
  text-align: center;
  margin: 10px 0 0 0;
  height: 5em;
}
```

Добавленное свойство устанавливает одинаковую высоту для всех подписей рисунков. В нашем случае размера достаточно, чтобы обеспечить правильное отображение строк текста (если требуется поместить в подписи больше текста, просто увеличьте значение высоты).

ПРИМЕЧАНИЕ

Совсем не обязательно применять свойство height, если вы уверены, что все плавающие элементы имеют одинаковую высоту. Это может понадобиться, если размеры фотографий различаются по высоте, подписи имеют разное количество строк текста или некоторые подписи отсутствуют.

7. Сохраните файл и воспользуйтесь предварительным просмотром веб-страницы в браузере. Посмотрите на правую часть рис. 8.12.

Если вы измените размеры окна браузера, то вид фотогалереи тоже изменится. В более широкое окно может вместиться четыре или даже пять изображений, но если размер уменьшить, вы увидите, что на одной строке отображается всего одно или два изображения.

Добавление теней. Наша фотогалерея выглядит хорошо, но можно придать ей еще более внушительный внешний вид. Добавление теневых эффектов для каждой фотографии придаст веб-странице иллюзию глубины и обеспечит реалистичность трехмерного пространства. Однако прежде, чем запускать программу для редактирования фотографий типа Photoshop, неплохо было бы узнать, что в этом нет никакой необходимости. Добавить теневые эффекты к любому изображению веб-страницы можно автоматически средствами CSS.

ПРИМЕЧАНИЕ

CSS 3 предлагает новое свойство box-shadow, которое может автоматически добавлять тени к любому стилю. Оно прекрасно настраивается, простое в использовании, но работает только в некоторых браузерах. Мы вернемся к нему позже в книге.

Сначала требуется найти соответствующий рисунок с размытыми правым и нижним краями черного цвета, который обеспечит теневой эффект. Пример имеется в папке 08\gallery_ex учебного материала, но вы можете создать свой собственный, например, в Photoshop, Fireworks или любой другой программе редактирования изображений, в которой есть фильтр эффекта тени или размытости. В Fireworks, к примеру, вам нужно создать белую область и применить к ней фильтр эффекта тени, а затем сохранить в формате PNG8.

1. Откройте в HTML-редакторе файл gallery.html, над которым вы работали в предыдущей части практического урока.

Во-первых, добавьте фоновое изображение в тег , в который заключено каждое изображение.

2. Добавьте следующий стиль в таблицу стилей веб-страницы фотогалереи:

```
.photo {
  background: url(drop_shadow.gif) right bottom no-repeat;
}
```

Этот стиль добавляет фоновый рисунок drop_shadow.gif в нижний правый угол элемента фотографии. Параметр no-repeat свойства означает, что изображение не должно повторяться.

Если вы взглянете на веб-страницу, то не увидите ничего особенного, потому что тень находится на заднем плане. Наверху, на переднем плане, расположена сама фотография, которую мы стилизовали в шаге 3: с белым фоном, черной границей-рамкой и отступами размером 4 пиксела. Теперь все, что нужно, — это показать фоновое изображение.

С помощью одной хитрой методики, придуманной Ричардом Руттером (http://www.clagnut.com), нужно немного сместить изображение вверх и влево — по сути, вытащить его за пределы тега-контейнера . CSS предлагает механизм, называемый *позиционированием*, который позволяет вам управлять точным местоположением, или размещением, элемента. Подробно о механизме позиционирования вы узнаете из гл. 13, а пока, чтобы показать тень, вы должны добавить три свойства в стиль .photo img, созданный в шаге 3.

ПРИМЕЧАНИЕ

Отрицательные значения полей — еще один способ достижения сдвига тени. Детальное описание представлено по адресу http://1976design.com/blog/archive/2003/11/14/shadows/.

3. Найдите стиль .photo img и добавьте в него три свойства позиционирования следующим образом:

```
.photo img {
  border: 1px solid #666;
  background-color: #FFF;
  padding: 4px;
  position: relative;
  top: -5px;
  left:-5px;
}
```

По сути, эти три свойства просто смещают фотографию вверх и влево на 5 пикселов, показывая находящийся на заднем плане рисунок с теневым эффектом элемента . В данном случае используется в качестве элемента-контейнера изображения с эффектом тени.

4. Сохраните файл и просмотрите веб-страницу в браузере. Она должна иметь вид, представленный на рис. 8.13.

Каждое изображение отображается со своей собственной тенью, и вам не пришлось даже запускать Photoshop!

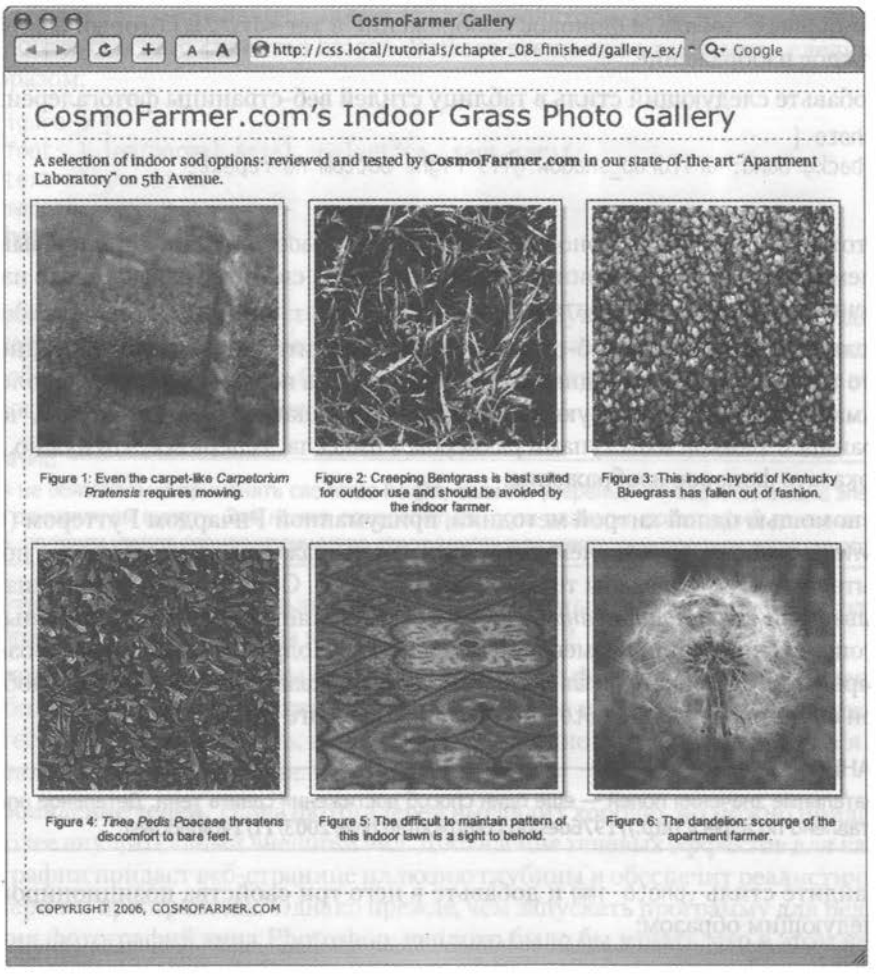


Рис. 8.13. Добавление теневого эффекта к фотографиям придает веб-странице объемность и улучшает визуальное восприятие любой фотогалереи

ПРИМЕЧАНИЕ

Для создания теневого эффекта мы использовали рисунок размером около 375 × 375 пикселов, и он подходит для фотографий такого же размера. Вы можете использовать эту методику и для больших изображений, просто придется создать соответствующий теневой эффект.

Вы можете применять описанный метод добавления теневого эффекта не только в фотогалерее, но и для любого изображения веб-страницы. Весь замысел состоит в следующем: необходимо заключить `<img>` с изображением в элемент-контейнер `<div>`, назначить для него фоновый рисунок теневого эффекта и немного сместить тег изображения `<img>` влево и вверх с помощью отрицательных значений параметров позиционирования. Пользуйтесь этим методом для добавления теневого эффекта к любому блочному элементу, например боковому меню или плавающей цитате.

Законченную версию веб-страницы этой обучающей программы вы сможете найти в папке 08_finished\gallery_ex учебного материала.

ПРИМЕЧАНИЕ

Наверняка вы обратили внимание на то, что созданные рисунки теней имеют резкое завершение с левой и верхней стороны. Они не имеют плавного перехода, как настоящие тени. Процесс создания более сложных, реалистичных теневых эффектов описан в Интернете по адресу http://www.alistapart.com/articles/cssdrop2/ и http://www.ploughdeep.com/onionskin/.

Обучающий урок 3: использование фоновых изображений

CSS-свойство background-image — секретное оружие современного веб-дизайна. Оно превращает скучный текст в великолепный графический образ. Вы можете использовать его для добавления фоновых изображений к любому HTML-тегу, и дизайн, который сумеете создать, ограничен только воображением. Пример теневого эффекта, приведенного в предыдущем обучающем уроке, — всего лишь один из способов креативного использования фоновых изображений. Основное их применение — в качестве общего фона веб-страницы, а также для создания собственных маркеров в списках. Эти общие, наиболее распространенные способы применения свойства background-image мы и рассмотрим в этом обучающем уроке.

Добавление на веб-страницу фонового изображения

Что бы это ни было: сложный замысловатый узор, логотип компании или полноэкранная фотография, — изображения очень часто используются в качестве фоновых рисунков веб-страниц. И неслучайно фактически это и есть основное применение свойства background-image.

1. Откройте файл sidebar.html из папки 08\bg_ex в HTML-редакторе.

   Веб-страница имеет двухстолбцовую разметку: она очень проста, содержит лишь немного отформатированного текста на белом фоне (рис. 8.14, *вверху*). Для начала добавим фоновое изображение. У страницы есть внешняя таблица стилей с базовым форматированием, но, поскольку нет необходимости пробираться через все стили в этом файле, добавим внутреннюю таблицу стилей для шагов этого примера.

2. Установите курсор между открывающим и закрывающим тегами <style> и добавьте следующий стиль:

```
body {
  background-image: url(images/bg_page.png);
  background-repeat: repeat-x;
  background-color: #FFF;
}
```

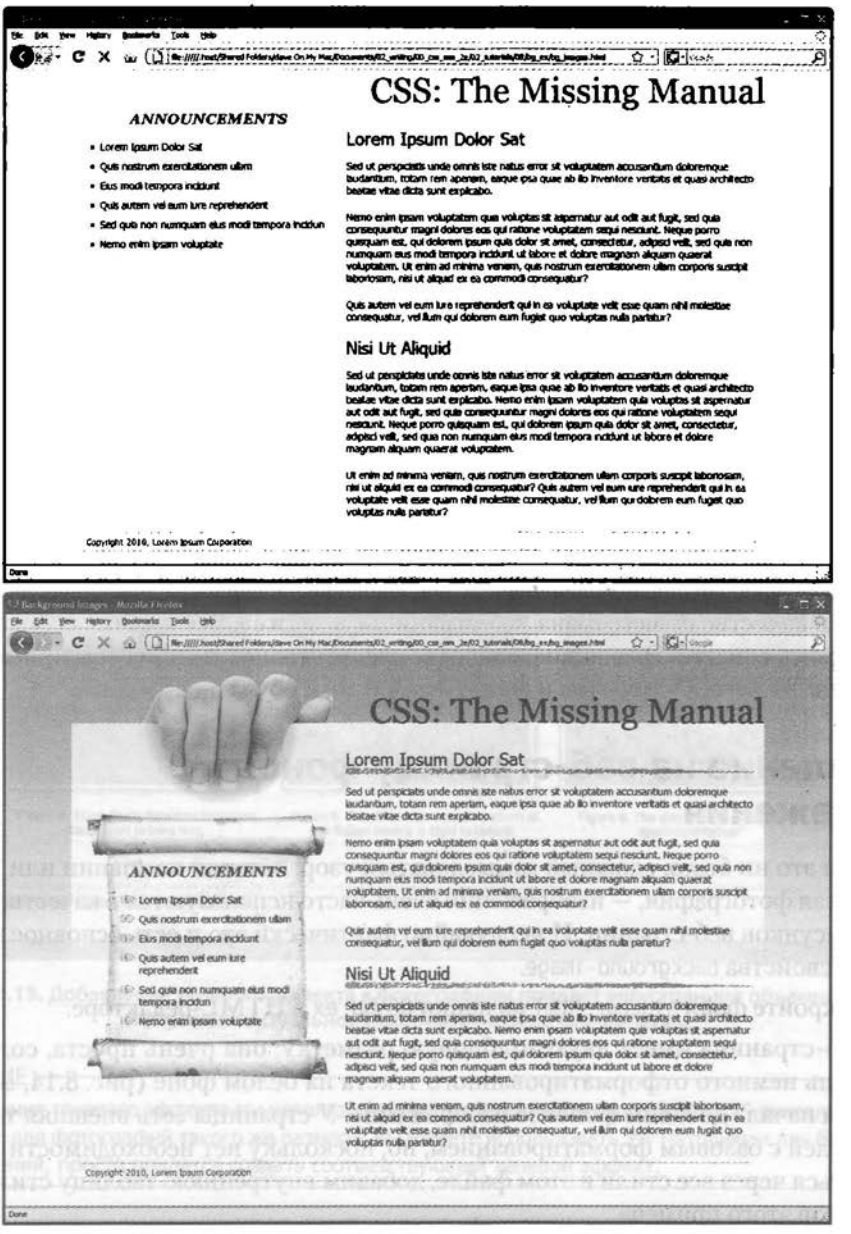


Рис. 8.14. С помощью фоновых изображений вы можете превратить упорядоченную веб-страницу *(вверху)* в еще более привлекательный сайт *(внизу)*

Первая строка стиля определяет само фоновое изображение `page_bg.jpg`, которое мы хотим вывести на веб-странице. Изображение находится в папке **images**. Оно представляет собой градиент, начинающийся светло-синим цветом вверху и плавно переходящий в белый внизу. Рисунок имеет размеры гораздо меньше

содержимого веб-страницы и без последующих команд будет многократно повторяться по ширине и высоте. Через определенный интервал по направлению сверху вниз, равный высоте фонового рисунка, ярко-зеленый цвет появится вновь и опять постепенно перейдет в белый, образовав такой же градиент. Чтобы предотвратить эту нелицеприятную картину, установим такое значение свойства background-repeat, чтобы изображение повторялось в направлении слева направо и заполняло по ширине все окно браузера независимо от его размеров, но при этом вертикально отображалось всего один раз. Последняя строка устанавливает цвет фона, чтобы он соответствовал концу градиента и изображение плавно исчезало в цвете фона страницы.

ПРИМЕЧАНИЕ

Используя сокращенный вариант свойства, вы можете превратить три строки кода, показанные в шаге 2, в одну строку:

```
background: #FFF url(images/bg_page.png) repeat-x;
```

3. Сохраните файл и просмотрите его в браузере.

   Синий градиент фонового изображения по-прежнему повторяется на веб-странице сверху вниз. Смотрится неплохо, но синий цвет есть и в фоне текста. Вы можете приукрасить текст, задав для него другой фоновый цвет.

4. Вернитесь к текстовому редактору и файлу bg_images.html. Добавьте другой стиль для тега , в котором находится содержимое страницы:

```
#wrapper {
  background-color: #FFF;
}
```

   У данного тега есть фиксированная ширина, он центрирован на странице и содержит весь ее текст. Этот стиль задает ему белый фоновый цвет, но с помощью изображения вы можете сделать его лучше.

5. Отредактируйте стиль, который вы создали в шаге 4, добавив фоновое изображение:

```
#wrapper {
  background-color: #FFF;
  background-image: url(images/bg_main.jpg);
  background-position: top left;
  background-repeat: no-repeat;
}
```

   Эти три строки кода добавляют фоновое изображение в левом верхнем углу ; параметр no-repeat для свойства background-repeat означает, что изображение появляется только один раз. Если вы сохраните файл и просмотрите его в браузере, то увидите изображение руки, которая как бы держит эту страницу. Очень здорово. Но единственная проблема заключается в том, что текст находится слишком высоко вверху и прикрывает изображение. Далее вы разместите на странице большой заголовок и левую боковую панель.

6. Добавьте еще два стиля к внутренней таблице стилей:

```
#banner {
  margin-top: 48px;
}
#announcement {
  margin-top: 115px;
}
```

Первая строка просто добавляет небольшие отступы, отталкивающие вниз баннер, содержащий заголовок, пока он не станет соприкасаться с белой страницей, а второй стиль передвигает вниз левую боковую панель, чтобы освободить достаточно места для изображения с рукой. Страница должна выглядеть так, как показано на рис. 8.15.

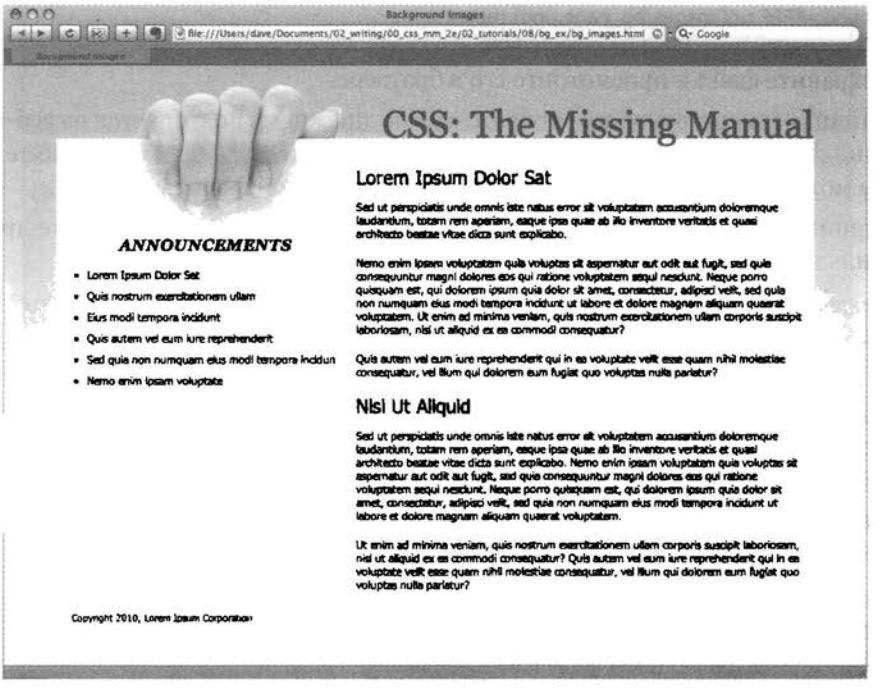


Рис. 8.15. CSS позволяет комбинировать цвет фона и фоновое изображение, что оказывается весьма полезным в настоящем примере

Замена границ изображениями

Свойство `border` (см. разд. «Добавление границ» гл. 7) — полезный инструмент в арсенале веб-дизайна, но ограниченность предлагаемых CSS стилей границ быстро надоедает. Линия, нарисованная карандашом от руки, привлечет внимание посетителей сайта гораздо больше, нежели прямая черная. Можете смело проигнорировать свойство `border` и добавить на свой вкус любую линию в виде фоново-

го рисунка — это очень просто. В настоящем обучающем уроке мы заменим линии подчеркивания под тегами заголовков <h2> собственным рисунком, похожим на линию, нарисованную от руки.

1. Вернитесь к файлу bg_images.html в текстовом редакторе. Добавьте стиль для тега <h2> внутри основного :

```
#main h2 {
  background-image: url(images/underline.png)
  background-repeat: no-repeat;
}
```

Свойство background-image назначает фоновое изображение для тегов <h2>, находящихся внутри тега с идентификатором main; а значение no-repeat гарантирует, что изображение появится только один раз.

Если сейчас вы просмотрите файл в браузере, то увидите, что рисунок подчеркивания расположен не там, где ему положено быть. Он находится над заголовками!

2. Добавьте в стиль #main h2 за свойством background-repeat следующее:

```
background-position: left bottom;
```

Мы изменили начальную позицию фонового изображения. Теперь оно выводится, начиная от левого нижнего угла тега <h2>. Однако при просмотре веб-страницы вы не заметите серьезных улучшений: подчеркивание сливается с текстом заголовка.

Есть простое решение. Поскольку нижняя координата фонового рисунка расположена у основания блока, образуемого тегом <h2>, необходимо всего лишь увеличить его общую высоту, чтобы сместить линию фонового рисунка немного вниз. Для этого воспользуемся небольшим отступом.

3. Подкорректируйте стиль #main h2 еще раз, чтобы он выглядел следующим образом:

```
#main h2 {
  background-image: url(images/underline.png);
  background-repeat: no-repeat;
  background-position: left bottom;
  padding-bottom: 7px;
}
```

Как вы помните, отступ является промежутком между границей (или краем фона) и содержимым. Это также увеличивает суммарную высоту блока — в данном случае добавляется 7 пикселов нижнего отступа. Теперь граница изображения находится в нижней части блока h2.

4. Сохраните файл и просмотрите веб-страницу в браузере.

Теперь все теги <h2> имеют рукописную линию подчеркивания. Займемся блоком бокового меню, сделаем его менее угловатым и плоским, а также улучшим вид маркированных списков.

Использование графики для маркированных списков

Стандартный маркер, используемый для маркированных списков, представляет собой черное пятнышко, что совсем не впечатляет. Но вы можете создать свои собственные маркеры путем применения свойства background-image, заменив однообразные и скучные значки любым изображением. Первое, что необходимо сделать, — скрыть стандартные маркеры, которые предваряют элементы-пункты списка.

1. Вернитесь к веб-странице sidebar.html в HTML-редакторе. Добавьте стиль для форматирования списка пунктов левой боковой панели:

```
#announcement li {
  list-style: none;
}
```

Маркированный список находится внутри тега с идентификатором announcement, так что этот производный селектор воздействует только на пункты списка (теги <li>) внутри этого . Этот стиль удаляет маркеры. Теперь добавим наш рисунок.

ПРИМЕЧАНИЕ

Вы можете точно так же применить свойство list-style: none; к стилям тегов <ul> или <ol>, чтобы удалить маркеры всех элементов-пунктов списка.

2. Добавьте в стиль #announcement li следующие два свойства:

```
background-image: url(images/flower_bullet.png);
background-repeat: no-repeat;
```

Мы встречались с ними раньше. Одно из них добавляет фоновое изображение, а другое — отменяет его повторение, чтобы рисунок отображался однократно.

При просмотре веб-страницы вы увидите, что сейчас маркеры накладываются на текст элементов списка и пункты списка отображаются очень тесно друг к другу (рис. 8.16, *слева*). Небольшие отступы и поля исправят эту проблему.

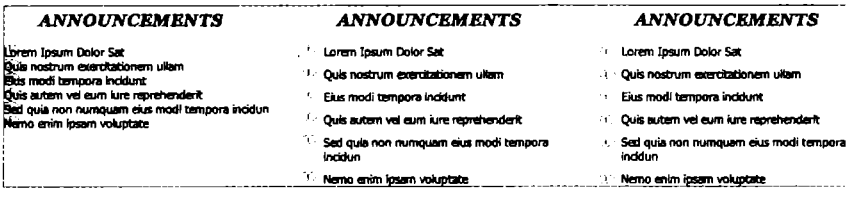


Рис. 8.16. Заменить стандартные маркеры списков своими собственными графическими значками чрезвычайно просто: всего несколько дополнительных шагов обеспечат точное и правильное размещение маркеров и текста пунктов списка

3. Добавьте в стиль #announcement li еще два свойства:

```
padding-left: 25px;
margin-bottom: 10px;
```

Левый отступ добавляет пустой промежуток, смещая текст в сторону, чтобы отобразить новый значок маркера. Нижнее поле обеспечивает рассредоточение элементов списка, образуя небольшие интервалы (см. рис. 8.16, *посередине*).

Однако остался еще один недостаток. Изображение маркера чуть выступает над текстом строки, и значок выделяется над текстом пунктов списка. Это легко исправить с помощью свойства `background-position`.

4. Завершим этот стиль, добавив свойство `background-position: 0px 4px;`. Законченный стиль должен выглядеть следующим образом:

```
#announcement li {
  list-style: none;
  background-image: url(images/bullet.png);
  background-repeat: no-repeat;
  background-position: 0 4px;
  padding-left: 25px;
  margin-bottom: 10px;
}
```

Это последнее изменение стиля позиционирует значок маркера в крайнюю левую позицию (это 0), отстоящую на 4 пиксела от верхнего края элемента-пункта списка, что обеспечивает совсем незначительное смещение значка, тем не менее достаточное для того, чтобы маркер выглядел безупречно.

СОВЕТ

Я рекомендую использовать именно свойство background вместо list-style-image для добавления собственных графических маркеров списков. Оно обеспечивает возможность точного позиционирования значков маркеров.

5. Сохраните файл и просмотрите веб-страницу в браузере.

Теперь пункты списка имеют трехмерные маркеры вместо скучных черных кружков (см. рис. 8.16, *справа*).

Закругление углов бокового меню

На данный момент боковое меню выглядит довольно неплохо. Текст приятно отформатирован, маркеры отлично смотрятся, но боковая панель теряется на белом фоне. Добавление фонового изображения позволит заметно выделить боковую панель.

Вы можете использовать одно изображение в виде свитка, показанное на рис. 8.14, в фоне тега `<div>`. А для того, чтобы убедиться, что весь текст попадает в данное изображение, вам следует ограничить количество содержимого, которое вы размещаете на боковой панели. Если будет слишком много текста, он просто не поместится в пределах изображения; и наоборот, если будет слишком мало текста, у вас окажется много пустого пространства. Более гибкий подход позволяет изображению увеличиваться по мере того, как на боковой панели увеличивается количество текста (рис. 8.17, *вверху*). Хорошо, что эту маленькую хитрость не так сложно реализовать — вам понадобится три разных изображения и три разных стиля.

В этом примере есть один с идентификатором announcement (это боковая панель), содержащий тег <h2> (со словом «Announcements») и маркированный список (тег <ul>). По существу, вы присоединяете верхнюю часть изображения к HTML-элементу вверху боковой панели (тег <h2> в данном примере), а нижнюю часть изображения — к последнему HTML-элементу боковой панели (тег <ul>), и изображение повторяется вертикально в теге , который создает боковую панель (см. рис. 8.17, *внизу*).

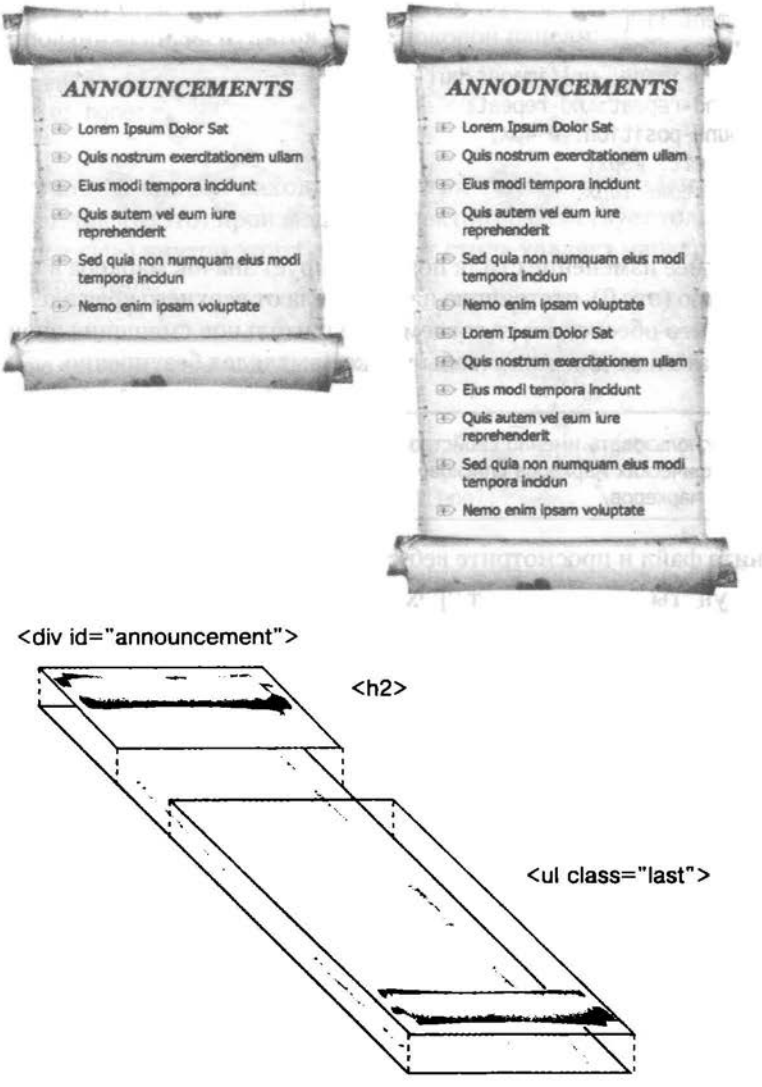


Рис. 8.17. Многие изображения на страницах, которые вы видите каждый день, выглядят как одно целое, но на самом деле составлены из различных отдельных рисунков, расположенных специальным образом в фонах тегов

1. Вернитесь к текстовому редактору и файлу bg_images.html. Перейдите к стилю #announcement (который вы создали в шаге 6 в подразделе «Добавление на веб-страницу фонового изображения») и добавьте одно свойство:

```
#announcement {
  background: url(images/scroll_middle.jpg) repeat-y center top;
  margin-top: 115px;
}
```

Новая строка добавляет к тегу фоновое изображение, повторяющееся по вертикали, и центрирует его в этом теге. Изображение повторяется как единое целое, и, если становится выше, фон также вырастает. Но если вы просмотрите страницу сейчас, то увидите, что маркированный список выдается как с левой, так и с правой стороны изображения бокового меню. Сделаем так, чтобы он соответствовал этому изображению, добавив немного левых и правых полей.

2. Перейдите к стилю #announcement li, созданному ранее, и добавьте два свойства в конце:

```
#announcement li {
  list-style: none;
  background-image: url(images/bullet.png);
  background-repeat: no-repeat;
  background-position: 0 4px;
  padding-left: 25px;
  margin-bottom: 10px;
  margin-left: 30px;
  margin-right: 40px;
}
```

Эти свойства перемещают левый и правый края каждого пункта маркированного списка на достаточное расстояние, чтобы очистить края фонового изображения. Далее вы добавите верхнюю часть свитка, размещая фоновое изображение в теге <h2> на боковой панели.

3. Внизу внутренней таблицы стилей добавьте следующий наследуемый селектор к стилю тега <h2> бокового меню:

```
#announcement h2 {
  background: url(images/scroll_top.jpg) no-repeat center top;
}
```

Здесь изображение помещается только один раз в центре в верхней части тега. Но, просматривая страницу прямо сейчас, вы увидите, что она выглядит не совсем правильно (рис. 8.18). Вы не можете видеть всю верхнюю часть свитка, и текст заголовка перекрывает ее нереальным образом. По существу, тег <h2> недостаточно высок, чтобы отобразить все изображение. Кроме того, вам нужно добавить некоторое пространство над текстом, чтобы опустить его вниз, под изображение. Вам на помощь придет свойство padding.



Рис. 8.18. Если вы не видите все изображение (верхнюю часть свитка здесь), это значит, что элемент недостаточно большой. Вы можете добавить отступы, чтобы получить дополнительное пространство для целого изображения

4. Отредактируйте стиль, который вы добавили в предыдущем шаге, чтобы он выглядел так:

```
#announcement h2 {
  background: url(images/scroll_top.jpg) no-repeat;
  padding-top: 70px;
}
```

Верхний отступ размером 70 пикселов выполняет две задачи: оттягивает текст вниз, освобождая верхний край свитка, и делает элемент <h2> таким высоким, чтобы его было достаточно для показа всего изображения. На самом деле заголовок Announcements не располагается наверху фонового изображения стиля — он находится наверху фонового изображения тега .

ПРИМЕЧАНИЕ

Метод, используемый здесь, предназначен для блока с фиксированной шириной. Это значит, что, если вы измените ширину боковой панели, вам необходимо переделать изображение в соответствии с новой шириной. Чтобы узнать о более гибких методах, позволяющих создавать закругленные углы и требующих большего количества кода CSS и HTML, посетите следующие страницы: www.vertexwerks.com/tests/sidebox и www.sperling.com/examples/box.

Далее нужно добавить нижнее изображение к тегу <ul>. Но сначала мы немного изменим HTML-код.

5. Перейдите к тегу <ul> (он находится под тегами и <h2>) и добавьте class="last":

```
<div id="announcement">
<h2>Announcements</h2>
<ul class="last">
```

Зачем добавлять класс? Как показано на рис. 8.17, нижнее изображение (конец свитка) должно быть прикреплено к последнему тегу в боковой панели; таким образом, изображение появляется в нижней части свитка. Теперь вы можете просто создать стиль, такой как #announcement ul, который будет работать в данном случае. Но если вы хотите удалить маркированный список и заменить его

абзацами текста, вам нужно повторно создать стиль. Используйте класс и создайте стиль, например, #announcement .last. Тогда всякий раз, когда будете изменять HTML в боковой панели, вы просто добавите class="last" к последнему тегу в боковой панели (можете сделать то же самое для тега <h2>; скажем, создать класс под названием #announcement .first и добавить class="first" к тому тегу, который идет первым внутри).

6. Во внутренней таблице стилей страницы добавьте еще один последний стиль:

```
#announcement .last {
  background: url(images/scroll_bottom.jpg) no-repeat center bottom;
  padding-bottom: 65px;
}
```

Как и в других стилях в этой части издания, здесь вы добавляете отдельное изображение в фон элемента. Однако в данном случае оно размещается в нижней части тега <ul>. Поскольку у нас есть немало элементов списка, тег <ul> на самом деле довольно высокий и стал выше, чем фоновое изображение. Следовательно, вам необходимо поместить его в нижней части тега, чтобы оно отображалось внизу тега .

7. Сохраните файл и просмотрите его в браузере.

Боковая панель должна выглядеть так, как показано на рис. 8.14, *внизу*. Это верно, только если вы не будете использовать IE 6 или IE 7. В этих браузерах вы не увидите ничего, кроме повторяющегося фона, — ни заголовка, ни маркеров, ни верхней или нижней частей свитка. Это еще одна ошибка браузера IE, которая, к счастью, была исправлена в восьмой версии. Существует способ справиться с ней и в IE 6 и IE 7.

8. Вернитесь к файлу bg_images.html и добавьте свойство zoom: 1 к стилю #announcement во внутренней таблице стилей. Конечная версия должна выглядеть так:

```
#announcement {
  background: url(images/scroll_middle.jpg) repeat-y center top;
  margin-top: 115px;
  zoom: 1;
}
```

Это странное свойство, работающее только в IE, добавляет то, что называется *разметкой*, и устраняет как эту проблему, так и многие подобные. Это небольшое волшебство CSS, которое, по идее, не должно ничего делать, но делает (за что мы любим веб-дизайн)!

Двигаемся дальше

Страница готова, но в качестве дополнительного задания переместите внутреннюю таблицу стилей, которую вы включили в этом примере, во внешнюю (styles.css), присоединенную к данной странице. Один из способов сделать это — просто вырезать стили из внутренней таблицы и вставить их во внешнюю. Тем не менее

в некоторых случаях стили с одинаковыми именами появляются в обеих таблицах стилей (внешняя таблица, например, содержит стиль #announcement, который применяется для представления информации о разметке боковой панели). Попытайтесь в конечном итоге получить внешнюю таблицу стилей, в которой не повторяются имена стилей. Вы можете сделать это путем копирования свойств из внутренней таблицы (например, скопируйте свойства #announcement) и вставьте их в соответствующие стили внешней таблицы (так, вставьте свойства в стиль #announcement в файле styles.css).

Вы найдете готовые обучающие уроки — как версию с двумя таблицами стилей, так и версию с одной внешней таблицей — в папках 08_finished/bg_ex и 08_finished/bg_ex_further.

9 Приводим в порядок навигацию сайта

Можно с уверенностью сказать, что без ссылок не было бы Интернета. Возможность находиться на одной веб-странице, а затем, щелкнув кнопкой мыши, перейти на другую страницу, расположенную на компьютере на другом конце света, — вот что делает Сеть такой полезной и незаменимой. Ссылки предоставляют своеобразный способ навигации и управления содержимым сайта. Именно поэтому дизайнеры веб-страниц прилагают столько усилий для создания привлекательных ссылок, работающих должным образом.

В этой главе вы познакомитесь с разработкой стилей ссылок, позволяющих им визуально выделяться из общего содержимого страниц. Вы узнаете, как обеспечить визуальные подсказки, чтобы посетители сайта могли видеть, какие ссылки они уже посещали, а какие — нет. Я также научу вас созданию на веб-страницах средствами CSS кнопок и панелей навигации, как это делают профессиональные разработчики. И в заключение, в практической части, мы создадим полный набор средств — элементов навигации, одинаково хорошо работающих во всех разновидностях браузеров.

Выборка стилизуемых ссылок

В CSS, как обычно, сначала нужно выбрать тот объект, для которого вы хотите определить стиль. Для этого надо сообщить CSS не только, *что* стилизовать, то есть конкретный объект, но и *условия* стилизации. Браузеры отслеживают процесс взаимодействия посетителя сайта со ссылками и затем отображают их по-разному в зависимости от статуса. Таким образом, применяя селекторы в CSS, вам предоставляется возможность адресовать стили к ссылкам в определенном состоянии.

Понимание состояний ссылок

Большинство браузеров распознают четыре основных состояния ссылок:

- непосещенная ссылка;
- посещенная ссылка (это означает, что по ссылке уже выполнялся переход, то есть URL-адрес сохранен в журнале истории браузера);
- ненажатая (над которой находится указатель мыши);
- нажатая ссылка (удерживаемая мышью).

Как описано в гл. 3, CSS предоставляет четыре псевдокласса селекторов, соответствующих этим состояниям: `:link`, `:visited`, `:hover`, `:active`. Используя их, вы можете применить различное форматирование для каждого состояния ссылки, обеспечивая соответствующие подсказки и однозначно давая понять посетителю сайта, какие ссылки он уже посещал, а какие — нет.

ПРИМЕЧАНИЕ

Браузеры Internet Explorer 8, Opera, Firefox и Safari также поддерживают псевдокласс :focus. Ссылки получают признак фокусировки :focus, когда посетитель вместо мыши пользуется клавиатурой для перехода по ссылкам с помощью табулятора. Этот псевдокласс также оказывается полезным при работе с заполняемыми текстовыми полями форм, как описано в обучающем уроке 2 гл. 10.

Предположим, вы хотите изменить цвет текста непосещенной ссылки с синего на ярко-оранжевый. Для этого добавьте следующий стиль:

```
a:link { color: #F60; }
```

Щелкаем кнопкой мыши на этой ссылке, и ее состояние изменяется на посещенное, а цвет в большинстве браузеров становится фиолетовым. Чтобы изменить его на насыщенно-красный, примените такой стиль:

```
a:visited { color: #900; }
```

СОВЕТ

Если требуется одинаково отформатировать все состояния ссылок, используйте один и тот же шрифт и размер для всех состояний, затем отформатируйте тег <a>, создав общий селектор a. Далее вы сможете использовать конкретные состояния ссылок, например a:visited, для изменения цвета или настройки какого-либо конкретного состояния.

Псевдокласс `:hover` предоставляет творческую свободу (мы изучим его в этой главе несколько позже). Он позволяет полностью изменить вид гиперссылки при наведении указателя мыши и перемещении его над ссылкой. Если раньше для визуального представления кнопок навигации при наведении на них указателя мыши вы применяли объемный код на языке JavaScript, то вам понравится способ достижения такого же эффекта посредством простейшей строки кода CSS (см. разд. «Стилизация ссылок» данной главы). Для начала рассмотрим простой пример, в котором стиль изменяет цвет ссылки при наведении и перемещении указателя мыши:

```
a:hover { color: #F33; }
```

СОВЕТ

Будьте внимательны при добавлении свойств CSS к псевдоклассу :hover. Свойства, изменяющие размер элемента, на который наведен указатель мыши, могут повлиять на другие элементы, находящиеся рядом. Например, если вы увеличиваете размер шрифта текстовой ссылки, на которую наведен указатель мыши, то при наведении указателя текст будет увеличиваться, выталкивая другие элементы с их позиций. Результат может оказаться раздражающим.

Теперь рассмотрим пример специально для «одержимых» веб-дизайнеров, которым нравится делать свои страницы непохожими на остальные. Изменим вид гиперссылки на доли миллисекунд, то есть на момент, когда посетитель щелкает кнопкой мыши на ссылке. Для этого напишем следующий стиль:

```
a:active {color: #B2F511; }
```

В большинстве случаев для обеспечения максимальной гибкости дизайна требуется включить в таблицу стилей псевдоклассы :link, :visited: и :hover. Но для того, чтобы этот метод работал, вы должны расположить ссылки в определенной последовательности: link, visited, hover и active. Для запоминания этого порядка запомните аббревиатуру по первым буквам соответствующих стилям английских слов: **LoVe/HAte** (в переводе «любовь/ненависть»). Ниже представлен способ добавления всех четырех стилей ссылок:

```
a:link { color: #F60; }
a:visited { color: #900; }
a:hover { color: #F33; }
a:active {color: #B2F511; }
```

Если вы измените последовательность, то состояния стилей hover и active перестанут функционировать. Например, если поместить a:hover перед a:link и a:visited, то при наведении на ссылку указателя мыши ее цвет не изменится.

ПРИМЕЧАНИЕ

Почему последовательность имеет такое значение? Здесь работает механизм каскадности (см. гл. 5). Все эти стили имеют одинаковый приоритет, значит, порядок их следования в программном коде отмечает более явно определенный стиль, который будет применен в конечном счете. Поскольку ссылка одновременно может принимать состояния unvisited (непосещенная ссылка) и hovered (ссылка, на которую показывает указатель мыши), то a:link, будучи указанным последним, согласно правилам каскадности имеет больший вес (приоритет), и a:hover с заменяющим цветом никогда не будет применен.

Выборка отдельных ссылок

Стили, которые мы создавали в предыдущем разделе, представляют собой простейшие стили ссылки <a>. Они производили выборку в определенном состоянии, но при этом стилизовали абсолютно все ссылки, независимо от их расположения на веб-странице. Что же делать, если вы хотите отформатировать одни ссылки одним способом, а другие — другим? Существует простое решение — применить стилевые классы к нужным тегам.

Допустим, на веб-странице имеется набор ссылок и некоторые из них указывают на другие сайты, которые вы хотите выделить (например, сайты ваших друзей, партнеров, спонсоров). Возможно, вы захотите их отформатировать таким образом, чтобы посетители заранее знали, что это какие-то особенные ссылки, и наверняка захотели бы перейти по ним. В данном случае можете применить стилевой класс:

```
<a href=http://www.hydroponicsonline.com
  class="external">Visit this great resource</a>
```

Для стилизации этой ссылки другим способом создадим следующий стиль:

```
a.external:link { color: #F60; }
a.external:visited { color: #900; }
a.external:hover { color: #F33; }
a.external:active {color: #B2F511; }
```

Определения только имени стилевого класса, без указания названия тега <a>, будет тоже достаточно:

```
.external:link { color: #F60; }
.external:visited { color: #900; }
.external:hover { color: #F33; }
.external:active {color: #B2F511; }
```

Теперь такое форматирование приобретут лишь ссылки, принадлежащие стилевому классу `'external'`.

ПРИМЕЧАНИЕ

В приведенных примерах в демонстративных целях мы изменили только цвет ссылок. На практике для форматирования вы можете использовать любые CSS-свойства. В следующем примере вы увидите, что существует множество способов креативной стилизации.

Группирование ссылок с помощью селекторов потомков. Если набор ссылок располагается в одной области веб-страницы, то вы можете сэкономить время, применив селекторы потомков. Предположим, у вас есть пять ссылок, которые ведут на основные разделы вашего сайта.

Они представляют собой главную панель навигации, и вы хотите придать им характерный вид. Просто заключите эти ссылки в тег и примените к нему стилевой класс или стиль с идентификатором: . Теперь появилась возможность произвести выборку и отформатировать только эти ссылки:

```
#mainNav a:link { color: #F60; }
#mainNav a:visited { color: #900; }
#mainNav a:hover { color: #F33; }
#mainNav a:active {color: #B2F511; }
```

Использование селекторов потомков облегчает процесс стилизации ссылок, которые должны выглядеть по-разному для каждой области веб-страницы (полное описание неисчерпаемых возможностей таких селекторов вы найдете в разд. «Использование селекторов потомков» гл. 15).

СОВЕТ

Использование маркированных списков для представления ссылок — очень распространенный метод (скоро вы увидите пример этого). В таком случае вы можете добавить идентификатор или класс к тегу <ul> списка, например <ul id="mainNav">, а затем создать селекторы потомков наподобие #mainNav a:link для их стилизации.

Стилизация ссылок

Теперь, когда вы знаете, как создавать селекторы, производящие выборку конкретных ссылок, нужно решить, как стилизовать ссылки. Да как угодно! Вам предоставляется полный арсенал свойств языка CSS, а творческие способности ограничены только воображением. Единственное, что нужно учесть, — ваши ссылки должны выглядеть как ссылки. Они не обязательно будут синего цвета с подчеркиванием, но пусть они отличаются от общего содержимого веб-страниц, чтобы посетители сайта сразу могли понять, что это.

Если сделать ссылку похожей на кнопку, у которой появляются границы и изменяется цвет фона при наведении на нее указателя мыши, большинство посетителей поймут, что они могут щелкнуть здесь. Ссылки, расположенные в объемных фрагментах текста, должны четко выделяться. Этого можно добиться при использовании текста полужирного начертания, сохранив традиционное подчеркивание, а также окрасив фон, добавив стиль-подсказку, изменяющий внешний вид ссылки при перемещении над ней указателя мыши. В качестве подсказки можно даже назначить рисунок (например, в виде стрелки), который визуально скажет посетителю о том, что щелчок кнопкой мыши перенесет его в какой-либо другой раздел текущего сайта или вообще на другой сайт.

СОВЕТ

Если вы не назначите свойству border тега <img> значение 0, то большинство браузеров добавит границу-рамку вокруг изображения, выступающего ссылкой. Чтобы этого не произошло, добавьте в таблицу стилей следующее: img { border: none;}.

Подчеркивание ссылок

Еще с самого начала развития Сети ярко-синий подчеркнутый текст сигнализировал: «Нажмите здесь, чтобы перейти туда-то». Однако стандартные атрибуты подчеркивания и цветового оформления ссылок — это то, что в первую очередь изменяет любой веб-дизайнер. Подчеркивание — слишком распространенный способ выделения, который порядком поднадоел (рис. 9.1, *1*). У вас есть возможность изменить ситуацию, одновременно обеспечив хорошее оформление ссылок.

- **Полное удаление подчеркивания.** Чтобы убрать стандартное подчеркивание, используйте свойство `text-decoration` со значением `none`:

  ```
  a {text-decoration: none;}
  ```

  Конечно, полное удаление подчеркивания может смутить посетителей сайта. Если вы не предусмотрите других визуальных подсказок, то ваши ссылки будут выглядеть точно так же, как и весь остальной текст веб-страницы (см. рис. 9.1, *2*). Если вы пойдете этим путем, то обеспечьте выделение текста ссылок каким-то другим способом, например полужирным начертанием (см. рис. 9.1, *3*), цветом фона, подсказкой в виде рисунка или преобразованием ссылки в имитированную кнопку.

- **Добавление подчеркивания только при наведении на ссылку указателя мыши.** Некоторые веб-дизайнеры убирают подчеркивание для всех ссылок, выделяют их каким-то другим способом, а затем включают атрибут подчеркивания только при наведении указателя мыши, как показано на рис. 9.1, *4*. Чтобы обеспечить такой эффект, просто удалите подчеркивание ссылок, а затем повторно введите его в псевдокласс `:hover`:

  ```
  a {
    text-decoration: none;
    background-color: #F00;
  }
  a:hover {
    background-color: transparent;
    text-decoration:underline;
  }
  ```



Рис. 9.1. Существует множество способов сделать стандартное подчеркивание *(1)* более привлекательным: для начала полностью убираем линию подчеркивания *(2, 3, 4)* или создаем более стильную черту путем применения свойства border *(5)* или фонового изображения *(6)*

- **Использование свойства нижней границы.** У вас есть возможность управлять цветом, толщиной или стилем стандартной линии подчеркивания ссылок, которое всегда отображается в виде линии толщиной 1 пиксел того же цвета, что и текст.
  - Большего разнообразия можно добиться путем использования вместо подчеркивания свойства `border-bottom`, как показано на рис. 9.1, *5*.
  - Скрыть обычное подчеркивание и добавить черту в виде пунктирной линии-границы можно следующим образом:

```
a {
  text-decoration: none;
  border-bottom: dashed 2px #9F3;
}
```

Вы можете изменить стиль, толщину линии и цвет границы. Чтобы увеличить пустой промежуток между текстом и границей, применяйте свойство `padding`.

- **Использование фонового изображения.** Можете больше преобразить вид ссылок, добавив фоновый рисунок. Например, на рис. 9.1, *б*, показана граница в виде рукописной линии.

  Подобная методика подчеркивания заголовков описана в обучающем уроке 2 гл. 8. Напомню суть метода. Сначала создается рисунок линии подчеркивания с помощью программы редактирования изображений типа Fireworks или Photoshop, в которой имеется инструмент Brush (Кисть), имитирующий рисование мелком, фломастером и т. д. Затем создается стиль для ссылки, убирающий стандартное подчеркивание и добавляющий фоновое изображение. Оно должно быть расположено по нижнему краю ссылки и повторяться в горизонтальном направлении. Можно также добавить небольшой нижний отступ для рисунка линии:

```
a {
  text-decoration: none;
  background: url(images/underline.gif) repeat-x left bottom;
  padding-bottom: 5px;
}
```

Эту методику хорошо использовать для коротких ссылок (состоящих из 1–3 слов), поскольку, если текст занимает более одной строки, Internet Explorer 6 и 7 добавляют рисунок только к нижнему краю последней строки (IE 8 все делает правильно).

Создание кнопок

Вы можете придать ссылкам вид кнопок, присутствующих в окнах и на панелях инструментов компьютерных программ. В этом вам помогут свойства `border`, `background-color` и `padding`. С их помощью очень просто создать широкий диапазон разновидностей прямоугольных кнопок (рис. 9.2).

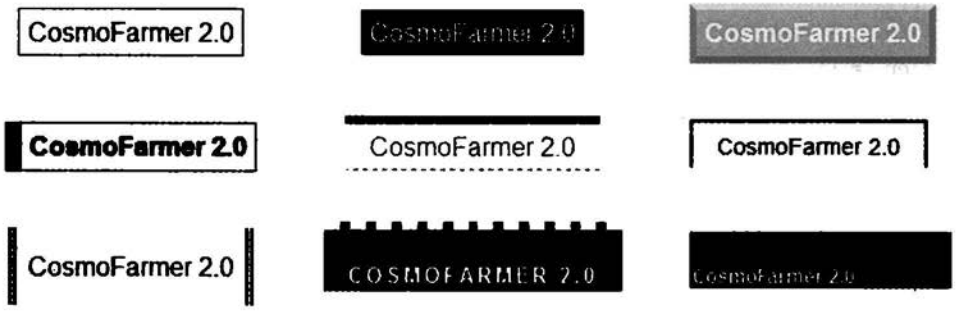


Рис. 9.2. Можно создать кнопки со сложными и привлекательными очертаниями, комбинируя линии-границы различных стилей, цветов, ширины со свойством цвета фона

Допустим, вы назначили стилевой класс ссылке, которую хотите стилизовать в виде кнопки: `<a link="stale.html" class="button">Free Donuts Here!</a>`. Чтобы

добавить вокруг этой ссылки простейшую рамку-контур черного цвета (см. рис. 9.2, *вверху слева*), добавьте следующий стиль:

```
a.button {
  border: solid 1px #000;
}
```

Можете также окрасить кнопку путем установки цвета фона:

```
a.button {
  border: solid 1px #000;
  background-color: #333;
}
```

ПРИМЕЧАНИЕ

В этих примерах и a.button, и .button подойдут в качестве названия стиля. В случае a.button стиль применяется только к тегам <a> с классом button, в то время как .button относится к любому тегу с таким именем класса. Если вы хотите быть уверенными, что стиль применяется только к конкретному тегу, то добавьте имя тега в начале. Добавление названия тега также полезно в качестве напоминания при просмотре кода CSS — оно дает понять, для форматирования чего предназначен стиль. Когда вы видите a.button, становится ясно, что стиль нацелен на определенные ссылки.

Имейте в виду: совсем не обязательно, чтобы линии границ со всех четырех сторон ссылки-кнопки были одинакового стиля. Возможно, что у ссылки даже не будет каких-то границ. Широко применяемая дизайнерская методика превращения плоской кнопки в объемную предполагает использование четырех границ разного цвета, как показано в верхнем правом углу на рис. 9.2. Придание кнопке объемности не представляет сложности, но вы должны помнить, что освещение заставляет предмет выглядеть таким. Представьте, что свет падает на одну из четырех сторон; эта обращенная к источнику света сторона — самая светлая, а противоположная — самая темная (поскольку приподнятая кнопка в выключенном состоянии препятствует прохождению света и вызывает появление тени). Остальные две стороны должны иметь цвета промежуточных оттенков между «светлыми» и «темными». Рассмотрим CSS-код для придания кнопке-ссылке, отображенной в верхнем правом углу рис. 9.2, объемного вида:

```
a.button {
  background: #B1B1B1;
  color: #FFF;
  font-weight: bold;
  border-width: 4px;
  border-style: solid;
  border-top-color: #DFDFDF;
  border-right-color: #666;
  border-bottom-color: #333;
  border-left-color: #858585;
}
```

Вы также можете (и я рекомендую) создать стиль в состоянии :hover. Кнопки будут реагировать на наведение посетителем на ссылку указателя мыши, обеспечивая интерактивность и обратную связь. В случае с объемной кнопкой при ее нажатии очень эффективен метод инверсной смены цветов: темный фон кнопки должен стать светлым, светлая граница — темной и т. д.

Использование изображений

Добавление для ссылок изображений — один из самых легких и эффективных способов улучшить внешний вид элементов навигации сайта. Существует множество методик и вариантов дизайна, но при этом следует заметить, что ни в одном из хороших, грамотных методов не применяется HTML-тег <img>. Вместо этого используется прием с CSS-свойством background-image, с помощью которого можно добавить привлекательность любой ссылке. Несколько примеров приведено на рис. 9.3 (более продвинутые методы использования изображений для создания графических кнопок и эффектов ролловеров[1] описаны в обучающем уроке этой главы).



Рис. 9.3. Даже простой рисунок может оживить ссылку и сделать ее более понятной для восприятия: значок земного шара *(5)* — один из способов обозначения внешних ссылок, а флажок *(6)* покажет посетителю, что он уже посещал эту ссылку

Чтобы вспомнить свойство background-image и все, что с ним связано, вернитесь к разд. «Фоновые изображения» гл. 8. Пока же напомню вам несколько вещей, которые нужно иметь в виду при использовании изображений со ссылками.

- **Не забывайте про значение no-repeat.** По умолчанию фоновые изображения выводятся на заднем плане стилизуемого элемента. Со многими рисунками, применяемыми для ссылок, эффект получается ужасный (см. рис. 9.3, *2*). Если вы не используете едва заметный узор, похожий на градиентную заливку, не забудьте выключить повторное отображение фонового рисунка: background-repeat: no-repeat;.

[1] Ролловер — это элемент веб-страницы, изменяющий свой вид в зависимости от внешнего воздействия.

- **Управляйте позиционированием с помощью свойства background-position.** Чтобы точно разместить фоновое изображение, используйте свойство `background-position` (см. разд. «Позиционирование фоновых изображений» гл. 8). Если необходимо позиционировать изображение по правому краю ссылки, но при этом центрировать его вертикально на строке, используйте следующий CSS-код: `background-position: right center;`.

  Для еще более точного размещения изображений используйте значения параметров в пикселах или em. Эти единицы измерения облегчают смещение рисунка на нужное расстояние от левого края ссылки. Комбинируя эти единицы измерения с процентными значениями, вы сможете не только легко центрировать изображение вертикально по отношению к тексту ссылки, но и обеспечить его небольшое смещение на точное расстояние от левого края: `background-position: 10px 50%;`.

СОВЕТ

При позиционировании фоновых изображений первый параметр свойства указывает горизонтальное (по направлению слева направо) смещение или выравнивание; а второй — вертикальное (по направлению сверху вниз) смещение или выравнивание.

К сожалению, не существует способа точно позиционировать изображение по отношению к правому или нижнему краю ссылки. Поэтому, если вы хотите поместить изображение на небольшом расстоянии от правого края, есть два варианта. Первый заключается в том, чтобы в программе редактирования изображений добавить к правой стороне рисунка пустое пространство. Размер этого отступа должен быть равен тому расстоянию, на которое вы хотите сместить изображение от правого края ссылки. После корректировки для его выравнивания по правому краю стилизуемого элемента используйте свойство `background-position`; например `background-position: right top;`. Кроме того, можете использовать процентные значения: `background-position: 90 % 75 %;` — такие параметры обеспечат позиционирование точки изображения, находящейся на расстоянии 90 % от его левого края, на расстояние 90 % от левого края стилизуемого элемента. Однако данный метод не обеспечивает точности позиционирования, так что в этом случае придется немного поэкспериментировать (о том, как работает позиционирование с процентными значениями, читайте в подразделе «Процентные значения» разд. «Позиционирование фоновых изображений» гл. 8).

- **Добавляйте отступы с помощью свойства padding.** Если вы используете для выделения ссылки на изображение или значок (см. рис. 9.3, *3*, *5* и *6*), нужно добавить отступ с той стороны, где рисунок примыкает к тексту. Например, во фрагменте *3* на рис. 9.3 ссылка имеет левый отступ шириной 30 пикселов для предотвращения наложения текста (слово «Home») на изображение дома, в то же время во фрагментах *5* и *6* небольшой правый отступ обеспечивает пространство для отделения текста ссылки от значков земного шара и флажка.

ПРИМЕЧАНИЕ

Поскольку тег `<a>` является встроенным элементом, добавление верхнего или нижнего отступов (или полей) не окажет никакого эффекта. Причина такого поведения описана в подразделе «Отображение встроенных и блочных элементов» разд. «Управление размерами полей и отступов» гл. 7. Однако можно преобразовать ссылку в блочный элемент, чтобы применить к нему верхние и нижние отступы и поля. Я опишу эту методику позже.

- **Пользуйтесь псевдоклассами.** Не забывайте о псевдоклассах `:hover` и `:visited`. Они помогут создать для ссылок замечательные динамические эффекты и обеспечат полезную обратную связь с посетителями сайта.

  Можно загружать и менять различные фоновые рисунки для любого из этих псевдоклассов, чтобы, например, изображение непосещенной ссылки в виде выключенной лампочки при наведении на нее указателя мыши сменялось на изображение включенной. Или можете не задавать для непосещенных ссылок никакого фонового рисунка, но после перехода по ним добавлять изображение флажка, чтобы визуально показать их статус, то есть что их уже посещали (рис. 9.3, *б*).

  Если вы решите использовать изображение для состояния `:hover` ссылки, помните, что браузеры не загрузят это изображение, пока посетитель сайта на самом деле не наведет на нее указатель мыши. Может возникнуть значительная задержка, прежде чем появится изображение. Однако уже после первой загрузки задержка исчезнет. О том, как решить данную проблему, читайте в следующем разделе.

Создание панелей навигации

Каждому сайту требуется хорошая управляющая панель с элементами навигации: во-первых, чтобы привести посетителей к нужной им информации, а во-вторых, чтобы им понравилось управление сайтом и они вернулись сюда вновь. Содержимое большинства сайтов организовано в виде разделов. Например, «Продукция», «Контактная информация», «Журнал отзывов» и т. д. Такая организация позволяет посетителям хорошо ориентироваться в содержимом сайта, и они точно знают, что и где смогут найти. В большинстве случаев ссылки на основные разделы сайта можно найти на панели навигации. CSS облегчает создание красивых и функциональных панелей, эффектов ролловеров и т. д.

Использование маркированных списков

По сути, панель навигации — не что иное, как набор ссылок, а точнее, она представляет собой список разделов сайта. Как разъясняется в гл. 1, задача языка HTML состоит в том, чтобы обеспечить структуру. Следовательно, вы всегда должны использовать теги в соответствии с их функциональным назначением, чтобы они соответствовали заключенному в них содержимому. К примеру, для списка элементов это должен быть тег `<ul>`, или тег неупорядоченного маркированного списка. Не имеет значения, будет список вообще без маркеров или будет располагаться горизонтально вдоль верхнего края веб-страницы: все форматирование тега `<ul>` обеспечено средствами языка CSS. Пример показан на рис. 9.4.

Применение HTML для создания панели навигации — это и есть использование языка разметки по прямому назначению. В каждом элементе списка имеется по одной ссылке. Кроме того, нам всего лишь необходимо стилизовать этот маркированный список (не нужно, чтобы его пункты составляли панель навигации).



Рис. 9.4. Используя CSS, вы можете превратить обычные теги <ul> в то, что захочется, например в вертикальное или горизонтальное управляющее меню (панель навигации)

Правильным подходом будет применение стилевого класса или стиля с идентификатором (ID-стиля) к тегу <ul>:

```
<ul class="nav">
  <li><a href="index.html">Home</a></li>
  <li><a href="news.html">News</a></li>
  <li><a href="reviews.html">Reviews</a></li>
</ul>
```

CSS-код немного отличается в зависимости от того, нужна нам горизонтальная или вертикальная панель управления. В любом случае вы должны выполнить два шага.

1. **Удалить маркеры.** Чтобы панель навигации не была похожа на маркированный список, удалим маркеры, установив для свойства `list-style-type` значение none:

   ```
   ul.nav {
     list-style-type: none;
   }
   ```

2. **Убрать отступы и поля.** Поскольку браузеры сами делают отступы для элементов списка, нам придется избавиться от них. Однако одни браузеры используют для этих целей свойство `padding`, а другие — `margin`, поэтому нам нужно установить для обоих свойств значение `0`:

   ```
   ul.nav {
       list-style-type: none;
     padding-left: 0;
     margin-left: 0;
   }
   ```

По сути, эти два шага обеспечивают всем элементам списка простой вид, как у обычного блочного фрагмента текста — абзаца или заголовка (с одним исключением: браузер не добавляет полей между элементами списка). С этого момента можно начинать стилизацию. Если вам нужна вертикальная панель навигации, продолжайте читать дальше; если же хотите создать горизонтальную панель управления, то переходите к следующему разделу.

Вертикальные панели навигации

Вертикальная панель навигации — это всего лишь набор ссылок, расположенных одна за другой в виде списка или стека. Удалив маркеры, а также поля и отступы слева (как описано в предыдущем разделе), вы выполните большую часть работы, но вы должны знать, что существует еще несколько дополнительных хитростей, обеспечивающих правильное отображение панели навигации.

1. **Отобразим ссылку в виде блочного элемента.** Поскольку тег <a> является встроенным, ссылка будет занимать по ширине пространство, равное ширине содержимого тега. Кнопки со ссылками в виде текста различной ширины (например, Home и Our Products) будут иметь различную ширину. Ступенчатое вертикальное отображение кнопок друг над другом выглядит не очень хорошо (рис. 9.5, *1*). Кроме того, верхние и нижние отступы и поля не оказывают никакого эффекта на встроенные элементы. Чтобы обойти эти ограничения, преобразуем ссылки в блочные элементы:

```
ul.nav a {
  display: block;
}
```

   Параметр отображения block не только выравнивает все кнопки по ширине, но и делает всю область ссылок активной при щелчке на ней кнопкой мыши подобно настоящим кнопкам. Это тот случай, когда посетители щелкают на области кнопки, где нет текста (например, отступы вокруг), но ссылка по-прежнему работает (в браузере Internet Explorer 6 и его более ранних версиях существует такая проблема; об обходном приеме читайте в обучающем уроке этой главы).

2. **Ограничим ширину кнопок.** Преобразование пунктов списка в блочные элементы обеспечивает ссылкам ширину тегов, в которые они вложены. Поскольку ссылки просто помещены на веб-страницу, их ширина окажется равной ширине окна браузера (см. рис. 9.5, *2*). Есть несколько способов ограничить ширину ссылок.

   Сначала можно просто определить тег <a>. Например, если вы хотите, чтобы все кнопки имели ширину 8 em, добавьте в стиль свойство width:

```
ul.nav a {
  display: block;
  width: 8em;
}
```

   Установка ширины для любого тега, в который заключены ссылки, например <li> или <ul>, приведет к тому же результату.

   Если текст кнопки занимает всего одну строку, то можно его центрировать вертикально, чтобы над и под ним были одинаковые промежутки. Просто установите высоту ссылки и назначьте такое же значение свойства межстрочного интервала: `a { height: 1.25em; line-height: 1.25em; }`.

ПРИМЕЧАНИЕ

Не обязательно явно назначать свойство width, если панель навигации вложена в элемент разметки веб-страницы, для которого уже установлена ширина. Из части 3 вы узнаете, каким образом элементарно создать боковое меню, примыкающее к левому или правому краю веб-страницы. Боковое меню имеет свою ширину, и помещенный в него маркированный список со ссылками-кнопками автоматически приобретает эту ширину.

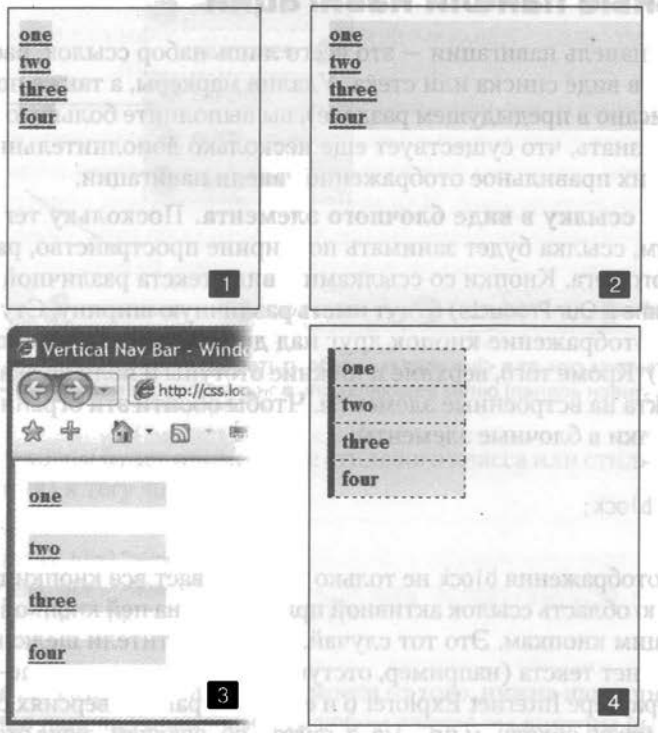


Рис. 9.5. Всего четырьмя простыми действиями можно превратить маркированный список ссылок в привлекательную панель навигации

К сожалению, если вы сами явно не установили ширину для тега <a>, в Internet Explorer 6 и ниже возникнут некоторые проблемы с этими ссылками. Во-первых, Internet Explorer 6 добавляет большие промежутки между такими ссылками (см. рис. 9.5, *3*).

Другая ошибка IE 6 проявляется, когда вы хотите отображать ссылку как блок. Хотя другие браузеры делают всю площадь блока доступной для щелчка кнопкой мыши, IE 6 по-прежнему ограничивает эту возможность текстом внутри ссылки. Другими словами, если вы добавите эффект наведения, например, чтобы подсветить фоновый цвет кнопки, в IE 6 фон будет подсвечиваться только при наведении указателя мыши на текст, но не на любое свободное место в пределах кнопки.

К счастью, эти две проблемы легко исправить. На самом деле, если вы устанавливаете точное значение ширины для тегов <a> навигационной панели, то вы уже заботитесь об ошибках в IE и можете пропустить следующий шаг.

3. **Устраним ошибку в Internet Explorer.** Чтобы удалить эти промежутки и сделать всю область доступной для щелчка, добавьте к ссылке специфичное для IE свойство zoom:

```
ul.nav a { zoom: 1; }
```

Больше об этой маленькой хитрости вы можете узнать из гл. 12, но в основном многие ошибки IE 6 (и некоторые IE 7) решаются просто добавлением свойства `zoom: 1` к элементу. Здесь нет никакой связи с CSS — это всего лишь способ, изменяющий воспроизведение элементов страницы в IE.

Добавление свойства `zoom` никак не повлияет на другие браузеры — они просто игнорируют любой код CSS, который не понимают. Вы могли бы при желании скрыть этот стиль от других браузеров, используя прием `* html`, например `* html ul.nav a { zoom: 1 ; }`. Кроме того, вы могли бы поместить этот код CSS в таблицы стилей для IE с помощью условных комментариев, характерных для этого браузера.

Теперь, когда мы закончили всю эту бесполезную работу, займемся основной задачей — стилизацией кнопок. Можно добавить отступы, поля, изображения, изменить фоновый цвет. Если хотите расположить кнопки так, чтобы они не соприкасались между собой, можете добавить нижнее (или верхнее) поле для ссылок.

ОБХОДНОЙ ПРИЕМ

Если сливаются границы

Если кнопки панели навигации расположены вплотную друг к другу и при этом вы применяете границы, окаймляющие каждую ссылку отдельно, то линии границ соседних ссылок сливаются между собой и превращаются в двойные. Чтобы избавиться от этого, можно добавить границу только к верхней стороне ссылок. Таким образом, мы получим всего одну границу, отделяющую кнопки, расположенные рядом.

Однако в результате этого обходного приема последняя, нижняя ссылка панели управления получается без границы. Чтобы решить проблему, можно либо создать стилевой класс с нижней границей и применить его к последней ссылке, либо, что еще лучше, добавить нижнюю границу к тегу `<ul>`, завершающему панель навигации (этот прием будет продемонстрирован на практике в обучающем уроке настоящей главы).

Горизонтальные панели навигации

CSS позволяет преобразовать список ссылок, расположенных вертикально одна за другой, в список с горизонтальным представлением, как было показано на рис. 9.4. В этом разделе описываются два распространенных подхода в создании из списка горизонтальной панели навигации. Первый метод основывается на использовании свойства `display: inline` — он прост, но с его помощью нельзя создать кнопки одинаковых размеров. Если вы хотите единообразия, обратитесь к методу с применением плавающих элементов `<ul>`.

Какой бы метод вы ни использовали, сначала удалите маркеры и левые отступы тегов `<ul>`, как показано на рис. 9.6, *1*.

Самый простой метод создания горизонтальной панели навигации заключается в изменении для элементов списка значения `block` (блочный элемент) свойства `display` на `inline` (встроенный элемент). Это делается средствами CSS.

1. **Преобразуем пункты списка во встроенные элементы.** Встроенные элементы не создают переносов строк ни перед элементом, ни после него, как это делают

блочные. Установка значения `inline` свойства `display` для тегов `<li>` приведет к отображению элементов списка в одну строку (см. рис. 9.6, *2*).

```
ul.nav li { display: inline; }
```

Вы должны быть уверены в том, что панель навигации кнопок-ссылок не слишком большая. Те кнопки, которые не поместятся в одну строку, перенесутся на другую.

2. **Стилизуем ссылки.** Можно убрать подчеркивание под ссылками и вместо этого добавить вокруг них окаймляющую границу. Чтобы обеспечить визуальную глубину и реалистичность настоящих кнопок, надо изменить цвет фона или добавить рисунок. При необходимости добавьте отступы вокруг текста ссылок. Если требуется разнести кнопки друг от друга, назначьте для них правое поле. Следующий стиль обеспечивает ссылкам визуальное подобие кнопок, как показано на рис. 9.6, *3* и *4*:

```
ul.nav a {
  border: 1px dashed #000;
  border-bottom: none;
  padding: 5px 15px 5px 15px;
  margin-right: 5px;
  background-color: #EAEAEA;
  text-decoration: none;
  color: #333;
}
```

ПРИМЕЧАНИЕ

Если для создания горизонтальной панели управления вы используете метод с встроенными элементами, то не изменяйте значение свойства display ссылок на block. Если вы это сделаете, то ссылки отобразятся в виде вертикального списка, причем будут занимать полную ширину веб-страницы (или ширину блочного элемента, в который заключен список).

3. **Добавим для тега `<ul>` верхний и нижний отступы.** Поскольку теги `<a>` — встроенные элементы, добавление для них верхнего и нижнего отступов фактически не увеличит высоты ссылок. Вместо этого границы и фон ссылок наложатся на соседние вышестоящие и нижестоящие элементы, как было показано в примере на рис. 7.6. Кроме того, в браузере Internet Explorer эти отступы могут привести к исчезновению верхних границ ссылок. В этом случае отступ тега `<a>` также немного приподнимет нижнюю границу тега `<ul>`, и она отобразится в виде фона под ссылками (обведено кружком на рис. 9.6, *3*).

   Чтобы решить проблему, добавим отступ для тега `<ul>`, который создает соответствующее пустое пространство, предотвращающее наложение границ и фона ссылок. Используйте для нижнего отступа `<ul>` такое же значение, как размер нижнего отступа ссылки.

   Для определения верхнего отступа тега `<ul>` добавьте 1 пиксел к значению верхнего отступа ссылки (при использовании em просто удостоверьтесь в том, что отступ тега `<ul>` больше).

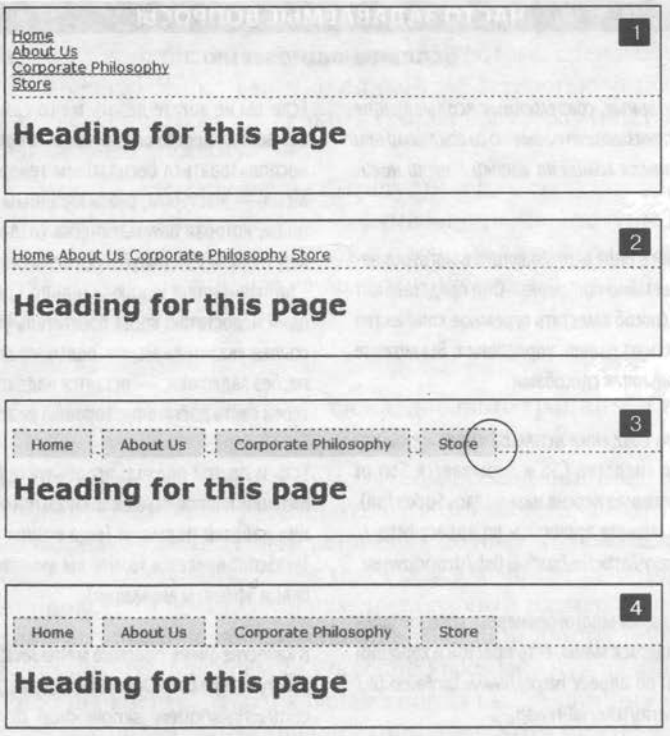


Рис. 9.6. Создание горизонтального меню из списка ссылок потребует выполнения всего нескольких действий

Например, стиль тега <ul>, соответствующий ссылкам в шаге 2, должен выглядеть следующим образом:

```
ul.nav {
    margin-left: 0;
    list-style: none;
    padding-left: 0;
  padding-top: 6px;
  padding-bottom: 5px;
    border-bottom: 1px dashed #000;
}
```

Как видно на рис. 9.6, *4*, нижний отступ позволяет хорошо вписаться нижней границе тега <ul>. Единственная проблема состоит в том, что между кнопками панели навигации всегда имеются пустые промежутки, поэтому, если вы хотите, чтобы они располагались вплотную, нужно использовать плавающие ссылки или установить для них правое поле с отрицательным значением. Другое решение описывается дальше.

СОВЕТ

Чтобы горизонтальная панель навигации отображалась горизонтально по центру веб-страницы, добавьте в стиль тега <ul> свойство text-align: center;.

ЧАСТО ЗАДАВАЕМЫЕ ВОПРОСЫ

Всплывающие меню

Как создать стильные, современные всплывающие меню, которые показывают подменю со ссылками при наведении указателя мыши на кнопки панели навигации?

Панели навигации в виде всплывающих многоуровневых меню чрезвычайно популярны. Они представляют собой отличный способ вместить огромное количество ссылок в компактную панель управления. Вы можете создать их несколькими способами.

Один из методов создания всплывающих меню использует только средства CSS и называется Son of Suckerfish (более ранняя версия называлась Suckerfish). Об этих версиях можете прочитать по адресу http://www.htmldog.com/articles/suckerfish/dropdowns/.

Что касается создания многоуровневого горизонтального раскрывающегося меню, есть простой и хороший обучающий урок по адресу http://www.tanfa.co.uk/css/examples/menu/tutorial-h.asp.

На этом же сайте имеется практический обучающий урок по созданию вертикального меню со всплывающими подменю: http://www.tanfa.co.uk/css/examples/menu/tutorial-v.asp.

Если вы не хотите делать меню самостоятельно, но, возможно, просто ограничены во времени, то можете воспользоваться бесплатным генератором Pure CSS Menu — мастером, реализованным в виде веб-страницы, которая автоматически создает необходимый код HTML и CSS (http://purecssmenu.com). Подход с применением исключительно средств CSS имеет один недостаток: когда посетитель убирает с кнопки-ссылки указатель мыши, подменю исчезает мгновенно, без задержки, — остается надеяться, что у посетителей сайта достаточно хорошая реакция.

Есть и другой подход: используйте CSS-код для стилизации кнопок-ссылок и JavaScript-код для управления работой подменю (еще одним преимуществом JavaScript является то, что вы можете добавлять красивые эффекты анимации).

В качестве очень простого меню JavaScript попробуйте jQuery Simple Drop Down Menu (http://javascript-array.com/scripts/jquery_simple_drop_down_menu). Здесь лишь основы, но меню работает очень хорошо. Более мощным меню с использованием JavaScript является система Superfish. Вы можете прочитать о ней и скачать необходимые файлы на странице http://users.tpg.com.au/j_birch/plugins/superfish.

Используем плавающие элементы для горизонтальной навигационной панели. Хотя методика с использованием `display: inline` для создания горизонтальной панели навигации проста, у нее есть один существенный недостаток: нет способа выровнять ширину кнопок. Установка ширины тегов `<li>` или `<a>` не приведет к требуемому результату, поскольку это встроенные элементы. Чтобы решить проблему, нужно прибегнуть к более сложному и хитрому решению — применению плавающих элементов.

ПРИМЕЧАНИЕ

Панели навигации на основе плавающих элементов трудно центрировать горизонтально посередине веб-страницы. Если требуется такое выравнивание, то лучше пользоваться методом со встроенными элементами, описанным в этой главе.

1. **Преобразуем пункты списка в плавающие элементы.** Добавление для тегов `<li>` свойства `float` с выравниванием по левому краю исключает их из обычного нисходящего потока элементов:

```
ul.nav li { float: left; }
```

Плавающие элементы списка (вместе с вложенными ссылками) располагаются на одной строке, точно так же, как изображения фотогалереи в обучающем уроке в гл. 8 (при необходимости можно с такой же легкостью установить выравнивание кнопок по правому краю экрана (окна браузера) или бокового меню, в которое заключены кнопки).

2. **Добавим в стиль ссылок свойство display: block.** Ссылки — это встроенные элементы, и параметры ширины (как верхние и нижние отступы и поля) к ним неприменимы. Преобразование ссылок в блочные элементы позволит установить точную ширину кнопок и добавить требуемые отделяющие промежутки для ссылок:

```
ul.nav a { display: block; }
```

3. **Стилизуем ссылки.** Измените цвет фона, добавьте границы и т. д. Эта часть работы аналогична той, которую мы выполняли в шаге 2.
4. **Определим ширину.** Если необходимо, чтобы все кнопки панели управления имели одинаковую ширину, установите параметры тега <a>. В свойстве width лучше использовать единицу измерения em, потому что она масштабируема. Значит, исключена ситуация, когда текст ссылок может увеличиться больше размеров кнопок, если посетитель изменит базовый размер шрифта браузера. Точное значение ширины зависит от максимальной длины текста каждой из кнопок. Очевидно, что для ссылки Corporate Philosophy нужна кнопка большей ширины. Если вы хотите, чтобы каждая кнопка имела ширину находящегося внутри нее текста, не задавайте значение ширины. Хотя можно добавить левые и правые отступы и таким образом задать для текста немного пространства, избавившись от тесноты.

СОВЕТ

Чтобы центрировать текст ссылок на кнопках, добавьте в стиль ссылок свойство text-align: center;.

5. **Добавим свойство overflow: hidden к стилю для тега <ul>.** Если для тега <ul> заданы границы, цвет фона или изображение, вам придется «удержать плавающий элемент», то есть плавающие пункты списка тега <ul> будут высовываться из-под нижней части списка (и находиться за пределами границ и фонового цвета тега <ul>).

```
ul.nav {
  overflow: hidden;
}
```

Internet Explorer 6 просто проигнорирует эту инструкцию.

6. **Добавим свойство zoom: 1 для IE 6.**

```
ul.nav {
  overflow: hidden;
  zoom: 1;
}
```

См. шаг 3 в подразделе «Вертикальные панели навигации» выше, где представлено краткое описание этого приема.

Ниже даны описания стилей, требуемых для создания панели навигации, изображенной на рис. 9.7. Обратите внимание, что кнопки имеют одинаковую ширину и текст в них центрирован.

```
ul.nav {
  margin-left: 0px;
  padding-left: 0px;
  list-style: none;
  border-bottom: 1px dashed #000;
  float:left;
  width: 100 %;
}
ul.nav li {
  float: left;
}
ul.nav a {
  width: 12em;
  display: block;
  border: 1px dashed #000;
  border-bottom: none;
  padding: 5px;
  margin-right: 5px;
  background-color: #EAEAEA;
  text-decoration:none;
  color: #333;
  text-align: center;
}
```

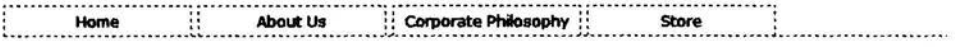


Рис. 9.7. Плавающие элементы списка позволяют создавать кнопки панели навигации одинаковой ширины

ЧАСТО ЗАДАВАЕМЫЕ ВОПРОСЫ

Где узнать, как создается панель навигации

Вы начинающий веб-дизайнер и никогда раньше не создавали панелей навигации, но ваш сайт должен иметь такую. Кажется, что вы самостоятельно с этим не справитесь. Можно ли где-нибудь найти подробное руководство к действию для начинающих?

Да. Практическая обучающая программа представлена в этой главе. В ней подробно, шаг за шагом описывается процесс создания панели навигации. Просто перейдите к обучающему уроку.

Вы можете найти обучающие программы и в Интернете, там размещены также инструменты, которые облегчат работу.

Подробная информация по превращению простых списков в необычные, неординарные элементы управления представлена в пошаговом обучающем уроке в Интернете по адресу http://css.maxdesign.com.au/listutorial/.

Готовые варианты дизайна стильных панелей навигации на основе списков можно найти по адресу http://css.maxdesign.com.au/listamatic/.

Если вы захотите создать панель управления с вкладками (как на страницах сайта Amazon.com), изучите материал следующей веб-страницы: http://css-discuss.incutio.com/?page=ListTabs.

ЧАСТО ЗАДАВАЕМЫЕ ВОПРОСЫ

И наконец, если вы вообще не хотите обременять себя созданием и настройкой собственных панелей-меню, воспользуйтесь мастером List-O-Matic по адресу http://www.accessify.com/tools-and-wizards/developer-tools/list-o-matic/. Этот сайт запрашивает определенную информацию (шрифты, цвета) и самостоятельно создает CSS-код панели навигации на основе списков. Он даже позволяет создавать подменю (известные как выпадающие меню).

Современные методы стилизации ссылок

Теперь, когда вы овладели элементарными правилами стилизации ссылок, познакомились с псевдоклассом :hover, добавлением фоновых изображений, у вас наверняка возникло желание улучшить навигацию сайта путем его более сложного форматирования. В следующих разделах книги мы познакомимся с несколькими наиболее популярными методами.

Большие активизируемые кнопки

Применение псевдокласса :hover представляет собой отличный способ превращения веб-страницы в интерактивную. Но как быть, если вы хотите выделить область, по размеру большую, нежели ссылка из двух слов? Предположим, у вас есть список новостей в боковом меню. Каждый элемент включает заголовок на одной строке, за которым следует абзац с кратким описанием новости. И, допустим, вы хотите каким-то образом выделить область, включающую и заголовок, и краткое описание, при наведении на них указателя мыши (рис. 9.8).



Рис. 9.8. Предоставьте посетителям сайта большую «кнопку» (область активизации): с помощью короткого, но умного CSS-кода вы можете превратить фрагмент веб-страницы, включающий заголовок и абзац текста, в одну большую кнопку-ссылку

К счастью, все браузеры: Internet Explorer 7 и выше, Firefox, Safari, Chrome и Opera — понимают использование псевдокласса :hover применительно к любым элементам веб-страниц, а не только к ссылкам. Если вам нужно подсветить абзац в то время, когда посетитель перемещает над ним указатель мыши, можете сделать это следующим образом:

```
p:hover { background-color: yellow;}
```

Только взгляните: ссылок нет вообще! Вы можете применить эффект наведения указателя мыши hover к каким угодно по размеру фрагментам веб-страницы: заголовкам, фотографиям и абзацам текста. Этого можно добиться путем заключения всего этого содержимого в тег . Таким образом, если в боковом меню каждую новость заключить в теги и применить к ним стилевой класс .newsItem, этот стиль изменит фоновый цвет всех новостей:

```
.newsItem:hover { background-color: #333; }
```

К сожалению, браузер Internet Explorer 6 (и его более ранние версии) не понимает этот стиль вообще. Этот браузер правильно обрабатывает эффект hover только тогда, когда он применен к ссылке. Поскольку тег ссылки — встроенный элемент, нельзя (по крайней мере, согласно правилам HTML) заключить в него другие блочные элементы. Таким образом, вам не удастся объединить заголовок и абзац с кратким описанием в одной ссылке. Вместо этого вы должны создать hover, который при наведении указателя мыши будет применяться одновременно к заголовку и описанию.

В принципе, и это не проблема. Все, что требуется, — включить творческое мышление. Есть простое решение: не помещайте заголовок и текстовое описание в отдельные теги, а держите их в ссылке вместе. Чтобы заголовок был похож на заголовок, используйте CSS. Рассмотрим пример HTML-кода для достижения такого эффекта. Этот отрывок представляет собой один из множества элементов маркированного списка:

```
<li class="story">
  <a href="virgo.html"><span class="title">Virgo: It's Your Month!</span>
  The stars are aligned in your favor. Next month? Not so much.</a>
</li>
```

В данном случае и заголовок, и краткое описание помещены в ссылку, и вы можете выделить оба элемента одним стилем:

```
li.story a:hover {
  background-image: url(highlight.gif);
}
```

В HTML-коде заголовок («Virgo, It's Your Month!») заключен в тег . Вы можете сделать этот текст похожим на блочный заголовок с помощью нескольких простых команд:

```
.story span.title {
  display: block;
  text-weight: bold;
  font-size: 150 %;
}
```

Весь фокус в том, что значение `block` свойства `display` заставляет браузер обрабатывать заключенный в тег `<span>` фрагмент текста как отдельный заголовок с переносом строки до и после фрагмента. Теперь, несмотря на то что заголовок и краткое описание похожи на отдельные блочные элементы, они по-прежнему являются частями одного встроенного тега `<a>`.

ПРИМЕЧАНИЕ

Вы также можете использовать JavaScript для создания лучшей, более гибкой «большой ссылки». Подробную информацию вы найдете в обучающем примере на странице www.creativepro.com/article/view-source-make-your-links-unforgettable.

CSS-стиль для предварительной загрузки ролловеров

Очень давно, когда еще не было языка CSS, для смены одной графической ссылки на другую при наведении на нее указателя мыши приходилось привлекать средства JavaScript. Теперь с помощью CSS вы можете достичь подобного эффекта посредством псевдокласса `:hover` и фонового изображения. Однако этот метод имеет одну проблему: если браузером не загружено сменное изображение для `hover` (ролловер), оно не загрузится, пока на данную ссылку не будет наведен указатель мыши, — для посетителя заметна задержка этого сменного изображения на время загрузки браузером.

Задержка произойдет всего один раз, во время первого наведения указателя мыши посетителем на ссылку, но все-таки ожидание загрузки графики — это издержки XX века.

Метод с применением JavaScript поможет избежать этой проблемы благодаря возможности предварительной загрузки сменных изображений задолго до того, как они понадобятся. В CSS нет таких средств, поэтому придется задействовать другой хитрый трюк, называемый CSS Sprites (изначально он назывался методом Pixy), который использует единственное изображение для создания различных состояний одной навигационной кнопки.

ПРИМЕЧАНИЕ

Оригинальный метод Pixy (предшественник того метода, который мы изучим) описан в Интернете по адресу http://wellstyled.com/css-nopreload-rollovers.html. Эволюционировавший метод CSS Sprites в настоящее время широко используется такими компаниями, как Yahoo и Google, причем не только для эффектов ролловера, но и для оптимизации скорости загрузки сайтов. Вы можете прочитать больше об этом на странице www.mezzoblue.com/archives/2009/01/27/sprite_optim.

Рассмотрим, как реализуется метод.

1. **В программе редактирования изображений создадим один комбинированный рисунок с различными вариантами отображения кнопки.** Нужно создать изображения, представляющие собой различные состояния кнопки: обычное ненажатое состояние, состояние при наведении указателя мыши, а также при необходимости состояние нажатой кнопки. Далее поместим изображения вертикально одно над другим. У нас будет два состояния кнопки: изображение в ненажатом состоянии должно располагаться сверху, а изображение-ролловер при наведении указателя мыши — снизу.

2. **Измерим расстояние от верхнего края получившегося комбинированного изображения до верхней границы каждого последующего изображения.** На рис. 9.9, *вверху*, верхняя граница сменного изображения-ролловера имеет смещение от верхнего края общего изображения в 39 пикселов.
3. **Создадим CSS-стиль для ссылки в обычном ненажатом состоянии.** Поместим изображение в качестве фонового в левый верхний угол стилизуемой ссылки (см. рис. 9.9, *посередине*).

   Стиль может выглядеть следующим образом:

   ```
   a { background: #E7E7E7 url(images/pixy.png) no-repeat left top; }
   ```

4. **Создадим стиль :hover.** Будем использовать свойство `background-position`, чтобы сместить изображение вверх. Таким образом, первое изображение исчезнет, а второе, нижнее, изображение-ролловер окажется видимым (см. рис. 9.9, *внизу*).

   ```
   a:hover { background-position: 0 -39px; }
   ```

   Такая методика, помимо предотвращения длительной задержки в загрузке дополнительного изображения-ролловера, обеспечит хранение всех изображений кнопки, отражающих ее состояния, в единственном файле.

СОВЕТ

CSS предоставляет и другие способы предварительной загрузки изображения. Его можно поместить в качестве фонового, но оно будет прикрыто сверху другим рисунком, основным. Предположим, логотип сайта отображается в верхнем левом углу веб-страницы. Вы можете поместить изображение-ролловер в этот же угол фона веб-страницы: body { background: url(rollover.gif) no-repeat left top; }. В процессе загрузки установится и фоновое изображение-ролловер. Посетители веб-страницы его не увидят, потому что оно находится на заднем плане веб-страницы и поверх него отображается логотип.

Другой метод заключается в том, что вы должны поместить изображение-ролловер в тег , который нужно изначально позиционировать средствами CSS за пределы видимой части веб-страницы (см. гл. 12). В любом случае браузер загрузит изображение-ролловер во время загрузки веб-страницы, и в дальнейшем не будет никаких задержек.

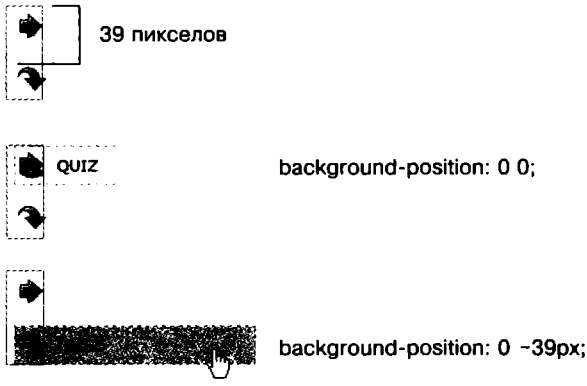


Рис. 9.9. Используя метод CSS Sprites, вы можете избежать задержки во время загрузки браузером изображения-ролловера в первый раз, включив все изображения различных состояний кнопки-ссылки в единственный файл рисунка

ПРИМЕЧАНИЕ

Некоторые сайты довели использование этого метода до крайности. Yahoo!, Amazon и Google (среди многих других) часто собирают вместе в одном файле множество маленьких изображений и показывают лишь часть файла, содержащую необходимую кнопку. Вы можете посмотреть пример от Amazon здесь: www.flickr.com/photos/mezzoblue/3217540317.

Для еще большего контроля сайт одного известного дизайнера использует отдельный графический файл для управления 15 различными кнопками панели навигации. Вы можете прочитать о таком методе на странице http://veerle.duoh.com/index.php/blog/comments/the_xhtml_css_template_phase_of_my_new_blog_part_2. А в действии данный метод можно увидеть в обучающем уроке этой главы.

Вкладки

С тех пор как компания Amazon популяризировала кнопки навигации в виде вкладок, этот вид панелей управления стал одним из самых распространенных способов наглядно отобразить структуру сайта. И тому есть серьезная причина. Способ доступа к информации посредством такого меню можно описать так: чтобы открыть новую «папку» с содержимым, нужно выбрать соответствующую вкладку.

Существует множество способов создания кнопок вкладок. В основном вы можете использовать границы и фоновые цвета вокруг ссылок для придания вида вкладки (см. рис. 9.7). Этот метод использует CSS — никаких изображений.

С помощью графики действительно можно добавить глубину и визуальную привлекательность вашим кнопкам. Один из распространенных методов — создание вкладки из одной графики, когда текст добавляется к кнопкам в графических редакторах, например Photoshop или Fireworks. Тем не менее обновление множества изображений кнопок каждый раз при смене навигации сайта может быстро поднадоесть. Кроме того, наличие различных изображений для каждой кнопки значительно замедляет загрузку веб-страниц.

Модным направлением становится методика, суть которой в следующем: в качестве фона ссылок помещается изображение вкладок, а в виде подписей к кнопкам используется обычный HTML-текст. Таким образом, обновление меню навигации вашего сайта становится очень простым и заключается в изменении текстовых подписей. И даже те, кто не имеет опыта работы с программой редактирования изображений типа Photoshop, могут с легкостью внести нужные изменения. Единственный случай, когда метод с вкладками работает не очень хорошо, — когда длина текста кнопок-ссылок сильно различается. Если на одной вкладке написано **Store**, а на другой — **Contact Us Today!**, то первая из них выглядит неидеально из-за избытка пустого пространства, а вторая — наоборот, слишком сжата (рис. 9.10, *1*).

На рис. 9.10 использование изображения вкладки больших размеров *(3)* — с шириной и высотой, бóльшими, чем максимальный размер вкладки, — гарантирует, что вкладки будут выглядеть надлежащим образом даже в том случае, когда посетители увеличат размер текста (базового шрифта браузера) *(5)*.

Все, что требуется в данном случае, — найти способ уменьшить изображение вкладки под размер текстовой подписи кнопки-ссылки. К счастью, веб-дизайнер Дуглас Бауман (Douglas Bowman) придумал хитрый способ сделать это. Подобно

методу «раздвижных штор», он использует одно большое по ширине и высоте изображение вкладки, созданное в программе редактирования изображений (см. рис. 9.10, *2*), впоследствии разрезанное на две части в виде двух файлов рисунков (см. рис. 9.10, *3*). Одно изображение представляет собой узкую полоску левой стороны вкладки. Это должен быть лишь маленький закругленный левый край вкладки. Второй рисунок — оставшаяся часть изображения в виде широкой вкладки. Она шире, чем максимально возможная, и формирует основную часть и правый край.

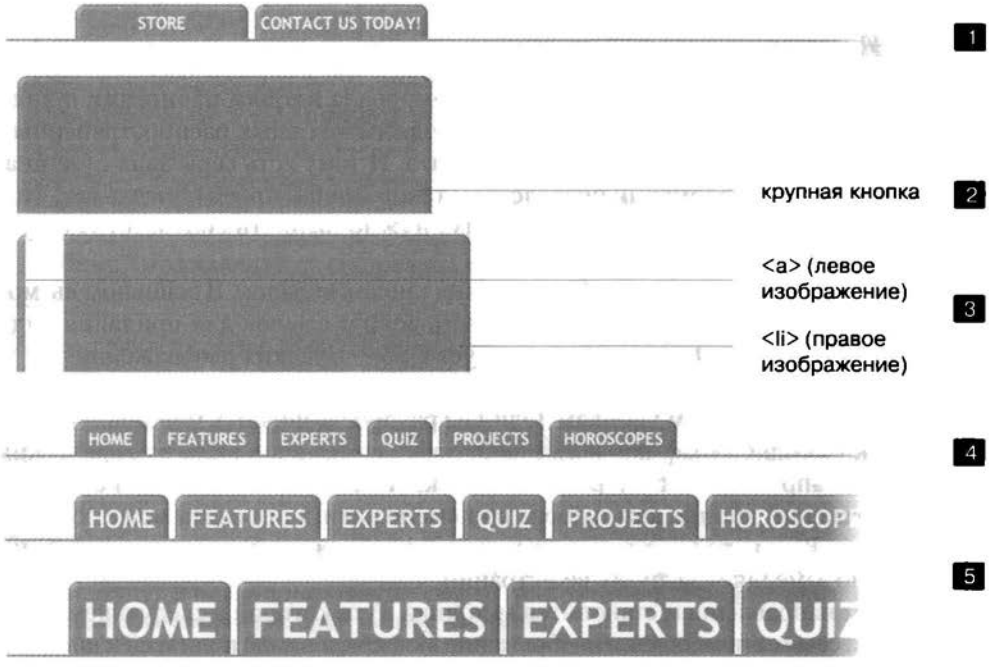


Рис. 9.10. Используя метод «раздвижных штор», можно создать панель навигации с вкладками для ссылок любых размеров

ПРИМЕЧАНИЕ

Методика «раздвижных штор» Дугласа Баумана — классика CSS-дизайна. Статью автора вы сможете найти в Интернете по адресу http://www.alistapart.com/articles/slidingdoors. Имеется также последующая статья, описывающая более продвинутую методику, по адресу http://www.alistapart.com/articles/slidingdoors2.

В чем же особенность? Поскольку тег может иметь только одно фоновое изображение, вы должны назначить рисунки для двух различных тегов. Сначала нужно создать маркированный список и преобразовать его в горизонтальную панель навигации, как описано в подразделе «Горизонтальные панели навигации» предыдущего раздела. Каждый тег `<a>` вложен в свой тег `<li>`, таким образом, у нас имеется два нужных нам тега.

Теперь добавьте широкое фоновое изображение в стиль тега <li>, причем поместите его в верхний правый угол тега. Это делается путем добавления в стиль, форматирующий <li> соответствующей кнопки, следующего свойства:

```
background: url(images/right_tab.gif) no-repeat right top;
```

Методика «раздвижных штор» основана на том факте, что фоновое изображение никогда не выходит за пределы блочного элемента, созданного его тегом. Другими словами, несмотря на то, что этот рисунок имеет очень большую ширину и высоту, вы никогда не увидите часть изображения, которая выступает за пределы тега <li>, будь то нижняя или левая сторона тега.

ПРИМЕЧАНИЕ

Если понравилась эта методика, но вы не умеете работать с программой Photoshop, можете выбрать и бесплатно скачать вариант дизайна вкладок в Интернете по адресам: http://www.exploding-boy.com/2005/12/15/free-css-navigation-designs и http://www.explodingboy.com/2005/12/21/more-free-css-navigation-menu-designs.

Поместите узкую левую часть изображения вкладки в левый верхний угол фона тега <a>, добавив в стиль ссылки следующее свойство:

```
background: url(images/left_tab.gif) no-repeat left top;
```

Поскольку тег <a> вложен в <li>, его фон располагается над <li>. И этот узкий левый край вкладки отображается поверх широкого изображения, создавая иллюзию единого изображения вкладки. Теперь можете впечатывать в свои ссылки любой текст, теги пунктов списка <li> будут растягиваться, вмещая требуемый текст и показывая более широкое изображение вкладки (см. рис. 9.10, *4*).

СОВЕТ

Практический пример веб-страницы с применением метода «раздвижных штор» вы сможете найти в обучающем уроке данной главы книги. Файл находится в папке 09/sliding_doors.

Стилизация отдельных видов ссылок

Веб-дизайнеры используют ссылки на все, что угодно: другие веб-страницы своих сайтов, веб-страницы посторонних сайтов, файлы Adobe Acrobat, документы Word, архивы ZIP и многое другое. Чтобы помочь ориентироваться посетителям сайта, вы можете подсказать им, к чему ведет ссылка, прежде чем они щелкнут на ней кнопкой мыши. Расширенные селекторы представляют отличный способ, чтобы сделать именно это. Хотя такой метод использует селекторы из CSS 3 (следующего, но еще не до конца готового стандарта CSS), почти все браузеры в настоящее время понимают эти селекторы.

ПРИМЕЧАНИЕ

Internet Explorer 6 не понимает эти селекторы. Однако, доля рынка, занимаемого этим браузером, продолжает сокращаться (на момент написания составляет уже менее 17 %), и использовать такой метод можно без ущерба для юзабилити вашего сайта. Большинству посетителей будет доступна улучшенная версия, в то время как обладатели IE 6 будут видеть привычный облик сайта.

Ссылки на другие сайты

Вы можете легко создать стиль, который определяет ссылки на другие сайты, используя селектор атрибута. Как рассказывалось в гл. 3, селекторы атрибутов позволяют стилизовать теги HTML, у которых есть определенный атрибут, например тег `<img>` с атрибутом `alt`, для которого установлено значение `Our Company`. Вы также можете стилизовать теги, атрибуты которых начинаются с определенных значений. Любая ссылка, которая ведет за пределы вашего сайта, должна быть абсолютным URL, то есть должна начинаться с **http://**, например **http://www.yahoo.com**. Таким образом, чтобы создать стиль, который воздействует только на ссылки с абсолютным URL, можно использовать такой селектор:

```
a[href^='http://']
```

Сочетание ^= переводится как «начинается с», так что этот селектор соответствует таким ссылкам, как `<a href="http://www.google.com/">`, `<a href="http://www.sawmac.com/missing/css2e/">` и т. д.

Вы могли бы стилизовать их любым образом, но распространенным является добавление рядом со ссылкой маленького изображения — значка, который указывает на то, что это внешняя ссылка. Вы увидите это в действии в обучающем примере.

Если вы решили использовать абсолютные ссылки на другие страницы вашего сайта, то вам нужно добавить еще один стиль для «выключения» стилизации, в противном случае ссылки будут выделяться как внешние, хоть в действительности они являются простыми ссылками на другие страницы в пределах сайта. Этот второй стиль просто использует более обстоятельный вариант селектора, приведенный выше. Например, если ваш сайт находится по адресу **www.mysite.com**, то вы можете создать селектор, который относится к таким ссылкам, следующим образом: `a[href^='http://www.mysite.com']`. Собирая все это вместе, если вы хотите добавить значок с земным шаром рядом с внешними, но не внутренними ссылками вашего сайта, создайте два этих стиля:

```
a[href^='http://'] {
  background: url(images/globe.png) no-repeat center right;
  padding-right: 15px;
}
a[href^='http://www.mysite.com'] {
  background: none;
  padding-right: 0;
}
```

ПРИМЕЧАНИЕ

Если вы хотите следовать последним новшествам CSS, то можете объединить селектор атрибутов с селектором :not() из CSS 3 для создания отдельного стиля, который будет воздействовать на все абсолютные адреса URL, кроме тех, что указывают на ваш собственный сайт:

a[href^='http://']:not(a[href^='http://www.mysite.com'])

Этот причудливый селектор переводится как «выбрать все ссылки, которые начинаются с http://, но не те, которые начинаются с http://www.mysite.com». Недостатком метода является то, что ни одна из версий Internet Explorer (даже восьмая) не понимает селектор :not(), поэтому значительная часть посетителей не почувствует пользы от этого стиля.

Ссылки на адреса электронной почты

Ссылки на адреса электронной почты — еще один особый вид ссылок. Обычно они выглядят как любые другие ссылки: имеют синий цвет и подчеркивание. Тем не менее они действуют не так, как другие ссылки. Нажатие ссылки на электронный адрес запускает программу для работы с почтой на компьютере посетителей, что некоторых из них действительно отвлекает. Поэтому желательно как-то подсказать пользователю, что это ссылка на адрес почты.

Для этого используется тот же базовый метод, который был описан применительно к внешним ссылкам выше. Поскольку все ссылки на почтовые адреса начинаются с mailto:, вы можете создать селектор наподобие следующего для стиля, форматирующего только данный тип ссылок:

```
a[href^='mailto:']
```

Скоро вы увидите такой пример в действии.

Ссылки на определенные типы файлов

Некоторые ссылки указывают на файлы, а не на другие веб-страницы. Вы часто могли видеть ежегодные отчеты различных компаний в виде загружаемого PDF-файла или ZIP-архива файлов (как, например, обучающие примеры для этой книги) на сайте. Ссылки на такие типы файлов, как правило, вынуждают браузер приступить к загрузке соответствующего файла на компьютер посетителя или, в случае файлов PDF, запустить плагин (подключаемый модуль), который позволит просматривать документ прямо в браузере. Выбрав ссылку, посетитель может оказаться в шоке от того, что на самом деле началась загрузка файла размером 100 Мбайт!

Вы можете идентифицировать определенные типы файлов наподобие внешних ссылок или ссылок на адреса электронной почты. Но вместо того, чтобы пытаться найти какую-то конкретную информацию в начале адреса URL ссылки, поищите ее в конце. Например, ссылка на документ PDF может выглядеть так `<a href="annual_report.pdf">`, а ссылка на ZIP-архив — `<a href="tutorials.zip">`. В каждом случае конкретный тип файла определяется расширением в конце адреса URL — `.pdf` или `.zip`.

CSS 3 предоставляет селектор атрибутов, который позволяет искать атрибуты, заканчивающиеся какой-либо конкретной информацией. Так, чтобы создать стиль ссылки на файлы PDF, используйте такой селектор:

```
a[href$='.pdf']
```

Сочетание $= означает «заканчивается на», и, соответственно, данный селектор указывает выбрать все ссылки, значение атрибута `href` которых заканчивается на `.pdf`. Вы можете создать аналогичные стили и для других типов файлов:

```
a[href$='.zip'] /* zip archive */
a[href$='.doc'] /* Word document */
```

Далее вы увидите примеры этого метода.

Обучающий урок 1: стилизация ссылок

В этом обучающем уроке мы потренируемся в стилизации ссылок разнообразными способами, в том числе путем добавления изображений-ролловеров и фоновых рисунков.

Чтобы начать обучающий урок, вы должны загрузить файлы, содержащие учебный материал. Как это сделать, описывается в конце гл. 2. Файлы текущей обучающей программы находятся в папке **09**.

Простейшее форматирование ссылок

Начнем с простого форматирования ссылок.

Запустите браузер и откройте файл веб-страницы `links.html` в папке **09\links**.

1. Эта страница содержит множество ссылок (обведены на рис. 9.11), которые указывают на другие веб-страницы текущего или иных сайтов, а также обозначают адреса электронной почты. Сначала изменим цвет ссылок данной страницы.

   Откройте файл `links.html` в HTML-редакторе и поместите указатель между открывающим и закрывающим тегами `<style>`.

2. Эта веб-страница уже имеет внешнюю таблицу стилей, придающую ей какое-то базовое форматирование и содержащую теги `<style>` внутренней таблицы.

ПРИМЕЧАНИЕ

Вы разместите стили для этого упражнения во внутренней таблице для того, чтобы было легче писать код и просматривать страницу. Когда все будет готово, желательно переместить стили во внешнюю таблицу стилей.

3. Добавьте во внутреннюю таблицу новый стиль:

```
<style type="text/css">
a {
  color: #207EBF;
}
</style>
```

   Этот стиль будет применяться ко всем тегам `<a>` на странице. С него хорошо начинать, поскольку он устанавливает общий внешний вид для ссылок на странице. Вы добавите больше стилей, которые позволят вам форматировать ссылки в определенных областях страницы. Теперь пришло время удалить это надоевшее подчеркивание под ссылкой.

4. Добавьте в только что созданный стиль `#main a` свойство `text-decoration: none;`.

   Он убирает подчеркивание, но в то же время ссылка на веб-странице становится менее заметной. Не забывайте, что ссылки всегда должны визуально выде-

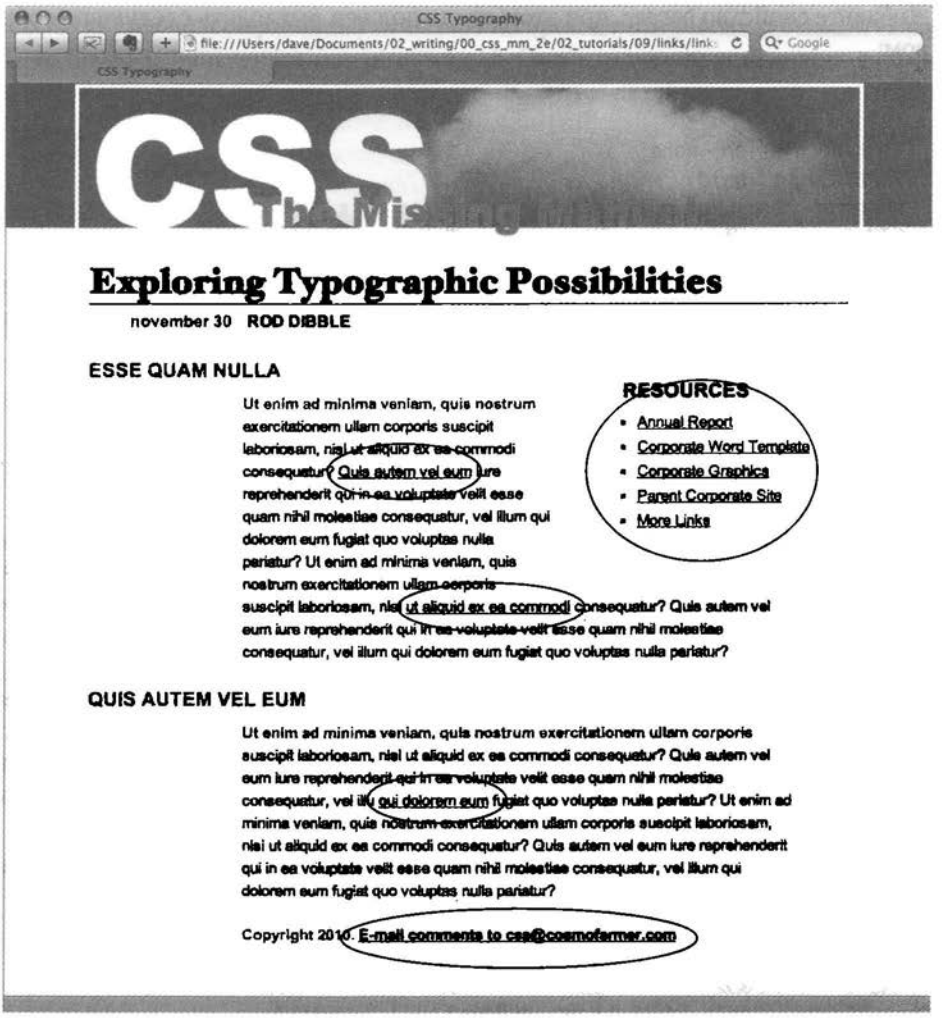


Рис. 9.11. Простейшая веб-страница со стандартной для браузеров стилизацией

ляться на общем фоне, чтобы у посетителей сайта не возникало сомнений в том, что это именно ссылки и на них можно щелкнуть кнопкой мыши.

На рис. 9.11 часть ссылок указывает на одни страницы сайта, часть — на другие и одна ссылка обозначает адрес электронной почты. В обучающем уроке мы отформатируем каждый тип по-разному.

5. Добавьте в стиль `a` свойство `font-weight: bold;`.

   Теперь ссылки отображаются полужирным шрифтом (для остального текста он также может быть установлен). Далее заменим подчеркивание на выделение другим способом, но сделаем это творчески, используя вместо свойства `text-decoration` границы.

6. Добавьте свойство border, при этом стиль должен выглядеть следующим образом:

```
#main a {
  color: #207EBF;
  text-decoration: none;
  font-weight: bold;
  border-bottom: 2px solid #F60;
}
```

Теперь ссылки выделяются, а использование границ вместо обычного подчеркивания позволит изменить стиль, цвет и толщину линий (рис. 9.12). А сейчас вы измените вид посещенных ссылок.

7. Добавьте стиль псевдокласса :visited для посещенных ссылок:

```
a:visited {
  color: #6E97BF;
}
```

Стиль изменяет внешний вид посещенных ссылок на более светлый с серым оттенком — это искусный способ отвлечь внимание от уже посещенной страницы. Если вы просмотрите страницу сейчас, щелкните кнопкой мыши на одной из ссылок (попробуйте, например, одну из тех, что находятся в центре страницы), а затем вернитесь к странице links.html. Вы должны увидеть, что ссылка стала светлее. Но вы также заметите, что остается выделение полужирным и все еще присутствует то оранжевое подчеркивание, которое вы назначили для стиля в шаге 6. Это пример каскадности в действии (см. гл. 5): стиль a:visited является более значимым, чем простой селектор, поэтому его свойство цвета переопределяет тот цвет, который был назначен стилем.

Сейчас добавим эффект-ролловер hover (наведения), чтобы фон изменял цвет при наведении на ссылку указателя мыши.

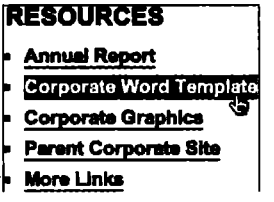


Рис. 9.12. С помощью нескольких стилей вы можете изменить внешний вид любой ссылки, а используя псевдокласс :hover, можно установить отдельный стиль, включаемый при перемещении указателя мыши над ссылкой

8. Добавьте в таблицу стилей стиль псевдокласса :hover:

```
a:hover {
  color: #FFF;
  background-color: #6E97BF;
  border-bottom-color: #6E97BF;
}
```

Этот псевдокласс применяется только на время, пока указатель мыши показывает ссылку или перемещается над ней. Свойство интерактивности эффекта-ролловера позволяет посетителям узнать, что выполняется какое-то действие (см. рис. 9.12).

Добавление для ссылки фонового изображения

Ссылка на адрес электронной почты внизу веб-страницы пока осталась нетронутой созданными стилями (рис. 9.13, *вверху*). Поскольку `mailto` указывает на адрес электронной почты, при щелчке на ней кнопкой мыши посетитель не перейдет на другую веб-страницу, а запустит почтовую программу. Чтобы обеспечить визуальное выделение этой ссылки, добавим небольшой значок почтового конверта.

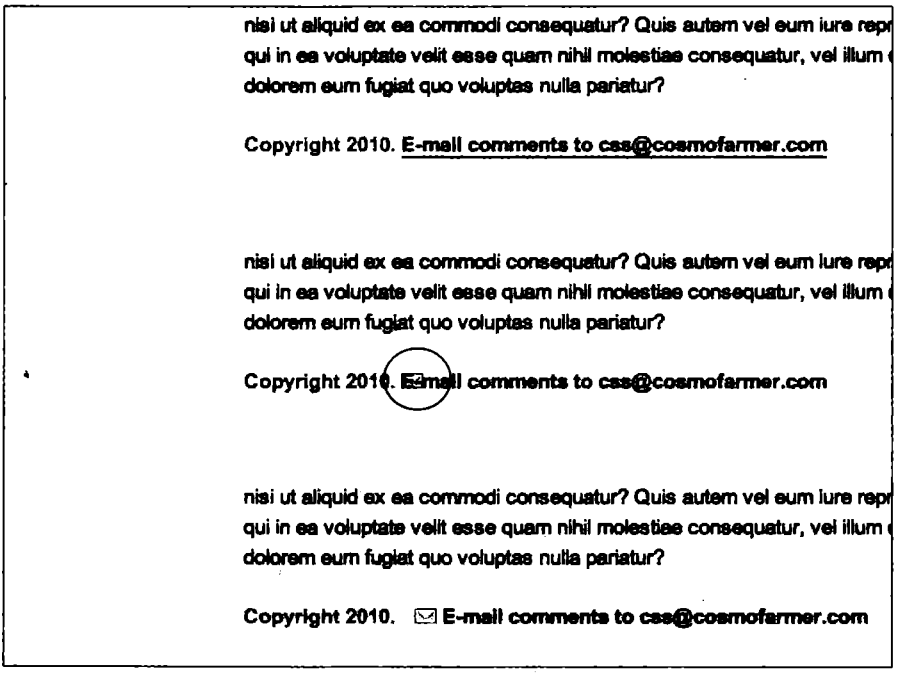


Рис. 9.13. Несколько тонких корректировок сделают назначение ссылки понятным: простая ссылка *(вверху)* превращается в явную, узнаваемую владельцами электронной почты *(внизу)*

1. Добавьте во внутреннюю таблицу стилей файла `links.html` еще один селектор потомков:

```
#legal a {
  color: #666666;
  border: none;
  background: url(images/email.gif) no-repeat left center;
}
```

Ссылка на электронный адрес заключена в тег с идентификатором legal, так что этот стиль влияет только на конкретную ссылку электронной почты. Для ссылки назначен серый цвет, а код border: none убирает подчеркивание, определенное в шаге 6. Свойство background добавляет фоновое изображение с левой стороны, а параметр no-repeat обеспечивает однократное отображение рисунка. Здесь трудность состоит в том, что фоновое изображение (значок конверта) располагается на заднем плане прямо под текстом ссылки, и она становится трудночитаемой (см. рис. 9.13, *посередине*).

2. Добавьте в только что созданный стиль #legal a левый отступ размером 20 пикселов:

```
padding-left: 20px;
```

Помните, что отступ добавляет промежуток между содержимым и границей элемента, поэтому установка небольшого левого отступа смещает текст ссылки на 20 пикселов, оставляя фоновое изображение на месте. И последний штрих: нужно немного отодвинуть всю ссылку от примечания об авторском праве.

3. Добавьте в стиль левое поле размером 10 пикселов. Должен получиться следующий окончательный вариант:

```
#legal a {
  color: #666666;
  border: none;
  background: url(images/email.gif) no-repeat left center;
  padding-left: 20px;
  margin-left: 10px;
}
```

Эта маленькая корректировка обеспечивает визуальное отделение изображения-значка почтовой ссылки от примечания об авторском праве (см. рис. 9.13, *внизу*). Таким образом, ссылка воспринимается посетителем как сочетание значка с текстом.

СОВЕТ

Вы также можете использовать расширенный селектор атрибута для выделения ссылок на адреса электронной почты. Как используются такие селекторы, вы увидите в следующем разделе.

Выделение внешних ссылок

Иногда требуется визуально показать, что ссылка соответствует другому внешнему сайту. Этим можно сказать посетителям, что дополнительная информация по теме находится где-то еще в Интернете, на другом сайте, или предупредить, что выбор этой ссылки приведет к переходу на другой сайт. Кроме того, вы, возможно, захотите идентифицировать ссылки, указывающие на загружаемые файлы или другие документы, не являющиеся веб-страницами.

На веб-странице, над которой вы работаете, правостороннее боковое меню Resources содержит различные типы ссылок, которые вы выделите значками, используя отдельный значок для каждого типа. Для начала вы настроите базовый стиль, который применяется ко всем ссылкам.

1. Добавьте во внутреннюю таблицу стилей веб-страницы `links.html` следующее:

```
#resources a {
  border-bottom: none;
}
```

Поскольку все ссылки, которые нужно отформатировать, находятся внутри `<div>` с идентификатором `resources`, селектор потомка `#resources a` воздействует только на эти ссылки. Этот стиль избавляется от подчеркивания, которое было добавлено в общем стиле ранее.

Далее вы добавите значок рядом с внешними ссылками.

2. Добавьте другой стиль в конец внутренней таблицы стилей веб-страницы `links.html`:

```
a[href^='http://'] {
  background: url(images/globe.png) no-repeat right top;
}
```

Этот стиль использует расширенный селектор атрибута. В основном он воздействует на любую ссылку, которая начинается с http://. Как и стиль ссылки на адрес электронной почты, который вы создали ранее, этот стиль добавляет фоновое изображение. Он помещает изображение на правую сторону ссылки.

ПРИМЕЧАНИЕ

В этом разделе вы используете расширенные селекторы атрибутов для стилизации различных ссылок на боковой панели страницы. Большинство браузеров, за исключением IE 6, понимают эти стили. Если вы хотите создать аналогичные стили, которые работают с IE 6, лучше всего использовать класс стилей, например .externalLink, а потом вручную применить имя класса к ссылкам на внешние сайты: `<a class="externalLink" href="http://www.twitter.com">`.

Этот метод предполагает много работы, поскольку вам нужно добавить классы к каждому типу ссылок — внешней, файлам PDF, документам Word и т. д. Только если ваш клиент или начальник не потребует иного, лучше быть дальновидным и использовать селекторы CSS 3 — сокращающееся сообщество пользователей IE 6 по-прежнему сможет выбрать ссылку, просто пользователи этого браузера не увидят значков.

Тем не менее у этого стиля есть проблема, аналогичная проблеме стиля ссылки на адрес электронной почты, — изображение появляется под текстом ссылки. К счастью, решение такое же — нужно добавить отступы, чтобы переместить изображение в нужное место. В этом случае, однако, вместо левого отступа вы добавите правый (поскольку значок появляется на правой стороне ссылки). Кроме того, так как все ссылки в разделе ресурсов будут иметь значки схожих размеров, вы можете сократить количество кода, добавив отступы к стилю `#resources a`, созданному в шаге 1.

3. Отредактируйте стиль `#resources a` так, чтобы он выглядел следующим образом:

```
#resources a {
  border-bottom: none;
  padding-right: 22px;
}
```

Если вы сохраните страницу и просмотрите ее в браузере, то увидите маленькие значки с земным шаром с правой стороны двух нижних ссылок боковой панели. Теперь отформатируем другие ссылки.

4. Добавьте еще три стиля во внутренней таблице стилей:

```
a[href$='.pdf'] {
  background: url(images/acrobat.png) no-repeat right top;
}
a[href$='.zip'] {
  background: url(images/zip.png) no-repeat right top;
}
a[href$='.doc'] {
  background: url(images/word.png) no-repeat right top;
}
```

Эти три новых стиля проверяют, как заканчивается атрибут `href`, идентифицируют ссылки как файлы Adobe Acrobat (`.pdf`), ZIP-архивы (`.zip`) или документы Word (`.doc`) и назначают различные значки в каждом конкретном случае.

5. Наконец, добавьте состояние `hover` (наведения указателя мыши) для ссылок на ресурсы:

```
#resources a:hover {
  color: #FFF;
  background-color: #6E97BF;
}
```

Этот стиль изменяет цвет текста и добавляет цвет фона (рис. 9.14).

Окончательную версию этого урока вы найдете в файле `09_finished/links/links.html`.

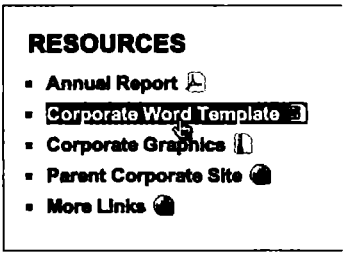


Рис. 9.14. Используя расширенные селекторы атрибутов, вы можете легко идентифицировать и стилизовать различные типы ссылок веб-страницы

Обучающий урок 2: создание панели навигации

На этом практическом занятии мы превратим простой список ссылок во впечатляющую панель навигации с эффектом-ролловером и выделением нажатой кнопки текущего раздела сайта.

1. Откройте файл веб-страницы `nav_bar.html` из папки **09\nav_bar** в HTML-редакторе.

   Как видите, в этом файле содержится совсем короткий программный код. Здесь присутствует внутренняя таблица со сбросом стандартных стилей CSS и одним правилом, определяющим простейшее форматирование тега `<body>` основного содержимого веб-страницы. Код HTML создает маркированный список, состоящий из шести ссылок. Веб-страница имеет вид, показанный на рис. 9.15, *1*. Первым шагом будет добавление HTML-кода, обеспечивающего целенаправленное форматирование ссылок списка средствами CSS.

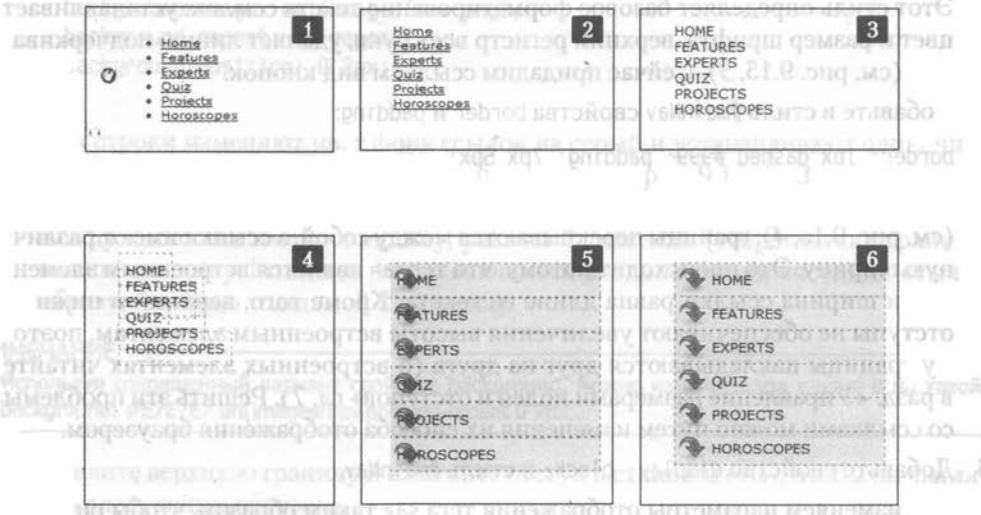


Рис. 9.15. Вам может показаться, что преобразование простого списка в сложную панель навигации требует выполнения объемной работы, но на самом деле нужно всего лишь создать несколько стилей

2. Найдите открывающий тег `<ul>` и добавьте в него идентификатор `id = "mainNav"`, чтобы он выглядел следующим образом:

```
<ul id="mainNav">
```

   Идентификатор `id` назначает этот список главной областью навигации, или панелью управления. Мы будем использовать его для создания производных селекторов, избирательно форматирующих только ссылки этого списка (а не все ссылки веб-страницы).

3. Добавьте во внутреннюю таблицу после `body` новый стиль:

```
#mainNav {
  margin: 0;
  padding: 0;
  list-style: none;
}
```

Этот стиль будет применен лишь к тегу <ul> с идентификатором mainNav. Он удаляет отступ и маркеры, которые браузер добавляет в маркированные списки, как показано на рис. 9.15, *2*. Теперь приступим к форматированию ссылок.

4. Добавьте производный селектор для форматирования ссылок списка:

```
#mainNav a {
  color: #000;
  font-size: 11px;
  text-transform: uppercase;
  text-decoration: none;
}
```

Этот стиль определяет базовое форматирование текста ссылок: устанавливает цвет и размер шрифта, верхний регистр всех букв, удаляет линию подчеркивания (см. рис. 9.15, *3*). Сейчас придадим ссылкам вид кнопок.

5. Добавьте в стиль #mainNav свойства border и padding:

```
border: 1px dashed #999; padding: 7px 5px;
```

Теперь при просмотре веб-страницы в браузере вы заметите пару проблем (см. рис. 9.15, *4*): границы перекрываются между собой, а ссылки имеют различную ширину. Это происходит потому, что тег <a> является встроенным элементом и ширина ссылки равна длине ее текста. Кроме того, верхний и нижний отступы не обеспечивают увеличения высоты встроенным элементам, поэтому границы накладываются друг на друга (о встроенных элементах читайте в разд. «Управление размерами полей и отступов» гл. 7). Решить эти проблемы со ссылками можно путем изменения их способа отображения браузером.

6. Добавьте свойство display: block; в стиль #mainNav.

Мы изменяем параметры отображения тега <a> таким образом, чтобы он обрабатывался браузером как абзац текста или другой блочный элемент, с расположением ссылок в виде списка, в точности одна над другой. Осталась единственная проблема — ссылки растянуты по всей ширине окна браузера — слишком много для кнопок. Исправим ситуацию, ограничив ширину тега <ul> в соответствующем стиле.

ПРИМЕЧАНИЕ

Если вы будете просматривать веб-страницу в браузере Internet Explorer 6 или его более ранних версиях, то заметите небольшие промежутки между кнопками. Не волнуйтесь. Мы исправим эту ошибку браузера далее.

7. Во внутренней таблице найдите стиль #mainNav и добавьте в него свойство width: 175px;.

При ширине списка 175 пикселов ссылки по-прежнему слишком широки, хотя и ограничиваются шириной элемента-контейнера (тег <ul>). В большинстве случаев список ссылок заключен в какой-то элемент разметки веб-страницы (например, боковое меню), для которого уже определена ширина, поэтому можете пропустить этот шаг (о том, как создавать боковые меню, читайте в части 3).

Сейчас перейдем к самому интересному.

8. Добавьте в стиль #mainNav свойство background следующим образом:

```
#mainNav a {
  color: #000;
  font-size: 11px;
  text-transform: uppercase;
  text-decoration: none;
  border: 1px dashed #999;
  padding: 7px 5px;
  display: block;
  background-color: #E7E7E7;
  background-image: url(images/nav.png);
  background-repeat: no-repeat;
  background-position: 0 2px;
}
```

Эти строки изменяют цвет фона ссылок на серый и устанавливают одиночное изображение с левой стороны каждой кнопки (см. рис. 9.15, *5*). Здесь тоже нужно кое-что исправить: текст ссылки накладывается на значок, а линии границ между кнопками имеют толщину, равную 2 пикселам (теоретически границы имеют толщину 1 пиксел, но сливающиеся воедино линии соседних ссылок образуют линию толщиной 2 пиксела).

ПРИМЕЧАНИЕ

Используя сокращенный вариант свойства background, можно изменить код в шаге 8 на такой: background: #E7E7E7 url(images/nav.png) no-repeat 0 2px;.

9. Удалите верхнюю границу и измените отступ в стиле #mainNav, чтобы он выглядел следующим образом:

```
#mainNav a {
  color: #000;
  font-size: 11px;
  text-transform: uppercase;
  text-decoration: none;
  border: 1px dashed #999;
  border-bottom: none;
  padding: 7px 5px 7px 30px;
  display: block;
  background-color: #E7E7E7;
  background-image: url(images/nav.png);
  background-repeat: no-repeat;
  background-position: 0 2px;
}
```

Выглядит неплохо: текст каждой ссылки отделен от значка, границы выделяются. Однако нижняя граница последней ссылки исчезла. Существует несколько способов разобраться с этим. Один состоит в том, чтобы создать стилевой класс с надлежащим параметром нижней границы border-bottom, а затем применить его к этой ссылке. Будет гораздо проще применить нижнюю границу к тегу <ul>,

содержащему список ссылок (поскольку этот тег не имеет отступов, нет промежутка, отделяющего верх <ul> от верхней стороны этой первой ссылки).

10. Добавьте верхнюю границу в стиль ul#mainNav, чтобы он выглядел следующим образом:

```
#mainNav {
  margin: 0;
  padding: 0;
  list-style: none;
  width: 175px;
  border-bottom: 1px dashed #999;
}
```

Вот и все, мы создали простейшую панель навигации с применением границ, отступов, фонового цвета и изображений (см. рис. 9.15, *б*).

Добавление ролловеров и выделение текущего раздела сайта стилем выбранных ссылок

Пришло время усовершенствовать панель навигации и придать ей некоторую интерактивность. Во-первых, обеспечим кнопкам главной панели управления эффект ролловера. Они должны изменять свой вид, акцентируя внимание посетителя на том, какую кнопку он собирается нажать.

Неплохо было бы дать посетителю знать, в каком разделе (на какой странице) сайта он находится. Мы можем обеспечить созданной панели навигации автоматическую интерактивность. Она будет просто изменять свой стиль в соответствии с выбранным разделом страницы. Звучит совсем просто, но в действительности это потребует некоторой разметки и настроек, как вы увидите в последующих шагах.

Эффект ролловера реализовать просто, но начнем по порядку.

1. Добавьте в конце таблицы стилей файла nav_bar.html следующий стиль:

```
#mainNav a:hover {
  font-weight: bold;
  background-color: #B2F511;
  background-position: 3px 50%;
}
```

Он определяет внешний вид ссылки-кнопки в состоянии hover. Стиль изменяет шрифт текста кнопки с обычного на полужирный, а также цвет фона на ярко-зеленый. Кроме того, он использует метод CSS Sprites, упоминавшийся ранее. То же самое изображение применяется в шаге 8 в прошлом разделе — оно в действительности содержит три различных значка (рис. 9.16). В этом случае изображение центрируется внутри кнопки, отображая средний значок файла.

Теперь при наведении указателя мыши на любую кнопку и его перемещении на ней кнопка моментально изменяет свой вид (откройте веб-страницу в своем браузере и посмотрите сами).

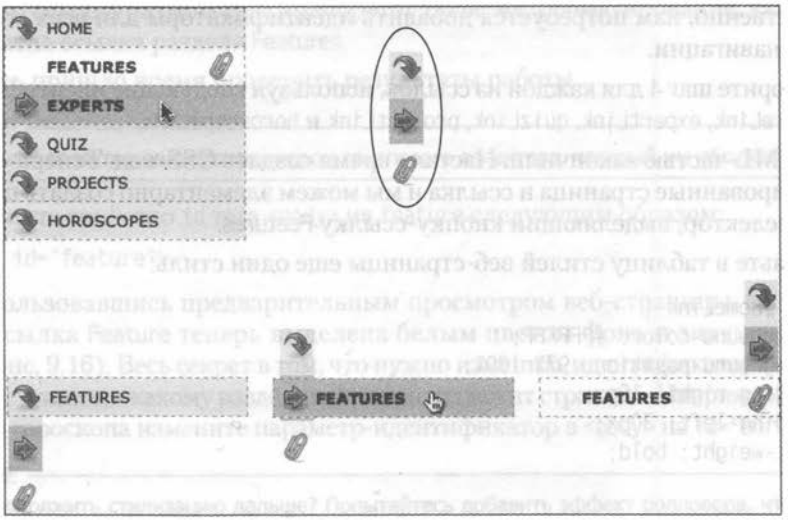


Рис. 9.16. С помощью несложного CSS-кода можно создать интерактивный эффект ролловера для кнопок панели навигации, а также автоматически подсветить раздел сайта текущей страницы

Теперь сделаем панель навигации более информативной, выделив кнопку, соответствующую текущему разделу, страницу которого открыл посетитель сайта. Чтобы сделать это, нам потребуется идентифицировать в HTML-коде панели навигации две вещи: раздел, к которому принадлежит страница, и разделы, на которые указывает каждая ссылка. В нашем примере предположим, что открыта домашняя страница.

ПРИМЕЧАНИЕ

Другим вариантом будет создание стилевого класса, который изменяет внешний вид ссылки, представляющей раздел страницы. Для веб-страницы гороскопа ссылка панели навигации может выглядеть следующим образом: `<a href="/horoscopes/" class="u_r_here">Horoscopes</a>`.

2. Найдите тег `<body>` и добавьте в него идентификатор `id="home"`:

```
<body id="home">
```

Теперь, когда мы знаем, какому разделу сайта принадлежит текущая открытая страница, можно использовать производный селектор для создания конкретного CSS-стиля, который будет применен к веб-странице раздела **Features**. Далее мы должны маркировать разделы, на которые указывает каждая ссылка-кнопка, что достигается путем добавления идентификаторов.

3. Найдите в HTML-коде панели навигации ссылку **Features** и добавьте в нее идентификатор `id="homeLink"`, чтобы тег выглядел следующим образом:

```
<a href="/index.html" id="homeLink">Home</a>
```

Этот идентификатор однозначно определяет ссылку, предоставляя возможность создания стиля, который будет применен только к ней.

Естественно, нам потребуется добавить идентификаторы для всех ссылок панели навигации.

4. Повторите шаг 4 для каждой из ссылок, используя следующие идентификаторы: `featureLink`, `expertLink`, `quizLink`, `projectLink` и `horoscopeLink`.

   С HTML-частью закончили. Настало время создать CSS-код. Теперь у нас есть маркированные страница и ссылка и мы можем элементарно создать производный селектор, выделяющий кнопку-ссылку **Features**.

5. Добавьте в таблицу стилей веб-страницы еще один стиль:

```
#home #homeLink {
  background-color: #FFFFFF;
  background-position: 97% 100%;
  padding-right: 15px;
  padding-left: 30px;
  font-weight: bold;
}
```

   Мы уже рассматривали эти свойства прежде. Снова вы используете метод CSS Sprites для регулировки позиции фонового изображения. На этот раз изображение отодвигается на 97 % вправо, а его нижняя часть помещается в нижнюю часть кнопки. Другими словами, значок будет отображаться внизу изображения (см. рис. 9.16). О том, как процентные значения работают с фоновыми изображениями, мы говорили в гл. 8.

   В данном случае наибольший интерес представляет селектор `#home #homeLink`. Это очень точный, явно определенный селектор, применяемый только к ссылке с идентификатором `homeLink`, которая также заключена внутри тега `<body>` с идентификатором `home`. Если вы измените идентификатор страницы, например, на `quiz`, с кнопки-ссылки раздела **Home** исчезнет выделение.

   Просмотрите веб-страницу в браузере, чтобы увидеть результат: теперь ссылка **Home** имеет белый фон и значок скрепки. Чтобы проделать все то же самое с остальными ссылками, вы должны немного расширить селектор.

6. Отредактируйте селектор только что созданного стиля:

```
#home #homeLink,
#feature #featureLink,
#expert #expertLink,
#quiz #quizLink,
#project #projectLink,
#horoscope #horoscopeLink {
  background-color: #FFFFFF;
  background-position: 97% 100%;
  padding-right: 15px;
  padding-left: 30px;
  font-weight: bold;
}
```

   Конечно, довольно объемный CSS-код. Этот стиль теперь адресуется ко всем ссылкам панели навигации, но только при выполнении определенных условий. Если вы в дальнейшем измените идентификатор `id` тега `<body>`, например, на

quiz, то ссылка раздела Quiz приобретет такое же форматирование, каким была выделена ссылка раздела Features.

Теперь пришло время проверить результаты работы.

ПРИМЕЧАНИЕ

Этот длинный селектор является примером группового селектора, который мы обсуждали в гл. 3.

7. Измените свойство id тега <body> на feature следующим образом:

```
<body id="feature">
```

Воспользовавшись предварительным просмотром веб-страницы, вы увидите, что ссылка Feature теперь выделена белым цветом фона и значком скрепки (см. рис. 9.16). Весь секрет в том, что нужно изменить идентификатор тега <body>, чтобы указать, к какому разделу сайта принадлежит страница. Например, для страницы гороскопа измените параметр-идентификатор в <body> на id="horoscope".

ПРИМЕЧАНИЕ

Готовы продолжить стилизацию дальше? Попытайтесь добавить эффект ролловера, чтобы закончить стиль, созданный в шаге 6 (используйте псевдокласс :hover в качестве составляющей селектора: #quiz #quizLink:hover). Попробуйте также добавить другое изображение для ссылки главной страницы Home (можете использовать файл home.png в папке с рисунками images).

Исправление ошибок Internet Explorer

Что бы мы описывали в обучающем уроке, если бы не существовало ошибок Internet Explorer? К сожалению, панель навигации не работает надлежащим образом в этом браузере шестой и более ранних версий (в IE 7 и 8 все работает исправно). Во-первых, есть промежуток между кнопками. Кроме того, браузер принимает за выбираемую ссылку не всю область, а только текст (рис. 9.17). В других же браузерах активизация кнопки-ссылки происходит при перемещении указателя мыши по любой области кнопки, включая текст ссылки и сам фон (пустое пространство рядом с текстом). К счастью, устранить ошибки несложно.

1. Отредактируйте стиль #mainNav a, добавив свойство zoom: 1:

```
#mainNav a {
  text-decoration: none;
  color: #000000;
  font-size: 11px;
  text-transform: uppercase;
  border: 1px dashed #999999;
  border-bottom: none;
  padding: 7px 5px 7px 30px;
  display: block;
  background-color: #E7E7E7;
  background-image: url(images/nav.png);
  background-repeat: no-repeat;
  background-position: 0 2px;
  zoom: 1;
}
```

Этого странного кода достаточно для настройки IE 6.



Рис. 9.17. Веб-страница при открытии в Internet Explorer имеет ряд проблем с панелью навигации — это промежуток между кнопками-ссылками и их подсветка только при наведении указателя мыши непосредственно на текст ссылки

2. Если у вас установлен Internet Explorer 6, просмотрите в нем страницу.

   Навигационная панель должна теперь работать в этом браузере так же корректно, как и в более «здравомыслящих» браузерах Internet Explorer 8, Firefox, Opera и Safari.

Окончательную версию навигационной панели смотрите в файле `09_finished\nav_bar\nav_bar_vertical.html`.

ПРИМЕЧАНИЕ

Во многих случаях я рекомендую эти специфические стили, ориентированные только на работу в Internet Explorer, изолировать от остальных. Как правило, они содержат информацию, не имеющую смысла в CSS, которая исправляет недоработки Internet Explorer. Перечитывая таблицы стилей в будущем, вы, наверное, удивитесь, зачем включили сюда CSS-код, и сами можете запутаться в нем. Предпочтительней будет отделить его, поместить во внешнюю таблицу стилей, присоединяя с использованием условных комментариев Microsoft. Детальное описание методики читайте в разд. «Управление браузером Internet Explorer» гл. 15.

Переход от вертикальной панели навигации к горизонтальной

Предположим, вы хотите создать горизонтальную панель навигации в верхней части веб-страницы. Никаких проблем — бóльшую часть самой сложной работы мы уже выполнили. Чтобы расположить кнопки горизонтально в одну строку, необходимо немного изменить уже созданную веб-страницу (мы будем дорабатывать

наш последний файл nav_bar.html, поэтому, если вы хотите сохранить вертикальную панель навигации, прежде чем продолжить, сделайте копию файла).

1. Убедитесь в том, что вы выполнили все шаги по созданию вертикальной панели навигации, и откройте файл nav_bar.html в HTML-редакторе.

   Сейчас вы увидите, как просто изменить ориентацию панели. Сначала подчистим кое-что из того, что мы уже сделали. Вам нужно удалить ширину, которую вы установили для тега <ul> ранее. Эта ширина препятствовала тому, чтобы навигационные кнопки охватывали всю длину страницы. Но, поскольку тегу <ul> необходимо расшириться, чтобы вмещать набор горизонтальных кнопок, эта ширина подойдет нам без изменений.

2. Найдите стиль #mainNav и удалите в нем свойство width: 175px;.

   Теперь привожу сам секрет преобразования вертикальной панели навигации в горизонтальную.

3. Добавьте в таблицу новый стиль (сразу после #mainNav):

```
#mainNav li {
  float: left;
  width: 12em;
}
```

   Этот стиль применяется к тегу <li> (представляющему собой элементы списка, каждый из которых содержит свою ссылку). Первая команда устанавливает для тега выравнивание по левому краю. Таким образом, каждый последующий тег <li> располагается с правой стороны предыдущего (такой же эффект вы могли наблюдать в обучающем уроке по созданию фотогалереи в гл. 8). Кроме того, устанавливая ширину тега <li>, мы одновременно определяем ширину для всех кнопок панели навигации. В данном случае значение 12ems обеспечивает достаточный размер, чтобы вместить самый длинный текст названия ссылки. Если будет необходимо увеличить длину, вы должны будете увеличить это значение.

СОВЕТ

Использование для установки ширины кнопок значений em считается лучшим методом. Если посетитель увеличит размер шрифта, кнопка со значением ширины в em также увеличится в размерах. Тем не менее большинство браузеров в наши дни используют функцию масштабирования страницы, так что, когда вы увеличите текст, вы на самом деле увеличите всю страницу, и даже кнопки и другие элементы, значения ширины которых определяются в пикселах, увеличатся в размерах. Поэтому в настоящее время большинство дизайнеров используют значения в пикселах практически для всего.

При просмотре веб-страницы в браузере вы увидите, что с панелью навигации в основном все в порядке. Требуются только небольшие корректировки (смотрите обведенные кружком области на рис. 9.18, *1*). Во-первых, нижняя граница, созданная нами в шаге 10, занимает всю ширину тега <ul>, более широкого, чем кнопки, непосредственно образующие панель навигации. Еще более странно то, что нижняя граница расположена не на заднем плане, а поверх кнопок. Кроме того, поскольку последние расположены рядом друг с другом, их левые и правые границы сливаются между собой, образуя линии толщиной 2 пиксела. Сейчас мы решим эти проблемы.

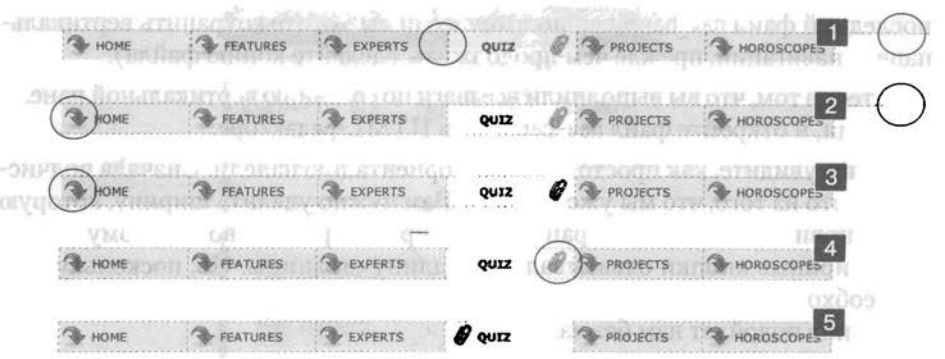


Рис. 9.18. Преобразование вертикальной панели в более компактную с горизонтальным расположением кнопок требует выполнения всего нескольких действий; гораздо больше усилий нужно приложить для настройки стилей, определяющих параметры границ и позиционирование фонового изображения

4. В стиле `#mainNav a` измените свойство `border-bottom: none;` на `border-left: none;`.

   Это действие приводит к удалению левых границ кнопок, чтобы они не удваивались, и в то же время к нижнему краю добавляется граница. Однако нижняя граница тега `<ul>` по-прежнему отображается поверх кнопок, и с левой стороны панели навигации отсутствует граница на крайней левой кнопке (смотрите обведенные кружком области на рис. 9.18, *2*). Никаких проблем — просто измените границы `<ul>`.

5. Найдите стиль `#mainNav` и измените в нем свойство `border-bottom: 1px dashed #999999;` на `border-left: 1px dashed #999999;`.

   Сейчас при просмотре веб-страницы в браузере вы заметите, что граница над кнопками исчезла, но с левой стороны панели навигации так и не появилась (см. рис. 9.18, *3*). Это еще одно свидетельство сложности использования плавающих элементов. Они вышли из обычного потока содержимого документа, и браузер больше не видит их в качестве составной части тега `<ul>`. Тег уменьшился до минимальных размеров, именно поэтому его нижняя граница отображается поверх остального содержимого (это может показаться слишком запутанным, но так оно и есть, именно поэтому работе с плавающими элементами посвящена отдельная глава книги — см. гл. 12).

   К счастью, несмотря на то, что проблема кажется достаточно сложной, ее решение совсем простое. Добавим одно CSS-свойство к маркированному списку.

6. Добавьте два свойства в конце стиля `#mainNav`:

```
#mainNav {
  margin: 0;
  padding: 0;
  list-style: none;
  border-left: 1px dashed #999999;
  overflow: hidden;
  zoom: 1; /* for IE 6 */
}
```

Код `overflow: hidden` заставляет расширяться неупорядоченный список. Почему свойство работает? Об этом вы узнаете в гл. 12. А код `zoom: 1` предназначен только для Internet Explorer 6.

Наконец, значок скрепки, выровненный по правому краю нажатой кнопки Quiz, выглядит неплохо (см. рис. 9.18, *4*), но лучше позиционировать его по левому краю.

7. Найдите стиль выбранной кнопки, созданный в шаге 6 (это стиль с длинным селектором). Измените координаты его свойства `background` с `97% 100%` на `3px 100%`. Теперь стиль должен выглядеть следующим образом:

```
#home #homeLink,
#feature #featureLink,
#expert #experLink,
#quiz #quizLink,
#project #projectLink,
#horoscope #horoscopeLink
{
  background-color: #FFFFFF;
  background-position: 3px 100%;
  padding-right: 15px;
  padding-left: 30px;
  font-weight: bold;
}
```

Просмотрите веб-страницу в браузере, и вы обнаружите, что горизонтальная панель навигации полностью работоспособна, а также замечательно выглядит (см. рис. 9.18, *5*). Кроме того, она отлично работает даже в браузере Internet Explorer.

ПРИМЕЧАНИЕ

Возможно, вы захотите центрировать текст кнопок-ссылок панели навигации. Для этого нужно сделать две вещи: добавить в стиль `#mainNav` свойство `text-align: center` и подкорректировать свойство `left-padding` этого же стиля, чтобы текст был расположен точно по центру кнопок.

Законченная версия веб-страницы, которая должна получиться, находится в файле `nav_bar_horizontal.html` в папке **09_finished\nav_bar** учебного материала.

10 Форматирование таблиц и форм

Возможности CSS не ограничиваются форматированием текста, изображений и ссылок. Вы можете создавать информационные таблицы с расписаниями, результатами спортивных мероприятий и списки воспроизведения с музыкой, которые становятся намного более читабельными, если к ним добавить границы, фоновые рисунки и другие визуальные улучшения. Ко всему прочему, вы можете использовать CSS для разработки соответствующих элементов форм, которые помогут вашим посетителям сделать заказ, подписаться на новости или воспользоваться вашими самыми новыми интернет-приложениями. В этой главе вы узнаете, как вывести на экран таблицы и формы с помощью HTML, а также как их скомпоновать и применить к ним соответствующие стили посредством CSS. В уроке в конце главы вы создадите таблицу и форму, используя приемы, которым попутно обучитесь.

Правильное использование таблиц

За короткую историю существования Интернета HTML-таблицы получили очень широкое распространение. Изначально созданные для отображения данных в табличном виде, они стали очень популярным инструментом для создания разметки. Столкнувшись с некоторыми ограничениями в HTML, дизайнеры творчески подошли к решению этой проблемы и начали применять столбцы и строки таблиц для размещения в них таких элементов страниц, как заголовки и боковые поля. Как вы увидите в части 3 этой книги, CSS намного лучше справляется с компоновкой веб-страниц. Вы можете сконцентрировать все ваше внимание на использовании таблиц по их прямому назначению — для отображения данных (рис. 10.1).

HTML и XHTML имеют в своем распоряжении внушительное количество тегов, предназначенных для создания страниц. С помощью этого фрагмента кода на HTML создается очень простая таблица, которую вы можете видеть на рис. 10.2.

```
<table>
  <caption align="bottom">
    Table 1: CosmoFarmer.com's Indoor Mower Roundup
```

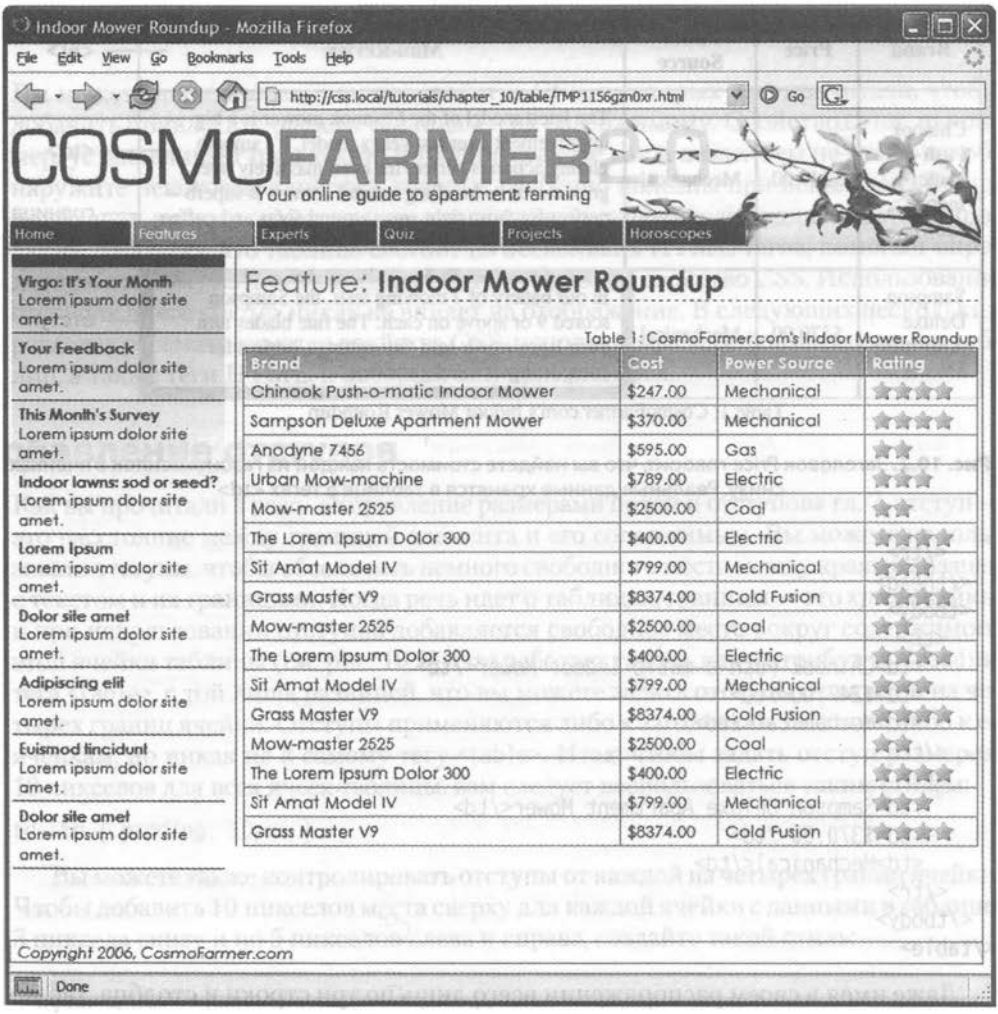


Рис. 10.1. Благодаря CSS в этой таблице, содержащей информацию о домашних газонокосилках, шрифты изменяются на более привлекательные, создаются границы и меняются фоновые цвета, но вся лежащая в основе структура создана с помощью HTML

```
</caption>
<colgroup>
  <col id="brand" />
  <col id="price" />
  <col id="power" />
</colgroup>
<thead>
  <tr>
    <th scope="col">Brand</th>
    <th scope="col">Price</th>
    <th scope="col">Power Source</th>
```

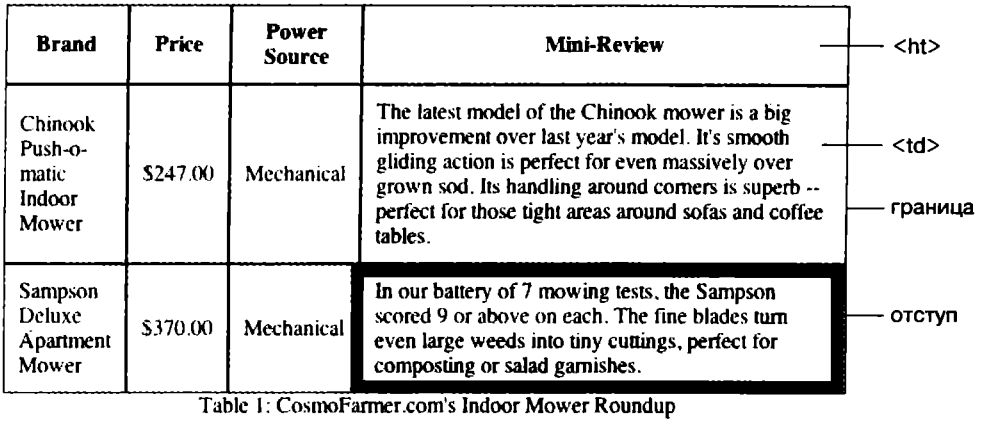


Рис. 10.2. Заголовок Price говорит, что вы найдете стоимость каждой из газонокосилок в ячейках снизу. Реальные данные хранятся в таблице в тегах <td>

```
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Chinook Push-o-matic Indoor Mower</td>
      <td>$247.00</td>
      <td>Mechanical</td>
    </tr>
    <tr>
      <td>Sampson Deluxe Apartment Mower</td>
      <td>$370.00</td>
      <td>Mechanical</td>
    </tr>
  </tbody>
</table>
```

Даже имея в своем распоряжении всего лишь по три строки и столбца, таблица использует девять уникальных тегов HTML: <table>, <caption>, <colgroup>, <col>, <thead>, <tbody>, <tr>, <th> и <td>. В целом наличие такого большого кода на HTML — не самый хороший показатель, но, с другой стороны, различные теги позволяют воспользоваться многими полезными средствами при создании стилей CSS. Заголовок для каждого из столбцов таблицы, задаваемый в теге <th>, может иметь различный вид в разных ячейках таблицы, если вы будете использовать при создании стиля тег <th>. Это избавит вас от нудной работы по созданию множества различных классов, таких как .tableHeader, а также от необходимости вручную задавать их для каждой ячейки таблицы. В следующем разделе вы познакомитесь с примерами использования различных тегов и ощутите все их преимущества.

ПРИМЕЧАНИЕ

Для получения более подробной информации о создании таблиц с помощью HTML посетите страницу www.456bereastreet.com/archive/200410/bring_on_the_tables/.

Создание стилей для таблиц

Вы можете использовать многие свойства CSS, о которых уже прочитали, чтобы добавить привлекательности таблицам и их содержимому. Свойство `color`, например, устанавливает цвет текста в таблице, так же как и везде. Тем не менее вы обнаружите некоторые свойства, которые особенно полезны при использовании их в таблицах, а также свойства, предназначенные исключительно для их форматирования. Тот факт, что таблица состоит из нескольких HTML-тегов, помогает определить, к какому из них применить соответствующее свойство CSS. Использование отступов в теге `<table>` никак не влияет на отображение. В следующих нескольких разделах рассматриваются свойства CSS, используемые для форматирования таблиц, а также теги HTML, к которым они применяются.

Добавление отступов

Как вы прочитали в разд. «Управление размерами полей и отступов» гл. 7, отступ — это расстояние между границей элемента и его содержимым. Вы можете использовать отступы, чтобы обеспечить немного свободного места между краями абзацев с текстом и их границами. Когда речь идет о таблицах, границы — это *края* ячейки, и при использовании отступов добавляется свободное место вокруг содержимого этой ячейки таблицы (см. рис. 10.2). Это работает так же, как и атрибут `cellpadding` тега `<table>`, с той лишь разницей, что вы можете задать отступы от каждой из четырех границ ячейки. Отступы применяются либо к заголовкам таблицы, либо к ее ячейкам, но никак не к самому тегу `<table>`. Итак, чтобы задать отступ размером 10 пикселов для всех ячеек таблицы, вам следует воспользоваться таким стилем:

```
td, th { padding: 10px; }
```

Вы можете также контролировать отступы от каждой из четырех границ ячейки. Чтобы добавить 10 пикселов места сверху для каждой ячейки с данными в таблице, 3 пиксела снизу и по 5 пикселов слева и справа, создайте такой стиль:

```
td {
  padding-top: 10px;
  padding-right: 5px;
  padding-bottom: 3px;
  padding-left: 5px;
}
```

Или используйте сокращенную версию свойства `padding`:

```
td {
  padding: 10px 5px 3px 5px;
}
```

СОВЕТ

Если вы с помощью тега `<img>` поместите в ячейку изображение и заметите, что внизу таблицы появится нежелательное свободное место, то установите block в качестве значения свойства display (см. подраздел «Отображение встроенных и блочных элементов» разд. «Управление размерами полей и отступов» гл. 7). Чтобы получить больше информации, посетите страницу http://developer.mozilla.org/en/docs/Images,_Tables,_and_Mysterious_Gaps.

Настройка горизонтального и вертикального выравнивания

Чтобы настроить месторасположение содержимого *внутри* самой ячейки, используйте свойства `text-align` и `vertical-align`. Первое свойство применяется для управления горизонтальным выравниванием и может принимать значения `left`, `right`, `center` и `justify` (рис. 10.3). Это унаследованное свойство (см. гл. 4, чтобы узнать больше о наследовании). Если вы хотите выровнять содержимое всех ячеек таблицы по правому краю, создайте следующий стиль:

```
table { text-align: right; }
```

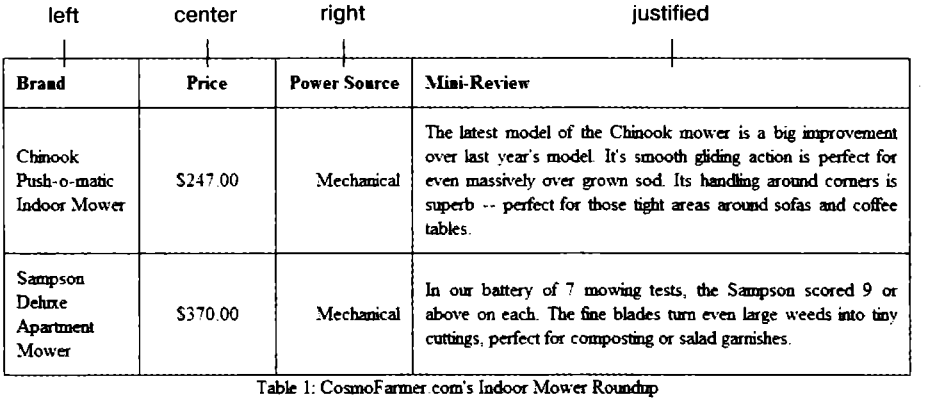


Рис. 10.3. В CSS при необходимости изменений в таблице вам придется внести их только во внешний файл с таблицами стилей, а не в 10 000 отдельных тегов <td>

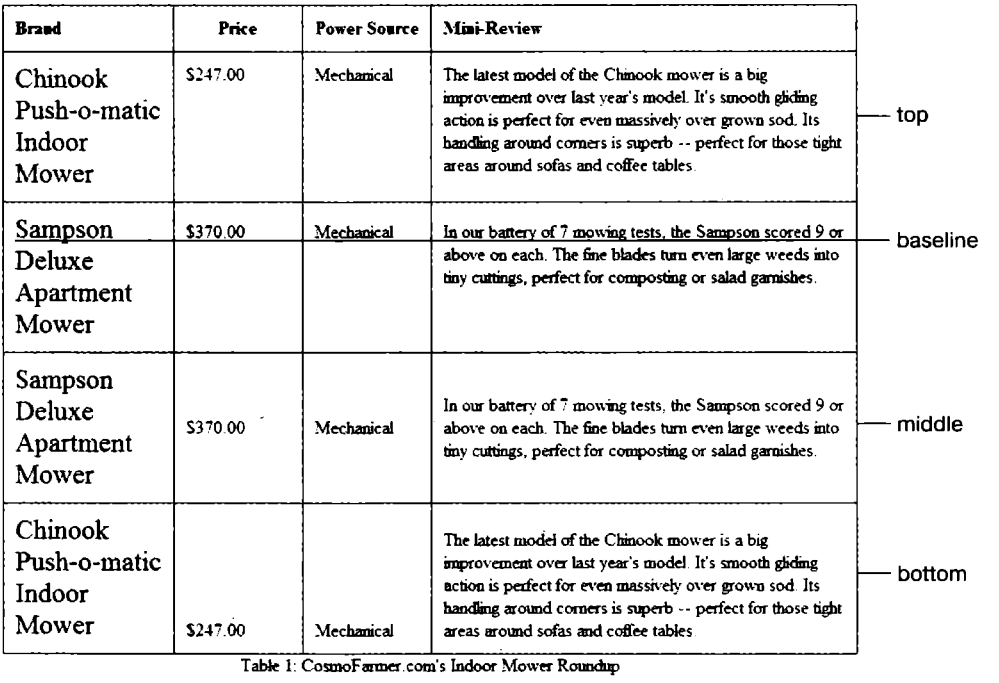
Это свойство удобно использовать в тегах <th>, так как браузеры обычно выравнивают его по центру. Простой стиль вида `th { text-align: left; }` выровняет и заголовки таблицы.

Свойство CSS `text-align` по отношению к ячейкам таблицы работает так же, как и атрибут `align` тега <td>. Тем не менее старайтесь использовать CSS, так как это поможет вам хранить информацию о стилях во внешней таблице стилей. В этом случае, если вы решите изменить выравнивание по правому краю на выравнивание по левому краю, вам придется внести изменения только во внешний файл с таблицами стилей.

У ячеек в таблицах есть также такой параметр, как высота. Обычно браузеры выравнивают содержимое вертикально по центру ячейки (см. пример `middle` на рис. 10.4). Вы можете изменить это с помощью свойства `vertical-align`. Используйте одно из этих четырех значений: `top`, `baseline`, `middle` или `bottom`. Значение `top` помещает содержимое ячейки вверху, `middle` — по центру, а `bottom` — внизу. При использовании значения `baseline` выравнивание происходит так же, как и при использовании значения `top`, за исключением того, что браузер выравнивает первую строку текста в каждой ячейке заданной строки таблицы (рис. 10.4). Скорее всего, вы даже не заметите тонкостей работы параметра `baseline`. В отличие от `text-align` свойство `vertical-align` не унаследовано, поэтому вы можете использовать его только в стилях, которые применяются прямо в тегах <th> и <td>.

СОВЕТ

Итак, изученное форматирование таблиц применяется сразу ко всем вашим таблицам. Когда же вы хотите использовать для каждой таблицы свой индивидуальный стиль (или для каждой ее ячейки), измените выбранный селектор. Чтобы использовать особый дизайн в определенной таблице, задайте для нее имя класса — <table class="stocks"> — и создайте несколько производных селекторов вида .stocks td или .stocks th, чтобы применить различное форматирование к каждой ячейке. Если вы хотите изменить форматирование определенной ячейки в таблице, используйте в теге класс <td class="subtotal"> и создайте класс со стилем для форматирования этой ячейки.



Brand	Price	Power Source	Mini-Review
Chinook Push-o-matic Indoor Mower	$247.00	Mechanical	The latest model of the Chinook mower is a big improvement over last year's model. It's smooth gliding action is perfect for even massively over grown sod. Its handling around corners is superb -- perfect for those tight areas around sofas and coffee tables.
Sampson Deluxe Apartment Mower	$370.00	Mechanical	In our battery of 7 mowing tests, the Sampson scored 9 or above on each. The fine blades turn even large weeds into tiny cuttings, perfect for composting or salad garnishes.
Sampson Deluxe Apartment Mower	$370.00	Mechanical	In our battery of 7 mowing tests, the Sampson scored 9 or above on each. The fine blades turn even large weeds into tiny cuttings, perfect for composting or salad garnishes.
Chinook Push-o-matic Indoor Mower	$247.00	Mechanical	The latest model of the Chinook mower is a big improvement over last year's model. It's smooth gliding action is perfect for even massively over grown sod. Its handling around corners is superb -- perfect for those tight areas around sofas and coffee tables.

Table 1: CosmoFarmer.com's Indoor Mower Roundup

Рис. 10.4. Свойство CSS vertical-align — эквивалент атрибута align тега <td>. Когда в ячейке используются отступы, ее содержимое на самом деле никогда не выравнивается по верхним или нижним линиям границы: всегда есть промежуток, равный размеру отступа

Создание границ

Свойство CSS `border` (см. разд. «Добавление границ» гл. 7) работает с таблицами так же хорошо, как и с другими элементами, но вам нужно кое-что помнить. Во-первых, использование границ в стиле, который занимается форматированием тега `<table>`, выделяет только таблицу, а не какую-либо определенную ее ячейку. Во-вторых, при использовании границ в ячейках (`td { border: 1px solid black; }`) образуется видимый интервал между ячейками сверху, как это показано на рис. 10.5. Чтобы правильно управлять границами, вам необходимо понять принцип работы атрибута `cellspacing` тега `<table>`, а также свойства CSS `border-collapse`.

- **Управление промежутком между ячейками таблицы.** Если не указано другого, браузеры расставляют в таблице интервалы между ячейками около 2 пикселов. Этот промежуток действительно заметен, когда вы задаете границы для ячеек.

В CSS 2.1 доступно свойство `border-spacing`, которое можно использовать, чтобы контролировать размер этого промежутка. Однако поскольку Internet Explorer версии 7 и ниже не поддерживает его, лучше используйте атрибут `cellspacing` тега `<table>`. Вот код на HTML, который вставляет промежуток шириной 10 пикселов между всеми ячейками: `<table cellspacing="10px">` (при установке значения равным 0 все интервалы исчезают полностью, но если вы этого и хотите добиться, то лучше используйте свойство CSS `border-collapse`, которое мы обсудим дальше).

- **Удаление двойных границ.** Даже если вы уберете промежутки между ячейками, границы, заданные для ячеек, будут удваиваться. Это происходит потому, что нижняя граница одной ячейки добавляется к верхней границе лежащей внизу ячейки, и в итоге образуется линия, которая в два раза толще заданной ширины границы (см. рис. 10.5, *посередине*). Самый лучший способ избавиться от этого (а также избавиться от промежутков между ячейками) — использовать свойство `border-collapse`. Оно может принимать два значения — `separate` и `collapse`. Значение `separate` эквивалентно тому, как обычно и отображаются таблицы: с промежутками между ячейками и двойными границами. Задавая окаймление границ таблицы, можно избавиться от интервалов и двойных границ (рис. 10.5, *внизу*). Используйте значение `collapse` в стилях для форматирования таблиц:

```
table { border-collapse: collapse; }
```

Brand	Price	Power Source	Mini-Review
Chinook Push-o-matic Indoor Mower	$247.00	Mechanical	The latest model of the Chinook mower is a big improvement over last year's model. It's smooth gliding action is perfect for even massively over grown sod. Its handling around corners is superb -- perfect for those tight areas around sofas and coffee tables.

Table 1: CosmoFarmer com's Indoor Mower Roundup

Brand	Price	Power Source	Mini-Review
Chinook Push-o-matic Indoor Mower	$247.00	Mechanical	The latest model of the Chinook mower is a big improvement over last year's model. It's smooth gliding action is perfect for even massively over grown sod. Its handling around corners is superb -- perfect for those tight areas around sofas and coffee tables.

Table 1: CosmoFarmer com's Indoor Mower Roundup

Brand	Price	Power Source	Mini-Review
Chinook Push-o-matic Indoor Mower	$247.00	Mechanical	The latest model of the Chinook mower is a big improvement over last year's model. It's smooth gliding action is perfect for even massively over grown sod. Its handling around corners is superb -- perfect for those tight areas around sofas and coffee tables.

Table 1: CosmoFarmer.com's Indoor Mower Roundup

Рис. 10.5. Браузеры обычно вставляют пробелы между всеми ячейками в таблице

ВНИМАНИЕ

HTML-теги, которые используются для создания таблиц, также содержат атрибуты, выполняющие те же задачи, что и CSS. Атрибут border может добавить границу к таблице и к любой ее ячейке. В целом вам следует избегать использования этих атрибутов: CSS может сделать эту работу намного лучше, и для этого понадобится гораздо меньше кода.

Применение стилей к строкам и столбцам

Добавление чередования затемненных и незатемненных строк в таблицу, как на рис. 10.6, — это всего лишь один из обычных способов оформления таблицы. Изменив внешний вид каждой строки в таблице, вы позволите посетителям четко видеть нужные им данные. К сожалению, CSS (по крайней мере, с этой точки зрения) не позволяет нам сказать: «Сделай так, чтобы каждая строка в таблице выглядела таким образом!»



Font	Type	Foundry
Neue Helvetica	sans serif	Linotype
Aviano Sans	sans serif	insigne
Player	slab serif	Canada Type
URW Egyptienne	slab serif	URW++
Filosofia	serif	Emigre
Linotype Didot	serif	Linotype
New Century Schoolbook	serif	Linotype
Perpetua	serif	Monotype
Onyx	serif	Monotype
Platelet	monospaced	Emigre
Erased Typewriter 2	monospaced	Intellecta Design
Engraver's Old English	blackletter	Bitstream
Black Pearl	blackletter	FontMesa
FG Nina	script	Font Garden
Kidwriting	script	Corradine Fonts
Swan Song	script	Canada Type

Рис. 10.6. Благодаря чередующемуся изменению фонового цвета строк намного проще следить за всеми данными из строки целиком

Простое решение — применять класс вида `<tr class="odd">` к каждой строке, а затем создавать стиль для форматирования этой строки:

```
.odd { background-color: red; }
```

Вы также не ограничены цветами. Можно использовать фоновые изображения (см. разд. «Фоновые изображения» гл. 8) для создания более утонченного вида, например легкого затемнения строки таблицы с заголовками, как на рис. 10.6

(вы увидите похожий пример в обучающем уроке 1 этой главы). Вы можете также использовать селектор для каждой ячейки в этой строке. Это очень удобно, когда вы применяете стили ко всем ячейкам в одном столбце, используя для каждой определенный стиль и вид, например <td class="price">. Чтобы создать особый вид для ячейки, когда она расположена в нечетной строке, создайте стиль с таким селектором: .odd .price.

ПРИМЕЧАНИЕ

Узнать о том, как воспользоваться более быстрым решением с помощью CSS 3, вы сможете позже в этой книге. Но этот метод работает только в некоторых браузерах, так что есть смысл обратить внимание и на другой, использующий JavaScript для создания чередующихся затемненных и незатемненных строк в таблице. Более подробную информацию вы найдете на странице www.creativepro.com/article/view-source-javascript-designers.

В процессе форматирования столбцов надо немного схитрить. В HTML есть теги <colgroup> и <col>, указывающие на группу столбцов и на один отдельный столбец соответственно. Вы можете добавить по одному тегу <col> для каждого столбца в таблице и далее идентифицировать их с помощью класса или ID (см. HTML-код в разд. «Правильное использование таблиц» этой главы).

Для этих тегов есть только два вида свойств: width и свойства фона (background-color, background-image и т. д.). Однако и они могут быть очень полезными. Когда вы хотите установить ширину всех строк в столбце, то можете пропустить все атрибуты HTML и просто применить к столбцам стиль, используя стиль, примененный к тегу <col>. Скажем, у вас есть следующий кусок кода HTML: <col id="price">. Вы можете добавить этот стиль к таблице стилей и установить ширину каждой ячейки в данном столбце равной 200 пикселам:

```
#price { width: 200px; }
```

Похожим образом тег <colgroup> группирует вместе несколько столбцов. Когда вы устанавливаете ширину в этом теге, браузер автоматически устанавливает ее для каждого столбца из группы. В таблице, содержащей расписание самолетов, может быть несколько столбцов, в которых будет отображаться различное время вылета самолетов из Бостона в Чикаго. Вы можете воспользоваться тегом <colgroup>, чтобы сгруппировать эти столбцы и указать для них ID для идентификации: <colgroup id="dates">. Затем, чтобы установить для каждого столбца ширину 10 em, вы можете создать следующий стиль:

```
#dates{ width: 10em; }
```

Несмотря на то что свойство width здесь используется только в теге <colgroup>, браузер на самом деле использует это значение — 10em — в каждом столбце из группы.

Чтобы выделить столбец, вы можете воспользоваться свойствами фона. И снова считайте, что у вас есть тег <col>, к которому применен идентификатор price:

```
#price { background-color: #F33; }
```

Имейте в виду, что фон в столбцах помещается под ячейками, поэтому, если вы установите фоновый цвет или рисунок в тегах <td> или <th>, фон столбцов не будет виден.

Создание стилей для форм

Веб-формы — это основной способ общения пользователя с сайтом. Передав информацию в форму, вы можете подписаться на новости, поискать какие-либо товары в базе данных, ввести изменения в профиль или заказать набор конструкторов «Звездные войны», о котором уже давно мечтали. Совсем не обязательно, чтобы ваши формы выглядели так же, как и большинство других форм в Интернете. Применяя CSS, вы можете создать стили для полей ввода, чтобы они выглядели так же, как и другие элементы сайта: использовали те же шрифты, фоновые изображения и поля. Нет никаких отдельных CSS-свойств для форм, но вы можете использовать практически любое из описанных в этой книге свойств и для элемента типа «форма». Результаты тем не менее могут быть различными (рис. 10.7).

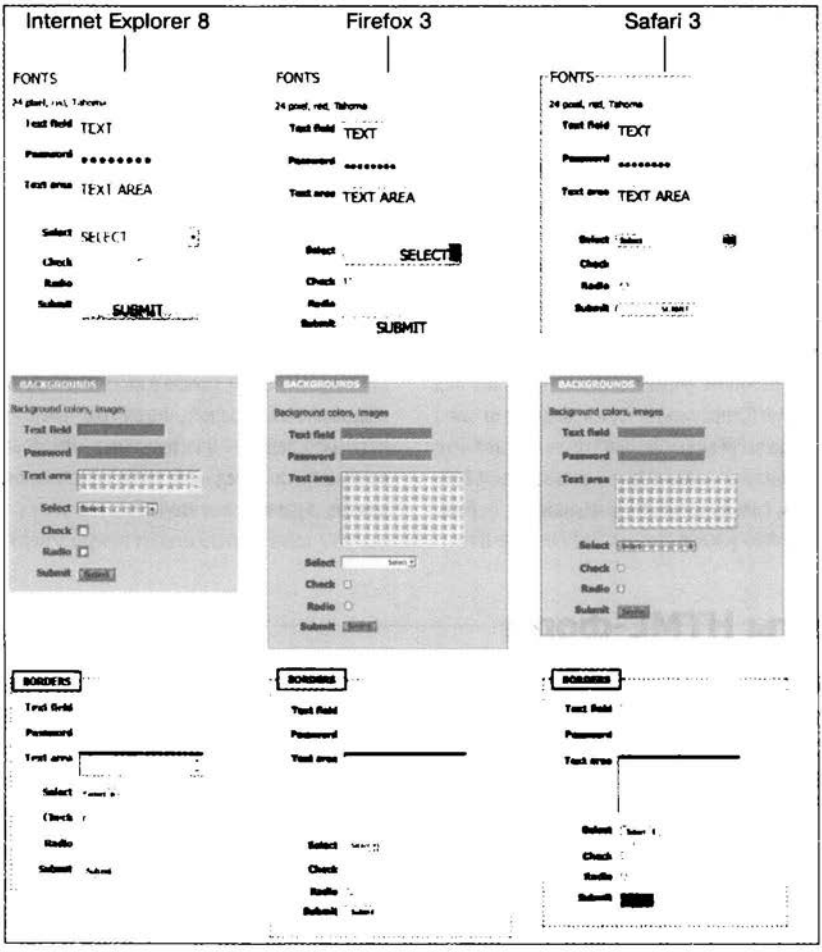


Рис. 10.7. Возможности по форматированию полей ввода с помощью стилей варьируются от браузера к браузеру: интерфейс Internet Explorer *(слева)*, Firefox *(посередине)* и Safari 3 *(справа)*

Поддержка в браузерах форматирования форм сильно различается. Safari 2 ограничивается использованием стилей только для некоторых элементов формы, таких как текстовые поля и теги `<fieldset>` и `<legend>`. Он не позволит вам изменить внешний вид кнопок, флажков, кнопок-переключателей или выпадающих меню. Но с появлением Safari 3 компания Apple, кажется, сдает позиции. В этом браузере не могут быть стилизованы всего лишь несколько элементов, таких как шрифты в выпадающем меню. В Internet Explorer и Firefox некоторые элементы могут выглядеть по-разному. Internet Explorer поддерживает фоновые цвета, границы вокруг флажков и переключателей, а Firefox — нет. Самое лучшее, что вы можете сделать, — это очень внимательно разрабатывать формы и не ждать, что они будут выглядеть одинаково во всех браузерах. В следующем разделе будет рассказано, какие свойства с какими тегами работают лучше, а также будут перечислены браузеры, которые правильно их интерпретируют.

ИНФОРМАЦИЯ ДЛЯ НОВИЧКОВ

Придерживайтесь одного и того же стиля при оформлении форм

Даже если бы поддержка форм, оформленных с использованием CSS, не варьировалась от браузера к браузеру (см. рис. 10.7), есть еще несколько причин, по которым надо с осторожностью подходить к изменению привычных и узнаваемых элементов интерфейса, таких как кнопки Submit (Отправить) или раскрывающиеся списки.

Большинство пользователей уже привыкли к тому, как выглядят и работают формы. В целом внешний вид кнопки Submit (Отправить) остается таким же от сайта к сайту. Когда люди видят ее, они уже знают, для чего предназначена эта кнопка и как ею пользоваться. Если вы слишком сильно измените внешний вид формы, это может привести к тому, что у пользователей возникнут некоторые затруднения при ее заполнении.

Добавив точечную границу к какому-либо полю на форме, вы можете добиться того, что пользователь просто не будет обращать на него внимания и будет его пропускать (см. примеры на рис. 10.7, *внизу по центру* и *внизу справа*). Если это текстовое поле предназначено для ввода адреса электронной почты, на который вы будете отправлять новости, можно в итоге потерять несколько пользователей из-за того, что они его просто пропустили. И напоследок — удостоверьтесь, что пользователи понимают, что перед ними размещена именно форма для ввода данных, а не что-либо иное.

Элементы HTML-форм

Большой выбор HTML-тегов позволяет вам создавать формы. Некоторые из них форматировать проще (например, текстовые поля), некоторые — сложнее (кнопки). Рассмотрим несколько простых тегов для форм, а также типы свойств, с которыми они работают.

- **Fieldset.** Тег `<fieldset>` предназначен для группировки элементов, связанных друг с другом. Большинство браузеров нормально отображают фоновые цвета, фоновые изображения и границы для этого тега. Однако в Internet Explorer фон может выходить за границы верхней линии тега (посмотрите на верхнюю часть среднего изображения на рис. 10.7, нас интересует левый столбец). При использовании отступов появляются пробелы между верхними границами тегов

<fieldset> и их содержимым (несмотря на то что Internet Explorer, к сожалению, игнорирует отступы сверху, вы можете применить сверху свойство margin к первому элементу тега <fieldset>).

- **Legend.** Тег <legend> идет за тегом <fieldset>, и в нем содержится название для группы. Оно появляется в центре верхней границы тега fieldset. Если в теге <fieldset> хранится, например, адрес доставки, вы можете добавить такой тег: <legend>Shipping Address</legend>. С помощью CSS вы можете изменить свойства шрифта тега <legend>, добавить фоновые цвета и изображения, а также определить собственные границы.
- **Текстовые поля ввода.** Теги <input type="text"> (<input type="text" /> в XHTML), <input type="password"> (<input type="password" />) и <textarea> создают текстовые поля в форме. Эти теги лучше всего поддерживаются браузерами. Вы можете изменить размер шрифта, тип шрифта, цвет и другие свойства текста в полях ввода, а также добавить границы и фоновые цвета. Internet Explorer, Firefox и Opera также позволяют добавлять фоновые изображения в текстовые поля; в Safari 2.0 такой возможности нет. Вы можете задать ширину этих полей с помощью CSS-свойства width. Однако свойство height поддерживается только тегом <textarea>.
- **Кнопки.** Кнопки на форме, такие как <input type="submit"> (<input type="submit" />), позволяют вашим пользователям передавать данные в форму, очищать содержимое формы или предпринимать какие-либо другие действия. Хотя браузер Safari 2.0 и его более ранние версии не поддерживают этого форматирования, другие браузеры предоставляют вам очень богатые возможности — вы можете создавать границы и изменять фон. Вы также можете выравнивать текст на кнопке по левому или правому краю либо посередине с помощью свойства text-align.
- **Раскрывающиеся списки.** Списки, созданные с помощью тега <select>, также можно форматировать, используя стили. В Safari 2.0 это ограничивается возможностью изменить тип шрифта, цвет и размер, в то время как большинство браузеров позволяют вам устанавливать фоновый цвет, изображение и границы.

ВНИМАНИЕ

Чтобы узнать дополнительные результаты работы других браузеров при использовании CSS для формирования элементов формы, посетите страницу www.456bereastreet.com/archive/200701/styling_form_controls_with_css_revisited.

Если у вас есть свободное время, зайдите на сайт заядлого дизайнера Кристофера Шмидта (Christopher Schmitt) (www.webformelements.com). Здесь вы найдете 3520 скриншотов элементов форм в Mac OS, Windows, сделанных в различных браузерах. Наконец, вы можете бесплатно скачать главу книги CSS Cookbook Кристофера Шмидта, которая включает 164 (да, именно столько!) страницы с информацией о стилизации форм и примеры: http://oreilly.com/catalog/9780596527419/appendixd/appd.pdf.

- **Флажки и переключатели.** Большинство браузеров не позволяют применять форматирование к этим элементам. Opera тем не менее дает возможность

увидеть фоновый цвет внутри флажка или переключателя. Internet Explorer добавляет фоновый цвет вокруг этих элементов. Поскольку браузеры сильно различаются в способе представления этих элементов страницы, лучше оставить флажки в покое.

ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

Атрибут: основной селектор полей формы

Когда заходит речь об оформлении форм, стили для тегов — это не единственная изюминка. В конце концов, текстовые поля, переключатели, флажки, поля для ввода паролей и кнопки — все они используют один и тот же HTML-тег `<input>`. Хотя ширина 200 пикселов вполне подходит для текстового поля, скорее всего, вам не захочется, чтобы поле для установки флажков было бы таким же большим, поэтому вы не сможете использовать тег `<input>` для изменения ширины. Самым лучшим, учитывающим кроссбраузерность, способом форматирования исключительно текстовых полей ввода будет создание отдельных классов для каждого из них, например `<input type="text" class="textfield" name="email" />`, и создание стилевого класса для форматирования.

Однако вы можете воспользоваться преимуществами более совершенного селектора CSS — селектора атрибутов, позволяющего настроить внешний вид своей формы, не обращаясь при этом к классам. Он отбирает теги HTML на основе одного из атрибутов тега. Атрибут `type` отвечает за определение того, какой тип элемента формы создается в теге `<input>`. Значение этого типа для текстового поля ввода — `text`. Чтобы создать стиль, который будет изменять фоновый цвет всех однострочных полей ввода на синий, вам необходимо создать следующий селектор и стиль: `input[type="text"] { background-color:blue; }`.

Если вы измените значение `text` на `submit`, то создадите стиль исключительно для кнопок.

Поскольку Internet Explorer 7 и 8, Firefox, Safari, Chrome и Opera уже умеют работать с селекторами атрибутов, вы можете спокойно их использовать. Недостаток состоит в том, что пользователи Internet Explorer 6 не смогут увидеть ваши стилизованные элементы форм. Но если это не проблема (допустим, ваш шеф использует IE 8 или вы просто добавляете дизайнерские стили, которые не затронут функциональность формы), свободно экспериментируйте с этими полезными селекторами. Селекторы атрибутов могут быть использованы не только для элементов формы. Вы можете применять селектор атрибутов при создании стилей для любых тегов с каким-то определенным атрибутом. Вот пример селектора, используемого для оформления ссылок на сайт http://www.cosmofarmer.com/: `a[href="http://www.cosmofarmer.com"]`.

В CSS 3 вероятно использование даже более продвинутых селекторов атрибутов: планируется ввести возможность выбора атрибутов, значения которых начинаются с какого-либо определенного текстового значения (например, `http://`) или которые заканчиваются каким-либо определенным текстовым значением (например, `.jpg` или `.pdf`). Более подробно об этом вы прочитаете позже в книге.

Компоновка форм с помощью CSS

Все, что надо для создания формы, — добавить несколько фрагментов текстовых и других элементов на веб-страницу. Однако зачастую визуально это получается беспорядочно (рис. 10.8, *слева*). Формы обычно выглядят лучше, когда запросы и поля ввода расположены в виде столбца (см. рис. 10.8, *справа*).

Вы можете добиться этого несколькими путями. Наиболее простой способ — таблицы HTML. Хотя поля ввода и надпись — это не просто набор данных для

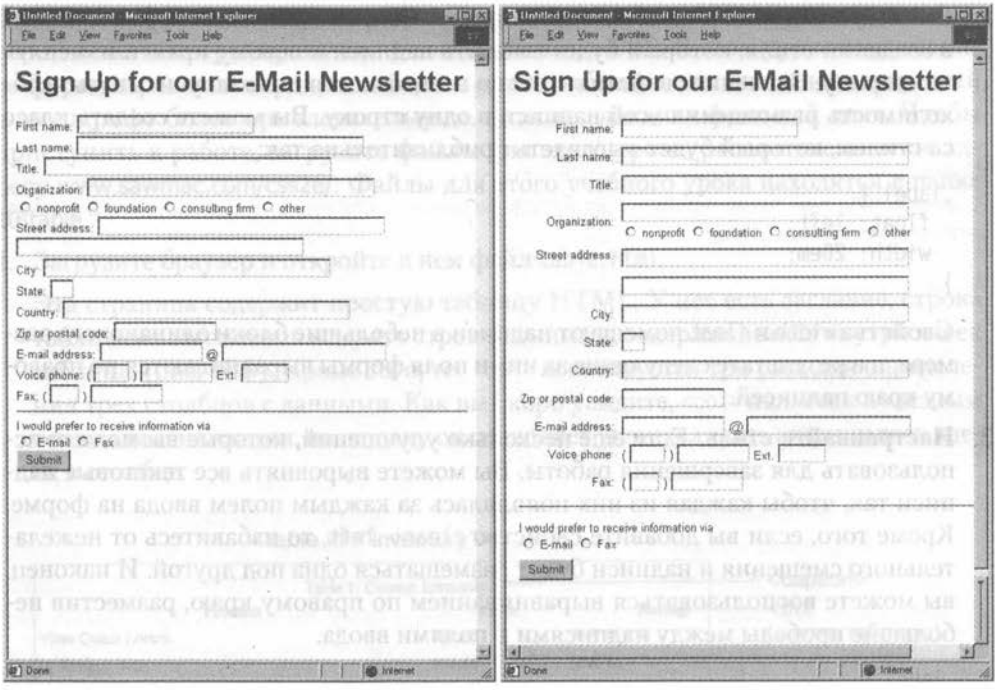


Рис. 10.8. Иногда элементы формы не очень хорошо компонуются с текстом, это приводит к тому, что они располагаются зигзагом *(слева)*. Решением данной проблемы может стать расположение ваших элементов в столбец *(справа)*

таблицы, они хорошо приспосабливаются к расположению в формате столбцов и строк. Просто расположите надписи (**Имя**, **Номер телефона** и т. д.) в одном столбце, а поля ввода формы — в другом. Используя CSS, вы также можете создать форму с двумя столбцами, как на рис. 10.8 (как преимущество — меньше кода на HTML). Рассмотрим простой вариант этого.

1. **Помещайте каждую надпись в тег.** Очевидным выбором для этого будет тег `<label>`, так как он и был разработан для надписей. Но вы не можете всегда использовать только его. Рядом с переключателями обычно располагаются вопросы типа «Какой ваш любимый цвет?», а между кнопками помещается тег `<label>`. Какой же тег вы будете использовать для создания такого вопроса? Вам следует обратиться к тегу `<span>`: `<span>What's your favorite color?</span>`. Затем добавьте по классу для каждого из этих тегов: `<span class="label">` и присоедините класс только к тем тегам `<label>`, которые хотите видеть в левом столбце (на рис. 10.8 это были бы метки для **First name**, **Last name** и т. д., но не теги `<label>` для переключателей).

ВНИМАНИЕ

Посетите страницу www.htmldog.com/guides/htmladvanced/forms/, чтобы получить быстрый обзор тега <label>.

2. **Сместите надписи и измените их ширину.** Секрет этой техники заключается в создании стиля, который будет смещать надписи к левому краю и изменять их ширину. Значение свойства `width` по возможности должно учитывать необходимость размещения всей надписи в одну строку. Вы можете создать класс со стилем, который будет выглядеть приблизительно так:

```
.label {
  float: left;
  width: 20em;
}
```

Свойства `width` и `float` помещают надписи в небольшие блоки одинакового размера, и в результате следующие за ними поля формы выравниваются по правому краю надписей.

3. **Настраивайте стиль.** Есть еще несколько улучшений, которые вы можете использовать для завершения работы. Вы можете выровнять все текстовые надписи так, чтобы каждая из них появлялась за каждым полем ввода на форме. Кроме того, если вы добавите свойство `clear: left`, то избавитесь от нежелательного смещения и надписи будут размещаться одна под другой. И наконец, вы можете воспользоваться выравниванием по правому краю, разместив небольшие пробелы между надписями и полями ввода.

```
.label {
  float: left;
  width: 20em;
  text-align: right;
  clear: left;
  margin-right; 15px;
}
```

В итоге вы получили аккуратную форму. Можете добавить и другие улучшения, выделив, например, надписи полужирным шрифтом и изменив их цвет. В обучающем уроке 2 этой главы находится пример, в котором пошагово расписаны все необходимые для этого действия.

ВНИМАНИЕ

Если вы хотите получить вдохновение и увидеть несколько интересных дизайнерских решений для форматирования веб-форм, посетите страницу www.smashingmagazine.com/2006/11/11/css-based-forms-modern-solutions, где демонстрируются всевозможные дизайны форм, основанные на использовании CSS.

Обучающий урок 1: создание стилей для таблиц

HTML отлично подходит для создания таблиц, но вам необходим также и CSS для создания стилей. Как вы можете видеть в разд. «Правильное использование таблиц» этой главы, для создания таблицы на HTML достаточно простого кода.

Скачав материалы для уроков, вы найдете уже готовую HTML-таблицу, на которой можете потренироваться в использовании CSS. В этом уроке вы измените форматирование строк, столбцов и ячеек таблицы, замените используемый в них шрифт более привлекательным, а также зададите фоновый цвет. Чтобы приступить к работе, загрузите файлы для этого урока, расположенные по адресу **www.sawmac.com/css2e/**. Файлы для этого учебного урока находятся в папке **10\table**.

1. Загрузите браузер и откройте в нем файл `table.html`.

   Эта страница содержит простую таблицу HTML. У нее есть заглавие, строка табличных заголовков и девять строк с данными, содержащимися внутри ячеек таблицы (рис. 10.9). Кроме того, тег `<col>` используется три раза для определения трех столбцов с данными. Как вы скоро увидите, `<col>` является полезным тегом для стилизации, поскольку он позволяет устанавливать ширину всех ячеек в столбце.



Рис. 10.9. Применив форматирование таблиц с использованием границ, фоновых цветов и других свойств CSS, можно не просто преобразовать скучные HTML-таблицы *(вверху)*, но и расположить данные в таблице в более читабельной форме *(внизу)*

2. Откройте файл `table.html` в текстовом редакторе.

   Для начала создадим стиль, который задает ширину таблицы и используемый шрифт. К этой таблице применим идентификатор `inventory`, так что вы можете использовать селектор ID для форматирования только этой таблицы.

ВНИМАНИЕ

С данной таблицей также поставляется несколько внешних стилей, но вы добавите новый в виде внутренней таблицы стилей.

3. Поместите курсор между открывающим и закрывающим тегами `<style>` и добавьте следующий стиль:

```
#inventory {
  font-family: "Trebuchet MS", Arial, Helvetica, sans-serif;
  width: 100%;
}
```

   Если вы не установите ширину таблицы, она растянется по размерам содержимого. В данном случае мы установим ширину, равную 100 %, чтобы таблица растянулась по всей ширине содержащего ее тега `<div>` (это область, содержащая заголовок **Welcome to the Lorem Ipsum Store** и саму таблицу). При изменении шрифта в теге `<table>` используется наследование, чтобы изменился шрифт и во всех остальных тегах внутри страницы — `<caption>`, `<th>`, `<td>` и т. д. Далее создадим стиль для заглавия таблицы.

4. Добавьте еще один стиль после только что созданного для таблицы:

```
#inventory caption {
  text-align: right;
  font-size: 1.3em;
  padding-top: 25px;
}
```

   Этот наследуемый селектор влияет только на тег `<caption>`, который появляется внутри другого тега с идентификатором `inventory` (это `<table>` на нашей странице). Тег `<caption>` сообщает о содержимом таблицы. В нашем случае он не должен быть в центре внимания, поэтому мы оставили шрифт маленьким и перенесли сам текст к правому краю. Свойство `padding-top` добавляет немного пространства над надписью, двигая ее (и таблицу) еще ниже под заголовок.

ПОДСКАЗКА

Если у вас есть текст над таблицей и вы хотите сместить его и таблицу вниз от элемента над ней, используйте только отступ для тега `<caption>`. Свойство `margin` не будет работать по одной странной причине: если вы добавите верхнее поле для тега `<table>`, Firefox вставит промежуток между текстом и таблицей правильно, в то время как остальные браузеры добавят этот промежуток над текстом. Если вы попытаетесь добавить поле вверху `<caption>`, Firefox сделает это правильно, IE проигнорирует, а Safari добавит промежуток между текстом и таблицей. Единственным надежным способом является использование верхнего отступа для тега `<caption>`.

Когда вы читаете одну из строк в таблице, очень легко сбиться. Необходим хороший визуальный ориентир. Если мы добавим границы вокруг ячеек, они визуально выделят информацию.

5. Добавьте еще один групповой стиль:

```
#inventory td, #inventory th {
  font-size: 1.4em;
  border: 1px solid #DDB575;
}
```

Селектор предназначен для форматирования тегов заголовков таблицы (<th>) и ячеек (<td>), он уменьшает размер шрифта и рисует границу вокруг каждого заголовка и каждой ячейки. Как правило, браузеры помещают пробелы между всеми ячейками, поэтому вокруг границ и появляются небольшие промежутки (рис. 10.10, *выделено кружком*). Из-за них вся таблица выглядит какой-то приземистой. Сейчас мы это исправим.

Product	Price	Rating
Vitae Quam Lorem	$19.95	★★★★★
In Tempus Velit	$14.55	★★★
Lorem Ipsum Dolor Sat	Priceless	★★★★★
Quis Felis Fringilla	$29.95	★★
Nunc Sem Pharetra	$75.99	★★★
Vel Faucibus Elit	$82.00	★
Non Adipiscing Vitae	$1.95	★★★★
Aenean Orci Ante	$17.95	★★★★
Venenatis Non Adipiscing	$44.00	★★★★★

Product	Price	Rating
Vitae Quam Lorem	$19.95	★★★★★
In Tempus Velit	$14.55	★★★
Lorem Ipsum Dolor Sat	Priceless	★★★★★
Quis Felis Fringilla	$29.95	★★
Nunc Sem Pharetra	$75.99	★★★
Vel Faucibus Elit	$82.00	★
Non Adipiscing Vitae	$1.95	★★★★
Aenean Orci Ante	$17.95	★★★★
Venenatis Non Adipiscing	$44.00	★★★★★

Рис. 10.10. Стандартное отображение таблицы в браузере с пробелами между ячейками *(вверху)*. На месте их соприкосновения граница удваивается. С помощью свойства border-collapse решаются обе эти проблемы *(внизу)*

6. Добавьте свойство border-collapse в стиль, который вы создали в шаге 3. Теперь его содержимое будет таким:

```
#inventory {
  font-family: "Trebuchet MS", Arial, Helvetica, sans-serif;
  width: 100%;
  border-collapse: collapse;
}
```

Свойство border-collapse убирает промежутки между ячейками. Оно также способствует тому, что соприкасающиеся границы ячеек сливаются в одну, что избавляет таблицу от толстых, некрасивых границ. Если не воспользоваться свойством border-collapse, то нижняя граница заголовка таблицы соединится с верхней границей ячеек таблицы, это приведет к тому, что граница удвоится и ее толщина будет равняться 2 пикселам. Если вы сейчас взглянете на таблицу, то увидите, что данные в ней выглядят намного лучше, но информация в каждой из ячеек немного сжата. Добавим небольшой отступ, чтобы избавиться от этого.

7. Добавление отступов к групповому селектору, созданному в шаге 5:

```
#inventory td, #inventory th {
  font-size: 1.4em;
  border: 1px solid #DDB575;
  padding: 3px 7px 2px 7px;
}
```

Хотя верхняя строка таблицы с заголовками и выделяется из-за использования в ней полужирного шрифта, вы можете сделать так, чтобы она выглядела еще ярче и нагляднее.

8. Создайте новый стиль после стиля td,th; он будет использоваться только для форматирования заголовка таблицы:

```
#inventory th {
  text-transform:uppercase;
  text-align: left;
  padding-top: 5px;
  padding-bottom: 4px;
}
```

Этот стиль является отличным примером использования каскадности. Групповой селектор td,th определяет общие свойства форматирования для двух типов ячеек. Введя стиль, предназначенный только для тега <th>, вы в дальнейшем сможете использовать его для изменения внешнего вида заголовков таблицы. Например, свойства padding-top и padding-bottom здесь заменяют аналогичные настройки, определенные в селекторе в шаге 7. Но, поскольку вы не переопределяли настройки левого и правого отступа, теги <th> сохранят 7 пикселов каждого из отступов, определенных в шаге 7. Этот стиль также изменяет все буквы на прописные и выравнивает текст по левому краю ячейки.

Заголовки таблицы по-прежнему не выглядят достаточно привлекательно, и, кроме того, складывается впечатление, что таблица размещена на фоне страницы. Чтобы исправить положение, можно добавить фоновое изображение.

9. Измените стиль th, добавив в него фоновое изображение и изменив цвет шрифта:

```
#inventory th {
  text-transform:uppercase;
  text-align: left;
  padding-top: 5px;
  padding-bottom: 4px;
  background: url(images/bg_th.png) no-repeat left top;
  color: #FFF;
}
```

В этом случае мы получаем едва различимую заливку сверху вниз, а верхняя и левая границы изображения отлично контрастируют с темной верхней и левой границами ячейки, в результате чего она выглядит объемной.

СОВЕТ

Кстати, вы могли просто изменить фоновый цвет, чтобы добиться такого же эффекта.

Когда в таблице хранится слишком много данных, расположенных в множестве строк и столбцов, иногда бывает трудно определить, к какой строке какие данные относятся. Одним из возможных решений такого рода проблемы, которое часто используют дизайнеры, является *чередование цвета строк*. Этого эффекта вы можете добиться, используя для каждой последующей строки таблицы класс, описанный в стиле.

10. Добавьте еще один стиль во внутреннюю таблицу стилей вашей веб-страницы:

```
#inventory tr.alt td {
  background-color: #FFF;
}
```

Этот составной наследуемый селектор указывает применить форматирование для каждого тега <td>, находящегося внутри тега <tr>, к которому применен класс alt, и только в том случае, когда они оба помещены внутри другого тега с идентификатором inventory. Сам стиль сделает фон ячеек белым, но для того чтобы он заработал, необходимо применить класс alt для каждой иной строки.

11. В HTML-коде страницы найдите тег <tr>, перед которым идет тег <td>, содержащий следующий текст: Vitae Quam Lorem. Добавьте в этот тег <tr> class="alt", и вы получите следующее:

```
<tr class="alt">
<td>Vitae Quam Lorem</td>
```

Вам придется проделать это с каждой второй строкой. Заносить каждую строку таблицы в отдельный тег может быть очень утомительно, особенно если вы

часто добавляете или удаляете их из таблицы. Чтобы попытаться автоматизировать этот процесс, воспользуйтесь небольшой программой на JavaScript, которую можно найти по адресу www.alistapart.com/articles/zebratables/.

12. Повторите шаг 11 для всех остальных тегов <tr>.

    Возможно, вы захотите проверить страницу в браузере после добавления класса к каждому <tr>, чтобы убедиться, что вы правильно размещаете все другие строки таблицы.

    Наконец, нужно отрегулировать ширину ячеек, которые попадают под столбцы **Price** и **Rating**. Один из способов заключается в дотошном добавлении названий классов к тем ячейкам и создании стилевого класса с установленной шириной. Но лучше будет воспользоваться преимуществом тега <col>, который позволяет присваивать класс или идентификатор ячейкам столбца. Как вы можете видеть на рис. 10.9, эти два столбца имеют идентификаторы price и rating. Вы можете легко установить ширину для них с помощью одного группового селектора.

13. Добавьте еще один стиль к внутренней таблице стилей страницы:

```
#price, #rating {
  width: 100px;
}
```

    Теперь ширина каждого из двух столбцов составляет 100 пикселов. Наконец таблица выглядит здорово во всех браузерах... или почти во всех. В IE 6 таблица выпадает со страницы. Когда для ширины таблицы установлено значение 100 %, IE 6 на самом деле делает таблицу немного большей, чем содержащий ее контейнер, который, в свою очередь, вынуждает таблицу теряться под левой боковой панелью. Об этой достаточно распространенной проблеме выпадания вы узнаете далее в книге. Но пока простым решением будет сделать таблицу немного меньше 100 %.

14. Отредактируйте стиль #inventory, используемый для форматирования таблицы, изменив значение его ширины на 98 %:

```
#inventory {
  font-family: "Trebuchet MS", Arial, Helvetica, sans-serif;
  width: 98%;
  border-collapse: collapse;
}
```

    Сейчас таблица всего лишь чуть тоньше, чем ее контейнер, и выглядит корректно в IE 6 (если вы не хотите, чтобы таблица была тоньше в других браузерах, можете использовать специфичный для IE 6 стиль, устанавливающий ширину 98 % только для этого браузера, — позже будет рассказано об этой методике).

Откройте страницу в браузере. Она должна выглядеть так же, как и страница, изображенная на рис. 10.9, *внизу*. Вы также можете найти готовый пример в папке **10_finished\table**.

Обучающий урок 2: создание стилей для форм

В этом уроке вы узнаете, как с помощью CSS привести форму в порядок, а также сделать ее более привлекательной. Если вы откроете в браузере страницу `form.html` из папки **10\form**, то увидите, что в ней есть простая форма для подписки на вымышленном сайте **CosmoFarmer.com** (рис. 10.11). Форма задает несколько вопросов, и в ней используются несколько элементов ввода, таких как поля, переключатели и раскрывающиеся списки. Если рассматривать ее в качестве формы для подписки, выглядит она вполне нормально, но слегка мрачновато. На следующих страницах мы приукрасим шрифты, выделим вопросы и поля ввода, а также добавим несколько других улучшений.

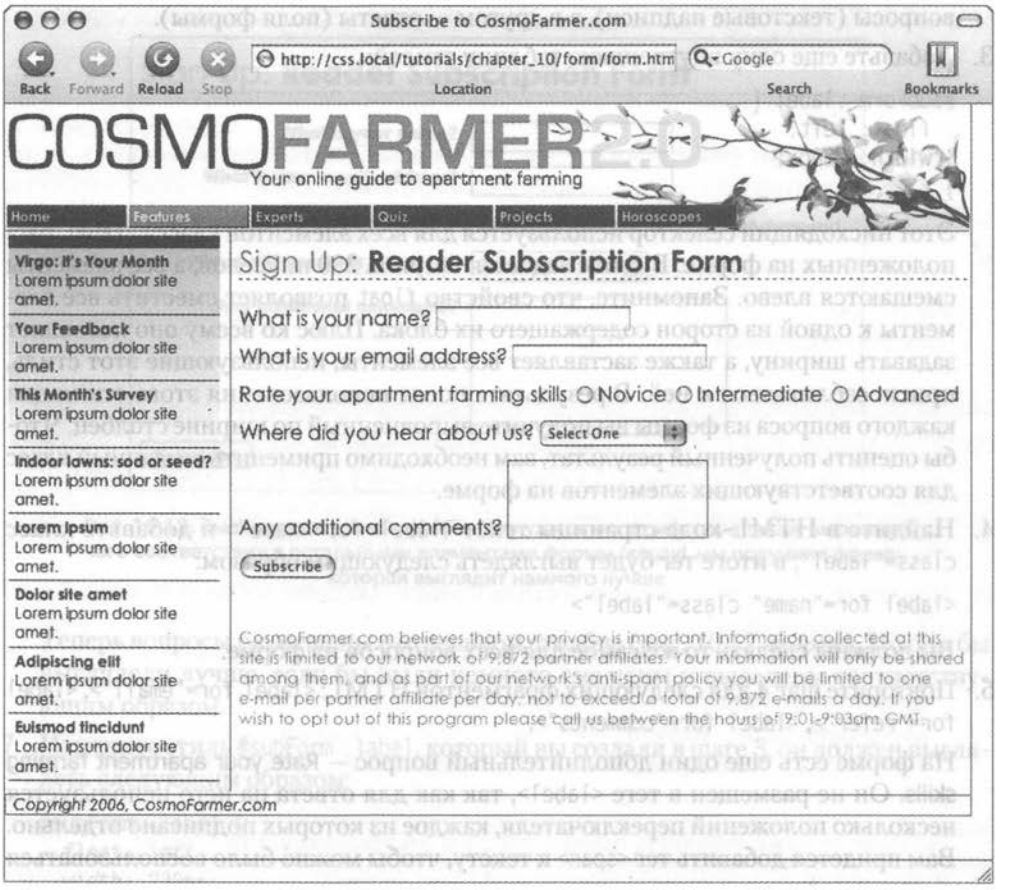


Рис. 10.11. Тег <table> обычно используется для размещения вопросов на форме. Можно также использовать CSS, чтобы создать беспорядочно перемешанный набор текстовых надписей и полей формы (как те, что вы видите на рисунке) и сделать разметку формы более понятной и привлекательной

1. Откройте файл `form.html` в текстовом редакторе.

   Для этой страницы уже создана внешняя таблица стилей в отдельном файле, но вы добавите несколько своих собственных внутренних стилей. Начнем с уменьшения размера шрифта, используемого в форме.

2. Поместите курсор между открывающим и закрывающим тегами `<style>` и добавьте следующий стиль:

   ```
   #subForm {
     font-size: .8em;
   }
   ```

   У этой формы есть ID — `subForm`, поэтому новый стиль изменяет размер шрифта для всего текста, размещенного в тегах `<form>`.

   Пришло время поработать над разметкой. Чтобы удобнее разместить элементы формы, воспользуемся двумя столбцами, в одном из которых будут находиться вопросы (текстовые надписи), а в другом — ответы (поля формы).

3. Добавьте еще одну внутреннюю таблицу стилей:

   ```
   #subForm .label {
     float: left;
     width: 230px;
   }
   ```

   Этот нисходящий селектор используется для всех элементов класса `.label`, расположенных на форме. В стиле задается ширина 230 пикселов, а все элементы смещаются влево. Запомните, что свойство `float` позволяет сместить все элементы к одной из сторон содержащего их блока. Плюс ко всему оно позволяет задавать ширину, а также заставляет все элементы, использующие этот стиль, приспосабливаться к ней. В результате после использования этого стиля для каждого вопроса из формы вы получите выровненный по ширине столбец. Чтобы оценить полученный результат, вам необходимо применить созданный класс для соответствующих элементов на форме.

4. Найдите в HTML-коде страницы текст `<label for="name">` и добавьте класс `class="label"`, в итоге тег будет выглядеть следующим образом:

   ```
   <label for="name" class="label">
   ```

   Вы должны сделать то же самое для всех вопросов на форме.

5. Повторите шаг 4 для следующих фрагментов HTML: `<label for="email">`, `<label for="refer">`, `<label for="comments">`.

   На форме есть еще один дополнительный вопрос — **Rate your apartment farming skills**. Он не размещен в теге `<label>`, так как для ответа на него используется несколько положений переключателя, каждое из которых подписано отдельно. Вам придется добавить тег `<span>` к тексту, чтобы можно было воспользоваться стилем `label`.

6. Найдите текст **Rate your apartment farming skills** и поместите его в тег `<span>`, использующий класс `label`:

   ```
   <span class="label">Rate your apartment farming skills</span>
   ```

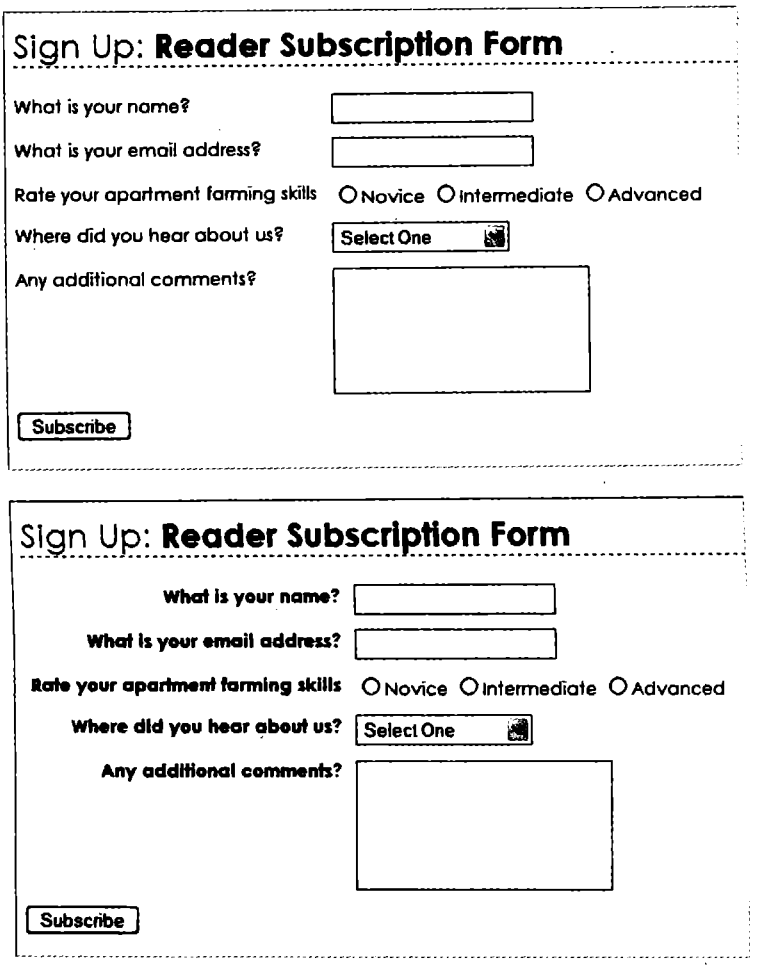


Рис. 10.12. Выделив вопросы на форме полужирным шрифтом, а также выровняв их в соответствии с остальными элементами формы *(внизу)*, мы получили форму, которая выглядит намного лучше

Теперь вопросы размещены в отдельном столбце (рис. 10.12, *вверху*). Но они бы выглядели лучше, если бы были немного смещены и выделены соответствующим образом.

7. Измените стиль `#subForm .label`, который вы создали в шаге 3, он должен выглядеть следующим образом:

```
#subForm .label {
  float: left;
  width: 230px;
  margin-right: 10px;
  text-align: right;
  font-weight: bold;
}
```

Просмотрите страницу в браузере. Форма должна выглядеть так же, как и форма, изображенная на рис. 10.12, *внизу*. Остался последний шаг. Текст вопросов смещен влево, поэтому если один из них не будет помещаться в строке, следующий вопрос будет выравниваться справа. Исправьте это, воспользовавшись свойством `clear`.

ВНИМАНИЕ

Вы можете видеть похожую проблему на рис. 7.12 и в гл. 7 получить дополнительную информацию.

8. Добавьте свойство `clear` в стиль `#subForm .label`:

```
#subForm .label {
  float: left;
  width: 230px;
  margin-right: 10px;
  text-align: right;
  font-weight: bold;
  clear: left;
}
```

Форма выглядит уже намного лучше, но кнопка **Subscribe**, размещенная у левой границы, смотрится слегка не к месту. Выровняем ее, как и остальные элементы на форме.

9. Добавьте еще один стиль во внутреннюю таблицу стилей.

```
#subscribe {
  margin-left: 240px;
}
```

У тега `<input>`, который создает кнопку **Subscribe**, уже есть ID, равный `subscribe`, поэтому этим стилем мы изменяем размер кнопки до 240 пикселов и выравниваем ее в соответствии с правым краем стиля `#subForm .label`. Многие браузеры предоставляют также дополнительные возможности по оформлению кнопок, так что дерзайте.

10. Измените только что созданный стиль кнопки **Subscribe**, добавив фоновый цвет и задав другой шрифт:

```
#subscribe {
  margin-left: 240px;
  background-color: #CBD893;
  font-family: "Century Gothic", "Gill Sans", Arial, sans-serif;
}
```

Вы можете даже изменить шрифт текста в раскрывающемся списке.

11. Добавьте стиль для размещенного на форме списка:

```
#refer {
  font-family: "Century Gothic", "Gill Sans", Arial, sans-serif;
}
```

В итоге текст вопросов и кнопка **Subscribe** выглядят великолепно, но зачем нам на этом останавливаться? Пришло время слегка приукрасить и поля ввода на форме. Начнем с изменения шрифтов и фоновых цветов.

12. Создайте новый групповой селектор для трех полей ввода на форме:

```
#name, #email, #comments {
  background-color: #FBEF99;
  font-family: "Lucida Console", Monaco, monospace;
  font-size: .9em;
}
```

После использования этого группового стиля фон полей ввода для текста (для которых применен собственный идентификатор) станет светло-желтым, а также изменится размер и тип шрифта, который будут использовать посетители при вводе текста. Поля ввода выглядят слегка тесными, а также кажется, что они размещаются немного ниже соответствующих им текстовых обозначений. Исправить это посредством CSS проще простого.

13. Исправьте только что созданный стиль, изменив ширину и выравнивание по верхнему краю:

```
#name, #email, #comments {
  background-color: #FBEF99;
  font-family: "Lucida Console", Monaco, monospace;
  font-size: .9em;
  width: 300px;
  margin-top: -2px;
}
```

Вы можете сделать форму более удобной для пользователей, выделив активные элементы с помощью специального псевдокласса `:focus` (см. разд. «Выборка стилизуемых ссылок» гл. 9). Мы добавим его в следующем пункте.

14. В конце внутренней таблицы стилей добавьте стиль для раскрывающегося списка и трех полей ввода для текста:

```
#name:focus,
#email:focus,
#comments:focus,
#refer:focus
{
  background-color: #FDD041;
}
```

Псевдокласс `:focus` на момент написания этой книги работает только в браузерах Internet Explorer 8, Firefox, Safari и Opera. Однако он не вызывает никаких ошибок, поэтому, если вы будете использовать браузер Internet Explorer 6 или 7, можете смело добавить стиль `:focus` в качестве улучшения внешнего вида.

Просмотрите страницу в браузере. Сейчас она должна будет выглядеть так, как на рис. 10.13.

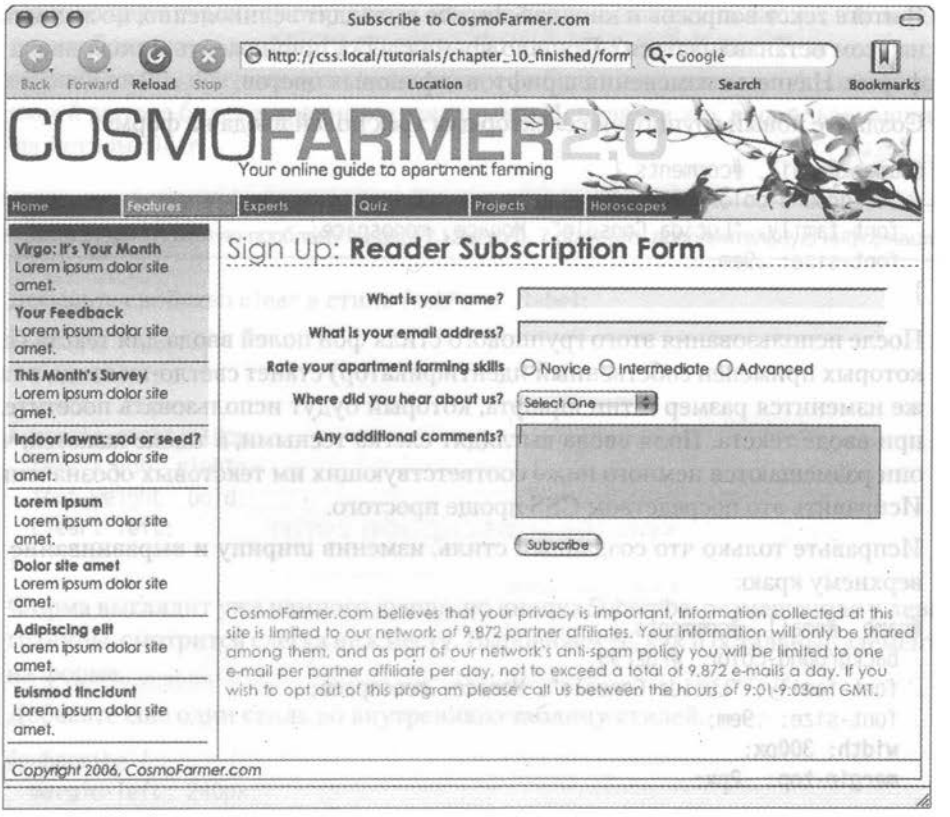


Рис. 10.13. Используя псевдокласс :focus, вы можете сделать форму более интерактивной: будет подсвечиваться активное поле ввода. Здесь мы собираемся добавить текст в поле с комментариями, и его фоновый цвет выглядит темнее

Готовую версию созданной в этом уроке формы вы можете найти в папке 10_finished\form.

ЧАСТЬ 3

Макет страницы

Глава 11. Введение в разметку CSS

Глава 12. Разметка страницы на основе плавающих элементов

Глава 13. Позиционирование элементов на веб-странице

Глава 14. CSS для распечатываемых веб-страниц

Глава 15. Совершенствуем навыки в CSS

Глава 16. CSS 3 — на гребне волны

11 Введение в разметку CSS

CSS удобен при форматировании текста, навигационных панелей, изображений и других элементов веб-страницы, но его по-настоящему потрясающие способности проявляются, когда нужно создать макет всей веб-страницы.

В то время как HTML обычно отображает содержимое страницы на экране сверху вниз, размещая один блочный элемент за другим, CSS позволяет вам создавать расположенные бок о бок столбцы и помещать изображения или текст в любом месте на странице (и даже наслаивать поверх других элементов), поэтому веб-страницы имеют намного более интересный внешний вид.

Разметка CSS — это достаточно обширная тема. Так, в следующих двух главах вы подробно узнаете о двух наиболее важных техниках CSS. А в этой главе я предлагаю краткий обзор принципов, на которых построена разметка, и немного полезных инструкций для решения возникающих в процессе работы проблем.

Типы разметок веб-страницы

Быть веб-дизайнером означает иметь дело с неизвестным. Какими браузерами пользуются ваши посетители? Установлена ли у них последняя версия плагина Flash Player? Возможно, самая большая проблема, с которой сталкиваются разработчики, состоит в создании привлекательных дизайнов, учитывающих различные размеры экрана. Мониторы различаются размерами и разрешением: от маленьких 15-дюймовых с разрешением 640 × 480 пикселов до чудовищных 30-дюймовых экранов с разрешением примерно 5 000 000 × 4 300 000 пикселов.

Веб-разметка предполагает три основных подхода к решению упомянутой проблемы. Оформление практически каждой веб-страницы сводится к использованию одного из двух вариантов: с *фиксированной* либо с *непостоянной* шириной. Фиксированная ширина дает вам наибольший контроль над разметкой, но может доставить неудобства некоторым посетителям. Людям с действительно маленькими мониторами придется прокручивать страницу вправо, чтобы все увидеть, а те, у кого большие мониторы, будут видеть пустоту в части экрана, где могла полностью отобразиться ваша превосходная страница. Использование непостоянной ширины (которая увеличивается или уменьшается, чтобы соответствовать окнам браузера) делает управление дизайном страницы серьезным испытанием, но при

этом эффективнее используется окно браузера. *Гибкий дизайн* сочетает некоторые преимущества обоих типов.

- **Фиксированная ширина.** Многие дизайнеры предпочитают плотно распределять страницу по ширине, как на странице на рис. 11.1, *вверху*. Независимо от ширины окна браузера, ширина содержимого страницы не изменяется. В некоторых случаях страница «цепляется» за левый край окна браузера или, что бывает чаще, сосредотачивается посередине. С дизайном, основанным на фиксированной ширине, вам не нужно волноваться по поводу того, что случится с вашей страницей на очень широком (или маленьком) экране.

  Многие страницы с фиксированной шириной придерживаются ширины экрана менее 1000 пикселов, позволяя окну, полосам прокрутки и другим частям браузера подходить по размерам для экранов с разрешением 1024 × 768. Очень распространенной является ширина 960 пикселов. Большинство сайтов имеют именно фиксированную ширину.

ПРИМЕЧАНИЕ

Чтобы увидеть примеры дизайнов с фиксированной шириной, нацеленных на большие мониторы, зайдите на сайты www.alistapart.com, www.espn.com или www.nytimes.com.

- **Непостоянная ширина.** Иногда легче плыть по течению вместо того, чтобы бороться с ним. Свободный дизайн приспосабливается так, чтобы соответствовать любой ширине браузера. Ваша страница становится шире либо уже, когда посетитель изменяет размеры окна (см. рис. 11.1, *посередине*). Хоть этот тип дизайна наилучшим образом использует доступное пространство окна браузера, вам придется приложить больше усилий, чтобы убедиться в том, что страница хорошо выглядит при различных размерах окна. На очень больших мониторах такой тип дизайна может смотреться слишком размашисто, с длинными строками текста, неподходящими для их нормального прочтения.

ПРИМЕЧАНИЕ

Чтобы увидеть примеры дизайнов с непостоянной шириной, зайдите на сайт http://maps.google.com.

- **Гибкий дизайн.** Это на самом деле дизайн, использующий фиксированную ширину с особенной гибкостью размеров. Работая с этим видом дизайна, вы определяете ширину страницы, используя em-значения. У единицы измерения em изменяется размер, когда меняется размер шрифта браузера, так что ширина дизайна в конце концов основана на размере базового шрифта браузера (см. подраздел «Ключевые слова, проценты и единица измерения em» разд. «Установка размера шрифта» гл. 6). При изменении размера шрифта браузера ширина страницы и элементов внутри ее также изменяется — это напоминает масштабирование дизайна.

  Гибкие дизайны утрачивают свои позиции в наше время, поскольку современные версии самых популярных браузеров заменили привычную команду увеличения размера текста командой масштабирования страницы. Например, в Internet Explorer 8, Firefox 3, Safari 4, Opera и Chrome, нажатие Ctrl-+ увеличивает все на странице (тот же результат вы получаете с использованием гибкого дизайна).

ПРИМЕЧАНИЕ

Свойства max-width и min-width предлагают компромисс между фиксированной и свободной шириной (см. врезку «Информация для опытных пользователей» далее).

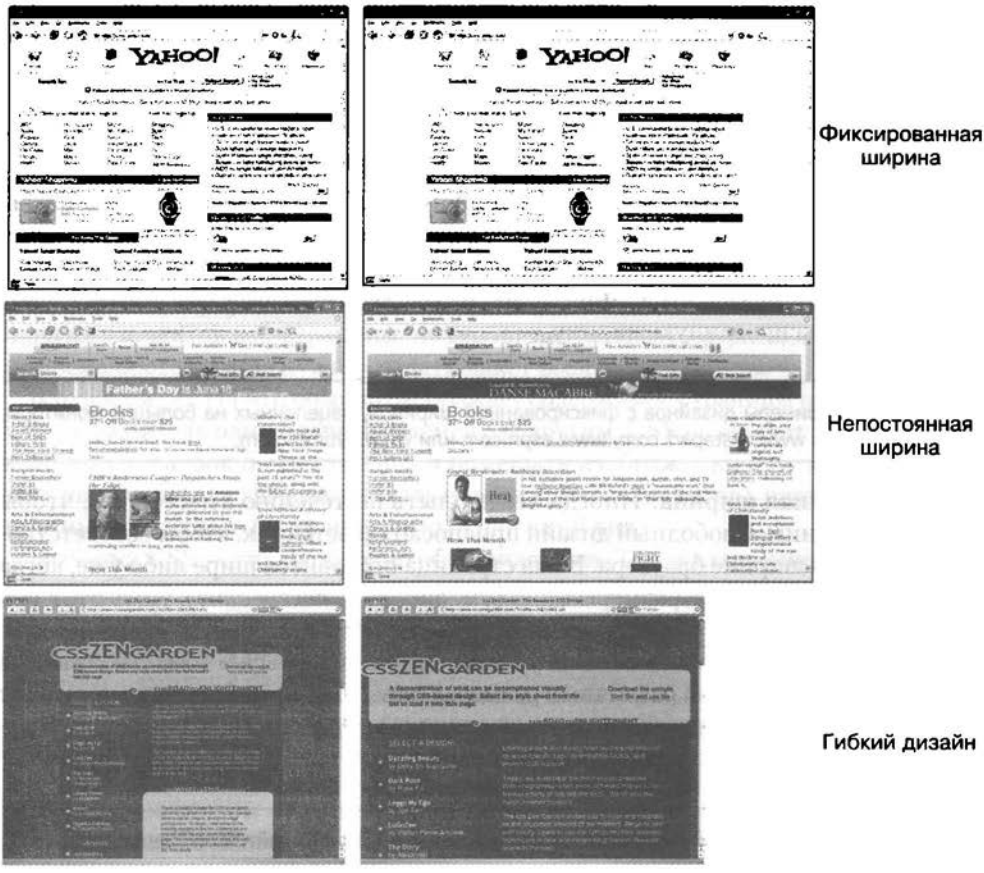


Рис. 11.1. Три типа оформления страницы

Итак, у вас есть несколько способов работы с неопределенными размерами окон и шрифта браузера. Вы можете просто проигнорировать тот факт, что у ваших посетителей мониторы с различными разрешениями, и насильно установить единственную фиксированную ширину для вашей страницы либо создать свободный дизайн, который заполнит окно любой ширины, с которой столкнется. Гибкий дизайн изменяет ширину, только если размер шрифта (а не размер окна) изменяется.

В обучающих уроках в конце следующей главы вы создадите страницы с фиксированной и свободной шириной.

Как работает CSS-разметка

Как говорилось в гл. 1, на заре развития Интернета ограничения HTML вынудили дизайнеров придумывать новые способы оформления сайтов. Самым распростра-

ненным инструментом был тег <table>, который, как изначально предполагалось, должен был служить для отображения информации в виде электронной таблицы, составленной из строк и столбцов с данными. Разработчики использовали HTML-таблицы, чтобы создать своего рода основу для организации содержимого страницы (рис. 11.2). Однако, поскольку тег <table> не был предназначен для разметки, дизайнерам часто приходилось управлять им непривычными способами (например, помещая таблицу внутри ячейки другой таблицы), чтобы получить нужный результат. Этот метод забирал много времени, добавлял кусок лишнего HTML-кода и усложнял дальнейшее изменение дизайна. Но это все, что было у дизайнеров до CSS.

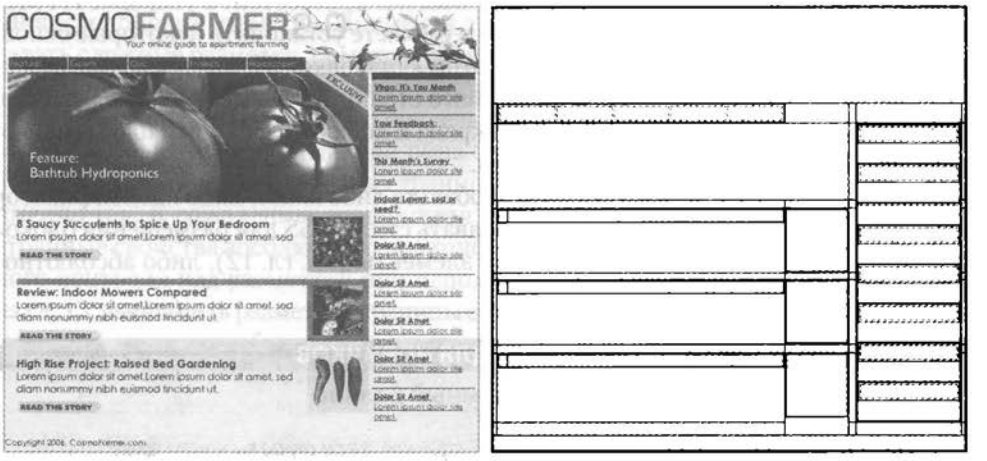


Рис. 11.2. Формирование составного дизайна страницы *(слева)* с использованием тега <table> подобно созданию закрепленных подмосток *(справа)*, а изменение дизайна требует демонтажа этих подмосток и построения новых

Если вы долгое время работали с тегом <table>, вам нужно постепенно переходить на новый уровень, начиная использовать для разметки CSS. Сразу забудьте о строках и столбцах (важных понятиях при работе с таблицами). Создавая табличную разметку, вы вставляли код HTML в отдельные ячейки таблицы. Каждая из них выступала в роли контейнера на странице, содержащего заголовок, изображение, текст или комбинацию разных типов информации.

В CSS есть эквивалент ячейке таблицы — тег . Он также является контейнером для содержимого, которое вы хотите разместить в определенном месте на странице. Кроме того, как вы видите, в разработках на CSS часто один тег вкладывается в другой, что очень похоже на вложение одних таблиц внутрь других для получения определенных результатов. Однако, к счастью, CSS-метод использует намного меньше HTML-кода.

Мощный тег

Разметка приводит к расположению содержимого в различных областях страницы. В CSS для организации содержимого обычно применяется тег . Как вы читали в разд. «Написание HTML-кода для CSS» гл. 1, тег — это HTML-элемент,

у которого нет никаких собственных свойств форматирования (помимо того факта, что браузеры рассматривают этот тег как блок с обрывом строки до и после него). Вместо этого он используется для обозначения логического группирования элементов, или *распределения* на странице.

Вы обычно указываете тег с двух сторон кусков HTML-кода, подходящих друг к другу. Такой элемент, как навигационная панель, обычно занимает вершину страницы, так что есть смысл задавать тег и вокруг него. Кроме того, придется определить тег для всех основных областей вашей страницы, таких как баннер, область с главным содержимым, боковое меню, нижний колонтитул и т. д. Есть возможность указать тег вокруг одного или нескольких дополнительных отделений. Одна из распространенных методик состоит в упаковке HTML-кода внутри тега <body> в тег . Тогда вы сможете установить некоторые основные свойства страницы, применяя CSS к этой «упаковке». Появится возможность установить полную ширину для содержимого страницы, левые и правые края или разместить все содержимое посередине экрана и добавить фоновый цвет или изображение для главного столбца страницы.

Как только вы расставите теги , добавьте либо класс, либо идентификатор (ID) к каждому из них, и вы сможете создавать стили CSS для размещения блоков на странице, используя либо плавающие элементы (см. гл. 12), либо абсолютное позиционирование (см. гл. 13).

ИНФОРМАЦИЯ ДЛЯ НОВИЧКОВ

Находим разумный баланс

Хоть разделы `div` крайне необходимы CSS-разметкам, не бросайтесь активно применять их на вашей странице. Распространенное заблуждение состоит в том, что вы должны указывать теги для всех элементов страницы. Скажем, ваша главная навигационная панель — неупорядоченный список ссылок (как, например, та, что описана в разд. «Создание панелей навигации» гл. 9). Поскольку это важный элемент, вы, возможно, пожелаете указать вокруг него: `<div id="mainNav"><ul>...</ul></div>`.

Однако нет никаких причин добавлять тег , когда <ul> настолько же удобен. Пока тег <ul> содержит ссылки главной навигационной панели, вы можете просто добавить ваш стиль ID к этому тегу: `<ul id = "mainNav">`.

Кроме того, нет необходимости использовать , когда под рукой есть другой, более подходящий элемент HTML. Например, вы захотели процитировать кусок текста — создать блок, выровненный по правому краю, содержащий важную цитату из содержимого страницы. В этом случае вы можете пропустить и просто использовать тег <blockquote>, позиционируя его с помощью свойства `float`, которое будет рассмотрено в следующей главе.

Но это не значит, что нужно бояться использовать теги . Добавив их на парочку больше, вы не увеличите размер файла и не замедлите загрузку страницы. Если поможет вам решить проблему и ни один из других тегов не подходит, естественно, используйте . Кроме того, — это единственный способ сгруппировать в единое целое несколько различных тегов HTML. На самом деле не так редко можно увидеть один тег , окруженный несколькими другими.

Основное практическое правило при работе с HTML — стараться свести количество кода к минимуму, но при этом использовать его столько, сколько нужно. Если добавление пары лишних тегов имеет смысл для проектирования дизайна, то смело используйте их.

Методики разметки CSS

На текущей стадии развития CSS разметка бывает двух видов — с абсолютным и относительным позиционированием. Большинство веб-страниц использует для разметки CSS-свойство `float`. Вы уже сталкивались с этим, казалось бы, простым свойством в гл. 8, где оно было представлено как способ позиционирования изображения внутри столбца с текстом перемещением его либо в левую, либо в правую сторону. Тот же принцип применяется к тегам `<div>`: устанавливая для них ширину и перемещая влево или вправо, вы можете создать столбец (текст после тега `<div>` «обертывается» вокруг плавающего `<div>`, как если бы он был еще одним столбцом). Используя свойство `float` с множеством тегов `<div>`, вы получаете возможность создавать многостолбцовые разметки. Применяя эту методику дальше, вы будете быстро создавать сложные разметки, помещая одни плавающие теги `<div>` внутри других.

Абсолютное позиционирование позволяет поместить элемент в любом месте страницы с точностью до одного пиксела. Вы можете поместить элемент, например, в 100 пикселах от верхнего края окна браузера и в 15 пикселах от левого края. Такие программы для разметки страницы, как InDesign и Quark Xpress, работают именно так. К сожалению, изменчивая природа веб-страниц, а также некоторые странные свойства абсолютного позиционирования затрудняют возможность полного контроля над разметкой при использовании этой методики. Как вы узнаете из гл. 13, выполнить разметку страницы с помощью абсолютного позиционирования можно, но этот способ лучше подходит для небольших задач вроде позиционирования логотипа в конкретном месте на странице.

Не беспокойтесь, если сейчас все это звучит довольно сложно для вас. Вы увидите все эти методы в действии в следующих двух главах.

ПРИМЕЧАНИЕ

Есть несколько других способов выполнить разметку CSS, но они ограничены Internet Explorer 8, Firefox, Safari и другими современными браузерами. Иначе говоря, эти альтернативные методы не будут работать для подавляющего большинства веб-серферов, пользующихся IE 6 и 7. Тем не менее мы рассмотрим эти способы в гл. 16.

Стратегии разметки

Разметка веб-страницы с помощью CSS — это скорее искусство, чем наука. Поэтому нет четкой формулы, которой нужно придерживаться для разметки содержимого с HTML и создания CSS. То, что работает с одним дизайном, может не подойти для другого.

Изучение разметки CSS происходит на личном опыте. Нужно узнавать, как работают различные свойства CSS (особенно абсолютное и относительное позиционирование), читать о методах разметки, следовать обучающим примерам, таким как представленные в следующих двух главах, и много-много практиковаться.

Тем не менее определенно есть некоторые стратегии, которые можно принять, приступая к созданию разметки. Они больше похожи на установки или инструкции, а не на жесткие правила. Но, как только вы начнете проектировать дизайн для каких-либо проектов, имейте в виду эти инструкции.

Начнем с содержимого

Многие дизайнеры хотели бы сразу перейти непосредственно к своей теме — цветам, шрифтам, значкам и изображениям. Но, начиная с визуального дизайна, вы ставите телегу впереди лошади.

Самые важные элементы веб-страницы — это ее содержимое (заголовки, абзацы текста, потрясающие фотографии, навигационные ссылки, флэш-ролики, а также все то, ради чего люди придут к вам на сайт). Посетители захотят выяснить для себя, что ваш сайт может предложить им. Содержимое стоит во главе угла, и вам сначала нужно подумать, что вы хотите сказать, прежде чем решить, как это должно выглядеть. В конце концов, создавая фантастическую по красоте трехмерную боковую панель, вы ничего не добьетесь, если вам особо нечего вводить туда.

Кроме того, идея страницы должна диктовать ее дизайн. Если вы решите, что на домашней странице нужно продавать услуги вашей компании, и захотите выделить отличное предложение для клиентов, то вполне вероятно, что особое значение будет иметь большая фотография с приветливыми сотрудниками, а также и цитата из отзыва одного из довольных клиентов. Поскольку оба этих элемента важны для страницы, вы должны сделать их видными и неотразимыми.

Импровизируйте с дизайном

Даже если вы чувствуете себя более комфортно, создавая код HTML и CSS вручную в любимом текстовом редакторе, нежели рисуя в графическом редакторе, не стоит начинать с кода. Такие программы, как Photoshop, Illustrator или Fireworks, очень просто использовать для создания графических дизайнов. Они дают свободу для импровизации с различными цветами, шрифтами, изображениями, позиционированием, при этом не заставляя тратить время на написание кода HTML и CSS. Вы можете экспериментировать с новыми дизайнерскими идеями и задумками, пока не получите то, что вам нравится.

Если вы новичок в работе с графическими программами, то даже простое рисование прямоугольников для обозначения расположения различных элементов страницы может улучшить ваше представление о том, какой должна быть разметка страницы. Изменить дизайн с двумя столбцами на дизайн с четырьмя намного проще, меняя размер прямоугольников в программе Illustrator, чем переписывая код. Даже карандаш и бумага — это отличный способ для того, чтобы почувствовать, где должно быть расположено содержимое на странице, каким оно должно быть по размерам и каким должен стать общий тон (светлым или темным).

СОВЕТ

Компания Yahoo предлагает бесплатный набор трафаретов (Stencil Kit) (http://developer.yahoo.com/ypatterns/wireframes), которые можно использовать в Illustrator, Visio, OmniGraffle и других графических программах для создания макетов веб-страниц. Предоставляемые элементы пользовательского интерфейса, например кнопки, поля, окна и ссылки, могут превратить создание разметки в простое перетаскивание значков.

Идентифицируйте разделы

После того как вы создали визуальный макет, нужно подумать о том, как создать разметку HTML и CSS для достижения дизайнерских задумок. Этот процесс обычно включает представление различных структурных блоков страницы и идентификацию элементов, которые выглядят как отдельные разделы. Например, на рис. 11.3 есть немало элементов, которые выглядят как небольшие разделы: наиболее крупным является раздел с тремя объявлениями внизу (помечены буквой A на рис. 11.3). Каждый раздел, как правило, является кандидатом на выделение в отдельный тег `<div>` (если только нет более подходящего тега HTML, что мы обсуждали ранее).

Часто визуальная подсказка в макете может помочь решить, нужен ли вообще тег `<div>`. Например, линия границы, обведенная вокруг заголовка и нескольких абзацев, означает, что вам понадобится обернуть эту группу HTML-тегов тегом `<div>`, к которому применена граница.

Кроме того, когда вы видите фрагменты текста, идущие бок о бок (например, три фрагмента с содержимым в нижней части рис. 11.3), вы знаете, что нужно каждый из них заключить в отдельный тег `<div>`. HTML-теги, как правило, сами не располагаются бок о бок, поэтому вам придется использовать некоторые техники разметки (например, плавающие теги, описанные в следующей главе).

Желательно также группировать теги `<div>`, расположенные рядом, в один общий тег `<div>`. Например, в нижней части рис. 11.3 вы видите набор тегов `<div>`, которые обеспечивают структуру страницы. Разделы `news` и `footer` являются контейнерами для своих собственных наборов `<div>`. Хоть это и не всегда необходимо, такой подход помогает обеспечивать гибкость. Например, вы можете уменьшить основной раздел (изображение руки и подзаголовок) по ширине и передвинуть раздел с новостями вправо от него, который сформировал бы отдельный столбец. Новости могли бы идти друг под другом, а не бок о бок.

Плывите по течению

Теги обычно не располагаются рядом и не наслаиваются друг на друга. Как правило, они действуют подобно тексту в программах для его обработки: заполняют всю ширину страницы и растекаются от верхнего до нижнего края. Каждый тег уровня блока — заголовок, абзац, маркированный список и т. д. — располагается наверху следующего тега уровня блока. Поскольку это стандартный подход, вам обычно не нужно ничего делать, если вы планировали вводить теги `<div>` один за другим.

Например, на рис. 11.3 четыре тега `<div>` — `banner`, `main`, `news` и `footer` — охватывают всю ширину своего контейнера (тега `<body>`) и расположены друг за другом по вертикали. Это типичная ситуация для тегов уровня блока, и вам не нужно делать ничего особенного, применяя CSS, чтобы расположить эти четыре `<div>` таким образом.

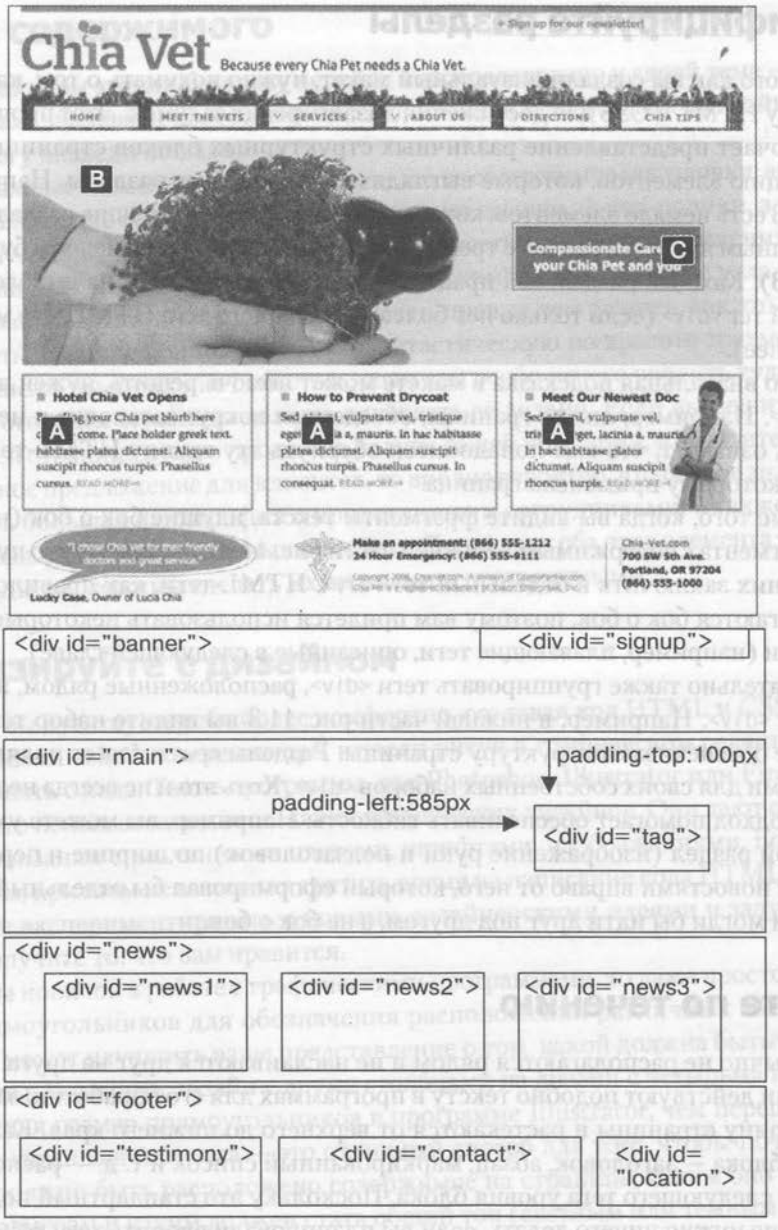


Рис. 11.3. Корректное расположение HTML-элементов, необходимое для преобразования макета Photoshop в код HTML и CSS, включает в себя видение структуры страницы и «упаковку» связанных групп HTML-элементов в теги

Помните о фоновых изображениях

Вы, несомненно, видели черепичные изображения, заполняющие фон веб-страницы, или градиенты, добавляющие эффект глубины баннеру. Свойство `background-image`

предоставляет еще один способ добавить фотографии на страницу, не прибегая к помощи тега <img>. Вставка изображения в фон существующего тега не только позволяет сохранить пару байтов данных, которые были бы потрачены на тег <img>, но и упрощает решение некоторых проблем, связанных с разметкой.

Например, на рис. 11.3 изображение руки по центру *(B)* на самом деле является обычным фоновым изображением. Это облегчает размещение другого тега с подзаголовком «Compassionate care...» *(C)*, поскольку он просто расположен поверх фона родительского тега. Кроме того, фотография с доктором чуть ниже, в правой части страницы, — обычное фоновое изображение, размещенное в текущем теге . Добавление небольшого отступа справа отодвинет текст в этом разделе от фотографии.

ПРИМЕЧАНИЕ

Есть минусы использования фотографий в качестве фонов тегов (или других тегов HTML). Во-первых, браузеры обычно не печатают фон, так что если у вас есть страница с картой, содержащей какие-то важные маршруты, вставьте ее с помощью тега <img>, а не в качестве фонового изображения. Кроме того, поисковые системы обходят стороной CSS, поэтому, если вы думаете, что изображение может привлечь дополнительный трафик на ваш сайт, используйте тег <img> и добавьте описательный атрибут alt.

Куски головоломки

Этот совет можно было давать в разделе «креативного решения проблем». Часто то, что выглядит как одно целое, на самом деле состоит из нескольких частей. Например, в обучающем примере гл. 8 боковая панель, похожая на бумажный свиток, была фактически сооружена из трех фоновых изображений в трех отдельных тегах HTML (см. рис. 8.17). Кроме того, трюк с «раздвижными дверями», обсуждавшийся ранее, использует подобную методику.

Вы можете увидеть простой пример этого в нижней части рис. 11.3. Здесь есть четыре расположенных друг за другом тега , каждый из которых имеет белый фон. Если у вас возникли проблемы с размещением больших элементов на странице (очень большого рисунка, занимающего много столбцов, или массивного фонового изображения, которое займет не одну область страницы), подумайте о том, можно ли добиться желаемого внешнего вида страницы, разбив большой элемент на мелкие куски, которые потом сложатся, как части головоломки.

Расслоение элементов

Если вы работали с Photoshop, Illustrator или Fireworks, то, вероятно, использовали концепцию слоев. Слои позволяют создавать отдельные полотна, которые накладываются друг на друга и образуют единое изображение. В этих программах можно легко сделать логотип поверх заголовка с текстом или разместить одну фотографию над другой. Если вы хотите сделать то же самое на веб-странице, у вас есть несколько вариантов.

Часто самым простым способом наложить слой поверх фотографии является добавление изображения в фон другого тега (см. совет выше). Поскольку фоновое

изображение идет за тегом, все, что находится внутри этого тега — текст, другие изображения, — будет расположено поверх фотографии.

Но что делать, если вы хотите наложить фотографию поверх какого-нибудь текста? В таком случае нужно обратиться к единственному свойству CSS, позволяющему наслаивать элементы, — свойству `position`. Мы рассмотрим его в гл. 13, поскольку размещение элементов поверх чего-либо требует абсолютного позиционирования.

Не забывайте о полях и отступах

Иногда самое простое решение оказывается лучшим. Вам не всегда нужен причудливый CSS-код, чтобы поместить элемент в нужное место на странице. Вспомните, что отступы и поля представляют собой обычное пустое пространство и, используя их, вы можете двигать элементы по странице. Например, подзаголовок (*C* на рис. 11.3) размещается простой установкой верхнего и левого отступа для его родительского тега. Как вы можете видеть на схеме в нижней половине рис. 11.3, подзаголовок помещен внутри другой тега `<div>` (`<div id="main">`). Этот тег на самом деле не имеет другого содержимого, кроме подзаголовка, — фотография является фоновым изображением, так что добавление отступа перемещает тег `<div>` подзаголовка вправо вниз.

12 Разметка страницы на основе плавающих элементов

Разметка, основанная на плавающих элементах, использует преимущества свойства `float` для расположения элементов бок о бок, создавая на веб-странице столбцы. Как было описано в разд. «Управление обтеканием содержимого плавающих элементов» гл. 7, вы можете использовать это свойство для создания эффекта обтекания, скажем, для фотографии, но, когда вы применяете его к тегу `<div>`, оно становится мощным инструментом разметки страницы. Свойство `float` перемещает элемент в одну сторону страницы (или другого блока с содержимым). Любой HTML-объект, который появляется ниже плавающего элемента, изменяет свое положение на странице и обтекает его.

Свойство `float` принимает одно из трех значений: `left`, `right` или `none`. Чтобы переместить изображение к правому краю страницы, вы можете создать класс стиля и применить его к тегу `<img>`:

```
.floatRight { float: right; }
```

То же самое свойство, примененное к тегу `<div>` с содержимым, может также создать боковое меню:

```
#sidebar {
  float: left;
  width: 170px;
}
```

ПРИМЕЧАНИЕ

Значение none отменяет любое перемещение и определяет элемент как обычный (не плавающий). Это полезно только для отмены перемещения, которое уже относится к элементу. Допустим, у вас есть элемент, к которому применен специфический класс, такой как "sidebar", и этот элемент смещен вправо. На одной из страниц вы, возможно, захотите, чтобы элемент с этим классом не передвигался, а был определен в общем потоке страницы. Создавая более специфичный CSS-селектор (см. разд. «Управление каскадностью» гл. 5) с float: none, вы можете предохранить этот элемент от перемещения.

На рис. 12.1 показаны эти два стиля в действии. Здесь блок новостей перемещен к левому краю. У него есть фиксированная ширина, однако у главного содержимого ее нет, что делает этот дизайн свободным. Главный раздел страницы просто расширяется, заполняя окно браузера. Вверху справа фотография с ванной перемещена в правую сторону.



Рис. 12.1. Используйте свойство float для разметки веб-страницы с множеством столбцов

Простой дизайн с двумя столбцами, как на рис. 12.1, требует выполнения всего нескольких действий.

1. **Укажите вокруг каждого столбца тег с атрибутом класса или ID.** На рис. 12.1 заголовки новостей, перечисленные в левой части, заключены в один раздел (`<div id = "news">`), а главное содержимое страницы — в другой (`<div id = "main">`).
2. **Переместите тег с боковым меню вправо или влево.** Когда вы работаете с плавающими элементами, важен порядок исходного кода (порядок, в котором вы добавляете HTML к файлу). HTML для плавающего элемента должен появиться *перед* HTML для элемента, который указывается вокруг него.

   На рис. 12.2 показаны три варианта разметки с двумя столбцами. Диаграммы на левой стороне показывают порядок HTML-кода страницы: тег `<div>` для баннера, за которым следует `<div>` для бокового меню, и, наконец, тег `<div>` для главного содержимого. Справа вы видите фактическую разметку страницы. Боковое меню идет *перед* главным содержимым в HTML, так что оно может переместиться или влево (*вверху* и *внизу*), или вправо (*посередине*).

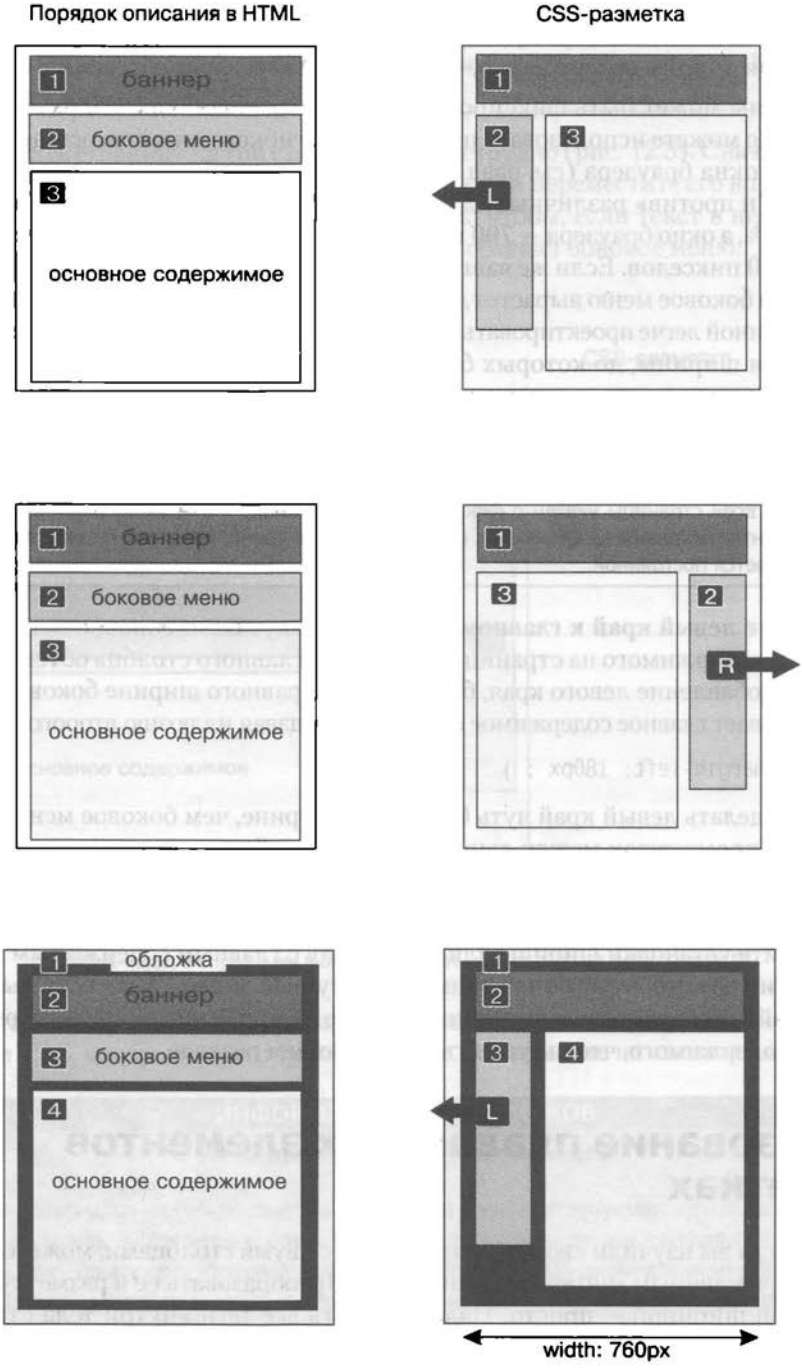


Рис. 12.2. Создание разметки с двумя столбцами — лишь вопрос перемещения тега влево *(вверху)*. Чтобы заставить боковое меню переместиться в правую сторону страницы *(посередине)*, просто измените CSS-стиль бокового меню на float: right

3. **Установите ширину для плавающего бокового меню.** Всегда задавайте плавающим элементам ширину. Таким образом вы позволите браузеру создать место для другого содержимого, чтобы получить эффект обтекания.

   У ширины может быть фиксированный размер, такой как 170 px или 10 em. Вы также можете использовать проценты для гибкого дизайна, основанного на ширине окна браузера (см. разд. «Типы разметок веб-страницы» в гл. 11 обо всех «за и против» различных единиц измерения). Если у бокового меню ширина 20 %, а окно браузера — 700 пикселов в ширину, то ширина бокового меню будет 140 пикселов. Если же ваш посетитель изменит размер окна до 1000 пикселов, то боковое меню вырастет до 200 пикселов. Боковые меню с фиксированной шириной легче проектировать, так как не нужно рассматривать все различные значения ширины, до которых боковое меню могло бы растянуться. Однако проценты позволяют вам поддерживать одинаковые пропорции между двумя столбцами, что может быть визуально привлекательнее.

ПРИМЕЧАНИЕ

Когда дизайн всей страницы указан с фиксированной шириной, значения ширины для бокового меню в процентах основаны на элементе с фиксированной шириной. Ширина не зависит от размера окна и остается постоянной.

4. **Добавьте левый край к главному содержимому.** Если боковое меню короче другого содержимого на странице, то текст из главного столбца обтекает меню снизу. Добавление левого края, большего или равного ширине бокового меню, выравнивает главное содержимое страницы, создавая иллюзию второго столбца:

   ```
   #main { margin-left: 180px ; }
   ```

   Удобно делать левый край чуть больше по ширине, чем боковое меню: это добавляет промежуток между двумя элементами. Если для установки ширины бокового меню вы используете проценты, то задавайте немного большее значение для левого края.

Избегайте установки ширины для раздела `div` с главным содержимым — браузеры расширяют его, чтобы оно занимало доступное место. Даже если вы хотите иметь дизайн с фиксированной шириной, вам не нужно устанавливать ширину для главного содержимого, что вы увидите в следующем разделе.

Использование плавающих элементов в разметках

Теперь, когда вы изучили свободную разметку с двумя столбцами, можете применять ее бесчисленным множеством способов. Преобразовать ее в разметку с фиксированной шириной — просто. Надо обернуть все теги внутри тела страницы другим тегом (например,), а затем создать стиль для этого нового элемента-контейнера, для которого определена ширина, скажем, 960 пикселов (см. рис. 12.2, *внизу*). Эта установка ширины удерживает все внутри контейнера.

ПРИМЕЧАНИЕ

Существует также вариант создания страницы с фиксированной шириной без применения дополнительного тега , «оборачивающего» все элементы: задайте ширину для тега <body>. Вы уже видели пример использования этого метода в гл. 7.

Разметить страницу на три столбца также нетрудно (рис. 12.3). Сначала добавьте другой тег `<div>` между этими двумя столбцами и переместите его вправо. Затем добавьте правый край к среднему столбцу так, чтобы, если текст в нем окажется длиннее нового правого бокового меню, он не обтекал боковое меню.

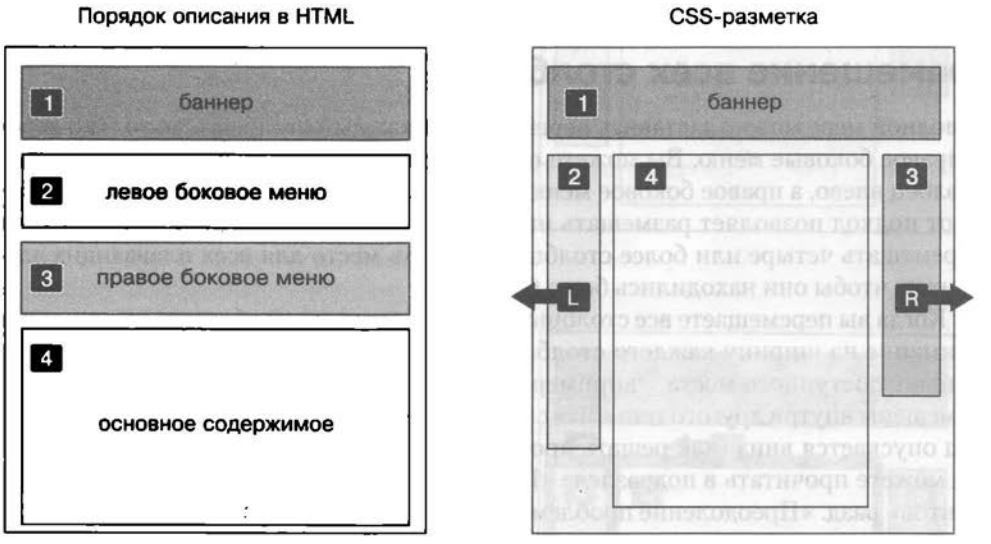


Рис. 12.3. При дизайне с тремя столбцами вы перемещаете левое и правое меню и добавляете левый и правый края к центральному столбцу. Диаграмма слева показывает порядок исходного HTML-кода, а справа вы можете увидеть, как выглядит веб-страница

В остальной части этого раздела рассматриваются многочисленные методы планировки CSS, которые используют разметки, основанные на «плавании».

ИНФОРМАЦИЯ ДЛЯ НОВИЧКОВ

Не изобретайте повторно колесо

Если такие термины, как *«свободная разметка»* и *«содержащий элемент»*, кажутся немного пугающими, не расстраивайтесь. Прежде всего, обучающие уроки этой главы шаг за шагом проведут вас через весь процесс разметки веб-страниц с помощью CSS. Однако нет такого правила, которое бы говорило о том, как вы должны создавать свои собственные CSS-разметки с нуля. В Интернете вы найдете множество предварительно созданных и протестированных дизайнов, которые можете применять для себя. Сайт Layout Gala предлагает 40 различных CSS-дизайнов, которые работают в большинстве распространенных браузеров, включая Internet Explorer 5 (http://blog.html.it/layoutgala/). Дизайны там — это просто «каркасы», состоящие из тегов `<div>` и CSS-кода, который размещает их. Все, что вам нужно сделать, — заполнить их собственными

ИНФОРМАЦИЯ ДЛЯ НОВИЧКОВ

настройками форматирования, такими как стиль шрифта и изображения. Джейкоб Мейерс (Jacob Meyers) предлагает на выбор 224 различные разметки, которые можно найти по адресу http://layouts.ironmyers.com.

Есть также немало генераторов разметок — эти онлайн-инструменты позволяют по вашему желанию настроить такие основные параметры, как количество столбцов, тип разметки и т. д. The Grid System Generator (www.gridsystemgenerator.com) позволяет устанавливать ширину страницы, количество столбцов и поля между ними. Затем можно скачать файлы с кодом HTML и CSS, созданные специально для разработчиков. А на сайте www.pagecolumn.com вы найдете подобный инструмент, предоставляющий различные типы разметок, включая свободные разметки и разметки с фиксированной шириной.

Перемещение всех столбцов

В полной мере можно заставить перемещаться каждый столбец, а не только левое и правое боковые меню. Вы можете переместить первое боковое меню и средний столбец влево, а правое боковое меню — вправо, как показано на рис. 12.2, *вверху*. Этот подход позволяет размещать на странице более трех столбцов. Вы можете перемещать четыре или более столбца, пока есть место для всех плавающих элементов, чтобы они находились бок о бок.

Когда вы перемещаете все столбцы в дизайне, то должны обратить пристальное внимание на ширину каждого столбца. Если совокупная ширина всех столбцов меньше доступного места, например, когда окно браузера меньше или столбцы помещены внутри другого тега `<div>` с установленной шириной, то последний столбец опускается вниз (как решать проблему перемещения плавающих элементов, вы можете прочитать в подразделе «Предотвращение перепадов плавающих элементов» разд. «Преодоление проблем перемещения» этой главы).

Вдобавок, перемещая не только боковые меню, вы сможете изменить порядок своих разделов `div` в HTML. Возьмите, например, левую диаграмму на рис. 12.3, которая показывает порядок тегов `<div>` для страницы. По принципу своей работы плавающие элементы должны появляться перед любым содержимым, которое определяется вокруг них, так что в этом примере область главного содержимого должна идти *после* бокового меню.

Порядок тегов `<div>` в HTML может показаться чем-то не очень важным, пока вы не попытаетесь просмотреть веб-страницу *без* CSS, что имеет место для многих альтернативных браузеров, включая экранных дикторов, которые читают содержание страницы вслух для слабовидящих посетителей. Без CSS весь материал бокового меню (которое часто включает навигационные элементы, рекламу или другую информацию, не относящуюся к главной теме страницы) появится перед содержимым, ради которого зашел посетитель. Неудобство, которое заключается в необходимости прокручивать одно и то же содержимое бокового меню на каждой странице, может отпугнуть многих посетителей. Кроме того, ваша страница будет менее доступна пользователям с плохим зрением, которым придется выслушивать, как их экранный диктор читает длинный список ссылок и рекламы, прежде чем получить реально необходимую информацию.

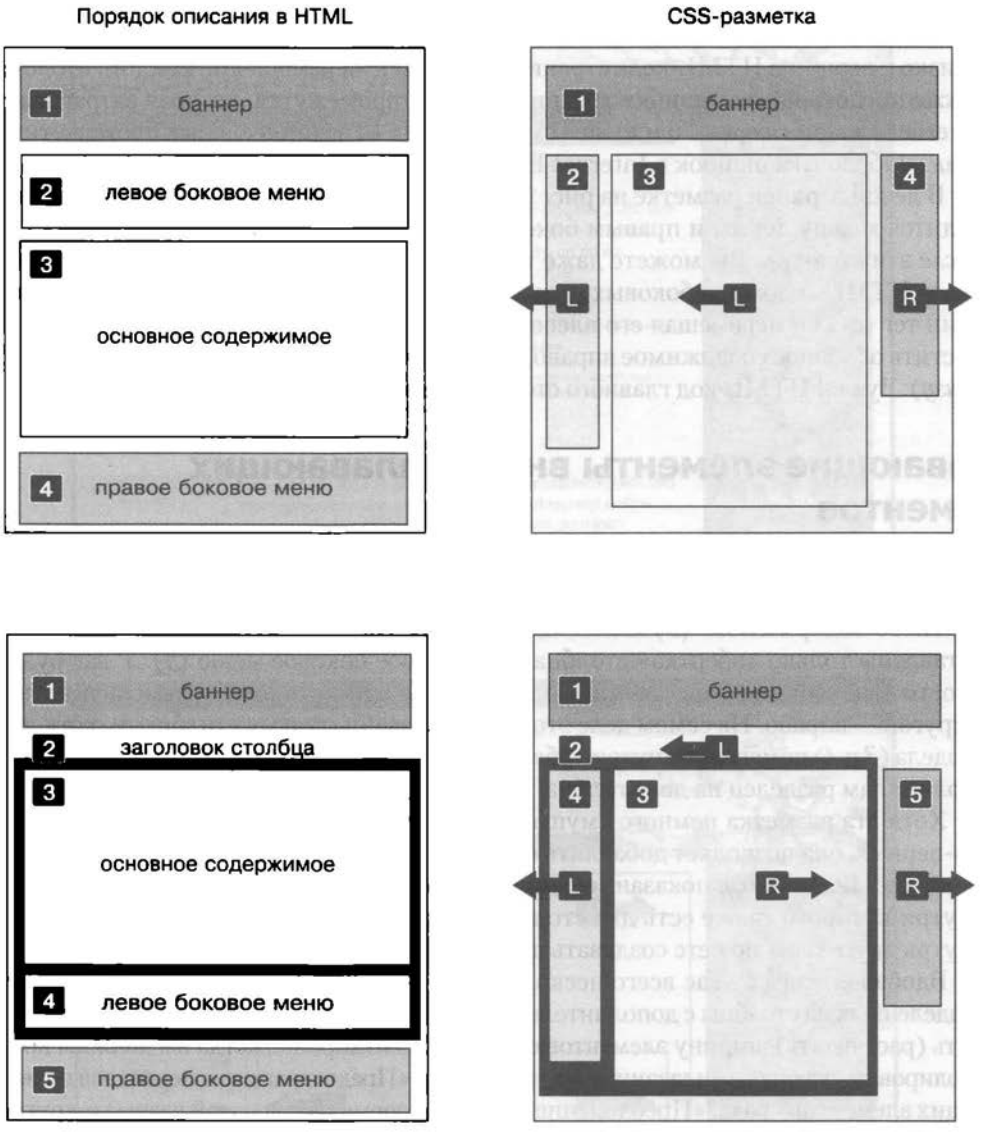


Рис. 12.4. Существует несколько способов сделать страницу плавающей. Используя различные методы перемещения, вы можете легко изменять порядок исходного HTML-кода страницы *(слева)*. Схемы на правой стороне представляют заключительную разметку веб-страницы

Если же это никак не затрагивает вас, то стоит поволноваться насчет поисковых машин. Многие из них ограничивают количество HTML-кода, прочитываемое при поиске сайта. На особенно длинной веб-странице они просто остановятся на определенном месте, возможно упустив важное содержимое, которое должно быть внесено в указатель поисковой машины. Кроме того, большинство поисковых

машин придают бóльшее значение тому HTML-коду, который находится в начале файла. Таким образом, если вас волнует рейтинг вашего сайта при поиске такими машинами, то имеет смысл убедиться в том, что важное содержимое максимально близко к вершине HTML-кода страницы. Наконец, перемещение каждого столбца также предотвращает ошибку трехпиксельного промежутка, которая затрагивает Internet Explorer версий 6 и ниже (см. подраздел «Трехпиксельные промежутки» разд. «Обработка ошибок в Internet Explorer 6» этой главы).

В левой верхней разметке на рис. 12.4 HTML-код с основным содержимым находится между левым и правым боковыми меню, что лучше, чем размещать его *после* этих блоков. Вы можете даже поместить описание главного содержимого перед HTML-кодом для боковых меню, определяя для него и левого бокового меню один тег `<div>` и перемещая его влево. Затем внутри этого тега `<div>` нужно переместить основное содержимое вправо, а левое боковое меню — влево (см. рис. 12.4, *внизу*). Вуаля! HTML-код главного столбца идет перед остальными тегами `<div>`.

Плавающие элементы внутри плавающих элементов

Нижняя разметка на рис. 12.4 иллюстрирует другую полезную методику — перемещение элементов *внутри* плавающих элементов. Предположим, что разделов главного содержимого *(3)* и левого бокового меню *(4)* не существует, а были оставлены только «обертка» столбца *(2)* и правое боковое меню *(5)*. У вас будет просто базовый дизайн с двумя столбцами, где один столбец перемещен влево, а другой — вправо. На самом деле это все еще дизайн с двумя столбцами, хотя два раздела (*3* и *4*) помещены внутри «обертки» столбца *(2)*. Различие в том, что левый столбец сам разделен на два столбца.

Хотя эта разметка немного смущает, она бывает полезной во многих случаях. Во-первых, она позволяет добавлять столбцы внутрь других столбцов. На разметке на рис. 12.5, *вверху*, показан маленький блок для подсказок в среднем столбце, внутри которого также есть два столбца. Вкладывая одни плавающие элементы внутрь других, вы можете создавать достаточно сложные дизайны.

Вдобавок, когда у вас всего несколько плавающих элементов, в свою очередь разделенных на столбцы с дополнительными плавающими элементами, легче вычислить (рассчитать) ширину элементов страницы. Это хорошо, когда вы должны контролировать перепады «плавания» (см. подраздел «Предотвращение перепадов плавающих элементов» разд. «Преодоление проблем перемещения» этой главы) и другие проблемы, возникающие, когда столбцы становятся слишком широкими.

Рассмотрим страницу на рис. 12.5, *вверху*. Чтобы добиться такого размещения объектов, создайте столбцы в столбцах, передвигая элементы внутри других плавающих элементов. В среднем столбце предусмотрена область подсказок, которая позволяет добавить простую заметку с двумя колонками, что придает привлекательности странице.

На рис. 12.5, *внизу*, видно, что неважно, в каком направлении перемещается контейнер (в этом случае — вправо), — вы просто передвигаете два дополнительных столбца влево и вправо.

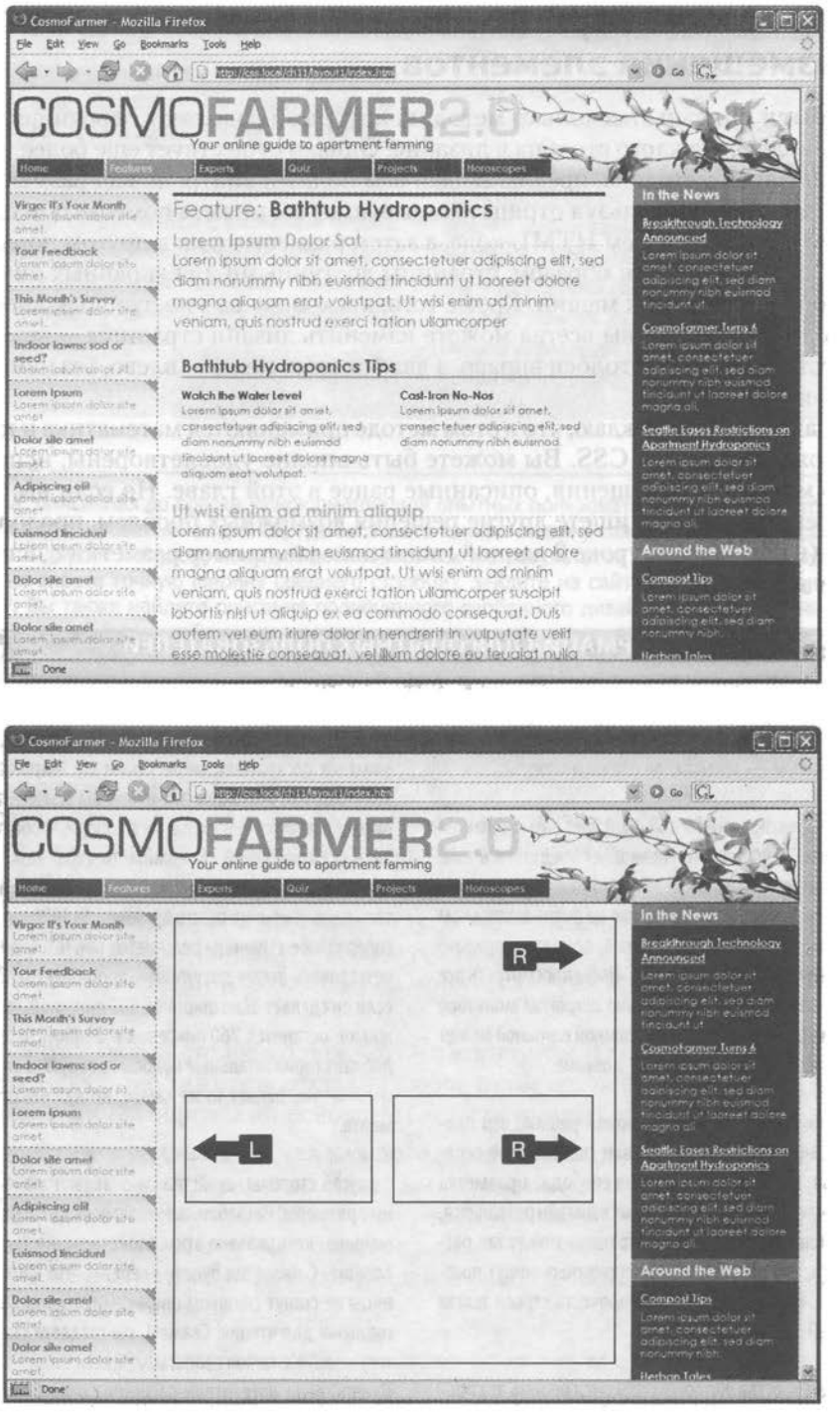


Рис. 12.5. Пример разметки с плавающими элементами внутри плавающих элементов

Использование отрицательных полей для размещения элементов

Выше были описаны несколько методик, которые позволяют переупорядочить HTML-код для каждого столбца в дизайне. Однако существует еще более усовершенствованная методика, предлагающая вам полный контроль над размещением ваших столбцов. Используя отрицательные поля, вы можете поместить теги `<div>` в любом порядке в своем HTML-коде, а затем разместить их в другом порядке на экране, оставляя, таким образом, страницы доступными для экранных дикторов, браузеров и поисковых машин. Кроме того, поскольку вам не нужно волноваться об исходном порядке, вы всегда можете изменить дизайн страницы — возможно, переместить главный столбец вправо, а два боковых меню — влево, без изменения HTML-кода страницы.

Однако я предупреждаю, что в этом методе применяются математика и некоторые сложные приемы CSS. Вы можете быть вполне удовлетворены, используя только методы перемещения, описанные ранее в этой главе. Но если вы готовы совершенствоваться и ищете другие решения возможных проблем, продолжайте читать (в обучающем уроке 2 данной главы показан пример разметки с использованием этого метода).

ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

Поиски компромисса

Два типа разметки — с фиксированной и непостоянной шириной — имеют свои плюсы и минусы.

Дизайн с фиксированной шириной дает вам много возможностей управления и позволяет убедиться в том, что разметка выглядит неизменно даже при различных размерах мониторов. Однако на очень маленьком мониторе посетителю, возможно, придется горизонтально прокручивать страницу, чтобы просмотреть все содержимое. А на действительно широком мониторе ваш прекрасный дизайн с постоянной шириной может напоминать маленькую щепку в океане.

Разметки с непостоянной шириной решают эти проблемы, но у них также есть свои собственные ограничения. На маленьком экране свободная разметка может настолько сжаться, что весь дизайн развалится, а на очень широком экране страница может так растянуться, что доставит пару неприятных минут посетителям, которые попытаются прочесть строки текста шириной 30 дюймов.

Есть CSS-свойства, которые стремятся решить эту проблему: `min-width`, `min-height`, `max-width` и `max-height`. Свойства `min-` говорят браузеру сделать элемент по крайней мере таким же широким или высоким, как указанное значение. Вы можете применить свойство `min-width` к тегу `<body>`, чтобы управлять всей шириной содержимого страницы, таким образом: `body { min-width: 760px; }`. Если посетитель расширит окно своего браузера до 1000 пикселов, содержимое страницы растянется так, что будет соответствовать всему доступному пространству. Однако если он сделает окно шириной 500 пикселов, то содержимое останется 760 пикселов в ширину, а браузер добавит горизонтальные полосы прокрутки. Свойство `min-height` делает то же самое, но для высоты элемента.

С другой стороны, свойства `max-` задают максимальные размеры. Разрабатываемый элемент может стать меньше, чем указано этим свойством, но только не больше. С ними вы будете уверены, что ваши страницы не станут слишком широкими, а значит, непригодными для чтения. Скажем, вы создаете стиль для тега `<body>` с таким свойством: `max-width: 1200px`. Теперь, если посетитель расширит свой браузер до 1800 пикселов (на своем невероятно дорогом 30-дюй-

ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

мовом мониторе), содержимое страницы не растянется на весь экран, а останется 1200 пикселов в ширину. Свойство `max-height` делает то же самое, но для высоты элемента.

Комбинируя эти два свойства, вы можете создать стиль, который изменяет размеры элемента только в пределах установленных значений, так что ваш дизайн никогда не станет слишком маленьким либо слишком большим:

```
body {
  min-width: 760px;
  max-width: 1200px;
}
```

Единственная проблема с этими так или иначе полезными свойствами состоит в том, что Internet Explorer версий 6 и ниже полностью игнорирует их. Если вы чувствуете в себе смелость, попробуйте `http:///`метод, использующий JavaScript, который динамически изменяет размеры элементов веб-страницы, в зависимости от размеров окна браузера: www.doxdesk.com/software/js/minmax.html.

ПРИМЕЧАНИЕ

Методика, описанная во врезке «Информация для опытных пользователей», работает только для разметок с фиксированной шириной, где вы знаете точную ширину каждого столбца. Чтобы познакомиться с другим методом, который добивается тех же результатов со свободными разметками (где вы не знаете точную ширину среднего столбца), зайдите на сайт www.alistapart.com/articles/holygrail/. Вы также найдете описание полноценного свободного дизайна с использованием отрицательных полей (все столбцы изменяют ширину в соответствии с окном браузера) в книге Flexible Web Design от Зое Майкли Джилленуотер (Zoe Mickley Gillenwater) (издательство New Riders).

Рассмотрим, как определяется разметка страницы при использовании отрицательных полей.

1. **Добавьте тег , который определяет все содержимое страницы.** Этот шаг предусматривает создание контейнера для установки фиксированной ширины всего содержимого страницы и предоставляет легкий способ сделать баннер, столбцы и нижний колонтитул одинаковой ширины.
2. **Установите ширину для созданного тега .** Создайте стиль для тега `<div>`, который задаст ему ширину. Например, 960 пикселов — типичный размер, удобный для посетителей, работающих на мониторах с разрешением 1024 × 768 пикселов.
3. **Опишите каждый столбец в теге с атрибутом ID или class.** Эта часть процесса — такая же, как и при создании любой разметки, основанной на плавающих элементах. Она определяет основные блоки разметки (рис. 12.6, *вверху слева*).
4. **Переместите разделы div для каждого столбца.** Вы должны передвинуть каждый столбец своей страницы. Когда вы передвигаете их влево, они располагаются бок о бок. Другой вариант — переместить левое боковое меню и главное содержимое влево, а правое боковое меню — вправо (поскольку все столбцы определены в теге `<div>` из шага 1, правое боковое меню остается рядом с центральным столбцом).

ПРИМЕЧАНИЕ

Если вы не используете главный тег , как описано в шаге 1, то должны переместить правое боковое меню влево. В ином случае оно зацепится за правый край окна браузера, создавая большой промежуток между правым боковым меню и главным содержимым.

5. **Установите ширину для каждого столбца.** Вы всегда должны определять ширину плавающего элемента. В зависимости от дизайна вы также можете добавить отступы, поля и границы. Тем не менее имейте в виду, что полная ширина столбцов в окне браузера — это сумма значений CSS-свойства `width`, левых и правых полей, отступов и границ для каждого столбца. И здесь появляется математика. Если вы невнимательны, полная ширина столбцов может превысить ширину, предоставленную главным тегом `<div>`, что вызовет страшные перепады при перемещении (см. подраздел «Предотвращение перепадов плавающих элементов» разд. «Преодоление проблем перемещения» этой главы).

ПРИМЕЧАНИЕ

Свойство width не определяет полную ширину, которую элемент занимает на экране. Поля, отступы и границы также учитываются. Если вам плохо знакомы эти параметры, прочтите разд. «Понятие блочной модели» гл. 7, чтобы освежить в памяти теорию блочной модели в CSS.

6. **Добавьте левое поле к главному столбцу.** Именно здесь методика отрицательных полей отличается от методов разметки, описанных ранее в этой главе. Левое поле должно занимать то же пространство, что и левое боковое меню. Если ширина левого бокового меню — 160 пикселов, то установите левое поле главного столбца шириной 160 пикселов: `margin-left: 160px`. Схема на рис. 12.6, *вверху справа*, показывает страницу такой, какая она есть в данный момент. Заштрихованная область слева от главного столбца *(3)* представляет его левое поле.

   Или, скажем, левое боковое меню занимает 160 пикселов в ширину, а вы хотите иметь промежуток размером 10 пикселов между меню и главным содержимым страницы. В этом случае добавьте 10 пикселов к левому полю главного столбца: `left edge: 170px`.

   Наконец, когда у левого бокового меню есть отступы и границы, их также нужно разложить на составляющие. Предположим, левое боковое меню шириной 160 пикселов имеет правую границу размером 2 пиксела и левый и правый отступы по 10 пикселов и вы хотите иметь промежуток 10 пикселов между ним и главным столбцом. Сложите все это вместе, чтобы получить значение левого поля для главного столбца. Так, 160 (значение ширины) + 2 (правая граница) + 10 (левый отступ) + 10 (правый отступ) + 10 (промежуток между боковым меню и главным столбцом) = 192 пиксела (`margin-left: 192px`).

7. **Примените отрицательное поле к тегу `<div>` левого бокового меню.** На данный момент левое боковое меню перемещено влево, но, поскольку оно идет после главного столбца в исходном HTML-коде, оно появляется в правой части главного столбца (см. рис. 12.6, *вверху справа*). Чтобы поставить его на место, используйте отрицательное поле, которое передвинет боковое меню параллельно главному столбцу к левому краю страницы. Единственная сложность — вычислить, какое значение должно быть у отрицательного поля, чтобы переместить боковое меню точно на расстояние, необходимое для размещения его в левой части страницы. Другими словами, левое поле должно занимать пространство, равное расстоянию от правого края главного столбца до левого края основного тега `<div>`.

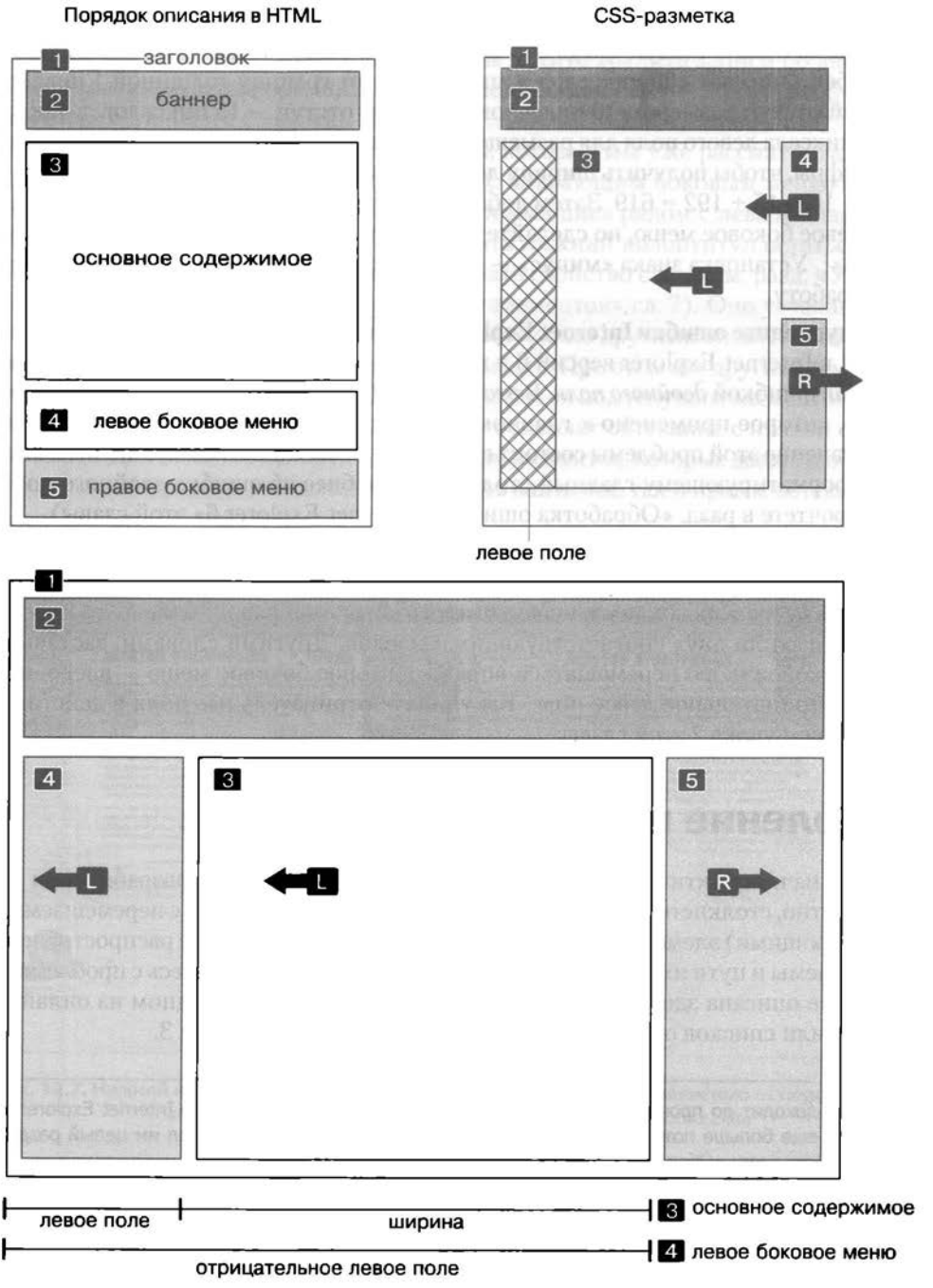


Рис. 12.6. Можно поместить разделы div в любом порядке в HTML-коде *(вверху слева)* и расположить их произвольно на экране *(внизу)*. Секрет заключается в использовании отрицательных полей для втягивания элемента, описываемого в коде позже, над элементом, предшествующим ему

Чтобы определить это значение, сложите ширину главного столбца, левые и правые поля и отступы, а также левую и правую границы. Скажем, главный столбец занимает 400 пикселов в ширину, имеет границу толщиной 1 пиксел, левый отступ размером 10 пикселов и правый отступ — 15 пикселов, а также 192 пиксела левого поля для размещения левого бокового меню. Просто сложите цифры, чтобы получить ширину левого поля левого бокового меню: 400 + 1 + + 1 + 10 + 15 + 192 = 619. Затем добавьте левое поле к стилю, форматирующему левое боковое меню, но сделайте его значение отрицательным: `left-margin: -619px;`. Установка знака «минус» — решающий момент, который обеспечивает всю работу.

8. **Предупредите ошибки Internet Explorer.** Когда вы используете отрицательные поля, в Internet Explorer версий 6 и ниже проявляется странная ошибка, называемая ошибкой *двойного поля*. В этом случае Internet Explorer удваивает левое поле, которое применено к главному столбцу, полностью разрушая дизайн. Устранение этой проблемы состоит в добавлении свойства `display: inline` к стилю, форматирующему главный столбец (подробнее об ошибке двойного поля вы прочтете в разд. «Обработка ошибок в Internet Explorer 6» этой главы).

Как только вы закончите с математикой, подход, основанный на отрицательных полях, вознаградит вас своей гибкостью. Если вы хотите поменять боковые меню так, чтобы левое боковое меню переместилось вправо, а правое — влево, то просто поменяйте стили двух соответствующих разделов. Другими словами, заставьте первое боковое меню перемещаться вправо, а второе боковое меню — влево, используя отрицательное левое поле. Вы увидите отрицательные поля в действии в обучающем уроке 2 этой главы.

Преодоление проблем перемещения

Когда вы начнете активно работать с CSS, то, как и многие веб-разработчики до вас, вероятно, столкнетесь с некоторыми сложностями при работе с перемещаемыми (плавающими) элементами. В этом разделе описаны некоторые распространенные проблемы и пути их решения. Если же вы когда-либо столкнетесь с проблемой, которая не описана здесь, то всегда можете спросить о ней на одном из онлайн-форумов или списков обсуждений, перечисленных в приложении 3.

ПРИМЕЧАНИЕ

Когда дело доходит до проектирования страниц, которые должны работать в Internet Explorer 6, появляется еще больше потенциальных ловушек. Их так много, что я посвятил им целый раздел этой главы (см. разд. «Обработка ошибок в Internet Explorer 6»).

Отмена и установка перемещения для элементов

Перемещаемые элементы — мощные средства проектирования, поскольку позволяют содержимому обтекать их вокруг. Перемещение фотографии позволяет тексту, находящемуся под ней, продвинуться вверх и «обернуться» вокруг изображения (см. рис. 12.1). Даже если вы создаете дизайны, основанные на плавающих

столбцах, иногда *не* нужно, чтобы содержимое передвигалось и оказывалось рядом с перемещаемым элементом. Например, вы хотите хранить записи об авторском праве, контактную информацию или другие подробности у основания веб-страницы, ниже остального содержимого.

В дизайнах с двумя и тремя столбцами, которые мы уже рассматривали, если главный столбец короче любого столбца с плавающим боковым меню, нижний колонтитул может передвинуться вверх, оказавшись рядом с левым плавающим столбцом (рис. 12.7, *слева*). Чтобы заставить нижний колонтитул остаться внизу под боковыми меню, вы можете использовать свойство `clear` (см. разд. «Управление обтеканием содержимого плавающих элементов» гл. 7). Оно устанавливает, с какой стороны элемента запрещено его обтекание другими элементами. Вы можете отменить обтекание с левого края элемента. При этом все другие элементы на этой стороне опустятся вниз и будут располагаться под текущим элементом (`clear: left;`). Аналогично свойство `clear: right;` отменяет обтекание с правой стороны элемента. Для нижнего колонтитула и других элементов, которые должны оказаться у основания страницы, вы должны устранить как левое, так и правое обтекание:

```
#footer { clear: both; }
```

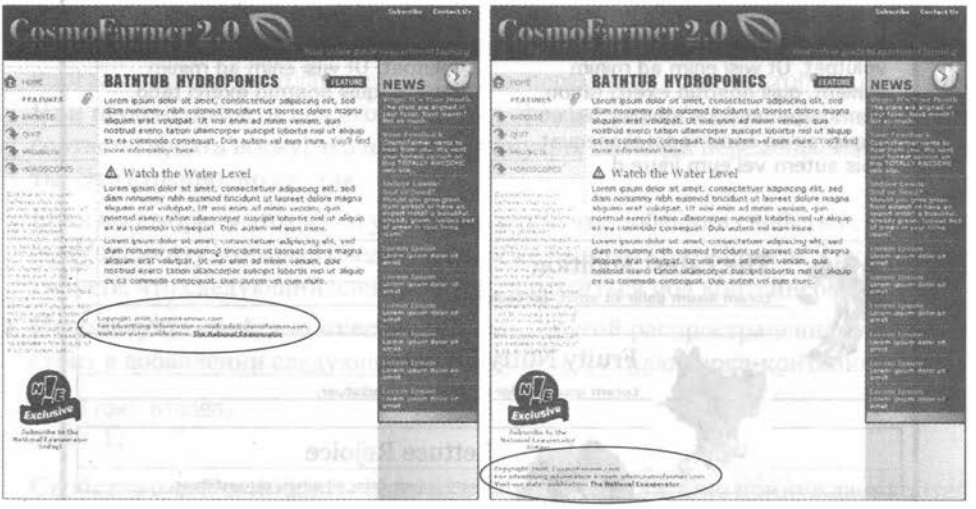


Рис. 12.7. Нижний колонтитул должен оставаться внизу страницы, независимо от перемещаемых элементов. Чтобы добиться этого, используйте свойство clear

Другая проблема появляется, когда вы перемещаете один или несколько элементов внутри неперемещаемого содержимого тега, такого как тег `<div>`. Когда перемещаемый элемент выше, чем остальное содержимое в разделе, он придерживается основания содержащего его объекта. Эта путаница особенно заметна, если у тега есть фоновый цвет или граница. В веб-странице на рис. 12.8 есть тег `<div>`, в котором определены тег `<h1>` и два столбца, созданные перемещаемыми. Фон и граница, которые появляются только вокруг заголовка, в действительности применяются ко всему тегу `<div>`, включая область с двумя столбцами. Однако, поскольку

столбцы перемещаемые, они выходят за пределы основания блока вместо того, чтобы расширять границы области.

Пример подобной проблемы показан на рис. 12.8, *внизу*. В этом случае каждое изображение перемещено влево внутри содержащего тега `<div>`, в котором задана граница. Поскольку изображения выше, чем их блоки, они выходят за пределы оснований блоков. К сожалению, этот пример еще хуже, чем предыдущий, потому что каждый рисунок заставляет нижнее изображение перемещаться вправо, создавая «ступенчатый» эффект.

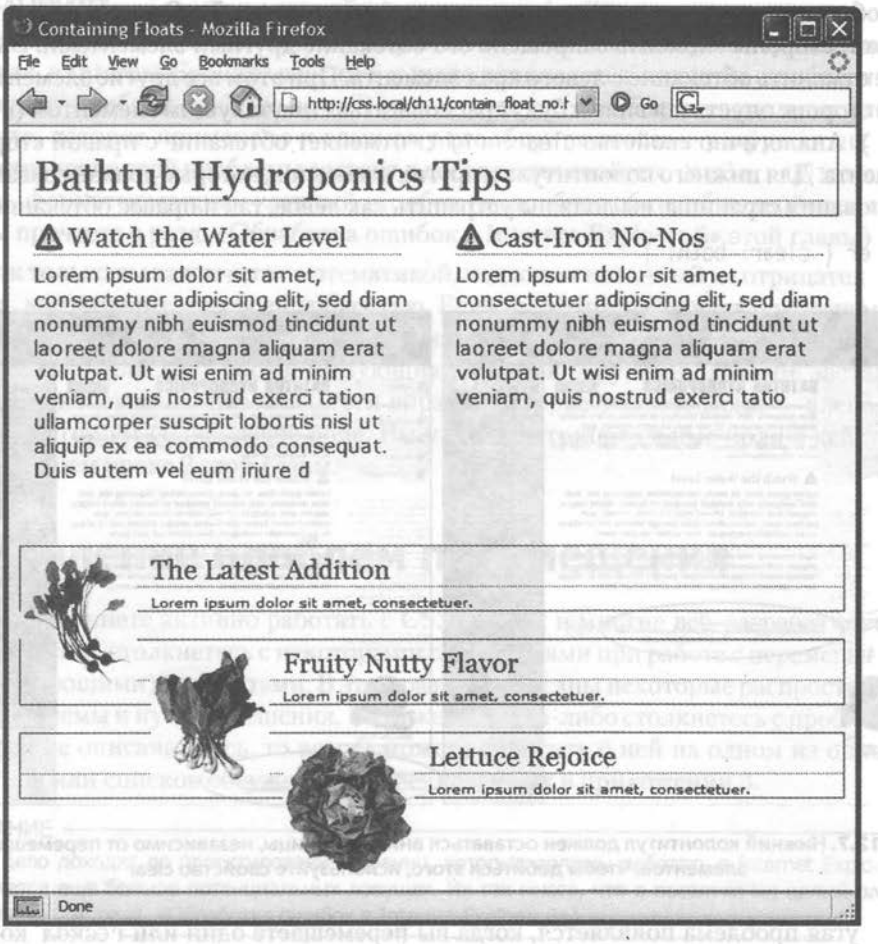


Рис. 12.8. Перемещаемый элемент может выйти за пределы содержащего его блока. Если в блоке определены фоновый цвет или граница, то выходящие элементы могут выглядеть так, как будто даже не являются частью блока *(вверху)*. Кроме того, перемещаемый элемент может врезаться в другие элементы, создавая «ступенчатый» эффект *(внизу)*

ПРИМЕЧАНИЕ

Чтобы получить хорошее объяснение того, почему перемещаемые элементы могут выходить за пределы содержащих их блоков, зайдите на сайт www.complexspiral.com/publications/containing-floats/.

У вас есть много способов избавиться от проблем, возникающих с перемещаемыми элементами. Мы рассмотрим их все, чтобы у вас был выбор.

- **Добавьте «освобождающий» элемент к основанию блока.** Это решение — наиболее простое. Нужно лишь добавить тег, определяющий, например, разрыв строки или горизонтальную линию, в качестве последнего элемента в теге , содержащем перемещаемый элемент (то есть прямо перед закрывающим тегом). Затем используйте свойство clear, чтобы укрепить этот дополнительный тег под перемещаемыми элементами.

  Этот метод расширяет блок, выявляя его фон и границу. Вы можете указать разрыв строки —
 (HTML) или
 (XHTML) — и добавить к нему класс: <br class = "clear"/>. Затем создайте для него такой стиль:

  ```
  br.clear { clear: both; }
  ```

  Проблема, связанная с этим решением, заключается в добавлении дополнительного кода HTML.

- **Сделайте перемещаемым элемент-контейнер.** Более легкий путь состоит в том, чтобы просто сделать перемещаемым тег , который содержит плавающие элементы. Перемещаемый контейнер расширяется так, чтобы полностью вмещать любые плавающие элементы. На рис. 12.9, *вверху*, тег , содержащий заголовок и два плавающих столбца, перемещен в левую сторону страницы. При необходимости вся его область — фон и границы — расширятся, чтобы соответствовать всему, что находится внутри, включая перемещаемые элементы. Это странно, но это так.

  Если вы выбрали этот путь, убедитесь в том, что добавили свойство clear к любому элементу, который следует за перемещаемым контейнером. Так вы гарантируете, что следующий элемент будет находиться под контейнером.

- **Используйте свойство overflow:hidden.** Другой распространенный метод состоит в добавлении следующих двух свойств к стилю блока-контейнера:

  ```
  overflow: hidden;
  zoom: 1;
  ```

  Свойство overflow:hidden — одно из странностей CSS. Оно принуждает контейнер расширяться и содержать плавающие элементы. Часть кода (zoom: 1) добавлена только для Internet Explorer 6 (и более ранних версий) и никак не влияет на другие браузеры. Если вы хотите, можно поместить это свойство в отдельной таблице стилей, используя условные комментарии IE (все странности zoom: 1 были описаны в гл. 7).

  В целом этот метод работает очень хорошо. Тем не менее, если у вас есть абсолютно расположенные элементы внутри контейнера, они могут не отображаться. Вы можете попасть в такую ситуацию, если у вас есть выпадающее меню внутри другого тега, и, когда появляются выпадающие элементы, кажется, что они находятся за пределами элемента-контейнера. Если это так, используйте какой-либо другой способ из описанных на этих страницах.

- **Примените «легко очищающий метод».** Пользуясь этой методикой, созданной Тони Эслеттом (Tony Aslett), разработчиком сайта CssCreator.com, и разработчиками сайта PositionIsEverything.com, вы добавляете всего несколько стилей и имя класса к тегу , содержащему плавающий элемент. Конечно, определение «легко очищающий метод» не совсем корректное, так как CSS-код в нем совсем не легкий. Вы должны добавить три различных стиля к своей таблице стилей: один применяется для Firefox, Safari, Opera, Internet Explorer 8 и других современных браузеров, а другие относятся к Internet Explorer 7 и более ранним версиям. Все вместе это выглядит следующим образом:

```
.clear:after {
  content: ".";
  display: block;
  height: 0;
  font-size: 0;
  clear: both;
  visibility: hidden;
}
.clear {
  zoom: 1;
}
```

Как только вы внесли эти стили в таблицу стилей, вы просто добавляете название класса к разделу div, содержащему плавающие элементы, который выходят за пределы блока: <class div = "clear"> (рис. 12.9, *внизу*). Этот метод очень надежный, но, в отличие от двух предыдущих, потребует добавления дополнительного кода HTML.

ПРИМЕЧАНИЕ

Использование свойства zoom приводит к тому, что ваша страница не пройдет проверку CSS-кода на корректность. Чтобы обойти это, вы можете поместить это выражение (наряду с любыми другими специальными стилями для Internet Explorer) во внешнюю таблицу стилей и присоединить ее к вашим веб-страницам, используя любую из методик, описанных в гл. 15.

Создание столбцов на всю высоту

HTML-таблицы не совсем подходят для разметки веб-страницы. Они добавляют много кода, их трудно обновлять и они не работают так же хорошо в альтернативных браузерах, например тех, что используются в мобильных телефонах. Но, что касается разметок, у таблиц есть один плюс — это возможность создавать столбцы равной высоты. Их применение позволяет добавлять фоновый цвет или графику к отдельному столбцу и делать так, чтобы они заполняли всю высоту страницы. Фоны двух боковых меню на рис. 12.10, *вверху*, заполняются на всю высоту экрана, создавая цельные границы с обеих сторон страницы.

Плавающие элементы CSS, с другой стороны, немного недорабатывают в этом отношении. Ячейки таблицы в строке всегда одной и той же высоты, чего не скажешь о блоках. Высота плавающего элемента обычно определяется его содержи-

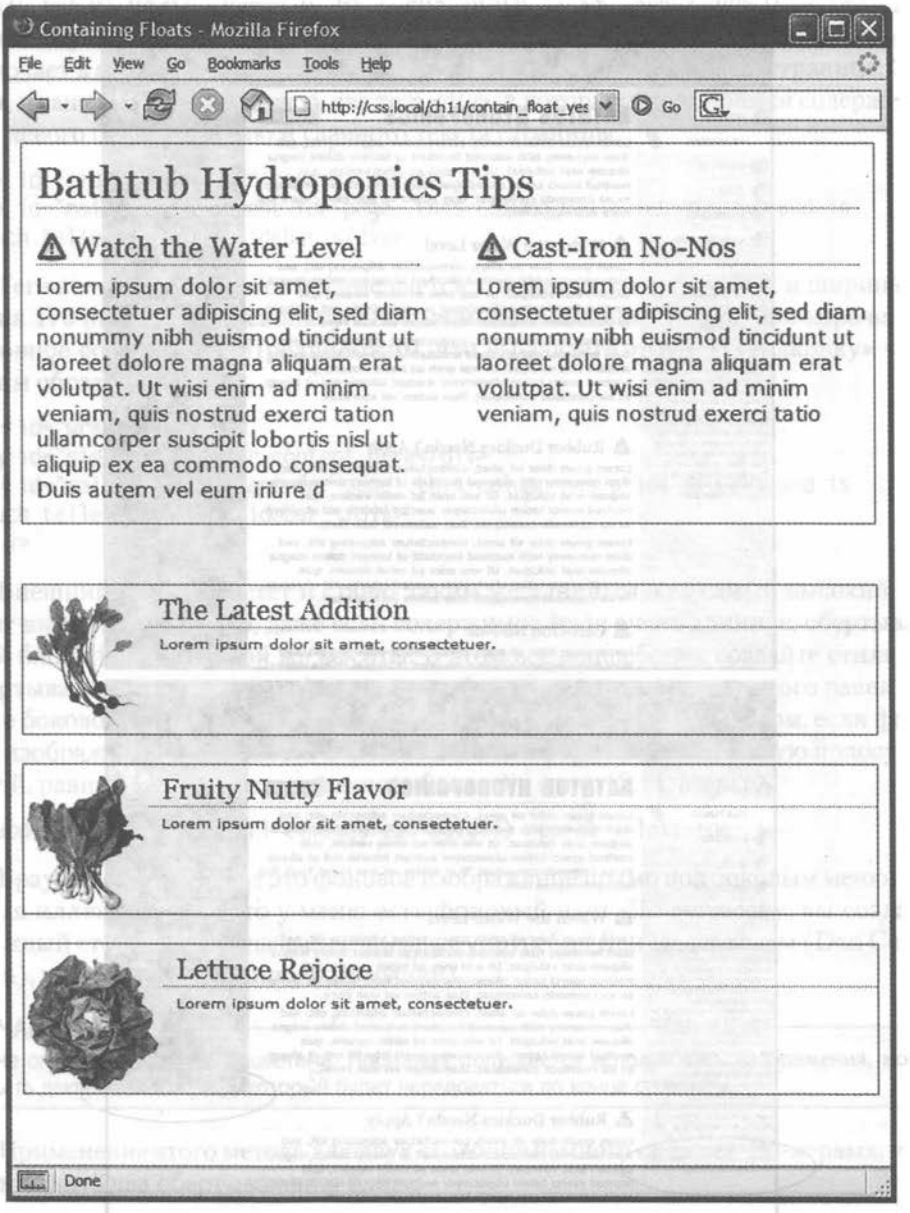


Рис. 12.9. Есть несколько способов сохранить плавающие элементы внутри содержащего их блока. Можно сделать плавающим сам блок *(вверху)* или использовать специальную комбинацию CSS, названную «легко очищающий метод» *(внизу)*

мым. Когда содержимого немного, элемент не очень высокий. Поскольку фоновое изображение или фоновый цвет заполняют только плавающий элемент, у вас могут получиться окрашенные столбцы, которые не доходят до основания страницы, как в обведенных кружком областях на рис. 12.10, *внизу*.



Рис. 12.10. Левое и правое боковые меню *(вверху)* показывают, как четкие фоновые цвета помогают визуально разделять области страницы. Когда фон бокового меню резко прерывается *(внизу)*, появляется пустое пространство, что выглядит непривлекательно

Но, как и у большинства проблем, связанных с CSS, здесь существует обходной путь. Секрет состоит в том, чтобы добавить фоновые изображения к тегу, который включает в себя «приземистое» боковое меню и другие столбцы на странице. Скажем, в вашем HTML-коде есть два тега , в которых определяется содержимое для левого бокового меню и главного текста страницы:

```
<div id="sidebar">Sidebar content here</div>
<div id="main">Main content for page, this column has a lot of text and is
  much taller than the sidebar.</div>
```

Тег бокового меню перемещается к левому краю страницы, и ширина его равна 170 пикселам. Поскольку в боковом меню немного текста, оно короче, чем основное содержимое. Предположим, что вы задаете этот тег-«упаковку» таким образом:

```
<div id="wrapper">
<div id="sidebar">Sidebar content here</div>
<div id="main">Main content for page, this column has a lot of text and is
  much taller than the sidebar.</div>
</div>
```

Внешний блок вырастет и станет таким же длинным, как самый высокий элемент внутри его, так что, даже если содержимое #main очень длинное, обертывающий блок div будет такой же высоты. В этом все волшебство: создайте стиль для обертывающего тега с фоновым изображением, ширина которого равна ширине бокового меню, подобрав нужный фоновый цвет. Таким образом, если фоновое изображение повторяется вертикально, оно формирует сплошную полосу высотой, равной высоте обертывающего блока div (рис. 12.11, *вверху*).

```
#wrapper { background: url (images/col_bg.gif) repeat-y left top; }
```

Браузеры показывают это фоновое изображение прямо под боковым меню, создавая иллюзию того, что у меню есть фоновый цвет. По существу, вы создаете «ложный столбец» (это название впервые употребил Дэн Цедерхольм (Dan Cederholm), разработавший данную методику).

ПРИМЕЧАНИЕ

Вы не ограничены чистыми цветами. Поскольку допускается использовать изображения, можно создать декоративный узор, который будет чередоваться до конца страницы.

Применение этого метода для двух столбцов немного сложнее. Во-первых, нужно добавить два обертывающих блока:

```
<div id="wrapper1">
  <div id="wrapper2">
    <div id="sidebar1">Sidebar content here</div>
    <div id="sidebar2">Second sidebar</div>
    <div id="main">Main content for page, this column has a lot of text
      and is much taller than the two sidebars.</div>
  </div>
</div>
```

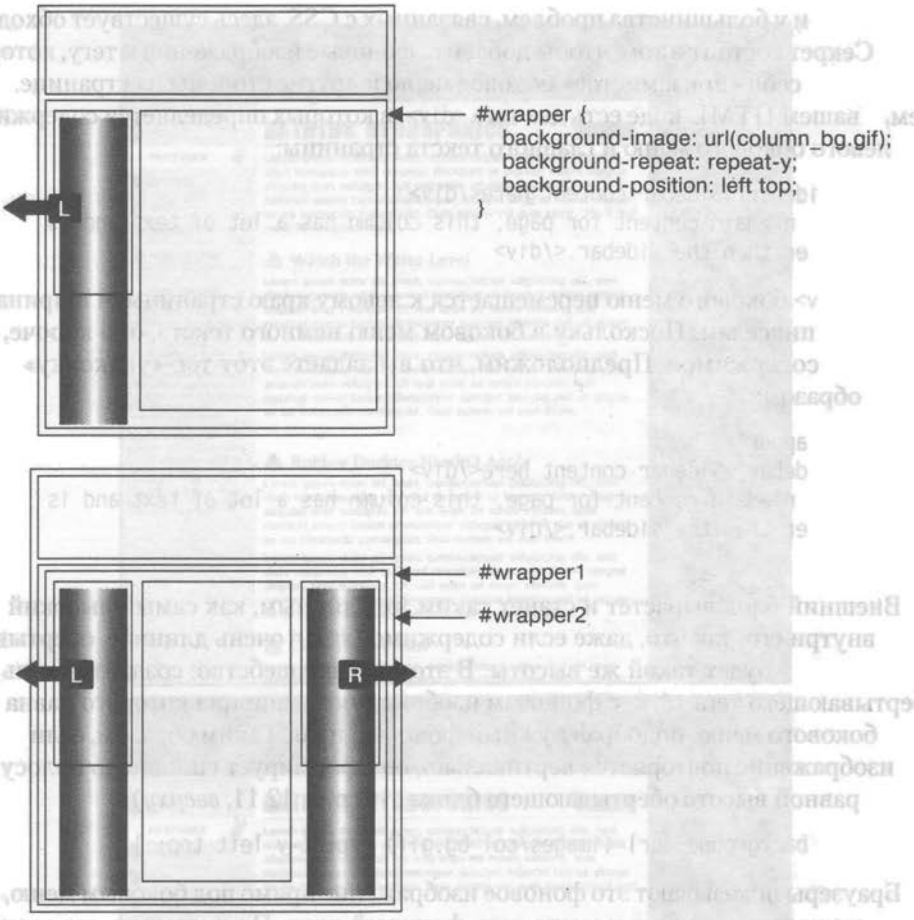


Рис. 12.11. Для получения фоновых изображений на всю высоту плавающих столбцов нужно обратиться к некоторым дополнительным тегам . Добавляя к ним фоновые цвета, вы создаете видимость столбцов одинаковой высоты

ПРИМЕЧАНИЕ

Если «упаковка» и каждый столбец имеют фиксированную ширину, вы можете создать видимость «ложного столбца» для левого и правого меню всего с одним изображением и «обертывающим» блоком. Просто сделайте рисунок таким же широким, как «упаковка», и задайте для левой стороны изображения цвет и ширину левого бокового меню, а для правой стороны — цвет и ширину правого бокового меню. Центральная часть должна соответствовать фоновому цвету центрального столбца.

Если первое боковое меню проявляется на левой стороне страницы, а второе боковое меню — на правой стороне, то создается два стиля. Примените один стиль к первому обертывающему тегу , чтобы добавить фон к левому боковому меню; другой же стиль примените ко второму обертывающему тегу , добавив фон к правому боковому меню (рис. 12.11, *внизу*).

```
#wrapper1 { background: url(images/col1_bg.gif) repeat-y left top; }
#wrapper2 { background: url(images/col2_bg.gif) repeat-y right top; }
```

При добавлении фонового изображения к правому столбцу убедитесь, что вы размещаете изображение вверху справа во второй «упаковке» так, что оно свободно простирается под вторым боковым меню на правой стороне страницы.

ПРИМЕЧАНИЕ

Если для определения ширины столбцов вы применяете проценты, то будет труднее создать иллюзию столбцов на всю высоту, используя графику. Однако это по-прежнему возможно. Чтобы узнать, как это делается, зайдите на сайт www.communitymx.com/content/article.cfm?page=1&cid=AFC58.

Предотвращение перепадов плавающих элементов

Может случиться так, что внезапно один из ваших столбцов просто опустится под другими (рис. 12.12, *вверху*). Видно, что есть много места для всех столбцов, чтобы они отлично сосуществовали бок о бок, но это не получается. Можно сказать, что у вас проявились перепады плавающих элементов.

Плавающий столбец опускается вниз, потому что недостаточно места, которое бы ему соответствовало. Будьте осторожны, если вы устанавливаете ширину для *каждого* столбца. Если ширина доступного места в окне браузера (или содержащего блока в дизайне с фиксированной шириной) меньше *общей* ширины столбцов, то могут появиться перепады. Кроме того, не забывайте о блочной модели в CSS: как обсуждалось в подразделе «Вычисление фактических размеров блочных элементов» разд. «Определение параметров высоты и ширины» гл. 7, ширина элемента, отображаемого в окне браузера, не определяется лишь значением его свойства `width`. Отображаемая ширина любого элемента — это комбинация его ширины, размеров левой и правой границ, левого и правого отступов, а также левого и правого полей. Для соответствия столбцам окно браузера (или содержащего блока) должно подстроиться под общую ширину.

Возьмем, например, простую разметку с тремя столбцами, представленную на рис. 12.12. Как вы можете видеть на верхнем изображении, эти три столбца не согласуются. Рассмотрим, из чего состоят элементы приведенной страницы.

- **Обертывающий блок.** Обертывающий тег `<div>` с фиксированной шириной заключает в себя весь дизайн. Его ширина равна 760 пикселам, таким образом, все три столбца не могут быть в сумме шире, чем это значение.
- **Первое боковое меню (с левой стороны).** Его ширина — 150 пикселов, но у него также есть отступы по 10 пикселов, что делает общую ширину меню равной 170 пикселам (150 пикселов ширины меню + 10 пикселов левого отступа + 10 пикселов правого отступа).
- **Главное содержимое.** Имеет ширину 390 пикселов, а также включает по 1 пикселу обеих границ и по 15 пикселов левого и правого полей, что делает полную ширину равной 422 пикселам (390 пикселов ширины содержимого + 1 пиксел левой границы + 1 пиксел правой границы + 15 пикселов левого поля + 15 пикселов правого поля).

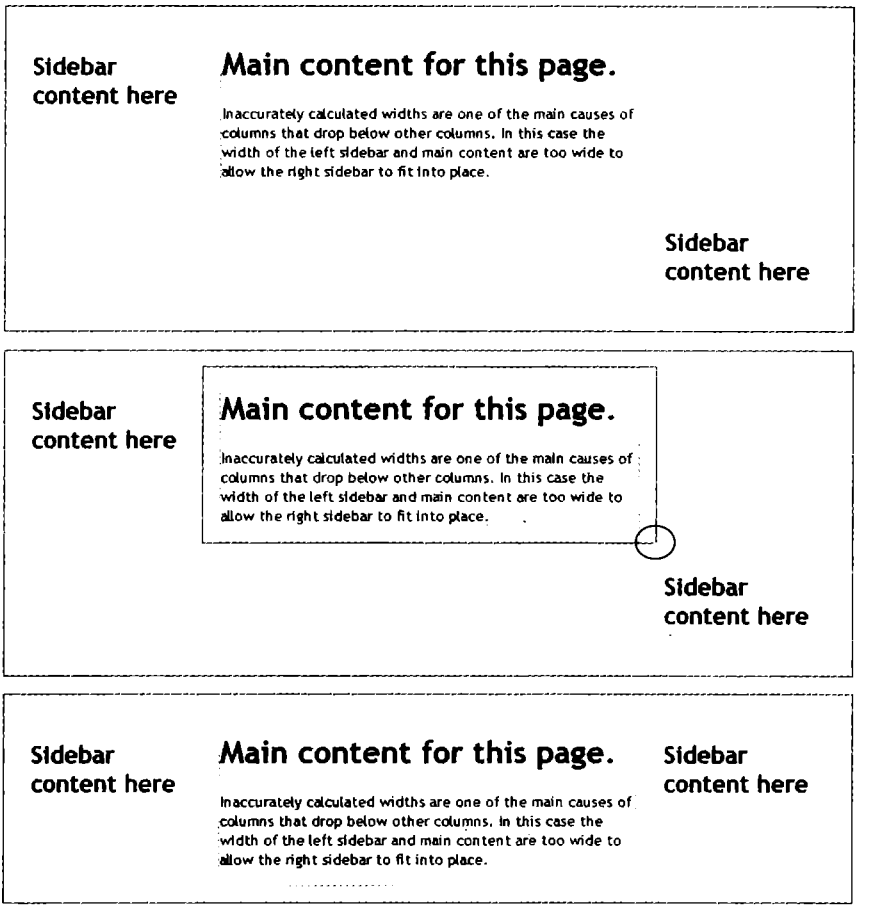


Рис. 12.12. Когда ширина плавающих элементов хоть на волосок больше содержащего их блока, последний элемент опускается ниже других *(вверху)*. Отрегулировав все элементы путем удаления небольших значений ширины, отступов или полей, вы можете решить проблему *(внизу)*

- **Второе боковое меню (с правой стороны).** Ширина этого элемента равна 150 пикселам. Он также включает в себя по 10 пикселов левого и правого отступов; в результате получается 170 пикселов, как и для первого бокового меню.

Фактическая ширина всех добавленных элементов составляет 762 пиксела. Это на 2 пиксела больше, чем ширина обертывающего тега `<div>`. На рис. 12.12, *посередине*, показана рамка вокруг тега `<div>` для главного содержимого, которая обозначает его полную ширину вместе с полями. Всего двух лишних пикселов ширины *(обведены кружком)* достаточно для того, чтобы заставить столбец опуститься вниз.

Решение заключается в удалении области размером 2 пиксела из любого элемента. Для этого измените левое и правое поля главного содержимого с 15 до 14 пикселов, что даст вам дополнительное место, необходимое для размещения всех трех столбцов, которые теперь будут находиться рядом.

Чаще всего именно из-за неверно оцененной ширины столбцов происходят перепады плавающих элементов, однако это не единственная причина. Рассмотрим, что еще может стать поводом для перепадов.

- **Ошибки округления при расчете ширины в процентах.** Будьте осторожны, когда устанавливаете ширину в процентах. Браузеры иногда делают ошибки при вычислении фактического количества пикселов, необходимого для отображения объекта на экране. Таким образом, они могут округлить числа в бóльшую сторону, делая элементы немного шире, чем доступное место. Будьте внимательны и сделайте полную ширину немного меньше 100 %.
- **Ошибка двойного поля в Internet Explorer 6.** В некоторых случаях Internet Explorer версий 6 и ниже удваивает поле, добавленное к плавающему элементу, делая его шире, чем он отображается в других браузерах. Если перепад плавающих элементов у вас проявляется только в этом браузере версии 6 или ниже, то причиной может быть именно эта ошибка. Как решить эту проблему, рассказывается в подразд. «Ошибка двойного поля» следующего раздела.
- **Трехпиксельный промежуток в Internet Explorer 6.** Иногда Internet Explorer версий 6 и ниже добавляет дополнительные 3 пиксела к краю плавающего элемента. Опять-таки, если вы видите, что перепады плавающих элементов наблюдаются только в этом браузере, причиной может быть данная ошибка. Как решить эту проблему, рассказывается в подразделе «Трехпиксельные промежутки» следующего раздела.
- **Курсивный текст.** Internet Explorer 6 отличается и здесь. Если плавающий элемент содержит курсивный текст, Internet Explorer 6 иногда делает этот элемент шире. Когда у вас происходят перепады плавающих элементов, а внутри их есть курсивный текст, проверьте, появляется ли эта проблема во всех браузерах или только в Internet Explorer. Чтобы устранить ее, вы можете удалить курсивный текст из бокового меню или добавить свойство `overflow: hidden` к стилю, форматирующему боковое меню.

Подведя итоги, можно сказать, что перепады плавающих элементов всегда вызваны недостаточным количеством места для размещения всех столбцов. Вместо того чтобы стремиться использовать каждый свободный пиксел экрана, задайте всем элементам немного больше места «для раскачки». Возьмите в привычку делать общую ширину столбцов немного меньше, чем это необходимо, и вам придется гораздо реже устранять неисправности, связанные с перепадами плавающих элементов.

Обработка ошибок в Internet Explorer 6

Браузер Internet Explorer очень многие CSS-стили воспринимает ошибочно, особенно когда дело доходит до разметок, основанных на перемещаемых элементах. Эти ошибки могут затронуть размещение этих элементов и выделенную им общую ширину. В лучшем случае в Internet Explorer ваша страница будет выглядеть практически так же, как и в других браузерах. В худшем случае эти ошибки могут

вызвать существенные проблемы отображения, например перепады плавающих элементов, которые рассматривались в предыдущем разделе.

В этом разделе мы рассмотрим самые распространенные проблемы и поговорим о том, как их решить.

ПРИМЕЧАНИЕ

Посмотрите врезку «Часто задаваемые вопросы» в разд. «Псевдоклассы и псевдоэлементы» гл. 3, чтобы узнать, стоит ли волноваться по поводу ошибок, связанных с этим браузером.

Ошибка двойного поля

Internet Explorer версий 6 и ниже иногда удваивает размер поля, который вы добавили к плавающему элементу. Эта проблема появляется, только если поле добавлено с той стороны, к которой перемещается элемент, — левое поле в перемещенном влево элементе или правое поле в перемещенном вправо элементе. На рис. 12.13 можно увидеть перемещенное влево боковое меню для навигации по сайту. Чтобы добавить небольшой промежуток между ним и левым краем окна браузера, к боковому меню добавляется левое поле шириной 10 пикселов.

Большинство браузеров, включая Internet Explorer 7 и 8, Safari и Firefox (см. рис. 12.13, *вверху*) добавляют требуемое пространство шириной 10 пикселов. Однако Internet Explorer 6 (см. рис. 12.13, *внизу*) удваивает поле до 20 пикселов. Даже при относительно маленьких полях страницы будут существенно различаться. Кроме того, если верстка очень плотная, с точно заданными плавающими элементами, расположенными бок о бок, то удвоенное поле может легко вызвать перепады плавающих элементов (см. подраздел «Предотвращение перепадов плавающих элементов» разд. «Преодоление проблем перемещения» этой главы).

ПРИМЕЧАНИЕ

Такое удвоение поля происходит, только если край элемента касается края содержащего его блока, поэтому, когда элемент перемещен влево вплотную к другому перемещенному влево элементу, его левое поле не удваивается.

Решение простое — добавьте свойство `display: inline;` к CSS-стилю для плавающего элемента:

```
#sidebar {
    float: left;
    margin-left: 10px;
    width: 160px;
  display: inline;
}
```

В этом случае свойство `display` применяется лишь для устранения ошибки браузера Internet Explorer. На самом деле единственная причина, по которой оно здесь, состоит в том, чтобы заставить боковой элемент «иметь разметку». Как альтернативный вариант вы можете использовать `zoom: 1` вместо `display: inline`. Перемещаемые элементы всегда являются блочными, и присвоение свойству `display` значения `inline` не изменяет этого (подробнее о свойстве `display` читайте в подразделе «Отображение встроенных и блочных элементов» разд. «Управление размерами

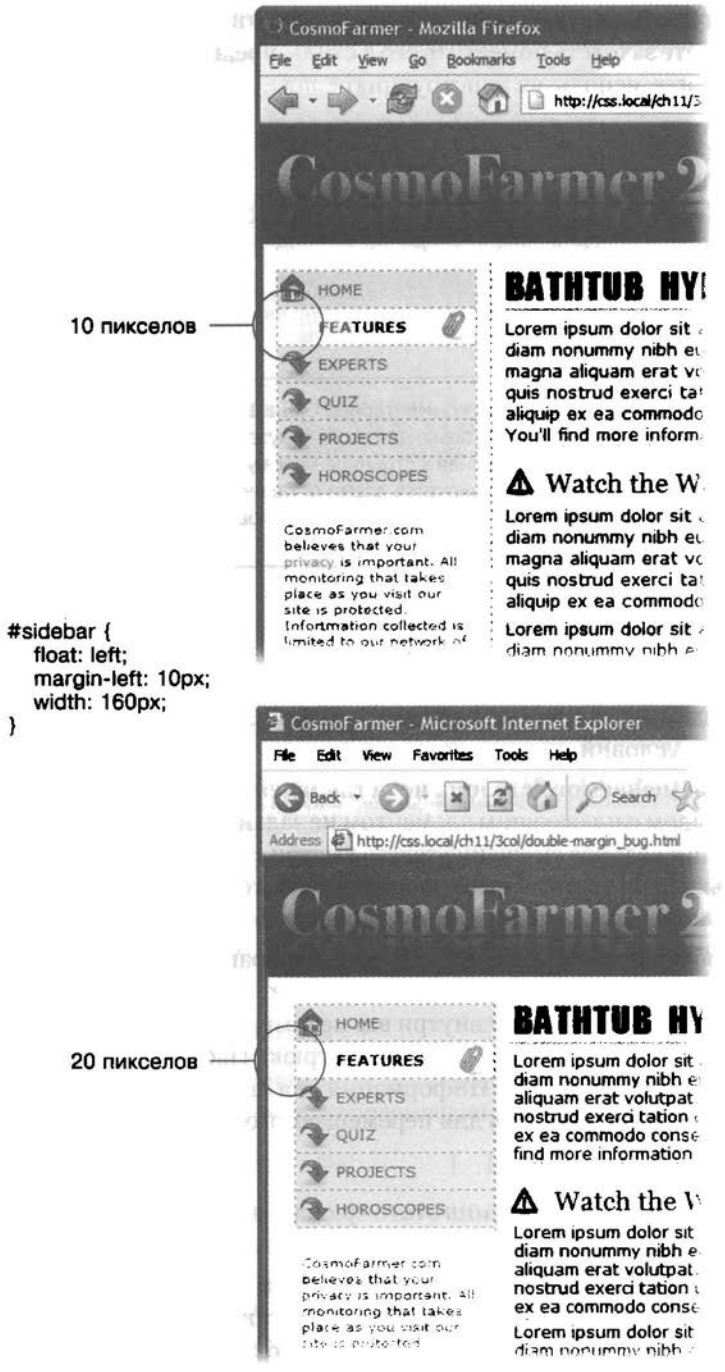


Рис. 12.13. Левое поле, относящееся к перемещенному влево элементу, должно смещать элемент на 10 пикселов от левого края страницы. Firefox *(вверху)* все делает правильно, а Internet Explorer 6 *(внизу)* удваивает это поле, значительно изменяя вид страницы

полей и отступов» гл. 7). Однако, хотя это свойство не затрагивает другие браузеры, вы можете захотеть поместить его в стиль, предназначенный специально для Internet Explorer, используя конструкцию * html:

```
#sidebar {
    float: left;
    margin-left: 10px;
    width: 160px;
  }
* html #sidebar {
  display: inline;
}
```

ПРИМЕЧАНИЕ

Особенность использования условных комментариев CSS в Internet Explorer позволяет еще лучше изолировать стили, предназначенные только для этого браузера, чем это делает конструкция * html. Внешняя таблица стилей, присоединенная к странице через условный комментарий, читается только браузером Internet Explorer и игнорируется всеми остальными (подробнее об этом рассказывается в подразделе «Изолируйте CSS для Internet Explorer условными комментариями» разд. «Управление браузером Internet Explorer» гл. 15).

Трехпиксельные промежутки

Internet Explorer версий 6 и ниже добавляет промежуток размером 3 пиксела между перемещаемым и неперемещаемым столбцами. Точное его размещение зависит от нескольких условий.

- **Для неперемещаемого столбца не установлена ширина или высота.** Если для столбца рядом с плавающим элементом не заданы никакие размеры, то вы увидите трехпиксельное смещение текста внутри столбца. Это смещение проявляется только вдоль плавающего элемента, так что, когда элемент заканчивается, текст вновь возвращается к левому краю столбца (рис. 12.14).

  Лучшее решение в этом случае — просто не обращать внимания. Дополнительный отступ не очень раздражает и не сдвигает элементы на странице. Но если стремление к совершенству внутри вас не позволяет допустить появление этой ошибки, можете устранить ее, выполнив трюк, известный как «добавление разметки» (описан во врезке «Информация для опытных пользователей» далее). Добавьте следующий стиль для перемещаемого столбца:

  ```
  * html #mainColumn { zoom: 1; }
  ```

  Обратная сторона применения этого трюка — появление новой ошибки, которая рассматривается далее.

- **Для неперемещаемого столбца установлена ширина или высота.** Если для столбца рядом с плавающим элементом все-таки заданы размеры, появляется другая ошибка — трехпиксельный промежуток между этими двумя объектами (рис. 12.15, *слева*). Эта ошибка более серьезная, чем описанная выше, так как такой промежуток может вынудить второй столбец опуститься ниже плавающего элемента (см. рис. 12.15, *справа*).

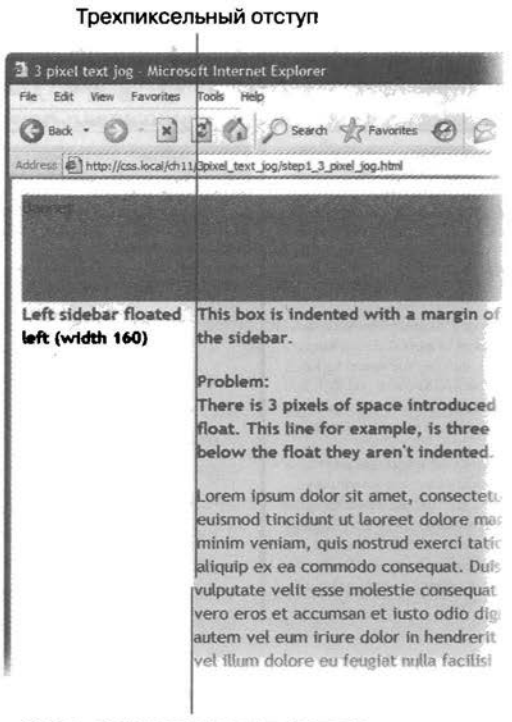


Рис. 12.14. Левое боковое меню выдвигается влево, в то время как центральный столбец не является перемещаемым. Левая сторона этого столбца углубляется достаточно далеко влево, так что столбец не «обертывается» вокруг основания бокового меню

Устранение этой проблемы проводится в два этапа. Сначала нужно удалить левое поле неперемещаемого столбца (но только для Internet Explorer версий 6 и ниже):

```
* html #mainColumn { margin-left: 0; }
```

Затем следует определить правое поле размером 3 пиксела для перемещаемого столбца. Это позволит подтянуть неперемещаемый столбец на его место:

```
* html #sidebar { margin-right: -3px; }
```

В любом нормальном браузере эти стили не имеют никакого смысла. Некоторые полные решимости CSS-эксперты (очевидно, у которых много свободного времени) придумали данные стили, чтобы заставить Internet Explorer работать без сбоев. Если хотите получить больше информации об этом феномене, зайдите на сайт http://www.positioniseverything.net/explorer/threepxtest.html.

Еще одно возможное решение состоит в том, чтобы сделать перемещаемыми все столбцы. В примерах, показанных на рис. 12.14 и 12.15, удаление левых полей из фиксированного столбца и перемещение его либо влево, либо вправо позволит

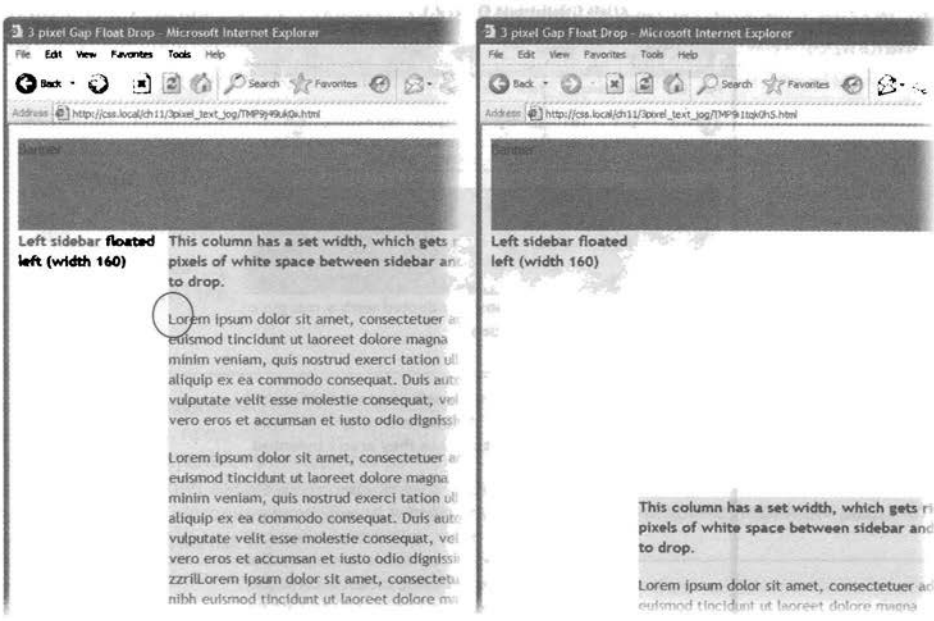


Рис. 12.15. Если один столбец перемещен влево, а другой столбец с фиксированной шириной «обертывается» вокруг него, Internet Explorer версий 6 и ниже добавляет трехпиксельный промежуток между ними *(слева)*. В дизайнах с фиксированной шириной это может привести к смещению столбца под плавающий элемент *(справа)*

избежать появления любого из описанных вариантов проблемы трехпиксельного промежутка:

```
#mainColumn {
  float: left;
}
```

Это решение кажется быстрым, но на самом деле оно сложнее, так как вам придется управлять еще одним плавающим элементом в дизайне.

ПРИМЕЧАНИЕ

Хотя многие ошибки в IE 7 были устранены, некоторые все же встречаются. Прочитать о них вы можете на странице http://css-discuss.incutio.com/?page=IE7. К счастью, браузер IE 8, который все стремительнее вытесняет IE 7, работает с CSS очень хорошо.

Другие проблемы Internet Explorer

Помимо описанных, в Internet Explorer версий 6 и ниже могут проявиться еще некоторые ошибки разметок, основанных на перемещаемых объектах. Многие из них настолько редкие, что вы можете никогда не столкнуться с ними в своих проектах. Но на всякий случай упомяну об отдельных возможных проблемах, которые могут появиться при просмотре страницы в данном браузере.

- **Если нижняя часть плавающего элемента просто исчезает,** это может быть *«ошибка гильотины»*. Для получения информации о причине и способах решения зайдите на сайт http://www.positioniseverything.net/explorer/guillotine.html.
- **Содержимое внутри плавающего элемента не показывается,** но иногда проявляется, если вы изменяете размеры окна браузера или прокручиваете страницу. Эту странность называют ошибкой peek-a-boo («на просвет»). Узнайте о ней подробнее на сайте http://www.positioniseverything.net/explorer/peekaboo.html.

ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

Получили разметку?

Как вы уже, вероятно, поняли, в Internet Explorer 6 проявляется много ошибок при отображении страниц, созданных с помощью CSS. Некоторый базовый CSS-стиль, который прекрасно выглядит в Internet Explorer 8, Firefox или Safari, разрушается в Internet Explorer версий 6 и ниже. К счастью, популярность IE 6 уменьшается, но тем не менее им все еще пользуются. Как оказывается, вы можете устранить многие ошибки этого браузера, подключая специальное свойство `layout`. Оно не относится непосредственно к CSS и при этом ничего не делает с правилами HTML. Это просто концепция, встроенная проектировщиками в Internet Explorer (версий 7 и более ранних). При необходимости браузер определяет, нужна элементу страницы разметка или нет.

В Internet Explorer перемещаемые элементы, пункты списка и абсолютно позиционированные элементы отображаются по-разному в зависимости от того, есть ли у них разметка. В подразделе «Исправление ошибок Internet Explorer» обучающего урока 2 гл. 9 вы видели, что Internet Explorer 6 не делает всю область ссылки пригодной для щелчка, когда ссылка определена как блочный элемент. Вы можете решить эту проблему, создав стиль только для данного браузера:

```
* html a .nav a { zoom: 1; }
```

Назначение стиля не в том, чтобы увеличить ссылку. Свойство `zoom` является специфичным для Internet Explorer и предназначено для увеличения элемента страницы (используя JavaScript). Однако свойство `zoom` также инициирует разметку в Internet Explorer 6. По причинам, известным только Microsoft, подключение разметки заставляет браузер рассматривать всю блочную область ссылок как пригодную для щелчка.

Помимо `zoom`, есть несколько других CSS-свойств, также включающих разметку в Internet Explorer: `position: absolute;`, `float: left;`, `float: right;`, `display: inline-table;`, любое значение `width` и `height`. Свойство `zoom` хорошо тем, что не влияет на то, как элемент выглядит в любом другом браузере, в отличие от реальных свойств CSS, таких как `width` и `height`, поэтому Safari, Firefox и другие браузеры благополучно не замечают его. Это значит, что вы можете использовать свойство `zoom` везде, где нужно устранить ошибку IE, добавляя разметку к элементу и при этом не опасаясь за порчу страницы в других браузерах. Обратная сторона применения этого свойства состоит в том, что для CSS оно некорректно и не будет признано правильным W3C (см. врезку «Информация для новичков» в разд. «Внешние таблицы стилей» гл. 2).

Internet Explorer 7 (на момент написания этой книги) все еще выдает несколько ошибок, которые необходимо устранять, добавляя разметку. Любое из перечисленных выше свойств добавит разметку к элементу в этом браузере, то же самое сделают и следующие свойства: `min-width`, `max-width`, `min-height` и `max-height` (см. врезку «Информация для опытных пользователей» в разд. «Использование плавающих элементов в разметках» этой главы).

В этой книге постоянно рассказывается, что следует использовать разметку, чтобы преодолеть различные ошибки Internet Explorer. Для всестороннего рассмотрения данной темы зайдите на сайт www.satzansatz.de/cssd/onhavinglayout.html. Кроме того, Microsoft также предлагает свои варианты исправления ошибок на сайте http://msdn.microsoft.com/en-us/library/bb250481(VS.85).aspx.

Обучающий урок 1: разметки с множеством столбцов

В этом обучающем уроке мы рассмотрим, как создавать разметку на несколько столбцов, основанную на плавающих элементах. Кроме того, вы научитесь создавать свободные дизайны на два и три столбца и дизайны с фиксированной шириной.

Чтобы приступить, загрузите файлы с сайта **www.sawmac.com/css2e/**. Как это сделать, описано в конце гл. 2. Файлы для этого урока находятся в папке **12\layout1**.

Структурирование HTML

Первый шаг в создании разметки, основанной на CSS, — идентификация различных элементов на странице. Вы делаете это, «упаковывая» части HTML-кода в теги `<div>`, каждый из которых представляет отдельный элемент страницы.

1. Откройте файл `start.html` в текстовом редакторе и установите курсор на пустой строке после HTML-комментария: `<!-- sidebar goes here -->`.

   Как вы можете видеть, часть работы в HTML уже сделана: на данный момент описаны баннер и нижний колонтитул. Перед тем как создавать какие-нибудь стили, вы должны добавить структуру и содержимое для страницы. Затем вы добавите тег `<div>` для левого бокового меню.

2. Добавьте открывающий тег `<div>` для бокового меню: `<div id = "sidebar">`. Затем нажмите клавишу **Enter**, чтобы создать новую строку.

   Если бы вы создавали веб-страницу с нуля, то в этом пункте пришлось бы добавлять HTML-код для бокового меню на странице и, возможно, определять список статей для сайта, ссылки на родственные сайты и т. д. В данном случае вам не придется этого делать. Код для неупорядоченного списка ссылок ждет вас в другом файле. Осталось только скопировать его и добавить на страницу.

3. Откройте файл `sidebar.txt`, скопируйте его содержимое, а затем вернитесь к файлу `start.html`. Вставьте скопированный HTML-код после тега `<div>`, который вы создали в шаге 2.

   Боковое меню почти готово. Осталось лишь закрыть тег `<div>`.

4. Сразу же после кода, который вы только что вставили, введите `</div>`.

   Вы только добавили на страницу первый элемент разметки. Чуть позже мы разработаем стиль этого элемента так, чтобы он был похож на столбец. Но сначала вы должны добавить еще немного кода.

5. Установите курсор в пустую строку после следующего HTML-комментария: `<!-- main content goes here -->`, а затем наберите `<div id = "main">`.

   Этот раздел будет хранить главное содержимое страницы. Нужный HTML-код вы также возьмете в другом файле.

6. Откройте файл `story.txt`, скопируйте его содержимое, вернитесь к файлу `start.html` и вставьте скопированный код после тега `<div>`, который вы только что создали. Добавьте закрывающий тег `</div>` точно так же, как в шаге 4.

   Это весь HTML-код, который нужен для создания дизайна. Теперь пришло время переключиться на создание CSS.

Создание стилей разметки

Если вы предварительно просмотрите страницу, то увидите, что для баннера, навигационных кнопок и текста стили уже разработаны. Так получилось потому, что у этой страницы есть присоединенная внешняя таблица стилей с некоторым базовым форматированием. Далее нужно создать стили для форматирования столбцов страницы.

1. В текстовом редакторе щелкните кнопкой мыши перед закрывающим тегом </head> в верхней части файла. Введите <style type = "text/css">, а затем нажмите клавишу Enter.

   Этот код — открывающий тег для внутренней таблицы стилей. Как в других программах-примерах этой книги, вы задаете стили во внутренней таблице стилей, которая упрощает создание и тестирование стилей. Как только вы все сделаете, нужно переместить стили во внешнюю таблицу стилей, как описано в разд. «Внешние таблицы стилей» гл. 2.

2. Добавьте следующий стиль для бокового меню:

   ```
   #sidebar {
     float: left;
     width: 160px;
     margin-top: 10px;
   }
   ```

   Он перемещает боковое меню на левую сторону страницы, задает ему ширину 160 пикселов и добавляет немного места, чтобы отделить меню от баннера, находящегося выше. Свойство width — важное в этом стиле. Если только вы не перемещаете изображение, для которого задана ширина, всегда нужно устанавливать ширину плавающего элемента. В ином случае браузер установит ширину на основе содержимого внутри плавающего элемента, что приведет к противоречивым результатам.

3. Нажмите клавишу Enter и наберите </style>, чтобы завершить внутреннюю таблицу стилей. Предварительно просмотрите страницу в браузере.

   Боковое меню теперь представляет собой столбец, выровненный по левому краю. Когда текст главного столбца достигает основания бокового меню, он обтекает основание бокового меню, как показано на рис. 12.16. Хоть это и типично для плавающих элементов, это не то, что нам нужно сейчас. Чтобы основной текст содержимого появился в виде отдельного столбца, следует добавить достаточное по размеру левое поле. Это позволит структурировать основной текст за пределами бокового меню.

4. Создайте стиль для второго столбца:

   ```
   #main {
     margin-left: 180px;
   }
   ```

   Поскольку ширина бокового меню 160 пикселов, поле размером 180 пикселов отодвигает основное содержимое на дополнительные 20 пикселов, создавая промежуток между этими двумя столбцами. Это дополнительное пустое



Рис. 12.16. Плавающий элемент в действительности не создает столбец на странице. Он просто смещает любое содержимое, которое обтекает его, до того места, где он заканчивается *(обведено кружком)*. После этого содержимое вновь переходит на свое место под элементом

место не только облегчает прочтение текста, но и улучшает внешний вид страницы.

Предварительно просмотрев страницу, вы увидите, что пришли к разметке с двумя столбцами.

Добавление еще одного столбца

Как вы могли видеть, дизайн с двумя столбцами создать несложно. Добавив третий столбец, вы сможете преподнести своим посетителям еще большее количество информации. Такую разметку также несложно создавать — действия в этом случае практически такие же, как в предыдущей части этого обучающего урока.

1. Откройте файл `secondary.txt`. Скопируйте из него весь HTML-код, а затем вернитесь к файлу `start.html`.

   HTML-код для следующего столбца идет между двумя разделами страницы.

2. Установите курсор сразу после закрывающего тега `</div>` для элемента бокового меню (прямо перед комментарием `<!-- main content goes here -->`). Затем нажмите клавишу **Enter**, чтобы добавить пустую строку.

   Часто бывает трудно найти нужный закрывающий тег `</div>`, когда используется много блоков для структурирования страницы. Вот почему HTML-комментарии, такие как этот, могут помочь идентифицировать и отслеживать HTML-код на вашей странице.

3. Введите `<div id = "secondary">`, нажмите **Enter** и вставьте HTML-код, который вы скопировали в шаге 1. Вновь нажмите клавишу **Enter** и добавьте закрывающий тег `</div>`.

   Когда вы закрываете тег `<div>`, вы заканчиваете HTML-код третьего столбца страницы. Теперь начнем разрабатывать для него стиль.

4. Под стилем `#main`, который вы создали в шаге 4 в предыдущем подразделе, добавьте новый стиль во внутреннюю таблицу стилей:

```
#secondary {
  float: right;
  width: 180px;
}
```

   Задав такой стиль, вы передвигаете столбец в правую часть страницы, чтобы по обе стороны главного содержимого располагались боковые меню. Здесь также появляется проблема с первым столбцом (см. рис. 12.16) — главное содержимое обтекает новое боковое меню. Чтобы устранить это, нужно добавить в стиль `#main` правое поле.

5. Отредактируйте стиль `#main` следующим образом:

```
#main {
  margin-left: 180px;
  margin-right: 200px;
}
```

   Теперь для страницы создана завершенная разметка с тремя столбцами.

6. Проверьте страницу в браузере. Если вы измените размеры окна, то увидите, что страница приспосабливается, чтобы соответствовать им.

СОВЕТ

В этом дизайне для левого и правого боковых меню установлена фиксированная ширина, так что, даже когда вы сделаете окно браузера очень большим, они не изменят свои размеры. Вы также можете заставить эти столбцы менять ширину, просто устанавливая значения их ширины в процентах, а также задавая левые и правые поля в процентах в стиле #main.

Новый столбец, который вы только что добавили, выглядит не очень хорошо, но далее мы придадим ему более приятный внешний вид.

Добавление «ложного столбца»

Правое боковое меню визуально недостаточно выделяется. Это можно исправить с помощью темного цвета фона и некоторого форматирования текста.

1. Отредактируйте стиль #secondary, созданный ранее, добавив темный цвет фона. Завершенный стиль должен выглядеть так:

```
#secondary {
  float: right;
  width: 180px;
  background-color: #294E56;
}
```

Теперь фон правого бокового меню действительно выделяется, в отличие от текста, который также является темным.

2. Добавьте еще один стиль внизу внутренней таблицы стилей, чтобы сделать весь текст в этом боковом меню белым:

```
#secondary * {
  color: #FFF;
}
```

Этот стиль использует преимущества универсального селектора (см. подраздел «Универсальный селектор» разд. «Стилизация групп тегов» гл. 3). Он, по существу, дает указание «установить белый цвет текста для каждого тега внутри #secondary». Это краткий способ создания; если же указывать стиль для каждого элемента, то селекторов будет намного больше: #secondary h1, #secondary h2, #secondary p, #secondary a и т. д.

Теперь нужно создать несколько стилей, чтобы отрегулировать размер шрифта, поля и другие свойства его отображения.

3. Добавьте следующие стили во внутреннюю таблицу стилей:

```
#secondary h3 {
  font-size: 1.5em;
  background: #73AFB7;
  padding: 3px 5px 3px 10px;
}
#secondary h4 {
  font-size: 1.2em;
  margin: 10px 10px 5px 10px;
}
#secondary p {
  font-size: 1.2em;
  margin: 3px 10px 10px 10px;
  line-height: 110%;
}
```

Каждый из этих стилей регулирует размер шрифта для различных текстовых тегов, используемых в боковом меню. Кроме того, добавляется цвет фона к за-

головкам, которые определяют каждый раздел в боковом меню. Если вы теперь предварительно просмотрите страницу в браузере, она будет выглядеть так, как на рис. 12.17.



Рис. 12.17. У правого бокового меню темный фон, который выглядел бы лучше, если бы простирался до конца страницы. Однако поскольку в среднем разделе страницы больше содержимого, фон правого бокового меню резко обрывается *(обведено кружком)*. Решение — использование фонового изображения

ПРИМЕЧАНИЕ

В стиле #secondary, который определяет разметку этого бокового меню, нет отступов. Однако текст не наталкивается прямо на края бокового меню, потому что другие стили добавляют пространство между меню и текстом внутри его. В частности, отступ в стиле h3 и поля в стилях h4 и p добавляют необходимое место, что приносит двойную пользу. Без отступа в боковом меню вы можете сделать так, чтобы цвет фона основных заголовков в боковом меню («In the News» и «Around the Web») распределялся по всей ширине меню.

4. В боковом меню есть еще одна проблема: его фоновый цвет внезапно заканчивается, не доходя до основания страницы. Все будет выглядеть намного лучше, если этот темный цвет будет простираться до правой стороны окна.

Фоновому цвету правого бокового меню нужна помощь. Если хотите продлить цвет по всей высоте страницы, вы должны поместить изображение непосредственно в фон самой страницы и повторять его вертикально, чтобы оно оставалось видимым независимо от высоты страницы.

5. Добавьте тег стиля body вверху внутренней таблицы стилей:

```
body {
  background: url(images/bg/bg_column.gif) repeat-y right top;
}
```

Файл bg_column.gif содержит простое одноцветное изображение такой же ширины, что и правое боковое меню. Свойство repeat-y повторяет рисунок вверх и вниз, а значение right помещает его в правую часть страницы.

Установка ширины

В настоящее время у страницы свободный дизайн (см. разд. «Типы разметок веб-страницы» гл. 11), означающий, что она расширяется, чтобы заполнить всю ширину окна браузера. Но допустим, вам больше хочется, чтобы страницы все время оставались одной и той же ширины, поэтому вам не нравится, как они выглядят на широкоформатных мониторах или что случается с дизайном, когда окно браузера уменьшается до очень маленьких размеров. Изменить свободный дизайн на дизайн с фиксированной шириной легко. Начните с добавления HTML-кода.

1. Сразу после открывающего тега <body> (<body id = "feature">) добавьте новый тег :

```
<div id="wrapper">
```

Вы «упаковываете» всю страницу в блок, который будете использовать для управления шириной страницы. Вы должны убедиться, что этот тег закрыт.

2. Добавьте закрывающий тег прямо перед закрывающим тегом </body>:

```
</div>
</body>
```

Теперь, когда созданный блок включает в себя все содержимое страницы, вы можете управлять ее шириной, устанавливая ширину для данного тега.

3. Сразу под тегом стиля body, который вы создали ранее, добавьте стиль, который устанавливает ширину для нового блока:

```
#wrapper {
  width: 760px;
}
```

Предварительно просмотрите страницу в браузере, и вы увидите, что баннер, нижний колонтитул и другое содержимое на странице зажаты в блоке шириной 760 пикселов. Однако изображение, которое добавляет фон правому боковому меню, свободно перемещается в зависимости от ширины окна браузера. Так происходит потому, что оно выравнивается относительно правого края окна. Чтобы устранить это, просто определите изображение как фон в стиле #wrapper.

4. Удалите стиль body, который вы создали в шаге 4 предыдущего подраздела. Добавьте объявление фона к стилю #wrapper, чтобы он выглядел таким образом:

```
#wrapper {
  width: 760px;
  background: url(images/bg/bg_column.gif) repeat-y right top;
}
```

Предварительно просмотрите страницу в браузере. Она теперь должна выглядеть так, как показано на рис. 12.18.



Рис. 12.18. Превращение свободного дизайна в дизайн с фиксированной шириной — лишь вопрос «упаковки» всего HTML-кода тега <body> в новый тег , а затем создания стиля, который устанавливает ширину для этой «упаковки»

Окончательная версия этой программы-примера находится в папке 12_finished\layout1.

Обучающий урок 2: разметка с отрицательными полями

В этом уроке вы узнаете, как создавать плавающие разметки с множеством столбцов, используя методику отрицательных полей, рассмотренную в подразделе «Использование отрицательных полей для размещения элементов» разд. «Использование плавающих элементов в разметках» текущей главы. Файлы для этого примера хранятся в папке 12\layout2.

Центрирование разметки

В отличие от последнего примера, весь HTML-код для этой страницы уже вставлен. Все, что вам остается сделать, — добавить CSS-код, чтобы создать разметку. На странице есть шесть главных разделов, каждый из которых заключен в тег `<div>` с примененным к нему ID.

1. В текстовом редакторе откройте файл `start.html` из папки 12\layout2.

   Рисунок 12.19 показывает основную структуру HTML-кода *(слева)* и заключительный результат, к которому вы должны стремиться *(справа)*. В настоящее время разделы просто сложены один на другой, потому что вы еще не добавили CSS-разметку.

   Начните с создания разметки с фиксированной шириной и расположите ее посередине окна браузера.

2. В области `<head>` в HTML-коде поместите курсор между открывающим и закрывающим тегами `<style>`.

   К странице прилагается внешняя таблица стилей, выполняющая бóльшую часть всего визуального форматирования. Вы добавите стили разметки во внутреннюю таблицу стилей, которую при желании позже сможете переместить во внешнюю таблицу стилей, как описано в подразделе «Прикрепление таблиц стилей с использованием CSS» разд. «Внешние таблицы стилей» гл. 2.

3. Добавьте новый стиль для раздела `wrapper`:

```
#wrapper {
  border-right: 2px solid #000000;
  border-left: 2px solid #000000;
  background: #FFFFFF url(images/column_bg.png) repeat-y right top;
}
```

   Эти свойства добавляют границу к каждой стороне страницы и помещают фоновое изображение в правую часть. Изображение повторяется вертикально вниз и обеспечивает фон для правого бокового меню. Эта настройка — пример использования методики «ложного столбца», описанной в подразделе «Создание столбцов на всю высоту» разд. «Преодоление проблем перемещения» текущей главы.

   Поскольку обертывающий тег `<div>` охватывает и другие теги на странице, установка его ширины определяет ширину для всей страницы.

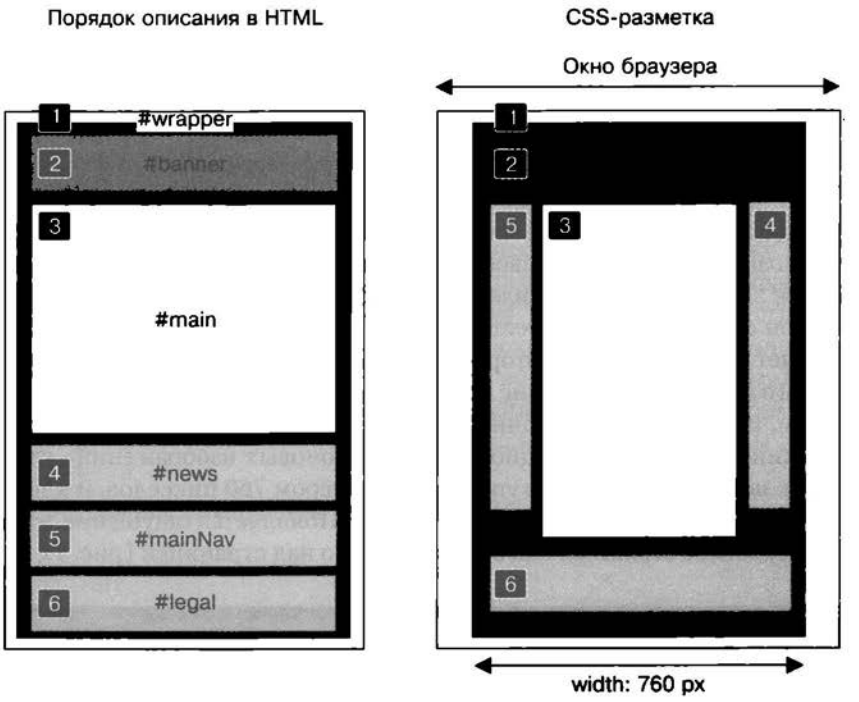


Рис. 12.19. Схемы показывают страницу в начале *(слева)* и в конце *(справа)*. Сначала она является лишь набором тегов , заключенных в один тег . Устанавливая ширину для «упаковки» и задавая ее левое и правое поля, вы создаете разметку, центрированную в окне браузера

4. Отредактируйте стиль #wrapper, который вы только что создали. Добавьте ширину, а также левое и правое поля, чтобы стиль выглядел следующим образом:

```
#wrapper {
  width: 760px;
  margin-right: auto;
  margin-left: auto;
  border-right: 2px solid #000000;
  border-left: 2px solid #000000;
  background: #FFFFFF url(images/column_bg.png) repeat-y right top;
}
```

Свойство width устанавливает полную ширину содержимого страницы размером 760 пикселов. Чтобы центрировать «упаковку» div, установите для левого и правого полей значение auto. Оно говорит браузеру, что он должен вычислять поля автоматически. Кроме того, поскольку значение применено и к левому, и к правому полям, браузер должен распределить ширину между ними. Другими словами, браузер добавляет одинаковое пространство с обеих сторон «упаковки», невзирая на ширину окна.

Теперь предварительно просмотрите страницу в браузере. Измените размер окна браузера, и вы увидите, что страница остается выровненной посередине окна. Но вы все еще можете улучшить ее внешний вид.

5. В начале внутренней таблицы стилей добавьте новый стиль для тега <body>:

```
body {
  margin: 0;
  padding: 0;
  background: #E6E6E6 url(images/page_bg.png) repeat-y center top;
}
```

Первые два свойства удаляют любое свободное место вокруг содержимого страницы, позволяя «упаковке» свободно касаться краев окна браузера. Самое интересное заключается в объявлении свойства background. Во-первых, оно изменяет фон страницы на неестественный серый (#E6E6E6). Во-вторых, свойство добавляет изображение и повторяет его вертикально от верхнего края страницы до самого основания. Значение center помещает рисунок в центре окна. Таким образом, независимо от ширины окна изображение останется выровненным посередине (см. разд. «Позиционирование фоновых изображений» гл. 8). Сам рисунок немного шире, чем «упаковка» размером 760 пикселов, и к нему применен эффект тени с левой и правой сторон. Появляется ощущение, что основное содержимое страницы как бы приподнято над страницей (рис. 12.20).



Рис. 12.20. Центрируя изображение в фоне тега <body>, вы создаете эффект тени под содержимым страницы

Перемещение столбцов

Теперь пришло время создать для этого дизайна три столбца. На левой схеме на рис. 12.19 видно, что HTML-код для раздела `main`, где находится главное содержимое страницы, появляется перед HTML-кодом либо для левого бокового меню с ID, равным `mainNav`, либо для правого бокового меню с ID, равным `news`. Несколько факторов делают этот дизайн отличным от базовой разметки с плавающими элементами, которую вы создавали в предыдущем уроке. Поскольку HTML-код для главного раздела (`main`) идет первым, вы должны сделать его плавающим, чтобы боковые меню обтекали его с любой стороны. Раздел `main` на веб-странице должен располагаться посередине, между двумя боковыми меню. Обычно такое расположение невозможно. Но мы будем с умом использовать отрицательные поля (см. подраздел «Плавающие элементы внутри плавающих элементов» разд. «Использование плавающих элементов в разметках» этой главы), чтобы достичь поставленной цели.

1. Добавьте стиль к внутренней таблице стилей для раздела `main`:

```
#main {
  float: left;
  width: 419px;
  padding-left: 10px;
  border-left: 1px dashed #999999;
}
```

Пока этот стиль довольно простой. Он размещает раздел в левой части страницы, устанавливает его ширину, добавляет левую границу и небольшой промежуток между текстом и границей. Затем нужно позиционировать раздел `news`.

2. Добавьте следующий стиль, чтобы разместить правое боковое меню:

```
#news {
  float: right;
  width: 160px;
}
```

Наконец, вы должны расположить навигационное меню.

3. Добавьте еще один стиль во внутреннюю таблицу стилей:

```
#nav {
  float: left;
  width: 160px;
}
```

Страница должна выглядеть так, как на рис. 12.21. Видно, что есть три столбца, расположенные бок о бок, но в неправильном порядке. Большой столбец слева должен находиться посередине.

Именно сейчас следует присоединить отрицательные поля. Основной процесс идет таким образом: сначала необходимо добавить левое поле достаточного размера к главному столбцу, чтобы установить его в нужное место — посередине страницы. Затем вы должны добавить отрицательное поле к навигационному меню, чтобы отодвинуть его в левую сторону от главного содержимого.

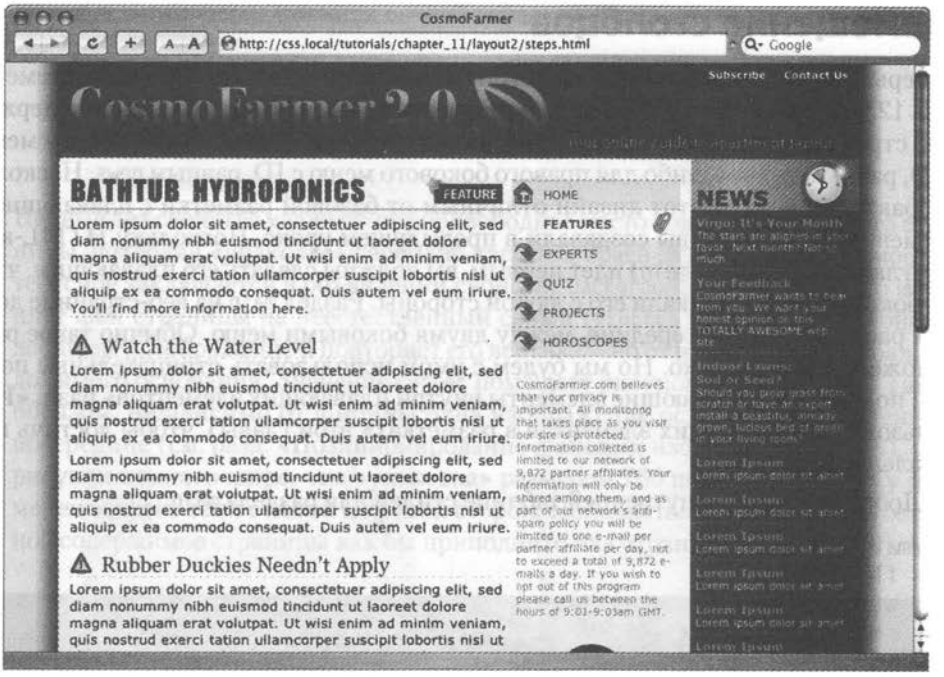


Рис. 12.21. Обычно размещение плавающих элементов полностью зависит от их порядка в HTML-коде. Главное содержимое *(слева)* идет первым, поэтому оно не появляется между двумя боковыми меню. Но вы можете изменить это, используя отрицательные поля

4. В стиле #main добавьте свойство margin-left: 160px:

```
#main {
  float: left;
  width: 419px;
  margin-left: 160px;
  padding-left: 10px;
  border-left: 1px dashed #999999;
}
```

Возвращаясь к шагу 3, мы видим, что навигационная панель (которая, как предполагается, будет крайним левым столбцом) имеет ширину 160 пикселов, столько же вы использовали для левого поля. Отодвигая раздел main (главную область) на 160 пикселов, вы освобождаете место для навигационной панели.

5. Добавьте отрицательное левое поле к стилю #nav следующим образом:

```
#nav {
  float: left;
  width: 160px;
  margin-left: -590px;
}
```

Почему выбрано такое значение, как –590 пикселов? В настоящий момент разделы nav и main выровнены по левому краю. Однако, поскольку HTML-код для

навигационной панели идет после кода главного содержимого, она может быть перемещена влево настолько, насколько будет смещен вправо раздел `main`. Чтобы навигационное меню достигло левого края страницы, оно должно полностью переместиться от правого края раздела `main` к левому краю страницы.

Ширина до правой части раздела `main` представляет собой сумму значений его ширины, левого поля, отступа и границы. Таким образом, она составляет: 160 (левое поле) + 10 (левый отступ) + 1 (левая граница) + 419 (ширина) = 590 пикселов. Так что, задавая навигационной панели левое поле шириной –590 пикселов, вы переместите ее за главный раздел — на его предыдущее место.

Просмотрите страницу в браузере, и вы увидите нечто похожее на то, что показано на рис. 12.22. Этот способ работает, если вы используете Firefox или Safari. Но с Internet Explorer 6 другая история. Слева от среднего столбца вы не увидите никакой навигационной панели — лишь пустое место. Причиной тому является ошибка двойного поля (см. разд. «Обработка ошибок в Internet Explorer 6» этой главы). Поскольку раздел `main` перемещен влево и у него есть левое поле, Internet Explorer 6 удвоит этот край, разрушая разметку. К счастью, эту ошибку можно легко устранить.

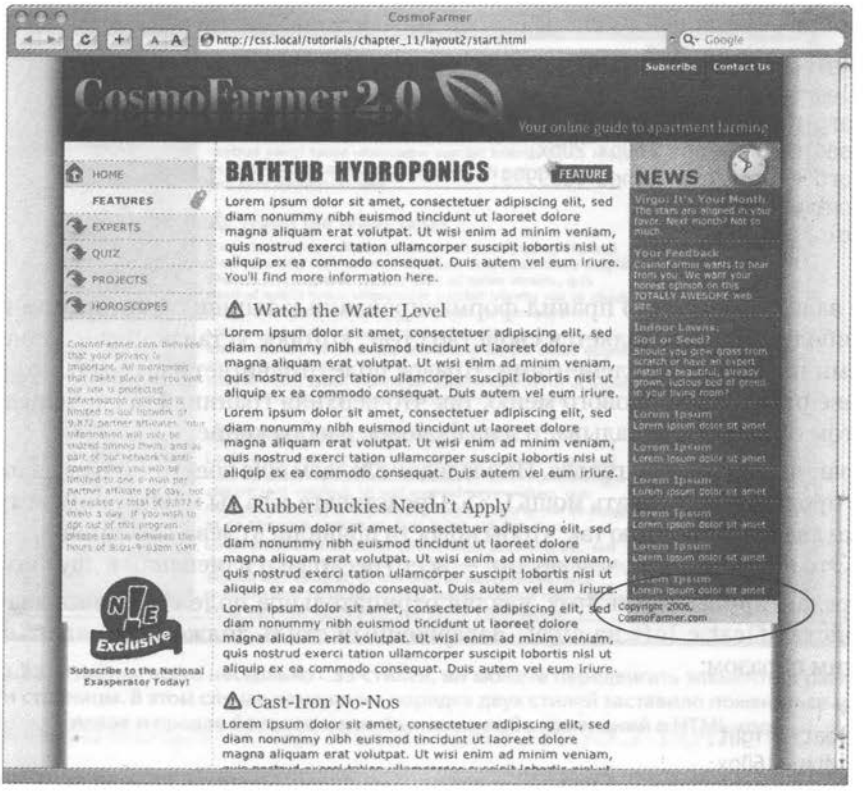


Рис. 12.22. Используя отрицательные поля, вы можете разместить столбцы в любом порядке на странице

6. Добавьте свойство `display: inline` к стилю `#main`:

```
#main {
  display: inline;
  float: left;
  width: 419px;
  margin-left: 160px;
  padding-left: 10px;
  border-left: 1px dashed #999999;
}
```

Теперь, если вы просмотрите страницу в Internet Explorer 6, ошибки не будет (см. рис. 12.22).

Заключительная настройка

Следующие несколько шагов продемонстрируют, насколько гибкой на самом деле является разметка CSS. Сначала мы позаботимся о записи об авторском праве, которая должна появиться у основания страницы. В настоящее время она «захвачена» всеми плавающими элементами (см. рис. 12.22). Запись нуждается в отмене перемещения, чтобы мы могли разместить ее в положенном месте.

1. В конце внутренней таблицы стилей добавьте последний стиль:

```
#legal {
  clear: both;
  margin-right: 160px;
  padding: 5px 5px 160px 20px;
  border-top: 1px dashed #999999;
  font-weight: bold;
  color: #666666;
}
```

Он задает множество правил форматирования к записи об авторском праве. Наиболее важным является свойство `clear`, которое устанавливает запись под всеми плавающими элементами. Правое поле отодвигает запись об авторском праве от правого бокового меню, так что верхняя граница (также определена в этом стиле) не накладывается на фоновое изображение.

Страница в основном готова. Напоследок сделаем еще кое-что, чтобы в который раз продемонстрировать мощь CSS. Представьте, что вы можете поменять местами два боковых меню так, чтобы новости появились слева, а навигация — справа. Это намного легче, чем звучит, — всего несколько изменений в двух стилях.

2. Отредактируйте стиль `#nav`, удаляя отрицательное поле и изменяя значение свойства `float` с `left` на `right`. Завершенный стиль должен выглядеть следующим образом:

```
#nav {
  float: right;
  width: 160px;
}
```

Навигационная панель перемещается в правую сторону. Теперь отправим новости влево.

3. В стиле #news добавьте свойство `margin-left: -590px` и установите для `float` значение `left`:

```
#news {
  float: left;
  width: 160px;
  margin-left: -590px;
}
```

Сохраните страницу и просмотрите ее в браузере (рис. 12.23). Столбцы поменялись местами без каких-либо серьезных изменений в HTML-коде. Все, что вы сделали, — поменяли два стиля. Если вы хотите зайти еще дальше, можете переместить фоновое изображение, которое относится к «упаковке», влево и определить границу, которая появляется в левой части главного содержимого, в его правой части так, чтобы она проходила вдоль навигационного меню.

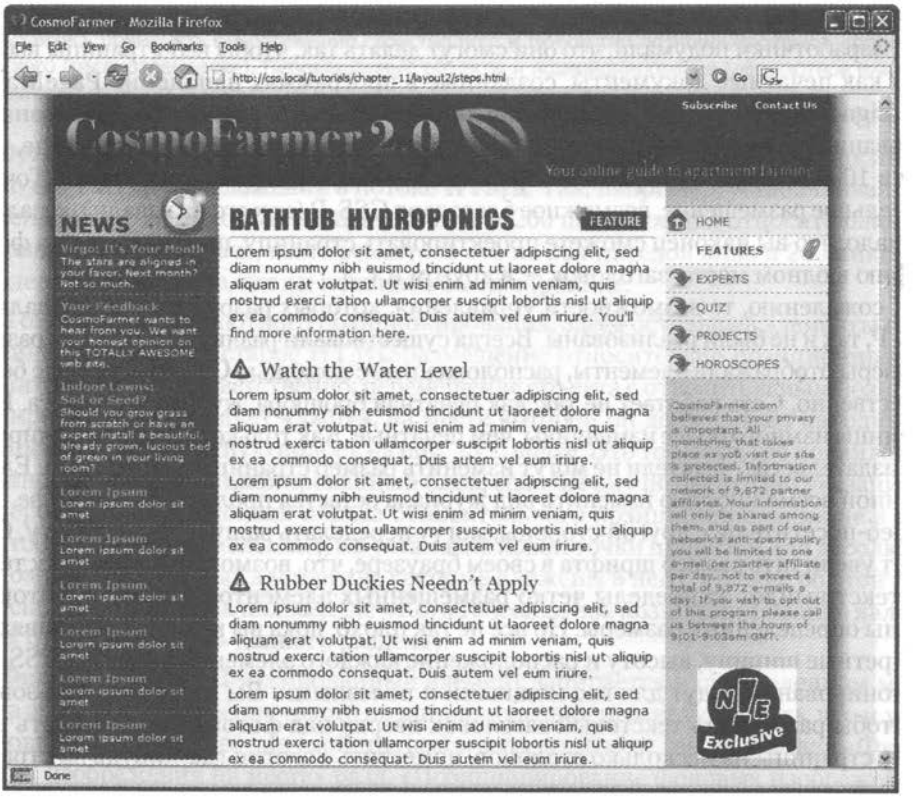


Рис. 12.23. Изменяя всего несколько CSS-стилей, вы можете передвигать элементы в различные области страницы. В этом случае изменение порядка двух стилей заставило поменяться местами левое и правое боковое меню без каких-либо изменений в HTML-коде

Конечная версия этого примера находится в папке 12_finished\layout2.

13 Позиционирование элементов на веб-странице

Когда Консорциум Всемирной паутины представил *CSS-позиционирование*, некоторые разработчики подумали, что они смогут делать так, чтобы веб-страницы выглядели как печатные документы, созданные в программах наподобие PageMaker, InDesign или Quark XPress. Благодаря лишь нескольким свойствам CSS-позиционирование позволяет поместить элемент в определенном месте на странице, скажем в 100 пикселах от вершины страницы и 200 пикселах от левого края. Точное пиксельное размещение, возможное благодаря CSS-P (как его называли), казалось, обещало, что вы наконец сможете проектировать страницу, просто помещая фотографию в одном месте, заголовок — в другом и т. д.

К сожалению, те возможности управления, которых разработчики ожидали от CSS-P, так и не были реализованы. Всегда существовали различия в том, как разные браузеры отображали элементы, расположенные с помощью CSS. Но, что еще более существенно, Сеть работает не так, как печатная брошюра, журнал или книга. Веб-страницы намного более изменчивы, чем печатные. Как только журнал тиражируется в издательстве, читатели не могут изменить размер страницы или шрифта. Единственной возможностью изменить вид журнала остается пролить на него кофе.

Веб-посетители, с другой стороны, могут переделать вашу ручную работу. Они могут увеличить размер шрифта в своем браузере, что, возможно, приведет к тому, что текст выйдет за пределы четко размещенных элементов разметки, которым заданы определенные размеры. Но все не так плохо: пока вы не пытаетесь навязать конкретные ширину, высоту и расположение каждого элемента, свойства CSS-позиционирования будут для вас мощными и полезными. Вы можете использовать их, чтобы разместить текстовый заголовок вверху над фотографией, создать разметку страницы на несколько столбцов, поместить логотип где-нибудь на странице и т. д.

Как работают свойства позиционирования

CSS-свойство `position` позволяет управлять тем, где и как браузер отобразит определенные элементы. Используя его, вы можете, например, поместить боковое меню на странице там, где вы этого желаете, или убедиться, что навигационная панель

наверху страницы останется там, даже когда пользователи прокручивают страницу вниз. CSS предлагает четыре типа позиционирования.

- **Абсолютное.** Такое позиционирование позволяет определять расположение элемента, задавая позиции `left`, `right`, `top` или `bottom` в пикселах, em или процентах (для получения дополнительной информации о выборе между различными единицами измерения см. гл. 6). Вы можете поместить блок на расстоянии 20 пикселов от верхнего и 200 пикселов от левого края страницы, как показано на рис. 13.1, *посередине* (далее вы узнаете, как писать код с этими инструкциями).

  Кроме того, абсолютно размещенные элементы полностью отделены от потока страницы, определенного HTML-кодом. Другими словами, остальные элементы на странице не знают, что существует абсолютно позиционированный элемент. Они могут даже полностью исчезнуть, попав под него, если вы будете невнимательны.

ПРИМЕЧАНИЕ

Не пытайтесь применять одновременно свойство float и любой тип позиционирования, кроме статического (рассмотрен ниже). Перемещаемые объекты и позиционирование (абсолютное или фиксированное) не могут применяться к одному и тому же элементу.

- **Относительное.** Элемент с таким позиционированием размещается относительно его текущего положения в потоке HTML. Так, например, устанавливая значение `top` — 20 пикселов и значение `left` — 200 пикселов для относительно размещенного заголовка, вы переместите его на 20 пикселов вниз и 200 пикселов влево от того места, где он появился бы на странице.

  В отличие от абсолютного позиционирования, здесь остальные элементы страницы регулируют старое HTML-размещение относительно позиционированного объекта. Соответственно, перемещение объекта с относительным позиционированием оставляет «дыру», на месте которой он должен был находиться. Посмотрите на темную полосу на рис. 13.1, *внизу*. Это то место, где появился бы относительно позиционированный блок, если бы ему не было дано указание на перемещение. Основная польза относительного позиционирования не в том, чтобы переместить элемент, а в установке новой точки привязки для абсолютно позиционированных элементов, которые вложены в него (больше об этой концепции вы узнаете в подразделе «Когда абсолютное позиционирование является относительным» данного раздела).

- **Фиксированное.** Фиксированный элемент запирается в определенном месте на экране. Определение такого позиционирования играет ту же роль, что и установка значения `fixed` для свойства `background-attachment` (см. подраздел «Фиксация изображения на месте» разд. «Позиционирование фоновых изображений» гл. 8). Когда посетитель прокручивает страницу, фиксированные элементы остаются на экране как абзацы и заголовки, в то время как фотографии прокручиваются вместе со страницей.

  Использование фиксированных элементов — отличный способ создать неподвижное боковое меню или воспроизвести эффект HTML-фреймов, где прокручивается только определенная часть страницы. Вы скоро узнаете, как создается такой эффект.

ПРИМЕЧАНИЕ

Internet Explorer 6 и более ранние версии не поддерживают фиксированное позиционирование (значение fixed) и игнорируют его.

- **Статическое** позиционирование просто означает, что содержимое соответствует нормальному нисходящему потоку HTML (см. рис. 13.1, *вверху*). Но зачем вам назначать элементу статическое позиционирование? Скорее всего, вы никогда не будете делать этого.

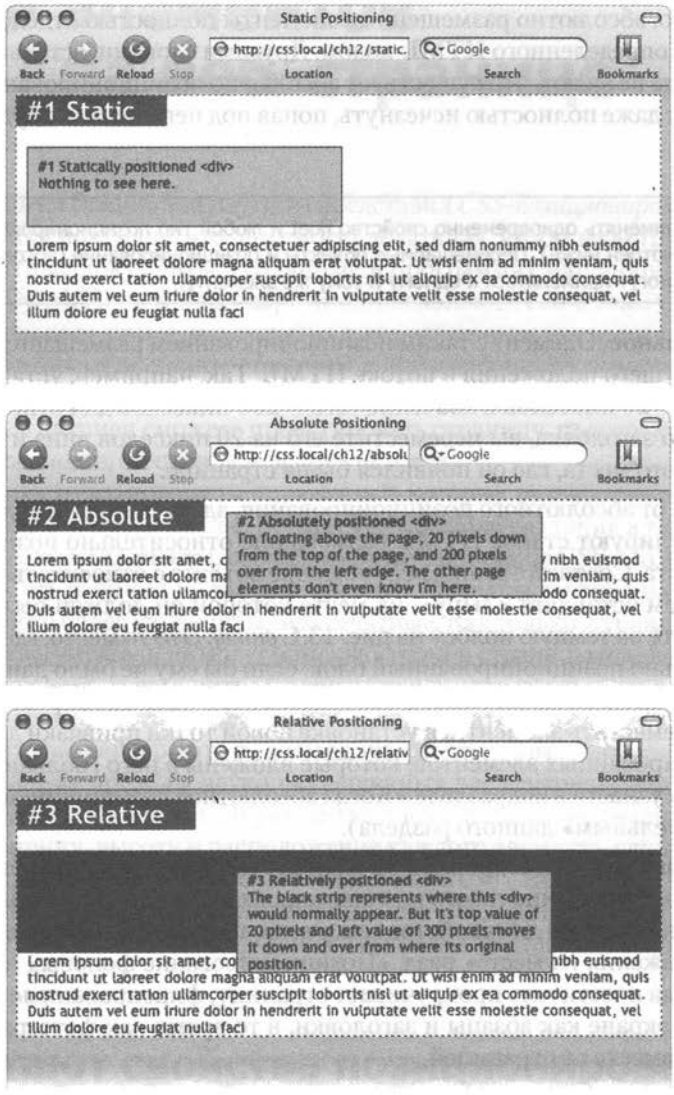


Рис. 13.1. CSS предлагает несколько путей воздействия на размещение элементов на веб-странице

Итак, можно сделать выводы. Статическое размещение — это то, как браузеры представляли содержимое с начала существования Сети. Они просто показывали HTML в нисходящем порядке (сверху вниз). Абсолютное размещение удаляет элемент из потока страницы, определяя его на самый верх, иногда перекрывая какое-нибудь другое содержимое. Относительное позиционирование размещает элемент относительно того места, где он появился бы на странице, и оставляет «дыру» там, где этот элемент находился бы без относительного позиционирования.

Чтобы изменить расположение любого элемента, просто используйте свойство `position`, которому присваивается одно из этих четырех значений: `static`, `absolute`, `relative`, `fixed`. Чтобы создать абсолютно позиционированный элемент, добавьте это свойство к стилю:

```
position: absolute;
```

Статическое позиционирование — это обычный метод расположения элементов, так что, если вы только не отменяете ранее созданный стиль, в котором уже указано расположение `absolute`, `relative` или `fixed`, вам не нужно определять это позиционирование. Кроме того, статические элементы не подчиняются ни одному из значений позиционирования, обсуждаемых далее.

Установка значений позиционирования — это обычно только часть сражения. Чтобы на самом деле определить элемент где-нибудь на странице, вы должны разобраться с различными свойствами позиционирования.

Установка значений расположения

Область отображения браузера, также называемая *областью просмотра*, имеет верхний, нижний, правый и левый края. Для каждого из этих четырех краев есть соответствующее CSS-свойство: `top`, `bottom`, `left` и `right`. Вам не нужно задавать значения для всех четырех сторон. Обычно достаточно двух, чтобы определить элемент на странице. Вы можете, если хотите, поместить элемент на расстоянии 10 em от левого края страницы и 20 em от вершины.

ПРИМЕЧАНИЕ

Internet Explorer 6 иногда неверно устанавливает элементы, которые позиционируются с помощью свойств bottom и right (см. врезку «Ошибки браузеров» далее).

Чтобы определить расстояние от края страницы до соответствующей стороны элемента, используйте любую из действующих в CSS единиц измерения: пикселы, em, проценты и т. д. При позиционировании можно также использовать отрицательные значения, например `left: -10px;`, чтобы переместить элемент частично за страницу (или от другого элемента). Это приведет к появлению особого визуального эффекта, который будет описан в подразделе «Исключение элемента за пределы блока» разд. «Мощные стратегии позиционирования» этой главы.

После свойства `position` вы указываете два свойства (`top`, `bottom`, `left` или `right`). Если вы хотите, чтобы элемент принял ширину, меньше допустимой (например, чтобы сделать тонкое боковое меню), то можете установить свойство `width`. Чтобы поместить баннер страницы в определенном месте относительно верхнего и левого краев окна, создайте следующий стиль:

```
#banner {
  position: absolute;
  left: 100px;
  top: 50px;
  width: 760px;
}
```

Этот стиль поместит баннер, как показано на рис. 13.2, *вверху*.

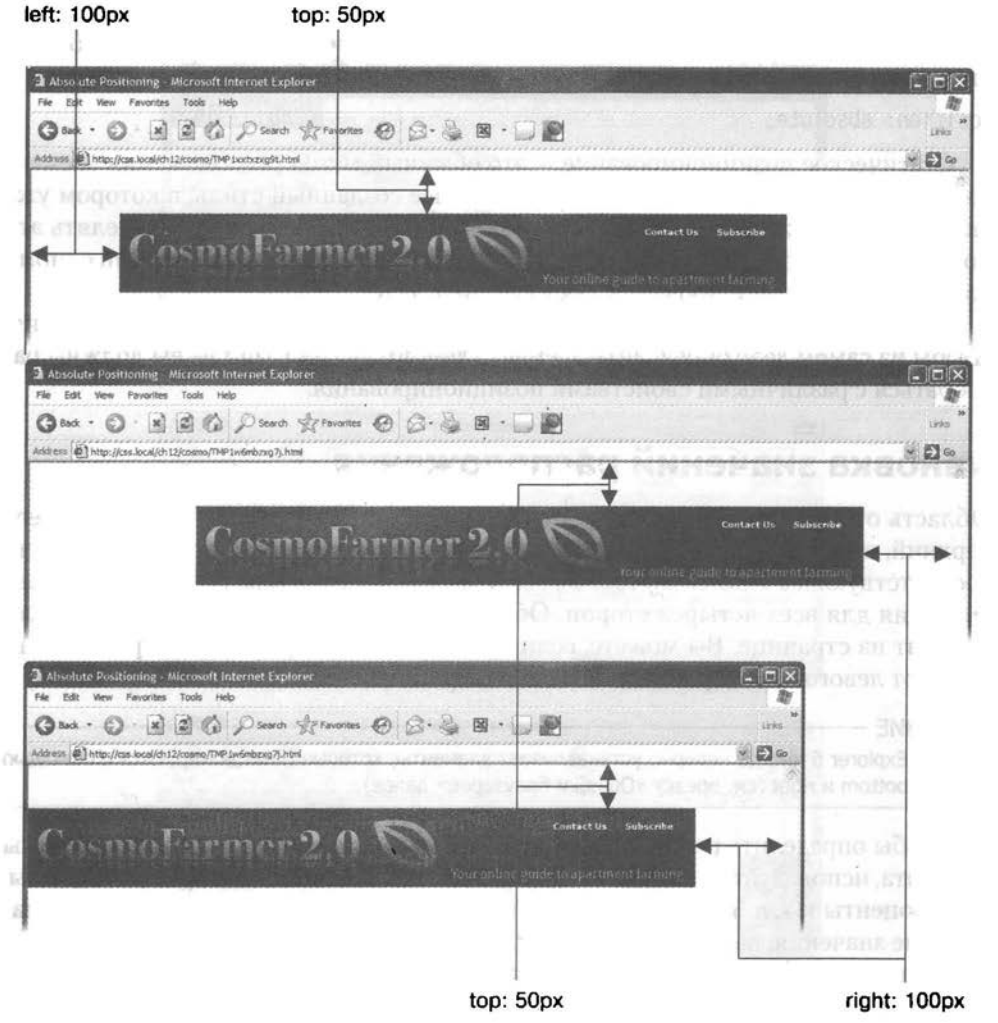


Рис. 13.2. Польза абсолютного расположения состоит в возможности поместить элемент относительно правого края окна браузера *(посередине)*. Если ширина окна изменится, расстояние от правого края окна до правого края элемента останется неизменным *(внизу)*

Рассмотрим другой пример: поместим элемент таким образом, чтобы он всегда оставался на фиксированном расстоянии от правой стороны браузера. Когда вы

используете свойство right, браузер измеряет расстояние от правого края окна до правого края элемента (см. рис. 13.2, *посередине*). Чтобы поместить баннер на расстоянии 100 пикселов от правого края окна, вы создадите тот же самый стиль, что и ранее, но просто измените left на right:

```
#banner {
  position: absolute;
  right: 100px;
  top: 50px;
  width: 760px;
}
```

Поскольку рассчитываемая позиция основывается на правой стороне окна, регулирование его размера автоматически меняет расположение баннера, что вы можете видеть на рис. 13.2, *внизу*. Хоть баннер и перемещается, расстояние от правой стороны элемента до правого края окна браузера остается неизменным.

Формально вы можете указать свойства для левой и правой позиций, а также для верхней и нижней и позволить браузеру определить ширину и высоту элемента. Скажем, вы хотите, чтобы центральный блок текста размещался на расстоянии 100 пикселов от вершины окна и 100 пикселов от левой и правой его сторон. Чтобы определить блок, вы можете использовать стиль абсолютного позиционирования, который установит свойствам top, left и right значение 100 пикселов. В окне браузера левый край блока начинается на расстоянии 100 пикселов от левой стороны окна, а правый край находится на расстоянии 100 пикселов от правой стороны (рис. 13.3, *вверху*). Точная ширина блока при этом зависит от ширины окна браузера. Более широкое окно увеличит блок; чем тоньше окно, тем меньше будет блок. Левая и правая позиции, однако, остаются теми же.

К сожалению, Internet Explorer версии 6 и ниже не понимает это должным образом (см. рис. 13.3, *внизу*). Браузер показывает левую позицию корректно, но просто игнорирует любые правые значения. Таким образом, пока Internet Explorer 6 нет рядом, лучше придерживайтесь значений left или right и используйте свойства width для определения ширины абсолютно позиционированного элемента.

Свойства width и height, о которых вы узнали в гл. 7, аналогично работают для позиционированных элементов. Чтобы поместить серый блок размером 50 × 50 пикселов в верхний правый угол окна браузера, создайте такой стиль:

```
.box {
  position: absolute;
  right: 0;
  top: 0;
  width: 50px;
  height: 50px;
  background-color: #333;
}
```

Здесь надо вспомнить предостережение, которое давалось в разд. «Определение параметров высоты и ширины» гл. 7: будьте внимательны с установкой высоты для элементов. Если вы не разрабатываете какую-нибудь графику с уже заданной

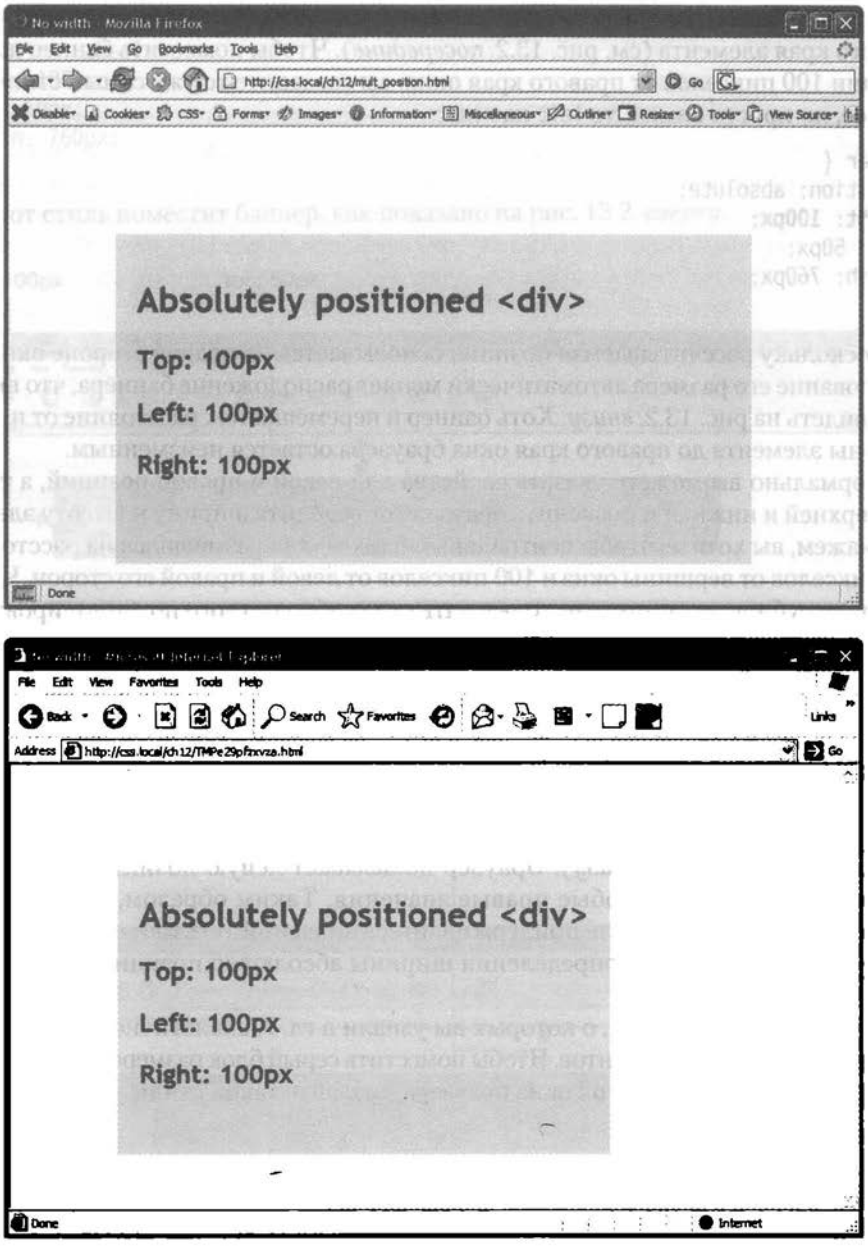


Рис. 13.3. Получив только левую и правую позиции, Firefox правильно вычисляет ширину серого блока *(вверху)*. Internet Explorer 6, однако, игнорирует значение right и устанавливает ширину блока на основании ширины его содержимого

высотой, то не можете знать, насколько высоким будет на странице конкретный элемент. Вы могли бы определить для бокового меню высоту 200 пикселов, но если вы заканчиваете его описание добавлением слов или картинок, которые сделают

боковое меню больше чем 200 пикселов в высоту, то в конечном счете содержимое меню выйдет за его пределы. Даже если вы уверены, что содержимое поместится, посетитель всегда может увеличить размер шрифта своего браузера, сделав текст достаточно большим, чтобы он мог «выпасть» из блока.

К тому же, когда вы определяете ширину и высоту в стиле, а содержимое внутри разрабатываемого элемента оказывается шире или выше, могут произойти странные вещи (см. подраздел «Управление поведением блочных элементов с помощью свойства overflow» разд. «Определение параметров высоты и ширины» гл. 7, где рассказывается, как использовать CSS-свойство `overflow`).

Когда абсолютное позиционирование является относительным

Ранее в этой главе мы говорили о расположении элемента в конкретном месте в окне браузера. Однако абсолютное позиционирование не всегда работает таким образом.

На самом деле абсолютно позиционированный элемент помещается относительно границ его ближайшего предка. Проще говоря, если вы уже создали элемент с абсолютным расположением (скажем, тег `<div>`, который появится на расстоянии 100 пикселов от вершины окна браузера), то любые абсолютно позиционированные элементы с HTML-кодом внутри этого тега `<div>` размещаются относительно верхнего, нижнего, левого и правого краев области тега.

ПРИМЕЧАНИЕ

Если вы не помните, что такое родительские элементы и предки, перейдите к подразделу «Дерево HTML» разд. «Стилизация вложенных тегов» гл. 3.

В верхнем изображении на рис. 13.4 светло-серый блок абсолютно позиционирован на расстоянии 5 em от верхнего и левого краев окна браузера.

Есть также тег `<div>`, вложенный в этот блок. Применение абсолютного позиционирования для этого тега позволит поместить его относительно абсолютно позиционированного родительского элемента. Установка свойству `bottom` значения 0 переместит блок не к основанию экрана, а к основанию его предка. Аналогично свойство `right` для этого вложенного тега `<div>` относится к правому краю его родителя (см. рис. 13.4, *внизу*).

Каждый раз, когда вы используете абсолютное позиционирование, чтобы установить элемент на странице, его точное место зависит от расположения любых других тегов, в которые вложен разрабатываемый элемент. Позиционирование подчиняется следующим правилам:

- тег расположен относительно окна браузера, если у него абсолютное позиционирование и он не находится внутри любого другого тега, к которому применено абсолютное, относительное или фиксированное позиционирование;
- тег определен относительно сторон другого элемента, если он находится внутри другого тега с абсолютным, относительным или фиксированным позиционированием.

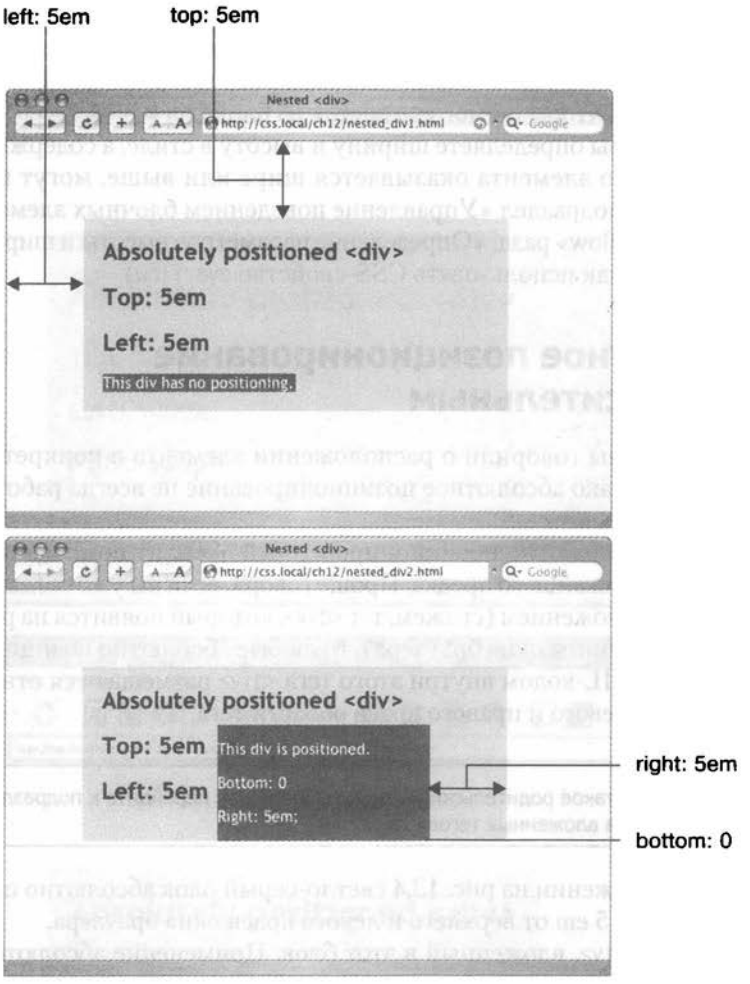


Рис. 13.4. Вы можете разместить элемент относительно окна браузера *(вверху)* или относительно позиционированного предка *(внизу)*

Когда и где использовать относительное позиционирование

Вы получаете существенную выгоду, размещая элемент относительно другого тега: если этот тег передвигается, позиционированный элемент передвигается вместе с ним. Скажем, вы определили изображение внутри тега <h1> и хотите, чтобы оно появилось в правой части заголовка. Если вы просто поместите изображение в конкретном месте в окне браузера в левой части заголовка, то рискуете потерять его из виду. Если заголовок будет перемещен, абсолютно расположенное изображение останется «приклеенным» к назначенному ему месту. Вместо этого лучше определить изображение относительно тега <h1> так, что, когда заголовок будет перемещен, изображение переместится вместе с ним (две веб-страницы внизу рис. 13.5).



Рис. 13.5. Пример использования относительного позиционирования применительно к изображению

ПРИМЕЧАНИЕ

Используйте свойство background-image (см. разд. «Фоновые изображения» гл. 8), чтобы поместить изображение в фон тега <h1>. Однако если изображение будет выше, чем тег <h1>, или вы хотите, чтобы оно появилось за пределами границ заголовка (см. третью сверху веб-страницу на рис. 13.5), то используйте описанную здесь методику относительного позиционирования.

Чтобы установить изображение на странице, вы могли использовать значение `relative` свойства `position`, но здесь также есть недостатки. Когда вы устанавливаете относительное позиционирование для элемента, а затем размещаете его (возможно, используя свойства `left` и `top`), элемент перемещается на определенное расстояние от того места, где он появился бы в потоке HTML. Другими словами, он перемещается относительно своего текущего положения. В результате на его исходном месте остается пустое пространство, где элемент находился бы без вашего позиционирования (см. рис. 13.1, *внизу*).

Рассмотрим представленный рисунок. На верхней веб-странице видно, что круглая графическая кнопка помещена внутри тега `<h1>` (см. рис. 13.5, *вверху*). На второй странице к кнопке применено абсолютное позиционирование: `right: -35px; top: -35px;`, которое перемещает ее за пределы области тега `<h1>` и определяет в верхнем правом углу окна браузера (на самом деле кнопка находится немного за пределами окна благодаря отрицательным значениям позиционирования) (см. рис. 13.5, *вторая страница сверху*).

В коде третьей веб-страницы к тегу `<h1>` добавлено свойство `position: relative`, которое создает новую среду позиционирования для тега `<img>`. Те же самые значения `top` и `right` перемещают тег `<img>` к верхнему правому углу заголовка (см. рис. 13.5, *третья страница сверху*). Когда вы передвигаете заголовок ниже, графика также передвигается (см. рис. 13.5, *внизу*).

Лучший способ использовать относительное расположение состоит в том, чтобы создать новую среду позиционирования для вложенных тегов. Например, тег `<h1>` в примере в начале этого раздела является владельцем тега `<img>`, который находится внутри его. При установке относительного позиционирования тега `<h1>` любое абсолютное позиционирование, которое вы примените к тегу `<img>`, будет определено относительно четырех сторон заголовка `<h1>`, а не окна браузера. Код CSS выглядит следующим образом:

```
h1 { position: relative; }
h1 img {
  position: absolute;
  top: 0;
  right: 0;
}
```

Установка свойствам `top` и `right` изображения значения 0 перемещает его в верхний правый угол заголовка, а не окна браузера.

В CSS термин «относительный» (relative) означает не совсем то же самое, что в реальном мире. Все же, если вы хотите поместить тег `<img>` относительно тега `<h1>`, вашей первой мыслью может быть определение для изображения относительного позиционирования. На самом деле элемент, который вы хотите разместить, — изображение — получает абсолютное позиционирование, в то время как элемент,

относительно которого вы хотите определить изображение, — заголовок — получает значение relative. Представьте, что relative означает «относительно меня». Когда вы применяете относительное позиционирование для тега, это означает, что «все элементы, заданные внутри его, должны размещаться относительно его местоположения».

ПРИМЕЧАНИЕ

Поскольку вы будете часто применять относительное позиционирование просто для задания новой среды расположения вложенных тегов, вам даже не нужно использовать значения top, bottom, left и right. У тега <h1> есть свойство position: relative, но нет значений top, bottom, left и right.

ОШИБКИ БРАУЗЕРОВ

Internet Explorer забывает свое место

Иногда вы будете использовать свойства позиционирования bottom и right, чтобы поместить что-нибудь в нижнем правом углу страницы или в нижнем углу другого элемента. Скажем, вы хотите разместить ссылку Contact Us в нижнем левом углу баннера. Если у стиля баннера абсолютное или относительное позиционирование и вы помещаете ссылку абсолютно, то большинство браузеров разместят ее относительно сторон баннера, как и должны.

Однако это не относится к Internet Explorer версий 6 и ниже. Кажущиеся предельно понятными, свойства bottom и right могут ввести этот браузер в заблуждение. Internet Explorer иногда продолжает использовать нижний и правый края веб-страницы как ссылки, так что в итоге ваш элемент будет расположен значительно ниже или гораздо правее, чем вы ожидали.

Как и большинство других ошибок в этом браузере, она исправлена в седьмой и последующих версиях. Исправление здесь заключается в том, чтобы задать элементу, относительно которого вы хотите поместить какой-либо объект, специальное свойство layout.

Если вы используете свойства bottom или right для абсолютно позиционированного элемента, а Internet Explorer все равно выдает этот элемент в другом месте, перейдите к врезке «Информация для опытных пользователей» в разд. «Обработка ошибок в Internet Explorer 6» гл. 12 и примените одно из решений, описанных там.

Кроме того, примеры решения этой проблемы приводятся повсюду в этой главе.

Наложение элементов

Как вы можете видеть на рис. 13.6, абсолютно позиционированные элементы расположены на переднем плане веб-страницы и могут даже находиться спереди (или сзади) других позиционированных элементов. Такое наложение определяется *индексом позиционного уровня* (z-index). Если вы знакомы с понятием слоев в Photoshop, Fireworks или Adobe InDesign, то знаете, как работает индекс позиционного уровня: он представляет порядок, в котором элементы накладываются на верхнюю поверхность страницы.

Когда вы используете индекс позиционного уровня, чтобы управлять порядком наложения элементов, родительские элементы определяют порядок наложения для своих детей. В двух верхних примерах на рис. 13.6 текстовый блок не позиционирован: он разделяет индекс позиционного уровня непосредственно самой страницы, для которой значение z-index равно 0. Таким образом, кнопки

в этих двух текстовых блоках накладываются поверх страницы. Однако для текстового блока в нижнем изображении задано абсолютное позиционирование с `z-index`, равным `1000`. Тег `<div>`, содержащий текстовый блок, становится отправной точкой для укладки в него изображений. Так что, хоть `z-index` области `<div>` равен `1000`, его вложенные дети (с более низкими значениями `z-index`) появятся сверху, в то время как сам текстовый блок находится над страницей.

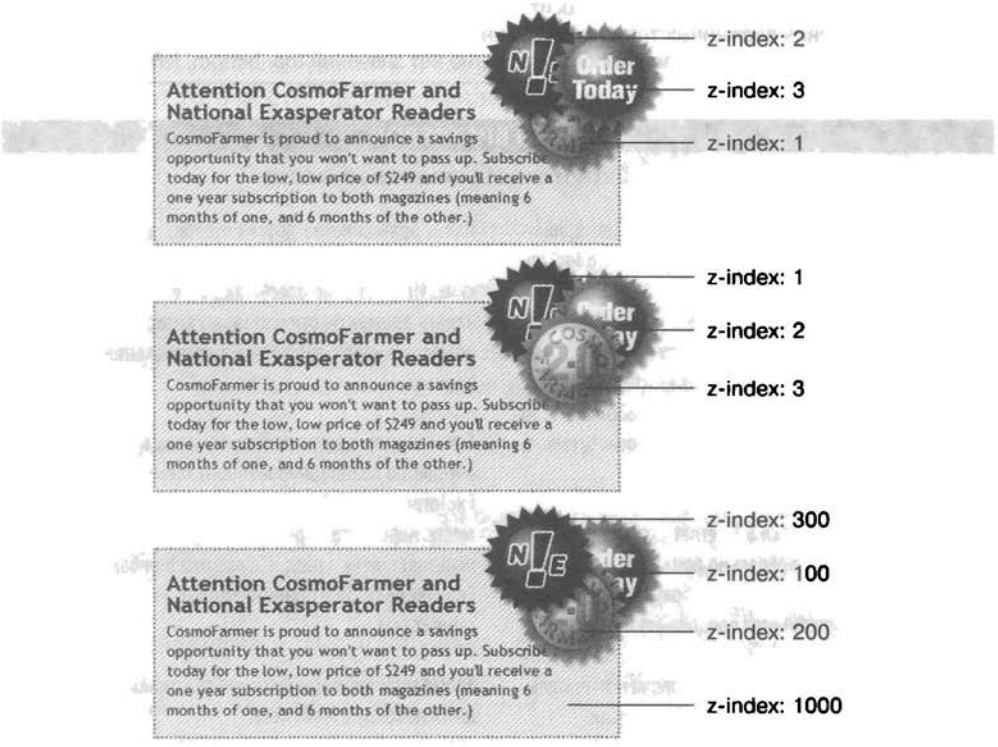


Рис. 13.6. Веб-страница имеет многослойную структуру

Чтобы сказать это другими словами, подумайте о веб-странице как о листе бумаги, а об абсолютно размещаемом элементе — как о записке, приклеиваемой сверху. Каждый раз, когда вы добавляете на страницу абсолютно позиционированный элемент, вы «приклеиваете» на нее записку. Конечно, если вы делаете это, то рискуете закрыть что-нибудь уже написанное на странице.

Обычно порядок наложения элементов следует их порядку в HTML-коде страницы. На странице с двумя абсолютно позиционированными тегами `<div>` второй тег из HTML появится выше другого тега `<div>`. Однако вы можете управлять порядком, в котором накладываются элементы, используя CSS-свойство `z-index`. Свойство получает числовое значение:

```
z-index: 3;
```

Чем больше значение, тем ближе к вершине набора появится элемент. Скажем, у вас есть три абсолютно позиционированных изображения и части каждого из них

в некоторой степени перекрываются. Изображение, имеющее больший индекс позиционного уровня, появится над другими (см. рис. 13.6, *вверху*). Когда вы меняете индекс позиционного уровня одного или нескольких изображений, вы изменяете порядок их наложения (см. рис. 13.6, *посередине*).

СОВЕТ

Это прекрасно, если у вас будут промежутки в значениях индекса позиционного уровня. Другими словами, значения 10, 20, 30 определяют то же самое, что и 1, 2, 3. В действительности, простор в числовых значениях дает вам возможность позже вставлять в набор больше элементов. Если вы хотите быть уверенными, что ничего никогда не появится над размещаемым элементом, задайте ему очень большой индекс позиционного уровня, например z-index: 10 000;. Но не переусердствуйте, поскольку максимальное значение, которое обрабатывает Firefox, равняется 2 147 483 647.

Скрытие частей страницы

Другое CSS-свойство, часто используемое с абсолютно позиционированными элементами, — свойство `visibility`. Оно позволяет скрыть часть страницы (или показать ее). Скажем, вы хотите, чтобы над изображением внезапно появлялось всплывающее сообщение, когда пользователь передвигает над ним указатель мыши. Вы делаете заголовок невидимым, когда страница загружается впервые (`visibility: hidden`), и переключаетесь на режим видимости (`visibility: visible`), когда указатель передвигается над изображением. На рис. 13.7 можно увидеть пример такого сообщения.

Для создания такого эффекта, как на рис. 13.7, *внизу*, программисты обычно используют JavaScript, но вы можете использовать CSS-псевдокласс `:hover`.

Значение `hidden` свойства `visibility` подобно значению `none` свойства `display` (см. разд. «Добавление границ» гл. 7), но между ними есть существенное различие. Когда вы устанавливаете свойству `display` элемента значение `none`, он буквально исчезает со страницы, не оставляя следов. Однако установка свойству `visibility` значения `hidden` предотвращает показ браузером содержимого элемента, но оставляет пустое пространство в том месте, где должен был находиться элемент. При использовании с абсолютно позиционированными элементами, которые уже удаляются из потока страницы, свойства `visibility: hidden` и `display: none` ведут себя одинаково.

Самый распространенный способ переключать режим элемента от скрытого к видимому и наоборот — через JavaScript. Однако вам не нужно изучать программирование на JavaScript, чтобы использовать свойство `visibility` (или `display`). Вы можете использовать псевдокласс `:hover` (см. разд. «Псевдоклассы и псевдоэлементы» гл. 3), чтобы сделать невидимый элемент видимым.

СОВЕТ

Для получения информации о базовом CSS-методе добавления всплывающего окна с подсказкой — вспомогательного текста, который появляется, когда посетитель передвигает указатель мыши над ссылкой, — зайдите на страницу http://psacake.com/web/jl.asp. У вас также есть возможность воспользоваться JavaScript: плагин the jQuery Tooltip является полноценной и простой в использовании подсказкой, основанной на фреймворке jQuery: http://bassistance.de/jquery-plugins/jquery-plugin-tooltip.



Рис. 13.7. Свойство visibility полезно для скрытия части страницы, которую вы хотите показать позже. На верхнем рисунке показан список кинофильмов. Перемещение указателя мыши над одним из изображений приводит к появлению сообщения *(внизу)*

Мощные стратегии позиционирования

Как говорилось в начале этой главы, если вы попытайтесь использовать CSS-позиционирование для размещения каждого элемента страницы, то можете столк-

нуться с проблемой. Поскольку просто невозможно предсказать все мыслимые комбинации браузеров и параметров настройки, используемые вашими посетителями, позиционирование под управлением CSS работает лучше всего в качестве «тактического оружия». Используйте его аккуратно, чтобы обеспечить точное размещение определенных элементов.

В этом разделе вы узнаете, как использовать абсолютное позиционирование, чтобы добавлять маленькие, но важные детали к дизайну своей страницы, как абсолютно позиционировать определенные элементы разметки и закреплять важные элементы страницы в одном месте, в то время как остальное содержимое будет прокручиваться.

Позиционирование внутри элемента

Один из самых эффективных способов использования позиционирования состоит в том, чтобы размещать маленькие элементы относительно других объектов на странице. Абсолютное позиционирование может повторять тип выравнивания по правому краю, которое вы задаете с плавающей разметкой. На рис. 13.8 (*1*) дата на верхнем заголовке кажется высоко расположенной, но с CSS вы можете переместить ее к правому краю нижнего заголовка.

Чтобы поместить дату отдельно от остальной части заголовка, вам нужно заключить ее в HTML-тег. Тег (см. врезку «Бриллиант без огранки» в разд. «Селекторы классов: точное управление» гл. 3) — распространенный выбор при использовании класса для расположенного в линию текста, чтобы определить его отдельно от остальной части абзаца.

```
<h1><span class="date">Nov. 10, 2006</span> CosmoFarmer Bought By Google</h1>
```

Теперь стоит вопрос о создании стилей. Во-первых, вы должны задать содержащему элементу (здесь — тегу <h1>) значение relative (относительное позиционирование). Затем примените абсолютное позиционирование (значение absolute) для элемента, который желаете разместить, — даты. Здесь приведен CSS-код для нижнего изображения в первом примере на рис. 13.8 (*1*):

```
h1 {
  position: relative;
  width: 100%;
  border-bottom: 1px dashed #999999;
}
h1 span.date {
  position: absolute;
  bottom: 0;
  right: 0;
  font-size: .5em;
  background-color: #E9E9E9;
  color: black;
  padding: 2px 7px 0 7px;
}
```

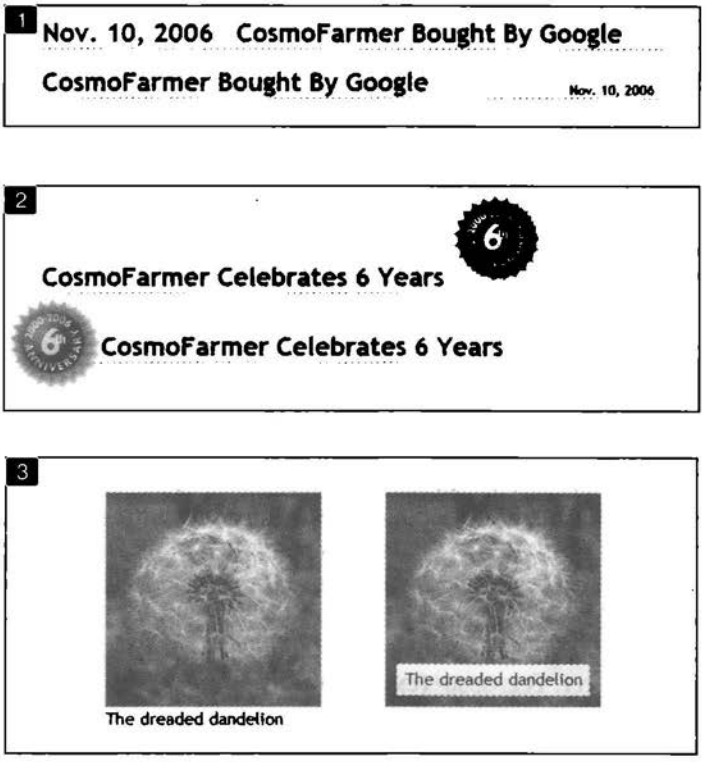


Рис. 13.8. Абсолютное позиционирование прекрасно подходит для простых элементов дизайна, например для размещения даты в нижнем правом углу заголовка *(1)*, выдавливания изображения из содержащего его блока *(2)* или размещения заголовка поверх фотографии *(3)*

Некоторые из приведенных выше свойств, например `border-bottom`, представлены просто для наглядности. Ключевые свойства выделены полужирным шрифтом: `position`, `bottom` и `right`. Как только вы задаете заголовку относительное позиционирование, вы можете поместить тег `<span>`, содержащий дату, в нижний правый угол заголовка, установив свойствам `bottom` и `right` значение 0.

ПРИМЕЧАНИЕ

Internet Explorer версий 6 и ниже может неправильно понять размещение элемента, когда вы используете свойства bottom и right. В этом примере объявление width: 100%; в стиле тега `<h1>` устраняет проблему, что обсуждалось во врезке «Ошибки браузеров» выше в этой главе.

Исключение элемента за пределы блока

Вы можете также использовать позиционирование для размещения элемента таким образом, чтобы он «выглядывал» из-за другого элемента. На рис. 13.8 (*2*) верхнее изображение показывает заголовок с графикой. Так, тег `<img>` помещен внутрь тега `<h1>` как часть заголовка. Использование абсолютного позиционирования и отрицательных значений свойств `top` и `left` позволяет переместить изображение

к левому краю заголовка и вытеснить его за верхний и левый края. Рассмотрим CSS-код, который используется в этом примере:

```
h1 {
  position: relative;
  margin-top: 35px;
  padding-left: 55px;
  border-bottom: 1px dashed #999999;
}
h1 img {
  position: absolute;
  top: -30px;
  left: -30px;
}
```

Основная концепция этого кода такая же, что и в предыдущем примере, но с некоторыми дополнениями. Во-первых, значения `top` и `left` изображения отрицательные, так что графика фактически появляется на расстоянии 30 пикселов над вершиной заголовка и 30 пикселов слева от левого края заголовка. Будьте внимательны, когда используете отрицательные значения. В результате может получиться так, что элемент будет частично (или полностью) расположен вне страницы или будет перекрывать содержимое другого элемента. Для предотвращения того, чтобы отрицательно позиционированный элемент выходил за рамки окна браузера, добавьте достаточно полей или отступов либо к самому элементу, либо к включающему его относительно позиционированному тегу — в данном примере к тегу `<h1>`. Дополнительные поля обеспечат достаточно места для выступающего изображения. В данном случае, чтобы предотвратить перекрытие изображением любого содержимого над заголовком, добавьте поле сверху. Левый отступ шириной 55 пикселов также «вытолкнет» текст заголовка из-под абсолютно позиционированного изображения.

Как и в предыдущем примере, Internet Explorer готов доставить неприятности. Но, что еще хуже, добавление `width: 100%` на этот раз даже не устранит проблемы. Поскольку у тега `<h1>` есть отступ, установка его ширине значения 100 % в действительности сделает заголовок шире, чем 100 % страницы (см. разд. «Определение параметров высоты и ширины» гл. 7 для получения подробной информации). Решение существует, но оно использует нестандартное CSS-свойство `zoom`. Просто добавьте `zoom: 1` к стилю тега `<h1>`:

```
h1 {
  position: relative;
  margin-top: 35px;
  padding-left: 55px;
  border-bottom: 1px dashed #999999;
  zoom: 1;
}
```

СОВЕТ

Свойство zoom не причиняет вреда другим браузерам, хоть и препятствует тому, чтобы ваш CSS-код был корректно принят (см. врезку «Информация для новичков» в разд. «Внешние таблицы стилей» гл. 2). Вы можете использовать условные комментарии Internet Explorer, чтобы скрыть нестандартное свойство, как описывается в разд. «Управление браузером Internet Explorer» гл. 15. Лучшее решение состоит в том, чтобы создать отдельную внешнюю таблицу стилей только для этого браузера.

Использование CSS-позиционирования для разметки страницы

Как говорилось в самом начале главы, попытка поместить каждый элемент на странице в определенном месте окна браузера обычно заканчивается не очень хорошо. Аккуратно используя абсолютное позиционирование, вы сможете создать много стандартных макетов веб-страниц (как вы видели в предыдущей главе). Из этого раздела вы узнаете, как разрабатывать свободную разметку на три столбца, применяя именно абсолютное позиционирование. У страницы будет баннер, левое и правое боковые меню, область главного содержимого и нижний колонтитул для информации об авторском праве.

ПРИМЕЧАНИЕ

Эта раздел научит вас общему подходу к абсолютному позиционированию, который вы можете использовать практически с любым макетом страницы. В обучающем уроке в конце этой главы вы найдете практический пример по созданию разметки с абсолютным позиционированием.

Каждый раз, когда вы используете абсолютное позиционирование, держите в уме такое правило: нельзя разместить все. Чтобы получить хорошую разметку, обычно вы должны использовать абсолютное позиционирование только для нескольких элементов страницы.

Рассмотрим простую методику, которую вы можете использовать, чтобы выяснить, какие именно элементы нужно размещать. Скажем, вы хотите разработать схему с тремя столбцами, как показано на рис. 13.9, *справа*. Сначала рассмотрите, как различные разделы страницы следуют нормальному порядку HTML-кода без какого-либо CSS-позиционирования (см. рис. 13.9, *слева*). Затем для каждого размещаемого элемента своей страницы спросите себя: «Будет ли он находиться в нужном месте, если я вообще не стану размещать его?»

Схема на рис. 13.9, *слева*, демонстрирует, как разбить HTML-код страницы на разделы, каждый из которых определен в теге `<div>` с уникальным идентификатором (ID). Схема на рис. 13.9, *справа*, показывает конечную разметку на три столбца и необходимые типы позиционирования: относительное — для тега `<div>`, охватывающего содержимое (см. рис. 13.9, *R*); абсолютное — для корректного размещения левого и правого боковых меню (см. рис. 13.9, *AP*). Наконец, область с главным содержимым нуждается в небольших левом и правом полях, чтобы препятствовать ее перекрытию боковыми меню (показано стрелками на рис. 13.9).

Теперь пройдемся по элементам страницы, представленной на схеме на рис. 13.9, *слева*.

- **Баннер.** Баннер *(1)* находится наверху страницы. Это как раз то место, где вы хотите его видеть, так что для него не потребуется абсолютного позиционирования. Вы можете использовать комбинацию полей и отступов, чтобы немного «освободить» содержимое (возможно, добавить немного пустого пространства над баннером или слева от него).
- **«Упаковка» содержимого.** Этот тег `<div>` является специальным элементом, который содержит все остальные элементы на странице *(2)*. Поскольку он хранит содержимое страницы, просто спросите себя, хотите ли вы, чтобы оно появилось

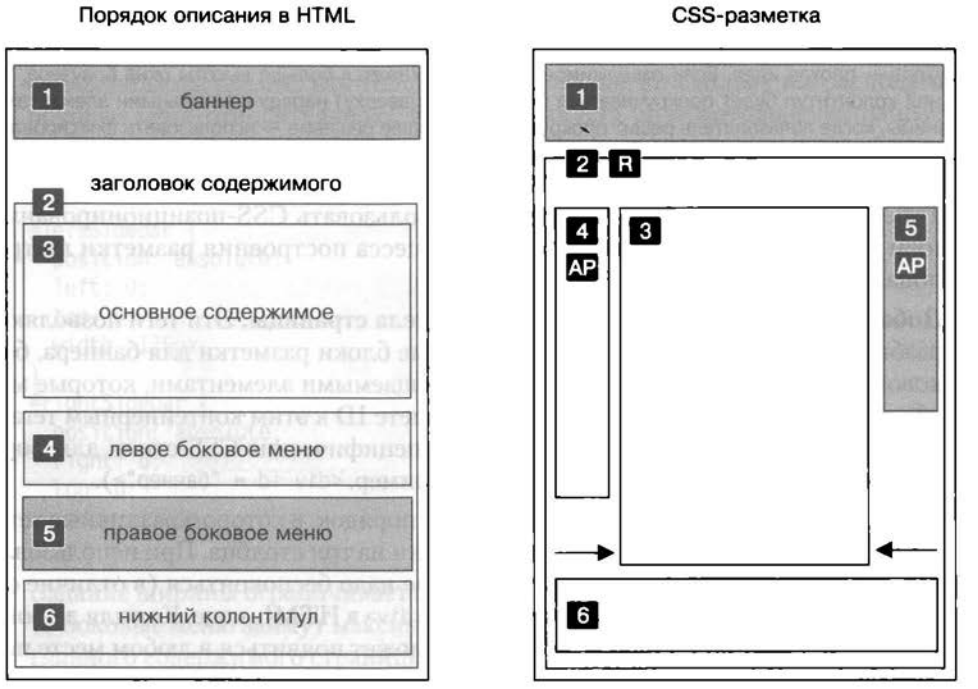


Рис. 13.9. Пример разметки страницы с использованием CSS-позиционирования

под баннером. По плану основное содержимое страницы должно располагаться именно под баннером, поэтому здесь также не нужно применять абсолютное позиционирование.

ПРИМЕЧАНИЕ

Вообще, роль этой «упаковки» для содержимого заключается в том, чтобы помочь вам разместить элементы, например боковое меню, на странице.

- **Основное содержимое.** Главная часть страницы также находится непосредственно под баннером *(3)*. Вам надо будет задать поля слева и справа, чтобы создать место для боковых меню, но для этого не требуется абсолютное позиционирование.
- **Левое боковое меню.** На рис. 13.9, *слева*, меню *(4)* появляется внизу страницы, под главным содержимым. Но разве оно должно быть там? Конечно, нет. Таким образом, для этого раздела необходимо абсолютное позиционирование.
- **Правое боковое меню.** Вместо того чтобы находиться справа, как следует из названия, меню *(5)* появляется внизу страницы под левым боковым меню. Здесь снова нужно абсолютное позиционирование, чтобы поместить этот элемент в должном месте — под баннером в правой части страницы.
- **Нижний колонтитул.** На рис. 13.9, *слева*, нижний колонтитул появляется внизу страницы *(6)* — как раз там, где вы хотите его видеть, так что здесь не нужно применять специальное позиционирование.

ПРИМЕЧАНИЕ

Использовать абсолютное позиционирование для размещения нижнего колонтитула внизу окна браузера — плохая идея. Если содержимое страницы окажется больше высоты окна браузера, то нижний колонтитул будет прокручиваться (и окажется наверху) наряду с остальными элементами страницы, когда пользователь решит прокрутить ее. Лучшее решение — использовать фиксированное позиционирование, которое описано в следующем подразделе.

Теперь, когда вы сможете решить, где использовать CSS-позиционирование в своем проекте, приведу краткий обзор процесса построения разметки на три столбца.

1. **Добавление тегов для каждого раздела страницы.** Эти теги позволяют разбить содержимое страницы на отдельные блоки разметки для баннера, бокового меню и т. д. Как и в случае с перемещаемыми элементами, которые мы обсуждали в предыдущей главе, вы добавляете ID к этим контейнерным тегам и, таким образом, можете создавать специфические CSS-стили для форматирования каждой части страницы (например,).

   Левое изображение на рис. 13.9 показывает порядок, в котором различные теги появляются в HTML-коде для разметки на три столбца. При использовании абсолютного позиционирования вам не надо беспокоиться (в отличие от перемещаемых элементов) о порядке тегов в HTML-коде. Код для любого абсолютно позиционированного элемента может появиться в любом месте потока файла: сразу после открывающего тега <body>, прямо перед закрывающим тегом </body> или где-нибудь между ними. Свойства позиционирования определяют, где элемент появится на экране, а не его место в HTML-коде.

2. **Включите весь HTML-код для главного содержимого, боковых меню и нижнего колонтитула в другой тег .** Этот (см. рис. 13.9, *2*) собирает все разделы содержимого в одну «упаковку». Добавьте ID к тегу, после чего вы сможете разрабатывать его (например,). Этот тег обеспечивает среду для позиционирования боковых меню, что вы увидите далее.

3. **Задайте содержащему тегу относительную позицию.** Используйте свойство position и значение relative, чтобы создать стиль такого вида:

   ```
   #contentWrapper { position: relative; }
   ```

   Помните, что если вы не указываете значения top, bottom, left или right для относительно позиционированного элемента, он появится в том месте, где и должен был появиться без позиционирования, в нашем случае — непосредственно под баннером. Относительное позиционирование на самом деле изменяет расположение элементов в теге . Теперь, когда вы используете абсолютное позиционирование для размещения боковых меню (а это следующий шаг), вы можете установить значения top относительно «упаковки», а не окна браузера. Таким образом, если вы добавляете больше содержимого баннеру и делаете его выше, содержащий тег и все внутри его смещается аккуратно вниз по странице. Без тега ваши боковые меню должны были размещаться относительно верхнего края окна браузера и вы должны были указать их значения top.

4. **Примените абсолютное позиционирование для боковых меню.** Поскольку все остальное содержимое на странице прекрасно помещается там, где оно находится, вы должны разместить только боковые меню. Поскольку вы располагаете боковые меню относительно содержащего их тега , который настраивается в шаге 3, можете просто задать значение 0 позициям top и left левого бокового меню и позициям top и right правого бокового меню.

```
#leftSidebar {
  position: absolute;
  left: 0;
  top: 0;
  width: 175px;
}
#rightSidebar {
  position: absolute;
  right: 0;
  top: 0;
  width: 180px;
}
```

Задание ширины ограничивает боковые меню. Если вы не установите ширину, то боковые меню займут максимально возможное место, не оставив ничего для главного содержимого страницы.

СОВЕТ

Вы можете отрегулировать значения top, left и right как вам нравится. Если левое боковое меню будет лучше выглядеть после задания небольших отступов от левого и верхнего краев «упаковки», измените значения left и top, скажем, на 10px или укажите любое другое подходящее значение.

5. **Отрегулируйте края главного содержимого.** Поскольку для левого и правого боковых меню задано абсолютное позиционирование, они удаляются из потока страницы. Главное содержимое даже «не знает», что они существуют, так что оно просто идет прямо под ними. Однако главное содержимое находится в правильном месте на странице — под баннером, поэтому вам не нужно переносить его. Все, что вы должны сделать, — немного отдалить основное содержимое от левого и правого краев, чтобы очистить боковые меню.

Чтобы сделать это, добавьте левое и правое поля к тегу с главным содержимым. Установите значение для каждого поля, равное или большее, чем ширина бокового меню:

```
#mainContent {
  margin-left: 185px;
  margin-right: 190px;
}
```

В этом коде поля получились немного больше, чем ширина каждого бокового меню. Обычно полезно чуть увеличить края, чтобы были видны промежутки между различными элементами на странице.

Хоть все и получилось красиво, у этой разметки есть свое слабое место. Каждый раз, когда вы используете абсолютно позиционированные столбцы (как эти

боковые меню), они потенциально могут вырасти и перекрыть часть нижнего колонтитула или других, более низких элементов HTML (рис. 13.10). В отличие от перемещаемых элементов, где вы можете очистить элемент и заставить его появиться ниже свободно расположенного столбца, CSS не дает вам никакого способа очистить низ позиционированного столбца. Лучшее, что вы можете сделать, — найти обходной путь, как в следующем шаге.



Рис. 13.10. Если абсолютно позиционированный столбец длиннее, чем любое статически размещенное содержимое, которое должно появиться внизу столбца, произойдет наложение. Устранить проблему можно, создав поле для нижнего колонтитула

6. **При необходимости добавьте поля к нижнему колонтитулу, чтобы предотвратить его перекрытие боковыми меню.** Другой вариант — просто убедитесь, что

любые абсолютно позиционированные столбцы *никогда* не окажутся длиннее главного содержимого. Когда главное содержимое достаточно длинное, оно вытеснит нижний колонтитул, который окажется под столбцами и вы избежите проблем.

СОВЕТ

Если вам нравится применять все новшества, то можете попробовать JavaScript-решение этой проблемы, описанное по адресу www.shauninman.com/plete/2006/05/clearance-position-inline-absolute.php.

Вы можете изменять эту методику базовой разметки страницы любыми путями. Удалите правое боковое меню и свойство `right-margin` тега `<div>` с главным содержимым, и вы получите разметку с двумя столбцами. Или вместо этого удалите левое боковое меню, чтобы создать разметку с двумя столбцами, но с более тонким столбцом справа. Вы можете также использовать эту базовую схему в разметке с фиксированной шириной. Просто установите ширину для баннера и ширину для тега `<div>` с содержимым таким образом:

```
#banner, #contentWrapper { width: 760px; }
```

Создание CSS-стиля для шапки страницы с использованием фиксированного позиционирования

В каких-то случаях бывает нужно, чтобы некоторые элементы страницы были постоянно видимыми, например навигационная панель, поле для поиска или логотип сайта, а, как известно, большинство веб-страниц длиннее высоты экрана. *Фреймы* HTML были когда-то единственным способом «удержать» важные элементы, в то время как другое содержимое прокручивалось и исчезало из области видимости. Однако у HTML-фреймов есть важные недостатки. Поскольку каждый фрейм описывается в отдельном файле веб-страницы, приходится создавать несколько HTML-файлов, чтобы сделать одну полноценную веб-страницу (называемую страницей, содержащей *набор фреймов*). Это не только отнимало много времени у разработчиков, но и усложняло поиск по сайту для поисковых машин.

HTML-страницы, содержащие набор фреймов, могут также быть неудобными для посетителей, которые используют экранных дикторов из-за проблем со зрением, или тех, кто хочет распечатать страницы сайта.

Тем не менее идея фреймов все еще пригодна, так что CSS предлагает вариант позиционирования, который позволит вам получить видимость фреймов с меньшим количеством работы. Вы можете просмотреть страницу, созданную с использованием фиксированного позиционирования, на рис. 13.11. Используя значение `fixed` свойства `position`, вы можете имитировать HTML-фреймы, фиксируя некоторые элементы в определенных местах, но все еще разрешая пользователям прокручивать содержимое очень длинной веб-страницы.

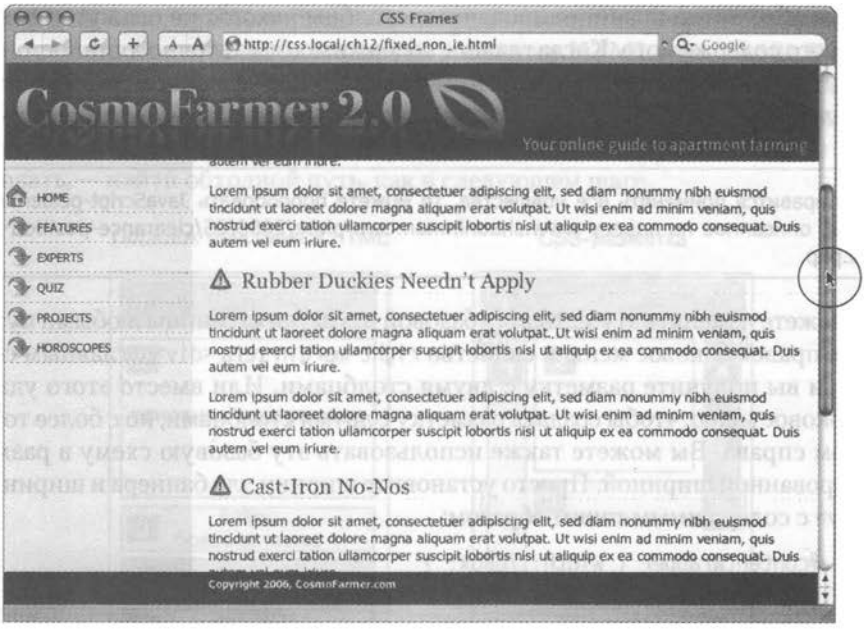


Рис. 13.11. Страница с фреймами, созданная посредством CSS. Полоса прокрутки *(выделена кружком)* позволяет прокручивать только большую текстовую область; верхний и нижний баннеры и боковое меню остаются неподвижными

ПРИМЕЧАНИЕ

Фиксированное позиционирование не работает в Internet Explorer версий 6 и ниже. Однако всего с небольшим дополнительным CSS-кодом (описанным в шаге 5 ниже) вы можете заставить страницу выглядеть прекрасно и в Internet Explorer 6 (хотя «фиксированные элементы» в итоге прокручиваются наряду с остальными). Поскольку же Internet Explorer 7 и 8 распознают фиксированное позиционирование, можете использовать эту методику и получить подобные результаты для всех посетителей вашего сайта.

Фиксированное позиционирование работает во многом подобно абсолютному — вы точно так же можете использовать свойства `top`, `bottom`, `left` или `right` для размещения элемента. Как и абсолютное, фиксированное позиционирование удаляет элемент из потока HTML. Он «плавает» над другими частями страницы, которые просто игнорируют его.

Рассмотрим, как можно создать страницу, похожую на изображенную на рис. 13.11, у которой есть фиксированный баннер, боковое меню и нижний колонтитул, а также прокручиваемая область с главным содержимым.

1. **Добавьте теги с атрибутами ID для каждого раздела страницы.** У вас может быть четыре основных тега `<div>` с такими ID, как `banner`, `sidebar`, `main` и `footer` (рис. 13.12). Порядок, в котором вы указываете эти теги в HTML, не имеет значения. Как и абсолютное позиционирование, фиксированное позволяет вам размещать элементы на странице независимо от их расположения в HTML.

ПРИМЕЧАНИЕ

Есть одно исключение: чтобы страница нормально выглядела для пользователей Internet Explorer 6, HTML-код для нижнего колонтитула должен появиться под HTML-кодом для области с главным содержимым, что вы увидите в шаге 5 ниже.

2. **Добавьте свой материал к каждому тегу .** Вообще, используйте фиксированные разделы `div` для материала, к которому у посетителя всегда должен быть доступ, в областях, которые вы желаете закрепить в конкретном месте. В этом примере баннер, боковое меню и нижний колонтитул содержат логотип, глобальную навигацию по сайту и сведения об авторском праве.

   Главное содержимое помещается в последний тег `<div>`. Однако не добавляйте слишком много информации к фиксированному тегу `<div>`. Если фиксированное боковое меню длиннее, чем окно браузера пользователя, посетитель не сможет увидеть боковое меню целиком. Поскольку фиксированные элементы не прокручиваются, у пользователя не будет никакой возможности (за исключением покупки большего монитора) увидеть содержимое бокового меню, которое не соответствуют окну его браузера.

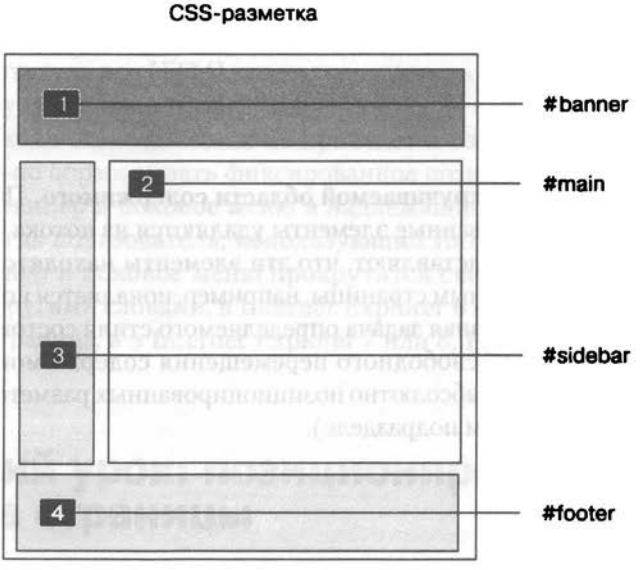


Рис. 13.12. С фиксированным позиционированием можно закрепить все элементы страницы, чтобы они всегда были в поле зрения при прокрутке. В этом примере заголовок *(1)*, боковое меню *(3)* и нижний колонтитул *(4)* зафиксированы, а область с главным содержимым *(2)* может прокручиваться

3. **Создайте стили для всех фиксированных элементов.** Значения `top`, `bottom`, `left` и `right` задаются относительно окна браузера, таким образом, просто определите, где вы хотите их видеть на экране, и добавьте значения. Кроме того, определите ширину для элементов.

ПРИМЕЧАНИЕ

В отличие от абсолютного позиционирования, фиксированное всегда задается относительно окна браузера, даже когда элемент с таким позиционированием помещается внутрь другого тега с относительным или абсолютным позиционированием.

Стили для размещения элементов *1*, *3* и *4* на рис. 13.12 выглядят следующим образом:

```
#banner {
  position: fixed;
  left: 0;
  top: 0;
  width: 100%
}
#sidebar {
  position: fixed;
  left: 0;
  top: 110px;
  width: 175px;
}
#footer {
  position: fixed;
  bottom: 0;
  left: 0;
  width: 100%;
}
```

4. **Создайте стиль для прокручиваемой области содержимого.** Поскольку фиксированно позиционированные элементы удаляются из потока HTML, другие теги на странице не представляют, что эти элементы находятся там. Так, тег с главным содержимым страницы, например, появляется под фиксированными элементами. Основная задача определяемого стиля состоит в том, чтобы использовать поля для свободного перемещения содержимого (концепция здесь такая же, как и для абсолютно позиционированных разметок, которые мы обсуждали в предыдущем подразделе).

```
#main {
  margin-left: 190px;
  margin-top: 110px;
}
```

5. **Отрегулируйте разметку для Internet Explorer версий 6 и ниже.** Internet Explorer 6 не понимает фиксированное позиционирование. Он рассматривает фиксированные элементы как статические и не пытается размещать их в конкретных местах на странице. В зависимости от того, в каком порядке идет ваш HTML-код, в Internet Explorer 6 страница может выглядеть странной из-за больших промежутков между баннером и боковым меню или, что еще хуже, навигационная панель и баннер могут в итоге оказаться ниже главного содержимого.

Выход состоит в том, чтобы указать Internet Explorer 6 рассматривать фиксированные элементы как абсолютно позиционированные, что приведет к удалению этих элементов из потока страницы и размещению их в правильных местах в окне браузера.

```
* html #banner { position: absolute; }
* html #sidebar { position: absolute; }
```

ПРИМЕЧАНИЕ

Эти стили используют выражение * html, чтобы скрыть свойства от браузеров, отличных от Internet Explorer версий 6 и ниже (см. врезку «Ошибки браузеров» в разд. «Определение параметров высоты и ширины» гл. 7). Вы можете также использовать условные комментарии Internet Explorer (см. разд. «Управление браузером Internet Explorer» гл. 15).

Вы могли заметить, что стиль #footer не был упомянут. Не стоит абсолютно позиционировать нижний колонтитул — в противном случае он будет «ездить» по окну браузера при прокрутке, располагаясь прямо поверх остального прокручиваемого содержимого.

В этом случае лучше всего, чтобы нижний колонтитул появился внизу страницы и прокручивался вверх, как и любой другой непозиционированный колонтитул (вот почему, как было упомянуто в шаге 1, вы должны поместить HTML-код для нижнего колонтитула под HTML-кодом для главного содержимого так, чтобы он появился внизу страницы в Internet Explorer 6).

Использование этой методики не приведет к тому, что Internet Explorer 6 станет правильно обрабатывать фиксированное позиционирование, но она хотя бы помещает баннер и боковое меню в надлежащие места, когда страница загружается. Когда пользователь, использующий Internet Explorer 6, прокрутит страницу, баннер и боковое меню прокрутятся сверху вниз, как и остальное содержимое. Другими словами, в Internet Explorer 6 ваша страница работает как обычная веб-страница, а в Internet Explorer 7 или 8, Firefox, Safari и Opera — еще лучше.

Обучающий урок: позиционирование элементов страницы

Этот обучающий урок позволит вам исследовать несколько различных способов использования абсолютного позиционирования, таких как создание разметки на три столбца, размещение элементов внутри баннера и добавление заголовков поверх фотографий. В отличие от предыдущей главы, где вы описывали куски HTML-кода в тегах и добавляли ID или имена классов к ним, в этих уроках бóльшая часть работы с HTML уже была выполнена. Вы можете сосредоточиться на оттачивании ваших новых навыков в CSS.

Чтобы приступить, загрузите учебные файлы, расположенные на сайте **www.sawmac.com/css2e/**. Как работать с ними, описывается в конце гл. 2.

Улучшение баннера страницы

Во-первых, сделаем несколько маленьких, но визуально важных изменений в баннере страницы. Создадим стили, которые обращаются к HTML-тегам с ID или классами, примененными к ним (опять же это уже сделано).

1. Запустите браузер и откройте файл `index.html` из папки **13**.

   На этой веб-странице (рис. 13.13) начните с перепозиционирования некоторых частей баннера.

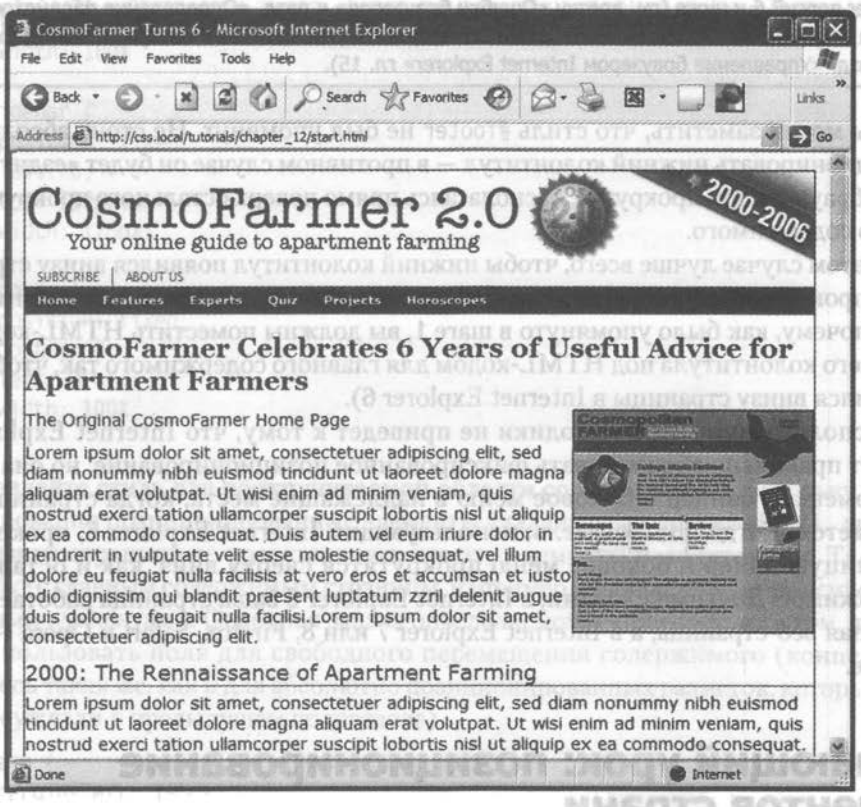


Рис. 13.13. Это обычная статическая HTML-страница, где все заполнено сверху донизу. Вы можете сделать ее более удобной для чтения, организовав все содержимое по столбцам

2. Откройте файл `index.html` в текстовом редакторе. Поместите курсор между открывающим и закрывающим тегами `<style>`.

   Наряду с тегами `<style>` для внутренней таблицы стилей у страницы уже есть присоединенная внешняя таблица стилей с некоторым базовым форматированием. Начните с перемещения маленького значка CosmoFarmer 2.0 в левую сторону баннера. Чтобы избавиться от квадратного вида, типичного для CSS-дизайна, разверните графику за пределы границ баннера, и, таким образом, он будет похож на надвинутую наклейку.

3. Во внутренней таблице стилей добавьте новый стиль:

```
#banner #badge {
position: absolute;
left: -18px;
top: -18px;
}
```

Графика находится внутри тега с ID, равным banner, а у самой графики ID равно badge. Этот стиль помещает левый верхний угол графики на 18 пикселов влево и 18 пикселов над верхним краем страницы.

Теперь предварительно просмотрите страницу, и вы увидите несколько проблем. Первое — то, что графика «нависает» над краем страницы, а вы хотите, чтобы она располагалась над краем области баннера. Займемся этой проблемой.

4. Добавьте следующий стиль перед тем, который только что создали:

```
#banner {
  position: relative;
}
```

Хорошей практикой считается помещать CSS-код для стилей, которые управляют основным разделом страницы (как этот стиль #banner), над кодом для стилей, форматирующих только части этого раздела (как стиль, который мы создали в шаге 3). Кроме того, группирование стилей для связанных разделов облегчает их поиск, когда нужно проанализировать или отредактировать CSS-код страницы. В этом случае стиль #banner идет первым во внутренней таблице стилей, потому что он применяется к большему куску HTML-кода. Но вы должны держать стиль #banner #badge рядом с ним, так как добавляете больше стилей на страницу (прочитать о методиках организации вашего CSS-кода вы можете в разд. «Организация стилей и таблиц стилей» гл. 15).

Стиль #banner создает новую среду расположения для любых вложенных тегов. Другими словами, значение relative делает так, что любые другие элементы внутри этого тега получают место относительно краев баннера. Это изменение в позиционировании меняет размещение стиля, который вы создали в шаге 3. Теперь он смещен на 18 пикселов вверх и влево от области баннера. Значок все еще немного «нависает» над страницей, поэтому нужно добавить небольшие поля вокруг страницы, чтобы отрегулировать изображения.

5. Добавьте стиль для тега <body>. Поместите его во внутреннюю таблицу стилей над другими двумя стилями, которые вы создали:

```
body {
  margin: 20px;
}
```

Благодаря этому полю видно все изображение (рис. 13.14). Однако теперь у вас есть другая проблема — логотип CosmoFarmer частично скрыт под значком. Наложение элементов — одна из опасностей абсолютного позиционирования. Вы можете устранить проблему, добавив небольшой отступ к логотипу.

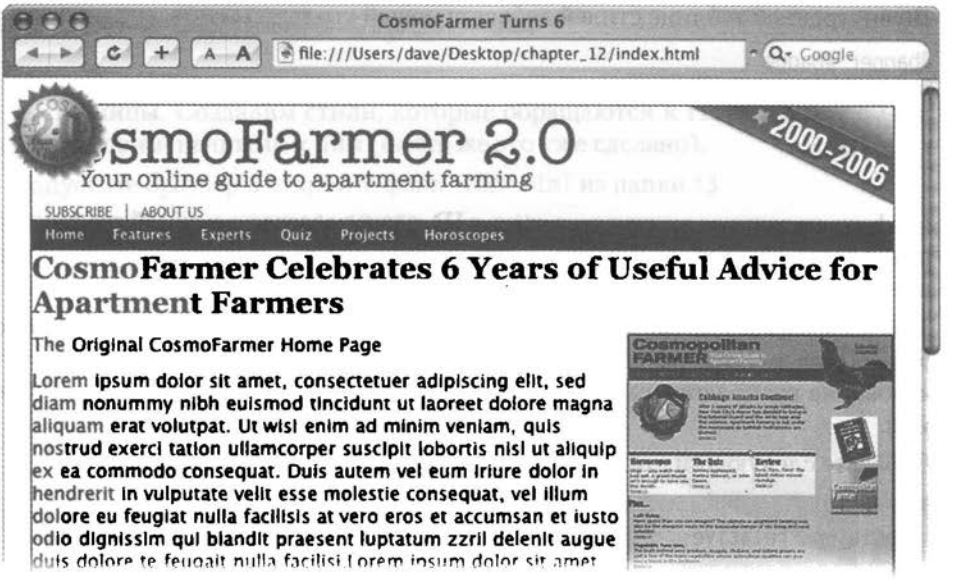


Рис. 13.14. Развертывание графики за пределы блока — верное дело при работе с абсолютным позиционированием. Пересекая границы баннера, графика смягчает квадратный вид остальной части баннера и придает динамичность дизайну

6. Добавьте новый стиль для логотипа во внутреннюю таблицу стилей. Поместите его под другими стилями, уже созданными вами:

```
#banner #logo {
  margin-left: 60px;
}
```

Как и для графического значка, для логотипа уже применен ID — `logo`. Этот стиль перемещает логотип далеко влево так, чтобы он был труднодоступным для абсолютно позиционированной графики. Однако произойдет странная вещь, если вы просмотрите его в Internet Explorer 6 или 7: когда вы передвигаете указатель мыши над навигационной панелью, логотип перескакивает туда, где был установлен раньше. Что такое? К счастью, проблема легко решается.

7. Отредактируйте созданный стиль `#banner #logo`, установив для него относительное позиционирование:

```
#banner #logo {
  margin-left: 60px;
  position: relative;
}
```

Добавление относительного позиционирования на самом деле никуда не перемещает логотип — это случится, только если вы добавите значения `top`, `bottom`, `left` или `right`. Однако по причинам, известным только Microsoft, оно заставляет работать браузер Internet Explorer.

Баннер пока смотрится хорошо, но ссылки SUBSCRIBE и ABOUT US выглядят неуклюже зажатыми между логотипом и навигационной панелью. На правой стороне баннера есть много места, так что мы переместим их туда.

ПРИМЕЧАНИЕ

Ссылки — это на самом деле неупорядоченный список, получающий свое форматирование от внешней таблицы стилей страницы. Чтобы получить подробную информацию о том, как превратить неупорядоченный список в горизонтальную навигационную панель, прочитайте разд. «Создание панелей навигации» гл. 9.

8. Добавьте новый стиль внизу внутренней таблицы стилей:

```
#banner ul {
  position: absolute;
  right: 60px;
  bottom: 5px;
}
```

Этот стиль определяет селектор потомков, который предназначен для неупорядоченных списков внутри баннера (в нашем случае есть только один список). Поскольку тег <ul> позиционирован абсолютно, он удаляется из потока страницы, позволяя навигационной панели находиться только под баннером.

Помните также, что этот тег находится внутри баннера, для которого вы ранее установили позиционирование relative. Соответственно, ссылки SUBSCRIBE и ABOUT US определяются относительно тега. Они размещаются на расстоянии в конкретную величину от правого и нижнего краев баннера, если только вы не просматриваете их в Internet Explorer версий 6 и ниже. Как говорилось во врезке «Ошибки браузеров» в разд. «Как работают свойства позиционирования» в начале этой главы, у Internet Explorer 6 есть проблемы при размещении элементов с использованием нижних координат относительно позиционированных элементов (как этот баннер). В итоге он применяет нижние координаты всей страницы. К счастью, решить эту проблему легко.

СОВЕТ

Если вы пользуетесь Internet Explorer, то можете увидеть ссылки, прокрутив веб-страницу вниз до самого конца.

9. Отредактируйте стиль #banner, который вы создали в шаге 4, и добавьте zoom: 1;.

```
#banner {
  position: relative;
  zoom: 1;
}
```

Эти строки — совершенно бессмысленный код, который все браузеры, за исключением Internet Explorer, просто игнорируют.

10. Предварительно просмотрите страницу в браузере.

Полностью завершенный баннер должен быть похож на тот, что показан на рис. 13.15. Это упражнение — отличный пример использования абсолютного позиционирования для внесения небольших изменений, которые добавляют вашей странице привлекательный вид.



Рис. 13.15. Абсолютное позиционирование оказывает огромное содействие при размещении маленьких элементов. При этом точный порядок ссылок в HTML-коде неважен, что дает много гибкости при разметке

Добавление надписи к фотографии

В гл. 8 мы рассмотрели один из способов добавления подписи к фотографии (см. подраздел «Добавление подписей к изображениям» разд. «Обучающий урок 1: создание фотогалереи»). В примерах той главы подписи располагались под фотографиями, чего мы и добиваемся в большинстве случаев. Но когда-нибудь может понадобиться добавить надпись непосредственно на фотографию, как, например, субтитры в телевизионных новостях, которые показываются в нижней части экрана.

1. Откройте файл `index.html` в текстовом редакторе.

   Обратите внимание на изображение оригинальной домашней страницы CosmoFarmer. В настоящее время оно выровнено по правому краю с использованием атрибута `align` тега `<img>`, но этот способ давно устарел. Вы выровняете изображение, используя CSS, но сначала нужно отредактировать кое-какой HTML-код.

2. Определите местонахождение тега `<img>`, который добавляет изображение `old_home.jpg`, а затем удалите выражение `align = "right"`.

   Рассмотрим, как тег выглядит целиком. Вам надо удалить выделенную часть:

   ```
   <img src="images/old_home.jpg" alt="The Original Cosmo Home Page"
     height="186" align="right" />
   ```

   Теперь, когда вы избавились от старого HTML-кода, нужно создать контейнер — тег `<div>`, — который содержал бы CSS-код для изображения и надписи к нему.

3. Сразу перед тегом `<img>` добавьте `<div class = "figure">`. После закрывающего тега `</p>` надписи (который появляется сразу после тега `<img>`) добавьте закры-

вающий тег . Когда вы это сделаете, HTML-код должен выглядеть следующим образом:

```
<div class="figure">
<img src="images/old_home.jpg" alt="The Original Cosmo Home Page"
  height="186"/>
<p>The Original CosmoFarmer Home Page</p>
</div>
```

Весь код для изображения и надписи находится в одном блоке, который вы можете выровнять и отформатировать как отдельный элемент.

4. Создайте стиль, чтобы отформатировать добавленный тег :

```
#main .figure {
  float: right;
  width: 200px;
  margin-bottom: 2px;
  margin-left: 10px;
}
```

Свойства в стиле должны быть к этому времени обновлены (особенно если вы читали гл. 8). Стиль выравнивает блок по правому краю страницы, а нижние и левые отступы добавляют немного места между блоком с изображением и текстом, который обтекает его.

Теперь нам нужно убрать надпись из нормального потока страницы и поместить ее поверх изображения. Чтобы сделать это, следует определить надпись относительно краев изображения. Однако, поскольку тег <img> является самозакрывающимся (то есть у него нет открывающего и закрывающего тегов), вы должны определить надпись относительно другого элемента. Это иное использование стиля figure тега , который вы только что добавили, — он обеспечивает среду расположения для надписи.

5. Добавьте position: relative к стилю, который вы только что создали:

```
#main .figure {
  float: right;
  width: 200px;
  margin-bottom: 2px;
  margin-left: 10px;
  position: relative;
}
```

Теперь вы можете определить надпись относительно тега , а это то же самое, что и определение ее относительно изображения.

6. Добавьте новый стиль после стиля #main.figure, созданного в предыдущих шагах:

```
#main .figure p {
  position: absolute;
  width: 168px;
  left: 10px;
  bottom: 10px;
  background-color: #FFF;
}
```

Этот стиль устанавливает надпись на расстоянии 10 пикселов от нижнего края и 10 пикселов от левого края тега . Свойство width ограничивает ширину текста, так что он не охватывает все изображение, а свойство background-color делает надпись четкой. Все, что нам осталось, — добавить некоторые детали форматирования, чтобы улучшить вид надписи.

7. Отредактируйте стиль, который вы только что создали, следующим образом:

```
#main .figure p {
  position: absolute;
  width: 168px;
  left: 10px;
  bottom: 10px;
  background-color: #FFF;
  border: 1px dashed #666666;
  font-size: 13px;
  font-weight: bold;
  text-align: center;
  padding: 5px;
  margin: 0;
}
```

Вы должны уделить особое внимание одной маленькой детали. Вы могли бы никогда этого не заметить, но некоторые браузеры помещают надпись всего на несколько пикселов ниже, чем другие браузеры (чтобы убедиться в этом, проверьте страницу в Internet Explorer, а затем в Firefox). Браузеры размещают встроенные элементы (например, изображения) по-разному относительно других объектов вокруг (вы можете увидеть подобную проблему с изображениями в примерах из разд. «Создание стилей для таблиц» гл. 10). Во всяком случае, решить эту проблему легко: используя CSS, заставьте рисунок отображаться как элемент блочной модели.

8. Добавьте еще один стиль к внутренней таблице стилей:

```
#main .figure img {
  display: block;
}
```

Предварительно просмотрите страницу. Надпись должна быть центрированной в нижней части изображения, как показано на рис. 13.16.



Рис. 13.16. Только абсолютное позиционирование позволяет вам накладывать один элемент поверх другого, как, например, надпись на этой фотографии

Разбиение страницы

Теперь пришло время обратить внимание на структуру страницы. При такой структуре, как сейчас, вы должны прокрутить страницу вниз, чтобы прочитать последние новости в боковом меню и увидеть объявления (что очень не нравится рекламодателям). В этом разделе будет рассказано, как использовать абсолютное позиционирование, чтобы создать гибкую разметку на три столбца, которая перенесет все содержимое к вершине страницы (и воспрепятствует тому, чтобы ваши спонсоры отказались от своих вложений).

Прежде чем начать, просмотрите структуру страницы на рис. 13.17.

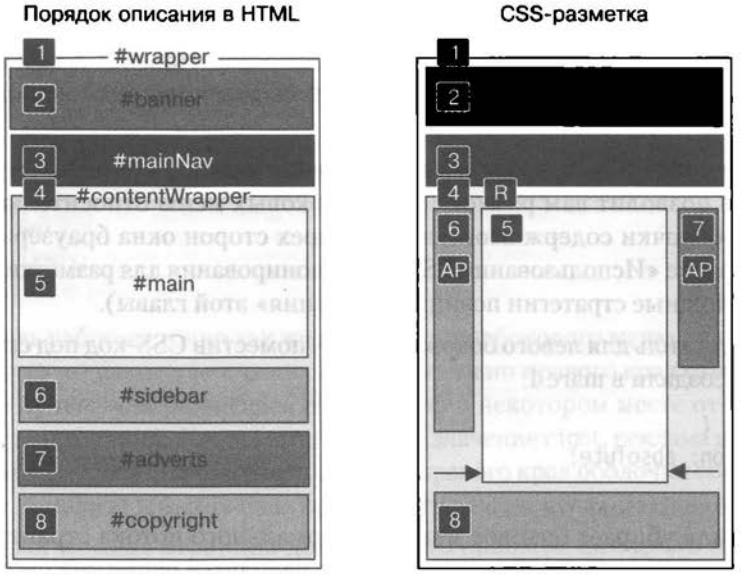


Рис. 13.17. Секрет абсолютно позиционированных разметок — правильное расположение элементов. Если есть проблемы, используйте поля или отступы (например, переместите элемент на несколько пикселов влево)

Каждый раздел страницы описан в собственном теге `<div>` с соответствующим ID. Главное содержимое страницы, боковое меню, реклама и записи об авторском праве заключены в тег с ID, равным `contentWrapper` (см. рис. 13.17, *4*). Все теги в теле страницы описаны в теге `<div>` с ID `wrapper` (см. рис. 13.7, *1*). Вам может показаться, что тегов `<div>` слишком много, но у каждого из них свое назначение.

Ваша задача состоит в том, чтобы классифицировать три тега `<div>` (`sidebar`, `main` и `adverts`) в три столбца. Вы должны использовать абсолютное позиционирование только для двух элементов — бокового меню и раздела рекламы (см. рис. 13.17, (*6* и *7*)). Вы убираете их из нормального потока страницы (где они появляются рядом с нижним полем) и прикрепляете к левому и правому краям, сразу под баннером. Абсолютное позиционирование также вызывает эффект «нависания» этих элементов над страницей и областью с главным содержимым (см. рис. 13.7, *5*).

Чтобы создать место для боковых меню, вы должны добавить небольшие поля слева и справа от главной области.

1. Создайте стиль для тега , который охватывает главное содержимое страницы (см. рис. 13.17, *3*). Добавьте его как последний стиль во внутреннюю таблицу стилей:

```
#contentWrapper {
  clear: both;
  position: relative;
}
```

Свойство clear помогает #contentWrapper очистить навигационную панель, которая была создана с перемещением влево (как рассказывается в разд. «Управление обтеканием содержимого плавающих элементов» гл. 7, вам всегда нужно очищать элементы, которые должны появиться под плавающими элементами).

Свойство position пригодится для боковых меню. Установка ему значения relative позволит вам разместить оба боковых меню относительно четырех полей оболочки содержимого, а не четырех сторон окна браузера (см. шаг 3 в подразделе «Использование CSS-позиционирования для разметки страницы» разд. «Мощные стратегии позиционирования» этой главы).

2. Создайте стиль для левого бокового меню, поместив CSS-код под стилем, который вы создали в шаге 1:

```
#sidebar {
  position: absolute;
}
```

Этот стиль убирает боковое меню из нормального потока страницы, но еще нигде не размещает его. Оно по-прежнему находится около нижнего края страницы, но если вы посмотрите на меню в браузере, то увидите, что оно расположено поверх рекламных объявлений. Чтобы разместить его, используйте свойства top и left.

3. Добавьте свойства top и left к стилю, который вы только что создали:

```
#sidebar {
  position: absolute;
  top: 15px;
  left: 0;
}
```

Поскольку это боковое меню размещается относительно тега с содержимым, свойство left, равное 0, сдвигает меню к левому краю. Значение top определяется методом проб и ошибок. Если установить для свойства top значение 0, то боковое меню будет касаться основания навигационной панели; 15 пикселов пустого места сделают так, что боковое меню станет больше соответствовать краткой сводке новостей.

Мы упустили одну вещь: для меню нужно определить ширину, так как на данный момент оно растягивается почти через всю страницу, перекрывая практически все содержимое.

4. Добавьте свойство `width` к стилю `#sidebar`.

   В завершенном виде стиль выглядит так:

```
#sidebar {
  position: absolute;
  top: 15px;
  left: 0;
  width: 170px;
}
```

   Повторите все действия для размещения рекламной области страницы.

5. Добавьте стиль `#adverts` внизу таблицы стилей:

```
#adverts {
  position: absolute;
  top: 15px;
  right: 5px;
  width: 125px;
}
```

   Этот стиль работает точно так же, как стиль для бокового меню, за исключением того, что вы размещаете рекламу относительно правого края страницы. Это намного лучше, чем размещать объявления в некотором месте относительно левого края страницы. Когда вы определите значение `right`, реклама всегда будет оставаться на том же самом расстоянии от правого края оболочки содержимого. Если вы измените ширину окна браузера, то реклама останется на месте.

   На данный момент страница должна быть похожей на ту, что показана на рис. 13.18, с обоими боковыми меню и рекламой, захватывающей главную историю страницы. Отрегулируйте края главного содержимого, чтобы предотвратить это наложение.

6. После стиля `#adverts`, который вы только что создали, добавьте стиль для области главной истории страницы:

```
#main {
  margin-left: 170px;
  margin-right: 135px;
}
```

   Стиль `#main` добавляет поля для области главной истории так, что они очищают левое и правое боковые меню. Теперь осталось лишь немного подкорректировать дизайн.

7. Добавьте немного отступов и границ к стилю `#main`:

```
#main {
  margin-left: 170px;
```



Рис. 13.18. Абсолютно позиционированные элементы (как два боковых меню здесь) плавают на вершине страницы, отдельно от основного потока HTML, поэтому они могут перекрывать и скрывать другие элементы

```
  margin-right: 135px;
  padding: 0 15px 15px 20px;
  border: 1px solid #666666;
  border-top: none;
  border-bottom: none;
}
```

Отступы добавляют немного места так, что текст не прикасается к боковым меню. Вы можете достичь того же самого, увеличив левое и правое поля в предыдущем шаге, но тогда вам также придется добавить изящную границу к левому и правому краям области, способствуя тем самым визуальному разделению трех созданных столбцов.

Заметьте небольшое сокращение в этом стиле. Во-первых, свойство `border` настраивает границу для всех четырех сторон области; последние два описания отключают границы сверху и снизу. Вы можете достичь этого же результата заданием двух свойств (`border-left` и `border-right`), но тогда вам придется повторять значения (`1px solid #666666`). Если вы хотите изменить цвет, толщину или стиль обеих границ, то должны отредактировать два свойства. Таким образом, всего одна строка кода заботится о левой и правой границах.

Разметка почти завершена. Осталась последняя деталь: когда вы предварительно просматриваете страницу в Internet Explorer 6, вы видите, что левое боковое

меню отдалено — находится на расстоянии 185 пикселов! Да, это очередная ошибка браузера. К счастью, она легко устраняется. Чтобы левое боковое меню вело себя корректно (см. рис. 13.17, *4*), задайте содержащему его тегу `<div>` специфическое для Internet Explorer свойство, известное как `layout` (см. врезку «Информация для опытных пользователей» в разд. «Обработка ошибок в Internet Explorer 6» гл. 12).

8. Добавьте свойство `width` к стилю `#contentWrapper`:

```
#contentWrapper {
  position: relative;
  clear: both;
  width: 100%;
}
```

Страница теперь правильно разбита на три столбца (рис. 13.19) с меньшим количеством шагов, чем при работе с баннером. Предварительно просмотрите ее в браузере и расширьте окно. Вы увидите, как гибкий дизайн позволяет странице соответствовать окну любой ширины.



Рис. 13.19. Создание гибкого дизайна на три столбца с абсолютным позиционированием требует всего нескольких действий, но основная концепция сводится к размещению двух дальних столбцов и заданию левого и правого полей для среднего столбца

Если вы предпочитаете дизайн с фиксированной шириной, то базовая структура этой страницы позволяет легко создать это. Просто установите ширину для

тега , который содержит остальные теги на странице (см. рис. 13.17, *1*), следующим образом:

```
#wrapper {
  width: 760px;
  margin: 0 auto;
}
```

Свойство margin здесь размещает разметку посередине страницы. Если вы предпочитаете выравнивание по левой стороне окна браузера, просто пропустите это свойство.

Окончательная версия этой программы-примера находится в папке 13_finished.

14 CSS для распечатываемых веб-страниц

Не всем нравится читать с монитора компьютера. Все чаще и чаще посетители Интернета распечатывают веб-страницы. Многие люди наслаждаются сайтами, сидя за обеденным столом, находясь в поезде или лежа в парке на траве в солнечный день. Распечатка квитанции после совершения покупки в Сети также становится распространенным делом. Что же происходит с вашим тщательно разработанным дизайном, когда чернила попадают на бумагу? Белый текст на черном фоне может привести к пустой трате чернил, а некоторые браузеры, возможно, даже не распечатают фон. Действительно ли пользователям нужно видеть навигационную панель вашего сайта распечатанной? Вероятно, нет.

Веб-дизайнеры обычно выходили из этого затруднительного положения, создавая отдельные, благоприятные для печати, версии своих сайтов, то есть, по существу, копию сайта, отформатированную специально для печати. Однако это означает не только больший объем работы, но и изменение множества файлов каждый раз, когда страница нуждается в редактировании. К счастью, CSS предлагает лучший способ — возможность сделать так, чтобы страница выглядела одним образом, когда отображается на экране, и другим — при печати (рис. 14.1). Использование специальных стилей для принтера позволяет сэкономить бумагу и чернила, давая возможность посетителям распечатывать лишь необходимую информацию и оставляя при этом все визуальные излишества на экранах мониторов. Заметьте, как на рис. 14.1 печатная версия расширилась и занимает всю ширину страницы. Кроме того, в ней отсутствуют логотип, навигационная панель и реклама. В чем же здесь секрет? В аппаратно-зависимых таблицах стилей.

Как работают аппаратно-зависимые таблицы стилей

Создатели CSS основательно подошли к разработке аппаратно-зависимых таблиц стилей. Они учитывали все возможные способы, которыми люди могли просматривать сайты. Разработчики понимали, что, хотя большинство пользователей просматривают страницы на мониторе компьютера, иногда они могут захотеть и распечатать текст с сайта. Вдобавок у новых устройств для интернет-серфинга,

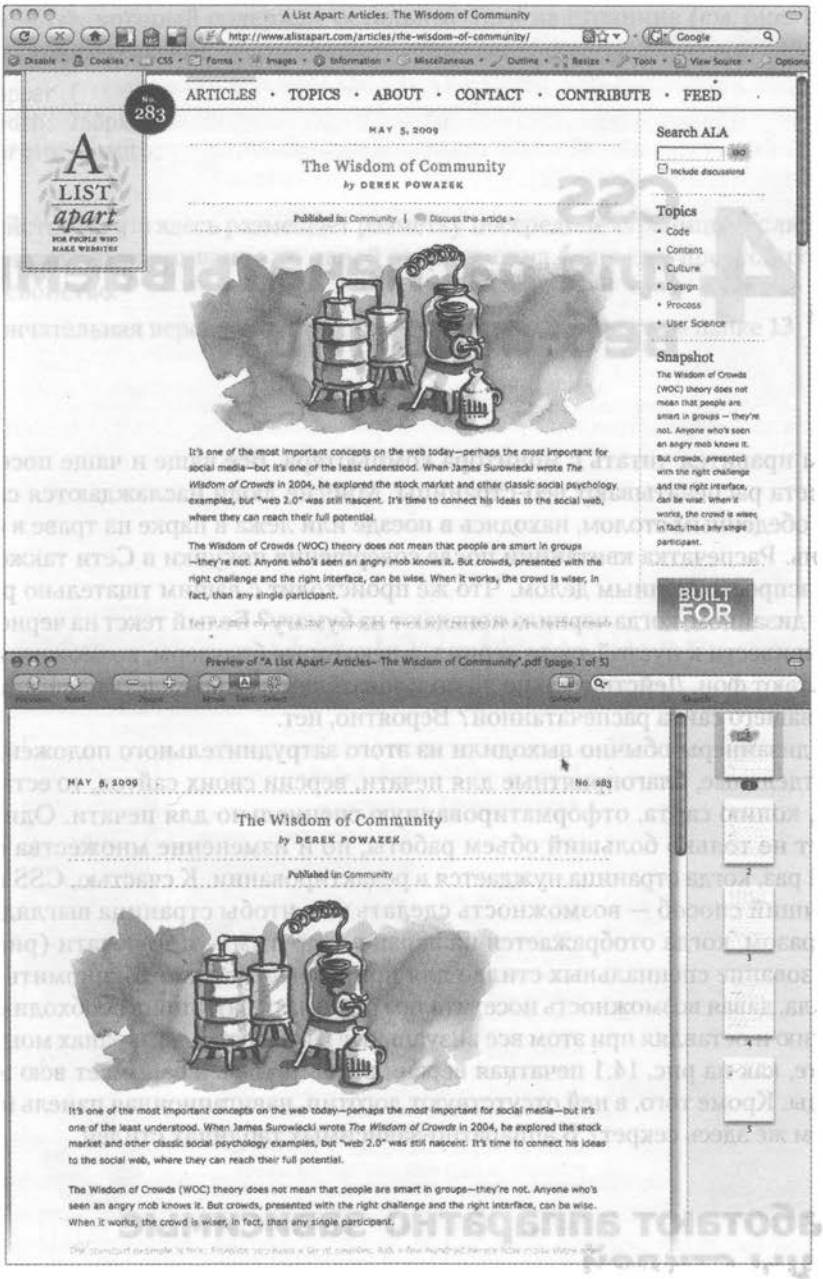


Рис. 14.1. Страница сайта, посвященного веб-разработке (A List Apart), выглядит привлекательной на экране *(вверху)* и простой и понятной на бумаге *(внизу)*

таких как мобильные телефоны, карманные компьютеры и телевизоры, есть свои собственные уникальные требования, когда дело доходит до работы с веб-страницами.

Чтобы обеспечить все мыслимые методы просмотра сайтов, CSS позволяет вам создавать стили и таблицы стилей, которые предназначаются для определенных типов устройств. CSS распознает десять различных типов устройств: `all`, `braille`, `embossed`, `handheld`, `print`, `projection`, `screen`, `speech`, `tty` и `tv`. Браузер применяет таблицу стилей, только когда активизирован тип устройства. Другими словами, браузер применяет одну таблицу стилей при просмотре на экране, а другую — при печати. Многие из этих типов предназначены для специализированных приложений, таких как Braille Reader (программа оптического распознавания текста для слепых пользователей), экранный диктор (для тех, кто хочет или кому нужно услышать, что содержится на странице), или телетайпа. Большинство этих типов еще не работают в реальном мире, поскольку нет устройств, которые были бы запрограммированы так, чтобы могли понимать их. Тем не менее вы должны знать три типа: `all`, `screen` и `print`.

- `all` — относится к каждому типу устройства. Когда стиль или таблица стилей использует тип `all`, каждое устройство, получающее доступ к странице, применяет те же самые стили. Принтеры и мониторы пытаются отформатировать страницу подобным образом (стили на самом деле используют этот тип по умолчанию, как только вы включаете их в страницу или получаете из внешней таблицы стилей, так что вам не нужно указывать «все типы устройств» при добавлении таблицы стилей на страницу).
- `screen` — данный тип отображается только на мониторе. Когда вы определяете этот тип, браузер игнорирует такие стили при печати страницы. Этот тип устройства позволяет вам отделить те стили, которые выглядят замечательно на экране, но ужасно — на бумаге, например задание белого текста на черном фоне.
- `print` — применяется только при печати страницы. Тип устройства `print` позволяет создать стили, которые используют благоприятные для печати размеры шрифта, цвета, графику и т. д.

ПРИМЕЧАНИЕ

Браузер Opera понимает тип устройства projection, когда находится в полноэкранном режиме. Чтобы больше узнать об этой особенности, зайдите на страницу www.opera.com/support/tutorials/operashow/.

Один из подходов состоит в создании стилей сначала лишь для экрана и присоединении их каким-либо из методов, описанных далее (таких как `internal` или `external`, `linked` или `imported`). На начальном этапе эти стили работают и для монитора, и для принтера. Затем вы создаете таблицу стилей только для принтера, которая применяется при печати страницы. Она отменит любые основные стили, которые негативно влияют на вид страницы при ее печати. Этот подход рассматривается в разд. «Создание таблиц стилей для печати» данной главы. В качестве альтернативы вы можете создать две различные аппаратно-зависимые таблицы стилей: одну для экрана, а другую для печати — и присоединить их к вашим вебстраницам, как это описано далее.

ПРИМЕЧАНИЕ

Еще один распространенный метод — создание трех таблиц стилей (одной для принтера, другой — для экрана и третьей — со стилями, которые должны появиться и при печати, и на экране монитора). Вы указываете типы устройств для таблиц стилей принтера и экрана и присоединяете третий, разделяемый набор стилей, как делали это обычно.

Как добавлять аппаратно-зависимые таблицы стилей

Аппаратно-зависимые таблицы стилей — это просто таблицы стилей CSS: они могут быть или внутренними, или внешними. Однако если вы хотите, чтобы браузер применил стили только для определенного устройства, например для экрана или принтера, то должны добавить таблицу стилей к своей странице немного другим способом.

Определение типа устройства для внешней таблицы стилей

Чтобы присоединить внешнюю таблицу стилей при определении конкретного типа устройства, используйте тег `<link>` с атрибутом `media`. Чтобы присоединить таблицу стилей, которая должна использоваться только при печати, добавьте такой HTML-код к своей веб-странице:

```
<link rel="stylesheet" type="text/css" media="print" href="print.css"/>
```

ПРИМЕЧАНИЕ

Формально CSS также позволяет определить тип устройства, когда вы используете правило @import для присоединения внешней таблицы стилей (см. разд. «Внешние таблицы стилей» гл. 2) таким образом: @import url(print.css) print;. Но, поскольку Internet Explorer не понимает этот код, вы должны избегать его использования.

Если вы не определите тип устройства, браузер решит, что вы имеете в виду все устройства, и будет использовать таблицу стилей для отображения на экране, при печати и т. д. Кроме того, вы можете определить множество типов устройств, разделяя их запятыми. Присоединенная внешняя таблица стилей, предназначенная для нескольких устройств, могла бы быть такой:

```
<link rel="stylesheet" type="text/css" media="screen, projection, handheld"
href="screen.css"/>
```

Вам, вероятно, не нужно указывать несколько типов, пока браузеры не начнут распознавать их все.

СОВЕТ

Когда вы создаете и проверяете таблицы стилей для принтера, оставьте атрибут media="print" и отключите все таблицы стилей, предназначенные только для экрана. Например, измените media="screen" на media="speech". Эта методика позволит вам просматривать страницу в браузере в том виде, как она будет выглядеть для печати. Как только таблица стилей для печати станет смотреться приемлемо, укажите для нее тип print и подключите любую таблицу стилей для отображения на экране.

Определение типа устройства внутри таблицы стилей

Вы можете также включить определенные аппаратно-зависимые стили непосредственно внутри таблицы стилей, используя правило `@media`. Возможно, вы захотите добавить к внутренней таблице несколько стилей, характерных для печати. Или решите хранить все стили в отдельной внешней таблице и просто добавить несколько стилей, предназначенных только для принтера. Вы можете сделать это, используя правило `@media`, таким образом:

```
@media print {
  /* описывайте стили для печати здесь */
}
```

Не забудьте указать закрывающую фигурную скобку (в последней строке), иначе правило не будет работать. Вот пример использования `@media` для включения двух стилей, предназначенных только для принтера:

```
@media print {
  h1 {
    font-size: 24pt;
  }
  p {
    font-size: 12pt;
  }
}
```

Фактически не имеет никакого значения, помещаете вы все стили в отдельный файл и используете правило `@media` или определяете специфические аппаратно-зависимые стили в их собственных внешних таблицах стилей (например, `screen.css` и `printer.css`). Добавление всех ваших стилей, предназначенных только для печати, в их собственную внешнюю таблицу стилей `printer.css` намного облегчает поиск и редактирование этих стилей.

Создание таблиц стилей для печати

Вы должны видеть, как распечатываются страницы вашего сайта, перед тем как с головой погрузиться в реконструкцию стилей для печати. Часто вся информация на веб-странице печатается без проблем, так что, вероятно, вам не придется добавлять к сайту таблицу стилей для принтера. Однако в некоторых случаях, особенно при использовании большого CSS-кода, после распечатки страницы выглядят ужасно. Так, поскольку браузеры не печатают фоновые изображения, только если им не указать делать это, у вас может образоваться много пустого пространства в тех местах, где были эти изображения. Но даже если страница выглядит на бумаге так же, как и на экране, у вас есть множество способов улучшить качество печатной версии путем добавления определенных стилей, предназначенных только для печати (рис. 14.2).

СОВЕТ

Чтобы просмотреть до печати, как будет выглядеть страница на бумаге, можно использовать команду Print Preview (Предварительный просмотр) браузера. В Windows она обычно доступна через меню File ▸ Print Preview (Файл ▸ Предварительный просмотр). В Mac вы сначала выбираете File ▸ Print (Файл ▸ Печать), а затем в появившемся окне — Print Preview (Предварительный просмотр). Используя предварительный просмотр, вы можете проверить, не слишком ли широка страница, чтобы соответствовать одному листу бумаги, и увидеть, где происходит обрыв страницы.

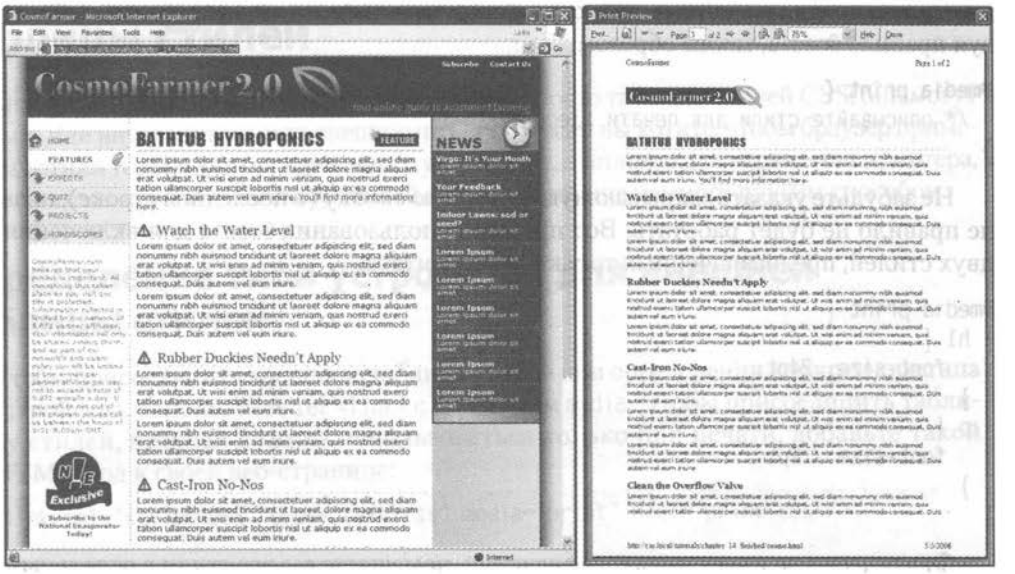


Рис. 14.2. С помощью таблицы стилей для принтера вы можете убрать боковое меню, навигационные панели и другое содержимое, разработанное для просмотра сайтов *(слева)*. Результатом будет просто отформатированный документ — идеальный для печати *(справа)*

Использование правила !important для отмены экранного стиля

Как говорилось ранее, часто бывает полезным создавать таблицу стилей, не указывая тип устройства (или используя код `media="all"`). Если вы готовы определить кое-какие характерные для печати правила, то можете просто создать отдельную таблицу стилей, чтобы отменить любые стили, которые не очень хорошо выглядят при печати.

Скажем, у вас есть тег `<h1>`, заданный так, чтобы выглядеть синим на экране, и вы также выбрали правила, определяющие интервал между буквами, размер шрифта и выравнивание текста. Если единственная вещь, которую вы хотите изменить для своих печатных страниц, — это использование черного цвета вместо синего, то вам не нужно создавать новый стиль с целым набором новых свойств. Просто создайте главную таблицу стилей, которая применяется в обоих случаях, и таблицу стилей для печати, который отменяет синий цвет для тега `<h1>`.

Одна проблема этого подхода состоит в том, что вы должны убедиться, что стили для принтера на самом деле имеют больший приоритет, чем основная таблица стилей. Таким образом, вам надо внимательно управлять каскадностью. Как обсуждалось в гл. 5, стили могут взаимодействовать между собой сложным образом: некоторые из них могут относиться к одному и тому же элементу и CSS-свойства этих стилей могут соединяться и отменять друг друга. Однако существует безошибочный способ убедиться в том, что одно свойство превосходит все остальные, — использование правила `!important`.

Когда вы добавите `!important` после значения свойства в CSS-коде, это конкретное свойство стиля будет доминировать в любых конфликтах с другими стилями. Добавьте правило `!important` к таблице стилей для печати, чтобы убедиться, что все теги `<h1>` будут напечатаны черными:

```
h1 {
  color: #000 !important ;
}
```

Этот стиль заголовка `h1` отменит еще более специфические стили, включая `#main h1, h1.title` или `#wrapper #main h1` из основной таблицы стилей.

Изменяем текстовые стили

Не обязательно, чтобы текст выглядел одинаково и на экране, и на бумаге. Хорошее начало создания таблицы стилей для принтера — изменение свойств `font-size` и `color`. Задание размеров в пикселах для текста не слишком удобно для принтера. Должен ли он понимать 12 пикселов как 12 пунктов? Если у вас принтер с 600 dpi (точками на дюйм), то такой текст будет маленьким и неразборчивым. И в то время, как яркий зеленый текст будет хорошо выглядеть на экране, он может стать трудночитаемым бледно-серым после печати.

Задание размеров в пикселах и em (см. разд. «Установка размера шрифта» гл. 6) имеет смысл для экранного текста, но для печати лучше использовать *пункты*. Это то, в чем Word и другие программы обработки текста измеряют высоту шрифта. На практике большинство браузеров так или иначе переводят пикселы и em в нечто более подходящее для принтеров. Базовый экранный размер шрифта для большинства браузеров — 16 пикселов — печатается как 12 пунктов. Однако нет никакого способа предугадать, как определенный браузер изменит размеры текста, так что для максимального управления печатью устанавливайте в своих таблицах стилей для печати размер шрифта в пунктах.

Чтобы все абзацы напечатались шрифтом размером 12 пунктов (общепринятый размер для печати), используйте следующее правило:

```
p {
  font-size: 12pt;
}
```

ПРИМЕЧАНИЕ

Как и с буквами em, вы не добавляете s, когда устанавливаете для шрифта размер в пунктах: 12pt, а не 12pts.

Аналогично экранные цвета часто плохо переводятся, когда печатаются на черно-белом лазерном принтере. Четкий черный текст на белом фоне намного легче читается чем, например, светло-серые буквы. Кроме того, как вы узнаете в следующем разделе, белый текст на черном фоне, хоть и выглядит очень разборчиво на экране, часто не печатается как следует. Чтобы сделать текст читабельным на бумаге, лучше печатать его черным цветом. Если хотите сделать весь текст абзаца черным, добавьте следующий стиль к своей таблице стилей для принтера:

```
p {
  color: #000;
}
```

Как было сказано в начале этого раздела, если ваша таблица стилей для печати конкурирует со стилями из другой присоединенной таблицы, используйте правило `!important`, чтобы быть уверенными в том, что стили для принтера имеют больший приоритет:

```
p {
  font-size: 12pt !important;
  color: #000 !important;
}
```

Чтобы весь текст на странице распечатался черным цветом, используйте универсальный селектор (см. подраздел «Универсальный селектор» разд. «Стилизация групп тегов» гл. 3) и правило `!important` для создания отдельного стиля, который форматирует каждый тег, устанавливая для него текст черного цвета:

```
* { color: #000 !important }
```

Конечно, этот совет применяется, только если ваш сайт печатается на черно-белом принтере. Если вы знаете, что большинство пользователей сайта работают с цветными принтерами, то можете сохранить все цвета текста или изменить их так, чтобы они стали даже более яркими при печати.

Стилизация фонов для печати

Добавление фоновых изображений и цветов к навигационным кнопкам, боковым меню и другим элементам страницы придает контрастность и визуальную привлекательность вашим веб-страницам. Но вы не можете быть уверены, что фон правильно отобразится, когда эти страницы будут печататься. Поскольку цветной фон съедает чернила и порошок принтера, большинство браузеров обычно не отправляют его на печать, а многие посетители Сети не включают фон при печати, даже если их браузеры имеют такую возможность.

Вдобавок, даже если фон на самом деле напечатается, он может конкурировать с любым текстом, наложенным на него. Это особенно верно, если текст сильно отличается от цветного фона на экране, но смешивается с фоном при печати на черно-белом принтере.

ПРИМЕЧАНИЕ

Белый текст на черном фоне обычно вызывает самую большую проблему — у вашего посетителя при печати получится чистый белый лист. К счастью, в современных браузерах есть возможность изменять белый текст на черный (или серый), когда идет печать без фона.

Удаление элементов фона

Самый легкий способ решить проблему с фоном — просто удалить его из таблицы стилей для печати. Скажем, вы полностью изменяете заголовок так, чтобы текст стал белым, а у фона был темный цвет. Если стиль, который создает этот эффект, называется `.headHighlight`, то продублируйте то же самое название в таблице стилей только для печати:

```
.headHighlight {
  color: #000;
  background: white;
}
```

Этот стиль устанавливает фону белый цвет — цвет бумаги. Кроме того, чтобы получить четкий печатный текст, этот стиль выбирает для шрифта черный цвет.

БРИЛЛИАНТ БЕЗ ОГРАНКИ

Убить двух зайцев

Вы можете применять свойство `background-color`, чтобы установить белый фоновый цвет, таким образом: `background-color: white`. Вы получаете тот же самый результат, используя краткую версию этого метода: `background: white`.

Помните, что свойство `background` (см. разд. «Сокращенный вариант свойства background» гл. 8) может определить фоновое изображение, то, как это изображение повторяется, и его положение.

Если же вы опустите любые значения, используя краткую версию, браузер установит для свойства значение по умолчанию. Другими словами, если вы опустите значение для фонового изображения, то браузер установит значение `none`. Значит, такая декларация, как `background: white;`, не только устанавливает фоновый цвет белым, но и удаляет любые фоновые изображения. Используя краткое свойство `background`, вы убиваете сразу двух зайцев: устанавливаете белый фон и удаляете изображения, написав при этом совсем небольшой код.

Оставление элементов фона

Если вы не хотите избавляться от фона, то можете оставить его в надежде, что пользователи установят свои браузеры так, чтобы те печатали фон. Если вы оставляете элементы фона в таблице стилей для печати, а над ними появляется текст, то убедитесь, что текст будет читабельным как с фоном, так и без него.

Другая вещь, которую надо рассмотреть, — использование фоновых изображений: нужно ли их печатать? Скажем, вы поместили логотип компании как фоновое изображение тега `<div>`, используемого в качестве баннера страницы. Поскольку логотип находится на заднем плане (в фоне), он может не распечататься. Ваша компания или клиент, возможно, не обрадуются, если на каждой странице

распечатанного сайта не будет логотипа. В этом случае у вас есть несколько вариантов. Вы можете вставить логотип как обычный тег <img> вместо фонового изображения. Этот способ работает, но что, если логотип выглядит замечательно на цветном мониторе и не очень хорошо, когда печатается на черно-белом принтере? Другой способ состоит в том, чтобы оставить один логотип в фоновом изображении и добавить другой, более благоприятный для печати, используя тег <img>. Затем вы спрячете изображение с экрана, но покажете его при печати. Рассмотрим, как это делается.

1. Добавьте тег <img> в то место в коде HTML, где вы хотите, чтобы он появился при распечатке:

```
<img src="logo.png" height="100" width="100" id="logo" />
```

2. Затем в главной таблице стилей (той, которая применяется при отображении страницы на экране) добавьте стиль, который спрячет это изображение:

```
#logo { display: none; }
```

3. В таблице стилей для печати добавьте в конец один стиль для показа изображения:

```
#logo { display: inline; }
```

Теперь этот логотип на экране появляться не будет, а при печати — будет.

СОВЕТ

Если вы хотите быть абсолютно уверенными, что фоновое изображение распечатается, можете применить еще один хитрый подход CSS, чтобы преодолеть нежелание браузера печатать фоновые изображения. Его описание можно найти по адресу http://web-graphics.com/mtarchive/001703.php.

ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

Показываем ссылки при печати

Предположим, сотрудница вручает вам распечатку очаровательной статьи, которую она нашла в Сети. Вы читаете и сталкиваетесь с таким переходом: «И вот я нашел секрет вечной жизни здесь». Подчеркивание говорит вам о том, что здесь расположена ссылка, которая разоблачит описываемый секрет. Но на листке бумаги, конечно, у вас нет возможности перейти по этой ссылке.

Чтобы предотвратить эту проблему на своих собственных страницах, вы можете сделать так, чтобы связанные URL печатались наряду с остальной частью текста: «секрет вечной жизни здесь (http://www.pyramind_scam.com/)». Используя усовершенствованный селектор `:after` и CSS-свойство `content`, вы можете напечатать текст, который не появляется на экране в конце стилизуемого элемента. К сожалению, селектор `:after` и свойство `content` не работают в Internet Explorer версии 7. Но они работают в Internet Explorer 8, Firefox и Safari, и, таким образом, вы можете печатать URL для тех пользователей, которые работают с этими браузерами.

Чтобы сделать это, добавьте стиль к таблице стилей для печати, который напечатает URL после каждой ссылки.

Вы можете даже добавить другие текстовые элементы, например круглые скобки, чтобы это выглядело лучше:

```
a:after {
  content: " (" attr(href) ") ";
}
```

ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

Однако этот CSS-код не различает внешние или внутренние ссылки, так что он также печатает зависимые от документа ссылки на другие страницы того же сайта: «Посетите домашнюю страницу (../../index.html)». Используя селекторы атрибутов (см. гл. 3), вы можете заставить стиль печатать только абсолютные URL (те, которые начинаются с http://), следующим образом:

```
a[href^="http://"]:after {
  content: " (" attr(href) ") ";
}
```

К счастью, несмотря на то, что этот стиль использует «еще не завершенные» правила CSS 3, все браузеры, которые понимают селектор :after и свойство content, понимают и этот селектор атрибута.

Если вы используете относительные ссылки на своем сайте, то можете применять другой способ печати корректных, полных URL-адресов. Для получения подробной информации просмотрите статью по адресу www.alistapart.com/articles/goingtoprint/.

Скрытие нежелательных областей страницы

Веб-страницы часто нагружаются вспомогательными информационными средствами, такими как навигационные панели, боковое меню, полное полезных ссылок, поля для поиска и т. д. Эти элементы хороши для веб-серфинга, но они мало чем полезны на бумаге. Ваша веб-страница может также содержать объявления, видео и другие дополнительные детали, на которые люди не хотят впустую тратить дорогие чернила и тонер. Вы можете помочь своим пользователям, убрав эти экранные излишества из того содержимого, которое посетители действительно хотят напечатать.

Как рассказывалось ранее в этой книге, один из способов разбить страницу состоит в том, чтобы определить теги для различных элементов: баннеров, основной навигации, содержимого, уведомления об авторском праве и т. д. Разрабатывая каждый тег , используя относительное или абсолютное позиционирование, вы можете разместить различные элементы страницы там, где вы хотите их видеть. Вы можете использовать такую же структуру для создания таблицы стилей, предназначенной только для печати, где с помощью свойства display скрываются нежелательные элементы.

Устанавливая значение display в none, вы можете заставить браузер удалить разработанный элемент со страницы. Так, чтобы препятствовать печати бокового меню, просто переопределите этот стиль в таблице стилей для печати и установите его свойству display значение none:

```
#sidebar {
  display: none;
}
```

Для большинства страниц требуется, чтобы таблица стилей для печати показывала только самые основные информационные элементы: логотип, главное содержимое, сведения об авторском праве, скрывая все остальное. Вы можете легко скрыть множество элементов таким селектором группы:

```
#banner, #mainNav, #sidebar, #ads, #newsLinks {
  display: none;
}
```

Помните, что эти стили относятся к вашей таблице стилей для печати, а не к главной таблице стилей, иначе вы никогда не увидите навигацию, баннеры или другие важные области своей экранной страницы. Однако время от времени вы можете захотеть скрыть что-нибудь от вашей основной таблицы стилей и показать это *только* при печати.

Скажем, вы помещаете логотип своего сайта как фоновое изображение в области баннеров страницы. Вы, возможно, захотите определить его так, чтобы наверху изображения логотипа, где ничего нет, были текст или ссылки. Вы (ваш босс или клиент), конечно, хотите, чтобы логотип появлялся на всех печатных страницах, но, поскольку не все браузеры печатают фоновые изображения, вы не можете быть уверены, что логотип будет напечатан. Одно из решений этой проблемы состоит в том, чтобы вставить тег `<img>`, содержащий измененную, благоприятную для печати версию логотипа; добавить ID к изображению; создать стиль ID в главной таблице стилей со свойством `display`, которому установлено значение `none`; а затем задать свойству `display` того же ID значение `block` в таблице стилей для печати. Вот! Логотип появится только при печати.

Добавление разрывов страницы для печати

Версия 2.1 стандарта вложенных таблиц стилей включает много CSS-свойств, нацеленных на лучшее форматирование печатной веб-страницы: от установки ориентации страницы до определения полей и размера бумаги (полный список вы найдете на сайте **www.w3.org/TR/CSS21/page.html**). К сожалению, современные браузеры распознают очень немногие из этих стилей для печати.

Два широко признанных свойства — это `page-break-before` и `page-break-after`. Первое свойство говорит браузеру вставить разрыв страницы перед данным стилем. Скажем, вы хотите, чтобы определенные заголовки всегда появлялись наверху страницы, как, например, названия различных разделов длинного документа (рис. 14.3). Вы можете добавить свойство `page-break-before: always` к стилю для форматирования этих заголовков. Аналогично, чтобы элемент появился в качестве последнего объекта на странице, добавьте свойство `page-break-after: always` к стилю этого элемента.

Свойство `page-break-before` также полезно для больших изображений, так как некоторые браузеры позволяют себе печатать их на двух страницах, затрудняя просмотр всего изображения целиком. Если у вас есть одна страница с тремя абзацами текста, за которыми идет изображение, то браузер напечатает часть изображения на одной странице, а оставшуюся часть — на второй. Если вы не хотите, чтобы пользователям понадобился скотч, чтобы собрать два изображения в одно, то применяйте свойство `page-break-before`. В этом случае рисунок напечатается на новой странице, которой он соответствует.

Рассмотрим быстрый способ, как использовать эти свойства в своих интересах. Создайте два класса стилей, названные, например, `.break_after` и `.break_before`:

```
.break_before { page-break-before: always; }
.break_after { page-break-after: always; }
```

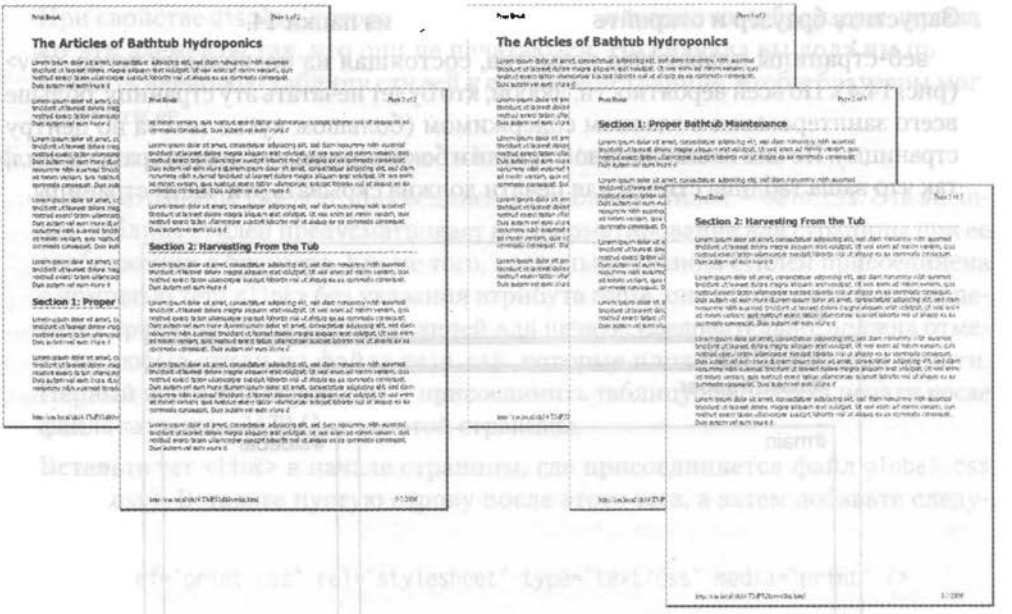


Рис. 14.3. При печати страницы браузер разделяет ее содержимое на множество страниц, чтобы вместить на каждой как можно больше текста *(слева)*. Если же вы хотите, чтобы разрывы страниц находились в более логичных местах, используйте свойство page-break-before *(справа)*

Теперь вы можете выборочно применять эти стили к элементам, которые должны печататься наверху или внизу страницы. Если вы хотите, чтобы определенный заголовок был напечатан наверху страницы, используйте такой стиль: `<h1 class="break_before">`. Даже если к элементу уже применен класс, вы можете добавить дополнительный класс таким образом: `<h1 class="sectionTitle break_before">` (об этой полезной методике читайте в разд. «Организация стилей и таблиц стилей» гл. 15).

Обучающий урок: создание таблицы стилей для печати

В этой программе-примере мы создадим таблицу стилей для печати. Чтобы печатная версия веб-страницы имела лучший вид, добавим стили, которые удаляют нежелательные элементы и фоны страницы, изменяют форматирование текста и печатают URL-адреса любых ссылок на странице.

Чтобы приступить, загрузите файлы примеров. Как это сделать, рассказывается в конце гл. 2. Файлы для этого примера находятся в папке 14.

Удаление ненужных элементов страницы

Прежде чем приступить к работе, вы должны определить, как размечена страница, чтобы можно было решить, какие элементы нужно печатать.

1. Запустите браузер и откройте файл `print.html` из папки **14**.

   У веб-страницы плавающая разметка, состоящая из нескольких тегов `<div>` (рис. 14.4). По всей вероятности, любой, кто будет печатать эту страницу, больше всего заинтересован в главном содержимом (большом куске текста по центру страницы). Печать навигационной панели и бокового меню — пустая трата чернил, так что ваша таблица стилей для печати должна скрывать эти части страницы.

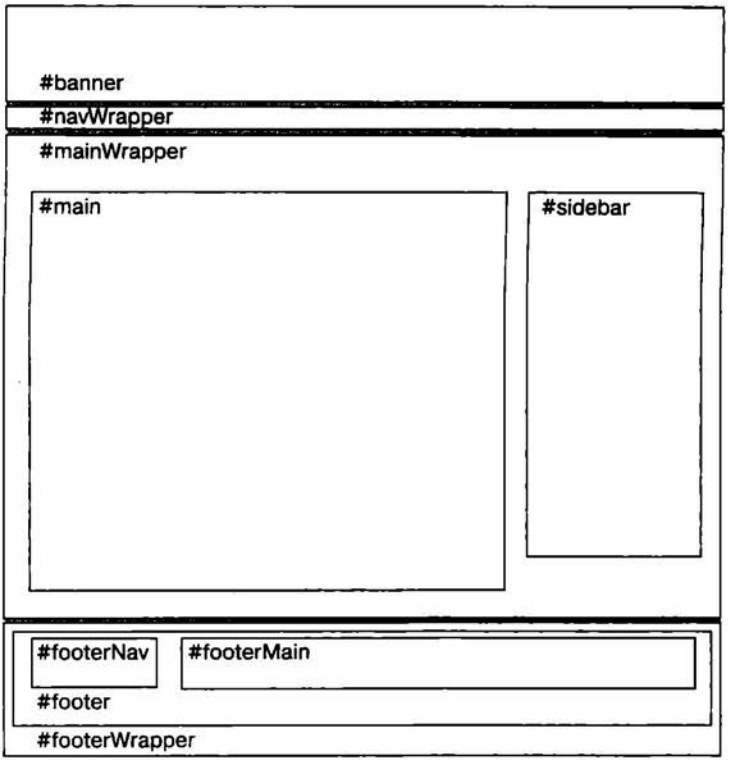


Рис. 14.4. Макет страницы позволяет управлять размещением элементов. При печати страницы лучше, чтобы некоторые элементы вовсе не появлялись. Баннер, навигационная панель и боковое меню не дадут никакой полезной информации в распечатанном документе

2. В текстовом редакторе создайте новый файл, названный `print.css`, и сохраните его в папке **14** вместе с основной таблицей стилей.

   В вашей новой таблице стилей для печати первоочередная задача состоит в том, чтобы скрыть навигационную панель и другие части страницы, которые вы не хотите печатать.

3. Используя свойство `display`, создайте новую группу селекторов, которые скрывают навигационные элементы и боковое меню:

```
#sitetools, #nav, #news {
  display: none;
}
```

При свойстве display, которому установлено значение none, браузеры скрывают эти элементы так, что они не печатаются. Но сначала вы должны присоединить внешнюю таблицу стилей к своей веб-странице, чтобы браузеры могли найти ее.

4. В текстовом редакторе откройте файл print.html из папки 14.

   У этой страницы уже есть присоединенная таблица стилей — main.css. Эта внешняя таблица стилей предусматривает все форматирование для страницы при ее отображении в браузере. Кроме того, поскольку таблица стилей присоединена с помощью тега <link> без указания атрибута media, она применяется и при печати страницы. Ваша таблица стилей для печати, следовательно, должна отменить любые стили из файла main.css, которые плохо выглядят при печати. Первый шаг в этом процессе — присоединить таблицу стилей для печати *после* файла main.css в HTML-коде этой страницы.

5. Вставьте тег <link> в начале страницы, где присоединяется файл global.css (main.css). Вставьте пустую строку после этого тега, а затем добавьте следующее:

   ```
   <link href="print.css" rel="stylesheet" type="text/css" media="print" />
   ```

   Если свойства из двух стилей с одинаковым названием конфликтуют, то побеждают свойства последней присоединенной к странице таблицы стилей, так что этот тег <link> должен идти *после* другого тега <link>. Таким образом, если у файла main.css есть класс, названный .copyright, который создает белый 10-пиксельный текст на черном фоне, вы можете создать другой стиль, названный .copyright, в таблице стилей для печати — с черным текстом размером 12 пунктов на белом фоне. Хотя два стиля имеют одно название, свойства из таблицы стилей для печати победят, потому что она присоединена последней (см. разд. «Особенности механизма каскадности: какие стили имеют преимущество» гл. 5 для получения более полной информации об этом механизме каскадности).

6. Сохраните файлы print.css и print.html, а затем просмотрите print.html в браузере.

   Вид страницы не отличается от того, который был в шаге 1 этого урока. Так происходит потому, что вы еще не печатали страницу. Вы можете увидеть воздействие таблицы стилей для печати, используя команду Print Preview (Предварительный просмотр) браузера.

7. Если вы пользуетесь операционной системой Windows, выберите File ▸ Print Preview (Файл ▸ Предварительный просмотр). Любителям Mac надо выбрать File ▸ Print (Файл ▸ Печать), а затем в появившемся окне Print (Печать) нажать кнопку Print Preview (Предварительный просмотр).

   В окне Print Preview (Предварительный просмотр) вы увидите, что правое боковое меню и навигационные элементы исчезли. Но дизайн все равно еще не впечатляет. Главное содержимое не наполняет страницу так, как должно. Далее рассмотрим, как устранить оставшиеся проблемы.

Установка разметки

Пока основное содержимое и нижний колонтитул с записью об авторских правах некорректно соответствуют печатной странице: основное содержимое не дотягивается до правого края страницы, а запись имеет отступ от левого края. Эти два элемента выглядели бы намного лучше, если бы заполняли всю печатную область.

В настоящее время разметкой управляют два стиля. На рис. 14.4 вы можете видеть, что страница разделена на несколько зон, каждая из которых создается собственным отдельным тегом . Стили #mainWrapper и #footer центрируют области с основным содержимым и с нижним колонтитулом и устанавливают для них ширину 900 пикселов. Кроме того, стиль #main имеет заданные значения ширины, а стиль #footerMain — левое поле. Поскольку вы не знаете, на бумаге какого размера будет распечатана эта страница, вы должны избавиться от всех заданных значений ширины и удалить все поля.

1. Вернитесь к текстовому редактору и файлу print.css. Добавьте один новый стиль, убирающий установленные значения ширины и поля тех областей страницы, которые будут распечатаны:

```
#banner, #mainWrapper, #footer, #main {
  width: auto;
  margin: 0;
  padding: 0;
}
```

   Первое объявление — width:auto — затрагивает несколько областей страницы. Оно отменяет установку ширины, равной 900 пикселам, для основного текста и нижнего колонтитула в файле main.css и оставляет точную ширину для браузера. Значение auto просто позволяет тегу заполнить всю доступную ширину, поэтому за размер бумаги можно не беспокоиться. Два других объявления — margin и padding — устраняют пространство вокруг этих разделов div.

   Содержимое с записью об авторском праве, представленное внутри тега с идентификатором footerMain, не имеет установленной ширины, но имеет левый отступ. При печати он будет выглядеть нелепо, поэтому следует устранить его.

2. Добавьте такой стиль к таблице стилей print.css:

```
#footerMain {
  margin: 0;
}
```

   Вы спрятали правое боковое меню, для позиционирования которого применялось свойство float, но основное содержимое все еще плавающее, чего не должно быть для печатной страницы.

3. Добавьте новый стиль в файл print.css:

```
#main {
  float: none;
}
```

При печати плавающих разметок наружу могут выйти различные проблемы. Как мы обсуждали ранее, плавающие элементы могут внезапно выскочить за пределы их блока-контейнера, не позволяя фонам и границам отображаться корректно. Решение заключается в следующем: добавьте объявление `overflow: hidden;` к стилю блока-контейнера. Тем не менее скрытие переполняющегося содержимого иногда приводит к тому, что печатный материал не отображается. Поэтому вы должны выключить это для двух стилей.

4. Добавьте еще один стиль к таблице стилей `print.css`:

```
#mainWrapper, #footer {
  overflow: visible;
}
```

Данный стиль гарантирует, что содержимое внутри этих двух разделов `div` полностью отображается.

У нас также есть несколько фоновых цветов и изображений, разбросанных по странице. Иногда фоновые изображения и цвета печатаются, но чаще всего этого не происходит. Все зависит от браузера и настроек пользователей. Некоторые браузеры не печатают фоновые элементы вовсе, другие же могут печатать их, но дают людям право выбора. Печать фона полезна, когда бумажный вариант страницы должен выглядеть как экранная версия. Но если ваш фон был создан просто для украшения и станет пустой тратой чернил, окажите услугу своим посетителям и отключите его.

5. Добавьте следующий стиль к таблице стилей `print.css`:

```
html, body, #banner, #footerWrapper {
  background: #FFF;
}
```

При просмотре на экране у страницы выделяются различные фоновые цвета и изображения. Например, баннер имеет фоновое изображение для заголовка «About Us», а нижний колонтитул — фиолетовый цвет в качестве фонового. Этот стиль устанавливает белый цвет фона для страницы и баннера и удаляет графику (я уже рассказывал о том, почему фоновое изображение исчезает).

Область с логотипом сайта также выглядит не очень хорошо при печати. Она слишком высокая, так как была расширена за счет применения фонового изображения, которое не появится при печати. Вы сможете отрегулировать высоту и улучшить внешний вид этого верхнего раздела, центрировав логотип и добавив линии над и под ним.

6. Добавьте новый стиль к таблице стилей `print.css`:

```
#banner {
  height: auto;
  text-align: center;
  border-bottom: 2pt solid #000;
  border-top: 2pt solid #000;
  padding: 10pt 0;
  margin-bottom: 15pt;
}
```

Первое свойство устраняет высоту баннера, позволяя ему принять размер в соответствии с содержащим его тегом . Другими словами, она будет такой же, как и высота логотипа внутри его. Свойство text-align центрирует логотип, придавая ему классический вид. Благодаря границам выше и ниже логотипа появляются линии, а отступ добавляет пространство между логотипом и границами. Обратите внимание, что в этих стилях используются пункты, поскольку это более привычный способ измерения элементов при печати.

Вы можете сохранить этот файл. Просмотрите print.html в браузере и используйте функцию Print Preview (Предварительный просмотр), чтобы увидеть, какой будет печатная версия страницы.

Переформатирование текста

В то время как цветной текст и заданные в пикселах шрифты могут хорошо отображаться на экране, лазерные принтеры лучше понимают шрифты, размеры которых указаны в пунктах. Кроме того, черный текст смотрится лучше на белой бумаге. В этом разделе мы отрегулируем текст так, чтобы он выглядел при печати как можно лучше.

1. В файле print.css добавьте следующее CSS-правило:

```
* {
  color: #000000 !important;
}
```

Этот стиль — CSS-эквивалент кувалды. По существу, он указывает каждому тегу использовать черный текст независимо ни от чего. Использование символа * (универсальный селектор) — быстрый способ определить стиль для каждого элемента на странице (см. подраздел «Универсальный селектор» разд. «Стилизация групп тегов» гл. 3), в то время как правило !important решает любые конфликты, вызванные каскадностью. Таким образом, хотя * — не очень специфический стиль, свойство color здесь превосходит то же самое свойство в более характерных стилях, таких как #main h1 или #nav #mainNav a.

После этого нужно установить новые размеры шрифта для текста.

2. Добавьте три следующих стиля:

```
h1 {
  font-size: 30pt !important;
}
h2 {
  font-size: 16pt !important;
  font-weight: bold !important;
}
p {
  font-size: 11pt !important;
}
```

Эти стили задают каждому из этих тегов более благоприятный для печати размер шрифта. Добавление правила !important приводит к тому, что указанные

размеры всегда применяются, независимо от любых конфликтов со стилями, находящимися в таблице стилей main.css.

ПРИМЕЧАНИЕ

В нашем случае h1, h2 и p — единственные теги, которые печатаются со страницы print.html, Ваши страницы могут нуждаться в переопределении размеров текста для других тегов, таких как списки и т. д.

Просто ради интереса добавьте несколько стилей, чтобы изменить размер шрифта записи об авторском праве и добавить линию границы над ней.

3. Добавьте два следующих стиля:

```
#footerMain {
  margin-top: 15pt;
  border-top: 1pt solid #000;
  padding-top: 5pt;
}
#footerMain p {
  font-size: 9pt !important;
}
```

Все свойства CSS в этих стилях должны быть устаревшими. Они регулируют пространство над нижним колонтитулом, добавляют линию границы, немного пространства между границей и запись об авторском праве, делают текст записи меньше. Обратите внимание на объявление !important в стиле #footerMain p. Вы должны добавить его, так как в стиле p из шага 2 есть !important. Поскольку у #footerMain p больший приоритет (мы обсуждали эту концепцию подробно ранее), чем у p, его объявление !important в итоге побеждает.

Отображение URL

В качестве последнего штриха добавим еще один стиль, который печатает URL-адрес рядом с текстом любой ссылки на странице. Таким образом, экранный текст Click here to found out more напечатается как Click here to found out more (http://www.outmore.com/), так что любой человек, читающий печатную версию, сможет посетить сайт, на который указывает ссылка. Эта методика использует некоторый усовершенствованный CSS-код, который Internet Explorer версии 6 (и 7 на момент написания книги) не поймет, но она тем не менее не причинит никакого вреда этому браузеру. Однако это огромное улучшение для пользователей, которые печатают ваш сайт из браузеров IE 8, Firefox и Safari.

1. Добавьте еще один, последний стиль к таблице стилей print.css:

```
a:after {
  content: " (" attr(href) ") ";
}
```

В строке content: этот стиль добавляет URL-адрес (часть attr(href)) в конец каждой ссылки (часть a: after).

2. Сохраните файл `print.css`. Откройте страницу `cosmo.html` в браузере и распечатайте ее.

   Распечатанная страница должна выглядеть так, как показано на рис. 14.5, — простой, содержащей только факты.



About Us Page 1 of 1

The Design List

About The Design List

Ut labore et dolore magna aliqua. Sed do eiusmod tempor incididunt eu fugiat nulla pariatur. Velit esse cillum dolore ullamco laboris nisi ut enim ad minim veniam.

Ut aliquip ex ea commodo consequat. Cupidatat non proident, sunt in culpa sed do eiusmod tempor incididunt. Excepteur sint occaecat mollit anim id est laborum. Ut labore et dolore magna aliqua. Quis nostrud exercitation velit esse cillum dolore ut enim ad minim veniam.

Related Sites

- Print Magazine (http://www.printmag.com/)
- Communication Arts (http://www.commarts.com/)
- AIGA (http://www.aiga.org/)
- Sawyer McFarland Media (http://www.sawmac.com/)

Not Just Another Discussion

Ut labore et dolore magna aliqua. Sed do eiusmod tempor incididunt eu fugiat nulla pariatur. Velit esse cillum dolore ullamco laboris nisi ut enim ad minim veniam.

Ut aliquip ex ea commodo consequat. Cupidatat non proident, sunt in culpa sed do eiusmod tempor incididunt. Excepteur sint occaecat mollit anim id est laborum. Ut labore et dolore magna aliqua. Quis nostrud exercitation velit esse cillum dolore ut enim ad minim veniam.

Could it Actually Be From Another Planet?

Ut labore et dolore magna aliqua. Sed do eiusmod tempor incididunt eu fugiat nulla pariatur. Velit esse cillum dolore ullamco laboris nisi ut enim ad minim veniam.

Copyright 2010, The Design List
All rights, including photographic, polygraphic, and mineral are reserved. Sed do eiusmod tempor incididunt eu fugiat nulla pariatur. Duis aute irure dolor mollit anim id est laborum. Sed do eiusmod tempor incididunt eu fugiat nulla pariatur. Duis aute irure dolor mollit anim id est laborum. Sed do eiusmod tempor incididunt eu fugiat nulla pariatur. Duis aute irure dolor mollit anim id est laborum. Sed do eiusmod tempor incididunt eu fugiat nulla pariatur. Duis aute irure dolor mollit anim id est laborum.

file:///Z|/02_writing/00_css_mm_2e/02_tutorials/14/print.html 5/22/2009

Рис. 14.5. Простота страницы делает ее замечательной для печати. Вы получите лестные отзывы от пользователей, которым нужна только информация, а не навигационные панели, реклама и фоновые изображения, поглощающие впустую чернила

Конечную версию страницы вы найдете в папке 14_finished.

15 Совершенствуем навыки в CSS

На данный момент мы рассмотрели большинство принципов каскадных таблиц стилей. С добавлением разметки, основанной на CSS, вы можете быстро и эффективно разрабатывать сайты. Но даже теперь, когда вы овладели всеми свойствами, которые предлагает CSS, научились устранять ошибки браузеров и освоили искусные приемы разработки красивых веб-страниц, вы все еще можете изучить несколько методик, которые сделают ваш CSS-код более легким для создания, использования и поддержки.

Эта глава содержит некоторые рекомендации по созданию и использованию CSS. Они не являются основополагающими, но могут ускорить вашу работу, приводя к меньшим расстройствам и большей производительности.

Добавление комментариев

Если приходится редактировать таблицу стилей через недели, месяцы или даже годы после того, как она была создана, вы можете задаться вопросом: «Зачем я создавал этот стиль? Что он делает?» Как в любых других проектах, создавая сайт, вы должны хранить заметки о том, что вы делали и для чего. К счастью, для этого вам не придется исписывать кучу бумаги. Вы можете включать ваши заметки прямо в таблицы стилей, используя комментарии CSS.

Комментарий CSS — это просто заметка, содержащаяся между двумя наборами символов: /* и */. Как и в HTML, комментарии CSS не читаются и не выполняются браузером, но позволяют добавлять полезные напоминания к вашим таблицам стилей. Скажем, вы создали стиль, который намеревается исправить ошибку в Internet Explorer:

```
* html .imageFloat {
  display: inline;
}
```

В то время, когда вы писали стиль, вы знали, что он делает, но будете ли вы так же хорошо помнить это три месяца спустя? Добавьте комментарий, и вы или кто-то

еще, кто будет работать с вашим сайтом, легко выясните, что делает этот стиль и зачем он был создан:

```
/* Исправление ошибки двойного поля в браузере Internet Explorer 6 */
* html .imageFloat {
  display: inline;
}
```

Если вы хотите оставить более обширный комментарий, то можете занимать несколько строк. Просто начните с символов /*, введите все комментарии, а затем закончите символами */. Это удобно при добавлении сопровождающей информации в начале таблицы стилей, как показано на рис. 15.1. Вы можете также использовать многострочные комментарии для представления полезной вводной информации, которая позволит отслеживать версию сайта или таблицы стилей, добавлять информацию об авторском праве и идентифицировать вас как владельца CSS-кода.

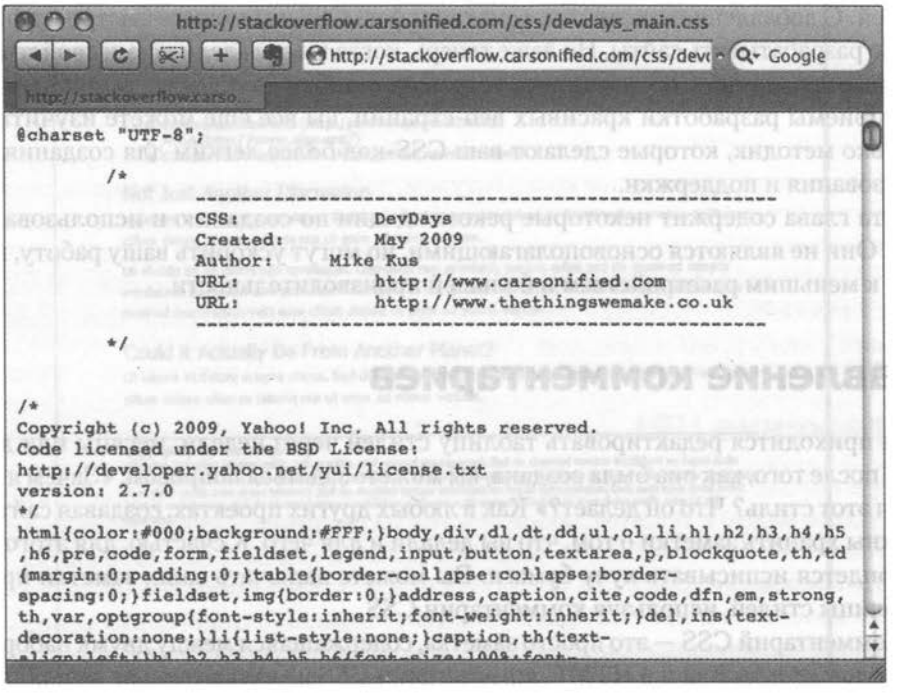


Рис. 15.1. Комментарии CSS помогут идентифицировать стили для их редактирования спустя некоторое время

Организация стилей и таблиц стилей

О создании стилей и таблиц стилей уже было рассказано многое. Но когда вы проектируете сайт, который будет поддерживаться, можете объединить несколько принципов, которые помогут вам в будущем. Придет день, когда вы должны будете

изменить вид сайта, наладить определенный стиль или передать свою тяжелую работу кому-то еще, кто теперь станет ответственным за нее. Помимо комментариев для себя или других людей, небольшое планирование и организация внутри вашего CSS-кода сгладят различные проблемы в будущем.

Давайте стилям понятные имена

Вы уже изучили технические аспекты именования различных типов селекторов — селекторы класса начинаются с выражения `a.` для идентификации стилей как классов, а селекторы ID начинаются с символа #. Кроме того, названия, которые вы задаете для ID и классов, должны начинаться с буквы и не могут содержать такие символы, как &, * или !. Помимо выполнения этих требований, необходимо соблюдать некоторые практические правила, что может помочь вам отслеживать стили и работать более эффективно.

- **Называйте стили по их назначению, а не по внешнему виду.** Очень заманчиво бывает использовать такие названия, как `.redhighlight`, при создании стиля для форматирования огненно-красного текста, бросающегося в глаза. Но что, если вы (босс, клиент) решите, что оранжевый, синий или бледно-зеленый будет смотреться лучше? В результате получится, что стиль, названный `.redhighlight`, на самом деле определяет текст бледно-зеленого цвета, что, конечно, сбивает с толку. Лучше использовать название, которое описывает *назначение* стиля. Например, если красный цвет текста предназначен для указания ошибок, которые допустил посетитель при заполнении формы, используйте название `.error`. Когда стиль должен оповестить посетителя о некоторой важной информации, назовите его `.alert`. В любом случае изменение цвета или другое форматирование стиля не вызовет путаницы, так как стиль все равно предназначен для указания ошибок или оповещения пользователей независимо от его цвета.
- **Не используйте названия, основанные на месте расположения элемента.** По той же самой причине, по которой вы избегаете названия стилей согласно их внешнему виду, вы должны избегать названия их по расположению. Иногда такое название, как `#leftSidebar`, кажется очевидным выбором: «Я хочу, чтобы весь материал в этом блоке был размещен у левого края страницы!» Но, возможно, кто-то захочет переместить левое боковое меню вправо, вверх или даже вниз страницы. И внезапно название `#leftSidebar` перестанет иметь какой-либо смысл вообще. Названия, более соответствующие назначению этого бокового меню, — `#news`, `#events`, `#secondaryContent`, `#mainNav` — идентифицируют его независимо от места расположения. Названия, которые вы видели до сих пор в этой книге, — `#gallery`, `.figure`, `.banner`, `#wrapper` и т. д. — подчиняются этому правилу.

  Всегда есть соблазн использовать имена типа `#header` и `#footer` (для элементов, которые всегда находятся в верхней или нижней части страницы), поскольку их названия более чем очевидны. Но вы во многих случаях сможете подобрать имена, которые лучше определяют содержимое элементов, например `#branding` вместо `#header`. С другой стороны, использование имен с информацией о позиции иногда имеет смысл. Скажем, вы хотите создать два стиля: один для передвижения изображения в левую часть страницы, а другой — для передвижения

изображения в правую часть. Поскольку эти стили существуют исключительно для размещения изображений в левой или правой части страницы, использование этой информации в имени стиля вполне оправданно. Так, `.floatLeft` и `.floatRight` — разумные имена.

- **Избегайте непонятных названий.** Такие названия, как `#s`, `#s1` и `#s2`, могут сделать ваши файлы меньше, но при этом доставят много хлопот при обновлении сайта. В итоге вам придется поломать голову, чтобы вспомнить, для чего же все-таки были созданы эти загадочные стили. Будьте краткими, но ясными: `#sidebar`, `#copyright` и `#banner` не потребуют много времени для набора, но их назначение сразу очевидно.

ПРИМЕЧАНИЕ

Еще больше советов насчет названий стилей вы найдете по адресу www.stuffandnonsense.co.uk/archives/whats_in_a_name_pt2.html. Вы также можете узнать многое, поискав соглашения об именовании, используемые на других сайтах. Панель инструментов веб-разработчика, описываемая во врезке «Информация для опытных пользователей» в разд. «Устранение столкновений стилей в браузере» этой главы, предоставит вам быстрый способ выявить название стиля.

Используйте несколько классов, чтобы сэкономить время

Часто два или более элемента на веб-странице определяют много подобных свойств форматирования. Вы, возможно, захотите использовать одни и те же стили границы для размещения нескольких изображений. Однако могут быть и некоторые различия в форматировании этих элементов. Возможно, вы захотите, чтобы некоторые изображения передвинулись влево и имели правое поле, в то время как другие должны передвинуться вправо и иметь левое поле (рис. 15.2).

Самое очевидное решение — создать два класса стиля, чтобы каждый имел одни и те же параметры границы, но различные свойства `float` и `margin`. Затем вы применяете один класс для изображений, которые должны передвинуться влево, а другой — для изображений, которые должны переместиться вправо. Но что, если вы должны обновить стиль границы для всех этих изображений? Вам понадобится отредактировать *два* стиля, и, если вы забудете один, у всех изображений на одной стороне страницы будет неправильное форматирование!

Рассмотрим страницы, изображенные на рис. 15.2. Страница сверху выглядит так, как она первоначально отформатирована. Снизу к обеим фотографиям применены одни и те же стили, соответственно создается эффект границы. Кроме того, к левому изображению применен дополнительный класс стиля, заставляющий его передвинуться влево; у правого изображения также есть второй класс, из-за которого оно передвинулось вправо.

Существует прием, который работает во всех браузерах и преимуществами которого пользуются удивительно немногие разработчики, — создание *нескольких классов*, применяемых к одному и тому же тегу. Это просто означает, что, когда вы используете атрибут `class` для тега, вы добавляете два (или больше) названия класса: `<div class = "note alert">`. В этом примере тег `<div>` получает инструкции форматирования от стилей `.note` и `.alert`.

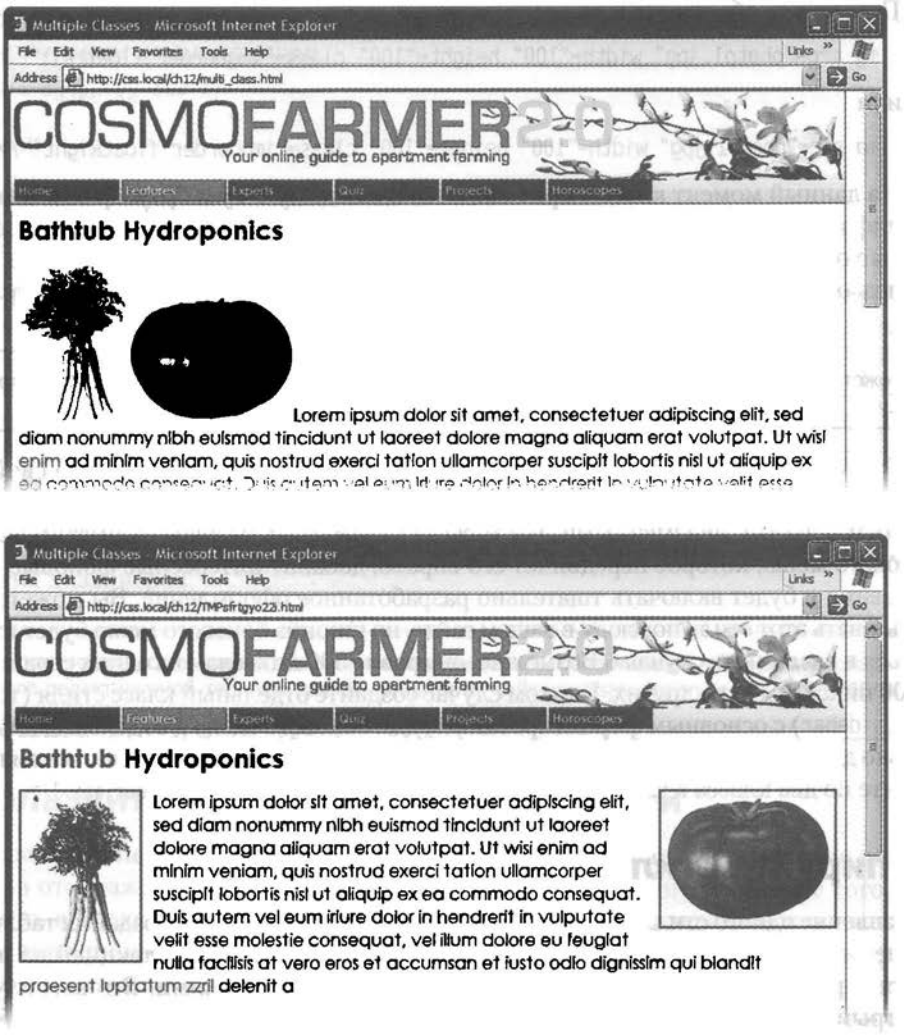


Рис. 15.2. Используя несколько классов в отдельном теге, вы можете применить два различных набора инструкций форматирования

Скажем, вы хотите использовать один и тот же стиль границы для группы изображений, но одну их часть желаете поместить слева, а другую — справа. Подойдите к этой задаче таким образом.

1. Создайте класс стиля, который включает свойства форматирования, разделяемые всеми изображениями.

   Этот стиль можно назвать `.imgFrame`, у него есть сплошная граница шириной 2 пиксела, задаваемая вокруг всех четырех сторон.

2. Создайте два дополнительных класса стиля, один — для изображений, передвигаемых влево, а другой — для передвигаемых вправо, например `.floatLeft` и `.floatRight`.

3. Примените оба класса к каждому тегу:

```
<img src="photo1.jpg" width="100" height="100" class="imgborder floatLeft" />
```

или

```
<img src="photo1.jpg" width="100" height="100" class="imgborder floatRight" />
```

На данный момент классы применяются к каждому тегу и браузер объединяет информацию о стиле для каждого класса, чтобы отформатировать тег. Теперь, если вы хотите изменить стиль границы, просто отредактируйте стиль `.imgFrame`. Это позволит обновить границы вокруг изображений, перемещенных влево и вправо.

СОВЕТ

Вы можете перечислить этим методом более двух классов; только не забывайте ставить пробел между их названиями.

Эта методика полезна, когда вы должны наладить только несколько свойств одного элемента, оставив при этом остальные элементы, отформатированные подобным образом, неизменными. Вы можете разработать базовое форматирование бокового меню, которое передвинет его вправо, добавит интересные фоновые изображения и будет включать тщательно разработанное оформление. Вы можете использовать этот стиль повсюду в вашем сайте, но ширина бокового меню будет изменяться в различных случаях. Возможно, она равна 300 пикселам на одних страницах и 200 пикселам — на других. В таком случае создайте отдельный класс стиля (такой как `.sidebar`) с основным форматированием для бокового меню и отдельные классы только для определения ширины бокового меню, например `.w300` и `.w200`. Затем примените по два класса к каждому боковому меню: `<div class = "sidebar w300">`.

Группируйте стили

Добавление одного стиля за другим — распространенный способ создания таблицы стилей. Однако через какое-то время то, что было простой коллекцией из пяти стилей, расширилось до массивного CSS-файла со 150 стилями. В таком случае быстрый поиск нужного стиля — то же самое, что поиск иголки в стоге сена. Если вы организуете стили с самого начала, то в конечном счете намного облегчите себе жизнь. Нет никаких конкретных и быстрых правил насчет того, *как* группировать стили, но есть две общепринятые методики.

- **Группируйте стили, которые применяются к связанным частям страницы.** Группируйте все правила, которые применяются к тексту, изображениям и ссылкам в баннере страницы, в одном месте; правила, которые относятся к главной навигации, — в другом; а стили для основного содержимого — в третьем.
- **Группируйте стили со связанными задачами (имеющими друг к другу отношение).** Помещайте все стили для разметки в одну группу, для оформления — в другую, для ссылок — в третью и т. д.

Использование комментариев для разделения групп стилей. Неважно, какой подход вы предпочтете, но убедитесь, что используете комментарии CSS, чтобы разделить каждую группу стилей. Скажем, вы собрали все стили, которые управ-

ляют разметкой ваших страниц, в одно место в таблице стилей. Опишите эту коллекцию таким комментарием:

```
/* *** Разметка *** */
```

или

```
/* --------------------------
Разметка
-------------------------- */
```

Укажите в начале /*, а в конце — */, тогда внутри вы можете использовать любую вычурную комбинацию из звездочек, черточек или других символов, которые вам нравятся, чтобы сделать эти комментарии легко узнаваемыми. Вы найдете столько их разновидностей, сколько существует веб-дизайнеров. Если вы ищете интересную идею, посмотрите, как таблицы стилей комментируются на следующих сайтах: www.wired.com, www.mezzoblue.com и http://keikibulls.com. Кроме того, используйте панель инструментов веб-разработчика, рассмотренную во врезке «Информация для опытных пользователей» в разд. «Устранение столкновений стилей в браузере» этой главы. Она поможет вам быстро вникать в таблицы стилей других дизайнеров.

СОВЕТ

Для ознакомления с методикой обозначения комментариев, облегчающей нахождение конкретного раздела редактируемой таблицы стилей, зайдите на сайт www.stopdesign.com/log/2005/05/03/css-tip-flags.html.

Используйте несколько таблиц стилей

Как вы читали в гл. 14, можно создавать различные таблицы стилей для разных типов отображения — одну для экрана, а другую для принтера. Кроме того, возможно, вы захотите иметь множество таблиц стилей для экрана исключительно в организационных целях. Здесь используется базовая концепция из предыдущего раздела — группирование связанных стилей.

Если таблица стилей становится настолько большой, что трудно находить и редактировать стили, то, возможно, пришло время создать отдельные таблицы стилей, каждая из которых имеет свои функции. Вы можете поместить стили для форматирования форм в одной таблице стилей, стили для разметки — в другой, стили, которые определяют цвета элементов, — в третьей, создать еще одну таблицу стилей для хранения ваших трюков в Internet Explorer и т. д. Конечно, количество отдельных файлов должно быть в пределах разумного, так как, скажем, 30 внешних CSS-файлов, через которые придется пройти, совсем не сэкономят время.

На первый взгляд может показаться, что получится больше кода в каждой веб-странице, так как будет намного больше внешних таблиц стилей, которые нужно присоединить или импортировать, — по одной строке кода для каждого файла. Но есть подход лучше: создайте отдельную внешнюю таблицу стилей, которая использует правило `@import`, чтобы включить множество таблиц стилей. Рисунок 15.3 иллюстрирует этот подход.

Рассмотрим, как реализовать этот подход.

1. **Создайте внешние таблицы стилей для форматирования различных типов элементов вашего сайта.** Например, создайте файл `color.css` со стилями для управления цветом сайта, файл `forms.css`, который управляет форматированием форм, файл `layout.css` для задания разметки и файл `main.css`, который охватывает все остальное (см. рис. 15.3, *справа*).

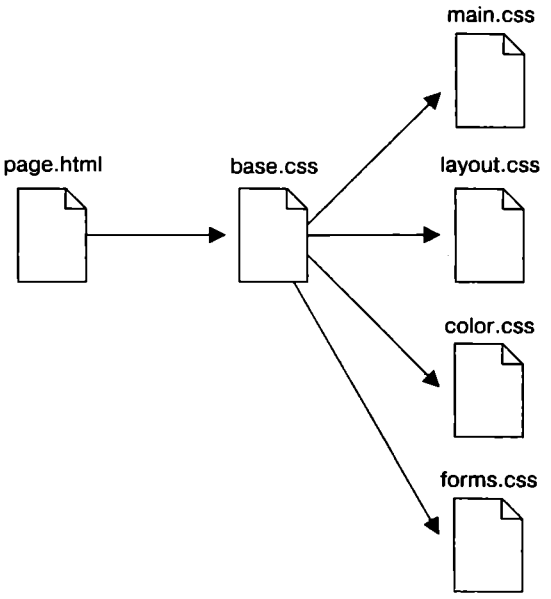


Рис. 15.3. HTML-страница может присоединить один CSS-файл (base.css в этом примере). HTML-код не должен изменяться, даже если вы хотите добавить или удалить дополнительные внешние таблицы стилей. Просто обновите файл base.css, добавляя или удаляя правило @import

ПРИМЕЧАНИЕ

Это всего лишь рекомендации. Организуйте ваши стили и таблицы стилей так, как вам кажется наиболее логичным и будет работать наилучшим образом для вас. Для дополнительной информации можете прочитать статью о модульном CSS-дизайне на сайте www.contentwithstyle.co.uk/content/modular-css.

2. **Создайте «пограничную» внешнюю таблицу стилей и импортируйте каждую таблицу стилей, которую вы создали в шаге 1.** Вы можете назвать файл `base.css`, `global.css`, `site.css` или как-то вроде этого. Данный CSS-файл не будет содержать каких-либо предписаний. Вместо этого используйте правило `@import`, чтобы присоединить другие таблицы стилей:

```
@import url(main.css);
@import url(layout.css);
@import url(color.css);
@import url(forms.css);
```

Это весь код, который должен быть в файле, хотя вы можете еще добавить некоторые комментарии с номером версии, названием сайта и т. д., чтобы помочь идентифицировать файл.

СОВЕТ

Лучший способ присоединения таблицы стилей только для Internet Explorer описывается в разд. «Управление браузером Internet Explorer» этой главы.

3. **Наконец, присоедините таблицу стилей из шага 2 к HTML-страницам вашего сайта, используя тег <link> или правило @import** (чтобы вспомнить, как использовать эти методы, откройте разд. «Внешние таблицы стилей» гл. 2). Например:

```
<link rel="stylesheet" href="base.css" type="text/css" />
```

Теперь, когда веб-страница загружается, браузер использует файл `base.css`, который, в свою очередь, говорит браузеру загрузить четыре остальные таблицы стилей.

Вам может показаться, что здесь происходит очень много загрузок. Однако браузер лишь однажды загружает эти файлы и сохраняет их в своем кэше — ему не приходится снова получать их по Интернету (см. врезку «Обходной прием» в обучающем уроке гл. 2).

Есть и другая польза от применения отдельной внешней таблицы стилей для загрузки нескольких других таблиц: если вы позже решите разделить стили на дополнительные таблицы стилей, то вам не придется работать с HTML-кодом вашего сайта. Вместо этого просто добавьте еще одно правило `@import` к той «пограничной» таблице стилей (см. шаг 2 выше). Если вы решите вытащить все стили, связанные с типами устройств, из файла `main.css` и поместить их в собственный файл `type.css`, вам не нужно будет затрагивать веб-страницы сайта. Просто откройте таблицу стилей со всеми правилами `@import` в ней и добавьте еще одно: `@import url(type.css)`.

Эта возможность также позволяет вам обмениваться данными между различными таблицами стилей для временных изменений дизайна. Скажем, вы решаете изменять цвет вашего сайта каждый день, месяц или сезон. Если вы уже поместили основные, определяющие цвет стили в отдельный файл `color.css`, то можете создать другой файл (например, `summer_fun.css`) с различным набором цветов. Затем измените в «пограничном» файле правило `@import` для файла `color.css`, чтобы загружать новый файл цветовых стилей (например, `@import url(summer_fun.css)`).

ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

Головная боль с кэшем

Кэш браузера, как правило, является помощником каждого владельца сайта. Как мы уже обсуждали в этой книге, кэш гарантирует, что частым посетителям вашего сайта не придется загружать один и тот же файл снова и снова, что замедлило бы открытие страниц и повысило плату за хостинг. Тем не менее кэш может стать головной болью, когда вам необходимо обновить внешний вид сайта. Например, если все его

ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

страницы ссылаются на внешнюю таблицу стиля под названием `main.css`, у посетителей этот файл будет кэширован. Но при обновлении файла и, соответственно, изменении внешнего вида сайта предыдущие посетители могут по-прежнему иметь доступ к старой таблице стилей, сохраненной на их жестком диске, вместо нового файла `main.css`.

Со временем кэш посетителей все же очистится и они получат новые CSS-файлы. Однако у вас есть один простой способ победить кэш — обновить тег `<link>` на каждой HTML-странице. Обычно тег `<link>` для внешней таблицы стилей выглядит следующим образом:

```
<link rel="stylesheet" type="text/css"
href="main.css">
```

Однако если добавить строку запроса после имени CSS-файла (например, `main.css?v=1`), то браузер будет видеть файл как `main.css?v=1`, а не как `main.css`. Если вы измените число после `v=` при обновлении внешней таблицы стилей, браузеры воспримут это как появление нового файла и загрузят внешнюю таблицу стилей с веб-сервера вместо использования кэшированной версии.

Предположим, что, когда вы запускаете ваш сайт, файл `main.css` является первой версией стилей CSS сайта. Вы можете использовать этот тег `<link>`:

```
<link rel="stylesheet" type="text/css"
href="main.css?v=1">
```

Затем, когда вы обновите файл `main.css`, можно изменить тег `<link>` на следующий:

```
<link rel="stylesheet" type="text/css"
href="main.css?v=2">
```

Браузер посчитает таблицу стилей отличной от сохраненной в кэше версии файла `main.css` и загрузит файл с веб-сервера. На самом деле `?v=1` ничего не делает и не влияет на то, как работает веб-сервер. Это просто способ сказать браузеру перезагрузить файл.

Недостатком этого метода является необходимость обновлять теги `<link>` для каждого HTML-файла сайта. Если вы также работаете на PHP, то вам доступен более автоматизированный способ справиться с этой проблемой: http://ikeif.net/2009/03/27/stop-cachingfiles-PHP-функции.

Устранение столкновений стилей в браузере

Когда вы просматриваете в браузере веб-страницу, основанную не на CSS, у HTML-тегов уже есть некоторое минимальное форматирование: заголовки полужирные, тег `<h1>` больше остального текста, ссылки подчеркнуты и синего цвета и т. д. В отдельных случаях разные браузеры применяют различное форматирование для каждого из этих элементов. Вы можете решить, что это скучно, так как выглядит *почти* одинаково и в Internet Explorer, и в Firefox, и в Safari.

Firefox обычно использует таблицу стилей CSS для форматирования большинства HTML-тегов. Чтобы увидеть это форматирование в Mac, определите местонахождение файла приложения Firefox, щелкните на нем правой кнопкой мыши, а затем выберите пункт **Show package contents** (Показать содержимое). Далее перейдите в папку **Contents ▸ MacOS ▸ res** и откройте файл `html.css` в текстовом редакторе. В Windows этот файл находится по адресу `C:\Program Files\Mozilla Firefox\res\html.css`.

ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

Панель инструментов веб-разработчика

Веб-дизайнерам нужно следить за огромным количеством вещей: HTML, CSS, ссылки, графика, формы и т. д. Поиск и устранение неисправностей в любом из этих пунктов может иногда стать реальным вызовом. Панель инструментов веб-разработчика (http://chrispederick.com/work/webdeveloper/), созданная Крисом Педериком (Chris Pederick), является расширением браузера Firefox (рис. 15.4). Если у вас нет Firefox, то установите его только из-за одной этой панели инструментов (www.mozilla.com/firefox).

Загрузите расширение, установите его и потратьте немного времени, выбирая различные настройки. Доступно много полезных возможностей, но здесь мы рассмотрим те, которые стоит отметить.

- Выберите CSS ▸ View CSS (CSS ▸ Показать CSS), и вы увидите все стили текущей страницы, даже те, что импортированы из множества таблиц стилей. Если вы когда-либо были на сайте и задавались вопросом: «Как же они делают это?» — этот инструмент предоставит вам бесплатную возможность попасть «за кулисы».
- Выбрав CSS ▸ Edit CSS (CSS ▸ Редактировать CSS), вы сможете редактировать стили текущей веб-страницы. Это не причинит вреда реальной веб-странице, но действительно позволит вам налаживать стили и моментально видеть результаты.
- Меню Information (Информация) предоставляет множество уточняющих сведений. Выбрав пункт Display Block Size (Показать размеры блока), вы увидите размеры блочных элементов, таких как таблицы и разделы `div`. Пункт Display Element Information (Показать информацию об элементе) представляет информацию о любом элементе страницы, на котором вы щелкнете кнопкой мыши (включая HTML-атрибуты, свойства CSS, положение элемента на странице). Выбор пункта Display Id & Class Details (Показать детали ID и классов) — отличный способ увидеть названия стилей, примененных к тегам страницы. Используйте его, чтобы видеть, как другие разработчики называют свои стили.
- Меню Tools (Сервис) предоставляет доступ к онлайн-инструментам проверки кодов HTML и CSS, а также проверки ссылок. Эти инструменты работают только с онлайн-страницами.

Microsoft предлагает подобный инструмент для Internet Explorer. Вы можете найти его на сайте www.microsoft.com, введя `IE Developer Toolbar` в поле поиска.

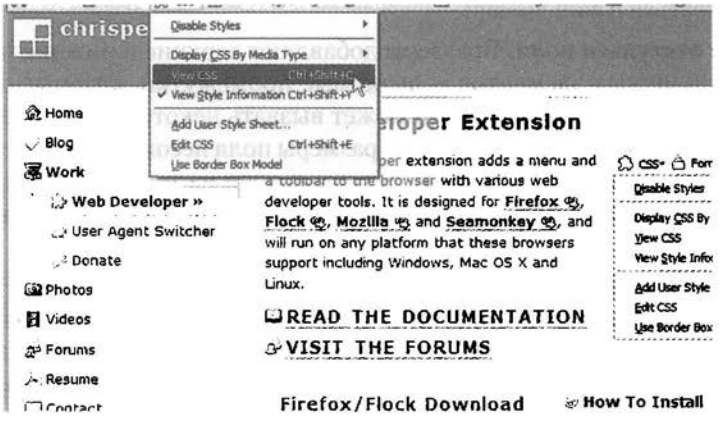


Рис. 15.4. Панель инструментов веб-разработчика позволяет просматривать стили любого сайта в Интернете, идентифицировать HTML-структуру страницы, получить информацию о том, как разработан стиль элемента, проверить страницу и ее CSS-код и многое другое

Из-за этих различий браузеров приходится обнулять исходное форматирование для тегов, чтобы ваши посетители могли увидеть красивый дизайн, над созданием которого вы столь усердно работали (рис. 15.5). Все, что нужно сделать, — настроить в начале своей таблицы стилей некоторые основные стили, которые убирают неприятное форматирование.

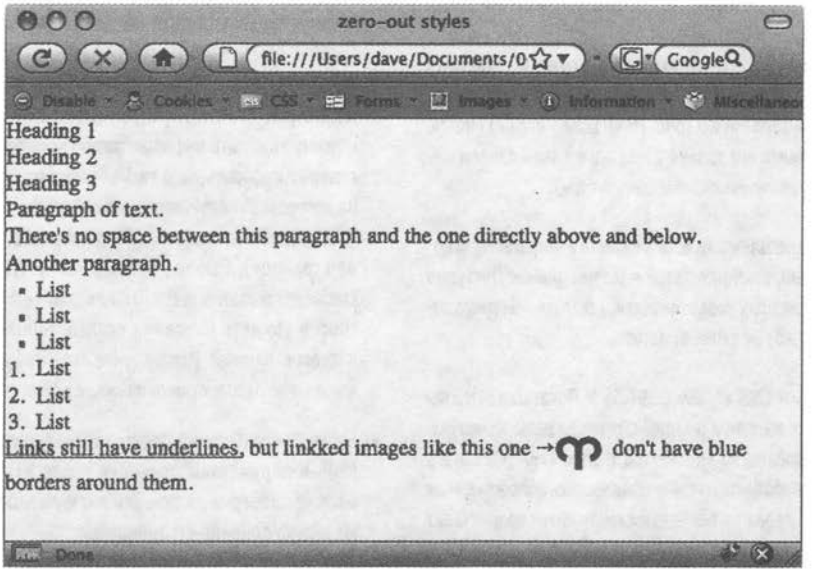


Рис. 15.5. Устраните различия в отображении страниц, обнулив исходные стили браузера. Затем создайте свои собственные стили, чтобы добавить поля, отступы и размеры шрифта, которые являются совместимыми с разными браузерами

Вот некоторые действия, которые можно выполнить, чтобы браузеры прекратили вмешиваться в ваш дизайн.

- **Уберите отступы и поля.** Браузеры добавляют верхние и нижние поля к большинству блочных элементов — знакомые промежутки, которые появляются между тегами <p>, например. Это может вызвать некоторые проблемы отображения, к примеру, когда конкретные размеры поля несовместимы с некоторыми браузерами. Лучше всего убрать отступы и поля из тегов блочных элементов, которые вы используете, а затем специально добавить нужные при создании новых стилей.
- **Применяйте совместимые размеры шрифта**. В то время как текст тега <p> показывается размером 1 em, браузеры применяют различные размеры для других тегов. Вы можете сделать так, чтобы все теги были размером 1 em, а затем создать дополнительные стили со специфическими размерами шрифта для разных тегов. Таким образом, у вас будет намного больше шансов получить размеры шрифта, совместимые с различными браузерами.
- **Улучшайте границы таблиц и создавайте согласующиеся ячейки.** Применяя границы к ячейкам таблицы, вы создаете промежуток между ячейками, а также

удваиваете границы между ними. Вы должны избавиться как от лишнего пространства, так и от дополнительных границ. Кроме того, тегам <th> и <td> задаются разные типы выравнивания и плотность шрифта.

- **Уберите границы в изображениях ссылок.** Internet Explorer, Firefox и другие браузеры добавляют цветные границы вокруг любого изображения внутри ссылки. Наверняка вам, как и многим пользователям, эти границы покажутся ненужными и непривлекательными. Удалите их и задайте заново там, где считаете необходимым.
- **Задавайте согласованные отступы для списков и типы маркировки.** Различные браузеры делают отступы для маркированных и нумерованных списков по-разному, и, как вы заметите, даже используемый тип маркировки может варьироваться. Желательно устанавливать согласованные отступы и типы маркировки.
- **Убирайте кавычки из цитируемого материала.** Если вы когда-либо использовали тег <q> для выделения цитаты (<q>To err is human</q> (<q>Человеку свойственно ошибаться</q>), например), то заметите, что некоторые браузеры (Firefox, Safari) автоматически добавляют кавычки (' ') вокруг цитаты, а некоторые (Internet Explorer 6 и 7) — нет. И даже среди браузеров, которые добавляют кавычки, тип этих кавычек может варьироваться. Например, IE 8 вставляет одинарные кавычки (' '), в то время как Firefox добавляет двойные кавычки (" "). Для согласованного представления содержимого лучше всего удалить кавычки. Для реализации описанных идей есть несколько базовых стилей, которые вы можете добавить в начало вашей таблицы стилей:

```
html, body, h1, h2, h3, h4, h5, h6, p, ol, ul, li, pre, code, address,
variable, form, fieldset, blockquote {
  padding: 0;
  margin: 0;
  font-size: 100%;
  font-weight: normal;
}
table {
  border-collapse: collapse;
  border-spacing: 0;
}
td, th, caption {
  font-weight: normal;
  text-align: left;
}
img, fieldset {
  border: 0;
}
ol {
  padding-left: 1.4em;
  list-style: decimal;
}
ul {
  padding-left: 1.4em;
```

```
  list-style:square;
}
q:before, q:after {
  content:'';
}
```

Первые два стиля здесь — групповые селекторы, которые применяют одно и то же форматирование к каждому из перечисленных тегов. Добавьте эти стили в начало вашей таблицы стилей, а затем внизу таблицы замените их в зависимости от конкретного случая. После обнуления полей и размера шрифта (`font-size`) в теге `<h1>` вы можете задать для него определенное значение верхнего поля и размер шрифта. Просто добавьте другой стиль, например такой:

```
h1 {
margin-top: 5px;
font-size: 2.5em;
}
```

Этот стиль появится в таблице стилей после групповых селекторов, удаляющих поля и изменяющих размер шрифта, поэтому благодаря каскаду (см. гл. 5) новые значения будут иметь приоритет.

Файл `reset.css` вы найдете в папке 15 обучающих материалов. Просто скопируйте код из этого файла в свои таблицы стилей.

ПРИМЕЧАНИЕ

Светиле веб-дизайна Тантеку Селичу (Tantek Celic) часто приписывают создание очень полезной методики ликвидации стандартного HTML-форматирования браузера. Вы можете найти специальный набор стилей на сайте http://tantek.com/log/2004/undohtml.css.

Использование селекторов потомков

Классы и ID (идентификаторы) отлично подходят для обозначения конкретных тегов, которые необходимо стилизовать. Например, вы можете добавить класс к абзацу `<p class="intro">` и определить, что только один абзац будет иметь внешний вид, определенный в стилевом классе `.intro`. Поскольку так просто добавлять класс или идентификатор к тегу, многие дизайнеры приняли за правило добавлять их ко *всему* (или почти ко всему), и это становится проблемой. У профессионалов есть даже диагноз этому заболеванию — *классицизм*. Добавление класса к каждому тегу — это не только трата времени, но и замедление загрузки HTML. К тому же есть лучший способ получить контроль над тегами, не прибегая к слишком большому количеству классов или идентификаторов, — использование селекторов потомков (наследуемых селекторов).

Селекторы потомков — мощный инструмент для эффективного создания сайтов. Как мы обсуждали в гл. 3, они позволяют точно указать теги, для которых вы хотите задать стили. В большинстве случаев нужно отформатировать все ссылки навигационной панели одинаково, но это не значит, что нужно отформатировать все ссылки на странице таким же образом. Нужно как бы сказать браузерам: «Форматируйте так *только* ссылки в навигационной панели», не применяя класс стиля

к каждой из этих ссылок. Другими словами, мы хотим иметь возможность форматировать один и тот же HTML-элемент по-разному, в зависимости от того, где он располагается, а это как раз то, что предлагают селекторы потомков (рис. 15.6).

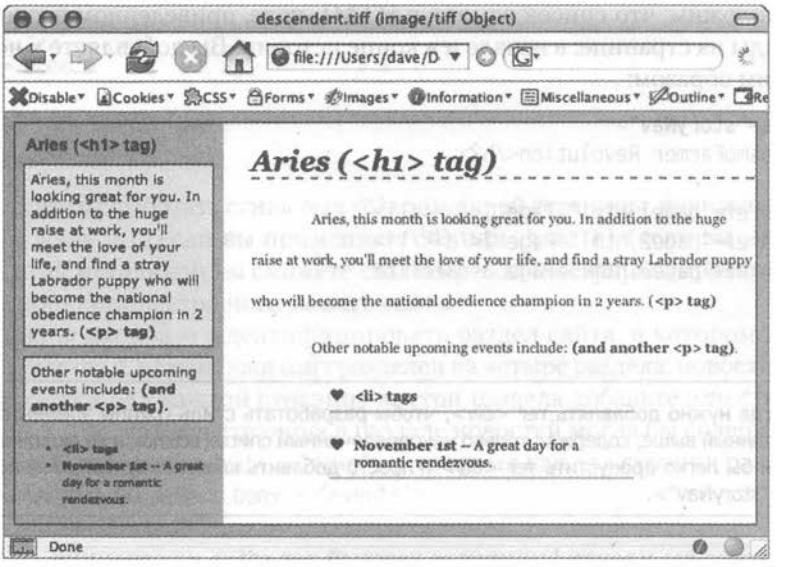


Рис. 15.6. Один и тот же HTML-код был вставлен в левое боковое меню и в область справа. При использовании селекторов потомков идентичные HTML-теги (<h1>, <p>, <ul> и <li>) форматируются по-разному, в зависимости от их расположения на странице

Разделяйте свои страницы

Одним из самых больших помощников в эффективном использовании селекторов потомков является тег . Поскольку этот тег позволяет создавать на странице логические разделы , вы можете применять его для идентификации различных элементов разметки, таких как баннер, боковое меню, колонка текста и т. д. Как мы обсуждали во врезке «Бриллиант без огранки» в разд. «Селекторы классов» гл. 3, можно организовать содержимое страницы в блоки, задавая для каждого HTML-тег .

Сгруппируйте заголовок текста и список ссылок для навигации по страницам:

```
<div>
<h2>The CosmoFarmer Revolution</h2>
<ul>
  <li><a href="page1.html">Page 1</a></li>
  <li><a href="page2.html">Page 2</a></li>
  <li><a href="page3.html">Page 3</a></li>
</ul>
</div>
```

После добавления идентифицируйте его для CSS либо с атрибутом class ( ), либо с атрибутом id ( ). Если вы

хотите вставить один и тот же тип элемента разметки несколько раз на странице (например, несколько врезок в отдельной истории), используйте класс. Для областей, которые появляются только однажды на странице, например для баннера, лучше выбрать ID-селектор.

Предположим, что список ссылок в HTML-коде, приведенном выше, появляется дважды на странице: в начале и в конце истории. Вы добавляете к нему класс следующим образом:.

```
<div class="storyNav">
<h2>The CosmoFarmer Revolution</h2>
<ul>
  <li><a href="page1.html">Page 1</a></li>
  <li><a href="page2.html">Page 2</a></li>
  <li><a href="page3.html">Page 3</a></li>
</ul>
</div>
```

СОВЕТ

Вам не всегда нужно добавлять тег , чтобы разработать стиль группы элементов. Если бы код, приведенный выше, содержал только неупорядоченный список ссылок и не включал тег <h2>, то вы могли бы легко пропустить тег и просто добавить класс к неупорядоченному списку: <ul class = "storyNav">.

Как только вы идентифицируете каждый тег на странице, будет очень легко использовать селектор потомков, нацеленный на элементы внутри конкретного тега . Скажем, вы хотите создать уникальный вид для каждой ссылки в приведенном коде. Вы создаете селектор потомков следующим образом:

```
.storyNav a {
  color: red;
  background-color: #ccc;
}
```

Теперь ссылки будут выделены красным цветом на светло-сером фоне, но только когда они находятся внутри другого тега, к которому применяется класс storyNav. Если вы хотите добавить другую ссылку (например, на страницу page4.html) к этому списку, вам не нужно обращаться к HTML-коду, чтобы отформатировать ее, как другие ссылки. Браузер все обрабатывает автоматически, когда применяет селектор потомков.

Форматирование других элементов внутри этого тега — просто вопрос создания селектора потомков, начинающегося с имени класса (.storyNav), за которым следует пробел и имя элемента, стиль которого вы хотите разработать. Чтобы отформатировать тег <h2>, который появляется внутри , создайте селектор потомков .storyNav h2.

Идентифицируйте содержимое тега <body>

Поскольку селекторы потомков обеспечивают такое специфическое определение стилей, вы можете легко создать стили, которые не только относятся к одной кон-

кретной области страницы, но и применяются исключительно к определенным типам страниц на вашем сайте. Скажем, вы хотите разработать тег <h1> для домашней страницы не так, как для остальных страниц сайта. Легкий способ разграничить теги <h1> для домашней страницы — добавить атрибуты class или id к тегу <body> домашней страницы:

```
<body id="home">
```

или

```
<body class="home">
```

Вы можете разработать стиль тега <h1> домашней страницы, используя селектор потомков: #home h1 (если вы применяете id) или .home h1 (если вы используете class). С этой методикой вы сможете создать различное форматирование для любого тега конкретной страницы вашего сайта.

Один из подходов — идентифицировать раздел сайта, в котором находится каждая страница. Скажем, ваш сайт разделен на четыре раздела: новости, события, статьи и ссылки. На каждой странице внутри раздела добавьте или class, или id к тегу <body>. Так, каждая страница в разделе новостей могла бы содержать следующий HTML-код: <class body = "news">, в то время как для страниц раздела событий было бы задано <class body = "events">.

СОВЕТ

Вы также можете использовать класс, чтобы идентифицировать тип разметки, который хотите установить для определенной страницы (например, с одним, двумя или тремя столбцами).

Кроме всего прочего, идентификация раздела страницы применяется для выделения кнопки этого раздела на навигационной панели. Выделенные кнопки действуют наподобие индикаторов, показывающих, где находится пользователь, как продемонстрировано на рис. 15.7. Если страница расположена в разделе новостей вашего сайта, вы можете выделить кнопку **News**, так что посетитель моментально сможет определить раздел, который сейчас просматривает.

Рассмотрим, как можно форматировать навигационную кнопку в зависимости от того, в каком разделе сайта она находится.

1. **Добавьте идентификатор к тегу <body>, указывающий раздел, в котором находится страница.** Например, <body id = "home">. Сделайте то же самое для каждого раздела, чтобы у страниц в разделе новостей был следующий код: <body id = "news">.
2. **Добавьте навигационную панель на страницу.** Пошаговые инструкции по созданию навигационной панели вы найдете в разд. «Создание панелей навигации» гл. 9.
3. **Идентифицируйте каждую ссылку навигационной панели.** Для ссылки на домашнюю страницу у вас может быть такой код: <href = "/index.html" id = "homeLink">Home. Атрибут id позволяет идентифицировать ссылку как указывающую на домашнюю страницу (можно сделать то же самое, используя class вместо id). Повторите это для других ссылок: <href = "/news/" id = "newsLink">News и т. д.

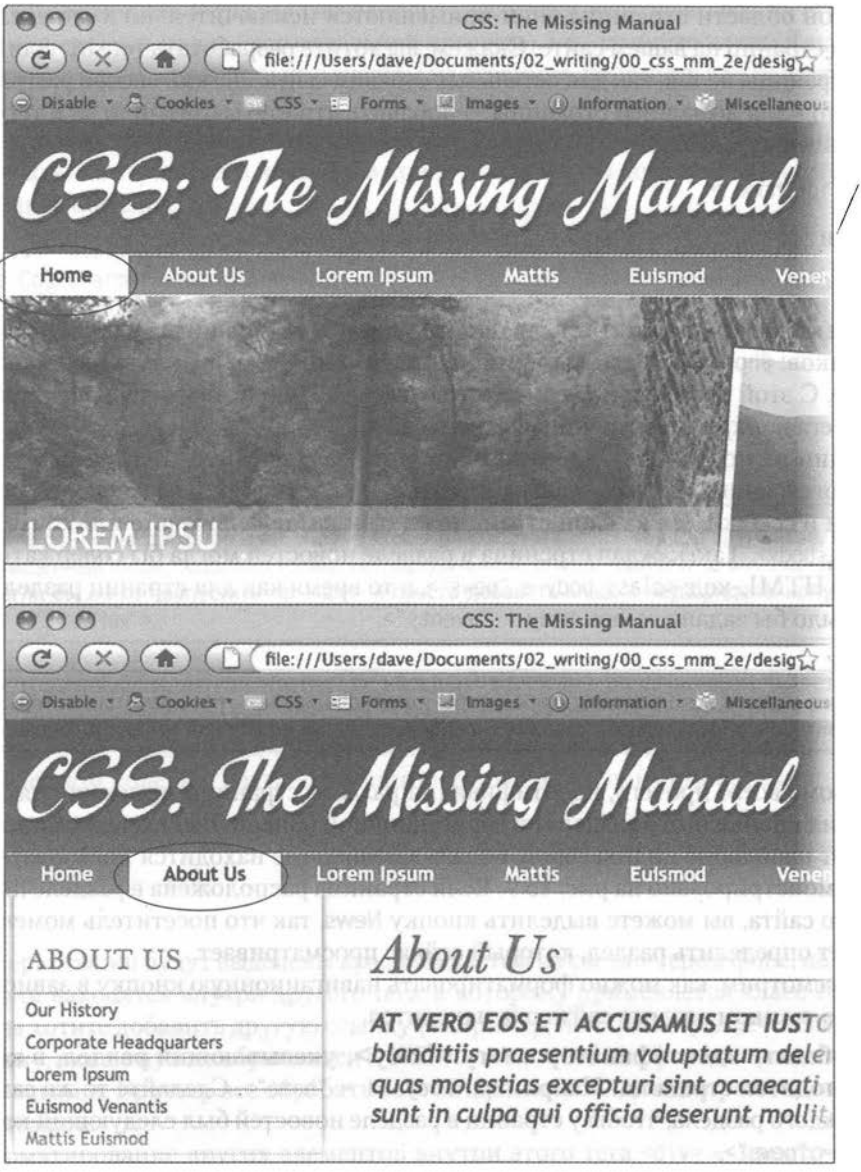


Рис. 15.7. Используя селекторы потомков, можно выделить кнопку на навигационной панели, просто изменив атрибуты class или id тега <body>. Здесь, когда для тега указан атрибут id home, выделяется кнопка Home *(вверху)*. Измените id на feature, и выделится кнопка About Us *(внизу)*

На данный момент у вас достаточно информации в HTML-коде, чтобы уникально отформатировать ссылку каждого раздела, используя CSS. В этом примере известно, что ссылка на домашнюю страницу вложена в тег <body> с id, равным home, только на домашней странице.

4. **Создайте селектор потомков, который позволит отформатировать ссылку каждого раздела по-своему.** Это форматирование применяется, когда ссылка находится внутри страницы для этого раздела. Для домашней страницы в этом примере селектор потомков должен быть примерно таким:

```
#home a#homeLink
```

Селектор форматирует ссылку #homeLink, только когда она находится внутри другого тега с id, равным #home. В большинстве случаев нужно, чтобы выделенная кнопка выглядела одинаково для каждого раздела сайта, так что лучше использовать групповой селектор (см. разд. «Стилизация групп тегов» гл. 3), чтобы собрать вместе все селекторы потомков для каждой кнопки раздела. Таким образом, вы применяете одно и то же форматирование к каждой кнопке, не создавая для нее отдельные правила. Групповой селектор для выделения кнопки текущего раздела светло-желтым фоном может быть таким:

```
#home a#homeLink,
#news a#newLink,
#articles a#articlesLink,
#links a#linksLink {
  background-color: #FBEF99;
}
```

СОВЕТ

При создании группового селектора, который включает в себя несколько селекторов потомков, указывайте каждый селектор на отдельной строке, как в этом примере. Так будет легче распознать селекторы в группе, когда придется вновь редактировать таблицу стилей.

Используя ту же методику, создайте дополнительные стили, определяющие различные внешние виды для всех состояний ссылок: при наведении указателя мыши, щелчке кнопкой мыши и пр. (см. «Выборка стилизуемых ссылок» гл. 9).

Приведенные примеры — лишь некоторые способы использования селекторов потомков. Эти селекторы могут сделать ваши таблицы стилей немного более сложными. У вас будут такие стили, как #home, .navbar, например, вместо простого класса .navLink. Настроив стили, в дальнейшем вы будете выполнять лишь небольшое форматирование. HTML-код, который вставляется в различные области страницы, автоматически форматируется совершенно по-разному. Это практически волшебство!

Управление браузером Internet Explorer

Браузеры не всегда ведут себя так, как вы этого ожидаете. Такие браузеры, как Safari, Firefox и Internet Explorer 8, достаточно хорошо обрабатывают CSS-код и последовательно и предсказуемо отображают основанные на CSS веб-страницы. Однако сделать так, чтобы ваш дизайн работал в Internet Explorer 5, 6 и 7, куда сложнее. Хоть эти браузеры и стары по современным стандартам, они все еще достаточно популярны.

В этой книге часто рассказывалось о серьезных ошибках Internet Explorer 5 и 6 и о способах их устранения. Существует проблема двойного поля (см. подраздел «Ошибка двойного поля» разд. «Обработка ошибок в Internet Explorer 6» гл. 12), ошибка блочной модели в Internet Explorer 5 (см. подраздел «Вычисление фактических размеров блочных элементов» разд. «Определение параметров высоты и ширины» гл. 7). Методики для управления этими ошибками включают прием `* html` (см. врезку «Ошибки браузеров» в разд. «Определение параметров высоты и ширины» гл. 7). Но просто знания этих методик недостаточно. Вам нужно считаться со всей веб-аудиторией и быть уверенными, что налаживание работы в Internet Explorer не вызывает проблем для других посетителей.

СОВЕТ

Список страниц с описанием различных ошибок CSS в разных браузерах вы можете найти на сайте http://cssdiscuss.incutio.com/?page=BrowserBugs.

Дизайн для современных браузеров

Поскольку Internet Explorer 6 и 7 очень распространены, многие веб-разработчики используют один из этих браузеров для проверки дизайна своих сайтов. Находя проблему отображения веб-страницы, они исправляют свой CSS-код, пока страница не станет выглядеть так, как надо. К сожалению, поскольку Internet Explorer 6 и 7 не всегда правильно понимают форматирование, которое дизайнеры используют для этих браузеров, исправление веб-страниц приводит к тому, что более современные браузеры, такие как IE 8, Firefox и Safari, также отображают их некорректно.

Устаревший подход проектирования для Internet Explorer 6 и 7 был бы прекрасен, если бы все посещали ваш сайт именно из этих браузеров. Но, поскольку все больше людей переходят на его восьмую версию или другие современные браузеры, например Firefox или Safari, то ваши точно настроенные для Internet Explorer 6 страницы начнут выходить из строя. Лучший подход — проектировать сайты, держа в уме Internet Explorer 8, Firefox, Chrome и Safari. Убедитесь, что ваш CSS-код работает в этих браузерах, и вы будете уверены, что правильно используете CSS. После того как ваш сайт станет замечательно выглядеть в этих браузерах, настанет время устранить проблемы, которые неожиданно возникают в Internet Explorer 6 и 7.

Это может показаться непреодолимым препятствием, но будьте мужественны. Чаще всего вы будете сталкиваться с одними и теми же ошибками, которые, в свою очередь, требуют одних и тех же способов устранения. Так что, как только вы станете опытными в идентификации и устранении ошибок двойного поля и др., вам будет нетрудно добавить необходимые приемы, чтобы наладить страницы для старых версий Internet Explorer.

СОВЕТ

Для получения более подробной информации о том, как Internet Explorer может исказить ваши тщательно разработанные веб-страницы, зайдите на сайты www.positioniseverything.net/explorer.html и www.positioniseverything.net/ie-primer.html.

Изолируйте CSS для Internet Explorer условными комментариями

Прием * html, описанный в гл. 7, является одним из способов устранения ошибки Internet Explorer версий 6 и ниже без неблагоприятного воздействия на остальные браузеры. Но, поскольку по мере добавления новых стилей ваши таблицы становятся все больше, появляется беспорядок. Даже если вы изолируете эти изменения в одной части вашей таблицы стилей, все еще можно оставить некоторый недействительный CSS-код (как zoom: 1), который будет препятствовать тому, чтобы ваш главный CSS-файл был признан корректным.

Другой способ собрать стили только для Internet Explorer в отдельном месте — использовать возможности условных комментариев (рис. 15.8). Это изобретение Microsoft предоставляет способ вставки HTML-кода только для Internet Explorer. Другие браузеры принимают этот код за комментарий и игнорируют его.

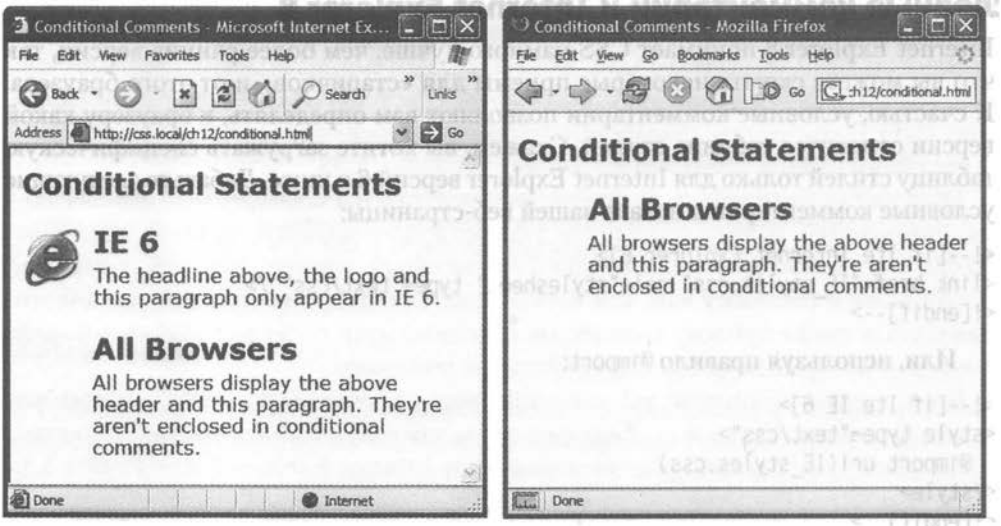


Рис. 15.8. Условные комментарии позволяют сделать так, чтобы отдельный HTML-код появлялся только в определенной версии Internet Explorer *(слева)*. Другие браузеры просто проигнорируют код внутри комментариев *(справа)*

Условные комментарии могут даже предназначаться для различных версий браузера. Вы можете поместить все стили только для Internet Explorer 6 в отдельной внешней таблице стилей (такой как IE6_styles.css) и использовать условные комментарии, чтобы связывать ее только с этим браузером. Применяя этот подход, вы легко сможете удалить данные стили, когда Internet Explorer 6 наконец повторит судьбу динозавров. Нужно будет просто удалить внешнюю таблицу стилей.

Ваши посетители, работающие в других браузерах, также находятся в выгодном положении. Когда вы используете условные комментарии, другие браузеры и вовсе не загружают эти внешние таблицы стилей. В результате сайт открывается и выполняется быстрее.

Вот основная структура условных комментариев:

```
<!--[if Internet Explorer]>
Some HTML code that only applies to IE goes here.
<![endif]-->
```

`<!--[if Internet Explorer]>` — непосредственно само условие. Оно переводится как «если этот браузер — Internet Explorer, то обрабатываем следующий HTML-код». Таким образом, с кодом, который идет после этой строки и заканчивается выражением `<![endif]->`, будет работать лишь Internet Explorer. Так вы можете добавлять текст, изображения, стили и даже ссылки к внешним таблицам стилей — все только для Internet Explorer.

ПРИМЕЧАНИЕ

Другие браузеры просто рассматривают условные комментарии как обычные HTML-комментарии и игнорируют их.

Условные комментарии и Internet Explorer 8

Internet Explorer 8 понимает CSS намного лучше, чем более ранние версии, так что вы можете скрыть некоторые приемы для «старичков» и от этого браузера. К счастью, условные комментарии позволяют вам определять, к браузеру какой версии относится таблица стилей. Скажем, вы хотите загружать специфическую таблицу стилей только для Internet Explorer версий 6 и ниже. Добавьте следующие условные комментарии в начале вашей веб-страницы:

```
<!--[if lte Internet Explorer 6]>
<link href="IE_styles.css" rel="stylesheet" type="text/css" />
<![endif]-->
```

Или, используя правило `@import`:

```
<!--[if lte IE 6]>
<style type="text/css">
  @import url(IE_styles.css)
</style>
<![endif]-->
```

Слово `lte` обозначает «меньше или равно», так что `if lte IE 6` обозначает «если этот браузер — шестая или более ранняя версия Internet Explorer».

Условные комментарии и каскадность

Используйте любой удобный метод для присоединения внешней таблицы стилей (см. разд. «Внешние таблицы стилей» гл. 2), но добавляйте условные комментарии после любых других присоединенных таблиц стилей. Большинство приемов Internet Explorer стремятся переопределить стили, уже представленные в таблице стилей, — стили, которые работают для других браузеров. Из-за каскадности правила, определенные на странице позже, могут отменять ранее заданные стили. Чтобы убедиться в том, что ваши переопределенные, специфические для Internet Explorer стили успешно приняты этим браузером, добавляйте их после любой другой таблицы стилей, присоединенной к странице.

Рассмотрим код, который вы могли бы использовать, чтобы связать:

- таблицу стилей для всех браузеров;
- таблицу стилей только для Internet Explorer 7;
- таблицу стилей для Internet Explorer версий 6 и ниже.

```
<link href="global_styles.css" rel="stylesheet" type="text/css" />
<!--[if IE 7]>
<link href="IE6_styles.css" rel="stylesheet" type="text/css" />
<![endif]-->
<!--[if lte IE 6]>
<link href="IE5_styles.css" rel="stylesheet" type="text/css" />
<![endif]-->
```

ПРИМЕЧАНИЕ

Для получения более полной информации об условных комментариях Internet Explorer зайдите на сайт http://msdn.microsoft.com/en-us/library/ms537512(VS.85).aspx.

16 CSS 3 — на гребне волны

Несмотря на то что CSS 2.1 уже достаточно много лет, только сейчас, когда вышел Internet Explorer 8, можно сказать, что все наиболее популярные браузеры правильно обрабатывают эту версию CSS. Конечно же, Всемирная паутина развивается, а это значит, что W3C-консорциум вовсю работает над новым стандартом CSS 3 (рис. 16.1). Большинство разработчиков браузеров уже используют отдельные части стандарта из этой практически готовой редакции CSS.

Если вам нравится быть на гребне волны и хочется поскорее исследовать последние новинки, то эта глава для вас. Однако будьте готовы принять во внимание тот факт, что некоторые свойства CSS 3 еще не обрабатываются всеми браузерами. В этой главе я расскажу вам, как использовать общепринятые свойства нового стандарта, а также приведу возможные обходные пути по их использованию для браузеров, которые пока еще эти свойства не понимают.

Прежде всего запомните: веб-страницы вовсе *не должны* выглядеть одинаково во всех браузерах. Главное — убедиться в том, что сайты *работают* у всех посетителей, даже в том случае, когда те используют Internet Explorer 6 или Firefox 3.5. Если страница не читается ввиду различий, связанных со способом ее отображения браузерами (например, появляется проблема выпадения столбца, о которой говорилось в подразделе «Предотвращение перепадов плавающих элементов» разд. «Преодоление проблем перемещения» гл. 12), то это именно та проблема, которую вам необходимо решить.

Иначе говоря, весьма похвально улучшать внешний вид страниц, используя свойства, которые поддерживают лишь некоторые браузеры. Например, свойство `text-shadow` позволяет вашему тексту отбрасывать тень (более подробно об этом рассказывается в разд. «Тень для текста» текущей главы). Это свойство работает в браузерах Safari, Opera 9.5, Firefox 3.5 и Chrome, но недоступно ни для одной из версий Internet Explorer. Едва заметная тень может улучшить внешний вид заголовков в перечисленных браузерах, но этого не заметят пользователи IE. И все же чувствуйте себя свободно, изучая эти свойства CSS, даже несмотря на то, что не каждый сможет ими насладиться.

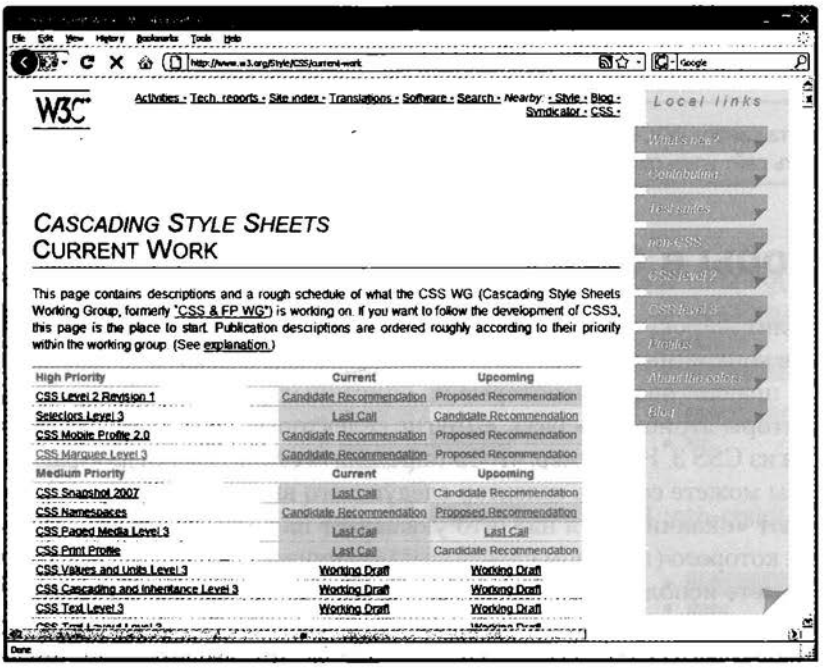


Рис. 16.1. W3C-консорциум представляет самую последнюю информацию о CSS на сайте www.w3.org/Style/CSS/current-work. Здесь вы сможете прочесть обо всех изменениях в CSS 3 и узнать о том, как проходит процесс принятия модулей

Обзор CSS 3

С 1996 года CSS значительно «повзрослел» и стал основным инструментом вебдизайнеров. Благодаря их же постоянным просьбам дать возможность делать еще более красивые сайты последующие версии CSS приобретали больше свойств и, как следствие, бóльшую сложность. Версию CSS 3 W3C разделил на независимые *модули*, каждый из которых описывает отдельные наборы свойств. Назначение некоторых модулей известно, например CSS Transitions (анимация или трансформация переходов между двумя состояниями, скажем, подсветка текста при наведении на него указателя мыши) или CSS Web Fonts (инструкции для изменения различных свойств шрифта). Функции других пока остаются загадкой, например CSS Math или CSS Namesace.

Основная цель подобного модульного деления состоит в том, чтобы дать возможность CSS развиваться постепенно, шаг за шагом, не заставляя всех ждать, когда же наконец будет готов один гигантский набор инструкций. Каждый модуль развивается сам по себе, а производители браузеров могут использовать отдельные инструкции модулей сразу, как только те будут готовы.

Хотя ни один из модулей CSS 3 еще окончательно не готов, некоторые наиболее привлекательные свойства уже нашли свое применение в браузерах (даже в том

самом Internet Explorer). Остаток главы как раз и посвящен этому набору новых возможностей.

СОВЕТ

Помимо сайта самого W3C-консорциума, для получения самой свежей информации о CSS 3 можно использовать сайт www.css3.info.

Селекторы в CSS 3

CSS 3 вводит широкий спектр новых селекторов, которые позволяют вам определять стили конкретных элементов страницы с гораздо большей точностью, нежели ранее. По правде говоря, некоторые из описанных ранее атрибутов в подразделе «Селекторы атрибутов» разд. «Другие селекторы» гл. 3, представляют собой селекторы из CSS 3. Например, чтобы определить стиль ссылки для файлов Adobe Acrobat, вы можете создать селектор следующего вида: `a[href$=".pdf"]`. Символы `$=` означают «оканчивается на», что указывает на элемент страницы, значение аттрибута которого (в данном случае `href`) оканчивается на `.pdf`. Подобным образом вы можете использовать символы `^=` для того, чтобы точно указать атрибут, значение которого начинается с заданных вами символов (в том же подразделе гл. 3 есть описание того, как использовать селекторы данного вида для определения ссылок на другие веб-страницы).

Эти дополнительные селекторы атрибутов, рассмотренные в данном разделе, работают в браузерах Firefox, Safari, Chrome, Opera и даже Internet Explorer 7 и выше (но не в IE 6, как вы уже догадались). Другие же новые селекторы работают в большинстве браузеров, кроме IE.

ПРИМЕЧАНИЕ

Если вы хотите знать, какие из селекторов CSS 3 понимает ваш любимый браузер, посетите страницу набора по тестированию CSS-селекторов (CSS Selectors Test Suite) по адресу www.css3.info/selectors-test. Эта замечательная JavaScript-программа протестирует ваш браузер и даст одну из двух оценок (зеленый цвет — тест пройден, красный — нет) по каждому селектору CSS 3. Браузеры Safari 4, Firefox 3.5, Opera 9.5 и Chrome проходят все 43 теста, в то время как Internet Explorer 8 — лишь 22.

Селекторы дочерних узлов

CSS предоставляет несколько методов для форматирования тегов, которые являются дочерними по отношению к остальным. Как говорилось в разд. «Стилизация вложенных тегов» гл. 3, дочерний элемент — это любой тег, который находится внутри другого тега. Возьмем, к примеру, слово, выделенное полужирным, внутри абзаца: тег `<strong>` является дочерним по отношению к тегу `<p>`. А в маркированном списке элементов, например, теги `<li>` являются дочерними для элемента `<ul>`.

- `:first-child` — вы уже видели этот селектор раньше в разд. «Псевдоклассы и псевдоэлементы» гл. 3. Являясь на самом деле частью стандарта CSS 2, этот

селектор позволяет оформить только самый первый дочерний элемент тега (рис. 16.2). Например, чтобы оформить первый элемент маркированного списка, вы можете написать вот так:

```
ul :first-child
```

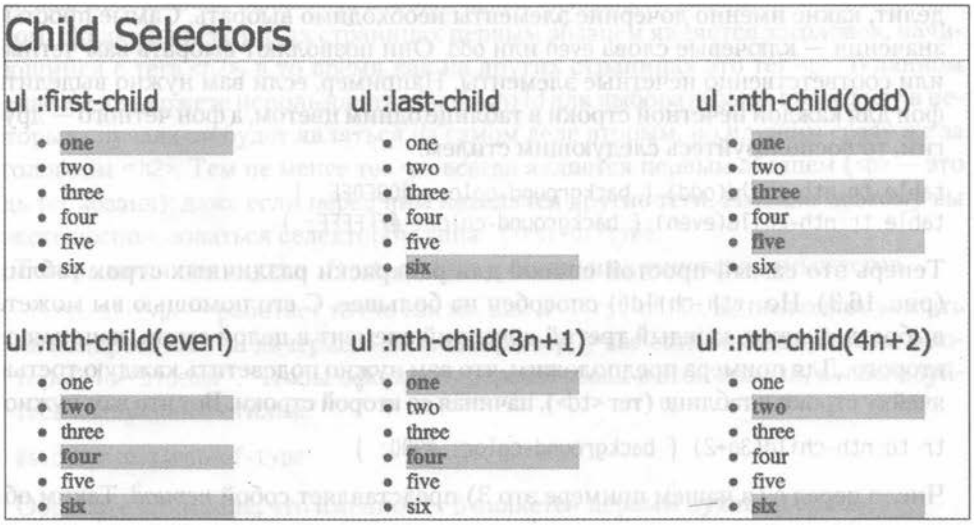


Рис. 16.2. Широкий перечень дочерних селекторов в CSS дает возможность выбирать дочерние элементы. Эти селекторы великолепно подходят, когда нужно выбрать первый, последний или чередующиеся пункты списка

Обратите внимание на пробел, расположенный между `ul` и `:first-child`. Эта конструкция означает «найти первый дочерний элемент тега ». Конструкция вида `ul:first-child` без пробела имеет совершенно другой смысл: «найти все теги <ul>, которые являются дочерними для какого-то другого тега».

Для того чтобы выделить цветом любой текст, который встречается первым в теге `div` класса `announcement`, воспользуйтесь следующим стилем:

```
.announcement :first-child { color: #F33F00; }
```

Стиль будет применен только к первому элементу внутри тега , будь то <h1>, <h2>, <p> или любой другой тег. Это дает вам дополнительную гибкость, поскольку стиль первого дочернего элемента, расположенного внутри , не зависит от его конкретного типа.

Селектор `:first-child` работает во всех современных браузерах за исключением IE 6 и его более ранних версий.

- `:last-child` — этот селектор является новым в CSS 3 и указывает на последний дочерний элемент какого-либо тега. Его можно использовать, например, чтобы подчеркнуть линией границу последнего абзаца в теге или же чтобы выделить последний элемент в списке (см. рис. 16.2, *посередине вверху*). Этот селектор не работает во всех браузерах Internet Explorer и даже в восьмой версии.

- `:nth-child()` — сложный, но весьма полезный селектор. Благодаря ему вы можете запросто выделить каждую четную строку в таблице или каждый третий элемент в списке либо применить отдельный стиль для любой другой альтернативной комбинации дочерних элементов (см. рис. 16.2).

  Этот селектор требует указания значения параметра в скобках, который определит, какие именно дочерние элементы необходимо выбрать. Самые простые значения — ключевые слова `even` или `odd`. Они позволяют выбрать вам четные или соответственно нечетные элементы. Например, если вам нужно выделить фон для каждой нечетной строки в таблице одним цветом, а фон четного — другим, то воспользуйтесь следующим стилем:

```
table tr:nth-child(odd) { background-color: #09F0FF; }
table tr:nth-child(even) { background-color: #FFFFFF; }
```

  Теперь это самый простой способ для раскраски различных строк таблиц (рис. 16.3). Но `nth-child()` способен на большее. С его помощью вы можете выбрать, скажем, каждый третий дочерний элемент в целой серии, начиная со второго. Для примера предположим, что вам нужно подсветить каждую третью ячейку строки в таблице (тег `<td>`), начиная со второй строки. Вот что вам нужно:

```
tr td:nth-child(3n+2) { background-color: #900; }
```

  Число перед `n` (в нашем примере это 3) представляет собой *период*. Таким образом, `3n` означает каждый третий элемент, а `4n` — каждый четвертый. Знак `+` вместе со вторым числом (в примере выше это `+2`) говорит, с какого элемента следует начать: `+2` означает начать со второго дочернего элемента, а `+5` — с пятого. Следовательно, `nth-child(5n+4)` выберет каждый пятый элемент, начиная с четвертого дочернего.

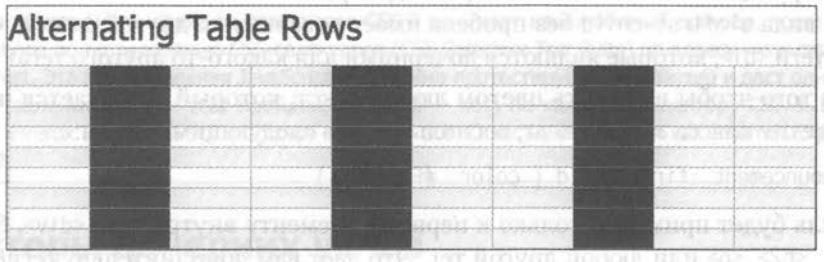


Рис. 16.3. Простейший путь для раскрашивания таблиц — использование новых селекторов. Можно даже раскрашивать чередующиеся столбцы, указывая каждую отдельную ячейку или, как в данном примере, каждый третий столбец, начиная со второго

К несчастью, только Firefox 3.5 и его более поздние версии понимают этот новый чудесный селектор. Другие браузеры — Safari, Opera, Chrome и Internet Explorer — просто проигнорируют его. Вместо него для раскраски таблиц вы можете использовать менее технологичное, работающее во всех браузерах решение, описание которого доступно в подразделе «Применение стилей к строкам и столбцам» разд. «Создание стилей для таблиц» гл. 10.

Селекторы типов

В CSS 3 введены новые селекторы, которые работают практически так же, как и селекторы дочерних элементов, рассмотренные в предыдущем разделе, за тем лишь исключением, что позволяют указать на конкретный *тип* выбираемого тега. Допустим, вам необходимо оформить первый абзац внутри боковой панели особым образом, однако на некоторых страницах первым абзацем является заголовок, начинающийся с тега <h2>, в то время как на других страницах это тег <p>. В данном случае вы не сможете использовать `:first-child` для выбора абзаца, поскольку в некоторых случаях он будет являться на самом деле вторым, но идущим сразу же за заголовком <h2>. Тем не менее тег <p> всегда является первым абзацем (<p> — это ведь тег абзаца), даже если перед ним находятся другие теги. Именно поэтому вы можете воспользоваться селектором типа `:first-of-type`.

Теперь заглянем внутрь `:first-of-type` и остальных типозависимых тегов.

- `:first-of-type` — работает точно так же, как и `:first-child`, но позволяет указать на конкретный тип дочернего тега. Например, у вас есть боковая панель, в которой `id="sidebar"`. Чтобы оформить первый абзац в этой панели, воспользуйтесь следующим стилем:

  ```
  #sidebar p:first-of-type
  ```

  Обратите внимание, что именно тег `p` является первым нужным типом.

- `:last-of-type` — работает точно так же, как и `:last-child`, но дает возможность указать на последний тег конкретного типа. Например, если вы хотите оформить последний абзац боковой панели, но не уверены в том, что он является последним и за ним не следуют другие теги (список, заголовок или изображение), то воспользуйтесь следующим стилем:

  ```
  #sidebar p:last-of-type
  ```

ПРИМЕЧАНИЕ

Помните, что выбранный этими селекторами тег также является дочерним. Конструкция p:first-of-type означает «первый дочерний элемент, содержащий тег абзаца».

- `:nth-of-type` — работает точно так же, как и `:nth-child`, но позволяет указать на элементы конкретного типа. Этот тег может быть удобен в том случае, когда имеется большой объем текста, содержащий фотографии. Тег <img> может находиться в любом месте внутри текста, это значит, что ваш абзац может содержать часть изображений. Скажем, вам нужно расположить некоторые изображения слева, а другие — справа от текста, как показано на рис. 16.4. Вы можете использовать следующий стиль:

  ```
  img:nth-of-type(odd) { float: left; }
  img:nth-of-type(even) { float: right; }
  ```

  Вы можете заметить, что здесь используются те же ключевые слова (`odd` и `even`) и формулы (например, `2n+1`), что и для селектора `:nth-child()`.

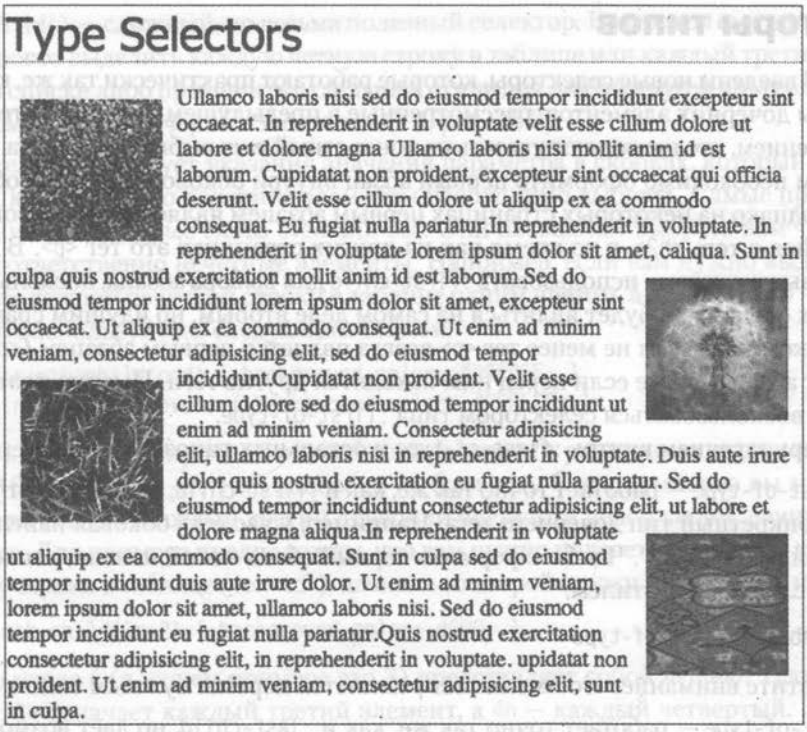


Рис. 16.4. Используя селектор :nth-type(), вы можете запросто выделять изображения через одно, чтобы расположить их в шахматном порядке

К сожалению, селекторы типов не работают в Internet Explorer. Можете применять их только для браузеров Firefox 3.5, Safari, Opera и Chrome.

ПРИМЕЧАНИЕ

Полный список селекторов в CSS 3 находится по адресу www.w3.org/TR/css3-selectors.

ЧАСТО ЗАДАВАЕМЫЕ ВОПРОСЫ

Расширения от различных разработчиков

Мне встречались некоторые CSS-стили, в которых используются свойства вроде -moz-opacity или -webkit-border-radius. Что это означает?

Такие разработчики браузеров, как Apple, Mozilla, Opera и Microsoft, иногда добавляют свои свойства, называя их таким образом, чтобы указать на то, что работать они будут лишь в их браузерах. Разработчики используют такой способ, когда хотят добавить те свойства, которые не являются частью стандарта CSS, или если W3C-консорциум еще не решил, как свойство должно работать на самом деле. Свойства, начинающиеся на `-moz-`, работают в браузере Firefox, `-webkit-` — в Safari, `-ms-` — в Internet Explorer, а `-o-` доступны в браузере Opera. Обычно, когда W3C принимает свойства и определяется с их деталями, производители прекращают их поддерживать. Например, раньше Firefox использовал свойство `-moz-opacity`, но теперь, начиная с версии 3, этот браузер уже понимает свойство `opacity` третьей версии стандарта CSS.

Прозрачность

Если вам нравятся призраки, сквозь которых все видно, то opacity — свойство CSS 3 — вам придется по душе. Это простое свойство позволяет сделать любой элемент страницы частично прозрачным так, что вы сможете использовать в качестве фона страницы изображения, сделав их немного поблекшими, размытыми. В связке с JavaScript свойство opacity можно использовать для создания эффекта динамического появления элементов на странице, регулируя уровень их прозрачности. Кроме этого, можно воспользоваться псевдоклассом :hover у ссылок и оживить панель навигации или даже страницу просмотра фотографий так, как это показано на рис. 16.5.

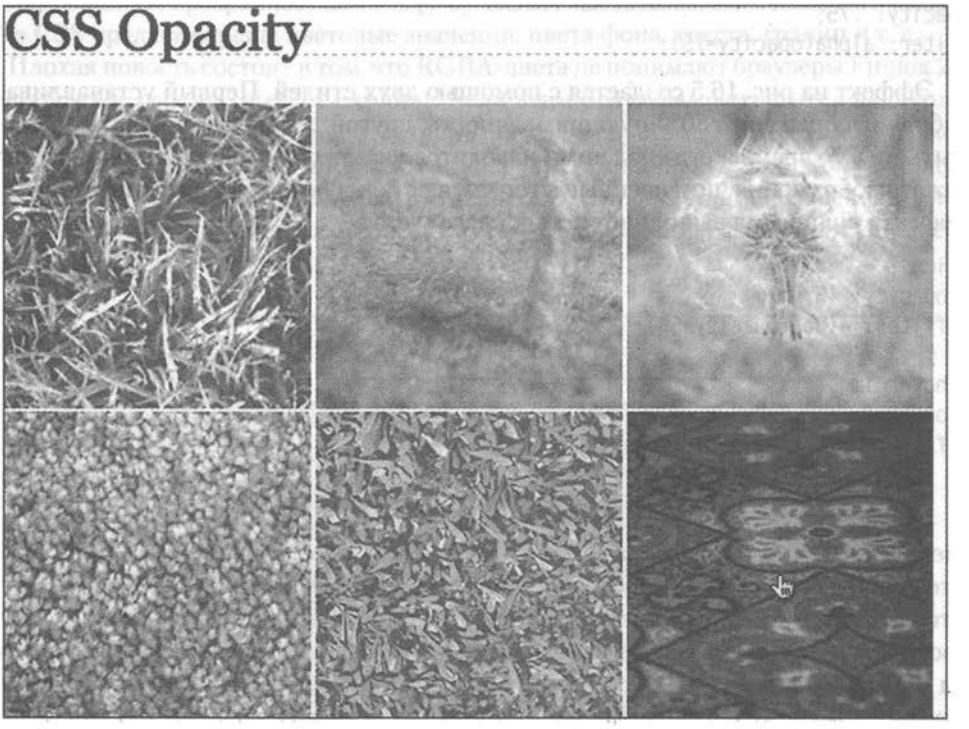


Рис. 16.5. Здесь изображения имеют прозрачность 50 % (практически сливаются с белым фоном страницы). Однако наведение указателя мыши на одно из них *(внизу справа)* делает его абсолютно четким и ясным

Свойство opacity может принимать значение в диапазоне от 0 (полностью прозрачный) до 1 (непрозрачный). Так, если хотите сделать некоторые изображения класса see-thru прозрачными наполовину, воспользуйтесь следующим стилем:

```
.see-thru { opacity: .5; }
```

Это свойство работает во всех браузерах, кроме Internet Explorer. И все же IE поддерживает схожее свойство, которое позволяет добиться такого же результата.

Для того чтобы предыдущий код стал работоспособным во всех браузерах, включая IE 6, перепишите его таким образом:

```
.see-thru {
  opacity: .5;
  filter: alpha(opacity=50);
}
```

Свойство `filter` доступно только в IE и позволяет добавлять всевозможные визуальные эффекты для ваших страниц. В нашем случае фильтр `alpha` дает возможность установить прозрачность элемента от 0 (полностью прозрачный) до 100 (непрозрачный). Так, чтобы установить размытие элемента на 75 %, используются следующие две строки:

```
opacity: .75;
filter: alpha(opacity=75);
```

Эффект на рис. 16.5 создается с помощью двух стилей. Первый устанавливает набору изображений 50%-ную прозрачность, другой делает их полностью неразмытыми в то время, когда над ними находится указатель мыши. В основе эффекта лежат изображения, помещенные в тег `<a>`, а остальную часть работы по созданию эффекта появления выполняет псевдокласс `:hover`:

```
a img {
  opacity: .5;
  filter: alpha(opacity=50);
}
a:hover img {
  opacity: 1;
  filter: alpha(opacity=100);
}
```

Правда, существует одна проблема со свойством `opacity`: все дочерние теги наследуют его. Для примера рассмотрим текст, расположенный внутри тега `<div>`. Этот блок находится над изображением, которое играет роль фона и расположено в теге `<body>`. Вы же собираетесь сделать этот блок цветным, но добавив прозрачности так, чтобы изображение фона частично просвечивалось. К сожалению, если вы примените свойство `opacity` к тегу `<div>`, то это свойство унаследует и текст внутри блока даже в том случае, если будет находиться в других тегах, будь то `<h1>` или `<p>`. Другими словами, текст станет прозрачным и нечитаемым. К счастью, в CSS 3 существует другое решение этой проблемы — цвета в формате RGBA.

RGBA-цвет

Шестнадцатеричное представление цвета, например `#FF0066`, вам уже знакомо. Кроме этого, из гл. 6 вы узнали о RGB-записи: например `rgb(25, 255, 0)`. В CSS 3 введен новый способ определения цвета RGBA, что можно расшифровать как красный (Red), зеленый (Green), голубой (Blue) и альфа (Alpha). Это точно такой же RGB-цвет, только с добавлением *альфа-прозрачности*. Альфа-прозрачность работает

точно так же, как и *размытие* (свойство `opacity`), описанное в предыдущем разделе. Возможные значения от 0 до 1 указывают, насколько плотным должен быть цвет: 0 значит, что цвет полностью невидим, а 1 определяет «целый» цвет, то есть абсолютно непрозрачный.

Предположим, вы хотите добавить фоновый цвет блоку `<div>` класса `caption`, но с условием, чтобы этот цвет был прозрачным, а содержимое, находящееся под блоком, оставалось читаемым (рис. 16.6). Просто создайте следующий стиль:

```
.caption {
  background-color: rgba(95,156,140,.75);
}
```

В этом промере цветом фона является цвет морской волны (`95, 156, 140`), но с установленной прозрачностью 75 % (`.75`). RGBA-цвета можно использовать везде, где в CSS предусмотрены цветовые значения: цвета фона, текста, границ и т. д.

Плохая новость состоит в том, что RGBA-цвета не понимают браузеры Firefox 2, Opera 9 и все версии Internet Explorer. Если этим браузерам встретится наш код, то они просто его проигнорируют, оставив цвет фона пустым. У вас есть несколько способов для решения этой проблемы: во-первых, не волноваться по этому поводу и позволить браузерам игнорировать этот цвет. Это простейший, но, очевидно, не самый лучший способ. Например, текст на рис. 16.6 станет неразборчив, если цвет фона будет проигнорирован.

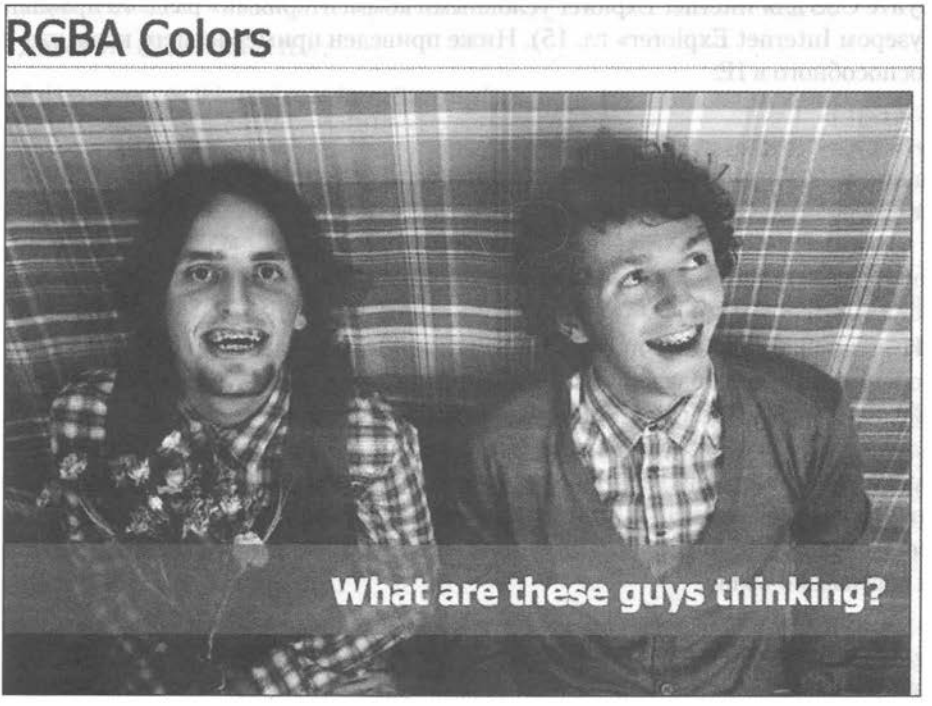


Рис. 16.6. RGBA-цвет — новая возможность в CSS 3 — позволяет запросто сделать отдельные элементы страницы частично прозрачными так, как это сделано с подписью на рисунке

СОВЕТ

Каждый редактор изображений, будь то Photoshop или Fireworks, одинаково хорошо понимает как шестнадцатеричное, так и обычное представление RGB-цвета. Если у вас уже есть шестнадцатеричное значение цвета, а вы хотите привести его в обычный вид RGB, то воспользуйтесь следующей страницей: www.javascripter.net/faq/hextorgb.htm.

Другой способ — явно указать цвет фона в RGB- или шестнадцатеричном формате. Такой подход предполагает два шага. Первый позволяет договориться со всеми браузерами, которые не понимают RGBA-цвета, кроме IE. Изменим наш стиль:

```
.caption {
  background-color: rgb(95,156,140); /* Для Opera и других браузеров */
  background-color: rgba(95,156,140,.75);
}
```

Здесь первая строка `background-color` использует RGB-цвета, которые доступны для всех браузеров. Когда браузеры Opera, Internet Explorer 8 или Firefox 2 встретят второе объявление цвета фона, они просто его проигнорируют, поскольку не понимают RGBA-цветов. Этот код будет работать в Opera 9 и IE 8, но IE 6 или IE 7 не отобразят вообще никакого цвета.

Следовательно, вам необходим дополнительный стиль для IE, где нужно воспользоваться комментариями с условием (техника рассматривается в подразделе «Изолируйте CSS для Internet Explorer условными комментариями» разд. «Управление браузером Internet Explorer» гл. 15). Ниже приведен пример нашего кода, уже работоспособного в IE:

```
<!--[if IE]>
<style type="text/css">
.caption {
  background-color: rgb(95,156,140);
}
</style>
<![endif]-->
```

Имитирование RGBA-цвета в Internet Explorer. Существует еще один способ, который позволяет браузеру IE присоединиться к RGBA-«вечеринке». Как и со свойством `opacity`, Internet Explorer допускает применение свойства `filter` так, что вы сможете добиться подобного эффекта, как при использовании RGBA-цветов. Правда, это немного накладно и вам придется добавить в свой код еще одно свойство, чтобы все заработало. Итак, чтобы увидеть результат этой работы, добавьте к своей странице приведенный ниже код, и вы получите тот самый эффект, о котором рассказано выше:

```
<!--[if IE]>
<style type="text/css">
.caption {
  background-color: transparent;
```

```
  filter:progid:DXImageTransform.Microsoft.gradient(
  startColorstr=#BF5F9C8C,endColorstr=#BF5F9C8C);
  zoom: 1;
}
</style>
<![endif]-->
```

Этот код делает то же самое, что и одна эта строка:

```
.caption { background-color: rgba(95,156,140,.75); }
```

Рассмотрим, как все работает.

- Во-первых, вам нужно поместить код для IE в условные комментарии. В примере в комментариях находится сам код стиля, но вы можете использовать их, чтобы сослаться на отдельный файл стиля для браузера IE (описание техники см. в подразделе «Изолируйте CSS для Internet Explorer условными комментариями» разд. «Управление браузером Internet Explorer» гл. 15).
- Используйте такое же имя для селектора, которое вы использовали ранее для создания нашего эффекта. В данном случае это селектор `.caption`.
- Для правильной работы в браузере IE 8 вам сначала нужно сделать цвет фона прозрачным: `background-color: transparent`. Этот стиль очищает цвет фона для элемента так, что свойство `filter`, рассматриваемое дальше, сможет работать корректно.
- Добавьте свойство `filter`:

  ```
  filter:progid:DXImageTransform.Microsoft.gradient(
  startColorstr=#BF5F9C8C,endColorstr=#BF5F9C8C);
  ```

  Много кода, не правда ли? Просто потратьте время, переписав все, как видите в книге, разве что можете поместить все это в одну строку. В примере два параметра, которые вы можете именить, — `startColorstr` и `endColorstr` — имеют значение `#BF5F9C8C`. Первые два символа этого значения — `BF` — представляют собой прозрачность 75 %. Это всего лишь число между 0 и 255, приведенное в шестнадцатеричном виде. Последние шесть — `5F9C8C` — шестнадцатеричное значение цвета.

  Другими словами, преобразуйте значение альфа-прозрачности в шестнадцатеричное число (воспользуйтесь табл. 16.1 как подсказкой). Затем преобразуйте в шестнадцатеричный вид сам цвет (воспользуйтесь калькулятором по адресу **www.javascripter.net/faq/hextorgb.htm**). Теперь совместите оба числа (значение прозрачности идет первым) и присвойте полученное значение параметрам `startColorstr` и `endColorstr`.
- В заключение добавьте свойство `zoom: 1`. Оно необходимо для браузеров IE 6 и IE 7. Подробнее об этом хитром приеме вы можете узнать во врезке «Ошибки браузера» в разд. «Управление обтеканием содержимого плавающих элементов» гл. 7.

Таблица 16.1. Перевод значения альфа-канала в шестнадцатеричное представление

Альфа-канал	Шестнадцатеричный эквивалент
0 (полностью прозрачный)	00
.1	19
.2	33
.3	4C
.4	66
.5	7F
.6	99
.7	B2
.8	CC
.9	E6
1 (непрозрачный)	FF

Тень для текста

Как вы, наверное, помните из обучающего урока гл. 8 по созданию галереи картинок, не существует никаких средств для создания объема на веб-странице. В CSS 3 включено одно такое свойство, которое позволяет добавить тень тексту, чтобы придать ему глубины, например заголовкам, спискам или отдельным абзацам (рис. 16.7). Это свойство `text-shadow`.

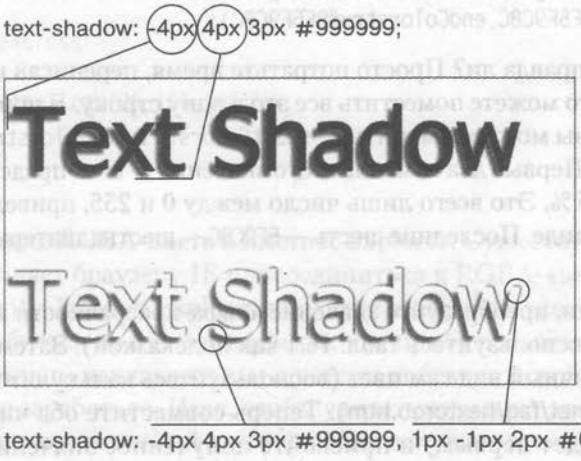


Рис. 16.7. Свойство text-shadow — прекрасная возможность добавить нежности (или, если настаиваете, жесткости) заголовкам либо другому тексту. Однако это чудо не работает в Internet Explorer и версиях Firefox младше 3.5

Свойству `text-shadow` необходимо присвоить четыре параметра: смещение по горизонтали (на какое расстояние должна выступать тень слева или справа

от текста), смещение по вертикали (как далеко сверху или снизу от текста должна находиться тень), степень размытости тени и ее цвет. Вот пример кода, использующего свойство text-shadow для создания эффекта, как на рис. 16.7:

```
text-shadow: -4px 4px 3px #999999;
```

Первое значение -4px можно прочесть как «поместить тень на 4 пиксела левее текста», тогда как положительное значение разместит тень правее текста. Второй параметр — 4px — размещает тень на 4 пиксела ниже текста. Отрицательное значение параметра расположит тень выше текста. Значение 3px определяет смазанность тени. Нулевое значение (полное отсутствие смазанности) даст очень четкую и резкую тень, в то время как большие значения сделают ее еще более размытой. И наконец, последний параметр определяет цвет отбрасываемой тени.

Кроме этого, у вас есть возможность добавить множество отбрасываемых теней для создания более сложных эффектов, как в нижней части рис. 16.7. Просто разделите запятой значения параметров дополнительных теней так, как это показано ниже:

```
text-shadow: -4px 4px 3px #666, 1px -1px 2px #000;
```

Нет никаких ограничений на количество теней, которые вы можете добавить таким образом, за исключением лишь вашего вкуса. Жаль, но этот эффект будет работать только в браузерах Firefox 3.5, Safari, Chrome и Opera. Все версии Internet Explorer, а также Firefox 3 и более ранние его версии просто проигнорируют данное свойство. Именно поэтому не стоит связывать с этим эффектом степень читаемости текста. Эффект, приведенный в нижней части на рис. 16.7, это объясняет: текст, который вы видите, белого цвета, и он читаем исключительно по той причине, что отбрасываемые тени создают его границы. В IE этот текст не будет виден: получится белый текст на белом фоне.

ПРИМЕЧАНИЕ

Если вам очень хочется добавить к тексту тени, можете воспользоваться сценарием, совместимым со всеми браузерами, который добавляет тень не только тексту, но и к любому другому элементу страницы (http://eyebulb.com/dropshadow).

ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

Секундочку, есть еще кое-что!

Эта глава затрагивает возможности CSS 3, наиболее поддерживаемые разработчиками. Рассматриваемые свойства уже сейчас работают в достаточно большом количестве браузеров, а к тому времени, как вы их выучите, станут поддерживаться вообще всеми. В этом разделе представлены несколько новых свойств CSS 3, которые имеют чуть больше ограничений, но, возможно, вас заинтересуют эксперименты с ними.

Скругленные углы. Всем нравятся скругленные углы (по крайней мере, так казалось в 2005 году, во время революции Web 2.0). Однако их так трудно создать. В CSS 3 появилось свойство, которое закругляет углы различных элементов. Вы только представьте кругленький уголок боковой панели без всяких дополнительных картинок и кода. Свойство border-radius (уже доступно в браузерах Firefox, Safari и Chrome)

ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

позволяет управлять кривизной любого или сразу всех четырех углов элемента. Чтобы узнать подробнее, как этого добиться, посетите страницу www.css3.info/preview/rounded-border.

Отбрасываемая тень. Если вам по душе тени вашего текста, то почему бы не попробовать их для различных элементов страницы. Свойство `box-shadow` позволяет задать прямоугольную тень любому элементу. Вы можете сделать боковую панель, парящую над страницей, или просто выделить подвал темной густой тенью. Работает, как и `text-shadow`, но только в Firefox 3.5 и Safari. И даже несмотря на это, вы можете использовать специфичные для каждого браузера свойства (см. врезку «Часто задаваемые вопросы» в начале этой главы). Например, серую дымчатую тень на 10 пикселов правее и ниже блока `<div>` вы можете сделать с помощью следующего стиля:

```
div {
  -webkit-box-shadow: 10px 10px 5px #888;
  -moz-box-shadow: 10px 10px 5px #888;
}
```

Граница в виде изображения. Надоели сплошные, пунктирные и штрихпунктирные линии границ? А как насчет собственных изображений для создания границ? Воспользуйтесь свойством `border-image` в браузерах Firefox 3.5 и Safari (на самом деле `-moz-border-image` и `-webkit-border-image`). Чтобы взглянуть на них в действии, запустите Firefox 3.5 или Safari и посетите страницы www.css3.info/preview/border-image и http://ejohn.org/blog/border-image-in-firefox.

Свобода для шрифтов

Как уже говорилось в гл. 6, простое свойство `font-family` позволяет вам выбрать тип шрифта для CSS-стиля. Но, к сожалению, вы сможете использовать только однотипные шрифты, указывая их в качестве атрибута для `font-family`. При этом следует учитывать, что посетители вашего сайта увидят этот шрифт только в том случае, если он будет установлен на их компьютере. Поскольку большинство людей не приобретают для себя дополнительные шрифты, веб-дизайнеры стараются использовать подручные «веб-безопасные» шрифты, которые имеются у большинства пользователей (см. гл. 6).

Благодаря директиве `@font-face` в CSS 3 перед вами открываются новые возможности. Используя это нововведение, вы можете поместить на своем веб-сервере файл шрифта в формате Open Type (`.otf`) или True Type (`.ttf`), а затем воспользоваться директивой `@font-face`, чтобы браузер посетителя скачал указанный шрифт на время просмотра сайта. Однако перед тем, как вы запрыгаете от счастья, ознакомьтесь с некоторыми вещами.

- **Директива @font-face работает не во всех браузерах, а только в Firefox 3.5, Safari 3, Chrome и Opera 10.** Более ранние версии этих браузеров и Internet Explorer вернутся к старому методу отображения шрифтов, то есть будут применять предустановленные пользователем шрифты. Правда, существует одно исключение: Internet Explorer, начиная с 4 версии, понимает `@font-face`, но использует только специальный формат шрифта — Embedded Open Type (`.eot`), который можно создать, используя программу-конвертер, работающую лишь в Windows (http://msdn.microsoft.com/en-us/library/ms533034.aspx).

ПРИМЕЧАНИЕ

И все же у вас есть возможность заставить работать директиву @font-face как в IE, так и в других современных браузерах. Если интересно, то вы можете прочесть об этом процессе на странице http://jontangerine.com/log/2008/10/font-face-in-ie-making-web-fonts-work. Есть еще один относительно простой способ получения нужного вам результата, который будет работать только при включенном JavaScript: http://wiki.github.com/sorccu/cufon/about.

- **Большой размер файла шрифта может замедлить скорость работы сайта.** Когда вы используете директиву `@font_face`, предусматривается, что посетители должны скачать файл шрифта целиком. Причем этот файл не маленький. Простой шрифт, содержащий только одно начертание символов (полужирный — это уже другое начертание, то есть второй файл), имеет размер около 172 Кбайт. Файлы более сложных шрифтов во много раз больше. Прежде чем ваши посетители смогут прочесть текст, им придется подождать, пока шрифт для этого текста не загрузится. Если файл шрифта маленький, а соединение с Интернетом хорошее, то они, конечно, ничего и не заметят. Но если файлов со шрифтами много, то ваш сайт будет загружаться очень долго.
- **Наличие правовых ограничений.** Шрифты считаются программным обеспечением, поэтому они защищены законом об авторских правах. Большинство шрифтов являются коммерческим продуктом, распространяемым, например, такими компаниями, как Adobe.com или Fonts.com, а значит, равно как и другое программное обеспечение, они находятся под защитой лицензий, которые определяют, где и при каких условиях следует использовать эти шрифты. В большинстве случаев у вас нет права на размещение этих коммерческих шрифтов на своем веб-сервере для последующего использования вместе с директивой `@font-face`. Кроме того, вы размещаете у себя на сервере файл шрифта, который фактически становится доступным для скачивания каждому. Тем не менее существует обилие свободно распространяемых шрифтов, которые можно встраивать с помощью директивы `@font-face`. Взгляните хотя бы на эту страницу: www.tinyurl.com/font-face-rule.

Если не считать этих, надеюсь, временных проблем, директива `@font-face` является замечательным средством разработки для веб-дизайнеров. Для нее требуется задать два параметра: во-первых, название для шрифта, которое будет использоваться в вашем стиле, а во-вторых, URL, указывающий на месторасположение файла шрифта. Вот, например, типичное использование директивы `@font-face`:

```
@font-face {
  font-family: Lavoisier;
  src: url('lavoisier.otf');
}
```

Итак, свойство `font-family` определяет имя, которое вы будете использовать в стиле. В нашем случае это `Lavoisier`. Совсем не обязательно это имя должно совпадать с настоящим именем шрифта, который вы собираетесь использовать. Шрифт можно назвать «Site Font» или же MyFont.

ПРИМЕЧАНИЕ

Если в названии шрифта используется более одного слова, то вам следует поместить его в кавычки. Например, так делать неправильно:

font-family: Site Font;

А вот так верно:

font-family: "Site Font";

Свойство `src` используется для указания URL-адреса шрифта как для формата True Type (`.ttf`), так и для Open Type (`.otf`). Как и везде, где используется адрес URL (например, для фонового изображения), путь к файлу указывается относительно файла CSS-стиля. Так, если вы используете внешний файл стиля, то путь должен быть указан начиная с этого файла и до того места, где на сервере расположен файл шрифта. Вы также можете задавать абсолютные адреса.

Лишь однажды определив имя шрифта, вы можете указывать его в своем файле стилей. Рассмотрим пример того, как в теге `<h1>` используется шрифт `Lavoisier`:

```
h1 {
  font-family: Lavoisier, Arial, Helvetica, sans-serif;
  color: #FF993E;
  font-size: 48px;
}
```

ПРИМЕЧАНИЕ

И все-таки это замечательная идея — указывать имя шрифта в списке лишь после того, как он определен с помощью @font-face, как в примере выше: font-family: Lavoisier, Arial, Helvetica, sans-serif. Так вы справляетесь с браузерами, которые не понимают директиву @font-face.

Директива `@font-face` предоставляет еще одну полезную возможность. Скажем, вы хотите использовать шрифт, который достаточно распространен, но, к сожалению, не во всех браузерах. Так, например, некоторые шрифты установлены на все компьютеры с Windows, но их нет у пользователей Apple Mac. Пользователи Windows только напрасно потратят время (и пропускную способность канала), загружая то, что у них уже есть. В таком случае у вас есть возможность указать локальный шрифт. Если браузер найдет его на компьютере, то не станет загружать его снова. Вот пример:

```
@font-face {
  font-family: Lavoisier;
  src: local(Lavoisier), url('lavoisier.otf');
}
```

В данном случае вам необходимо писать то имя шрифта, которое используется в вашей системе (подробнее в примечании ниже). Вернемся к примеру: если посетитель уже имеет на своем компьютере шрифт Lavoisier, то браузер не станет его скачивать. Но если этот шрифт не будет найден, браузер его скачает.

ПРИМЕЧАНИЕ

Чтобы отыскать название шрифта, вы можете воспользоваться списком доступных шрифтов, например, в программе Microsoft Word. Кроме того, пользователи Windows могут применять Панель управления для доступа к установленным в компьютере шрифтам (www.ehow.com/how_2148826_access-fonts-control-panel.html). Пользователи Apple Mac могут открыть программу управления шрифтами: Application ▸ Font Book (Приложение ▸ Шрифты).

Генерируемое содержимое страницы

Иногда появляется необходимость добавить какой-нибудь небольшой текст, но так, чтобы это не затрагивало HTML-код страницы. Примером может служить желание добавить заголовок «Announcement» перед самым первым абзацем колонки новостей (рис. 16.8). Конечно, можно осуществить это, внеся изменения в HTML, но потребуется сделать дополнительную работу и ввести код. Кроме того, это дополнение не совсем относится к самому тексту объявления. Это лишь вступительная, вводная часть текста. Если же впоследствии вам захочется изменить слово «Announcement» на «News», вы встретитесь с трудоемкой задачей поиска нужного текста среди всех страниц сайта и его последующего изменения.

Однако существует более простой путь, чтобы внести изменения, не затрагивая при этом настоящего содержимого текста. Такая возможность уже присутствует в CSS 2.1 и называется *генерируемым содержимым страницы* (generated content). Тем не менее эта возможность была не очень полезна до момента выхода браузера Internet Explorer 8. Теперь же все основные браузеры, за исключением IE 6 и IE 7, могут ее использовать.

ПРИМЕЧАНИЕ

Вы уже встречались с генерируемым содержимым страницы ранее как с частью способа, решающего проблемы с маркированными или нумерованными списками (см. врезку «Информация для опытных пользователей» в разд. «Стилизация списков» гл. 6).

Генерируемое содержимое добавляется с помощью свойства `content` и псевдоэлементов `:before` или `:after`. Псевдоэлемент `:before` используется для того, чтобы поместить дополнительный текст *перед* нужным тегом, а псевдоэлемент `:after` соответственно *после*. Например, вы хотите добавить символ абзаца ¶ в самом начале каждого абзаца как дань уважения типографике или просто для красоты. Вот необходимый вам стиль:

```
p:before {
  content: "¶";
}
```

Селектор `p:before` указывает на самое начало содержимого абзаца, а свойство `content` определяет то, что вы хотите там видеть. В нашем примере это значок абзаца ¶. Между кавычками вы можете поместить любой текст, и он появится в указанном месте: перед самим тегом или в его конце. Будьте внимательны! Текст в кавычках выводится так, как есть, а значит, у вас нет возможности включать

HTML-теги в качестве генерируемого содержимого. Нет, конечно же, вы можете их там указать, но они отобразятся на экране как обычный текст. Вот пример, поясняющий эту мысль:

```
p:before {
  content: "<h2>Announcement</h2>";
}
```

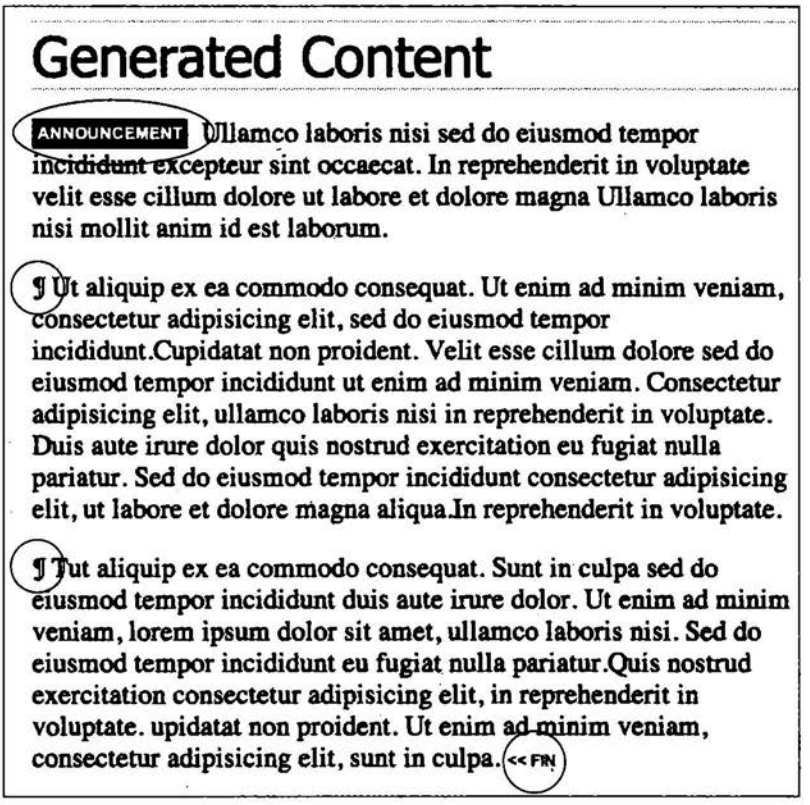


Рис. 16.8. Генерируемое содержимое страницы позволяет внести изменения в основной текст без дополнительного HTML-кода

Вы не увидите заголовка второго уровня, только текст <h2>Announcement</h2>.

ПРИМЕЧАНИЕ

На всякий случай: символ абзаца ¶ имеет техническое название pilcrow.

У вас также есть возможность вставить изображение, указав его адрес:

```
p:before {
  content: url("images/symbol.png");
}
```

Или даже совместить текст и изображение так, как это сделано ниже:

```
p:before {
  content: "Paragraf" url("images/symbol.png");
}
```

Этот код выводит слово «Paragraf», а следом за ним рисунок `symbol.png` перед всеми абзацами текста.

ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

Будущее верстки

Дизайнеры всегда ищут способы для того, чтобы создавать красоту, опираясь на недостаточно развитые технологии. Как пример можно привести использование таблиц как основы для создания макетов страниц (абсолютное и относительное позиционирование рассмотрено в гл. 12 и 13 соответственно). Тем не менее ни одна из этих техник не позволяет добиться того уровня гибкости, которым обладают программы для издательского дела. И все же рабочая группа создателей CSS 3 представила на суд профессионалов три разных модуля, которые должны помочь сделать верстку веб-страниц более простой.

Вы все читали журналы, а значит, знаете, что узкие колонки читать легче, нежели текст на всю ширину страницы. Сейчас не существует простого способа создания колонок на веб-странице, но новый модуль `multi-column layout` должен значительно облегчить эту задачу. Браузеры Firefox и Safari уже поддерживают этот модуль. Подробная информация и примеры его использования находятся по адресу www.css3.info/preview/multi-column-layout.

Модуль `grid positioning` — это другой набор свойств CSS 3, призванный облегчить жизнь графических дизайнеров (www.w3.org/TR/css3-grid). Основная цель данного модуля — предоставить возможность создать невидимую решетку — сеть ячеек, которую затем можно использовать для позиционирования элементов и придания им нужного размера. Эти свойства позволят вам разделить страницу, скажем, на шесть колонок шириной 150 пикселов каждая, а затем, используя встроенные направляющие и линейки, разместить все элементы на странице: «Сделай вот этот блок шириной три колонки и помести его вот сюда, во вторую верхнюю колонку». Жаль, что этому модулю еще далеко до того, чтобы стать реальностью, и это значит, что пока ни один из браузеров его не поддерживает.

И наконец, третье наиболее интересное дополнение: модуль `template layout` (www.w3.org/TR/css3-layout). Его свойства позволят вам использовать буквы, чтобы определять базовую структуру страниц. Вот пример, который делит страницу на два ряда. Верхний ряд представлен буквами `aaa` и является одним целым, а нижний — `bcd` — состоит из трех отдельных столбцов:

```
body {
  display: "aaa"
  "bcd";
}
```

Размещение элементов внутри страницы происходит таким же образом:

```
#head { position: a }
#nav { position: b }
#adv { position: d }
#body { position: c }
```

Итак, элемент с ID `head` занимает весь верхний ряд, в то время как `nav`, `adv` и `body` находятся ниже бок о бок, занимая каждый свое собственное место. Потрясающая вещь, но, к несчастью, этому модулю, как и предыдущему, далеко до реальности.

Используя свойства CSS, вы можете оформлять генерируемое содержимое, равно как и все остальные элементы страницы.

Из примера на рис. 16.8 видно, что слово «Announcement» использует другой шрифт и имеет другой цвет фона. Этот первый абзац принадлежит классу `.announcement`, а ниже я привожу его стиль, который создает вид, как на вышеупомянутом рисунке:

```
.announcement:before {
  content: "ANNOUNCEMENT";
  font: bold .6em Arial, Helvetica, sans-serif;
  color: #FFF;
  padding: 4px;
  background-color: red;
  margin-right: 10px;
}
```

Все перечисленные свойства применяются к самому тексту «ANNOUNCEMENT», так как находятся непосредственно в объявлении псевдоэлемента `:before`.

Приложения

Приложение 1. Справочник свойств CSS
Приложение 2. CSS в Dreamweaver CS4
Приложение 3. Ресурсы по CSS

1 Справочник свойств CSS

Владение на профессиональном уровне каскадными таблицами стилей (CSS) означает знание того, как использовать огромное количество свойств CSS, которые управляют внешним видом текста, изображений, таблиц и форм. Чтобы помочь вам в ваших поисках, в этом приложении я собрал свойства и их значения, которые вы будете использовать для создания собственных стилей. Этот список охватывает практически все стандартные свойства CSS 2.1 — те, которые поддерживает большинство браузеров.

ПРИМЕЧАНИЕ

В приложении не описываются свойства, которые ни один браузер не распознает. Если свойство работает только в определенных браузерах, я привожу их список. Для получения полной информации просмотрите спецификацию CSS 2.1 Консорциума Всемирной паутины (W3C) на сайте www.w3.org/TR/CSS21/. О некоторых новых свойствах в CSS 3 вы можете прочитать в гл. 16. Правда, они работают не во всех браузерах.

Значения CSS

У каждого свойства CSS есть соответствующее ему значение. Свойство `color`, которое форматирует цвет шрифта, требует установки значения, чтобы определить, какой цвет вы хотите использовать. Свойство `color: #FFF;` задает белый текст. Разные свойства требуют указания различных значений, каждое из которых относится к одной из четырех основных категорий: цвета, размеры, ключевые слова и адреса URL.

Цвета

Вы можете назначать цвета многим свойствам, включая свойства для шрифта, фона и границ. CSS предоставляет несколько различных способов определения цвета.

Ключевые слова

Ключевое слово, обозначающее цвет, — это просто название цвета, например `white` или `black` (белый или черный). В настоящее время существует 17 признанных ключевых слов, обозначающих цвета: `aqua`, `black`, `blue`, `fuchsia`, `gray`, `green`, `lime`, `maroon`, `navy`, `olive`, `orange`, `purple`, `red`, `silver`, `teal`, `white` и `yellow`. Некоторые браузеры принимают больше ключевых слов, а CSS 3 обещает предложить их еще больше в будущем (http://www.w3.org/TR/css3-color/). Например, это цвет RGBA, описанный в гл. 16.

Значения RGB

Мониторы компьютеров определяют оттенки, используя красный, зеленый и синий цвета. Эти RGB-значения могут создать почти весь спектр цветов. Практически каждый дизайн, рисунок и графическая программа позволяют вам определить цвета, используя RGB, так что можно легко передавать цвет из одной из этих программ CSS-свойству. CSS представляет значения RGB несколькими способами.

- **Hex-значения.** Это метод, обычно используемый в Сети для идентификации цвета. Hex-значения цветов состоят из трех двухсимвольных чисел в шестнадцатеричной (то есть с основанием 16) системе счисления. `#FF0033` представляет RGB-значение, составленное из красного (`FF`, что равняется 255 в десятичной системе счисления), зеленого (`00`) и синего (`33`). Символ # указывает CSS, что впереди следует ожидать шестнадцатеричное число, и является обязательным. Если вы пропустите #, браузер не отобразит нужный цвет.

СОВЕТ

Если во всех трех значениях цифры повторяются, можно сократить hex-значение, используя только первое число каждой пары. Например, #361 означает то же самое, что и #336611.

- **Проценты RGB.** Вы также можете определить цвет, используя значения в процентах, например: `rgb (100%, 0%, 33%)`. Вы можете получить эти числа из программ редактирования изображений и дизайна, которые могут определять цвета, используя проценты.
- **Десятичные значения.** Наконец, можно применять десятичные RGB-значения, чтобы определить цвет. Формат подобен тому, что был в варианте с процентами, но вы используете число от 0 до 255 для указания каждого цвета: `rgb (255, 0, 33)`.

Не имеет значения, какой метод вы используете, — они все работают. Для последовательности вы должны выбрать один способ определения RGB-значения и придерживаться его. В операционных системах Windows и Mac есть выборщики цвета, которые позволяют подобрать идеально подходящий цвет из всей палитры цветов, а затем показывают вам его RGB-значение. В качестве альтернативы можно применять бесплатный выборщик цвета с сайта www.ficml.org/jemimap/style/color/wheel.html или более продвинутый выборщик, позволяющий создавать и сохранять палитру цветов, с сайта http://kuler.adobe.com.

Длины и размеры

CSS предоставляет множество различных способов определить размер шрифта, ширину поля или толщину границы. Чтобы указать размер шрифта, вы можете использовать дюймы, цицеро, точки, сантиметры, миллиметры, единицы измерения em и ex, пикселы и проценты.

Однако при всем многообразии единиц измерения многие из них не относятся к миру экранного отображения по причинам, обсуждаемым в разд. «Установка размера шрифта» гл. 6. На самом деле вам стоит задуматься только о трех: пикселах, em и процентах.

Пикселы

Пиксел — это одна точка на экране компьютера. Пикселы дают последовательный метод идентификации длин и размеров шрифта от компьютера к компьютеру: 72 пиксела на одном мониторе составляют 72 пиксела на другом мониторе. Это тем не менее не означает, что фактическая длина в реальном мире будет одной и той же для всех. Поскольку люди устанавливают для своих мониторов различные разрешения — 800 × 600, 1024 × 768, 1600 × 1200 или еще какое-нибудь — 72 пиксела могут занять 1 дюйм на одном мониторе и всего 0,5 дюйма — на другом. Однако пикселы дают вам наиболее последовательный контроль над отображением.

ПРИМЕЧАНИЕ

Есть только один недостаток в использовании пикселов: люди, которые пользуются Internet Explorer версии 6 или ниже, не могут изменить размеры элемента, если они заданы в пикселах. Если ваш текст окажется слишком маленьким для кого-то, то посетитель будет не в состоянии увеличить его, чтобы сделать более приемлемым для чтения (см. разд. «Установка размера шрифта» гл. 6).

Единица измерения em

В типографии em — это единица измерения, которая представляет высоту заглавной буквы M для определенного шрифта. В веб-дизайне 1 em — это высота базового шрифта в браузере, которая обычно составляет 16 пикселов. Однако любой может изменить эту базовую установку размера, так что 1 em будет равняться 16 пикселам в одном браузере и 24 пикселам — в другом. Другими словами, em — это относительная единица измерения.

Вдобавок к исходной установке размера шрифта в браузере em может унаследовать информацию о размере от тегов. Размер `.9em` установит высоту текста приблизительно 14 пикселов в большинстве браузеров с основным размером шрифта 16 пикселов. Но если у вас есть тег `<p>` с размером шрифта `.9ems`, а затем тег `<strong>` с размером шрифта `.9ems` внутри этого тега `<p>`, то em-размер тега `<strong>` составит не 14, а 12 пикселов (16 × 0,9 × 0,9). Так что помните о наследовании, когда применяете em-значения.

Проценты

CSS использует проценты в различных целях, таких как установка размеров текста, определение ширины или высоты элемента, размещение изображения в фоне стиля и т. д. Для размеров шрифта проценты вычисляются, основываясь на унаследованном значении текста. Скажем, общий размер шрифта для абзаца — 16 пикселов. Если вы создали стиль для отдельного абзаца и установили размер его шрифта равным 200 %, то этот текст отображается высотой 32 пиксела. Однако при применении к ширине проценты вычисляются на основе ширины страницы или другого родительского элемента с заданной шириной.

Ключевые слова

У многих свойств есть свои собственные специфические значения, которые влияют на то, как проявляют себя свойства. Они представлены ключевыми словами. Свойство `text-align`, выравнивающее текст на экране, может принять одно из четырех ключевых слов: `right`, `left`, `center` и `justify`. Поскольку ключевые слова из-

меняются от свойства к свойству, читайте описания свойств, которые идут далее, чтобы узнать, какие ключевые слова им соответствуют.

Есть одно ключевое слово, которое используется всеми свойствами, — inherit. Оно заставляет стиль унаследовать значение от родительского элемента. Ключевое слово inherit дает возможность заставить стили унаследовать свойства, которые обычно не наследуются от родительских тегов. Например, вы используете свойство border, чтобы добавить границу вокруг абзаца. Остальные теги, такие как и внутри тега <p>, не наследуют это значение, но вы можете указать им сделать это с помощью ключевого слова inherit:

```
em, strong {
  border: inherit;
}
```

Таким образом, в этом случае теги и отображают одно и то же значение border, как и их родительский тег <p>. Так, элементы и абзаца оба получают свои собственные границы, как и весь текст абзаца. Если бы они изначально наследовали значение свойства border, то у вас получились бы одни границы внутри других (а это серьезное основание, чтобы не наследовать данное свойство). Если вы изменяете значение border тега <p> на другой цвет или толщину линии, теги и также наследуют это значение и отображают такой же вид границы.

ПРИМЕЧАНИЕ

Границы — не очень полезный пример, главным образом потому, что inherit — не очень полезное значение. Но это не было бы полным руководством, если бы я не рассказал обо всех этих фактах.

URL-адреса

Значения URL позволяют указывать на другой файл в Сети. Например, свойство background-image принимает URL — путь к файлу в Интернете — в качестве своего значения, которое позволяет вам назначать графический файл как фон для элемента страницы. Эта методика удобна при добавлении повторяющегося изображения в фон страницы или при использовании собственной графики для маркированных списков (см. разд. «Фоновые изображения» гл. 8).

В CSS вы определяете URL так: url(images/tile.gif). Стиль, который добавляет изображение, названное tile.gif, к фону страницы, выглядел бы следующим образом:

```
body { background-image: url(images/tile.gif); }
```

В отличие от HTML, в CSS кавычки вокруг URL являются необязательными, так что url ("images/tile.gif"), url('images/tile.gif') и url(images/tile.gif) эквивалентны.

ПРИМЕЧАНИЕ

Сам URL такой же, как и HTML-атрибут href, используемый для ссылок, что означает то, что вы можете использовать абсолютный URL, такой как http://www.missingmanuals.com/images/tile.gif, путь относительно корня, такой как /images/tile.gif, или URL относительно документа, например ../../images/tile.gif. Прочитайте врезку «Информация для новичков» в разд. «Фоновые изображения» гл. 8 для получения полной информации об этих видах путей.

Свойства текста

Следующие свойства влияют на форматирование текста на веб-странице. Поскольку большинство свойств этой категории наследуются, вам не обязательно применять их к тегам, специально предназначенным для текста (таким как тег <p>). Вы можете применить эти свойства к тегу <body> так, чтобы остальные теги унаследовали и использовали те же самые настройки. Применение этой методики — быстрый способ создать всеохватывающий шрифт, цвет и т. д. для страницы или раздела.

color

Устанавливает цвет текста. Поскольку оно наследуется, то, если вы задаете красный цвет для тега <body>, например у всего текста на странице, и у всех других тегов внутри тега <body> также будет красный цвет.

Значения: любое корректное значение цвета.

Пример: `color: #FFFF33;`.

ПРИМЕЧАНИЕ

Заранее установленный цвет ссылок для тега <a> отменяет наследование цвета. В вышеупомянутом примере любые ссылки в теге <body> все еще были бы стандартного синего цвета. Чтобы узнать, как изменить заранее установленный цвет ссылок, читайте разд. «Выборка стилизуемых ссылок» гл. 9.

font

Используя это свойство, вы можете сокращенно определить следующие текстовые свойства в отдельном объявлении стиля: `font-style`, `font-variant`, `font-weight`, `font-size`, `line-height` и `font-family`.

В этом свойстве между значениями нужно ставить пробелы и включить по крайней мере `font-size` и `font-family`, причем они должны быть последними двумя элементами в объявлении. Остальные являются дополнительными. Если вы не установите свойство, браузер будет использовать предустановленное значение, потенциально отменяя унаследованные свойства.

Значения: любое значение, корректное для определенного свойства шрифта. Если вы включаете значение `line-height`, то задавайте высоту линии, слеш (/), а затем размер шрифта, например: `1.25em/150%`.

Пример: `font: italic small-caps bold 1.25em/150% Arial, Helvetica, sans-serif;`.

font-family

Указывает шрифт, который браузер должен использовать при отображении текста. Шрифты обычно определяются как наборы трех-четырех параметров, учитывая тот факт, что конкретный шрифт может быть не установлен на компьютере посетителя (см. разд. «Стилизация текста» гл. 6).

Значения: названия шрифтов, разделенные запятыми. Когда название шрифта состоит из нескольких слов, между которыми стоит пробел, это название нужно указывать в кавычках. Последний из перечисленных шрифт — обычно общий тип шрифта, указывающий браузерам выбрать подходящий шрифт, если остальные недоступны: `serif`, `sans-serif`, `monotype`, `fantasy` или `cursive`.

Пример: `font-family: "Lucida Grande", Arial, sans-serif;`.

font-size

Устанавливает размер текста. Это свойство наследуется, что может привести к некоторому странному поведению при использовании относительных единиц измерения, таких как проценты и em.

Значения: любая корректная единица измерения CSS (см. подраздел «Длины и размеры» разд. «Значения CSS» этого приложения), а также следующие ключевые слова: `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large`, `xx-large`, `larger` и `smaller`; `medium` представляют обычный, заранее заданный размер шрифта браузера, а остальные размеры — кратные ему. Каждое из них увеличивает или уменьшает текст в определенной степени. Несмотря на то что все изменения цвета, как предполагается, должны быть последовательными увеличениями или уменьшениями от предыдущего значения, на самом деле это не так. По существу, `xx-small` эквивалентен 9 пикселам (берем в расчет то, что вы не корректировали базовый размер шрифта браузера); `x-small` — 10 пикселам, `small` — 13, `large` — 18, `x-large` — 24 и `xx-large` — 32 пикселам. Из-за неуверенности в том, как каждый браузер обрабатывает эти ключевые слова, многие дизайнеры используют вместо них пикселы, em или проценты.

Пример: `font-size: 1.25em;`.

ПРИМЕЧАНИЕ

Когда свойство font-size наследуется от другого тега, эти ключевые слова умножают унаследованный размер шрифта на тот же самый коэффициент (1,2 в большинстве браузеров).

font-style

Делает текст курсивным. Если свойство применено к курсивному тексту, то возвращает его вновь к обычному. Значения `italic` и `oblique` функционально одинаковы.

Значения: `italic`, `oblique`, `normal`.

Пример: `font-style: italic;`.

font-variant

Делает буквы в тексте прописными, например: SPECIAL PRESENTATION. Значение `normal` задает шрифт, не содержащий прописных букв.

Значения: `small-caps`, `normal`.

Пример: `font-variant: small-caps;`.

font-weight

Делает текст полужирным или отменяет это начертание для текста, который уже был отформатирован таким образом.

Значения: CSS на самом деле предоставляет 14 различных ключевых слов для свойства `font-weight`, но только два из них фактически работают с сегодняшними браузерами и компьютерными системами: `bold` и `normal`.

Пример: `font-weight: bold;`.

letter-spacing

Регулирует расстояние между буквами, чтобы растянуть слова (добавляя расстояние между буквами) или сжать их (удаляя расстояние).

Значения: любая корректная единица измерения CSS, хотя em и пикселы распространены больше всего. Проценты для этого свойства не работают в большинстве браузеров. Используйте положительное значение, чтобы увеличить расстояние между буквами, и отрицательное значение, чтобы убрать промежутки. Значение `normal` сбрасывает `letter-spacing` в стандартное значение в браузере — `0`.

Примеры: `letter-spacing: -1px; letter-spacing: 2em;`.

line-height

Регулирует расстояние между строками текста в абзаце (в программах для обработки текста часто называется *межстрочным интервалом*). Нормальная высота линии составляет 120 % от размера текста (см. разд. «Форматирование абзацев текста» гл. 6).

Значения: большинство корректных размеров CSS (см. подраздел «Длины и размеры» разд. «Значения CSS» этого приложения), хотя em, пикселы и проценты наиболее распространены.

Пример: `line-height: 200%;`.

text-align

Выравнивает блок текста по левому, правому краю или по центру страницы либо элемента-контейнера.

Значения: `left`, `center`, `right`, `justify` (значение `justify` часто затрудняет чтение текста на мониторах).

Пример: `text-align: center;`.

text-decoration (наследуется)

Добавляет линии над или под текстом, а также посередине. Подчеркивание распространено для ссылок, так что не стоит подчеркивать текст, который не является ссылкой. Цвет линии такой же, как цвет шрифта тега, к которому применен стиль. Свойство также поддерживает значение `blink`, которое делает текст мерцающим (но большинство браузеров так или иначе игнорируют значение `blink`).

Значения: underline, overline, line-through, blink, none. Значение none отключает все оформление. Используйте его, чтобы скрыть подчеркивание, которое обычно появляется под ссылками. Вы также можете добавить множество оформлений, перечисляя значения (кроме none) через пробел.

Пример: `text-decoration: underline overline line-through;`.

text-indent

Устанавливает размер отступа для первой строки блока текста. Первая строка может иметь отступ (как во многих печатных книгах) или выступать над левым краем остального текста.

Значения: любая корректная единица измерения CSS. Пикселы и em наиболее распространены; проценты ведут себя иначе, чем со свойством font-size. Здесь проценты основываются на ширине области, содержащей текст, которая может быть шириной всего окна браузера. Таким образом, 50 % сделают отступ первой строки на половину окна (см. разд. «Определение параметров высоты и ширины» гл. 7). Чтобы первая строка выступала за левым краем, используйте отрицательное значение. Эта методика хорошо работает в связке с положительным свойством margin-left, которое добавляет отступ/выступ с левой стороны других строк текста на установленную величину.

Пример: `text-indent: 3em;`.

text-transform

Изменяет регистр букв в тексте так, что все буквы в тексте становятся строчными или только первая буква каждого слова остается прописной.

Значения: uppercase, lowercase, capitalize, none. Значение none возвращает текст к какому бы то ни было регистру из фактического кода HTML. Если aBCDefg — буквы, которые в действительности напечатаны в HTML, то none отменит другие унаследованные установки регистра от тега-предка и отобразит на экране aBCDefg.

Пример: `text-transform: uppercase;`.

vertical-align (наследуется)

Устанавливает базовую линию внутреннего элемента относительно базовой линии окружающего содержимого. С ним вы можете сделать так, чтобы символ появился немного выше или ниже окружающего текста. Используйте его для создания символов верхнего индекса, таких как ™, ® или ©. При применении к ячейке таблицы значения top, middle, bottom и baseline управляют вертикальным размещением содержимого внутри ячейки (см. подраздел «Настройка горизонтального и вертикального выравнивания» разд. «Создание стилей для таблиц» гл. 10).

Значения: baseline, sub, super, top, text-top, middle, bottom, text-bottom, процентное значение или абсолютное значение (например, пикселы или em). Проценты вычисляются на основании значения line-height элемента.

Примеры: `vertical-align: top; vertical-align: -5px; vertical-align: 75%;`.

white-space (наследуется)

Управляет тем, как браузер отображает пробелы в коде HTML. Обычно, если вы включаете более одного пробела между словами, например "Hello Dave", браузер отображает только один пробел — "Hello Dave". При использовании значения pre вы можете сохранить пробелы точно так же, как в HTML. Это значение делает то же, что и HTML-тег <pre>. Кроме того, браузеры разобьют строку текста на месте пробела, если строка не будет соответствовать ширине окна. Чтобы препятствовать обтеканию текста, используйте значение nowrap. Однако это значение вынуждает весь текст абзаца оставаться на одной строке, так что не используйте его с длинными абзацами (если вам, конечно, не хочется заставить посетителей бесконечно прокручивать страницу вправо).

Значения: nowrap, pre, normal. Еще два значения — pre-line и pre-wrap — не работают во многих браузерах.

Пример: white-space: pre;.

word-spacing

Работает так же, как свойство letter-spacing, но регулирует расстояние между словами, а не между буквами.

Значения: любая правильная единица измерения CSS, хотя em и пикселы наиболее распространены; проценты не работают в большинстве браузеров. Используйте положительное значение, чтобы увеличить расстояние между словами, и отрицательное значение, чтобы убрать интервал (сжать слова). Значение normal устанавливает стандартное расстояние между словами (word-spacing), принимаемое за 0.

Примеры: word-spacing: -1px; word-spacing: 2em;.

Свойства списков

Следующие свойства затрагивают форматирование маркированных (<ul>) и нумерованных списков (<ol>).

list-style

Применение этого свойства — сокращенный метод определения трех свойств, перечисленных далее. Вы можете включить значение для одного или более этих свойств, разделяя каждое пробелом. Вы можете даже использовать это свойство как кратчайший путь для написания отдельного свойства и сэкономить время при наборе кода: list-style: outside вместо list-style-position: outside. Если вы определите и тип, и изображение, браузер отобразит стандартный маркер (кружок, квадрат и т. д.), *только* если не сможет найти изображение. Таким образом, если путь к изображению, определенному для маркера, не работает, маркированный список все равно будет с маркерами.

Значения: любое подходящее значение для list-style-type, list-style-image и/или list-style-position.

Пример: list-style: disc url (images/bullet.gif) inside;.

list-style-image

Определяет изображение, которое используется для маркера в маркированном списке.

Значения: значение URL или none.

Пример: list-style-image: url (images/bullet.gif);.

СОВЕТ

Свойство background-image также позволяет определить для маркера изображение и предоставляет больше возможностей управления (см. разд. «Фоновые изображения» гл. 8).

list-style-position

Позиционирует маркеры или числа в списке. Эти маркеры могут появиться за пределами текста, выступая за левый край, или внутри текста (прямо там, где обычно начинается первая буква первой строки). Значение outside определяет стандартное отображения маркеров и чисел.

Значения: inside, outside.

Пример: list-style: inside;.

list-style-type

Устанавливает тип маркера для списка: круг, квадрат, римские цифры и т. д. Вы можете даже превратить неупорядоченный (маркированный) список в упорядоченный (нумерованный), изменяя свойство list-style-type, но это работает не во всех браузерах (включая Internet Explorer для Windows). Используйте значение none, чтобы полностью удалить маркеры или числа из списка.

Значения: disc, circle, square, decimal, decimal-leading-zero, upper-alpha, loweralpha, upper-roman, lower-roman, lower-greek, none.

Пример: list-style-type: square;.

Отступы, границы и поля

Следующие свойства управляют пространством вокруг элемента.

border

Чертит линию вокруг четырех сторон элемента.

Значения: толщина границы задается в любой корректной единице измерения CSS (кроме процентов).

Вы также можете определить стиль для линии: solid, dotted, dashed, double, groove, ridge, inset, outset, none и hidden (см. рис. 7.7, где показаны примеры различных стилей). Значения none и hidden делают одно и то же — удаляют любую границу.

Наконец, вы можете определить цвет, используя любой корректный тип цвета CSS (ключевое слово, например green, или hex-значение, такое как #33fc44).

Пример: `border: 2px solid #f33;`.

border-top, border-right, border-bottom, border-left

Добавляют границу к одному краю. Например, border-top добавляет границу к вершине элемента.

Значения: те же, что и для border.

Пример: `border-left: 1em dashed red;`.

border-color

Определяет цвет, используемый для всех четырех границ.

Значения: любой корректный тип цвета CSS (ключевое слово, например green, или hex-значение, такое как #33fc44).

Пример: `border-color: rgb(255,34,100);`.

У этого свойства есть еще и сокращенная запись, позволяющая присваивать различные цвета каждой из четырех границ.

Значения: любой корректный тип цвета CSS для каждой границы: верхней, правой, нижней и левой. Если вы укажете только два значения, то первое будет присвоено верхней и нижней, а второе — правой и левой границам.

Пример: `border-color: #000 #F33 #030 #438F3C;`.

border-top-color, border-right-color, border-bottom-color, border-left-color

Функционируют так же, как свойство border-color, но устанавливают цвет только для одной границы. Используйте эти свойства, чтобы отменить цвет, установленный свойством border. Таким образом, вы можете настроить цвета для отдельных границ, притом что вы используете более общий стиль border, чтобы определить базовый размер и стиль для всех четырех сторон.

Значения: как и для свойства border-color, описанного выше.

Пример: `border-left-color: #333;`.

border-style

Определяет стиль, используемый для всех четырех границ.

Значения: одно из ключевых слов: solid, dotted, dashed, double, groove, ridge, inset, outset, none и hidden (см. рис. 7.7, где показаны примеры различных стилей). Значения none и hidden действуют одинаково — удаляют любую границу.

Пример: `border-style: inset;`.

У этого свойства есть еще и сокращенная запись, позволяющая присваивать различные стили для каждой из четырех границ: верхней, правой, нижней и левой.

Значения: одно из ключевых слов, упомянутых выше, для каждой из четырех границ. Если вы укажете только два значения, первое будет присвоено верхней и нижней, а второе — правой и левой границам.

Пример: `border-style: solid dotted dashed double;`.

border-top-style, border-right-style, border-bottom-style, border-left-style

Функционируют точно так же, как свойство `border-style`, но применяются только к одному краю (границе).

Значения: как и для свойства `border-style`, описанного выше.

Пример: `border-top-style: none;`.

border-width

Определяет толщину линии, используемой для рисования всех четырех границ.

Значения: любая корректная единица измерения CSS, кроме процентов. Наиболее распространенные — em и пикселы.

Пример: `border-width: 1px;`.

У этого свойства есть еще и сокращенная запись, позволяющая присваивать различную толщину линии для каждой из четырех границ.

Значения: любая корректная единица измерения CSS, кроме процентов, — для каждой из четырех границ. Если вы укажете только два значения, первое будет присвоено верхней и нижней, а второе — правой и левой границам.

Пример: `border-width: 3em 1em 2em 3.5em;`.

border-top-width, border-right-width, border-bottom-width, border-left-width

Функционируют точно так же, как свойство `border-width`, но применяются только к одному краю.

Значения: как и для свойства `border-width`, описанного выше.

Пример: `border-bottom-width: 3em;`.

outline

Применение этого свойства — краткий способ объединить `outline-color`, `outline-style` и `outline-width` (перечислены далее). Контур, задаваемый этим свойством, работает точно так же, как граница, за исключением того, что он не занимает места (то есть не увеличивает ширину или высоту элемента) и относится ко всем четырем краям. Оно больше применяется как средство выделения чего-нибудь на странице, чем как элемент дизайна. Свойство `outline` работает в Firefox, Safari, Chrome, Opera и только в восьмой или более поздней версии Internet Explorer.

Значения: те же самые, что относятся к border, с одним исключением (см. описание свойства outline-color далее).

Пример: outline: 3px solid #F33;.

outline-color

Определяет цвет для контура (см. описание свойства outline выше).

Значения: любой корректный цвет CSS плюс значение invert, которое просто изменяет цвет контура (цвет, на котором расположен контур) на противоположный. Если контур нарисован на белом фоне, значение invert сделает его черным. Работает точно так же, как свойство border-color.

Пример: outline-color: invert;.

outline-style

Определяет тип линии для контура: пунктирная, сплошная, штриховая и т. д.

Значения: те же самые, что и для свойства border-style, описанного выше.

Пример: outline-style: dashed;.

outline-width

Определяет толщину контура. Работает так же, как и свойство border-width.

Значения: любая корректная единица измерения CSS, кроме процентов. Наиболее распространенные — em и пикселы.

Пример: outline-width: 3px;.

padding

Устанавливает размер области между содержимым, границей и краем фона. Используйте его, чтобы добавить пустое пространство вокруг текста, изображений или другого содержимого (см. рис. 7.1 для наглядной демонстрации).

Значения: любая корректная единица измерения CSS, такая как пикселы или em. Значения в процентах основываются на ширине содержащего элемента. Заголовок, являющийся потомком тега <body>, использует ширину окна браузера, чтобы вычислить значение в процентах, так что отступ шириной 20 % добавляет 20 % ширины окна. Если посетитель изменяет размеры своего браузера, размер отступа изменяется пропорционально. Вы можете определить отступ для всех четырех краев, используя одно значение, или установить отдельные размеры отступа для каждого края в таком порядке: top, right, bottom, left.

Примеры: padding: 20px; padding: 2em 3em 2.5em 0;.

padding-top

Работает точно так же, как свойство padding, но устанавливает отступ только для верхнего края.

Пример: padding-top: 20px;.

padding-right

Работает точно так же, как свойство padding, но устанавливает отступ лишь для правого края.

Пример: padding-right: 20px;.

padding-bottom

Работает точно так же, как свойство padding, но устанавливает отступ только для нижнего края.

Пример: padding-bottom: 20px;.

padding-left

Работает точно так же, как свойство padding, но устанавливает отступ лишь для левого края.

Пример: padding-left: 20px;.

margin

Устанавливает размер области между границей элемента и краями других элементов (см. рис. 7.1). Свойство позволяет добавлять пустое пространство между двумя элементами: между двумя изображениями или между боковым меню и областью главного содержимого страницы.

ПРИМЕЧАНИЕ

Вертикальные поля между элементами могут исчезать. Иными словами, браузеры используют только верхнее или нижнее поле и игнорируют остальные, создавая меньший промежуток, чем ожидается (см. подраздел «Конфликты полей» разд. «Управление размерами полей и отступов» гл. 7).

Значения: любая корректная единица измерения CSS, такая как пикселы или em. Значения в процентах основаны на ширине содержащего элемента. Заголовок, который является потомком дочернего тега <body>, использует ширину окна браузера, чтобы вычислить значение в процентах, так что поле шириной 10 % добавляет 10 % ширины окна к краям заголовка. Если посетитель изменяет размеры своего браузера, изменяется размер поля. Как и в случае с отступом, вы определяете поля для всех четырех границ, используя одно значение, или устанавливаете отдельные поля в таком порядке: top, right, bottom, left.

Примеры: margin: 20px; margin: 2em 3em 2.5em 0;.

margin-top

Работает точно так же, как свойство margin, но устанавливает поле только для верхней стороны.

Пример: margin-top: 20px;.

margin-right

Работает точно так же, как свойство `margin`, но устанавливает поле лишь для правой стороны.

Пример: `margin-right: 20px;`.

margin-bottom

Работает точно так же, как свойство `margin`, но устанавливает поле только для нижней стороны.

Пример: `margin-bottom: 20px;`.

margin-left

Работает точно так же, как свойство `margin`, но устанавливает поле лишь для левой стороны.

Пример: `margin-left: 20px;`.

Фон

CSS предоставляет несколько свойств для управления фоном элемента, включая изменение цвета фона, размещения изображения позади элемента и управления тем, как позиционировано это фоновое изображение.

background

Обеспечивает краткий метод определения свойств, которые проявляются в фоне элемента, таких как цвет, изображение и размещение этого изображения. Оно объединяет пять свойств фона (описаны далее) в одну компактную строку так, что вы можете получить тот же самый результат с меньшим количеством кода. Однако если вы не установите одно из этих свойств, браузеры будут использовать стандартное значение. Например, если вы не определите, как должно повторяться фоновое изображение, браузеры будут повторять это изображение слева направо и сверху вниз (см. разд. «Управление повтором фоновых изображений» гл. 8).

Значения: те же самые значения, что используются для свойств фона, перечисленных далее. Порядок свойств неважен (за исключением позиционирования, как описано ниже), но лучше указывать свойства в такой последовательности: `background-color`, `background-image`, `background-repeat`, `background-attachment`, `background-position`.

Пример: `background: #333 url (images/logo.gif) no-repeat fixed left top;`.

background-attachment

Определяет, как реагирует фоновое изображение, когда ваш посетитель прокручивает страницу. Изображение либо прокручивается вместе с остальным содержи-

мым, либо остается на месте. Вы можете добавить логотип к верхнему левому углу очень длинной веб-страницы, используя значение fixed свойства background-attachment, и заставить это изображение находиться в верхнем левом углу, даже когда страница прокручивается (в Internet Explorer версий 6 и ниже это свойство работает только для тега <body>).

Значения: scroll или fixed. Scroll — это обычное поведение: изображение прокручивается вместе с текстом. Fixed закрепляет изображение на месте.

Пример: background-attachment: fixed;.

background-color

Добавляет цвет к фону стиля. Фон находится под границей и под фоновым изображением — это нужно иметь в виду, если вы используете такие стили границы, как dashed или dotted. В этих случаях фоновый цвет просвечивается через промежутки между черточками или точками.

Значения: любое корректное значение цвета (см. подраздел «Цвета» разд. «Значения CSS» этого приложения).

Пример: background-color: #FFF;.

background-image

Помещает изображение в фон стиля. Остальные элементы страницы находятся наверху фонового изображения, так что убедитесь в том, что текст является разборчивым в том месте, где он накладывается на изображение. Вы также можете всегда использовать отступ, чтобы отодвинуть содержимое от изображения. Изображение повторяется слева направо и сверху вниз, если только вы не устанавливаете иное свойством background-repeat.

Значения: URL-адрес изображения.

Примеры: background-image: url(images/photo.jpg); background-image: url(http://www.example.org/photo.jpg);.

background-position

Управляет размещением изображения в фоне (на заднем плане) элемента страницы. Если только вы не определите иначе, изображение начинается в верхнем левом углу элемента. Если изображение повторяется, свойство background-position контролирует начальную точку изображения. Если вы размещаете изображение в центре элемента, то браузер помещает его там, а затем повторяет вверх и влево, а также вниз и вправо. Во многих случаях точное расположение изображения не вызывает видимого различия при повторении фона, но позволяет вносить едва заметные изменения в позиционирование образца в фоне.

Значения: вы можете использовать любую корректную единицу измерения CSS, такую как пикселы или em, а также ключевые слова или проценты. Значения идут парами, где первое является горизонтальной позицией, а второе — вертикальной. Ключевые слова: left, center и right для горизонтального позиционирования и top,

center и bottom — для вертикального. Значения в пикселах и em рассчитываются от верхнего левого угла элемента, поэтому, чтобы поместить изображение на расстоянии 5 пикселов от левого края и 10 пикселов от верхнего, нужно использовать значение 5px 10px.

Значения в процентах устанавливают соответствие между точкой изображения и точкой в фоне элемента, рассчитанное указанными процентными отношениями от левого и верхнего краев изображения и от левого и верхнего краев элемента. Значение 50% 50% помещает изображение прямо посередине элемента (см. подраздел «Процентные значения» разд. «Позиционирование фоновых изображений» гл. 8). Вы можете смешать и сопоставить эти значения: если хотите, используйте значение в пикселах для горизонтального выравнивания и значение в процентах для вертикального.

Примеры: `background-position: left top; background-position: 1em 3em; background-position: 10px 50%;`.

background-repeat

Управляет тем, повторяется ли фоновое изображение, и если повторяется, то как. Обычно фоновое изображение повторяется от левого верхнего до правого нижнего края, заполняя весь фон элемента.

Значения: repeat, no-repeat, repeat-x, repeat-y. Значение repeat стандартное, оно повторяет изображение слева направо, сверху вниз. No-repeat помещает изображение в фоне один раз без какого-либо повторения. Repeat-x повторяет изображение только сверху вниз — это удобно при добавлении графического бокового меню. Repeat-y повторяет изображение только слева направо, так что вы можете добавить графическую полосу вверху, посередине или внизу элемента.

Пример: `background-repeat: no-repeat;`.

Свойства разметки страницы

Следующие свойства управляют размещением и размером элементов на веб-странице.

bottom

Это свойство применяется с абсолютным, относительным и фиксированным позиционированием. При использовании с абсолютным или фиксированным позиционированием bottom определяет позицию нижнего края стиля относительно нижнего края ближайшего расположенного к нему предка. Если стилизованный элемент не находится внутри каких-либо позиционированных тегов, то он будет размещаться относительно нижнего края окна браузера. Вы можете использовать это свойство, чтобы поместить сноску внизу окна браузера. При использовании с относительным позиционированием положение элемента вычисляется от нижнего края элемента (см. подраздел «Когда и где использовать относи-

тельное позиционирование» разд. «Как работают свойства позиционирования» гл. 13).

Значения: любая корректная единица измерения CSS, такая как пикселы, em или проценты. Проценты вычисляются на основании ширины содержащего элемента.

Пример: `bottom: 5em;`.

ПРИМЕЧАНИЕ

В Internet Explorer версий 6 и ниже может быть проблема при позиционировании элемента, если используется свойство bottom (см. врезку «Ошибки браузеров» в разд. «Как работают свойства позиционирования» гл. 13).

clear

Препятствует тому, чтобы элемент «обертывался» вокруг плавающего элемента. Вместо этого свободный элемент опускается вниз — под основание плавающего элемента.

Значения: `left`, `right`, `both`, `none`. Значение `left` указывает, что элемент не может «обертываться» вокруг перемещенных влево элементов. Точно так же `right` опускает элемент под любым перемещенным вправо элементом. Значение `both` предотвращает «обертывание» элемента вокруг перемещенных либо влево, либо вправо элементов. Значение `none` отключает свойство, так что используйте это значение для отмены ранее установленного свойства `clear`.

Пример: `clear: both;`.

clip

Создает прямоугольное окно, которое показывает часть элемента. Если у вас была фотография вашего выпускного класса, на которой главный хулиган стоял крайним справа, вы могли бы создать область отображения, выводящую на первый план изображение вашего мучителя. Остальная фотография останется нетронутой, а область отсечения отобразит только одного задиру. Свойство `clip` наиболее эффективно, когда используется в связке с языком JavaScript, программирующим анимацию области отсечения. Вы можете начать с маленькой области отсечения и расширять ее, пока не будет показана вся фотография.

Значения: координаты прямоугольной области. Заключите координаты в круглые скобки и поставьте перед ними ключевое слово `rect`, например: `rect(5px,110px,35px,10px);`.

Рассмотрим, как действует порядок этих координат: первое число указывает верхнее смещение — верхний край окна отсечения. В этом примере смещение равно `5px`, так что будет скрыто все, что находится в четырех первых рядах пикселов. Последнее число — левое смещение — левый край окна отсечения. В этом примере смещение равно `10px`, так что будет скрыто все слева (первые 9 пикселов элемента). Второе число определяет ширину окна отсечения плюс последнее число; если левый край отсечения составляет 10 пикселов и вы хотите, чтобы видимая область была шириной 100 пикселов, второе число должно быть `110px`. Третье число — высота

области отсечения плюс верхнее смещение (первое число). Так, в этом примере высота области отсечения равна 30 пикселам (30px + 5px = 35px).

Пример: `clip: rect(5px,110px,35px,10px);`.

СОВЕТ

Поскольку порядок координат здесь немного странный, большинству дизайнеров нравится начинать с первого и последнего аргументов, а затем рассчитывать два других.

display

Определяет вид области, используемой для отображения элемента страницы — блочного или встроенного (см. подраздел «Отображение встроенных и блочных элементов» разд. «Управление размерами полей и отступов» гл. 7). Используйте его, чтобы переопределить вариант отображения по умолчанию. Вы можете сделать так, чтобы абзац (блочный элемент) отображался без разрывов строк над и под ним — точно так же, как, скажем, ссылка (встроенный элемент).

Значения: `block`, `inline`, `none`. Свойство `display` принимает 17 значений, большинство из которых не дают никакого эффекта в доступных на сегодняшний день браузерах. Значения `block`, `inline` и `none`, однако, работают практически во всех браузерах. Значение `block` вызывает разрыв линии над и под элементом, точно так же, как и другие блочные элементы (например, абзацы и заголовки); `inline` заставляет элемент отображаться на той же линии, что и окружающие элементы (точно так же, как текст внутри тега `<strong>` появляется на той же строке, что и остальной текст); `none` заставляет элемент полностью исчезнуть со страницы. Затем вы можете вновь вызвать его, применив некоторое программирование JavaScript или псевдокласс `:hover` (см. разд. «Псевдоклассы и псевдоэлементы» гл. 3).

Еще кое-какие другие свойства работают лишь в немногих браузерах (наиболее значимыми исключениями являются Internet Explorer 7 и 6). Вы можете использовать свойства табличного отображения для создания интересных разметок страниц.

Пример: `display: block;`.

float

Перемещает элемент к левому или правому краю окна браузера или, если этот элемент находится внутри другого, к левому или правому краю содержащего элемента. Можно сказать, что свойство `float` делает элемент плавающим. Элементы, которые появляются после плавающего, передвигаются, чтобы заполнить место справа (для плавающих влево элементов) или слева (для плавающих вправо элементов), а затем обтекают плавающий элемент. Используйте «плавание» для простых эффектов, таких как перемещение изображения к какой-либо стороне страницы, или для очень сложных разметок — таких, что были описаны в гл. 12.

Значения: `left`, `right`, `none`. Значение `none` полностью отключает «плавание», что может оказаться полезным, когда у определенного тега есть стиль, к которому применено «плавание» влево или вправо, и вы хотите создать более специфический стиль, чтобы отменить «плавание» этого тега.

Пример: `float: left;`.

height

Устанавливает высоту содержимого — области элемента, в которой определены, например, текст, изображения и др. Фактическая высота элемента на экране — это общая сумма высоты, верхнего и нижнего полей, верхнего и нижнего отступов, а также верхней и нижней границ.

Значения: любая корректная единица измерения CSS, например пикселы, em или проценты. Проценты вычисляются на основании высоты содержащего элемента.

Пример: `height: 50%;`.

ПРИМЕЧАНИЕ

Иногда содержимое получается больше установленной высоты: если вы вводите много текста, например, или ваш посетитель увеличивает размер шрифта в своем браузере. Браузеры обрабатывают эту ситуацию по-разному: Internet Explorer версий 6 и ниже просто делает область больше, в то время как в других браузерах содержимое простирается за пределы области. В этом случае поведением элементов управляет свойство overflow (см. подраздел «Управление поведением блочных элементов с помощью свойства overflow» разд. «Определение параметров высоты и ширины» гл. 7).

left

При использовании с абсолютным или фиксированным позиционированием (см. разд. «Как работают свойства позиционирования» гл. 13) это свойство определяет позицию левого края стиля относительно левого края самого близкого установленного предка. Если стилизованный элемент не находится внутри каких-либо позиционированных тегов, то он будет размещаться относительно левого края окна браузера. Вы можете использовать это свойство, чтобы поместить изображение на расстоянии 20 пикселов от левого края окна браузера. При использовании с относительным позиционированием расположение элемента вычисляется от его левого края.

Значения: любая корректная единица измерения CSS, такая как пикселы, em или проценты.

Пример: `left: 5em;`.

max-height

Устанавливает максимальную высоту для элемента. Таким образом, область элемента может быть короче установленного значения, но не может быть выше. Если содержимое элемента оказывается выше, чем `max-height`, оно выходит за пределы области. Вы можете управлять этим переполнением с помощью свойства `overflow`. Internet Explorer версий 6 и ниже не понимает свойство `max-height`.

Значения: любая корректная единица измерения CSS, такая как пикселы, em или проценты. Браузеры вычисляют проценты на основании высоты содержащего элемента.

Пример: `max-height: 100px;`.

max-width

Устанавливает максимальную ширину для элемента. Область элемента может быть у́же, чем это установленное значение, но не может быть шире. Если содержимое элемента шире, чем `max-width`, оно выходит за пределы области, чем вы можете управлять с помощью свойства `overflow`. В основном свойство `max-width` применяется в свободных разметках (см. разд. «Типы разметок веб-страницы» гл. 11). Благодаря этому свойству разработчик может быть уверен — дизайн страницы на очень больших мониторах не станет таким широким, что будет непригоден для чтения. Свойство не работает в Internet Explorer версий 6 и ниже.

Значения: любая корректная единица измерения CSS, такая как пикселы, em или проценты. Проценты вычисляются на основании ширины содержащего элемента.

Пример: `max-width: 950px;`.

min-height

Устанавливает минимальную высоту для элемента. Область элемента может быть выше установленного значения, но не может быть короче. Если содержимое элемента по высоте не такое, как установлено свойством `min-height`, высота области уменьшается, чтобы соответствовать заданному значению. Internet Explorer версий 6 и ниже не распознает это свойство.

Значения: любая корректная единица измерения CSS, такая как пикселы, em или проценты. Проценты рассчитываются на основе высоты содержащего элемента.

Пример: `min-height: 20em;`.

min-width

Устанавливает минимальную ширину для элемента. Область элемента может быть шире, чем установленное значение, но не может быть у́же. Если содержимое элемента по высоте не такое, как установлено свойством, область становится такой тонкой, как указывает значение `min-width`. Вы также можете использовать это свойство в свободных разметках, чтобы дизайн не «разваливался» при меньшей ширине окна. Когда окно браузера является более узким, чем определено `min-width`, добавляются горизонтальные полосы прокрутки. Internet Explorer версий 6 и ниже не поддерживает это свойство.

Значения: любая корректная единица измерения CSS, такая как пикселы, em или проценты. Проценты рассчитываются на основе ширины содержащего элемента.

Пример: `min-width: 760px;`.

ПРИМЕЧАНИЕ

Свойства max-width и min-width обычно применяются в связке при создании свободных разметок (см. врезку «Информация для опытных пользователей» в подразделе «Использование отрицательных полей для размещения элементов» разд. «Использование плавающих элементов в разметках» гл. 12).

overflow

Определяет, что должно случиться с элементом, который выходит за пределы своей области для содержимого, например, с фотографией, оказавшейся шире, чем установлено свойством `width`.

ПРИМЕЧАНИЕ

Internet Explorer версий 6 и ниже обрабатывает ситуации с переполнением иначе, чем другие браузеры (см. подраздел «Управление поведением блочных элементов с помощью свойства overflow» разд. «Определение параметров высоты и ширины» гл. 7).

Значения: `visible`, `hidden`, `scroll`, `auto`. Значение `visible` вынуждает излишнее содержимое простираться за пределы области, накладывая его на границы и другие элементы страницы. Internet Explorer версий 6 и ниже просто увеличивает область, чтобы вставить более крупный объект. Значение `hidden` скрывает любое содержимое за пределами отведенной ему области; `scroll` добавляет полосы прокрутки к элементу, в результате чего посетитель может выполнить прокрутку, чтобы увидеть любой объект за пределами содержащей его области, — нечто вроде минифрейма. Значение `auto` добавляет полосы прокрутки, только когда они необходимы, чтобы показать больше содержимого.

Пример: `overflow: hidden;`.

position

Определяет, какой тип позиционирования использует браузер при размещении элементов на странице.

Значения: `static`, `relative`, `absolute`, `fixed`. Значение `static` определяет обычный режим браузера — один блочный элемент расположен на вершине следующего, а содержимое простирается от верхнего до нижнего края экрана; `relative` размещает элемент относительно того места, где он в настоящее время появляется на странице. Другими словами, установка этого свойства может сместить элемент с его текущей позиции. Значение `absolute` полностью убирает элемент из потока страницы. Другие элементы «не видят» абсолютный элемент и могут появиться под ним. Это значение используется для установки элемента в конкретное место на странице или для его размещения в определенном месте относительно родительского элемента, который установлен с абсолютным, относительным или фиксированным позиционированием. Значение `fixed` фиксирует элемент на странице, поэтому, когда она прокручивается, фиксированный элемент остается на экране, что очень подобно фреймам HTML. Internet Explorer версий 6 и ниже игнорирует значение `fixed`.

Пример: `position: absolute;`.

ПРИМЕЧАНИЕ

Чаще всего применяются значения relative, absolute и fixed вместе с left, right, top и bottom. Все подробности о позиционировании вы можете узнать из гл. 13.

right

При использовании с абсолютным или фиксированным позиционированием это свойство определяет положение правого края стиля относительно правого края его самого близкого установленного предка. Если элемент, для которого разрабатывается стиль, не находится внутри каких-либо позиционированных тегов, то он будет размещаться относительно правого края окна браузера. Вы можете использовать это свойство, чтобы поместить боковое меню на определенном расстоянии от правого края окна браузера. При использовании свойства с относительным позиционированием положение элемента вычисляется от его правого края.

Значения: любая корректная единица измерения CSS, такая как пикселы, em или проценты.

Пример: `right: 5em;`.

ПРИМЕЧАНИЕ

В Internet Explorer версий 6 и ниже могут быть проблемы при позиционировании элементов с использованием свойства right (см. врезку «Ошибки браузеров» разд. «Как работают свойства позиционирования» гл. 13).

top

Действие этого свойства противоположно действию свойства `bottom`, описанного выше. Другими словами, при использовании с абсолютным или фиксированным позиционированием это свойство определяет позицию верхнего края стиля относительно верхнего края его самого близкого установленного предка. Если стилизованный элемент не находится внутри каких-либо позиционированных тегов, то он будет размещаться относительно верхнего края окна браузера. Вы можете использовать это свойство, чтобы поместить логотип на установленное расстояние от верхнего края окна браузера. При использовании свойства с относительным позиционированием размещение элемента рассчитывается от его верхнего края.

Значения: любая корректная единица измерения CSS, такая как пикселы, em или проценты.

Пример: `top: 5em;`.

visibility

Определяет, отображает ли браузер элемент. Используйте это свойство, чтобы скрыть некоторые объекты страницы, например абзац, заголовок или содержимое тега `<div>`. В отличие от значения `none` свойства `display`, установка которого скрывает элемент и удаляет его из потока страницы, значение `hidden` свойства `visibility` позволяет не удалять элемент из потока страницы. Вместо этого остается пустое пространство в том месте, где был бы элемент. По этой причине свойство `visibility` чаще применяется с абсолютно позиционированными элементами, которые уже были удалены из потока страницы.

Скрытие элемента не принесет большой пользы, если вы не сможете показать его снова. Программирование JavaScript — самый распространенный способ пере-

ключать свойство visibility для показа и скрытия элементов на странице. Вы можете также использовать псевдокласс :hover (см. разд. «Псевдоклассы и псевдоэлементы» гл. 3), чтобы изменить свойство visibility элемента, когда посетитель наводит указатель мыши на некоторые части страницы.

Значения: visible, hidden. Кроме того, можно использовать значение collapse, чтобы скрыть строку или столбец в таблице.

Пример: visibility: hidden;.

width

Устанавливает ширину области элемента, которая содержит текст, изображения и др. Количество места на экране, фактически отведенного для элемента, может быть намного больше, так как включает ширину левого и правого полей, левого и правого отступов, а также левой и правой границ.

Значения: любая корректная единица измерения CSS, такая как пикселы, em или проценты. Проценты рассчитываются в зависимости от ширины содержащего элемента.

Пример: width: 250px;.

z-index

Управляет порядком расположения позиционированных элементов. Относится только к элементам, у которых для свойства position установлено значение absolute, relative или fixed. Свойство определяет, где появляется элемент относительно оси *Z*. Если два абсолютно позиционированных элемента накладываются друг на друга, тот, у которого более высокий индекс позиционирования, окажется сверху.

Значения: целочисленные значения, такие как 1, 2 или 10. Вы также можете использовать отрицательные значения, но различные браузеры обрабатывают их по-разному. Чем больше число, тем «выше» расположен элемент. Элемент с z-index, равным 20, появится ниже элемента с z-index, равным 100 (если эти два элемента накладываются) (см. рис. 13.6).

Пример: z-index: 12;.

СОВЕТ

Значения не должны задаваться в строгом порядке. Если для элемента A установлено значение 1 свойства z-index, вам не обязательно устанавливать значение 2 для элемента B, чтобы поместить его сверху. Вы можете использовать 5, 10 и т. д., чтобы получить тот же самый результат, главное, чтобы число было больше. Так, чтобы убедиться в том, что элемент всегда будет появляться над другими элементами, просто задайте ему очень большое значение, например 10 000. Но помните, что максимальное значение, которое может обрабатывать Firefox, равняется 2 147 483 647, поэтому не делайте z-index больше этого числа.

Свойства таблицы

Существует несколько свойств CSS, которые относятся исключительно к таблицам HTML. В гл. 10 можно найти детальные инструкции по использованию CSS с таблицами.

border-collapse

Определяет, расширены границы вокруг ячеек таблицы или сжаты. Когда они расширены, браузеры добавляют пространство размером несколько пикселов между каждой ячейкой. Даже если вы уберете это пространство, установив значение 0 атрибута cellspacing HTML-тега <table>, браузеры все еще будут отображать двойные границы. Таким образом, нижняя граница одной ячейки появится над верхней границей другой ячейки, расположенной ниже, что вызовет удвоение линий границ. Установка значения collapse свойства border-collapse устраняет и область между ячейками, и это удвоение границ (см. разд. «Создание стилей для таблиц» гл. 10). Свойство работает, только когда относится к тегу <table>.

Значения: collapse, separate.

Пример: border-collapse: collapse;.

border-spacing

Устанавливает расстояние между ячейками в таблице. Оно заменяет HTML-атрибут cellspacing тега <table>. Однако Internet Explorer 7 и ниже не понимает свойство border-spacing, так что лучше продолжать использовать атрибут cellspacing в тегах <table>, чтобы гарантировать пространство между ячейками во всех браузерах.

ПРИМЕЧАНИЕ

Если вы хотите удалить пространство, которое браузеры обычно вставляют между ячейками, просто установите значение collapse свойства border-collapse.

Значения: два CSS-значения длины. Первое устанавливает горизонтальное разделение (пространство с обеих сторон каждой ячейки), а второе — вертикальное (пространство, отделяющее основание одной ячейки от вершины другой ячейки, находящейся под первой).

Пример: border-spacing: 0 10px;.

caption-side

Когда свойство относится к заголовку таблицы, оно определяет, появится заголовок вверху или внизу таблицы. Поскольку, согласно правилам HTML, тег <caption> должен идти сразу за открывающим тегом <table>, заголовок обычно появляется вверху таблицы.

Значения: top, bottom.

Пример: caption-side: bottom;.

ПРИМЕЧАНИЕ

К сожалению, это свойство не дает никакого эффекта в браузерах Internet Explorer 6 или 7 (в восьмой версии оно работает), так что безопаснее придерживаться эквивалентного HTML-кода: <caption align= "bottom"> или <caption align = "top">.

empty-cells

Определяет, как браузер должен отобразить ячейку таблицы, которая совершенно пуста. В HTML это выглядело бы следующим образом: `<td></td>`. Значение `hide` скрывает любую часть ячейки, вставляя пустое пространство, так что границы, фоновые цвета и изображения не показываются в пустой ячейке. Примените это свойство к стилю, форматирующему тег `<table>`.

Значения: `show`, `hide`.

Пример: `empty-cells: show;`.

ПРИМЕЧАНИЕ

Свойство empty-cells не дает никакого эффекта в Internet Explorer 7 и более ранних версиях.

table-layout

Управляет тем, как браузер чертит таблицу, и может немного влиять на скорость отображения страницы браузером. Установка значения `fixed` вынуждает браузер привести все столбцы к той же ширине, что задана для столбцов из первой строки, из-за чего таблица чертится быстрее. Значение `auto` — стандартное, при котором «браузер делает свое дело», поэтому, если вы довольны тем, как быстро ваши таблицы появляются на странице, не беспокойтесь об этом свойстве. Если же вы используете его, то применяйте к стилю, форматирующему тег `<table>`.

Значения: `auto`, `fixed`.

Пример: `table-layout: fixed;`.

Различные свойства

CSS 2.1 предлагает некоторые дополнительные и иногда интересные свойства. Они позволяют улучшать веб-страницы, задавая специальное содержимое, различные указатели мыши, предлагают больше управления печатью страницы и т. д. К сожалению, не все браузеры одинаково правильно понимают эти свойства.

content

Определяет текст, который появляется либо до, либо после элемента. Используйте это свойство с псевдоэлементами `:after` или `:before` (см. врезку «Информация для опытных пользователей» в подразделе «Стилизация фонов для печати» разд. «Создание таблиц стилей для печати» гл. 14). Вы можете добавить открывающую кавычку перед цитируемым материалом и закрывающую кавычку после него. Это свойство не поддерживается в Internet Explorer 6 и 7, так что его использование ограниченно.

Значения: текст в кавычках (`"как этот"`), ключевые слова `normal`, `open-quote`, `close-quote`, `no-open-quote`, `no-close-quote`.

Примеры: `p.advert:before {content: "And now a word from our sponsor…";}` и `a:after {content: "("attr(href)")";}`.

ПРИМЕЧАНИЕ

Добавление текста таким способом (как пример с открывающими и закрывающими кавычками) называют генерируемым содержимым. Прочитайте простое объяснение этого феномена на сайте www.westciv.com/style_master/academy/css_tutorial/advanced/generated_content.html.

cursor

Позволяет изменять вид указателя мыши, когда он передвигается над определенным элементом. Например, вы можете задать ему вид вопросительного знака, когда кто-то проводит указателем мыши над ссылкой, предоставляющей дополнительную информацию по какой-либо теме.

Значения: `auto`, `default`, `crosshair`, `pointer`, `move`, `e-resize`, `ne-resize`, `nw-resize`, `nresize`, `se-resize`, `sw-resize`, `s-resize`, `w-resize`, `text`, `wait`, `help`, `progress`. Вы также можете применять URL-значение, чтобы использовать для указателя собственное изображение (но прочтите примечание ниже). Указатель, который находится над ссылкой, выглядит как стрелка, так что, если хотите, чтобы какие-то элементы на странице показывали пользователю, что он может щелкнуть на них, добавьте к стилю объявление `cursor: pointer`.

Пример: `cursor: help; cursor: url(images/cursor.cur);`.

ПРИМЕЧАНИЕ

Не все браузеры распознают значения URL для указателя. Для получения более полной информации зайдите на сайт www.quirksmode.org/css/cursor.html.

orphans

Определяет минимальное количество строк текста, которые можно оставить внизу распечатанной страницы. Предположим, вы печатаете страницу на лазерном принтере и пятистрочный абзац приходится на две страницы, причем всего одна строка находится внизу первой страницы, а четыре оставшиеся — на второй. Поскольку одна строка выглядит «висячей», вы можете указать браузеру разбить абзац, только если, скажем, по крайней мере три строки оставлены внизу распечатанной страницы (на момент написания книги только браузер Opera понимает это свойство).

Значения: числа, например 1, 2, 3 или 5.

Пример: `orphans: 3;`.

page-break-after

Определяет, происходит ли разрыв страницы при печати после определенного элемента. С ним вы можете убедиться в том, что конкретный абзац всегда будет последним элементом, который появится на печатной странице.

Значения: auto, always, avoid, left, right. Значение auto — стандартное, оно позволяет браузеру определять, когда и как разбивать содержимое на печатные страницы; always вынуждает элемент, который следует далее, появиться вверху отдельной печатной страницы, и это единственное значение, которое работает одинаково во всех браузерах; avoid предотвращает обрыв страницы после элемента; это отличный способ прикрепить заголовок к абзацу, который идет за ним, но, к сожалению, большинство браузеров не распознают его. Значения left и right определяют, появляется ли следующий элемент на лево- или правосторонней странице, что может вынудить браузер напечатать дополнительную пустую страницу. Но, поскольку никакие браузеры не понимают этих значений, не волнуйтесь о напрасной трате бумаги. Браузеры обрабатывают значения left и right так же, как и always.

Пример: `page-break-after: always;`.

page-break-before

Работает так же, как и page-break-after, за исключением того, что разрыв страницы происходит перед стилизованным элементом. При этом элемент помещается на вершину следующей печатной страницы. Вы можете использовать это свойство, чтобы убедиться в том, что каждый заголовок для различных разделов длинной веб-страницы появится вверху страницы.

Значения: те же самые, что и для page-break-after.

Пример: `page-break-before: always;`.

page-break-inside

Препятствует тому, чтобы элемент был разбит на две печатные страницы. Если вы хотите держать фотографию и ее заголовок вместе на отдельной странице, укажите для них обоих один тег , а затем примените к нему стиль со свойством page-break-inside (на момент написания книги только Opera понимает это свойство).

Значения: avoid.

Пример: `page-break-inside: avoid;`.

widows

Противоположное свойству orphans, описанному выше. Оно определяет минимальное количество строк, которое должно появиться наверху печатной страницы. Скажем, принтер может поместить четыре из пяти строк абзаца внизу страницы и должен будет переместить последнюю строку на вершину следующей страницы. Чтобы предотвратить появление таких «висячих» строк, используйте свойство widows, заставляющее браузер переместить по крайней мере две или три строки вместе на вершину печатной страницы (только Opera понимает это свойство, поэтому оно ограничено в использовании).

Значения: числа, например, 1, 2, 3 или 5.

Пример: `widows: 3;`.

2 CSS в Dreamweaver CS4

Dreamweaver CS4 от Adobe — программа для создания сайтов, которая берет на себя основную работу по формированию HTML/XHTML- и CSS-кода. Вместо того чтобы печатать строки кода в текстовом редакторе, вы можете нажимать удобные кнопки и меню на экране и смотреть, как дизайн постепенно рождается перед вашими глазами. У программы даже есть мощные инструменты управления сайтом, которые помогают вам следить за страницами и ссылками сайта.

Хотя эта книга дает вам все, что вы должны знать для создания вашего собственного CSS-кода с нуля, нет ничего плохого в том, чтобы воспользоваться визуальным редактором, таким как Dreamweaver, для экономии времени. На самом деле знание того, как работает CSS, о чем и рассказано в этой книге, имеет огромное значение при настройке или устранении неисправностей в страницах, созданных в Dreamweaver.

ПРИМЕЧАНИЕ

Это приложение описывает исключительно особенности CSS в Dreamweaver CS4. Чтобы изучить все возможности Dreamweaver для проектирования и сопровождения вашего сайта, прочтите книгу Dreamweaver CS4: The Missing Manual (издательство O'Reilly).

Создание стилей

Работу над большинством связанных с CSS задач нужно начинать на панели, которая является в Dreamweaver центром команд для создания стилей. Чтобы открыть ее, выберите меню **Window ▸ CSS Styles** (Окно ▸ Стили CSS) (или нажмите сочетание клавиш **Shift+F11**).

Используйте окно на рис. П2.1 в качестве ориентира на панели **CSS Styles** (Стили CSS).

Кнопка **All** (Везде) вверху панели позволяет открыть список всех внутренних и внешних стилей для текущего документа. Другая кнопка — **Current** (Текущий) — дает возможность ближе взглянуть на отдельные стили.

СОВЕТ

Щелчок кнопкой мыши на значке «минус» (–) слева от таблицы стилей сворачивает список стилей, скрывая их (в Mac кнопка выглядит как небольшой треугольник, но выполняет то же самое).

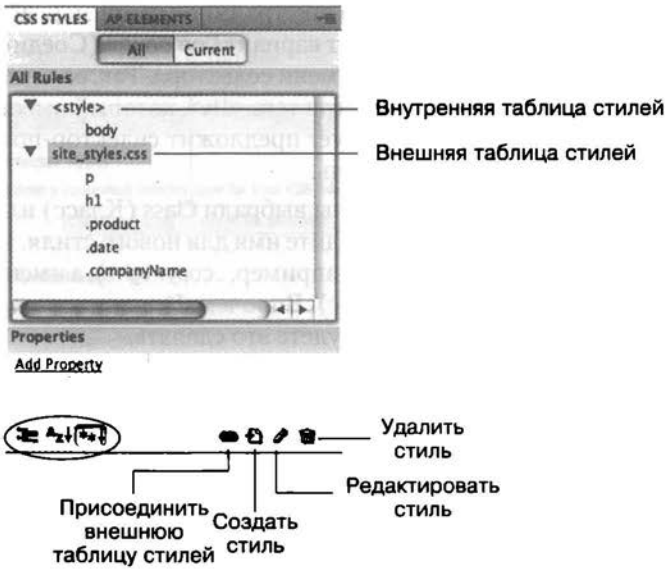


Рис. П2.1. Когда нажата кнопка All (Везде), панель CSS Styles (Стили CSS) отображает список всех доступных на текущей странице стилей, включая внешние и внутренние таблицы стилей

Внутренняя таблица стилей обозначается на панели посредством тега `<style>`. В примере на рис. П2.1 есть лишь один тег стиля во внутренней таблице стилей (для тега `<body>`).

Внешние таблицы стилей перечислены посредством имени файла (`main.css`).

Правила внешних таблиц стилей перечислены под именем файла (`p`, `h1`, `.copyright` и т. д.). Первые два стиля — теги стилей (заметьте, что имена соответствуют различным HTML-тегам), в то время как пять последних — классы стилей (обратите внимание на точку в начале каждого имени).

Список **Properties** (Свойства) в нижней половине панели позволяет редактировать стиль. Три кнопки в нижнем левом углу панели (обведены на рис. П2.1) управляют тем, как отображается список свойств.

Этап 1. Установка типа CSS

Dreamweaver предлагает несколько способов создания нового стиля: нажмите кнопку нового стиля на панели **CSS Styles** (Стили CSS) (см. рис. П2.1); щелкните правой кнопкой мыши в любом месте указанной панели, а затем выберите пункт **New** (Создать) в контекстном меню (либо выберите **Format** ▸ **CSS Styles** ▸ **New** (Формат ▸ Стили CSS ▸ Создать)). Появится окно **New CSS Rule** (Создать правило CSS) (рис. П2.2), где вы начнете создавать свой новый стиль. В этом окне есть следующие элементы.

- **Selector Type** (Тип селектора). В этом списке выберите тип стиля, который создаете: **Class** (Класс), **ID** (Идентификатор) или **Tag** (Тег).

  Используйте четвертый тип — **Compound** (Соединение) — для создания более сложных типов стилей, таких как псевдоклассы, селекторы атрибутов и наследуемые селекторы.

Кроме того, если вы выбрали что-то на странице (например, абзац, картинку или заголовок), Dreamweaver выделит вариант **Compound** (Соединение) и предложит наследуемый селектор в поле имени селектора. Так, если щелкнуть кнопкой мыши в абзаце, находящемся внутри тега `<div>`, который имеет идентификатор `main`, то в этом случае Dreamweaver предложит селектор-потомок `#main p`, когда вы будете создавать новый стиль.

- **Selector Name** (Имя селектора). Если вы выбрали **Class** (Класс) или **ID** (Идентификатор) в списке типа селектора, введите имя для нового стиля. Имена классов стилей должны начинаться с точки (например, `.copyright`), а имена стилей ID — с символа решетки (например, `#banner`). Впрочем, Dreamweaver автоматически добавит нужный символ, если вы забудете это сделать.

  Если же вы выбрали значение **Tag** (Тег), то укажите HTML-тег, который хотите переопределить, в раскрывающемся списке **Tag** (Тег) (который появляется в данном случае).

СОВЕТ

Если вы опытный человек в HTML, то просто введите название тега без скобок в поле Name (Имя), минуя раскрывающийся список Tag (Тег). Например, когда вы хотите создать стиль для всех маркированных списков, наберите ul.

Если вы выбрали значение **Compound** (Соединение), Dreamweaver позволит вам набрать любой правильный тип CSS-селектора в поле **Selector** (Селектор). Используйте эту возможность для создания наследуемых селекторов, селекторов атрибутов и других составных селекторов, а также для создания тега или класса стиля.

При добавлении класса, идентификатора, тега или другого селектора в поле имени селектора (**Selector Name**) Dreamweaver резюмирует, к каким элементам HTML будет применяться этот селектор. Например, на рис. П2.2 показано окно **New CSS Rule** (Создать правило CSS) в процессе создания нового стилевого класса под названием `copyright`. В окне объясняется, что это правило будет применяться ко всем тегам HTML, для свойства `class` которых установлено значение `copyright` (другими словами, ко всем тегам, к которым был применен класс `copyright`). Для простых стилей, таких как тег или класс стилей, это объяснение вряд ли будет полезно, но для наследуемых и других составных селекторов окно с объяснением помогает установить, к какому элементу страницы применяется селектор.

- **Rule Definition** (Определение правила). Этот список внизу окна **New CSS Rule** (Создать правило CSS) позволяет указать, где будет храниться код CSS, который вы собираетесь создавать. Выберите значение **This document only** (Только в этом документе), чтобы добавить внутреннюю таблицу стилей. Чтобы создать новую внешнюю таблицу стилей, выберите значение **New Style Sheet File** (Создать файл таблицы стилей). Это позволит не только создать новый внешний CSS-файл (который вы можете сохранить где-нибудь в папке вашего сайта), но и добавить необходимый код в текущий документ, чтобы связать его с этим файлом. Если вы ранее связывали документ с внешней таблицей стилей, то название этой таблицы появится в данном раскрывающемся списке, указывая на то, что Dreamweaver будет хранить в ней новый стиль.

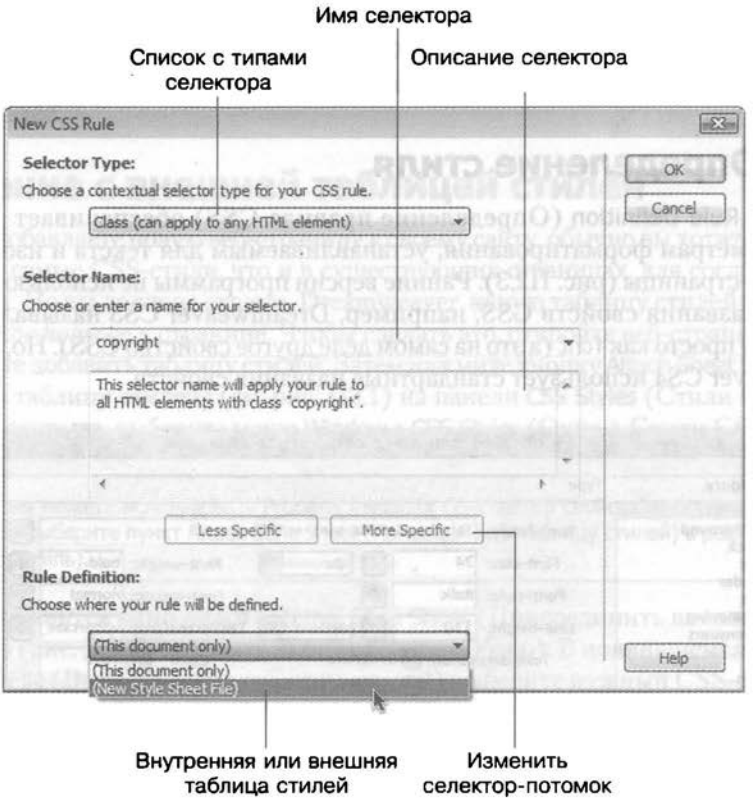


Рис. П2.2. В окне New CSS Rule (Создать правило CSS) вы выбираете тип стиля, задаете ему имя и решаете, где размещать — во внутренней или во внешней таблице стилей. Если вы уже знакомы с предыдущими версиями Dreamweaver, то заметите, что в новой версии окно существенно изменилось

СОВЕТ

Если вы создаете связку внутренних стилей для конкретной страницы, а позже понимаете, что хотели бы превратить их во внешнюю таблицу стилей, которую сможете использовать в других страницах, то удача на вашей стороне. В Dreamweaver есть множество инструментов для управления таблицами стилей. Совсем скоро вы узнаете о том, как использовать их.

Если вы указали, что хотите создать внешнюю таблицу стилей, нажатие OK приведет к появлению окна **Save Style Sheet As** (Сохранить файл таблицы как). Перейдите к папке со своим сайтом и введите название нового внешнего CSS-файла (не забывайте о расширении CSS в конце).

СОВЕТ

Если вы будете использовать эту таблицу стилей для всех страниц сайта, то, возможно, захотите сохранить ее в корневой папке сайта или в папке, специально предназначенной для таблиц стилей. Присвойте ей общее название, такое как site_styles.css или main.css (здесь не нужно набирать расширение имени файла — Dreamweaver добавит его автоматически).

Независимо от того, в какое положение вы установили переключатель Define in (Определить в), нажатие кнопки OK в конечном счете приведет вас к окну CSS Rule Definition (Определение правила CSS).

Этап 2. Определение стиля

Окно CSS Rule Definition (Определение правила CSS) обеспечивает доступ ко всем параметрам форматирования, устанавливаемым для текста и изображений вашей веб-страницы (рис. П2.3). Ранние версии программы не использовали стандартные названия свойств CSS, например, Dreamweaver CS3 называл свойство `font-family` просто как `font` (а это на самом деле другое свойство CSS). Но, к счастью, Dreamweaver CS4 использует стандартные названия свойств.

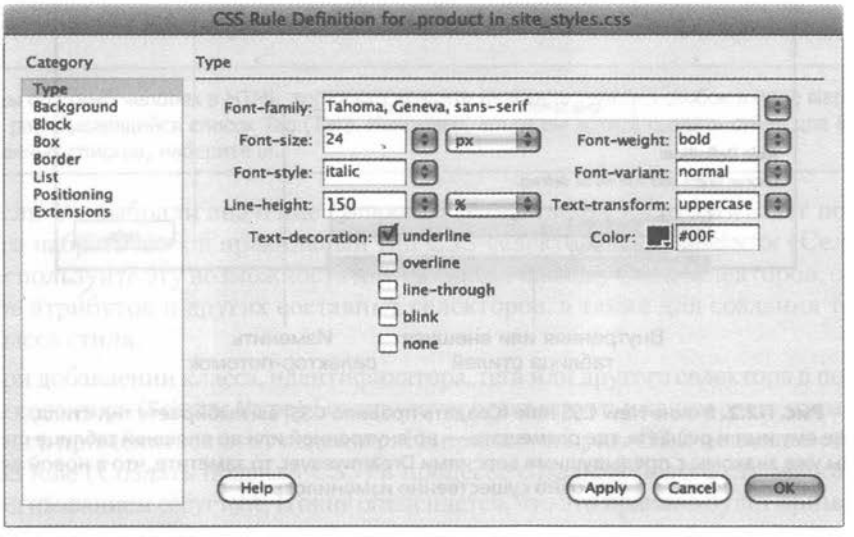


Рис. П2.3. Для максимального управления форматированием Dreamweaver позволяет устанавливать множество различных свойств вложенных таблиц стилей из окна CSS Rule Definition (Определение правила CSS)

Как только вы определили стиль, нажмите кнопку OK. Dreamweaver добавляет стиль к указанной таблице стилей и отображает его на панели CSS Styles (Стили CSS).

Настоящая сложность при создании стиля заключается в освоении всех доступных свойств, определяющих, например, границы, поля, фоновые цвета, после чего нужно установить, какие из них надежно работают в различных браузерах.

Добавление стилей к веб-страницам

Создав один раз стили, вы можете легко применять их в дальнейшем. На самом деле, если вы создали HTML-теги стилей, вам не нужно делать что-либо еще, чтобы применить стили, потому что их селекторы автоматически указывают, на какие

теги они воздействуют. Когда вы помещаете стили во внешнюю таблицу стилей, Dreamweaver автоматически связывает их с текущим документом. Чтобы использовать таблицу стилей в *другой* веб-странице, вы должны *присоединить* ее, как описано далее.

Соединение с внешней таблицей стилей

Когда вы добавляете новую веб-страницу к своему сайту, обычно вы хотите использовать те же самые CSS-стили, что и в существующих страницах, для согласованного просмотра. Но вы должны сказать Dreamweaver, какую таблицу стилей вы используете, *присоединив* ее к странице. Чтобы сделать это, откройте веб-страницу, к которой желаете добавить таблицу стилей. Затем нажмите кнопку **Attach Sheet Sheet** (Присоединить таблицу стилей) (см. рис. П2.1) на панели **CSS Styles** (Стили CSS) (если панель не открыта, выберите меню **Window** ▸ **CSS Styles** (Окно ▸ Стили CSS)).

СОВЕТ

Кроме того, вы можете использовать Property inspector (Инспектор свойств) в Dreamweaver (внизу окна). Просто выберите пункт Attach Style Sheet (Присоединить таблицу стилей) в раскрывающемся списке Style (Стиль).

Когда появится окно **Attach External Style Sheet** (Присоединить внешнюю таблицу стилей) (рис. П2.4), нажмите кнопку **Browse** (Обзор). В появившемся окне **Select Style Sheet File** (Выбрать файл таблицы стилей) выберите нужный CSS-файл и дважды щелкните на нем кнопкой мыши. Если Dreamweaver предлагает скопировать файл таблицы стилей в корневую папку вашего сайта, нажмите кнопку **Yes** (Да).

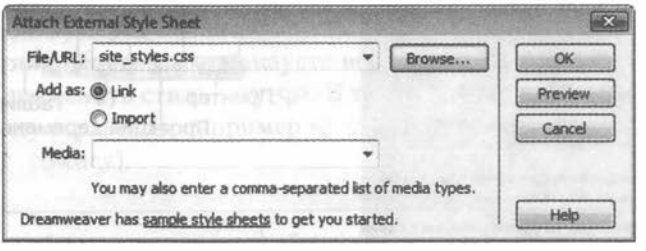


Рис. П2.4. Большинство значений в списке Media (Мультимедиа) не особенно полезны, так как нет устройств, запрограммированных для работы с ними. Выбрав же print (печать) или screen (экран), вы легко сможете управлять тем, как отображается страница при просмотре на мониторе и при печати на принтере

Окно **Attach External Style Sheet** (Присоединить внешнюю таблицу стилей) позволяет выбрать, как присоединить таблицу стилей и к каким типам устройств применить стили.

- При присоединении внешней таблицы стилей вы можете установить переключатель **Add as** (Добавить как) в положение **Link** (Ссылка) или **Import** (Импорт). Эти две настройки практически идентичны, как описано в гл. 2.
- В раскрывающемся списке **Media** (Мультимедиа) вы определяете, какой тип устройства должна использовать таблица стилей. Выбор значения **print** (печать) означает, что таблица стилей будет применена только тогда, когда документ

печатается. Большинство этих пунктов, таких как tv (телевизор) или tty (терминал), обычно не используются веб-дизайнерами. Вы можете прочитать о различных типах устройств в разд. «Как работают аппаратно-зависимые таблицы стилей» гл. 14. В большинстве случаев вы можете спокойно проигнорировать этот раскрывающийся список.

Значение all (везде) в списке Media (Мультимедиа) означает, что таблица стилей применяется при печати, при просмотре на мониторе, воспринимается Брайль-диктором (для слепых) и т. д.

В Dreamweaver CS4 также есть полезная панель инструментов для управления отображением таблиц стилей, нацеленных на различные типы устройств (рис. П2.5). Чтобы подключить панель инструментов Style Rendering (Отображение стиля) в Dreamweaver (см. рис. П2.5, *вверху справа*), выберите меню View ▸ Toolbars ▸ Style Rendering (Просмотр ▸ Панели инструментов ▸ Отображение стиля). Нажмите кнопку на панели инструментов, чтобы показать стили, соответствующие типам устройств, которые вы выбрали при присоединении страницы. Типы screen (экран) (см. рис. П2.5, *вверху слева*) и print (печать) (см. рис. П2.5, *внизу слева*) — два самых полезных, хотя вы можете скрыть все CSS-стили, нажав кнопку None (Переключить) в правой части панели инструментов (см. рис. П2.5, *внизу справа*).



Рис. П2.5. Примеры работы панели инструментов Style Rendering (Отображение стиля) в Dreamweaver

СОВЕТ

Вы можете предварительно посмотреть воздействие таблицы стилей на вашу страницу, нажав кнопку Preview (Просмотр) в окне Attach External Style Sheet (Присоединить внешнюю таблицу стилей).

После выбора требуемых параметров нажмите кнопку OK. Dreamweaver добавит необходимый HTML-код веб-страницы и автоматически отформатирует любые теги в документе согласно HTML-тегам стилей в таблице стилей. Вы увидите, как моментально изменяется форматирование в окне документа после присоединения внешней таблицы стилей.

Если таблица стилей содержит классы стилей, то вы не увидите результата их форматирования, пока не примените их к элементу на странице, как описано далее.

Применение класса стиля

Вы можете применить любой класс стиля к любому элементу, будь это слово, изображение или целый абзац текста, хотя это не всегда имеет смысл.

Применение класса стиля к тексту

Начните с нескольких слов. Затем выберите имя стиля в Property inspector (Инспектор свойств) — это вы можете сделать как в режиме HTML (имя при этом выбирается в списке **Class** (Класс) (рис. П2.6, *вверху*)), так и в режиме CSS (здесь вы используете список **Targeted Rule** (Целевое правило) (см. рис. П2.6, *внизу*)).

Чтобы отформатировать целый абзац, трижды щелкните кнопкой мыши внутри его (или заголовка) перед использованием Property inspector (Инспектор свойств) для выбора стиля. Когда вы стилизуете весь абзац, вы на самом деле говорите Dreamweaver применить стиль тегу <p>. В таком случае программа добавит свойство `class` к коду страницы, например `<p class="company">` (для стилевого класса под названием `.company`).

СОВЕТ

Вы также можете добавить класс ко всему абзацу или заголовку, просто щелкнув кнопкой мыши в любом месте внутри абзаца и выбрав имя класса в Property inspector (Инспектор свойств). Только убедитесь при этом, что не выбрали текст, иначе стиль будет применен к этому выбранному тексту, а не ко всему абзацу.

С другой стороны, если вы примените класс к выбранному элементу, не являющемуся тегом, например к слову, по которому вы дважды щелкнули, то Dreamweaver задаст для него тег таким образом: `<span class="company">Chia Vet</span>`. Этот тег применяет стиль для промежутка текста, который не идентифицируется в качестве отдельного тега.

Применение класса стиля к объектам

Чтобы применить класс стиля к объекту, например к изображению или таблице, для начала выберите его (селектор тегов внизу окна документа — отличный способ

выбрать тег). Далее используйте список **Class** (Класс) в **Property inspector** (Инспектор свойств) (см. рис. П2.6, *внизу*), чтобы выбрать название стиля.

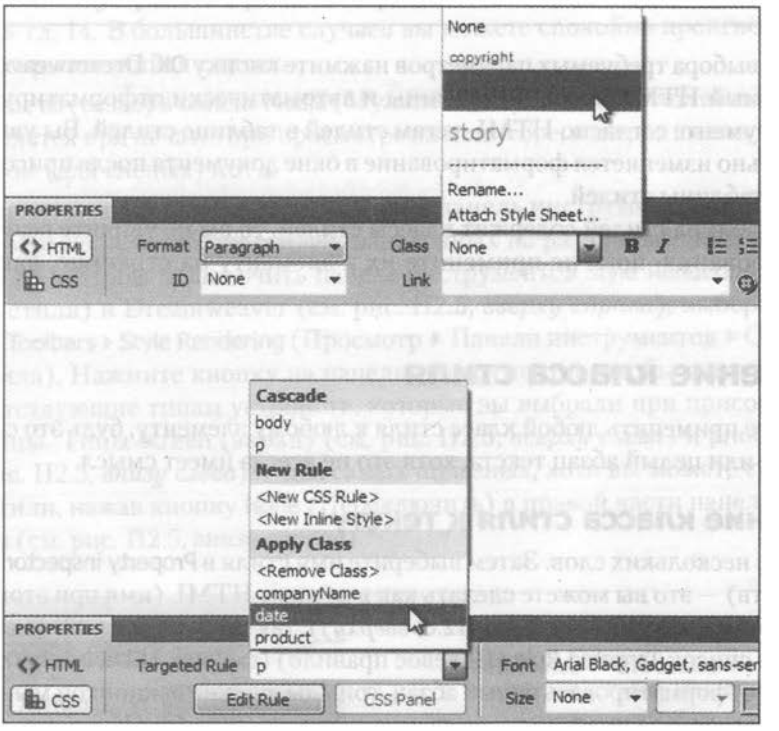


Рис. П2.6. Property inspector (Инспектор свойств) — самый легкий способ применить класс стиля

ПРИМЕЧАНИЕ

Вы можете применить любой класс стиля для любого элемента, однако это не всегда имеет смысл. Если вы задаете для изображения стиль, определяющий полужирный шрифт Courier красного цвета, оно не станет выглядеть по-другому.

Другие применения класса стиля

Вы можете применить класс стиля для любого элемента, который желаете разработать, выбирая меню **Text** ▸ **CSS Styles** (Текст ▸ Стили CSS), а затем определяя стиль в подменю. Или щелкните правой кнопкой мыши на названии стиля на панели **CSS Styles** (Стили CSS), а затем выберите **Apply** (Применить) в контекстном меню. Наконец, вы также можете выбрать класс из селектора тегов окна документа, как показано на рис. П2.7. Просто щелкните правой кнопкой мыши на названии тега, который желаете отформатировать, а затем выберите класс стиля в подменю **Set Class** (Установить класс). Селектор тегов также сообщает, относится ли к тегу класс стиля. Если это так, вы увидите название стиля в конце тега. На рис. П2.7 класс стиля, названный `.products`, применен к маркированному списку (к тегу `<ul>`).

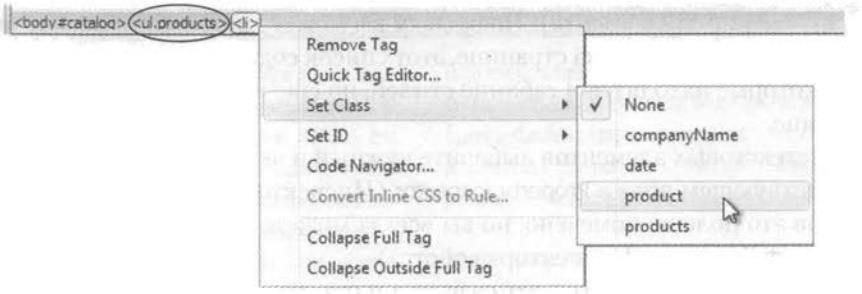


Рис. П2.7. Вы можете применить стиль класса непосредственно к тегу, используя селектор тегов окна документа

Отмена класса стиля

Чтобы убрать стиль из текста на веб-странице, просто укажите его, а затем выберите **None** (Нет) в списке **Class** (Класс) в режиме HTML (или выберите **Remove Class** (Удалить класс) в списке **Targeted Rule** (Целевое правило) в режиме CSS) в **Property inspector** (Инспектор свойств) (см. рис. П2.6). Чтобы убрать класс стиля из другого объекта, например изображения, укажите этот объект, а затем выберите значение **None** (Нет) в списке **Class** (Класс) в **Property inspector** (Инспектор свойств). Вы также можете выполнить команду **Format ▸ CSS Styles ▸ None**, чтобы убрать стиль из любой выборки (даже нетекстовых элементов, таких как изображения и таблицы).

СОВЕТ

Если вы применили класс стиля к отдельным фрагментам текста, вам не нужно фактически выделять весь текст, чтобы убрать стиль. Просто щелкните кнопкой мыши где-нибудь внутри его и выберите пункт None (Нет) списка Class (Класс) в Property inspector (Инспектор свойств) (или выберите Remove Class (Удалить класс) в списке Targeted Rule (Целевое правило)). Dreamweaver способен сообразить, что вы хотите убрать стиль, который был применен к тексту. Если вы применяли стиль к тегу, Dreamweaver удалит свойство class. Если вы применяли стиль, используя тег , то Dreamweaver удалит этот тег.

Вы не можете тем не менее удалить теги стилей из HTML-тегов. Предположим, вы переопределили тег <h2>. Если у вашей страницы есть три заголовка второго уровня (<h2>) и вы хотите, чтобы у третьего заголовка был отличный от других стиль, то вы **не** можете просто «удалить» стиль <h2> **из** третьего заголовка. Вместо этого вам нужно создать новый класс стиля со всеми желаемыми параметрами форматирования для этого заголовка и применить его непосредственно к тегу <h2> (благодаря магии CSS параметры форматирования класса переопределяют любые существующие параметры тега стиля (см. разд. «Особенности **механизма** каскадности: какие стили имеют преимущество» гл. 5)).

Применение идентификаторов к тегу

Чтобы применить идентификатор к тексту, просто выделите его, а затем используйте список **ID** (Идентификатор) в HTML-режиме **Property inspector** (Инспектор

свойств) (см. рис. П2.6, *вверху*). Поскольку вы можете применить каждое отдельное имя ID всего один раз на странице, этот список содержит только те идентификаторы, которые находятся в таблице стилей, но еще не были применены к тегам на странице.

Для нетекстовых элементов выберите нужный и введите имя идентификатора в соответствующем поле в Property inspector (Инспектор свойств) (для некоторых элементов это поле не помечено, но вы всегда можете найти его в крайней левой части Property inspector (Инспектор свойств)).

Вы также можете задействовать селектор тегов, как указано на рис. П2.7. Просто выберите пункт Set ID (Задать идентификатор) в контекстном меню, появляющемся при щелчке правой кнопкой мыши на теге.

СОВЕТ

Селектор тегов скажет вам, был ли применен ID к тегу. ID обозначается символом #, так что на рис. П2.7, например, body#catalog означает, что к тегу <body> применен ID с именем catalog.

Каждый раз, когда вы применяете ID к тегу, Dreamweaver добавляет немного HTML-кода к странице. Так, стиль ID под названием #copyright, примененный к абзацу, будет выглядеть в HTML следующим образом: `<p id="copyright">`.

Чтобы удалить ID из текстового элемента, укажите нужный кусок текста, а затем выберите значение None (Нет) в списке ID (Идентификатор) в Property inspector (Инспектор свойств). При работе с нетекстовым элементом выберите его и удалите имя идентификатора в этом списке.

Редактирование стилей

При создании сайта вы практически все время совершенствуете его дизайн. Тот бледно-зеленый цвет, который вы назначали в качестве фонового для ваших страниц, возможно, выглядел прекрасно в два часа ночи, но в нем что-то теряется при дневном свете.

К счастью, один из самых больших аргументов в пользу покупки Dreamweaver — возможность легко обновлять форматирование сайта.

ЧАСТО ЗАДАВАЕМЫЕ ВОПРОСЫ

Когда форматирование исчезает

Иногда, когда я копирую текст с одной веб-страницы и вставляю его в другую, все форматирование исчезает. Что же происходит?

Когда вы используете каскадные таблицы стилей, имейте в виду, что фактическая информация о стиле хранится либо в теге <head> веб-страницы (для внутренних таблиц стилей), либо в отдельном CSS-файле (внешняя таблица стилей). Если страница включает внутреннюю таблицу стилей, то, когда вы копируете текст, графику или другие элементы страницы, Dreamweaver копирует эти элементы и любые используемые

ЧАСТО ЗАДАВАЕМЫЕ ВОПРОСЫ

ими определения классов стилей. Когда вы вставляете HTML в другую страницу, стили описываются в ее теге <head>. Эта особенность может сохранить для вас немного времени, но не решит всех проблем. Программа, например, не копирует созданные вами теги стилей или наиболее усовершенствованные стили, которые вы могли бы создать (см. разд. «Другие селекторы» гл. 3). Так, если вы копируете и вставляете некоторый текст, скажем, определенный в теге <h1>, разработанном с тегом стиля h1, то тег <h1> и его содержимое будут добавлены в другую страницу, в отличие от тега стиля.

Вдобавок, если страница использует внешнюю таблицу стилей, то, когда вы копируете и вставляете текст, сами стили не присоединятся.

Если вы копируете абзац, к которому относится класс стиля, и вставляете его в другой документ, код абзаца будет добавлен (например, <p class = "company">), но сам стиль .company со всеми свойствами форматирования — нет.

Лучшее решение состоит в том, чтобы использовать общую внешнюю таблицу стилей для всех страниц вашего сайта. Таким образом, когда вы копируете и вставляете HTML, все страницы совместно используют одни и те же стили и форматирование. Так, в примере выше, если вы скопируете абзац, который включает класс стиля (class = "company") в другую страницу, которая совместно использует ту же самую таблицу стилей, то абзацы будут выглядеть одинаково на обеих страницах.

ПРИМЕЧАНИЕ

Когда вы редактируете или добавляете стили к внешней таблице стилей, Dreamweaver не всегда позволяет отменять сделанные изменения.

Dreamweaver предоставляет много способов редактирования стилей.

- Выберите стиль на панели **CSS Styles** (Стили CSS) (см. рис. П2.1), а затем нажмите кнопку **Edit Style** (Редактировать), чтобы открыть окно **CSS Rule Definition** (Определение правила CSS) (то же самое окно, которое вы использовали, когда создавали стиль в первый раз (см. рис. П2.3)). Внесите изменения, а затем нажмите кнопку **OK**, чтобы вернуться к окну документа. Dreamweaver переформатирует страницу, чтобы отразить те изменения, которые вы внесли в стили, используемые в текущем документе.
- Двойной щелчок кнопкой мыши на названии стиля на панели **CSS Styles** (Стили CSS) также откроет окно **CSS Rule Definition** (Определение правила CSS). В зависимости от настроек двойной щелчок на названии стиля на панели **CSS Styles** (Стили CSS) может отображать «сырой» код CSS на панели **Code view** (Обзор кода). У вас есть возможность изменить такое поведение, открыв окно **Preferences** (Установки) (сочетание клавиш **Ctrl+U**), выбрав раздел **CSS Styles** (Стили CSS) и установив нижний переключатель в положение **Edit using CSS dialog** (Изменить в представлении кода).
- Щелкните правой кнопкой мыши на названии стиля на панели **CSS Styles** (Стили CSS) и выберите в контекстном меню пункт **Edit** (Редактировать), что также откроет окно **CSS Rule Definition** (Определение правила CSS). Внесите свои изменения в стиль, а затем нажмите кнопку **OK**, чтобы вернуться к окну документа.

ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

Время для дизайна

Функция Dreamweaver под названием Design Time Style Sheets (Таблицы стилей времени разработки) позволит вам быстро «опробовать» различные таблицы стилей CSS при разработке веб-страницы. Вы можете скрыть внешние таблицы стилей, которые присоединили к веб-странице, и заменить новыми.

Таблицы стилей времени разработки полезны при работе с HTML-кодом, который вы впоследствии намереваетесь сделать частью полной веб-страницы. Элементы Dreamweaver Library (библиотеки Dreamweaver) — хороший пример; эта особенность позволяет вам создать кусок HTML-кода, который вы можете использовать в любых страницах своего сайта. Когда вы обновляете элемент библиотеки, обновляется каждая страница, которая использует его. Эта возможность экономит время, но, так как элемент библиотеки является только частью страницы, он не включает раздел <head>, необходимый либо для хранения стилей, либо для присоединения внешней таблицы стилей. Таким образом, при разработке элемента библиотеки вы работаете вслепую (или, по крайней мере, без стиля). Но, используя таблицы стилей времени разработки, вы можете обращаться ко всем стилям во внешней таблице стилей и даже предварительно просматривать результаты непосредственно в режиме Design view (Обзор дизайна).

Вы также будете обращаться к этой возможности, работая с новыми инструментами XML в Dreamweaver, которые позволяют добавлять фрагмент XSLT к полной веб-странице — по существу, позволяя преобразовывать XML в HTML. Но, чтобы точно спроектировать эти компоненты, вы должны будете использовать таблицы стилей времени разработки.

Чтобы применить таблицу стилей времени разработки к вашей веб-странице, нажмите соответствующую кнопку на панели инструментов Style Rendering (Отображение стиля) (см. рис. П2.5) либо выберите меню Format ▸ CSS Styles ▸ Design Time (Форматирование ▸ Стили CSS ▸ Время конструирования). Появится окно Design Time Style Sheets (Таблицы стилей времени разработки). Нажмите верхнюю кнопку со знаком «плюс» (+), чтобы выбрать внешнюю таблицу стилей для отображения в Dreamweaver. Заметьте, что нажатие этой кнопки не приводит к присоединению таблицы стилей к странице. При этом просто выбирается CSS-файл, который будет использован при просмотре страницы в Dreamweaver.

Чтобы должным образом просмотреть страницу с этой новой таблицей стилей, вам, возможно, понадобится убрать присоединенную внешнюю таблицу стилей. Чтобы сделать это, используйте нижнюю кнопку со знаком «плюс» (+) и добавьте таблицу к списку Hide (Скрыть).

Таблицы стилей времени разработки применяются, только когда вы работаете в Dreamweaver. Они никак не влияют на то, как отображается страница в самом браузере. Это и хорошо, и плохо. Хотя Dreamweaver позволяет применять к веб-странице классы стилей, которые вы берете из таблицы стилей времени разработки, он в действительности не присоединяет внешнюю таблицу стилей к соответствующей странице. Например, если вы используете таблицу стилей времени разработки, чтобы облегчить создание дизайна элемента библиотеки, Dreamweaver не гарантирует, что к веб-странице, использующей элемент библиотеки, будет присоединена таблица стилей. Вы должны присоединить ее сами, когда закончится время разработки, иначе ваши посетители никогда не смогут увидеть тот результат, который вы предполагали.

Редактирование на панели Properties (Свойства). Окно CSS Rule Definition (Определение правила CSS) (см. рис. П2.3) предлагает довольно утомительный способ редактирования CSS-свойств. Оно удобное в использовании, но открытие окна и переключение между категориями и меню могут замедлять работу, если вы хорошо

знаете CSS. Но, к счастью, Dreamweaver располагает таким инструментом, как панель **Properties** (Свойства). Она используется для ускорения процесса редактирования стилей. Панель **Properties** (Свойства) отображает текущие определенные свойства выбранного стиля, а также список остальных, еще не установленных свойств CSS.

Выберите стиль, который желаете отредактировать, на панели **CSS Styles** (Стили CSS), и на панели **Properties** (Свойства) отобразятся CSS-свойства в одном из трех видов:

- набор свойств — показываются только те свойства, которые были определены для выбранного стиля (рис. П2.8);

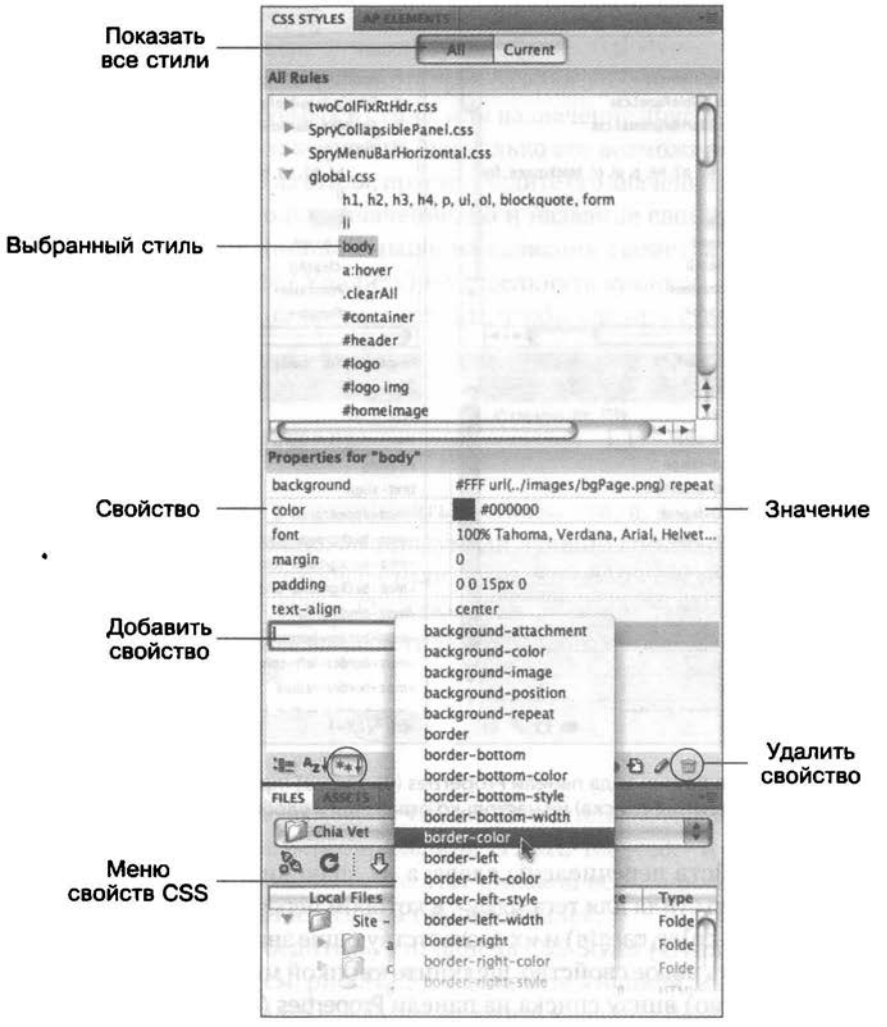


Рис. П2.8. Два вида панели CSS Styles (Стили CSS): All (Все) (показан здесь) и Current (Текущий). Панель Properties (Свойства) доступна в каждом из них, но вы получаете к ней доступ немного по-другому, находясь в режиме Current (Текущий)

- представление категории — группирует различные CSS-свойства в те же самые семь категорий, которые использовались в окне **CSS Rule Definition** (Определение правила CSS) (рис. П2.9, *слева*);
- представление списка — выводит алфавитный указатель всех свойств CSS (см. рис. П2.9, *справа*).

Специальные кнопки в левом нижнем углу панели **CSS Styles** (Стили CSS) позволяют переключаться между этими тремя видами (выделено на рис. П2.8 и П2.9).

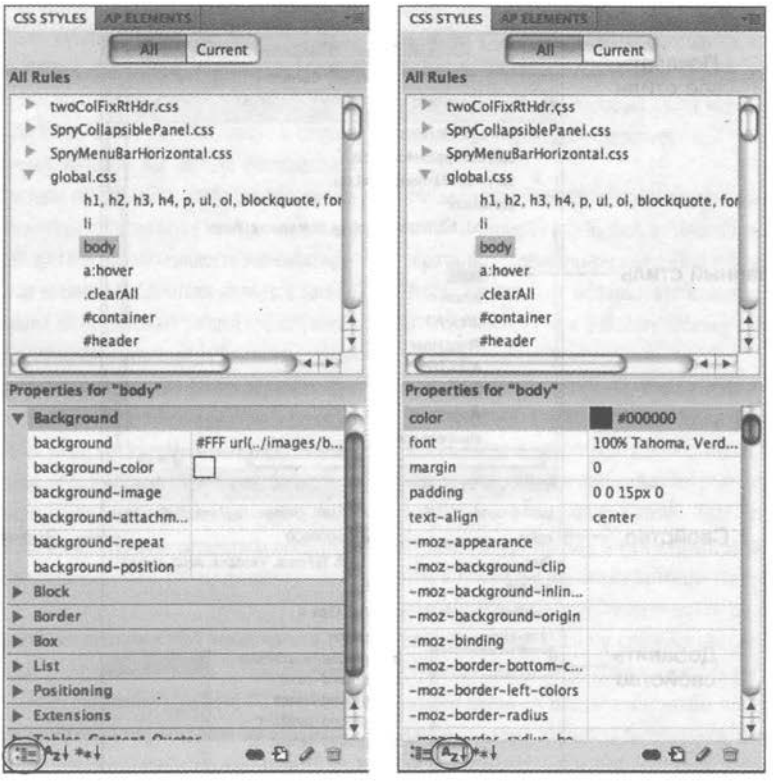


Рис. П2.9. Два других вида панели Properties (Свойства) (представление категории и представление списка) не настолько отлажены и удобны в использовании

Названия свойств перечислены слева, а их значения — справа. Рисунок П2.8 показывает пример стиля для тега <body>, в котором перечислено шесть свойств (таких как `background-color`, `margin`) и их соответствующие значения (#333333, 0px и т. д.).

Чтобы добавить новое свойство, щелкните кнопкой мыши на ссылке **Add Property** (Добавить свойство) внизу списка на панели **Properties** (Свойства) и выберите название свойства в раскрывающемся списке. Вы можете редактировать значение конкретного свойства в поле справа от названия. В большинстве случаев не нужно набирать значение. Dreamweaver обеспечивает вас инструментами, которые, веро-

ятно, понадобятся для каждого свойства: палитра цветов для любого свойства, которому нужен цвет; раскрывающийся список для свойств, у которых ограниченное количество возможных значений, например Repeat-y для свойства background-repeat, которое показано на рис. П2.8; и значок в виде папки **Browse** (Обзор) для свойств, которым требуется путь к файлу, таких как, например, свойство background-image.

Некоторые другие свойства, в свою очередь, требуют от вас какого-то знания CSS, чтобы набрать их вручную в правильном формате. Это то, что делает панель **Properties** (Свойства) отличным средством для тех, у кого уже есть достаточный опыт работы с CSS.

Но даже те, у кого нет такого опыта, найдут панель **Properties** (Свойства) полезной. Во-первых, ее использование — это лучший способ получить панорамный вид свойств стиля. Во-вторых, для действительно базового редактирования, такого как изменение цветов, используемых в стиле, или назначение другого шрифта, панель **Properties** (Свойства) настолько удобна, насколько это возможно.

Чтобы удалить свойство из стиля, просто удалите его значение в правом столбце. Dreamweaver удаляет не только значение, но и название свойства. Вдобавок вы можете щелкнуть правой кнопкой мыши на названии свойства и выбрать в контекстном меню пункт **Delete** (Удалить) или щелкнуть кнопкой мыши сначала на названии свойства, а затем на значке корзины, чтобы удалить свойство из таблицы стилей (см. рис. П2.8).

Управление стилями

Иногда вместо того, чтобы редактировать свойства стиля, вы хотите удалить его и начать заново. Или, возможно, вы придумали лучший способ организовать сайт и хотите переименовать некоторые стили согласно новой системе. Dreamweaver облегчает выполнение этих задач. Он даже позволяет вам дублировать стиль, так что вы можете быстро создать новый стиль, у которого есть немного общего с тем, что вы уже создавали с нуля.

Удаление стиля

В какой-то момент вы можете понять, что создали стиль, который в конце концов вам не нужен. Возможно, вы переопределили HTML-тег <code> и обнаружили, что даже не использовали его в своем сайте. Нет никакой необходимости хранить его, чтобы он занимал драгоценное место в таблице стилей.

Чтобы удалить стиль, убедитесь, что панель **CSS Styles** (Стили CSS) открыта, а кнопка **All** (Все) нажата (см. рис. П2.9). Щелкните кнопкой мыши на названии стиля, который желаете удалить, а затем нажмите клавишу **Delete** (или щелкните кнопкой мыши на значке корзины внизу панели). Вы также можете удалить все стили во внутренней таблице стилей, выбрав ее (обозначена посредством **<style>** на панели **CSS Styles** (Стили CSS)) и нажав клавишу **Delete** или щелкнув кнопкой

мыши на значке корзины. Когда вы «выбрасываете в корзину» внешнюю таблицу стилей, вы просто отсоединяете ее от текущего документа, на самом деле не удаляя CSS-файл.

К сожалению, удаление класса стиля не удаляет ссылки на этот стиль в страницах вашего сайта. Если вы создали стиль `.company`, применяли его повсюду в вашем сайте, а потом удаляете его из таблицы стилей, то Dreamweaver не удалит теги `<span>` или свойства класса, которые обращаются к этому стилю. В коде ваших страниц все еще будут появляться «осиротевшие» команды, такие как `<span class="company">CosmoFarmer</span>`, даже притом, что текст потеряет разработанный стиль. Вы можете удалить их вручную, используя мощный инструмент программы Dreamweaver — **Find and Replace** (Найти и заменить).

ЧАСТО ЗАДАВАЕМЫЕ ВОПРОСЫ

Когда отмена изменений не работает

Иногда, когда я редактирую стиль, скажем, меняю цвет шрифта, я могу отменить это изменение. Но в некоторых случаях у меня нет возможности сделать это. Почему так происходит?

Вы можете отменять только те изменения, что были внесены в документ, над которым вы работаете в настоящее время. Так, например, вы добавили внутреннюю таблицу стилей к документу. Если вы отредактируете один из этих стилей, Dreamweaver позволит отменить эти изменения. Поскольку стили из внутренней таблицы стилей являются частью веб-страницы, над которой вы работаете, то, выбирая Edit ▸ Undo (Правка ▸ Отменить), вы отменяете последние изменения стиля.

Однако если вы используете внешнюю таблицу стилей, то на самом деле работаете с двумя различными файлами в одно и то же время — с веб-страницей, которую разрабатываете, и файлом таблицы стилей, в котором добавляете, удаляете или редактируете стили. Таким образом, если вы разрабатываете веб-страницу и изменяете стиль, который содержится во внешней таблице стилей, вы в действительности вносите изменения в файл таблицы стилей. В этом случае, выбрав Edit ▸ Undo (Правка ▸ Отменить), вы отменяете только последние изменения, внесенные в файл веб-страницы. Если вы хотите отменить изменения, внесенные во внешние таблицы стилей, вам нужно использовать функцию связанных файлов, новую для Dreamweaver CS4. Имя внешней таблицы стилей появится на панели инструментов связанных файлов (Related Files), которая будет находиться под заголовком файла веб-страницы. Щелкните кнопкой мыши на имени файла, чтобы перейти к его коду, а затем выберите Edit ▸ Undo (Правка ▸ Отменить). Нажмите кнопку Source Code (Исходный код), чтобы вернуться в Сеть.

Переименование класса стиля

Изменить название стиля можно, выбрав его на панели **CSS Styles** (Стили CSS), подождав секунду и щелкнув на его имени еще раз. Таким образом, вы сделаете название стиля редактируемым и сможете ввести новое имя. Разумеется, если вы меняете стиль с названием `p` на стиль с названием `h1`, то, по существу, убираете стиль тега `<p>` и добавляете стиль тега `<h1>`. Другими словами, все абзацы потеряют стилевое форматирование, а все теги `h1` внезапно изменят свой внешний вид. В качестве альтернативного варианта можете открыть CSS-файл в режиме просмотра кода (Code view) и отредактировать название стиля. Однако, когда дело доходит до классов стилей, изменение имени не принесет ничего хорошего, если вы уже

применяли повсюду этот стиль в сайте. Старое название все еще будет появляться в HTML везде, где оно использовалось. Что действительно нужно сделать, так это переименовать стиль, а *затем* выполнить операцию «найти и заменить», чтобы изменить название повсюду, где оно появляется в вашем сайте. Dreamweaver предлагает для этого удобный инструмент.

Чтобы переименовать класс стиля, выполните следующее.

1. В списке **Class** (Класс) на панели **Property inspector** (Инспектор свойств) выберите пункт **Rename** (Переименовать). Появится окно **Rename Style** (Переименовать стиль) (рис. П2.10).

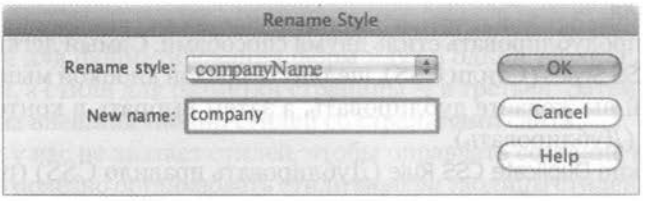


Рис. П2.10. Использование инструмента Rename Style (Переименовать стиль) — быстрый и простой способ изменить название класса стиля, даже если вы уже использовали его сотни раз повсюду на сайте

2. В раскрывающемся списке выберите название стиля, который желаете переименовать. В списке перечислены все классы стилей, доступные на текущей странице, включая внешние и внутренние стили.
3. Введите новое название стиля в поле **New name** (Новое имя). Вы должны следовать тем правилам для именования классов стилей, которые были описаны в разд. «Селекторы классов: точное управление» гл. 3. Но, как и при создании нового класса, вам не нужно ставить точку перед названием — Dreamweaver позаботится об этом.
4. Нажмите кнопку **OK**. Если стиль, имя которого вы изменяете, является внутренним, Dreamweaver внесет изменения. Работа выполнена.

   Если же стиль принадлежит внешней таблице стилей, то Dreamweaver предупредит вас о том, что другие страницы сайта также могут использовать этот стиль. Для успешного переименования стиля Dreamweaver должен использовать свой инструмент **Find and Replace** (Найти и заменить), чтобы выполнить поиск по сайту и обновить все страницы, которые используют старое название стиля. В этом случае переходите к следующему шагу.
5. Если вы не уверены, нажмите кнопку **Cancel** (Отмена), чтобы не изменять название. Если же все в порядке, нажмите кнопку **Yes** (Да), чтобы открыть окно **Find and Replace** (Найти и заменить), где вы должны нажать кнопку **Replace All** (Заменить все). Появится одно последнее предупреждение, напоминающее о том, что это действие не может быть отменено.

ПРИМЕЧАНИЕ

Если вы нажмете **No** (Нет) в окне предупреждения, которое появляется после шага 4, Dreamweaver, тем не менее, переименует стиль во внешней таблице стилей, но не обновит ваши страницы.

6. Нажмите кнопку Yes (Да). Dreamweaver пройдет через каждую страницу вашего сайта, послушно обновляя название стиля в любом месте, где оно появляется.

Дублирование стиля

Dreamweaver облегчает дублирование стиля CSS, которое удобно, когда вы создали, скажем, стиль HTML-тега, а потом решили, что лучше сделать его классом стиля. Или вы захотели использовать форматирование одного стиля как точку отсчета для нового стиля. В любом случае вы начинаете с дублирования существующего стиля.

Вы можете продублировать стиль двумя способами. Самый легкий метод — открыть панель CSS Styles (Стили CSS), щелкнуть правой кнопкой мыши на названии стиля, который вы желаете дублировать, а затем выбрать в контекстном меню пункт Duplicate (Дублировать).

Появится окно Duplicate CSS Rule (Дублировать правило CSS) (рис. П2.11), где вы можете задать продублированному стилю новое название, повторно настроить его тип, установить переключатель Define in (Определить в), чтобы переместить стиль из внутренней таблицы стилей во внешнюю, и т. д.

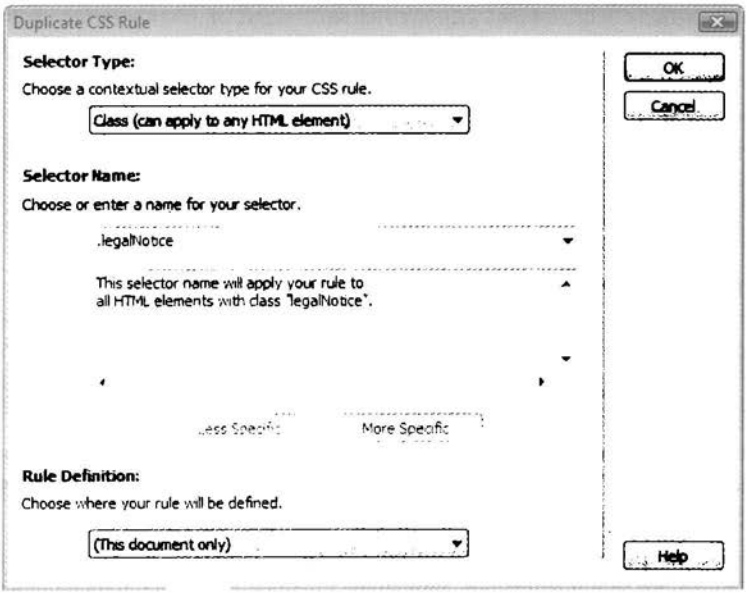


Рис. П2.11. Окно Duplicate CSS Rule (Дублировать правило CSS) выглядит и работает точно так же, как окно New CSS Rule (Создать правило CSS). Единственное отличие состоит в том, что продублированный стиль сохраняет все свойства исходного стиля

Когда вы нажимаете кнопку OK, Dreamweaver добавляет стиль-дубликат на страницу или во внешнюю таблицу стилей. Вы можете отредактировать новый стиль точно так же, как и любой другой, о чем говорилось ранее в этом приложении.

Перемещение и управление стилями

В старые времена, когда браузеры только начали адаптироваться к CSS, веб-дизайнеры использовали лишь несколько стилей для форматирования заголовков и текста. Отслеживание стилей сайта было несложным занятием. Но сегодня, когда CSS прекрасно работает практически во всех браузерах, а основанная на CSS разметка становится нормой, таблица стилей может включать в себя сотни стилей.

Возможно, вы захотите взять очень длинную и сложную таблицу стилей и разделить ее на несколько меньших, легких для чтения внешних таблиц стилей. Одна распространенная методика веб-дизайна состоит в хранении стилей, которые служат для выполнения связанных функций, в отдельной таблице стилей, например, все стили для форматирования форм идут в одной таблице, стили для текста — в другой, а стили для разметки страницы — в третьей. Затем вы можете связать каждую из внешних таблиц стилей со страницами сайта.

Даже если у вас не хватает стилей, чтобы оправдать создание нескольких таблиц, все равно полезно организовать стили *внутри* таблицы стилей. Чтобы следить за своим CSS-кодом, веб-дизайнеры часто группируют связанные стили в одной таблице. Например, все стили для основной разметки содержатся в одном разделе таблицы, основные селекторы тегов — в другом разделе, а специфичные стили для текста, изображений и другого содержимого сгруппированы в соответствии с частью страницы, где они применяются (боковая панель, баннер и т. д.). Сгруппировав связанные стили, вы быстрее найдете какой-либо из них, когда его нужно будет отредактировать.

Dreamweaver предоставляет простой и логичный способ перемещения стилей внутри таблицы стилей и перемещения стилей от одной таблицы стилей к другой.

- Для перемещения стиля из одного места в другое внутри той же таблицы стилей перетащите стиль на панель **CSS Styles** (Стили CSS) (рис. П2.12, *слева*). Порядок, в котором перечислены стили, соответствует их порядку в реальном коде CSS, так что перетаскивание одного стиля под другой вызывает перестановку и в коде таблицы стилей. Вы можете выбрать и переместить несколько стилей за один раз, нажимая **Ctrl** перед выбором каждого стиля, который вам нужен, а затем перетаскивая всю выделенную группу стилей (чтобы снять выделение, нажмите **Ctrl** и выберите стиль еще раз). Выберите сразу несколько стилей, щелкнув на одном из них, а затем на другом, держа при этом нажатой клавишу **Shift**: в таком случае выбираются все остальные стили, которые находятся между этими двумя.

ПРИМЕЧАНИЕ

Вы увидите полный список стилей в таблице стилей (и будете иметь возможноть отсортировать их), только если кнопка All (Все) на панели CSS Styles (Стили CSS) (см. рис. П2.12, слева) нажата.

- Для перемещения одного или нескольких стилей между двумя различными таблицами стилей перетащите стили от одной таблицы к другой на панели **CSS Styles** (Стили CSS). Это работает как для перемещения стилей из внутренних

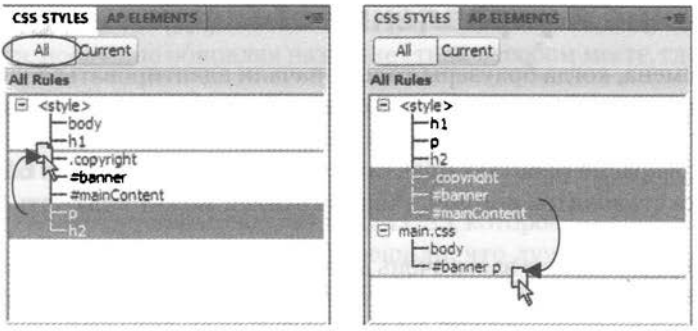


Рис. П2.12. На панели CSS Styles (Стили CSS) вы можете перемещать стили в любые места внутри таблицы стилей либо между различными таблицами стилей

таблиц во внешние, так и для перемещения стилей из внешних таблиц в другие. Скажем, вы создали внутреннюю таблицу стилей для текущей страницы, а также присоединили внешнюю таблицу к той же странице. Перемещение стиля из внутренней таблицы (представленной `<style>` на панели **CSS Styles** (Стили CSS)) во внешнюю (представленную именем файла, например `main.css`), перемещает стиль из внутренней таблицы стилей во внешнюю таблицу стилей (рис. П2.13, *справа*). Затем Dreamweaver удаляет код CSS для стиля из первой таблицы стилей. Вы можете также использовать этот метод для перемещения стилей между двумя внешними таблицами стилей.

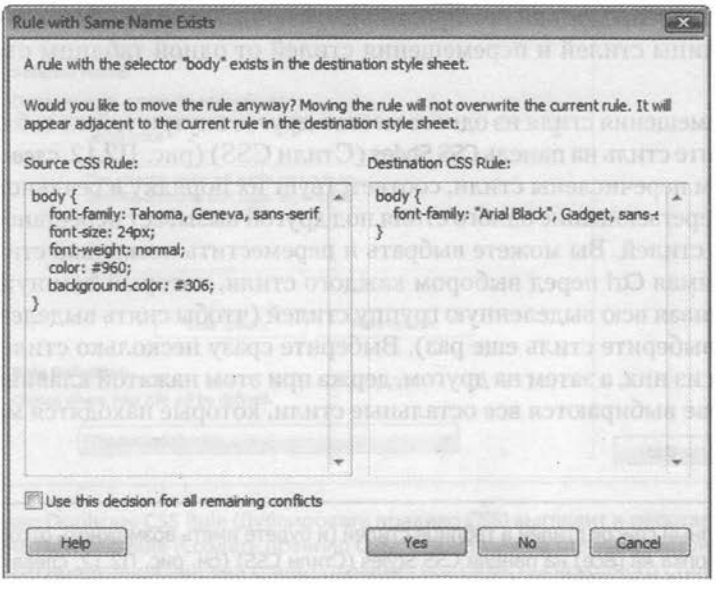


Рис. П2.13. При перетаскивании стиля от одной таблицы стилей к другой есть вероятность, что стиль с таким именем уже существует во второй таблице. В таком случае появляется окно с сообщением, предлагающее либо отменить операцию, либо выполнить ее, несмотря на возможную путаницу

Если вы перетащите стиль в другую таблицу стилей, а в ней уже будет содержаться стиль с тем же именем, может возникнуть путаница. Например, у вас есть стиль для тега <body>, определеный во внутренней таблице стилей. Кроме того, к странице присоединена еще и внешняя таблица стилей, в которой также имеется стиль для тега <body> (вполне вероятно, с другими свойствами). Если вы перетащите стиль тега тела страницы из одной таблицы в другую, то тем самым будете пытаться добавить одноименный стиль во второй раз. Когда это произойдет, Dreamweaver проинформирует вас о возможной проблеме.

ПРИМЕЧАНИЕ

К сожалению, Dreamweaver не предоставляет возможности изменять порядок, в котором следуют внутренние и внешние таблицы стилей на странице. Они присоединяются к странице в последовательности согласно их добавлению. Например, если вы присоединили внешнюю таблицу стилей к веб-странице, а затем создали внутреннюю таблицу стилей, код этой внутренний таблицы будет идти после ссылки на внешнюю таблицу. Порядок может оказывать достаточно серьезное влияние на то, как работает каскадность. Для изменения порядка следования таблиц стилей в HTML вам следует перейти в режим Code view (Просмотр кода) и вырезать/вставлять код.

У вас есть два варианта в такой ситуации. Можно отказаться от перемещения стиля: нажмите кнопку **No** (Нет), окно закроется и стиль не будет перемещен. Если вы нажмете **Yes** (Да), Dreamweaver переместит стиль в таблицу стилей. Он не заменит старый стиль новым и не попытается объединить их вместе в один с тем же именем. Вместо этого он просто разместит новый стиль рядом с уже существующим в пределах одной таблицы. Другими словами, в вашей таблице стилей окажутся два отдельных стиля с одинаковыми именами. Хотя такая ситуация является корректной для CSS (пройдет валидацию), это чревато путаницей в дальнейшем. Желательно удалить один из стилей и при необходимости отредактировать оставшийся стиль так, чтобы в нем содержались все недостающие свойства из удаленного стиля.

ПРИМЕЧАНИЕ

Dreamweaver сообщает о том, что разместит стили, имеющие одно и то же имя, рядом (см. рис. П2.13), но на самом деле это не так. Dreamweaver поместит перемещаемый стиль туда, куда вы перетащите его в перечне стилей таблицы.

- Вы также можете переместить один или несколько стилей во внешнюю таблицу стилей, которая не присоединена к текущей странице. Как отмечалось ранее в книге, внешние таблицы стилей представляют собой наиболее эффективный способ для разработки дизайна коллекции страниц сайта. Однако часто оказывается проще использовать внутренние таблицы стилей, когда вы только начинаете проектировать дизайн.

  Таким образом, пока вы работаете с CSS, вам нужно редактировать всего один файл (веб-страницу с внутренней таблицей стилей), а не два (саму веб-страницу и внешний файл CSS). Но, завершая работу с дизайном, лучше всего переместить стили из внутренней таблицы стилей во внешнюю. Это очень легко.

  На панели **CSS Styles** (Стили CSS) выберите те стили, которые хотите переместить во внешнюю таблицу стилей. Щелкните правой кнопкой мыши и выберите

пункт Move CSS Rules (Переместить правила CSS) (рис. П2.14, *вверху*). Откроется окно Move to External Style Sheet (Переместить во внешнюю таблицу стилей) (см. рис. П2.14, *внизу*). Вы можете либо добавить правила к существующей внешней таблице стилей, нажав кнопку Browse (Обзор) и выбрав внешний CSS-файл на сайте, либо установить переключатель в положение A new style sheet (Новая таблица стилей), чтобы создать новый файл и переместить стили в него. Когда вы нажмете OK, стили будут перенесены в существующий CSS-файл или же появится окно, позволяющее вам назвать и сохранить новый файл.

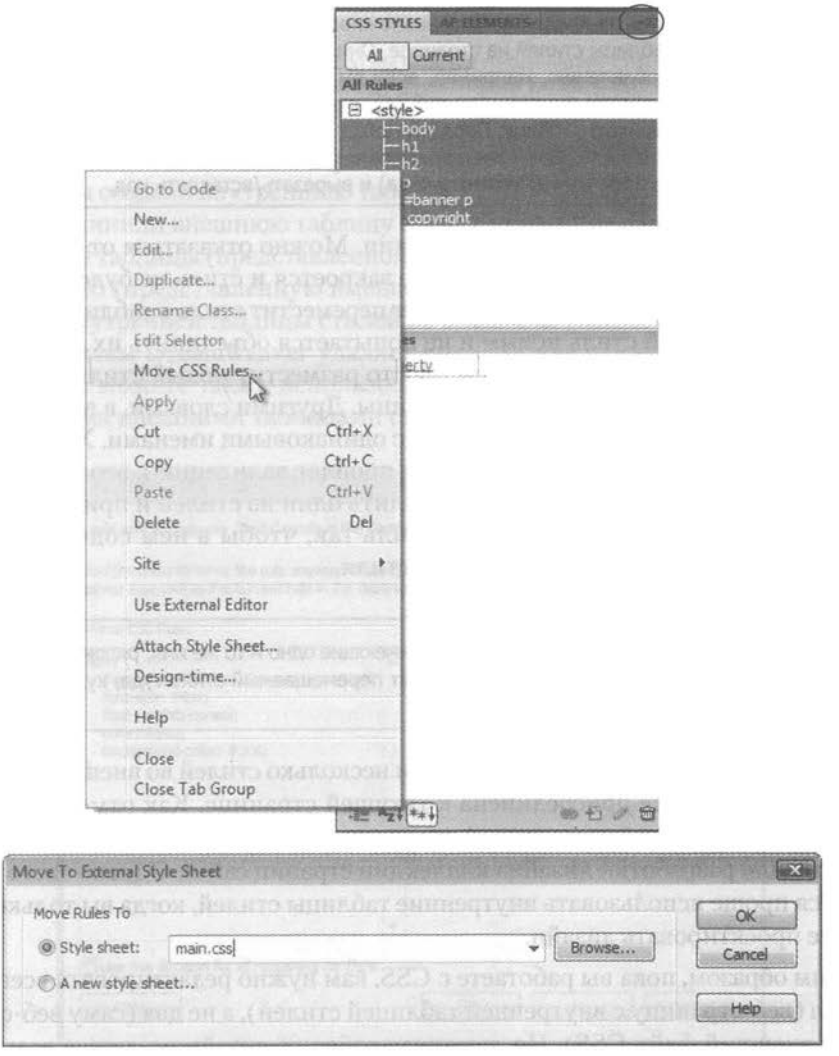


Рис. П2.14. Перемещение внутренних стилей во внешнюю таблицу стилей — двухэтапный процесс. На первом этапе необходимо указать стили, щелкнуть правой кнопкой мыши и выбрать пункт Move CSS Rules (Переместить правила CSS) а на втором — указать программе Dreamweaver, в какую таблицу стилей их перемещать

В любом случае Dreamweaver удаляет стили из внутренней таблицы стилей и помещает их во внешнюю таблицу стилей; а если внешний файл CSS еще не был присоединен к текущей странице, Dreamweaver сделает это за вас, позволяя избежать присоединения таблицы стилей вручную.

СОВЕТ

Если вы перемещаете все стили из внутренней таблицы стилей во внешнюю, Dreamweaver все еще оставляет несколько ненужных тегов <style> на веб-странице. Чтобы удалить их, просто выберите <style> из списка стилей на панели CSS Styles (Стили CSS), а затем нажмите клавишу Delete (Удалить) или щелкните на значке корзины в правом нижнем углу панели.

Просмотр CSS-кода на панели CSS Styles (Стили CSS)

Как вы читали в гл. 4 и 5, наследование и каскадность являются двумя очень важными понятиями в CSS. Наследование предоставляет способ передачи общих свойств, таких как цвет шрифта, потомкам тега, для которого был разработан стиль. Задавая тегу <body> страницы цвет шрифта, вы заставите другие теги на странице использовать тот же самый цвет шрифта. Каскадность — набор правил для определения того, что должен сделать браузер, если к одному и тому же тегу относится множество стилей и между ними появляются конфликты. Каскадность позволяет решить, что нужно делать, если один стиль указывает определенному абзацу отображать текст размером 24 пиксела в высоту, в то время как другой стиль указывает, что текст должен быть высотой 36 пикселов.

Режим текущего выбранного элемента

Благодаря всем этим наследованиям и каскадностям стилям очень легко столкнуться непредсказуемым образом. Чтобы помочь вам различать, как взаимодействуют между собой стили, и знать о возможных конфликтах, Dreamweaver CS4 предлагает другой вид панели CSS Style (Стили CSS) (рис. П2.15). Когда вы нажимаете кнопку Current (Текущий), панель переключается в режим текущего выбранного элемента (Current Selection mode), который представляет информацию о том, как на выбранный элемент на странице — изображение, абзац, таблицу — воздействуют наследованные стили.

Это действительно невероятный инструмент, бесценный при диагностике странного поведения CSS, связанного с наследованием и каскадностью. Но, как и любой потрясающий инструмент, он требует хорошего руководства для изучения принципов работы. Панель переполнена огромным количеством информации; вот краткий обзор того, что она представляет.

- **Краткое изложение свойств стиля для текущего выбранного элемента в области окна Summary for Selection (Сводка по выделению).** Помните все обстоятельства того, как родители передают атрибуты дочерним тегам и как они накапливаются по мере того, как стили каскадируют по странице (это означает, что есть

возможность того, чтобы тег <h1> был отформатирован множеством стилей из множества таблиц стилей)? Раздел **Summary for Selection** (Сводка по выделению) подобен общему итогу внизу электронной таблицы. По существу, он говорит вам, как выбранный элемент — абзац, изображение и т. д. — будет выглядеть, когда браузер подсчитает все стили и отобразит страницу. Для настоящих фанатов CSS эта панель стоит всех денег, отдаваемых за Dreamweaver.

- **Происхождение конкретного свойства показано в области окна About (О)** (см. рис. П2.15, *вверху*). Если ваш заголовок оранжевого цвета, но вы никогда не создавали тег <h1> с таким стилем, то можете узнать, какой стиль и из какой таблицы стилей передает этот цвет заголовку. Ту же информацию вы можете получить, наведя указатель мыши над свойством в разделе **Summary** (Сводка). Вдобавок отметим, что, когда окно **About** (О) является видимым, вам недоступна более полезная панель **Rules** (Правила), обсуждаемая далее. По этой причине желательно пропускать данное окно.
- **Список стилей, которые относятся к текущему выбранному элементу, появляется в области окна Rules (Правила)** (см. рис. П2.15, *внизу*). Поскольку любой элемент может быть получателем бесчисленных свойств CSS, переданных по наследству родительскими тегами, будет полезным увидеть список всех стилей, содействующих текущему виду выбранного на странице объекта.
- **Порядок каскада в области окна Rules (Правила)** (см. рис. П2.15, *внизу*). Стили, которые относятся к текущему выбранному элементу, перечислены не просто так, а в определенном порядке, где самый общий стиль наверху, а наиболее специфические — внизу. Это означает, что, когда одно и то же свойство существует в двух или более стилях, стиль, указанный последним, побеждает.

Несколько примеров помогут продемонстрировать то, как читать панель **CSS Style** (Стили CSS), когда она находится в режиме текущего выбранного элемента. Рисунок П2.15 показывает свойства CSS, затрагивающие выбранный текст (в этом случае абзац внутри области с основным содержимым) на веб-странице. Область **Summary for Selection** (Сводка по выделению) дает вам знать, что если вы просмотрите эту страницу в браузере, то абзац будет отображен полужирным шрифтом типа Tahoma черного цвета, выровненным по левому краю, без отступа, размером 14 пикселов, с межстрочным интервалом 130 % и 5 пикселами пространства для левого поля. Когда вы выбираете свойство в области **Summary for Selection** (Сводка по выделению), а затем нажимаете кнопку **Show Property Information** (Показать сведения о выбранном свойстве) (см. рис. П2.15, *вверху*), в области **About** (О) показывается, откуда прибывает свойство. В данном случае свойство `margin` принадлежит наследуемому селектору (`#mainContent p`), который определен во внешней таблице стилей в файле `global.css`.

Нижнюю область этой панели мы рассматривали ранее. Это область **Properties** (Свойства), которая используется для удаления, добавления и редактирования свойств стиля. Вы просто щелкаете кнопкой мыши в области справа от названия свойства, чтобы изменить его значение, или выбираете ссылку **Add Property** (Добавить свойство), чтобы выбрать новое свойство для стиля.

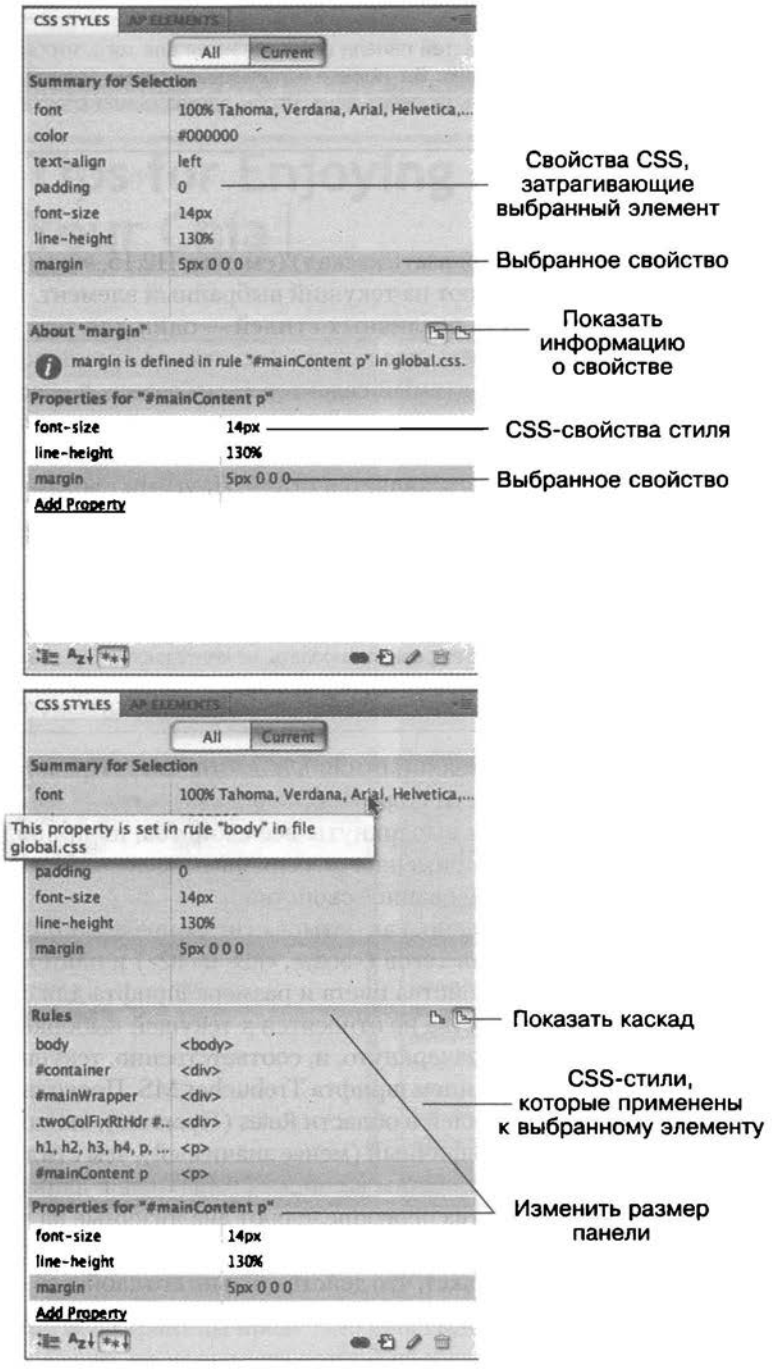


Рис. П2.15. Вид панели стилей Property Information (Сведения о свойстве) показывает, какой стиль и какие таблицы стилей использовались в определении свойства. Вид Cascade (Каскад) представляет список всех стилей (правил), наследуемых текущим выбранным элементом

СОВЕТ

Иногда одна или несколько из этих областей панели слишком малы для того, чтобы можно было увидеть отображенную в них информацию. Вы можете использовать серые полосы, содержащие названия областей окна, и перетаскивать их вверх-вниз, чтобы раскрыть или спрятать отдельную область окна.

Расшифровка каскада

Нажатие кнопки **Show Cascade** (Показать каскад) (см. рис. П2.15, *внизу*) открывает список всех стилей, которые влияют на текущий выбранный элемент. В этом случае вы можете видеть, что шесть различных стилей — один для тега <body>, два стиля ID (#container и #mainWrapper), наследуемый селектор, групповой селектор (h1, h2, h3, h4, p) и, наконец, наследуемый селектор #mainContent p — содействуют разработке стиля выбранного абзаца текста. Вдобавок, как было упомянуто выше, важен порядок, в котором перечислены стили. Чем ниже название появляется в списке, тем более специфическим является стиль. Другими словами, когда несколько стилей содержат одно и то же свойство, побеждает свойство, принадлежащее стилю, который находится ниже в списке.

СОВЕТ

Вы также можете увидеть каскад правил, перечисленных в Property inspector (Инспектор свойств). В документе выберите текст, который хотите проанализировать, нажмите кнопку CSS, а затем откройте список Targeted Rule (Целевое правило) — верхняя группа пунктов в списке представляет собой перечень каскадов, каким он появляется и в области Rules (Правила) панели CSS Styles (Стили CSS).

Щелчок кнопкой мыши на названии стиля в области **Rules** (Правила) показывает свойства этого стиля в области **Properties** (Свойства) ниже. Здесь не только приведены свойства стилей, но и вычеркнуты все свойства, не принадлежащие выбранному стилю. Свойство не применяется, если оно отменено более специфическим стилем либо это не унаследованное свойство.

Например, на рис. П2.16 показано, как четыре стиля определяют форматирование одного заголовка: три стиля тегов (<body>, <h2> и <h2>) и один класс стиля (.highlight). На рисунке слева свойства цвета и размера шрифта для стиля h2 зачеркнуты. Это означает, что свойства не относятся к текущей выборке. Свойство font-family, с другой стороны, не зачеркнуто, и, соответственно, текущая выборка текста отображается с использованием шрифта Trebuchet MS. Поскольку h2 появляется в верхней части списка стилей в области **Rules** (Правила), вы можете определить, что этот стиль менее специфичный (менее значимый), чем стили, перечисленные ниже. Стиль, идущий последним в списке (.highlight в данном случае), самый специфичный, и его свойства переопределяют аналогичные из любых других стилей при возникновении конфликтов. Выбор .highlight в области правил (см. рис. П2.16, *внизу справа*) покажет, что действительно его свойства — для задания размера шрифта и цвета — побеждают в битве каскадных стилей свойств.

СОВЕТ

Когда вы подводите указатель мыши к свойству, которое зачеркнуто, появляется подсказка, где объясняется, почему это свойство не применяет браузер. Если свойство зачеркнуто по причине его переопределения более специфичным свойством, Dreamweaver также сообщит вам, какой стиль победил.

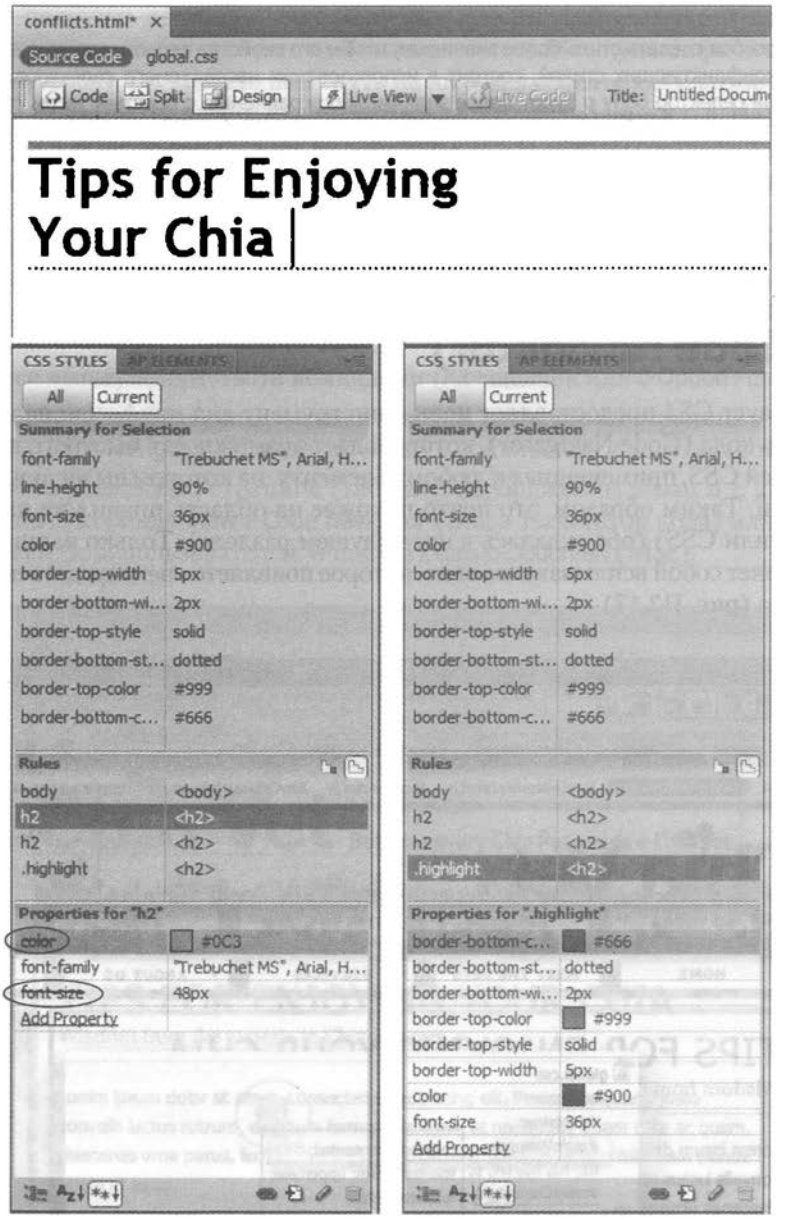


Рис. П2.16. Выбор вида Current (Текущий) на панели CSS Styles (Стили CSS) позволяет просматривать все свойства для стилей веб-страницы

Если ваши веб-страницы предельно просты и используют всего несколько стилей, вы, возможно, не испытаете острой необходимости применения панели CSS Styles (Стили CSS). Но по мере того, как вы будете становиться все более опытными в CSS, вы обнаружите, что эта панель — отличный способ разобраться во многих конфликтах стилей.

СОВЕТ

Один из способов сделать стиль более значимым, чтобы его свойства переопределяли аналогичные из других конфликтующих стилей, состоит в использовании наследуемого селектора. Например, наследуемый селектор body p будет более значимым по сравнению с простым тегом стиля p, хоть оба этих стиля воздействуют на одни и те же теги. Вы можете быстро переименовать стиль или создать более мощный наследуемый селектор с помощью панели CSS Styles (Стили CSS): выберите в ней имя стиля (включите вид All (Все)); щелкните на имени стиля второй раз, чтобы изменить его.

Использование навигатора кода (Code Navigator)

Dreamweaver CS4 предоставляет новый инструмент для профессионалов CSS — навигатор кода (Code Navigator), который дает возможность быстрого просмотра всех стилей CSS, примененных к любому элементу, на котором вы щелкнули кнопкой мыши. Таким образом, это нечто похожее на область правил на панели CSS Styles (Стили CSS) (обсуждалась в предыдущем разделе). Только навигатор кода представляет собой всплывающее окно, которое появляется непосредственно в окне документа (рис. П2.17).

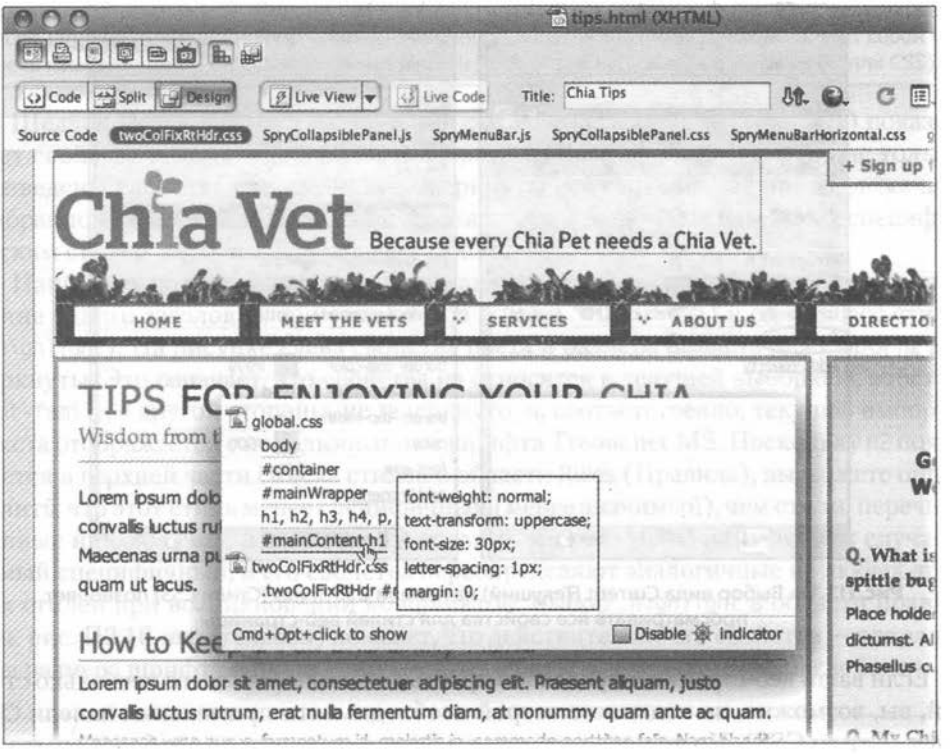


Рис. П2.17. Навигатор кода — новый инструмент в Dreamweaver CS4, позволяющий просматривать список стилей CSS для любого элемента на странице

Чтобы получить доступ к навигатору кода, удерживайте нажатой клавишу Alt и щелкайте на элементе на странице (пользователям Mac нужно при этом нажать клавишу c-Option). Вы можете щелкнуть на любом элементе CSS, который хотите проверить, например на изображении, заголовке, абзаце, таблице и т. д. На рис. П2.18 нажатие Alt (в Windows) и щелчок на заголовке Tips (Советы) открывает навигатор кода, в котором перечислены стили, применявшиеся к этому заголовку.

Есть и другие способы получить доступ к окну навигатора кода.

- Щелкните на значке Code Navigator (Навигатор кода) (обведен на рис. П2.18). Этот значок штурвала появляется над элементом, который вы выбрали на странице (или выше элемента, где в настоящее время находится указатель). Для его появления обычно достаточно подождать пару секунд, но вы можете воспользоваться вышеупомянутыми командами (Alt+щелчок или c-Option+щелчок).
- Щелкните правой кнопкой мыши на любом элементе на странице и выберите команду Code Navigator (Навигатор кода) в появившемся контекстном меню.
- Выберите элемент на странице (таблицу, изображение, абзац и т. д.), а затем выполните команду View ▸ Code Navigator (Вид ▸ Навигатор кода) или нажмите Ctrl+Alt+N (для пользователей Windows) или c-Option+N (Mac).

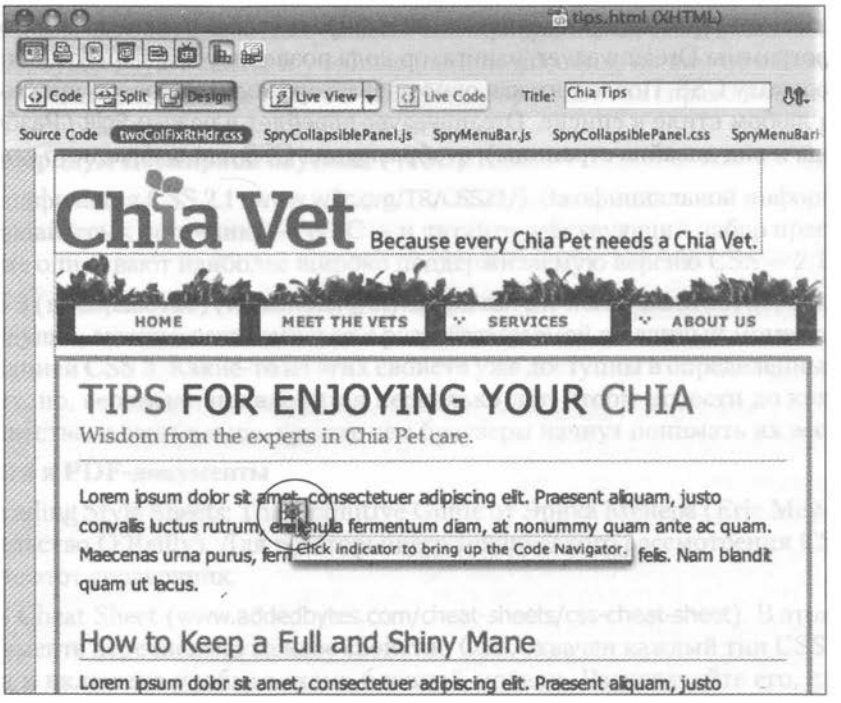


Рис. П2.18. Если постоянно появляющийся значок штурвала надоест вам, уберите его, выбрав соответствующий пункт для отключения в окне Code Navigator (Навигатор кода) (см. рис. П2.17)

После того как окно Code Navigator (Навигатор кода) откроется, вы увидите все стили CSS, которые воздействуют на текущий элемент. На рис. П2.18, например,

перечислены шесть стилей, определяющих форматирование заголовка «Tips for Enjoying Your Chia» — пять стилей находятся во внешней таблице `global.css` и еще один — во внешней таблице стилей `twoColFixRtHdr.css`. Если вы наведете указатель мыши на один из стилей, то увидите список его свойств CSS.

Навигатор кода предоставляет быстрый способ увидеть свойства для всех стилей, которые воздействуют на элемент страницы. На рис. П2.18 наведение указателя мыши на стиль `#mainContent h1` позволяет отобразить его свойства: значение 30 пикселов для `font-size`, 1 пиксел для `letter-spacing` и т. д.

Хотя применение **Code Navigator** (Навигатор кода) — быстрый способ просмотра стилей и их свойств, это средство не настолько полезно, как вид **Current** (Текущий) на панели **CSS Styles** (Стили CSS), который показывает, какие именно свойства (а не только какие стили) применены к текущему выбранному элементу. Кроме того, в окне **Code Navigator** (Навигатор кода) не всегда точно отображается каскадность CSS — он перечисляет стили в порядке значимости, но разделяет этот список по таблицам стилей, так что, если у вашей страницы несколько таблиц стилей, вы не сможете получить четкое представление о каскадности. Область **CSS Rules** (Правила CSS), с другой стороны, показывает полный список стилей от наименее до наиболее специфичных, независимо от того, сколько таблиц стилей вы используете.

Если вы мастер кода, предпочитающий набирать его, не полагаясь на окна и панели программы Dreamweaver, навигатор кода позволяет вам перейти непосредственно к коду CSS. После того как окно навигатора кода откроется, просто щелкните на любом стиле в списке. Dreamweaver перейдет в режим **Split** (Разбиение) (вид кода и вид дизайна страницы), отобразив код CSS для выбранного стиля.

3 Ресурсы по CSS

К сожалению, одна книга не может ответить на все ваши вопросы, касающиеся CSS. Однако, к счастью, есть много ресурсов по CSS, доступных как начинающим, так и опытным веб-разработчикам. В этом приложении вы найдете те ресурсы, которые помогут вам разобраться с общими понятиями CSS и научат решать конкретные задачи CSS, такие как создание навигационной панели или разметка веб-страницы.

Справочники

Справочники по CSS-свойствам бывают как официальные, так и малоизвестные. Конечно, существуют сайты и учебные онлайн-пособия, но вам не обязательно заходить в Интернет, чтобы узнать о CSS. Некоторые из этих справочников можно найти в «старомодном» бумажном варианте.

Консорциум Всемирной паутины (W3C)

- Спецификация CSS 2.1 (www.w3c.org/TR/CSS21/). За официальной информацией обращайтесь к источнику — W3C — и читайте действующий набор правил, которые описывают наиболее широко поддерживаемую версию CSS — 2.1.
- CSS 3 (в разработке) (www.w3.org/Style/CSS/current-work). Если хотите заглянуть в будущее, можете ознакомиться с разрабатываемой на данный момент спецификацией CSS 3. Какие-то из этих свойств уже доступны в определенных браузерах, но, вероятно, понадобится несколько лет, чтобы довести до конца эти новшества, и еще больше, прежде чем браузеры начнут понимать их все.

Книги и PDF-документы

- Cascading Style Sheets: The Definitive Guide от Эрика Мейера (Eric Meyer) (издательство O'Reilly). Для всестороннего технического рассмотрения CSS прочтите этот справочник.
- CSS Cheat Sheet (www.addedbytes.com/cheat-sheets/css-cheat-sheet). В этом PDF-документе перечислено каждое свойство CSS, охвачен каждый тип CSS-селектора и включена удобная схема блочной модели. Распечатайте его, сложите и носите в своем заднем кармане.

Другие онлайн-справочники

- SitePoint CSS Reference (http://reference.sitepoint.com/css). Полный справочник по CSS, включающий описание концепций, живые примеры и обзор третьей версии.

- WesternCiv's Complete CSS Guide (www.westciv.com/style_master/academy/css_tutorial/index.html). Подробный онлайн-справочник по CSS.
- CSS3.Info (www.css3.info). Актуальное описание развития CSS 3 с учетом всех новшеств плюс живые примеры и полезный тест CSS-селекторов, позволяющий испытать браузер на понимание синтаксиса селекторов для CSS 2 и 3.
- Mozilla's CSS Center (https://developer.mozilla.org/En/CSS). Здесь можно найти материалы от разработчиков Firefox — информацию, касающуюся этого браузера. Но также есть и очень полезные общие сведения по использованию CSS, включая несколько подробнейших статей о причудах каскадных таблиц стилей.

Помощь по CSS

Даже при наличии лучших справочников (таких как эта книга) иногда приходится обращаться за помощью к экспертам. Вы можете вступить в дискуссионный клуб, где специалисты по CSS отвечают на вопросы по электронной почте, либо внимательно просмотреть огромное количество информации на специальном форуме.

Рассылки. CSS-Discuss (http://css-discuss.org/). Самая большая из существующих рассылок, посвященная CSS. CSS-эксперты помогают разобраться в проблемах каскадных таблиц стилей.

ПРИМЕЧАНИЕ

Прежде чем задавать в CSS-Discuss вопрос, который ранее до вас уже задавали 47 000 человек, проверьте их вики-совместимый сайт, в котором участники группы свободно добавляют, редактируют и обновляют статьи друг друга. Этот сайт превратился в очень удобный каталог подсказок и приемов, практических методов и исчерпывающего объяснения тем по CSS. Зайдите на страницу http://css-discuss.incutio.com/.

Форумы

- Форум CSSCreator (www.csscreator.com/css-forum/). Очень активный форум, предлагающий помощь и советы по всему, начиная с базового CSS и заканчивая усовершенствованными разметками.
- CSS-форум на SitePoint.com (www.sitepoint.com/forums/forumdisplay.php?f=53). Другая полезная группа для любителей CSS.
- CSS-Tricks.com Forum (http://css-tricks.com/forums). На этом относительно новом форуме можно найти нужную информацию. Если вы программируете на PHP или JavaScript, то найдете здесь много полезного для себя.

Приемы и советы по CSS

Сеть упрощает для всех возможность стать издателем. Однако когда каждый является издателем, тяжело перебирать весь материал и находить в нем понятную, крат-

кую и точную информацию. В Интернете достаточно много качественных материалов по CSS, что, как ни странно, не очень хорошо. Приведу некоторые ссылки по теме CSS из числа лучших.

- CSS-Tricks.com (http://css-tricks.com). Этот блог, который ведет всего один человек, полон прекрасных советов по использованию CSS. Вы найдете здесь часто обновляющиеся советы и рекомендации, а также исчерпывающие видеоуроки.
- Sitepoint (www.sitepoint.com/subcat/css). Учебники по CSS от Sitepoint очень хороши, хоть и обновляются не так часто.
- Smashing Magazine (www.smashingmagazine.com/category/css). Здесь собраны одни из лучших ресурсов в Интернете, а в категории CSS вы найдете практически бесконечное количество ссылок, освещающих самые креативные подходы к применению каскадных таблиц стилей для создания веб-дизайна.

CSS-навигация

Глава 9 показывает вам, как с нуля создавать навигационные кнопки для сайта. Но учебные онлайн-пособия — отличный способ закрепить знания. Кроме того, как только вы поймете весь процесс в подробностях, вам не нужно будет делать это самим каждый раз. В Интернете вы можете найти примеры средств навигации для вдохновения.

Учебные пособия

- Listutorial (http://css.maxdesign.com.au/listutorial/). Пошаговые пособия по созданию навигационных систем из неупорядоченных списков.
- 30 Excellent CSS Based Navigation and Buttons Tutorial (www.instantshift.com/2009/01/11/30-excellent-css-based-navigation-and-buttons-tutorial). Множество пособий, с которыми вы можете ознакомиться.
- Create Apple's Navigation Bar with CSS (http://westciv.com/style_master/blog/apples-navigation-bar-using-only-css). Если вам нравится простой и понятный внешний вид сайта Apple.com, вас должно заинтересовать, как создавать подобные меню с помощью CSS.
- CSS Vertical Navigation Bar with Teaser (www.sohtanaka.com/web-design/cssvertical-navigation-with-teaser). Этот метод создает навигационную панель, причем настолько интерактивную, что можно подумать, будто она сделана с помощью JavaScript. Но нет, чистый CSS!

Примеры в режиме онлайн

- CSS Navigation Bar Code Generator (http://lab.mattvarone.com/navbar). Ленитесь или желаете сэкономить время? Позвольте этому онлайн-инструменту создать весь необходимый код для использования волшебного метода, рассмотренного ранее в этой книге.
- CSS Menus (http://13styles.com/category/css-menus). Загружайте бесплатные CSS-меню, код уже написан за вас!

- CSS Showcase (www.alvit.de/css-showcase/). Галерея навигационных меню, вкладок и методик CSS-навигации.
- Listamatic (http://css.maxdesign.com.au/listamatic/). Выставка основанных на CSS навигационных систем. Здесь также есть много ссылок на родственные сайты.
- Listamatic2 (http://css.maxdesign.com.au/listamatic2/). Еще CSS-меню, включая вложенные списки с подменю.
- CSS Play Menu Showcase (www.cssplay.co.uk/menus/index.html). Большое количество интересных меню, много полезных методик. Обязательно стоит посмотреть.

CSS и графика

Как только вы узнали, как создавать фотогалерею (см. гл. 8), вы стали еще более творческими. Перечислю некоторые сайты, которые показывают приемы с графикой в CSS.

- CSS Play Slideshow (www.cssplay.co.uk/menu/slide_show.html). Основанный исключительно на CSS показ слайдов от креативного разработчика Сту Николза (Stu Nicholls).
- Sliding PhotoGalleries (www.cssplay.co.uk/menu/gallery3l.html). Динамичная, управляемая CSS-галерея.
- CSS Image Maps (www.frankmanno.com/ideas/css-imagemap/). Создайте всплывающие подсказки для фотографий.
- CSS Photo Caption Zoom (http://randsco.com/_miscPgs/cssZoomPZ3.html). Сделайте огромную версию фотографии, которая появляется после проведения указателя мыши над маленьким изображением.
- Revised Image Replacement (www.mezzoblue.com/tests/revised-image-replacement/). Просмотрите различные способы замены HTML-заголовков стильной графикой.

Разметка CSS

Разметка CSS настолько гибкая и удобная для применения, что можно потратить всю жизнь, исследуя ее возможности. И некоторые люди, кажется, делают только это. Вы можете извлечь пользу из их трудов, читая статьи, изучая онлайн-примеры и экспериментируя с инструментами, которые могут сделать некоторую работу с CSS за вас.

Информация о блочной модели

- Interactive CSS Box Model (www.redmelon.net/tstme/box_model/) Забавный интерактивный инструмент для визуализации блочной модели.
- On Having Layout (www.satzansatz.de/cssd/onhavinglayout.html). Этот высокотехничный анализ Internet Explorer объясняет главную причину (и описывает некоторые решения) многих ошибок CSS, досаждающих пользователям браузера Internet Explorer версий 6 и ниже (и некоторых ошибок IE 7).

Плавающие разметки

- Perfect Multi-Column CSS Liquid Layouts (http://matthewjamestaylor.com/blog/perfect-multi-column-liquid-layouts). Отличные примеры самых разных типов плавающих (полноэкранных) разметок, которые работают всюду, начиная от IE 5.5 и заканчивая iPhone.
- Position is Everything (www.positioniseverything.net/articles/onetruelayout/). Интересное, даже немного «заворачивающее мозги» представление о том, как создавать плавающие разметки, которые преодолевают большинство ограничений перемещения.
- CSS Discuss Float Layouts (http://css-discuss.incutio.com/?page=FloatLayouts). Еще больше ссылок на ресурсы о плавающих разметках.

Абсолютное позиционирование

- CSS-Discuss Absolute Layouts (http://css-discuss.incutio.com/?page=AbsoluteLayouts). Хорошие ресурсы с некоторой полезной сопровождающей информацией.
- BarelyFitz Designs (www.barelyfitz.com/screencast/html-training/css/positioning/). Быстрый, практический обзор CSS-позиционирования.
- Making the Absolute, Relative (http://stopdesign.com/archive/2003/09/03/absolute.html). Справочник по использованию абсолютного позиционирования для создания искусных эффектов дизайна.

Примеры разметок

- CSS Layout Generator (www.pagecolumn.com). Выберите количество столбцов, настройте несколько кнопок, и этот сайт сгенерирует весь необходимый HTML- и CSS-код. Когда под рукой такой сайт, кому нужна книга?
- Even More Layout Generators (www.webdesignbooth.com/15-extremely-usefulcss-grid-layout-generator-for-web-designers). Если вас не устраивают возможности сайтов, автоматически создающих код HTML и CSS, здесь вы найдете список из 15 онлайн-инструментов.
- 960 Grid System (http://960.gs). Одна из лучших сред разработки CSS, предоставляющая набор базовых стилей, а также методику использования разделов `div` и имен классов для создания составных, многостолбцовых разметок с фиксированной шириной (подробное видеопредставление этой системы вы найдете на http://nettuts.com/videos/screencasts/adetailed-look-at-the-960-css-framework/).
- YUI Grids CSS (https://developer.yahoo.com/yui/grids). Собственная система CSS-разметки от Yahoo. Как и 960 Grid System, описанная выше, представляет собой базовую среду разработки сложных многостолбцовых разметок.
- Blueprint (www.blueprintcss.org). Еще одна популярная среда разработки CSS.
- Intensivstation Templates (http://intensivstation.ch/en/templates/). Крутые шаблоны, предопределяющие доменное имя.

Разнообразные ресурсы по разметкам

- Adaptive CSS Layouts (www.smashingmagazine.com/2009/06/09/smart-fixesfor-fluid-layouts). Предоставляет множество ресурсов для создания гибких разметок, адаптирующихся к полной ширине окна браузера.

- TJK Desigh (http://tjkdesign.com/articles/one_html_markup_many_css_layouts.asp). Огромная запись в блоге, в которой берется отдельная HTML-страница и демонстрируется восемь различных способов ее разметки только с CSS.
- Variable fixed width layout (www.clagnut.com/blog/1663/). Краткая запись в блоге о методике регулирования количества столбцов на странице, основанной на ширине окна браузера.
- 3-Column Layout Index (http://css-discuss.incutio.com/?page=ThreeColumnLayouts). Практически исчерпывающий список различных разметок с тремя столбцами.

Ошибки браузера

Использование CSS — лучший способ отформатировать веб-страницы, а Internet Explorer 6 для Windows — самый популярный браузер в мире... Так почему он не отображает CSS лучшим образом? Это давний вопрос, но одно ясно точно: вы бы держали сейчас более тонкую книгу, если бы не нужно было посвящать так много страниц обходным путям, связанным с этим браузером (и следующие сайты стали бы ненужными).

Windows Internet Explorer

- How to Attack an Internet Explorer (Win) Display Bug (www.communitymx.com/content/article.cfm?page=1&cid=C37E0). Замечательное введение в отладку ошибок CSS в Internet Explorer.
- RichInStyle's guide to IE 5/5.5 Bugs (www.richinstyle.com/bugs/ie5.html). Эта версия браузера все еще используется, и он все еще доставляет неприятности веб-дизайнерам. Если вам нужна помощь в том, чтобы страницы работали в Internet Explorer 5, посетите указанную страницу.
- Explorer Exposed! (www.positioniseverything.net/explorer.html). Информация о самых распространенных ошибках Internet Explorer и о том, как устранить их.

Выставочные сайты

Доскональное знание стандарта CSS не поможет, если ваше воображение не работает. Отличным источником вдохновения может стать творческая работа других людей. Через поисковые системы вы найдете больше выставочных сайтов по CSS, а я перечислю некоторые из них, где вы можете оценить и изучить красивые CSS-дизайны.

- ZenGarden (www.csszengarden.com). Источник всех сайтов-выставок CSS: много различных дизайнов для одного и того же HTML-кода.
- CSS Beauty (www.cssbeauty.com/). Замечательная галерея вдохновляющих CSS-дизайнов.
- CSS Elite (www.csselite.com). «Выставка всего самого лучшего, что есть в веб-дизайнах на CSS» — так говорят многие...

- CSS Mania (http://cssmania.com/). Еще один сайт-витрина, с марта 2004 года это наиболее обновляемая выставка CSS во всем мире.
- Showcase (http://css-discuss.incutio.com/?page=ShowCase). Список выставочных сайтов и замечательных примеров CSS-дизайна.

Книги по CSS

К сожалению, эта книга не может рассказать вам все, что нужно знать о CSS!

- Web Standards Solutions от Дэна Седерхольма (Dan Cederholm) (издательство Friends of Ed). Хоть и не строго о CSS, эта книга дает превосходное представление о том, как писать хороший HTML-код. Если у вас есть какие-то сомнения относительно того, какие теги вы должны использовать, чтобы создать навигационную панель, как лучше создавать HTML-формы или какой лучший метод для того, чтобы сделать код настолько простым, насколько это возможно, вам необходимо прочитать эту книгу.
- Bulletproof Web Design от Дэна Седерхольма (издательство New Riders). Замечательная книга, рассказывающая о том, как лучше всего создавать CSS-стили, которые могут устоять под напором посетителей, изменяющих размеры текста и окна браузера. Отличные подсказки по созданию разметок, навигационных панелей и др.
- CSS Mastery: Advanced Web Standards Solutions от Энди Бадда (Andy Budd) (издательство Friends of Ed). Много усовершенствованных подсказок по применению CSS, включая хорошие примеры разметок, основанных на CSS, и методики для упрощения вашего CSS и HTML-кода.
- Head First HTML with CSS & XHTML от Элизабет Фримэн и Эрика Фримэна (издательство O'Reilly). Яркое, хорошо иллюстрированное введение в сайты, совмещающие HTML и CSS.
- Flexible Web Design от Зое Майкли Джилленуотер (издательство New Riders) научит вас всему, что вы должны знать для создания гибких (то есть плавающих и полноэкранных) разметок.
- CSS Cookbook от Кристофера Шмидта. Четкие объяснения решений различных задач в CSS.

Программное обеспечение для CSS

Существует много различных способов создания каскадных таблиц стилей (CSS). Самый простой — придерживаться бесплатных текстовых редакторов, которые поставляются вместе с Windows и Mac OS, таких как Блокнот или TextEdit. Но есть и специальные CSS-редакторы, а также полноценные программы для разработки веб-страниц, например Dreamweaver, которые включают в себя инструменты для создания CSS.

На сайте http://css-discuss.incutio.com/?page=CssEditors приводится длинный список различных программ, пригодных для редактирования CSS-кода.

Windows и Mac

- Style Master (www.westciv.com/style_master/product_info/). Это мощный CSS-редактор с длинной историей, который содержит множество инструментов, включая простые мастера, шаблоны-примеры, обучающие программы и полный CSS-справочник.
- Dreamweaver (www.adobe.com/dreamweaver/). Предназначенный не только для CSS, этот инструмент веб-разработки «высшего сорта» включает все, что вам нужно для создания всего сайта. Визуальные инструменты для редактирования облегчают возможность увидеть эффект от CSS на вашей веб-странице по мере того, как вы ее создаете.

Только для Windows

- Top Style (www.newsgator.com/NGOLProduct.aspx?ProdID=TopStyle). Почтенный CSS-редактор, который также позволит вам редактировать HTML-документы, — комплектное приобретение (у одного продавца) для создания веб-страниц. Включает много инструментов для увеличения производительности. Есть также бесплатная «легкая» версия.
- Microsoft Expression Web (www.microsoft.com/expression/products/Web_Overview.aspx). Полноценный инструмент для проектирования сайтов, который очень хорошо работает с CSS.

Только для Mac. CSSEdit (www.macrabbit.com/cssedit/). Простой и недорогой CSS-редактор.

Алфавитный указатель

C

CSS 15
  версия 3 541
  в сравнении с HTML 15
  история 23
  книги 547
  помощь 542
  проверка правильности 50
  ресурсы 541
  сайты с примерами 544

D

doctype, объявление типа документа 40
  автоматическое добавление 42
  при использовании фреймов 41
Dreamweaver 22, 510
  XSLT-фрагменты 522
  библиотека 522
  внешние таблицы стилей, присоединение 515
  и каскадность 113
  классы стилей 517
  копируемый текст, форматирование 520
  панели
    CSS Styles (Стили CSS) 533
    Properties (Свойства) 522
  переименование класса стиля 526
  режим просмотра 53
  стили
    дублирование 528
    редактирование 520
    создание 510
    удаление 525
  таблицы стилей Design Time 522
DTD-файлы 40

E

em, единица измерения 45, 484
  гибкий дизайн 319
  для кнопок 287
  размер шрифта 136

F

Firefox
  overflow, свойство 183
  View Formatted Source, расширение 113
  отступы 152
  панель инструментов разработчика 447
  проверка
    CSS-код 50
    HTML-код 39
  размер шрифта, установка 134

I

Internet Explorer
  * html, прием 185
  измерение в пикселах 484
  настройка списков 153
  ошибки 353, 546
  панель инструментов разработчика 447
  размер шрифта, установка 134
  управление 455
  условные комментарии 457

J

JavaScript
  ID-селекторы 68
  динамические меню 260
  позиционирование столбцов 399
  создание всплывающих сообщений 389
  чередование строк в таблице 298

O

Opera
  overflow
    hidden, свойство 189
  padding, свойство 168
  projection, тип устройства 419

S

Safari
  Web Inspector (анализатор) 113
  отступы 152
  проверка HTML-кода, надстройка 39
  размер шрифта, установка 134
  раскрывающиеся меню 300
  формы 300

U

URL 485
  абсолютные 427
  печать 427

X

XHTML 19
XML (расширяемый язык разметки)
  Document Type Definition (DTD) 40

A

Атрибуты 18
  cellspacing 296
    в таблице 295
  class 67, 451
  id 69, 451
  src 153
  title 83

Б

Баннер 209, 404
Блочная модель 165
  блочные (прямоугольные) элементы 173
  встроенные (inline) элементы 173
  ошибки Internet Explorer 202
  ресурсы 544
Боковое меню
  закругление углов 237
  на всю высоту 349
  создание 198
Всплывающие
  меню 260
  сообщения 83, 389

В

Выравнивание
  таблиц 294
  текста 144
Вычисление ширины/высоты 181

Г

Генерируемое содержимое 77, 153, 508
Границы 175
  Internet Explorer 177
  вокруг плавающего элемента 189
  замена изображениями 234
  навигационные панели 257
  отступы 179
  плавающие элементы 188
  удаление 449
  форматирование 177
Графика 203
  для ссылок 251
  предварительная загрузка 265
  ресурсы 544

З

Значения 46, 482
  HTML 19
  URL 485

десятичные (RGB) 483
длины и размеры 483
ключевые слова 484
пикселы 484
проценты 483
цвета 482

И

Изображения 203
бесплатные примеры 218
заключение в рамку 219
как ссылки 251
надписи 408
обучающий урок 218
повтор 209
предварительная загрузка 265
ресурсы 544
удаление границ 449

К

Каскадность 106
и наследование 107
несколько стилей 110
обучающий урок 119
особенности 111
состояния ссылки 245
стили для принтера 423
условные комментарии 458
Каскадные таблицы стилей (Cascading Style Sheets) 15
Кнопки 249
активизируемые 263
навигационные, форматирование 453
центрирование текста 261
Комментарии 437
условные (для Internet Explorer) 457
Контрастность 138
Конфликт полей 171, 495
Кэш 47, 48
Маркеры
графические 152
диск 149
квадрат 149
окружность 149
удаление 254

Н

Навигация 243
обучающие примеры 543
панели навигации 253, 453
вертикальные 255, 278
всплывающие меню 260
горизонтальные 257, 286
границы 257
панель с вкладками 262, 267
ресурсы 543
Наследование 96, 107
в таблицах стилей 98
значимость 113
ключевых слов 485
обучающий урок 100
ограничения 98
преимущества непосредственно примененного стиля 109
размер шрифта 135, 137

О

Обтекание содержимого 185
порядок написания HTML-кода 188
фоны и границы 188
Объявления 46
блок объявления 45, 46
Основной (базовый) размер шрифта текста 134
Ошибки
Firefox
позиционирование фоновых рисунков 213
Internet Explorer 456, 546
@import, правило 420
after, псевдоэлемент 426
content, свойство 426
peek-a-boo (на просвет) 359
гильотины 359
границы 201
двойное поле 354
курсивный текст 353
многострочные ссылки 249
позиционирование элементов 381, 387, 407
размещение 229

трехпиксельные промежутки 356
удаление промежутков между ссылками 256
исправленный код, размещение 265
ресурсы 546

П

Перезагрузка страницы, принудительная 48
Плавающие элементы 185, 318
clear, свойство 189
внутри плавающих элементов 336
высота 228
горизонтальная панель навигации 260
границы 189
отмена перемещения 342
перепады 351
установка перемещения 342
ширина 332
Подписи 221, 408
Позиционирование 376
position, свойство, ключевые слова 379
абсолютное 377, 391
float, свойство 377
visibility, свойство 389
наложение элементов (z-index) 387
ресурсы 545
родственные отношения 383
столбцы 399
внутри элемента 391
горизонтальное/вертикальное смещение 252
наложение элементов 387
необходимость использования 394
обучающий урок 403
относительное 383
разметка 359
свойства 376
скрытие частей страницы 389
столбцы 399
стратегии 390
фиксированное 377
Internet Explorer 400, 402
фреймы 399
Потомки, отношения тегов 73
Правила
!important 423
@import
ошибки Internet Explorer 420
@media 421
Проверка кода
CSS 50
HTML 39, 68
Просмотр страницы, проблемы 134
Псевдоклассы
active 76, 244
focus 79, 244
hover 76, 244, 247
в Internet Explorer 264
изображения 253
кнопки 250
link 76, 245
visited 76, 244
изображения 253
значимость 112
Псевдоэлементы
after 78, 426
before 77
first-child 79
first-letter 77, 148
first-line 77, 148
значимость 112
форматирование абзацев 148
Путь
абсолютный 208
корневой относительный 208
относительный 59
относительный от документа 208

Р

Разметка
бесплатные шаблоны 545
использование CSS-позиционирования 394
на основе плавающих элементов 329
гибкий дизайн 319
непостоянная ширина 319, 332
ошибки Internet Explorer 353
проблемы 342
ресурсы 545
фиксированная ширина 319, 332, 333

на три столбца 396
предустановленная 334
сведения 545
свойства 498
с фиксированной шириной 318
использование отрицательных полей 339
свойства min- и max- 338
создание из свободной 332
Редакторы
EditPlus 22
FrontPage 22
автоматическое добавление doctype 42
HTML-Kit 21
skEdit 22
TextWrangler 22
Родственные отношения (наследование) 73, 96, 105, 107, 108, 109, 113

C

Свойства 46
background 216, 496
границы 179
кнопки 249
обучающий урок 192
печать 424
плавающие элементы 189
background-attachment 216, 496
fixed, значение 216
Internet Explorer 216
scroll, значение 216
сокращенный вариант 216
background-color 497
кнопки 249
печать 425
сокращенный вариант 216
background-image 203, 497
background-position 211, 252, 497
ключевые слова 211
повторное отображение рисунков 209
предварительная загрузка изображения 266
процентные значения 214
сокращенный вариант 216
точные значения 213
background-repeat 209, 498
border 165, 177, 491
img, тег 247
изображения 203
кнопки 249
окаймление границ 296
подчеркивание ссылок 247
таблицы 295
фон 179
border-collapse 295, 506
значения 296
border-color 492
border-spacing 506
border-style 492
border-width 493
bottom 498
caption-side 506
clear 189, 499
значения 191
колонтитулы 343
плавающие элементы 343
формы 304
clip 499
color 132, 482, 486
для печати 424
значения 132, 486
content 426, 507
cursor 508
display 175, 500
none, значение 175
в сравнении со свойством visability 389
горизонтальные панели навигации 257
и разметка в Internet Explorer 359
ошибка двойного поля 354
таблицы стилей для печати 427
empty-cells 507
float 185, 329, 500
изображения 204
ключевые слова 187
отмена свойством clear 189
порядок исходного кода 188, 330
предотвращение перепадов 351
разметка в Internet Explorer 359
фоны и границы 188
формы 304
font 145, 486
font-family 126, 486

font-size 133, 487
наследование 135, 137
font-style 138, 487
font-variant 139, 487
font-weight 138, 488
height 181, 228, 501
вычисление фактической высоты 181
и свойство overflow 183
left 501
letter-spacing 141, 488
line-height 142, 488
встроенные элементы 174
наследование 144
list-style 154, 490
list-style-image 153, 491
list-style-position 151, 491
list-style-type 149, 491
margin 146, 152, 166, 448, 495
в списках 254
конфликты полей 170, 495
обучающий урок 191
отрицательные значения 172, 229, 338
ошибка двойного поля 354
размеры полей 168
сокращенный набор 169
max-height и max-width 180, 338, 359, 501
orphans 508
outline 493
overflow 183, 503
auto, ключевое слово 183
hidden, ключевое слово 183, 188
scroll, ключевое слово 183
visible, ключевое слово 183
padding 152, 165, 448, 494
в списках 254
встроенные элементы 174, 252
в таблицах 293
для границ 179
для изображений 203, 252
для кнопок 249
размеры отступов 168
сокращенный набор 169
page-break-after и page-break-before 428, 508
page-break-inside 509
position 503
right 504
table-layout 507
text-align 144, 488
в таблицах 294
для горизонтальных панелей навигации 259
text-decoration 140, 488
подчеркивание ссылок 247
text-indent 146, 489
text-transform 139, 489
top 504
vertical-align 294, 489
visibility 389, 504
white-space 490
widows 509
width 181, 255, 505
в таблицах 298
вычисление фактической ширины 181
ширина столбцов 346
word-spacing 141, 490
z-index 387, 505
zoom 346, 393
наследование 96
позиционирование 252
разметка страницы 359
Селекторы 45
ID 68
(символ решетки) 93
в сравнении с HTML-кодом 68
в сравнении с селекторами классов 69, 322
обучающий урок 92
с использованием JavaScript 68
специальное применение 70
атрибутов 82
[] (квадратные скобки) 83
групповые 70, 455
обучающий урок 88
дочерних элементов 80
угловая скобка (>) 80
форматирование списков 82
другие 80

классов 65
. (точка) 65
в сравнении с ID-селекторами 322
несколько классов в одном теге 440
обучающий урок 89
правила именования 65
потомков 71, 73, 450
группирование ссылок 246
изображения 204
обучающий урок 91
сестринских элементов 82
типов (тегов) 63
универсальный (*) 71
Списки
маркированные 149, 213, 218, 236, 254
нумерованные 151
обучающий пример 264
свойства 490
создание 81
форматирование 82, 137, 150, 159
Ссылки 243
внешние
выделение (обучающий урок) 276
стилизация 246
группирование с помощью селекторов потомков 246
использование изображений 251
навигационные панели 253
на вкладках 267
обучающий урок 272
печать 426
подчеркивание 247
состояния 243
стилизация 76, 246
центрирование текста на кнопках 261
Стили 15, 44
в Dreamweaver
добавление 514
панель CSS Styles (Стили CSS) 533
редактирование 520
создание 510
управление 525
встроенные 53
группирование 442
каскадность 106
комбинированные 121
комментарии, добавление 437
конфликт 100
наследование 98
обучающий урок 52
определение, какой принимать 114
организация 438
правила именования 439
размещение 53
ссылки 76, 243
только для печати 421
форматирование 46
Столбцы
добавление в формы 303
используемые теги 298
на всю высоту страницы 346
плавающие элементы 330
позиционирование 398
разметка на три столбца 396
разметка с множеством столбцов (обучающий урок) 360
расчет ширины 352
ресурсы 544
стилизация 298

T

Таблицы 290
выравнивание 294
границы 295
отступы 293
свойства 505
стилизация 293
обучающий урок 304
чередование строк 297
Таблицы стилей 15, 47
аппаратно-зависимые 417
в Internet Explorer 420
добавление 420
типы устройств 419
внешние 47, 49, 57
@import, правило 110, 443
@media, правило 421
Design Time 522
в Dreamweaver 511
определение типа устройства 420

размещение 114
связывание с веб-страницей 51, 59
внутренние 47, 48
@media, правило 421
перевод во внешние 49, 56, 512
организация 438
печать 421
обучающий урок 429
правило !important 422
разрывы страницы 428
скрытие элементов 427
ссылки 426
стилизация фонов 424
текстовые стили 423
Теги
<a> (якорный) 18
 35, 321
как элемент-контейнер 169
организация таблиц стилей 451
перемещаемые элементы 343
<img> 203
border, свойство 247
в таблицах 293
для печати 215
поля/отступы 175
 36
позиционирование элемента 391
Текст
абзацы 142, 156
отступ первой строки 146
с выступающей строкой (выступ) 146
форматирование 156
буквица, создание 77
выравнивание 144
заголовки, форматирование 156
интервал
межсимвольный 141
межстрочный 141
контрастность 138
курсив 138
малые прописные буквы 139
на кнопках 264
наследование 145
на формах 300
нормальный размер шрифта 134
первая буква, первая строка абзаца 147
печать 422
полужирный 138
прописные буквы 139
свойства 486
списки 149, 159
текстовые редакторы 21
украшение 140
мерцание, ключевое слово blink 140
форматирование
обучающий урок 154
цветовое оформление 131, 486
шрифты 126, 448
Теневые эффекты, добавление 228
Типы устройств
all 419
braille 419
embossed 419
handheld 419
print 419
projection 419
screen 419
speech 419
tty (teletype) 419
tv 419

У

Установка цвета фона 179

Ф

Фоновые изображения
background-image, свойства
ссылки 251
background-image, свойство
URL 207
добавление в ссылки 247
обучающий урок 231
печать 215, 425
повторение 209
позиционирование 210
свойство background-image 204
обучающий урок 231
стилизация ссылок (обучающий урок) 272
фиксация на месте 216

Формы
в браузере Safari 300
обучающий урок 311
создание стилей 299
форматирование 302
Фреймы 399
наборы фреймов 399

Ц

Цвета 482

Э

Элементы
HTML-формы
кнопки 301
переключатели 301
поля ввода 301
раскрывающиеся списки 301
тег fieldset 300
тег legend 301
флажки 301
блочные 68, 173, 187
вложенные 74
внутренние 36
встроенные 173, 187
наложение 388
сестринские
соединение, + (плюс) 82
скрытие при печати 427
Элементы-контейнеры 168, 187
для плавающих элементов 345

Дэвид Макфарланд

Большая книга CSS

2-е издание

Серия «Бестселлеры O'Reilly»

Перевел с английского Дубенок И.

Заведующая редакцией	*К. Галицкая*
Ведущий редактор	*Е. Каляева*
Научные редакторы	*Е. Костомарова, П. Макович*
Литературный редактор	*Н. Гринчик*
Художник	*Л. Адуевская*
Корректоры	*О. Андросик, Е. Павлович*
Верстка	*Г. Блинов*

Подписано в печать 13.03.12. Формат 70×100/16. Усл. п. л. 45,15. Тираж 1000. Заказ № 244.

ООО «Мир книг», 198206, Санкт-Петербург, Петергофское шоссе, 73, лит. А29.

Налоговая льгота — общероссийский классификатор продукции ОК 005-93, том 2; 95 3005 — литература учебная.

Отпечатано с готовых диапозитивов в ГП ПО «Псковская областная типография».
180004, Псков, ул. Ротная, 34.

ИЗДАТЕЛЬСКИЙ ДОМ
ПИТЕР®
WWW.PITER.COM

ПРЕДСТАВИТЕЛЬСТВА ИЗДАТЕЛЬСКОГО ДОМА «ПИТЕР»
предлагают эксклюзивный ассортимент компьютерной, медицинской, психологической, экономической и популярной литературы

РОССИЯ

Санкт-Петербург м. «Выборгская», Б. Сампсониевский пр., д. 29а
тел./факс: (812) 703-73-73, 703-73-72; e-mail: sales@piter.com

Москва м. «Электрозаводская», Семеновская наб., д. 2/1, корп. 1, 6-й этаж
тел./факс: (495) 234-38-15, 974-34-50; e-mail: sales@msk.piter.com

Воронеж Ленинский пр., д. 169; тел./факс: (4732) 39-61-70
e-mail: piterctr@comch.ru

Екатеринбург ул. Бебеля, д. 11а; тел./факс: (343) 378-98-41, 378-98-42
e-mail: office@ekat.piter.com

Нижний Новгород ул. Совхозная, д. 13; тел.: (8312) 41-27-31
e-mail: office@nnov.piter.com

Новосибирск ул. Станционная, д. 36; тел.: (383) 363-01-14
факс: (383) 350-19-79; e-mail: sib@nsk.piter.com

Ростов-на-Дону ул. Ульяновская, д. 26; тел.: (863) 269-91-22, 269-91-30
e-mail: piter-ug@rostov.piter.com

Самара ул. Молодогвардейская, д. 33а; офис 223; тел.: (846) 277-89-79
e-mail: pitvolga@samtel.ru

УКРАИНА

Харьков ул. Суздальские ряды, д. 12, офис 10; тел.: (1038057) 751-10-02
758-41-45; факс: (1038057) 712-27-05; e-mail: piter@kharkov.piter.com

Киев Московский пр., д. 6, корп. 1, офис 33; тел.: (1038044) 490-35-69
факс: (1038044) 490-35-68; e-mail: office@kiev.piter.com

БЕЛАРУСЬ

Минск ул. Притыцкого, д. 34, офис 2; тел./факс: (1037517) 201-48-79, 201-48-81
e-mail: gv@minsk.piter.com

Ищем зарубежных партнеров или посредников, имеющих выход на зарубежный рынок.
Телефон для связи: **(812) 703-73-73. E-mail:** fuganov@piter.com

---

Издательский дом «Питер» приглашает к сотрудничеству авторов. Обращайтесь по телефонам: **Санкт-Петербург – (812) 703-73-72, Москва – (495) 974-34-50**

Заказ книг для вузов и библиотек по тел.: (812) 703-73-73.
Специальное предложение – e-mail: kozin@piter.com

Заказ книг по почте: на сайте **www.piter.com**; по тел.: (812) 703-73-74
по ICQ 413763617

ИЗДАТЕЛЬСКИЙ ДОМ ПИТЕР®
WWW.PITER.COM

**УВАЖАЕМЫЕ ГОСПОДА!
КНИГИ ИЗДАТЕЛЬСКОГО ДОМА «ПИТЕР»
ВЫ МОЖЕТЕ ПРИОБРЕСТИ
ОПТОМ И В РОЗНИЦУ У НАШИХ
РЕГИОНАЛЬНЫХ ПАРТНЕРОВ.**

ДАЛЬНИЙ ВОСТОК

Владивосток
«Приморский торговый дом книги»
тел./факс: (4232) 23-82-12
e-mail: bookbase@mail.primorye.ru

Хабаровск, «Деловая книга», ул. Путевая, д. 1а
тел.: (4212) 36-06-65, 33-95-31
e-mail: dkniga@mail.kht.ru

Хабаровск, «Книжный мир»
тел.: (4212) 32-85-51, факс: (4212) 32-82-50
e-mail: postmaster@worldbooks.kht.ru

Хабаровск, «Мирс»
тел.: (4212) 39-49-60
e-mail: zakaz@booksmirs.ru

ЕВРОПЕЙСКИЕ РЕГИОНЫ РОССИИ

Архангельск, «Дом книги», пл. Ленина, д. 3
тел.: (8182) 65-41-34, 65-38-79
e-mail: marketing@avfkniga.ru

Воронеж, «Амиталь», пл. Ленина, д. 4
тел.: (4732) 26-77-77
http://www.amital.ru

Калининград, «Вестер»,
сеть магазинов «Книги и книжечки»
тел./факс: (4012) 21-56-28, 6 5-65-68
e-mail: nshibkova@vester.ru
http://www.vester.ru

Самара, «Чакона», ТЦ «Фрегат»
Московское шоссе, д.15
тел.: (846) 331-22-33
e-mail: chaconne@chaccone.ru

Саратов, «Читающий Саратов»
пр. Революции, д. 58
тел.: (4732) 51-28-93, 47-00-81
e-mail: manager@kmsvrn.ru

СЕВЕРНЫЙ КАВКАЗ

Ессентуки, «Россы», ул. Октябрьская, 424
тел./факс: (87934) 6-93-09
e-mail: rossy@kmw.ru

СИБИРЬ

Иркутск, «ПродаЛитЪ»
тел.: (3952) 20-09-17, 24-17-77
e-mail: prodalit@irk.ru
http://www.prodalit.irk.ru

Иркутск, «Светлана»
тел./факс: (3952) 25-25-90
e-mail: kkcbooks@bk.ru
http://www.kkcbooks.ru

Красноярск, «Книжный мир»
пр. Мира, д. 86
тел./факс: (3912) 27-39-71
e-mail: book-world@public.krasnet.ru

Новосибирск, «Топ-книга»
тел.: (383) 336-10-26
факс: (383) 336-10-27
e-mail: office@top-kniga.ru
http://www.top-kniga.ru

ТАТАРСТАН

Казань, «Таис»,
сеть магазинов «Дом книги»
тел.: (843) 272-34-55
e-mail: tais@bancorp.ru

УРАЛ

Екатеринбург, ООО «Дом книги»
ул. Антона Валека, д. 12
тел./факс: (343) 358-18-98, 358-14-84
e-mail: domknigi@k66.ru

Екатеринбург, ТЦ «Люмна»
ул. Студенческая, д. 1в
тел./факс: (343) 228-10-70
e-mail: igm@lumna.ru
http://www.lumna.ru

Челябинск, ООО «ИнтерСервис ЛТД»
ул. Артиллерийская, д. 124
тел.: (351) 247-74-03, 247-74-09,
247-74-16
e-mail: zakup@intser.ru
http://www.fkniga.ru, www.intser.ru