

# БОЛЬШАЯ КНИГА

# CSS3

Дэвид Макфарланд

3-Е  
ИЗДАНИЕ



O'REILLY®

 ПИТЕР®

# CSS3

Third Edition



The book that should have been in the box®

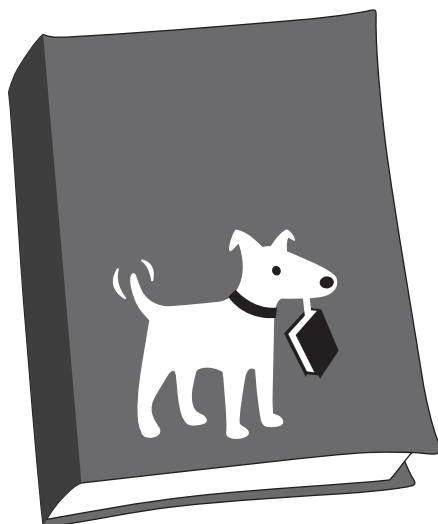
David Sawyer McFarland

O'REILLY®

Beijing | Cambridge | Farnham | Köln | Sebastopol | Tokyo

БОЛЬШАЯ КНИГА

# CSS3



Дэвид Макфарланд



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

2014

**Д. Макфарланд**  
**Большая книга CSS3**  
**3-е издание**

Серия «Бестселлеры O'Reilly»

Перевел с английского *Н. Вильчинский*

Заведующий редакцией	<i>Д. Виницкий</i>
Ведущий редактор	<i>Е. Каляева</i>
Научный редактор	<i>Н. Вильчинский</i>
Художник	<i>Л. Адуевская</i>
Корректор	<i>Е. Павлович</i>
Верстка	<i>О. Богданович</i>

ББК 32.988.02-018

УДК 004.738.8

**Макфарланд Д.**

M17 Большая книга CSS3. 3-е изд. — СПб.: Питер, 2014. — 608 с.: ил. — (Серия «Бестселлеры O'Reilly»).

ISBN 978-5-496-00428-2

С помощью технологии CSS3 (каскадные таблицы стилей) можно создавать уникальные, современные оформления веб-сайтов. Но даже самый опытный веб-разработчик может не знать всех приемов применения CSS3. Прочитав в этой книге множество практических примеров, а также советов, вы перейдете на новый уровень создания сайтов с помощью HTML и CSS. Вы узнаете, как разрабатывать веб-страницы, которые одинаково быстро работают и одинаково выглядят как на ПК, так и на смартфонах и планшетах.

В книге рассмотрены следующие темы: написание HTML5-, CSS3-тегов, которые распознаются во всех браузерах; форматирование текста, добавление на страницы навигации; создание таблиц и форм; приемы веб-дизайна для создания уникального оформления сайтов; создание сайтов для любых устройств с помощью адаптивного дизайна.

**12+** (В соответствии с Федеральным законом от 29 декабря 2010 г. № 436-ФЗ.)

ISBN 978-1449325947 англ.

Copyright © 2013 Sawyer McFarland Media, Inc. All rights reserved.  
Authorized Russian translation of the English edition of CSS3: The Missing Manual, 3rd Edition (ISBN 9781449325947). This translation is published and sold by permission of O'Reilly Media, Inc., which owns or controls all rights to publish and sell the same © Перевод на русский язык ООО Издательство «Питер», 2014  
© Издание на русском языке, оформление ООО Издательство «Питер», 2014

ISBN 978-5-496-00428-2

Права на издание получены по соглашению с O'Reilly. 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.

ООО «Питер Пресс», 192102, Санкт-Петербург, ул. Андреевская (д. Волкова), 3, литер А, пом. 7Н.  
Налоговая льгота — общероссийский классификатор продукции ОК 005-93, том 2; 95 3005 — литература учебная.

Подписано в печать 06.11.13. Формат 70×100/16. Усл. п. л. 49,020. Тираж 2700. Заказ

Отпечатано в полном соответствии с качеством предоставленных издательством материалов  
в ГППО «Псковская областная типография». 180004, Псков, ул. Ротная, 34.



# Краткое содержание

Об авторе . . . . .	13
О творческой команде . . . . .	14
Благодарности . . . . .	15
Введение . . . . .	16

## Часть 1. Основы CSS

Глава 1. HTML для CSS. . . . .	30
Глава 2. Создание стилей и таблиц стилей. . . . .	49
Глава 3. Селекторы: определение элементов стилизации. . . . .	69
Глава 4. Механизм наследования стилей . . . . .	106
Глава 5. Управление сложной структурой стилей: каскадность . . . . .	116

## Часть 2. Применение CSS

Глава 6. Форматирование текста . . . . .	138
Глава 7. Поля, отступы, границы . . . . .	205
Глава 8. Добавление графики на веб-страницы . . . . .	251
Глава 9. Приводим в порядок навигацию сайта . . . . .	306
Глава 10. Осуществление преобразований, переходов и анимации с помощью CSS . . . . .	348
Глава 11. Форматирование таблиц и форм. . . . .	387

## **Часть 3. Макет страницы**

<b>Глава 12.</b> Введение в разметку CSS . . . . .	414
<b>Глава 13.</b> Разметка страницы на основе плавающих элементов . . . . .	427
<b>Глава 14.</b> Адаптивный веб-дизайн . . . . .	464
<b>Глава 15.</b> Позиционирование элементов на веб-странице . . . . .	499

## **Часть 4. CSS для продвинутых**

<b>Глава 16.</b> CSS для распечатываемых веб-страниц . . . . .	527
<b>Глава 17.</b> Совершенствуем навыки работы с CSS . . . . .	546

## **Приложения**

<b>Приложение 1.</b> Справочник свойств CSS . . . . .	569
<b>Приложение 2.</b> Информационные ресурсы, касающиеся CSS . . . . .	603

# Оглавление

<b>Об авторе</b> .....	13
<b>О творческой команде</b> .....	14
<b>Благодарности</b> .....	15
<b>Введение</b> .....	16
Как работает CSS .....	16
Преимущества CSS .....	17
Что необходимо знать .....	17
HTML: структура языка .....	18
Типы документов .....	18
Как работают HTML-теги .....	19
Немного слов о XHTML .....	20
HTML5 — новый виток эволюции .....	21
Программное обеспечение, используемое для CSS .....	21
Об этой книге .....	23
Основные разделы книги .....	24
Основная терминология .....	25
О путях к папкам и командах меню .....	26
Соглашения, принятые в данной книге .....	26
Интернет-ресурсы .....	27
<b>От издательства</b> .....	28

## Часть 1. Основы CSS

<b>Глава 1. HTML для CSS</b> .....	30
HTML: прошлое и настоящее .....	30
Написание HTML-кода для CSS .....	33
Осмысление тегов <div> и <span> .....	35

Дополнительные теги в HTML5. . . . .	36
Составление своего представления о макете страницы . . . . .	38
Важность doctype. . . . .	44
Обеспечение использования самой последней версии Internet Explorer. . . . .	46
<b>Глава 2. Создание стилей и таблиц стилей . . . . .</b>	<b>49</b>
Что такое стиль . . . . .	49
Понимание таблиц стилей . . . . .	52
Внутренние таблицы стилей. . . . .	54
Внешние таблицы стилей. . . . .	55
Обучающий урок: создание первых стилей. . . . .	58
<b>Глава 3. Селекторы: определение элементов стилизации . . . . .</b>	<b>69</b>
Селекторы типов: дизайн страницы . . . . .	69
Селекторы классов: точное управление . . . . .	71
ID-селекторы: определение элементов веб-страниц . . . . .	74
Стилизация групп тегов. . . . .	76
Стилизация вложенных тегов. . . . .	77
Создание модулей . . . . .	81
Псевдоклассы и псевдоэлементы . . . . .	82
Селекторы дочерних элементов . . . . .	88
Селекторы типов дочерних элементов . . . . .	91
Селектор :not() . . . . .	94
Обучающий урок: примеры использования селекторов. . . . .	95
<b>Глава 4. Механизм наследования стилей . . . . .</b>	<b>106</b>
Что такое наследование? . . . . .	106
Упрощение таблиц стилей через наследование . . . . .	108
Ограничения наследования . . . . .	108
Обучающий урок: наследование . . . . .	110
<b>Глава 5. Управление сложной структурой стилей: каскадность . . . . .</b>	<b>116</b>
Каскадность стилей . . . . .	116
Особенности механизма каскадности: какие стили имеют преимущество . . . . .	121
Управление каскадностью . . . . .	124
Как избежать войн значимости . . . . .	127
Начинаем с чистого листа . . . . .	128
Обучающий урок: механизм каскадности в действии . . . . .	131

## Часть 2. Применение CSS

<b>Глава 6. Форматирование текста</b> . . . . .	138
Использование шрифтов . . . . .	138
Выбор обычного шрифта . . . . .	140
Использование веб-шрифтов . . . . .	143
Типы файлов шрифтов . . . . .	143
Правовые вопросы использования веб-шрифтов . . . . .	145
Поиск веб-шрифтов . . . . .	145
Создание нескольких форматов шрифтов. . . . .	146
Использование директивы @font-face. . . . .	149
Создание стилей с использованием веб-шрифтов. . . . .	151
Работа с полужирными и курсивными вариантами . . . . .	152
Простой способ добавления полужирного и курсивного вариантов . . . . .	153
Добавление полужирного и курсивного вариантов и поддержка Internet Explorer 8 . . . . .	155
Google web fonts . . . . .	158
Придание тексту цветового оформления . . . . .	166
Установка размера шрифта . . . . .	169
Форматирование символов и слов . . . . .	175
Добавление текстовых теней . . . . .	178
Форматирование абзацев текста . . . . .	180
Форматирование первой буквы, первой строки абзаца . . . . .	185
Стилизация списков . . . . .	187
Обучающий урок: форматирование текста в действии . . . . .	192
<b>Глава 7. Поля, отступы, границы</b> . . . . .	205
Понятие блочной модели . . . . .	205
Управление размерами полей и отступов . . . . .	207
Добавление границ . . . . .	215
Установка цвета фона . . . . .	219
Создание скругленных углов . . . . .	220
Добавление теней . . . . .	223
Определение параметров высоты и ширины. . . . .	226

Переопределение ширины блока с помощью box-sizing . . . . .	229
Задание максимальных и минимальных значений высоты и ширины. . . . .	233
Управление обтеканием содержимого плавающих элементов . . . . .	234
Обучающий урок: поля, фоновые параметры, границы. . . . .	239
Двигаемся дальше. . . . .	250
<b>Глава 8. Добавление графики на веб-страницы . . . . .</b>	<b>251</b>
CSS и тег <img> . . . . .	251
Добавление фоновых изображений . . . . .	252
Управление повтором фоновых изображений. . . . .	257
Позиционирование фоновых изображений . . . . .	258
Определение начальной позиции фонового изображения и порядка его отсечения . . . . .	265
Масштабирование фоновых изображений . . . . .	266
Сокращенный вариант свойства background . . . . .	268
Использование нескольких фоновых изображений . . . . .	270
Использование градиентных фонов . . . . .	273
Линейные градиенты . . . . .	273
Цветовые опорные точки . . . . .	276
Префиксы производителей и поддержка Internet Explorer . . . . .	278
Повторяющиеся линейные градиенты . . . . .	279
Радиальные градиенты . . . . .	281
Повторяющиеся радиальные градиенты . . . . .	283
Применение градиентов, создаваемых с помощью Colorzilla . . . . .	283
Обучающий урок 1: совершенствуем изображения . . . . .	286
Обучающий урок 2: создание фотогалереи . . . . .	291
Обучающий урок 3: использование фоновых изображений. . . . .	297
Получение свитка, появляющегося в Internet Explorer 8 . . . . .	304
<b>Глава 9. Приводим в порядок навигацию сайта . . . . .</b>	<b>306</b>
Выборка стилизуемых ссылок . . . . .	306
Стилизация ссылок . . . . .	310
Создание панелей навигации . . . . .	318
CSS-стиль для предварительной загрузки ролловеров . . . . .	327
Стилизация отдельных видов ссылок . . . . .	329
Обучающий урок 1: стилизация ссылок . . . . .	331
Обучающий урок 2: создание панели навигации . . . . .	338

<b>Глава 10.</b> Осуществление преобразований, переходов и анимации с помощью CSS . . . . .	348
Преобразования . . . . .	348
Переходы . . . . .	358
Анимация . . . . .	366
Обучающий урок . . . . .	379
<b>Глава 11.</b> Форматирование таблиц и форм . . . . .	387
Правильное использование таблиц . . . . .	387
Создание стилей для таблиц . . . . .	390
Создание стилей для форм . . . . .	396
Обучающий урок 1: создание стилей для таблиц . . . . .	402
Обучающий урок 2: создание стилей для форм . . . . .	407

## Часть 3. Макет страницы

<b>Глава 12.</b> Введение в разметку CSS. . . . .	414
Типы разметок веб-страницы . . . . .	414
Как работает CSS-разметка . . . . .	416
Стратегии разметки . . . . .	422
<b>Глава 13.</b> Разметка страницы на основе плавающих элементов . . . . .	427
Использование плавающих элементов в разметках. . . . .	430
Решение проблем плавающих элементов . . . . .	436
Обучающий урок. Многоколоночная разметка . . . . .	451
<b>Глава 14.</b> Адаптивный веб-дизайн. . . . .	464
Основы адаптивного веб-дизайна . . . . .	464
Настройка веб-страницы для RWD . . . . .	466
Медиазапросы . . . . .	467
Гибкие сетки . . . . .	476
Изменчивые изображения . . . . .	482
Обучающий урок по адаптивному веб-дизайну. . . . .	486
<b>Глава 15.</b> Позиционирование элементов на веб-странице. . . . .	499
Как работают свойства позиционирования. . . . .	499
Эффективные стратегии позиционирования. . . . .	513
Обучающий урок: позиционирование элементов страницы. . . . .	518

## Часть 4. CSS для продвинутых

<b>Глава 16.</b> CSS для распечатываемых веб-страниц . . . . .	527
Как работают аппаратно-зависимые таблицы стилей . . . . .	527
Как добавлять аппаратно-зависимые таблицы стилей . . . . .	530
Создание таблиц стилей для печати. . . . .	531
Обучающий урок: создание таблицы стилей для печати. . . . .	539
<b>Глава 17.</b> Совершенствуем навыки работы с CSS. . . . .	546
Добавление комментариев. . . . .	546
Организация стилей и таблиц стилей. . . . .	547
Устранение столкновений стилей в браузере . . . . .	555
Использование селекторов потомков . . . . .	559
Различный код CSS для Internet Explorer . . . . .	564

## Приложения

<b>Приложение 1.</b> Справочник свойств CSS . . . . .	569
Значения CSS . . . . .	569
Свойства текста. . . . .	573
Свойства списков. . . . .	578
Отступы, границы и поля. . . . .	579
Фоны. . . . .	584
Свойства разметки страницы . . . . .	587
Свойства анимации, преобразований и переходов . . . . .	593
Свойства таблицы . . . . .	598
Различные свойства. . . . .	600
<b>Приложение 2.</b> Информационные ресурсы, касающиеся CSS . .	603
Справочники . . . . .	603
Справочная информация по CSS . . . . .	604
Подсказки, приемы и советы по CSS. . . . .	604
CSS-навигация. . . . .	605
Разметка CSS. . . . .	606
Выставочные сайты . . . . .	607
Программное обеспечение для CSS . . . . .	607



# Об авторе



**Дэвид Соьер Макфарланд** — президент Sawyer McFarland Media, Inc., компании по обучению и разработке интернет-приложений в Портленде, штат Орегон (США). Он создает сайты с 1995 года: именно тогда он создал 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. Дэвид написал статьи о Сети для журналов *Practical Web Design*, *MX Developer's Journal* и *Macworld*, а также для портала CreativePro.com.

С Дэвидом вы можете связаться по электронной почте: [missing@sawmac.com](mailto:missing@sawmac.com) (если же вам нужна техническая помощь, то обращайтесь к ресурсам, указанным в приложении 2).

# О творческой команде

**Нэн Барбер** (редактор) начала работать с серией Missing Manual еще в прошлом веке. Живет в Массачусетсе вместе со своим мужем. Ее электронный адрес: [nanbarber@oreilly.com](mailto:nanbarber@oreilly.com).

**Холли Байер** (выпускающий редактор) живет в Старом Кембридже, штат Массачусетс, любит готовить, мастерить разные вещицы и увлекается современным дизайном в духе середины прошлого столетия. Ее электронный адрес: [holly@oreilly.com](mailto:holly@oreilly.com).

**Нэнси Рэйнхарт** (корректор) живет на Среднем Западе, проводя летние выходные на озере, плавая на лодке, купаясь и жадно читая книги. Нэн не только внештатный литературный редактор и корректор, но и автор любовных романов. С ее работами можно ознакомиться на сайте [www.nanreinhardt.com](http://www.nanreinhardt.com). Ее электронный адрес: [nanleigh1@gmail.com](mailto:nanleigh1@gmail.com).

**Дэниел Дж. Куинн** (технический редактор) внештатный веб-дизайнер, имеющий веб-сайт [DQuinn.net](http://DQuinn.net). Последние пять лет работал старшим специалистом по пользовательскому интерфейсу в отмеченном разными наградами агентстве по созданию цифровых продуктов Genuine Interactive. Специализировался на WordPress и стратегии контента для таких брендов, как Sam Adams, MassMutual и Children's Hospital Boston. В настоящее время Дэниел работает веб-разработчиком в отделе цифровых коммуникаций Гарвардского университета. Он периодически сотрудничает с командой местных дизайнеров. Его электронный адрес: [daniel@dquinn.net](mailto:daniel@dquinn.net).

**Джейсон Арнольд** (технический редактор) живет в Санта Роза, Калифорния с женой и тремя дочерьми. Работает техническим специалистом по телеметрическим системам в районной больнице Хелдсбурга и учит детей кэмпо-карате. В настоящее время он занят получением ученой степени в профессиональном колледже в Санта Роза. Свободное время он посвящает кэмпо-карате со своими дочерьми и всегда готов работать над дополнительными техническими проектами.

# Благодарности

Большое спасибо всем, кто помогал в работе над этой книгой, включая моих студентов, которые всегда оценивали технические издания с позиции новичков. Благодарю своих технических редакторов, Дэниела Куинна и Джейсона Арнольда, которые предостерегли меня от досадных ошибок, а также Зое Гилленуотер за ее бесценный вклад в первое издание книги. Кроме того, мы все в долгу перед веб-дизайнерами, которые стали новаторами, творчески используя CSS, и которые поделились своими наработками в сообществе, посвященном веб-дизайну.

Наконец, спасибо Дэвиду Погу, чей неуывадающий энтузиазм и выносливость ободряли меня; Нэн Барбер — за очистку моих рукописей и прочую помощь в написании этой книги; моей жене Сколле — за любовь и поддержку; моему сыну Грэхему, 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, чья улыбка всегда меня воодушевляла, и всей моей семье: маме, Дугу, Мари, Дэвиду, Марисе, Тессе, Филис, Лес, Дел, Патриции и Майку.

# Введение

Каскадные таблицы стилей, или Cascading Style Sheets (CSS), обеспечивают творческую свободу в разметке и дизайне веб-страниц. Пользуясь CSS, вы сможете украсить текст страниц привлекательными заголовками, буквицами, рамками, как в красочных гляцевых журналах. Можно точно разместить и позиционировать изображения, создать столбцы и баннеры, выделить ссылки динамическими эффектами. Кроме того, можно добиться постепенного появления и исчезновения элементов, перемещения объектов по странице или медленного изменения цвета кнопки при прохождении над ней указателя мыши.

Вы думаете, что все это довольно сложно? Напротив! Каскадные таблицы стилей как раз и предназначены для упрощения процесса оформления веб-страниц. Следующие несколько страниц будут посвящены основам CSS, а при изучении гл. 1 вы перейдете к непосредственной работе по созданию веб-страниц, улучшенных с помощью CSS.

## Как работает CSS

CSS работает с HTML, но не имеет к HTML никакого отношения. 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.

Например, вы могли бы использовать HTML, чтобы превратить фразу в заголовок, отделяя его от содержания страницы, но лучше применять CSS для форматирования заголовка, скажем, большим полужирным красным шрифтом с позиционированием на 50 пикселей от левого края окна. В CSS это форматирование текста включает в себя стиль — правило, описывающее внешний вид конкретной части веб-страницы. А таблица стилей (style sheet) является набором таких стилей.

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 на 50 пикселей.

Создав стиль один раз, можно применять его к текстовым фрагментам, изображениям, заголовкам и любым другим элементам страницы сколько угодно. Например, вы можете выбрать абзац текста и применить к нему стиль, тут же изменяющий размер, цвет и шрифт текста. Можно также сделать 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. 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 форматизирующие команды HTML. Например, тег `<font>`. Применяя CSS, вы можете уменьшить размер веб-страниц. В результате они будут стильно выглядеть и быстрее загружаться.
- Стилиевые 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.

## 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 продолжить 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 бесплатные обучающие сайты: HTML Dog ([www.htmldog.com/guides/htmlbeginner](http://www.htmldog.com/guides/htmlbeginner)) и W3Schools ([www.w3schools.com/html](http://www.w3schools.com/html)).

---

## 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>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Это заголовок веб-страницы</title>
  </head>
  <body>
    <p>А это абзац этой веб-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>` (со скобками), потом `head` (голова, заголовок), следом `body` (тело, тело документа), а в нем непосредственно содержимое веб-страницы. Все это завершается закрывающим тегом `</html>`.

## Типы документов

Все веб-страницы начинаются с объявления `doctype` — строки кода, определяющей, какой разновидностью HTML вы пользовались при написании страницы. В течение многих лет использовались два типа документов — HTML 4.01 и XHTML 1.0, и каждый из них имеет два стиля: строгий и промежуточный. Например, промежуточный `doctype` HTML 4.01 имеет следующий вид (другие `doctype`-объявления для HTML 4.01 и XHTML 1.0 выглядят примерно так же):

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

### ПРИМЕЧАНИЕ

Примеры всех разновидностей `doctype`-объявлений можно найти на веб-сайте [www.webstandards.org/learn/reference/templates](http://www.webstandards.org/learn/reference/templates).

Если посмотреть на код примера HTML-страницы, показанный в этом разделе, то вы увидите, что в нем используется краткая форма объявления `doctype`:

```
<!doctype html>
```

Это новое упрощенное объявление `doctype`, появившееся в HTML5. По сравнению с предшественниками в HTML5 заложена простота и рациональность использования. В этой книге используется объявление `doctype` из HTML5, поддерживаемое любым популярным браузером (даже старым Internet Explorer 6). И, поскольку HTML5 является будущим Интернета, далее применять старые `doctype`-объявления не имеет никакого смысла.

### ПРИМЕЧАНИЕ

То, что HTML `doctype`-объявление работает в старых браузерах, еще не значит, что эти браузеры понимают все теги или особенности HTML5. Например, Internet Explorer 8 или более ранние версии

не распознают новые теги HTML5. Чтобы внести в этих версиях в теги стиль с помощью CSS, нужно будет воспользоваться кодом JavaScript. Как вывести старые браузеры на современный уровень, будет показано далее.

---

Независимо от предпочитаемого типа документа, объявляемого с помощью doctype, важно, чтобы использовалось объявление хотя бы одного из них. Без этого ваши страницы будут выглядеть по-разному в зависимости от браузера, применяемого вашим посетителем, поскольку браузеры, не имеющие в качестве руководства объявления doctype, по-разному отображают информацию, которая стилизована с помощью CSS.

Каждое doctype-объявление требует от вас написания HTML-кода определенным образом. Например, тег для разбиения строк имеет в HTML 4.01 следующий вид:

```
<br>
```

Но в XHTML у этого тега следующий вид:

```
<br />
```

И здесь проявляется еще одно преимущество HTML5: он допускает применение любого из этих вариантов.

## 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. Теги являются разметочной (markup) частью гипертекстового языка разметки — Hypertext Markup Language.

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

На любой веб-странице обычно имеются как минимум следующие четыре элемента.

- Самая первая строка примера содержит объявление типа документа — doctype, рассмотренное в предыдущем разделе.
- Тег <html> требуется в начале веб-страницы и (с добавленным слешем) в конце: </html>. 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* (head) веб-страницы включает в себя ее название. Здесь также может содержаться другая информация, невидимая при просмотре веб-страницы, например описание страницы, которая предназначена для браузеров и поисковых машин. Раздел заголовка заключается в открывающий и закрывающий <head>-теги.

Кроме того, раздел заголовка может содержать информацию, которая используется браузером для оформления HTML, имеющегося на странице и для придания странице интерактивности. Вы увидите, что `<head>`-раздел может включать в себя код CSS (вроде того, который вы будете учиться создавать с помощью данной книги) или ссылку на другой файл, содержащий CSS-информацию.

- Тело (`body`) 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 тега `<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/">Щелкните здесь!</a>`).

Браузер знает, что при щелчке кнопкой мыши на ссылке со словами *Щелкните 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 URL-адрес) является его *значением*. В этом примере `http://www.missingmanuals.com` — значение атрибута `href`.

## Немного слов о XHTML

Как и всякая другая технология, язык HTML со временем претерпевал изменения. Хотя стандартный HTML справлялся со своей задачей достаточно неплохо, он всегда допускал некоторую небрежность при написании программ. Кроме всего прочего он позволял использовать строчные или прописные буквы в тегах или вообще их смесь (например, теги `<body>`, `<BODY>`, `<BoDy>` абсолютно идентичны), 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, смартфонов и другой техники, взаимодействующей с данными Интернета.

Кроме того, HTML не работает с одним из самых мощных языков Интернета: XML, или с расширяемым языком разметки — Extensible Markup Language.

Чтобы идти в ногу со временем, еще в 2000 году была введена улучшенная версия HTML, которую назвали XHTML, и зачастую ее можно найти на многих сайтах (фактически XHTML — это просто подогнанная под XML версия HTML). XHTML рассматривался (в 2000 году) как будущее развитие языка, но был вытеснен HTML5. Хотя браузеры по-прежнему разбираются в языке XHTML (и, наверное, еще долго будут его понимать), в этой книге он не рассматривается.



## HTML5 — НОВЫЙ ВИТОК ЭВОЛЮЦИИ

Когда W3C — группа, отвечающая за многие интернет-технологии — предложила XHTML как промежуточный шаг в переходе к XML в качестве основного языка Интернета, сообществу веб-разработчиков он был преподнесен как многообещающий язык. История показала, что это предсказание не оправдалось. Как оказалось, сложность перехода к XML удержала производителей браузеров от следования по пути XML, предписываемому World Wide Web Consortium. Взамен W3C закрыл рабочую группу по XHTML, отдав предпочтение HTML5 — новой версии HTML, отошедшей от XML и вернувшейся назад к HTML, но уже в его улучшенной версии. Некоторые браузеры, включая Google Chrome и Firefox, уже частично распознают HTML5.

HTML5 не является какой-то совершенно новой технологией. В отличие от XHTML, который был предназначен для стимулирования нового способа создания веб-страниц, HTML5 заботится о том, чтобы Интернет продолжал работать так же, как и прежде. Большинство основ HTML не претерпело изменений. Кроме того, в HTML5 были добавлены новые элементы, предназначенные для поддержки тех способов, с помощью которых веб-дизайнеры создают веб-сайты в настоящее время. Например, в HTML5 тег `<header>` может включать в себя содержимое, которое обычно встречается в верхней части страницы, например логотип и общие для всего сайта навигационные ссылки, новый тег `<nav>` содержит в себе набор ссылок, используемых для навигации по сайту, а тег `<footer>` размещает в себе все, что обычно помещается в нижней части страницы, например юридическую информацию, контакты по электронной почте и т. д.

Кроме того, в HTML5 добавлены новые теги, позволяющие делать на странице видео- и аудиовставки, новые теги формы, которые добавляют такие сложные элементы, как движки, появляющиеся окна выбора даты, а также встроенную браузерную поддержку проверки допустимости данных, введенных в форму (которая гарантирует правильное заполнение ваших форм посетителями). К сожалению, поддержка браузерами этих новых свойств страдает несовместимостью, и использовать новые теги без некоторых обходных путей довольно трудно.

Хотя HTML5 может быть еще не вполне готов для широкого применения, это не мешает уже сейчас пользоваться имеющимся в HTML5 doctype-объявлением и даже некоторыми основными свойствами этой новой версии языка. (Но Internet Explorer 8 и более ранние версии этого браузера нуждаются в помощи, о чем сообщается во врезке «Обходной прием. Как заставить Internet Explorer 8 понимать HTML5» раздела «Дополнительные теги в HTML5» гл. 1.) Код HTML5 будет встречаться во всей книге и особенно в следующей главе.

## Программное обеспечение, используемое 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.
- **TextWrangler** (Macintosh, [www.barebones.com/products/textwrangler](http://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 текстовых редакторов до мощных конструкторов.

- **EditPlus** (Windows, [www.editplus.com](http://www.editplus.com)) — недорогой текстовый редактор, который включает подсветку синтаксиса, поддержку FTP, автозавершение ввода и другие функции.
- **skEdit** (Macintosh, [www.skedit.com](http://www.skedit.com)) — дешевый редактор веб-страниц, полная поддержка FTP/SFTP, подсказка команд и другие полезные функции.
- **Coda2** (Mac; [www.panic.com/coda](http://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.
- **Sublime Text** (Mac; [www.sublimetext.com](http://www.sublimetext.com)) — относительно новый текстовый редактор. Любимый инструмент программистов, работающих на Mac.

- **Dreamweaver** (Macintosh и Windows, [www.macromedia.com/software/dreamweaver](http://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.
- **Expression Web 2** (Windows, [www.microsoft.com/expression](http://www.microsoft.com/expression)) — программа Microsoft для веб-дизайна. Она 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/ХТМЛ и CSS. Вам достаточно изучить всего одну из них для создания веб-страниц. Если у вас уже есть любимый редактор HTML/ХТМЛ, которым вы постоянно пользуетесь, но средства которого не позволяют редактировать код CSS, то можете найти CSS-ориентированный редактор. Их список приведен в приложении 2.

## 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/css3-transform/](http://www.w3.org/TR/css3-transform/), чтобы осознать, может ли это быть понятно обычному человеку.

Люди, приступающие к изучению CSS, как правило, не знают, с чего начать. А тонкости, имеющиеся в CSS, могут сбить с толку даже маститых веб-профессионалов. 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 начинается в конце гл. 1. Материал, содержащийся в этих главах, написан для тех, кто уже немного освоился в данной области и имеет средний уровень знаний. Если же вы плохо знакомы с созданием веб-страниц, то для лучшего понимания освещаемой темы должны ознакомиться с текстом врезок «Информация для новичков». С другой стороны, если у вас имеется большой опыт создания веб-страниц, то не обращайтесь 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 iPod, так и создатели CSS постоянно выпускают новые версии своих программных продуктов. CSS 1, появи-

вшийся в 1996 году, стал основой для дальнейшего совершенствования языка каскадных таблиц стилей. В этой самой первой версии языка CSS были введены

## ИНФОРМАЦИЯ ДЛЯ НОВИЧКОВ

основные понятия и команды: структура таблиц стилей, понятие идентификатора элемента (см. гл. 3) и большая часть свойств CSS.

В версии CSS 2 были введены 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.

В версию CSS 2.1 были включены несколько дополнений и изменений. Она не содержит радикальных изменений по сравнению с CSS 2, и большинство браузеров давно адаптированы под эту версию.

В последнее время появилось множество публикаций, посвященных CSS3 (а также его "двоюродному брату"

HTML5). В отличие от предыдущих версий CSS, CSS3 не является единым стандартом. По мере возрастания сложности CSS консорциум W3C разбил CSS на несколько отдельных модулей — модуль Selectors, модуль Values and Units, модуль Box Alignment и т. д. Поскольку каждый модуль может разрабатываться независимо от других модулей, какого-то единого стандарта под названием CSS3 не существует. Фактически работа над уровнем 3 модуля Selectors уже завершена, и начата работа над уровнем 4.

Иными словами то, что известно как CSS3, на самом деле является разобленной коллекцией различных модулей на разных стадиях завершения. Часть новейших модулей уже включены производителями браузеров в их продукты, а другие модули в любом отдельно взятом браузере могут не иметь никакой поддержки. В будущем какого-то CSS4 не будет, будут только новые версии различных модулей, каждый из которых будет находиться на своем уровне разработки.

Поэтому в данной книге рассматривается ядро CSS 2.1 (которое просто было перенесено в различные модули CSS3), а также наиболее впечатляющие, популярные и широко поддерживаемые новые свойства CSS.

## Основные разделы книги




Книга разделена на четыре части. Кроме того, издание содержит два приложения.

- **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. Обучающие уроки закрепят вводимые в частях основные понятия и позволят вам почувствовать эффективность использования CSS.
- **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. Вы также узнаете о том, как с помощью CSS создавать простую анимацию. 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: с управлением размещением элементов на странице. Вы познакомитесь со схемами дизайна (размещение разделов в два и три столбца) и узнаете, как добавить боковые панели. Вы также узнаете о двух основных методах позиционирования элементов на странице: абсолютном и относительном. Вы научитесь создавать веб-сайты, адаптируемые для лучшего восприятия в браузерах настольных систем, планшетных компьютеров и мобильных устройств.
- **Часть 4. CSS для продвинутых.** Научит вас создавать совершенные веб-страницы, которые выглядят одинаково хорошо как на экране монитора, так и в распечатанном виде. Кроме того, здесь описаны методы еще более эффективного применения CSS.
- **Приложения.** Представляют два набора информационных ресурсов. Справочник свойств CSS описывает каждое свойство в отдельности в простой и доступной для понимания форме, чтобы вы могли быстро узнать о полезных свойствах CSS, которые раньше вам могли не попадаться, или быстро освежить в памяти уже знакомые свойства. В последнем приложении дается описание инструментов и средств для создания и применения каскадных таблиц стилей.

## Основная терминология

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

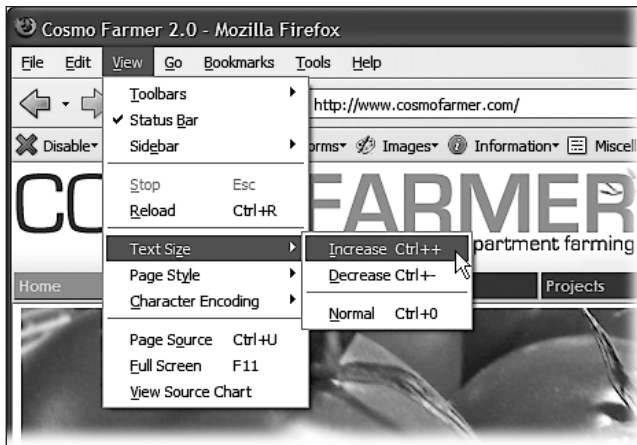
- **Щелчок кнопкой мыши.** Пользуясь манипулятором типа “мышь” вашего компьютера, вы можете выполнить три действия. Щелчок кнопкой мыши означает, что нужно навести указатель мыши на какой-либо объект на экране монитора, а затем, не перемещая указатель, нажать и отпустить левую кнопку мыши. Двойной щелчок 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 щелкаете кнопкой мыши, предварительно нажав клавишу  или Ctrl (две клавиши Ctrl находятся по обе стороны от клавиши Пробел).
- **Меню.** Это строка с кнопками, находящаяся в верхней части 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 соответствующем слове (пункте меню); в результате раскроется перечень команд меню.
- **Комбинации клавиш.** Убирая руки с клавиатуры для перемещения указателя мыши, вы теряете время и можете сбиться с намеченного пути. Поэтому многие опытные компьютерные специалисты вместо команд меню используют везде, где только можно, комбинации клавиш. Когда для быстрого вызова той или иной функции предлагается комбинация клавиш, подобная Ctrl+S (-S)

(эта комбинация сохраняет изменения, внесенные в текущий документ), значит, вам нужно удерживать в нажатом состоянии клавишу **Ctrl** или клавишу **⌘** и, пока она нажата, нажать клавишу **S**, а затем отпустить обе клавиши.

## О путях к папкам и командах меню

В этой книге вам будет попадаться текст такого рода: «Откройте папку **System\Library\Fonts**». Это сокращение более пространной инструкции, предписывающей открытие трех последовательно вложенных друг в друга папок, которая должна звучать следующим образом: «На жестком диске найдите папку под названием **System**. Откройте ее. В окне системной папки есть папка под названием **Library**; дважды щелкните кнопкой мыши, чтобы открыть ее. В этой папке находится папка с названием **Fonts**. Дважды щелкните кнопкой мыши и также откройте ее».

Кроме того, сокращенные команды со стрелками помогут вам открыть нужный пункт меню, как показано на рис. 0.1.



**Рис. 0.1.** Запись со стрелками позволяет упростить инструкции по использованию меню

Например, запись **View > Text Size > Increase** (Вид > Размер шрифта > Увеличить) является более кратким способом дать указание, выполняемое на рис. 0.1: «В пункте меню **View** (Вид) выберите подпункт **Text Size** (Размер шрифта), а затем выберите подпункт **Increase** (Увеличить)». Кстати, когда после команды меню вы видите комбинацию клавиш, подобную показанной здесь **Ctrl++**, это означает, что вы можете нажать эту комбинацию для быстрого вызова данной команды.

## Соглашения, принятые в данной книге

Здесь приводится список соглашений, принятых 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 внимательными. 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.

---

## Интернет-ресурсы

Эта книга создана для того, чтобы вы смогли работать над веб-проектами быстрее и профессиональнее. Именно поэтому большое количество материалов, рассмотренных в этом издании, выложено в Интернете.

По ходу чтения книги вам будут попадаться живые примеры — *пошаговые обучающие уроки*, которые вы сможете выполнять, используя исходные данные (графические образы и незаконченные веб-страницы), загруженные с сайта [www.sawmac.com/css3/](http://www.sawmac.com/css3/). От простого прочтения этих пошаговых уроков, покачиваясь в гамаке, толку будет мало. Эти уроки нужно прорабатывать на компьютере. Так вы сможете понять, каким образом профессиональные дизайнеры создают веб-страницы.

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

# От издательства

Ваши замечания, предложения и вопросы отправляйте по адресу электронной почты [vinitski@minsk.piter.com](mailto:vinitski@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.** HTML для CSS

**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 HTML для CSS

Для получения от CSS наибольшей отдачи HTML-код должен предоставить однородную, хорошо скроенную основу. Из этой главы вы узнаете, как создавать HTML-код, более приспособленный под нужды CSS. Спешу обрадовать, что повсеместное использование на веб-сайте CSS *существенно облегчает* создание кода HTML. Вам больше не нужно будет 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, но стилевое оформление верхней страницы создано полностью с использованием CSS, а для нижней применялся только 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). Создание веб-страниц с использованием только HTML требует для достижения тех же визуальных эффектов намного большего объема кода: 213 строк HTML-кода против 71 строки для версии с применением CSS.

## HTML: прошлое и настоящее

Как говорилось во введении, HTML на сегодняшний день является основой для написания любой веб-страницы во Всемирной паутине. При использовании CSS создание кода на языке HTML коренным образом меняется. Попрощайтесь с перепрофилированием неуклюжих HTML-тегов просто для того, чтобы добиться тех или иных визуальных эффектов. О некоторых HTML-тегах и атрибутах, таких как, например, устаревший тег `<font>`, вообще можно забыть.

## Прошлое HTML: лишь бы все выглядело хорошо

Когда группа ученых создала Интернет, который первоначально предназначался для совместного использования и поиска технической документации, никто не

обращался к графическим дизайнерам. HTML был нужен ученым только для структурирования информации с целью ее более простого восприятия. Возьмем для примера тег `<h1>`. Он выделяет в тексте важный заголовок, а тег `<h2>` создает подзаголовок с меньшим размером шрифта. Другой широко используемый тег — `<ol>` (ordered list — «упорядоченный список») — создает нумерованный список для перечислений типа: «Десять причин не играть с медузой».

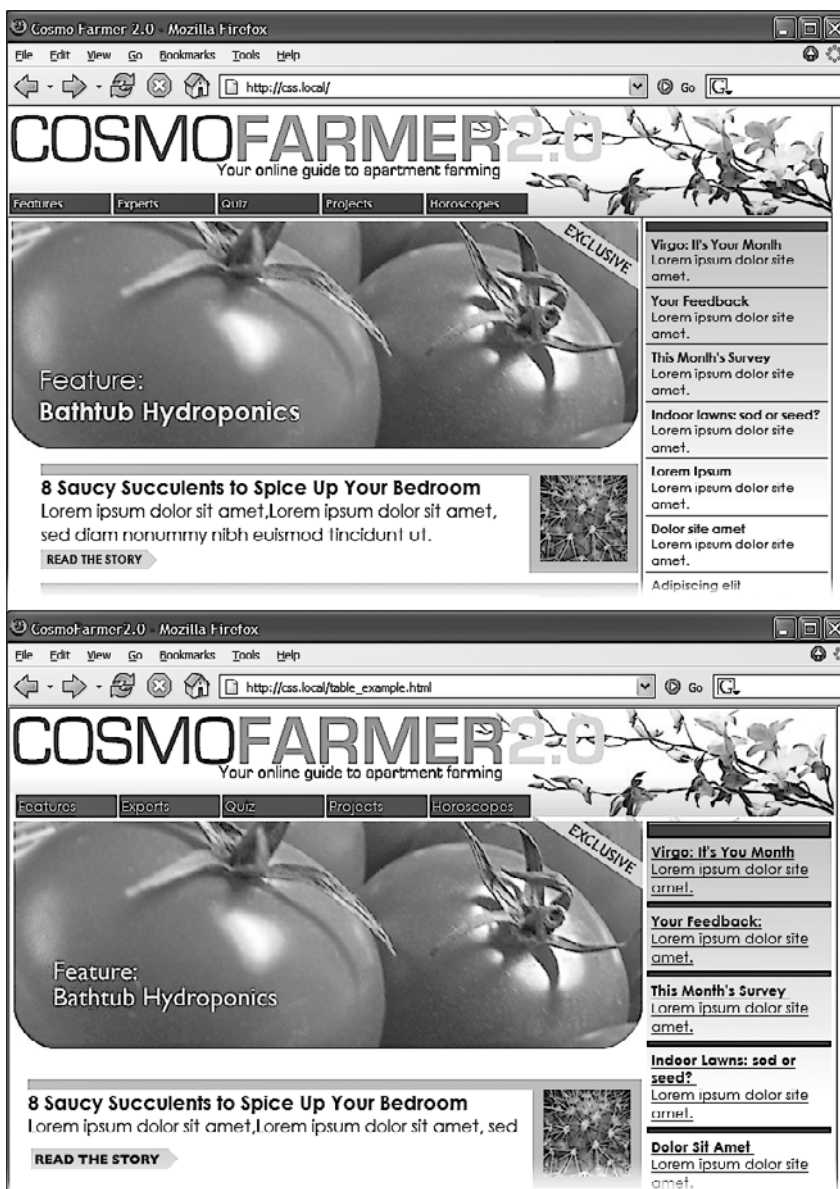


Рис. 1.1. Веб-дизайн с применением CSS упрощает написание HTML-кода

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, начертание 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, задействовали 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: подготовка рабочего материала для CSS**

Независимо от того, что представляет собой веб-страница — календарь, 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 дает возможность HTML вернуться к исполнению своей прямой обязанности — к 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 структу-

рирования. 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 значение 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

---

Руководство по HTML можно найти на сайте [www.w3schools.com/html/html\\_intro.asp](http://www.w3schools.com/html/html_intro.asp). Краткий список всех HTML-тегов приведен в справочнике по HTML Sitepoint.com по адресу <http://reference.sitepoint.com/html>.

---

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 (нумерованных) 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.

option is in the Illustrator Native Format Options dialog box, which appears when saving an Adobe Illustrator file using the Save As command.

This is an Adobe® Illustrator® File that was saved without PDF Content.  
To Place or open this file in other applications, it should be re-saved from Adobe Illustrator with the "Create PDF Compatible File" option turned on. This option is in the Illustrator Native Format Options dialog box, which appears when saving an Adobe Illustrator file using the Save As command.

**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 HTML-тега, о которых не следует забывать

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 `<pre>` («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.

## ИНФОРМАЦИЯ ДЛЯ НОВИЧКОВ

### 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 и занять верхние строки в списках таких поисковиков, как Google, Yahoo! и Bing. 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 форматированных тегов (например, `<font>`) и множества таблиц для создания дизайна страницы влияет на работу поискового сервера. Некоторые поисковые системы прекращают чтение HTML-кода страницы после про-

смотра 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 <http://support.google.com/webmasters/bin/answer.py?hl=en&answer=35769>.

## Осмысление тегов `<div>` и `<span>`

Теги `<div>` и `<span>` 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 `<div>` (предназначен для *деления на фрагменты*) определяет любой отдельный блок содержимого, как, 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 `<div>`. Тег `<div>` — замечательное средство разбивки веб-страницы на такие логические фрагменты, как баннер, нижний колонтитул, боковая панель и т. д. Впоследствии при использовании CSS вы сможете позиционировать любой из этих фрагментов в выбранное место веб-страницы, создавая сложную схему разметки (см. часть 3).

Тег `<span>` применим к *внутренним* (inline) элементам страницы, то есть к словам, фразам, находящимся 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 `<strong>` (чтобы выделить слово в абзаце полужирным шрифтом). Можно применять тег `<span>`, 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>
```

Разговор об этих тегах не ограничится этим кратким введением. Они часто используются на тех веб-страницах, где широко применяется CSS, и эта книга поможет вам научиться задействовать их в комбинации с CSS для получения творческого контроля над вашими веб-страницами.

## Дополнительные теги в HTML5

Тег `<div>` имеет довольно общий характер — это просто элемент на уровне блока, используемый для разбиения страницы на разделы. Одна из целей HTML5 заключается в предоставлении в распоряжение разработчиков широкого выбора из других, семантически более осмысленных тегов. Придание HTML большей семантической просто означает применение тегов, точно описывающих свое содержимое. Как уже упоминалось в данном разделе, вы можете воспользоваться тегом `<h1>` (заголовок 1), помещая в него текст, описывающий основное содержимое страницы. По аналогии с этим тег `<code>` четко дает понять, информация какого сорта в него помещена, — это программный код.

HTML5 включает в себя множество различных тегов, чьи имена отражают тип их содержимого, и эти теги могут использоваться вместо тега `<div>`. Тег `<article>` (статья), к примеру, применяется для обозначения раздела страницы, содержащего завершенную, независимую композицию. Иными словами, «статью», такую как запись блога, товар в интернет-магазине или просто основной текст страницы. Точно так же, тег `<header>` (верхний колонтитул) является признаком верхнего колонтитула или баннера, верхней части страницы, которая обычно содержит логотип, навигацию, относящуюся ко всему сайту, заголовок страницы с подзаголовком и т. д.

### ПРИМЕЧАНИЕ

---

Дополнительные сведения об HTML-тегах можно найти на сайте HTML5 Doctor (<http://html5doctor.com>) и на сайте [www.w3schools.com/html5/html5\\_intro.asp](http://www.w3schools.com/html5/html5_intro.asp).

---

Многие новые теги HTML5 предназначены для расширения возможностей обычного тега `<div>`. Для структурирования содержимого страницы часто используются и другие теги HTML5.

- Тег `<section>` (раздел) содержит группировку взаимосвязанного содержимого, например главу книги. Так, вы можете разбить содержимое главной страницы



на три раздела: вводную информацию о сайте, контактную информацию и самые последние новости.

- Тег `<aside>` (отступление) предназначен для обозначения содержимого, относящегося к окружающему это тег содержимому. Например, пометки на полях в печатном журнале.
- Тег `<footer>` (нижний колонтитул) содержит информацию, которая обычно помещается в нижнем колонтитуле страницы, например сведения об авторских правах, другую правовую информацию, некоторые ссылки для навигации по сайту и т. д. Но на количество тегов `<footer>` на одной странице ограничений не накладывается, вы можете, допустим, поместить нижний колонтитул внутри тега `<article>`, чтобы хранить в нем информацию, относящуюся к статье, например сноски, ссылки или выписки.
- Тег `<nav>` (навигация) используется для обозначения содержимого в виде основных навигационных ссылок.
- Тег `<figure>` (рисунок) применяется для иллюстраций. Вы можете поместить в него тег `<img>`, а также еще один новый тег HTML5 — `<figcaption>`, использующийся для вывода пояснений к фотографии или иллюстрации, которая находится внутри тега `<figure>`.

#### СОВЕТ

Разобраться в том, какой из тегов HTML5 лучше использовать, то есть должен ли ваш текст быть статьей — `<article>` или разделом — `<section>`, бывает порой непросто. Удобная блок-схема, помогающая понять предназначение новых элементов разбиения информации, имеющихся в HTML5, приведена в PDF-файле, который можно загрузить с сайта HTML5doctor.com: <http://html5doctor.com/downloads/h5d-sectioning-flowchart.pdf>.

Существуют и другие элементы HTML5, и многие из них просто предоставляяют более описательную альтернативу тегу `<div>`. В этой книге используются как теги `<div>`, так и новые теги HTML5, помогающие придать веб-странице более выраженную организацию содержимого. Недостаток HTML5 заключается в том, что Internet Explorer 8 и более ранние версии этого браузера не распознают новые теги без посторонней помощи (см. следующую врезку).

Нужно также заметить, что, кроме чувства сопричастности к самым последним течениям в веб-дизайне, от использования некоторых из этих тегов HTML5 на самом деле нет никакой ощутимой пользы. Например, использование тега `<article>` просто для того, чтобы содержать в нем сводку новостей веб-страницы, не улучшит ее внешний вид. Если хотите, можете и впредь задействовать тег `<div>`, избегая применения элементов разбиения страницы на разделы, предлагаемых HTML5.

#### ОБХОДНОЙ ПРИЕМ

##### Как заставить Internet Explorer 8 понимать HTML5

HTML5 предоставляет в ваше распоряжение много новых тегов. От тегов, которые проясняют характер своего содержимого, например тега `<nav>`, и до те-

гов, предоставляющих дополнительные возможности, таких как тег `<video>`, который предназначен для вставки видео, и тег `<audio>`, предназначенный для

### ОБХОДНОЙ ПРИЕМ

вставки звука или музыки. По мере изучения HTML5, вы, возможно, начнете применять эти новые теги на своих веб-страницах.

К сожалению, Internet Explorer 8 и более ранние версии этого браузера не распознают эти новые теги и не будут реагировать на применяемый к ним код CSS. Действительно, если использовать HTML5 и просматривать веб-страницы в Internet Explorer 8, эта книга не принесет никакой пользы. Однако есть один способ заставить Internet Explorer понимать весь CSS-код, применяемый к тегам HTML5. Нужно просто поместить перед закрывающим тегом `</head>`, находящимся в верхней части HTML-файла, следующий код:

```
<!--[if lt IE 9]>
<script src="//html5shiv.googlecode.com/svn/trunk/html5.js"></script>
<![endif]-->
```

Этот несколько необычный фрагмент кода использует так называемый условный комментарий Internet Explorer (Internet Explorer conditional comment, IECC) для вставки кода JavaScript, который видим только для версий Internet Explorer, предшествующих 9-й. Иными словами, на этот код реагируют только Internet Explorer 6, 7 и 8, а все остальные браузеры (включая более новые версии Internet Explorer) его просто игнорируют. Данный код заставляет более ранние версии Internet Explorer загружать небольшую программу на JavaScript, заставляющую браузер распознавать теги HTML5, и применять к ним код CSS, который предназначен для этих тегов.

Данный код влияет только на то, как браузер выводит теги HTML5 на экран или на принтер, и не заставляет браузер понимать теги HTML5, которые на самом деле совершают какие-либо действия. Например, Internet Explorer 8 и более ранние версии браузера не понимают тег `<video>` и не могут проигрывать видео HTML5 (даже с добавленным кодом JavaScript).

## Составление своего представления о макете страницы

Если с целью обозначения основной темы страницы используется тег `<h1>`, а для добавления текстового абзаца — тег `<p>`, то вы в итоге имеете намерение придать содержимому страницы привлекательную раскладку. Когда при прочтении части 3 данной книги вы научитесь применять CSS для создания макета страницы, вам уже не нужно будет при написании кода HTML, имеющегося на странице, следить за дизайном.

Макет страницы можно представить себе в виде искусного расположения прямоугольных элементов (пример такого расположения показан на рис. 1.3). В конечном счете двухколоночный дизайн, состоящий из двух вертикальных столбцов текста, фактически представляет собой всего лишь два смежных прямоугольника. Верхний колонтитул, состоящий из логотипа, подзаголовка, поля поиска и элементов навигации по сайту, представляет собой нечто иное, как широкий прямоугольник, который занимает всю верхнюю часть окна браузера. Иными словами, если вы представите себе группировку и макет содержимого страницы, то вы должны увидеть прямоугольники, установленные друг на друга, следующие друг за другом и находящиеся друг под другом.

На рис. 1.3 приведен основной двухколоночный макет, включающий баннер (в верхней части), столбец основного содержимого (в левой части), боковую панель (в правой части) и нижний колонтитул (в нижней части). Это основные структурные прямоугольники, составляющие макет данной страницы.



Рис. 1.3. Основной двухколоночный макет

В HTML эти прямоугольники, или структурные единицы, создаются с помощью тега `<div>`. Нужно, например, просто заключить HTML-теги, из которых составлена область баннера, в один `div`-контейнер, HTML-код, формирующий столбцы, — в другой `div`-контейнер и т. д. Если вы уже разбираетесь в HTML5, то можете создать дизайн, показанный на рис. 1.3, с помощью тега `<header>` для верхнего баннера, тега `<article>` для основного текста, тега `<aside>` или `<section>` для боковой панели, и тега `<footer>` — для нижнего колонтитула страницы. Иными словами, если вы собираетесь сгруппировать HTML-теги в каком-нибудь месте страницы, их нужно заключить в такие элементы разбиения на разделы, как `<div>`, `<article>`, `<section>` или `<aside>`.

## О каких элементах HTML следует забыть

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. Его единственная 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).
- **Не используйте теги `<b>` и `<i>`, для выделения текста.** CSS в состоянии сделать полужирным или курсивным любой текст, и нет никакой необходимости пользоваться специфическими формирующими тегами языка HTML. Чтобы *выделить* текст, применяйте тег `<strong>` (браузеры обычно отображают текст, выделенный этим тегом, как полужирный). Если вы хотите придать тексту чуть меньший акцент, то задействуйте для его выделения тег `<em>` (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. Обязательно запомните это.

---

- **Не пользуйтесь тегом `<table>` для макетирования страницы.** Применяйте его только с целью отображения табличной информации (например, электронных таблиц, списков, диаграмм). В части 3 этой книги вы узнаете, что CSS позволяет макетировать веб-страницы гораздо быстрее и с меньшим объемом кода, нежели при использовании тега `<table>`.
- **Избегайте применения неуклюжих атрибутов тега `<body>`,** всего лишь улучшающих внешний вид содержимого веб-страницы: `background`, `bgcolor`, `text`, `link`, `alink` и `vlink`, которые устанавливают цвета и изображения для страницы, текста, а также гиперссылок. CSS справляется с этим гораздо лучше (см. в гл. 7 и 8 CSS-аналоги этих атрибутов).
- **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 `<r>`; раньше разработчики шли сложными обходными путями, чтобы избежать этого интервала между абзацами, заменяя единственный тег `<r>` 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. раздел «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, — уста-

ревший 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» раздела «Форматирование абзацев текста» гл. 6), и установку границ (см. раздел «Добавление границ» гл. 7), и фоновые параметры (см. раздел «Установка цвета фона» гл. 7), и позиционирование изображений (см. раздел «Добавление фоновых изображений» гл. 8).

## 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, хотя HTML5 допускает больше послаблений. 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, даже правильный CSS-код может вместе с ошибочным HTML работать не так, как ожидалось. 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/> (рис. 1.4). Кроме того, можно 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 и CSS.

При нахождении ошибок на ваших веб-страницах W3C-валидатор сообщит о них. Используя браузер 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/>. Пользователи браузера Chrome могут загрузить расширение HTML Tidy для Chrome с веб-сайта <http://bit.ly/SCONRY>. Подобный инструмент для браузера Safari доступен на сайте [www.zappatic.net/safaritidy/](http://www.zappatic.net/safaritidy/).

HTML-валидатор W3C по адресу <http://validator.w3.org> позволяет быстро убедиться в правильности кода HTML-страницы. Можно нацелить валидатор на уже существующую в сети страницу, загрузить HTML-файл с вашего компьютера или просто вставить HTML-код веб-страницы в поле формы, а затем щелкнуть на кнопке Check (Проверка).

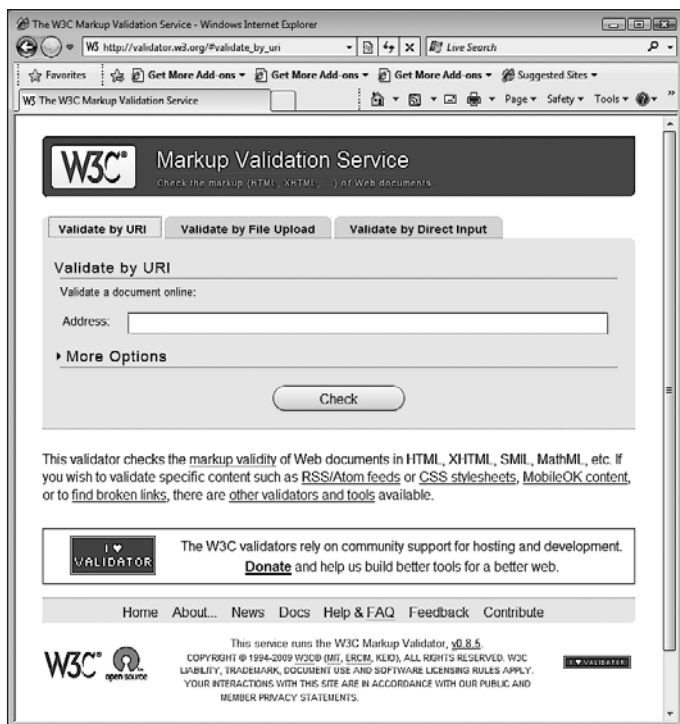


Рис. 1.4. HTML-валидатор от W3C позволит быстро проверить правильность кода веб-страницы

## Советы

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 начальных советов.

- 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, применяйте заголовок на уровень ниже, подзаголовок. Например, от заголовка уровня `<h2>` переходите к подзаголовку уровня `<h3>` (рис. 1.5). Лучше использовать заголовки по порядку и стараться не пропускать их номера, например, никогда не переходите от тега `<h2>` к тегу `<h5>`.
- Используйте тег `<p>` для абзацев текста.
- Применяйте маркированные списки (`<ul>`), если у вас есть перечень связанных элементов, таких как ссылки навигации, заголовки, подсказки и др.
- Пользуйтесь нумерованными списками (`<ol>`), чтобы определить ряд последовательно выполняемых операций или порядок набора элементов. В обучающих примерах этой книги (см. раздел «Обучающий урок: форматирование текста в действии» гл. 6) есть наглядный пример такого списка.





Рис. 1.5. Используйте теги заголовков: расположите их по тексту в порядке важности

- Чтобы создать словарь терминов и их определений, задействуйте тег `<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](http://www.w3schools.com/tags/tryit.asp?filename=tryhtml_list_definition).
- Если вы хотите включить цитату в виде отрывка текста с другого сайта, чье-либо высказывания и др., используйте тег `<blockquote>` для длинного контекста (высказываний), а тег `<q>` — для вставки краткой цитаты в более длинный абзац, например:
 

`<p>`Говорят, что Марк Твен как-то раз написал: `<q>`Самой холодной из всех проведенных мною зим было лето в Сан-Франциско`</q>`. К сожалению, на самом деле он никогда ничего похожего на эту знаменитую цитату не писал.`</p>`
- 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 `<div>` и `<span>`. Об их использовании будет рассказано в следующих главах.
- Не злоупотребляйте тегом `<div>`. Некоторые дизайнеры полагают, что `<div>` — это все, что им нужно, и при этом игнорируют теги, которые могут быть более уместны. Возьмем пример создания навигационной панели. Можно добавить

блок `<div>` на страницу и заполнить его кучей ссылок. Но ведь гораздо лучше использовать маркированный список: в конце концов, навигационная панель и сама является списком ссылок. Как уже ранее упоминалось, HTML5 предоставляет несколько новых тегов, которыми можно заменить тег `<div>`, например теги `<article>`, `<section>` и `<footer>`. Для панели навигации можно воспользоваться HTML5-тегом `<nav>`.

- Никогда не забывайте указывать закрывающие теги. Открывающий тег `<p>` требует соответствующего ему закрывающего тега `</p>`, как и любые другие теги, за исключением одиночных, например `<br/>` и `<img/>`.
- Проверяйте синтаксис своих веб-страниц с помощью W3C-валидатора (см. рис. 1.4 и врезку «Информация для новичков. Проверьте правильность кода веб-страниц»). Плохо написанный HTML-код, как и код с опечатками, вызовет множество непредсказуемых ошибок браузера.

## Важность doctype

HTML следует некоторым правилам, содержащимся в файле объявления типа документа (*Document Type Definition (DTD)*). DTD представляет собой текстовый файл, 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, включая то, что называется *объявлением типа документа* в начало веб-страницы. Это объявление типа документа находится в первой строке HTML-файла и определяет используемую вами версию HTML (например, HTML5 или HTML 4.01 Transitional). Если вы укажете doctype с ошибкой или опустите это объявление, то браузер переключится в другое состояние, называемое *режимом совместимости*.

Режим совместимости является попыткой производителей браузеров заставить свое программное обеспечение вести себя подобно устаревшим браузерам, выпущенным 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 режиме совместимости, 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 работы браузеров в режиме совместимости посетите сайты [www.quirksmode.org/index.html?css/quirksmode.html](http://www.quirksmode.org/index.html?css/quirksmode.html) и [https://developer.mozilla.org/en/Mozilla%2527s\\_Quirks\\_Mode](https://developer.mozilla.org/en/Mozilla%2527s_Quirks_Mode).

Указать правильный doctype достаточно просто. Нужно только знать используемую версию HTML. Если используется HTML5, все существенно упрощается. Объявление doctype приобретает простой вид:

```
<!doctype html>
```



Поместите его в верхнюю часть своего HTML-файла, и все будет в порядке. Если вы по-прежнему пользуетесь старыми версиями HTML или XHTML, например HTML 4.01 Transitional и XHTML 1.0 Transitional, объявление doctype будет иметь более запутанный вид.

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 имеет похожий вид. Кроме того, необходимо добавить кое-что еще в открывающий тег <html>. Эта строка 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">
```

Если все это обсуждение навеивает на вас тоску, упростите свою жизнь и воспользуйтесь doctype-объявлением, предлагаемым HTML5. Оно имеет краткую форму, легко запоминается и работает во всех браузерах. Это doctype-объявление можно использовать, даже если не применяются никакие новые теги HTML5.

#### ПРИМЕЧАНИЕ

Большинство визуальных средств создания веб-страниц, в число которых входят Dreamweaver и Expression Web, при создании новой веб-страницы добавляют doctype-объявление автоматически, а многие дружественные для HTML текстовые редакторы имеют для добавления doctype-объявлений специальные комбинации клавиш.

### ЧАСТО ЗАДАВАЕМЫЕ ВОПРОСЫ

#### Стоит ли волноваться насчет Internet Explorer 6, 7 или 8?

Считается, что Internet Explorer 6 уже отслужил свое и больше беспокоиться о нем не стоит. Но так ли это? А как насчет других версий Internet Explorer?

Если вы веб-дизайнер, то, наверное, на вашем компьютере уже стоят самые последние версии Internet Explorer, Firefox, Safari, Chrome или Opera. В прежних изданиях данной книги много говорилось об Internet Explorer 6 и даже целый раздел посвящался борьбе с изъянами Internet Explorer 6. Весь этот материал был удален, поскольку Internet Explorer 6 стремительно уходит в прошлое.

На август 2012 года для просмотра интернет-ресурсов этим устаревшим браузером пользовалось менее 0,6 % американцев, а в Великобритании количество таких пользователей составляло всего 1,1 %. Конкретная картина менялась в зависимости от опрашиваемой аудитории (например, на тот же август 2012-го на сайте <http://gs.statcounter.com> количество пользовате-

лей Internet Explorer 6 оценивалось на уровне 0,57 % а на сайте [www.iecountdown.com](http://www.iecountdown.com) — 6 %). Internet Explorer 6 по-прежнему довольно широко используется в Китае.

Но даже статистика, включающая географический регион, в котором находится аудитория вашего сайта, не дает достоверной картины относительно посетителей, использующих данный браузер. Если вы создаете сайт, нацеленный на разбирающихся в технике веб-дизайнеров, вполне вероятно, что ваш сайт практически никогда не будет просматриваться на Internet Explorer 6. Но если ваш сайт предназначен для жителей Китая, то вам, возможно, придется иметь дело с Internet Explorer 6. Лучше всего определить ту часть трафика, которая приходится на запросы от Internet Explorer 6 путем изучения регистрационных файлов веб-серверов или подписавшись на Google Analytics ([www.google.com/analytics](http://www.google.com/analytics)), чтобы можно было

### ЧАСТО ЗАДАВАЕМЫЕ ВОПРОСЫ

отслеживать браузеры ваших посетителей (а также другую важную информацию).

Для подавляющего большинства веб-дизайнерских проектов Internet Explorer 6 уже приказал долго жить. Кроме того, стремительно исчезает и Internet Explorer 7, чей рынок пользователей по всему миру оценивался на август 2012 года в 1,16 % ([http://gs.statcounter.com/#browser\\_version-ww-monthly-201108-201208](http://gs.statcounter.com/#browser_version-ww-monthly-201108-201208)).

А вот с Internet Explorer 8 по-прежнему нужно считаться. Windows XP все еще остается весьма популярной операционной системой, а компания Microsoft решила не переносить Internet Explorer 9 и последующий версии на эту операционную систему. Поэтому, пока популярна XP, будет популярен и Internet Explorer 8.

(Чтобы быть в курсе усовершенствований, которые появляются в веб-разработке, пользователи XP могут перейти на Chrome или Firefox.) В соответствии с данными StatCounter, на август 2012 года на Internet Explorer 8 приходилось 13,65 % от количества всех веб-браузеров ([http://gs.statcounter.com/#browser\\_version-ww-monthly-201108-201208](http://gs.statcounter.com/#browser_version-ww-monthly-201108-201208)). А согласно данным NetMarketShare, на долю Internet Explorer 8 приходилось ни много ни мало 26 % от всех браузеров (<http://marketshare.hitslink.com/browser-market-share.aspx?qprid=2&qpcustommd=0>). Поскольку Internet Explorer 8 не понимает теги HTML5, вы не сможете придать им формат средствами CSS непосредственным образом и будете вынуждены обратиться к обходным путям, в которых применяется JavaScript.

## Обеспечение использования самой последней версии Internet Explorer

Благодаря автоматическому обновлению продуктов Microsoft компьютеры под управлением Windows получают обновления до самых последних версий Internet Explorer. На момент написания данной книги у пользователей Windows версий 7 и 8 была установлена самая последняя версия Internet Explorer — 10. Эта версия поддерживает многие из самых впечатляющих и эффективных новых свойств HTML5 и CSS3. Из этой книги вы узнаете, что CSS3 предоставляет большое количество впечатляющих дизайнерских возможностей, таких как тени, градиенты, скругленные углы и т. д.

К сожалению, не все пользователи Windows смогут воспользоваться этим достижением веб-дизайна. Как уже говорилось в предыдущей врезке, широко применяемая операционная система Windows XP может работать только с Internet Explorer 8 или с более ранними версиями этого браузера. Фактически, Internet Explorer 8 по-прежнему является одной из наиболее часто используемых версий Internet Explorer. И в данной книге обязательно будут указываться те свойства CSS, которые не работают в Internet Explorer 8, а также возможные обходные пути, позволяющие все же их задействовать.

Поскольку популярность Internet Explorer 8 еще весьма высока, нужно считаться с рядом обстоятельств. Internet Explorer 8 подобен хамелеону: он может принимать внешний вид других версий. Если не проявить достаточную осмотрительность, он может отобразить вашу страницу совсем не в том виде, в котором бы хотелось. Например, особую важность приобретает включение в код правильного doctype-объявления. Как уже говорилось выше, без doctype-объявления браузеры переключаются в режим совместимости. А когда Internet Explorer 8 переходит в режим совместимости, он старается копировать внешний вид Internet Explorer 5 (!!!).

Но это еще не все! Internet Explorer 8 может притвориться Internet Explorer 7. Если кто-то, рассматривая ваш сайт в Internet Explorer 8, нажмет кнопку **Режим совместимости**, то Internet Explorer 8 станет Internet Explorer 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 Internet Explorer 7, а не в Internet Explorer 8. Если при создании сайта вы будете соблюдать все рекомендации, предложенные в этой книге, то этот режим вам *никогда* не понадобится.

К счастью, у нас есть возможность заставить Internet Explorer 8 быть самим собой. Для этого нужно добавить один meta-тег, чтобы Internet Explorer 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, отображая сайты, должны будут использовать свои стандартные режимы. К сожалению, эту строку придется помещать на каждой странице вашего сайта.

## ИНФОРМАЦИЯ ДЛЯ НОВИЧКОВ

### Кросс-браузерное тестирование

Как известно, существует множество различных браузеров. Если вы являетесь пользователем Windows, то по умолчанию вам доступен Internet Explorer. Кроме того, 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 и Google Chrome. Несмотря на то что все последние браузеры лучше всего приспособлены для работы с веб-страницами, основанными на использовании CSS, этого нельзя сказать об Internet Explorer 8, который все еще находит широкое применение.

Чтобы знать наверняка, что ваши сайты работают у самой широкой аудитории пользователей, вам потребуется протестировать их дизайн на как можно большем количестве браузеров. Для этого могу предложить несколько технологий.

**Для пользователей Windows.** Обычно компьютеры под управлением Windows могут запускать только одну версию браузера Internet Explorer — вы

не можете тестировать Internet Explorer 6, 7 и 8 на одном и том же компьютере. Если этого нельзя сделать в обычных условиях, то можно запустить на вашем Windows-компьютере виртуальные машины для создания отдельных версий Internet Explorer, которые затем можно протестировать. Это потребует немало объема работы, но зато даст вам наиболее точный способ тестирования страницы в нескольких версиях Internet Explorer. Подробности этой технологии приведены по адресу [www.howtogeek.com/102261/how-to-run-internet-explorer-7-8-and-9-at-the-same-time-using-virtual-machines](http://www.howtogeek.com/102261/how-to-run-internet-explorer-7-8-and-9-at-the-same-time-using-virtual-machines).

Нужно также будет установить на свой компьютер и другие основные браузеры: Firefox, Opera и Chrome. Apple прекратила поддержку Safari для Windows, поэтому вы можете просто положиться на браузер Chrome (у которого имеется точно такой же базовый механизм вывода информации на экран) или воспользоваться для тестирования вашей конструкции в Safari одной из служб, перечисленных далее в этой врезке (например, службой Adobe Browserlab).

## ИНФОРМАЦИЯ ДЛЯ НОВИЧКОВ

**Для пользователей Mac OS.** Тестирование на этой платформе дается намного сложнее. Без тестирования в Internet Explorer просто не обойтись, поскольку этот браузер по-прежнему является самым распространенным в мире, а отсутствие в Internet Explorer 8 многих свойств CSS3 означает, что ваш тщательно отработанный дизайн будет прекрасно выглядеть на вашей машине и полностью разваливаться в Internet Explorer 8. Для решения этой проблемы есть несколько вариантов: во-первых, можно купить (или одолжить) машину под управлением Windows; во-вторых, если на вашей Mac-машине стоит процессор Intel, можно установить Windows с помощью разработанной Apple программы Boot Camp ([www.apple.com/macosx/features/bootcamp.html](http://www.apple.com/macosx/features/bootcamp.html)); в-третьих, можно установить на Mac программное обеспечение для создания виртуальной машины, например VMware Fusion или Parallels Desktop. Это программное обеспечение позволит наряду с Mac OS запустить виртуальную Windows-машину.

**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 кросс-браузерного тестирования. Они позволяют видеть то, как выглядят страницы в разных операционных системах и браузерах.

- **CrossBrowserTesting.com** (<http://CrossBrowserTesting.com>) требует \$30 в месяц (!) за 150 минут использования и предлагает дополнительное преимущество — интерактивное тестирование. Вы получаете возможность просмотра своей страницы, запущенной удаленно на компьютере под вашим управлением. Вы можете тестировать свойства, которые не могут захватываться сним-

ком экрана, например проигрывание Flash-видео, анимацию и взаимодействие с интерактивными элементами, созданными с помощью JavaScript.

- **BrowserStack** ([www.browserstack.com](http://www.browserstack.com)) является еще одной службой, позволяющей тестировать ваши страницы в интерактивном режиме через ваш веб-браузер. За \$19 в месяц это похоже на аренду нескольких Windows- и Mac-машин с несколькими версиями установленных Internet Explorer, Chrome, Firefox и Safari.
- **Browsershots** ([www.browsershots.org](http://www.browsershots.org)) является бесплатной альтернативой, предоставляющей снимки экрана для широкого спектра браузеров, работающих под управлением Windows и Linux.
- Кроме того, внешний вид страниц в Internet Explorer 9, 8, 7, 6 и даже в 5.5 можно бесплатно посмотреть с помощью **NetRenderer** (<http://ipinfo.info/netrenderer>). Проверяемые страницы должны находиться на общедоступном веб-сайте. Зайдите на веб-сайт <http://ipinfo.info/netrenderer>, наберите URL-адрес страницы, и через некоторое время появится снимок экрана. К сожалению, это средство дает снимок не всей веб-страницы, а только той ее верхней части, которая обычно видна на экране. Но зато эта услуга предоставляется бесплатно.
- И наконец, можно воспользоваться средством **BrowserLab**, разработанным компанией Adobe (<https://browserlab.adobe.com>), для получения снимков веб-страницы во всех основных браузерах. Например, если для создания сайтов применялась программа Dreamweaver, вы можете запустить BrowserLab из программы и просматривать разрабатываемую страницу.

# 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 ли 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.

В нижней части рис. 2.1 представлен код таблицы стилей CSS, определяющей стиль основного содержимого веб-страницы, которая показана в верхней части рисунка.

---

## 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 пикселей 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 формирующие 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.

---

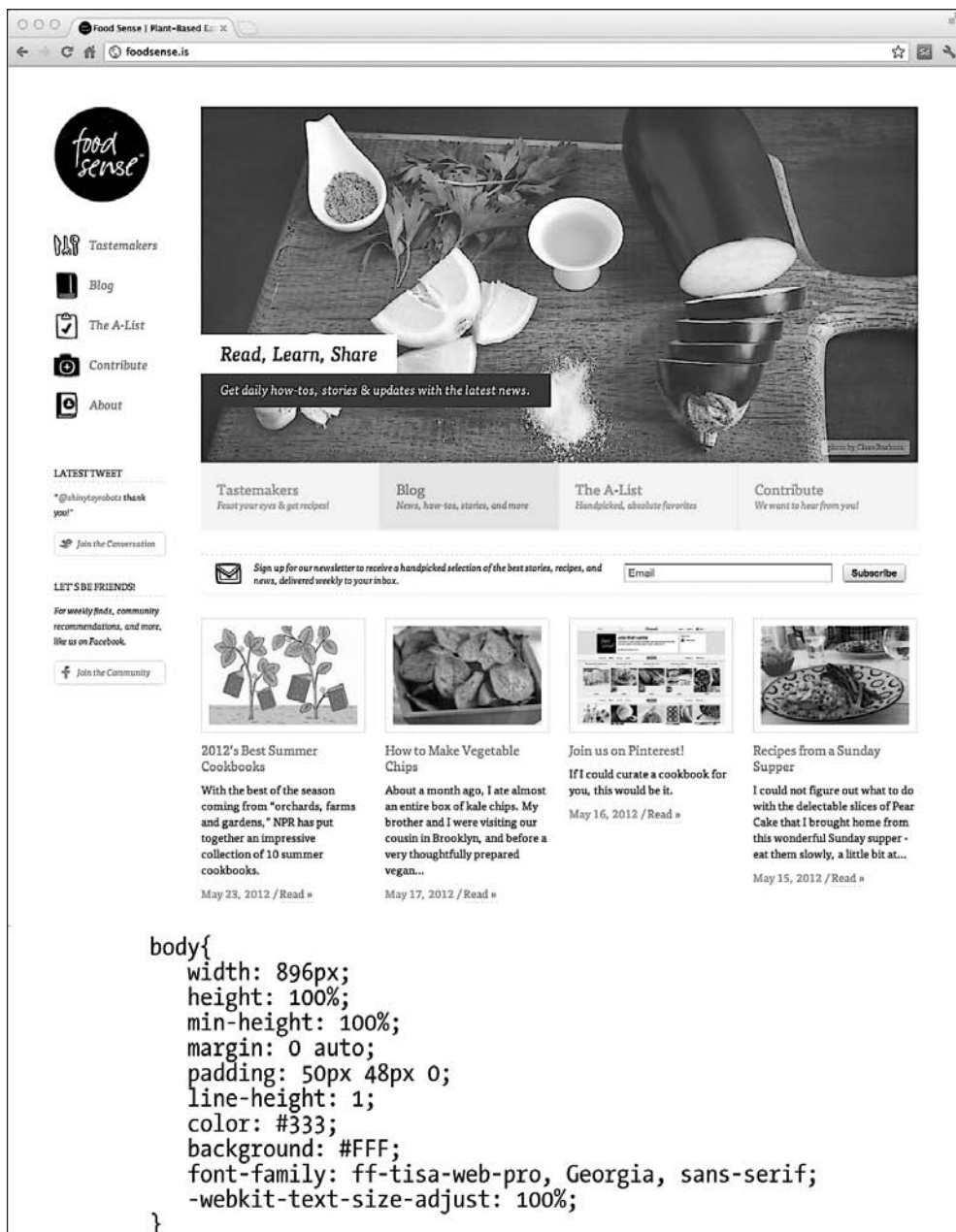


Рис. 2.1. Любая CSS-стилизованная веб-страница состоит из отдельных определений CSS-стилей

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 говорит браузеру: «Сделай текст всех абзацев веб-страницы, помеченных тегом <p>, красным и установи размер шрифта равным полуторакратной высоте латинской буквы m (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, сообщающего браузеру, что именно форматировать, и блока объявления, в котором перечислены формирующие команды, используемые браузером для стилизации фрагмента, определенного селектором.

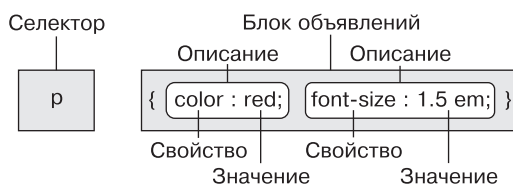


Рис. 2.2. Пример описания стиля (или правила)

- **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 значительному разнообразию селекторов, предлагаемых языком CSS, и своему 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 формирующие 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 формирующих команд. Каждое объявление имеет две части — *свойство* и *значение* свойства. Двоеточие отделяет имя свойства от его значения, и все объявление заканчивается 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 стилиевой 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. После имени свойства нужно добавить двоеточие, чтобы отделить его от значения.

#### ПРИМЕЧАНИЕ

В приложении 1 приведен 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, салатным 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 на одной строке, как изображено на рис. 2.2. У стилей может быть множество формирующих свойств, и есть возможность облегчить просмотр таблицы стилей путем разбивки объявлений на строки. Например, поместите селектор и открывающую скобку на одной строке, каждое объявление — далее на отдельных строках, а закрывающую фигурную скобку — отдельно на 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 читаемые 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 ее, вы собьете с толку браузер, в результате чего таблица стилей будет нарушена и веб-страница отобразится некорректно.

Однако не стоит переживать — описанная ошибка достаточно распространенная, 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 во врезке «Обходной прием. Не попадитесь с кэшированием».

#### 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) опять 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 повтор-

но 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.

**ОБХОДНОЙ ПРИЕМ**

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 Chrome и в Internet Explorer для этой цели предназначено сочетание клавиш Ctrl+F5, в Firefox — Ctrl+Shift+R, а Ctrl+R работает как в Safari, так и в Chrome.

## 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>
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.

---

**ПРИМЕЧАНИЕ**

В HTML5 для внутренней таблицы стилей в качестве открывающего требуется просто тег `<style>`. Предыдущие версии HTML и XHTML требуют в открывающем теге `<style>` дополнительного атрибута:

```
<style type="text/css">
```

Если опустить этот атрибут на странице, не относящейся к HTML5, то при попытке проверить веб-страницу будет выдана ошибка. Это является еще одним аргументом для использования HTML5.

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 создают много трудностей при обновлении дизайна сайта.

Например, нужно изменить представление заголовков первого уровня (заклученных 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 обнаружилось 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 включать в себя HTML-код, поэтому никогда не добавляйте 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 просто `styles.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. Проверяйте правильность кода веб-страниц» в разделе «Важность доctype» 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

### ИНФОРМАЦИЯ ДЛЯ НОВИЧКОВ

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.

манный 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.

### 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. Эта передовая технология будет рассмотрена чуть позже.

## Связывание таблиц стилей с HTML-кодом

Наиболее распространенный метод добавления внешней таблицы стилей в веб-страницы — применение HTML-тега `<link>`. Синтаксис применения данного тега немного различается в зависимости от того, чем вы пользуетесь — HTML5, HTML 4.01 или XHTML. Например, так выглядит вариант для HTML5:

```
<link rel="stylesheet" href="css/styles.css">
```

Вариант для HTML 4.01 похож на предыдущий, но требует дополнительного атрибута `type`:

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

И наконец, вариант для XHTML очень похож на вариант для HTML 4.01, но требует в конце тега ставить закрывающий слеш:

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

Поскольку для HTML5 требуется меньше кода, рекомендуется использовать HTML5. Для всех типов документов необходимы два атрибута:

- `rel="stylesheet"` — указывает тип ссылки; в данном случае это ссылка на таблицу стилей;

- `href` — задает 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 разделе «Организация стилей и таблиц стилей» гл. 17.

---

## 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/styles.css);
</style>
```

---

**ПРИМЕЧАНИЕ**

Если используется тип документа HTML 4.01 или XHTML, нужно к открывающему тегу `<style>` добавить атрибут `type="text/css"`, о чем уже упоминалось в одной из предыдущих врезок.

---

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/styles.css` — путь к внешнему CSS-файлу. Кавычки, в которые заключен URL, не обязательны. Таким образом, `url(css/styles.css)` и `url("css/styles.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/styles.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 разделе «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» раздела «Организация стилей и таблиц стилей» гл. 17).

---

Например:

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

Нужно поместить все правила @import перед CSS-стилями, как показано в примере. Веб-браузеры игнорируют любые таблицы стилей, импортируемые после CSS-правила, поэтому если изменить порядок показанного выше кода на обратный и стиль p появится первым, то браузер проигнорирует любые стили в таблицах стилей style.css или form.css.

Так какой метод лучше использовать? Хотя оба они работают, применение 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](http://www.stevesouders.com/blog/2009/04/09/dont-use-import)). Поэтому, если у вас нет явных предпочтений одному из методов, просто используйте тег <link>.

## 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/css3](http://www.sawmac.com/css3). 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 TextEdit) или программный комплекс для визуального проектирования (Dreamweaver или Microsoft Expression Web Designer) (перечень программного обеспечения приведен во введении).

## Создание встроенного стиля

Размещая стилевые команды CSS 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 сторон в их использовании.

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 применять 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 программе редактирования веб-страниц откройте файл 02\basic.HTML.

Этот простой и изящно написанный HTML5-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: #666666;"`.

Тег должен выглядеть следующим образом:

```
<h1 style="color: #666666;">
```

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, которая находится в кавычках. В данном случае мы добавили свойство `color`, которое воздействует на цвет текста, и установили его значение равным `#666666`, то есть шестнадцатеричному коду, определяющему серый цвет (об изменении цвета текста читайте в разделе «Придание тексту цветового оформления» гл. 6). Двоеточие отделяет название свойства от значения, которое мы хотим установить для данного свойства. Далее проверим результат в браузере.

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

Запустите 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 открытое 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 увидите, что заголовок стал серым. Встроенные стили могут содержать более одного свойства CSS. Добавим в тег еще одно свойство.

4. Вернитесь в HTML-редактор. После точки с запятой за кодом `#666666` наберите `font-size: 3em;`.

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

```
<h1 style="color: #666666;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 подраздел, 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>`.

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

```
<title>Большая книга CSS - Глава 2</title>
<style>
</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 на 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: #666666;`.

Это текст того же свойства стиля, что и во встроенном варианте, — свойство `color` со значением `#666666`. Точка с запятой обозначает окончание объявления свойства.

---

**ПРИМЕЧАНИЕ**

Исходя из синтаксиса языка CSS, не обязательно размещать каждое свойство стиля на 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;  
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 во второй части 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>CSS: The Missing Manual -- Chapter 2</title>  
<style>  
  h1 {  
    color: #666666;  
    font-size: 3em;  
    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 добавим другой стиль.

## ПРИМЕЧАНИЕ

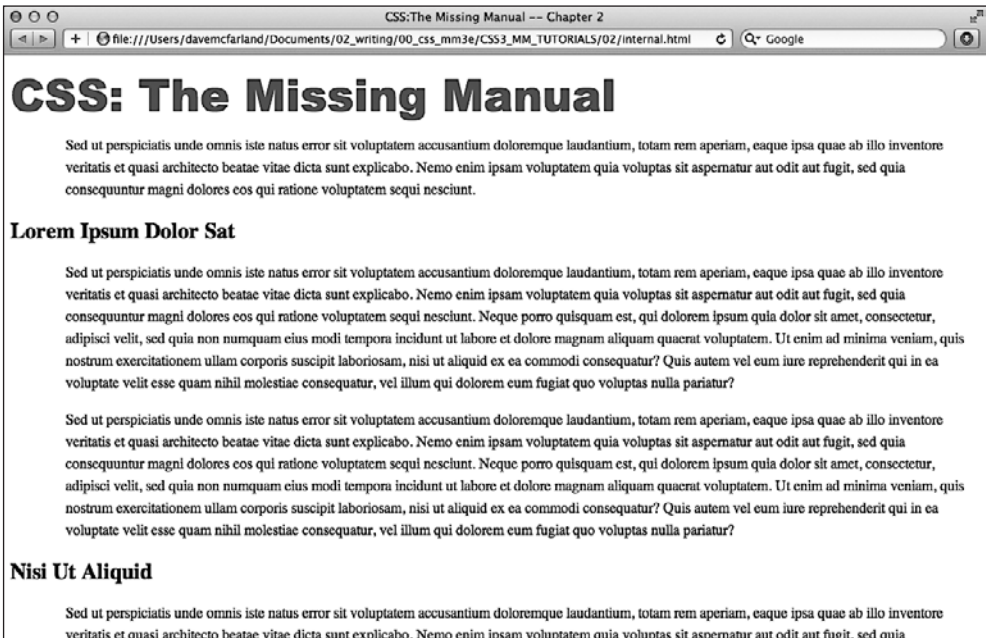
Никогда не забывайте добавлять закрывающий тег `</style>` в конце внутренней таблицы стилей. Если вы не сделаете этого, браузер отобразит на экране код стилей CSS, за которым последует сама веб-страница без всякого форматирования, а может быть и такое, что браузер вообще не покажет содержимого веб-страницы.

- 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.

- Просмотрите страницу в браузере (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 подразделе мы возьмем стили, созданные в предыдущем уроке, и поместим их во внешнюю таблицу стилей.

1. В программе редактирования создайте новый файл и сохраните его под именем `styles.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 `styles.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 для файла `styles.css` следующее правило:

```
html {
  padding-top: 25px;
  background-image: url(images/bg_page.png);
}
```

Это правило касается тега HTML, то есть тега, окружающего все остальные HTML-теги, которые имеются на странице. Свойство `padding-top` добавляет пространство между верхней частью тега и помещаемым внутри него содержимым. Иными словами, то, что мы только что набрали, приведет к добавлению 25 пикселей пустого пространства между верхней частью окна браузера и содержимым страницы. Свойство `background-image` добавляет к странице фоновое изображение. CSS-свойство `background-image` может отобразить графику множеством различных способов, но в данном случае оно будет выведено в виде неперекрывающейся мозаики слева направо и сверху вниз. Дополнительные сведения о свойствах фоновых изображений будут даны в гл. 8.

3. Добавьте второе правило сразу же после только что набранного для файла `styles.css` первого правила:

```
body {
  width: 80%;
  padding: 20px;
```

```
margin: 0 auto;
box-shadow: 10px 10px 10px rgba(0,0,0,.5);
background-color: #E1EDEB;
}
```

Это правило относится к тегу содержимого веб-страницы (`<body>`), который включает в себя всю информацию, видимую в окне браузера. В данном определении стиля задается множество различных свойств, каждое из которых еще будет рассмотрено более подробно в данной книге. Но если дать ему краткое описание, этот стиль создает прямоугольную область для содержимого страницы, составляющую 80 % от ширины окна браузера, имеющую внутри себя небольшое пустое пространство, которое сдвигает текст от краев прямоугольной области (речь идет о свойстве `padding`), и производит центровку прямоугольной области на странице (имеется в виду свойство `margin` и конкретный прием центровки содержимого страницы, рассматриваемый в обучающем уроке гл. 3). И наконец, прямоугольная область получает светло-синий цвет фона и прозрачную темную отбрасываемую тень.

Вместо повторного набора стилей, созданных в предыдущем уроке, просто скопируем стили, которые определены во внутренней таблице стилей, и вставим их в эту внешнюю таблицу стилей.

4. Откройте страницу `basic.html`, над которой работали, и скопируйте весь текст, содержащийся внутри тегов `<style>` (не копируйте сами теги `<style>`).

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

5. В файл `styles.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>`.

6. Сохраните файл `styles.css`.

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

7. 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. В этом уроке вы окунетесь в захватывающий мир веб-шрифтов. Все, что касается веб-шрифтов, будет рассмотрено в гл. 6, но основная идея заключается в том, что вы можете использовать практически любой шрифт, который нужен на веб-странице, даже шрифт, который посетители вашего сайта не установили на своих компьютерах, предоставив просто ссылку на файл с этим шрифтом. Существует множество различных способов применения веб-шрифтов, но в данном примере будет использована служба веб-шрифтов Google.

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

```
<link href="http://fonts.googleapis.com/css?family=Gravitas+One"
rel="stylesheet">
```

Как и прежде, вам пока не стоит вдаваться в подробности. На данном этапе нужно лишь знать, что, встретившись с данной ссылкой, браузер загружает шрифт, который называется Gravitas One с сервера Google server и который ваши CSS-стили могут свободно использовать для повышения выразительности текста.

Затем вы укажете ссылку на созданную ранее внешнюю таблицу стилей.

9. После только что добавленного тега `<link>` наберите следующее:

```
<link href="styles.CSS" rel="stylesheet">
```

Тег `<link>` — один из способов присоединить внешнюю таблицу стилей к веб-странице; другой вариант — использование CSS-правила `@import`, описанного выше. Тег `<link>` определяет местонахождение внешней таблицы стилей. Атрибут `rel` просто оповещает браузер о том, что ссылка производится на таблицу стилей.

---

#### ПРИМЕЧАНИЕ

В данном примере файл внешней таблицы стилей расположен в той же самой папке, что и веб-страница, поэтому 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 по этой теме смотрите по адресу [www.communitymx.com/content/article.cfm?cid=230AD](http://www.communitymx.com/content/article.cfm?cid=230AD)).

---

10. Сохраните файл и просмотрите его в браузере.

Вы увидите одни и те же текстовые стили для тегов `<h1>` и `<p>`, которые были созданы ранее во внутренней таблице стилей. Кроме того, теперь есть пятнистый желто-коричневый фон (фоновое изображение, примененное в теге `<html>`), а также синий прямоугольник. Этот прямоугольник является тегом `<body>`, и его ширина составляет 80 % от ширины окна браузера. Попробуйте изменить размер окна браузера и обратите внимание на то, что размер прямоугольника также изменяется. Прямоугольник тоже отбрасывает тень, но сквозь тень можно увидеть пятнистый фон. Все это получается благодаря типу цвета CSS3 `rgba color`, включающего установки прозрачности (которая будет рассмотрена далее).

---

#### ПРИМЕЧАНИЕ

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

---

Теперь вы увидите веб-шрифт, ссылка на который была сделана на восьмом шаге.

11. Запустите свой текстовый редактор и вернитесь к файлу `styles.css`. Добавьте к стилю для `h1` следующие две строки:

```
font-family: 'Gravitas One', 'Arial Black', serif;
font-weight: normal;
```

В конечном результате стиль должен приобрести следующий вид (добавления выделены полужирным шрифтом):

```
h1 {
  font-family: 'Gravitas One', 'Arial Black', serif;
  font-weight: normal;
  color: #666666;
  font-size: 3em;
  margin: 0;
}
```

Теперь при просмотре страницы вы увидите в заголовке новый шрифт Gravitas One.

#### ПРИМЕЧАНИЕ

Если вы не видите нового шрифта, который является разновидностью полужирного шрифта с тонкими линиями (засечками) на концах букв, значит, либо отсутствует подключение к Интернету, что помешает загрузить шрифт с сайта Google, либо вы допустили опечатку в теге <link> (при выполнении шага восемь) или в объявлении font-family (во второй строке показанного выше кода).

Чтобы лучше понять пользу от хранения стилей в их собственных отдельных файлах, вы можете прикрепить таблицу стилей к другой веб-странице.

- 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.

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

Сейчас нужно присоединить к этой веб-странице ссылку на веб-шрифт и уже созданную внешнюю таблицу стилей.

- Наберите тот тег <link>, который применялся в восьмом и девятом шагах.

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

```
<title>Another Page</title>
<link href="http://fonts.googleapis.com/css?family=Gravitas+One"
rel="stylesheet">
<link href="styles.css" rel="stylesheet">
</head>
```

- Сохраните страницу и просмотрите ее в браузере.

Достаточно всего двух строк кода, добавленных 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.

- Откройте файл styles.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;  
}
```

В данном случае веб-шрифт не используется и все зависит от наличия перечисленных шрифтов на машине посетителя сайта (все вопросы применения шрифтов будут рассмотрены позже). Затем создайте новый стиль для элемента h2.

17. Установите курсор после заключительной фигурной скобки } стиля элемента p, нажмите **Enter** и добавьте следующий стиль:

```
h2 {  
color: #B1967C;  
font-weight: normal;  
font-family: 'Gravitas One', 'Arial Black', serif;  
font-weight: normal;  
font-size: 2.2em;  
border-bottom: 2px white solid;  
background: url(images/head-icon.png) no-repeat 10px 10px;  
padding: 0 0 2px 60px;  
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, мы подробно рассмотрим их в следующих главах. О свойствах шрифта читайте в гл. 6, о свойствах, задающих 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.

18. 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. Чтобы этот новый стиль работал, придется немного отредактировать HTML-код.



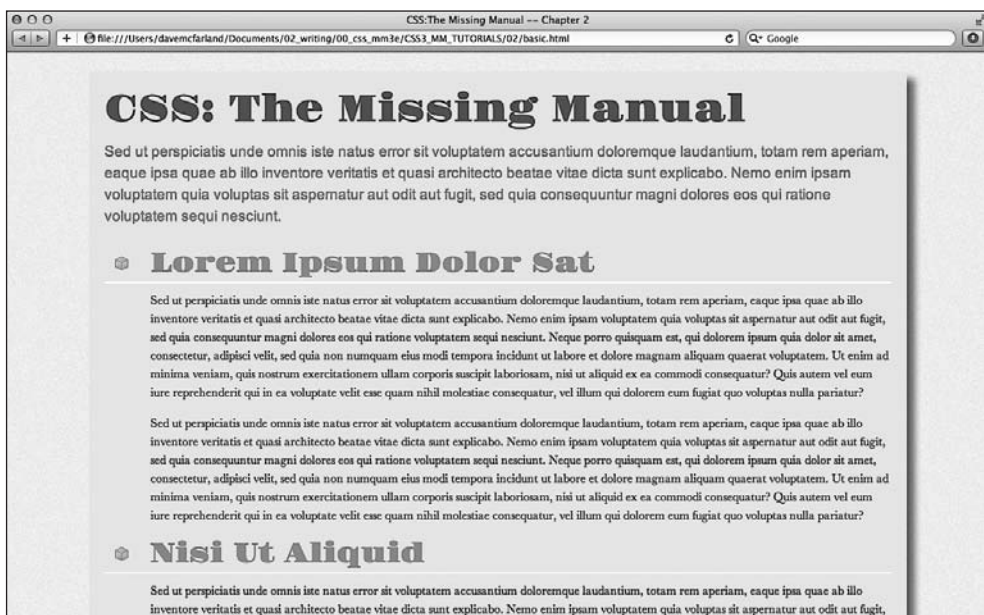
19. Сохраните файл `styles.css` и перейдите к файлу `basic.html` в своем текстовом редакторе. Найдите открывающий тег `<p>`, который следует за тегом `<h1>`, и добавьте `class="intro"`. Открывающий тег должен выглядеть следующим образом:

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

Вам не нужно добавлять точку перед словом `intro`, как вы это делали, создавая стиль на шаге 18 (почему так, вы узнаете в следующей главе). Этот дополнительный код HTML применяет стиль к первому абзацу (и только к первому).

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

20. 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.



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

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. 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`.)

Для тренировки добавьте стиль, чтобы отформатировать тег `<address>` (этот тег используется в нижней части HTML-файла для хранения сообщения об авторских правах). Можно, например, изменить его цвет и указать отступ 60 пикселей, чтобы он совпал с отступом тегов `<p>`.



# 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 (рис. 3.1).

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

**Рис. 3.1.** 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*) являются весьма эффективным средством 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 увеличился бы в объеме до невероятных размеров, страницы загружались бы медленнее, их обновление занимало бы много времени. Благодаря селекторами типов вам фактически ничего не нужно делать с HTML-кодом, просто создайте CSS-стили и позвольте браузеру выполнить все остальное.

Селекторы исключительно просто определить в CSS-стилях, так как они наследуют название формируемых элементов — `p`, `h1`, `table`, `img` и т. д. Например, страница, представленная на рис. 3.2, 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;  
}
```

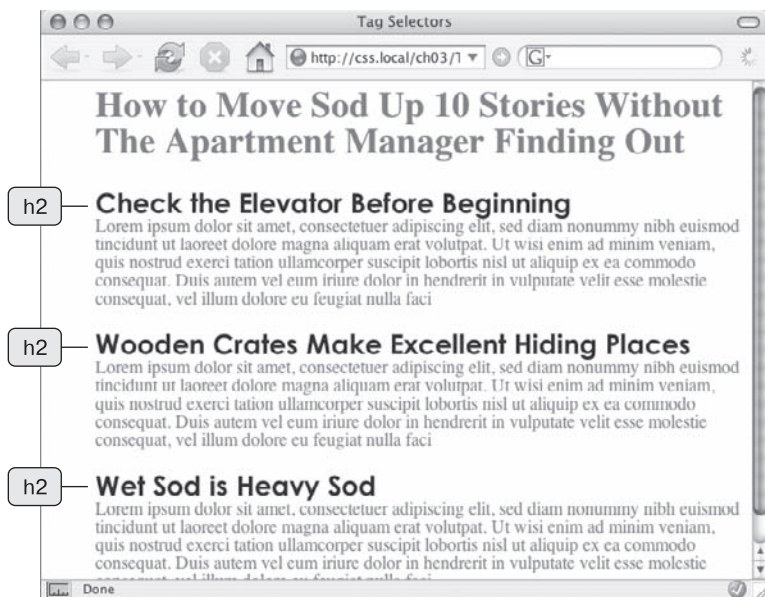


Рис. 3.2. Селектор тега воздействует на все экземпляры указанного элемента веб-страницы

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 той же веб-страницы, например, хотите задать для одного или двух рисунков красную рамку, оставив все остальные изображения нестилизован-ными, 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. 3.3. 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. 3.3) 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. 3.3), то используйте тег `<span>`. Для более детального ознакомления смотрите ниже врезку «Информация для новичков. HTML-теги `<div>` и `<span>`».

---

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.

- 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` рассматриваются языком CSS по-разному, как различные классы.

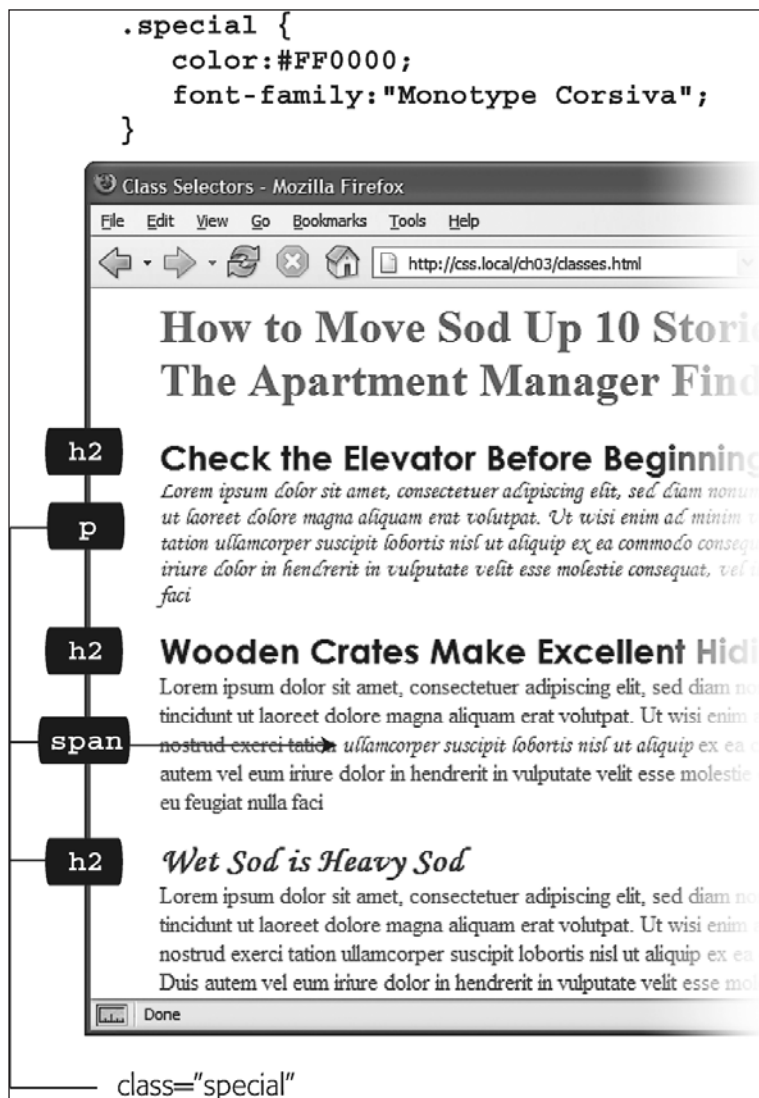


Рис. 3.3. Селекторы классов позволяют целенаправленно изменять дизайн фрагментов веб-страниц

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, необходимо 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>` таким образом:

```
<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>`. Например, чтобы выделить только одно слово в абзаце, применяя стиль `.special`, можно написать следующее:

```
<p>Welcome to <span class="companyName">Café Soy lent Green</span>, the restaurant  
with something a little different.</p>
```

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>`.

### ИНФОРМАЦИЯ ДЛЯ НОВИЧКОВ

#### HTML-теги `<div>` и `<span>`

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 панели, 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>`.

## ИНФОРМАЦИЯ ДЛЯ НОВИЧКОВ

Тег `<div>` имеет уникальную способность включать в себя несколько блочных элементов, являясь средством группировки (логотип и панель навигации или блок новостей, формирующий боковой столбец). После группировки содержимого веб-страницы таким образом вы можете применить определенное форматирование исключительно к тегам, находящимся внутри этого фрагмента `<div>`, 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.

Например, вы добавили на веб-страницу фотографию, у которой есть сопроводительное описание. Можно использовать `<div>` (с применением к тегу класса), чтобы объединить в группу оба элемента:

```
<div class="photo">
  
  <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 `<div>`, включая вложенные конструкции из этих тегов.

С другой стороны, `<span>` 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 линейными (inline) элементами), чтобы форматировать их отдельно друг от друга.

В данном примере стилевой класс, названный `.companyName`, стилизует линейные 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>
```

## 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 «Информация для опытных пользователей. Специальное применение идентификаторов», 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 применения ID-селекторов.

### ПРИМЕЧАНИЕ

В сообществе веб-разработчиков есть тенденция к отказу от ID-селекторов в CSS. Причины этого явления требуют более глубокого знания CSS, чем то, которое может быть получено при изучении этой книги на данный момент, поэтому здесь ID-селекторы не будут слишком часто встречаться, а все подробности того, почему применение ID-селекторов нельзя признать удачным решением, будут изложены позже.

Хотя многие веб-разработчики уже не пользуются ID-селекторами так же часто, как раньше, все же следует знать, что они собой представляют и как работают. Применение 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`. Например, для применения показанного выше стиля к тегу `<div>` нужно написать следующий код HTML:

```
<div id="banner">
```

Точно так же 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: `<div id="banner">`.

### 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. Но если вам это интересно, рассмотрим некоторые особенности.

- Одним из простых способов для JavaScript-программистов, позволяющих определить место и манипулировать частями страницы, является применение ID элементу страницы с последующим использованием JavaScript для ссылки на этот ID. Специалисты часто применяют идентификаторы к заполняемым элементам форм (например, к текстовым полям) для получения имени посетителя сайта и т. д. Это позволяет JavaScript получить доступ к полям форм и совершать с ними разные манипуляции, например, когда посетитель щелкает кнопкой мыши на кнопке отправки формы, убеждаться в том, что поля заполнены.
- 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

### ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

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 изменить 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 формирующие свойства, то можете создать групповой селектор с общими командами и отдельные стили с уникальным форматированием для каждого индивидуального элемента. Другими словами, два (или больше) стили могут форматировать один и тот же тег. Это и является мощной особенностью языка CSS (по этой теме см. гл. 5).

## Универсальный селектор

Групповые селекторы можно рассматривать как подручное средство для применения одинаковых свойств различным элементам. 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 задействовать 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 атрибут `class` со значением `.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 отдельных элементов страницы, но создание стилей для селекторов `class` или `ID` также требует добавления соответствующего `class`- или `ID`-атрибута к тегам HTML, которые требуют стиливого оформления. Это добавляет не только работу для вас, но и код для вашего HTML-файла. Все, что нужно в этом случае, — объединить простоту использования селекторов тегов с точностью селекторов классов и идентификаторов. И такое средство в CSS есть, это *селекторы потомков*.

Их применяют для того, чтобы единообразно отформатировать целый набор тегов, но только когда они расположены в определенном контексте — фрагменте веб-страницы. Их работу можно описать так: «Эй вы, `<a>`-теги на панели навигации, прислушайтесь. У меня есть для вас указания по форматированию. А все остальные `<a>`-теги проходите мимо, вам здесь нечего смотреть».

Селекторы потомков позволяют форматировать теги в зависимости от их связей с другими тегами. Чтобы разобраться, как они работают, придется немного покопаться в языке HTML. Понимание работы селекторов потомков поможет получить представление о функционировании других типов селекторов, работа которых описана далее.

### ПРИМЕЧАНИЕ

---

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

---

## Дерево HTML

Код HTML, на котором написана любая веб-страница, напоминает 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.

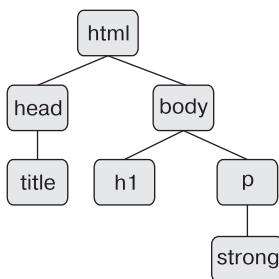
Тег, расположенный внутри другого, — его *потомок*. Тег `<title>` в следующем примере — потомок `<head>`:

```

<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. 3.4). 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. 3.4.** HTML состоит из вложенных тегов, которые представляют 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 селектор потомков, основывается на их родственных отношениях. Рассмотрим самые важные из них.

- **Предок.** Как описано в начале данного подраздела, HTML-элемент, который включает в себе другие элементы, — это предок. На рис. 3.4 это `<html>`, в то время как `<body>` — предок для всех содержащихся в нем тегов: `<h1>`, `<p>`, `<strong>`.
- **Потомок.** Элемент, расположенный внутри одного или более типов — потомок. На рис. 3.4 `<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. 3.4 `<html>` является родительским только для `<head>` и `<body>`. Тег `<p>` — родительский по отношению к `<strong>`.
- **Дочерний элемент.** Элемент, непосредственно подчиненный другому элементу более высокого уровня, является дочерним. На рис. 3.4 оба тега — `<h1>` и `<p>` — дочерние по отношению к `<body>`, но `<strong>` не является дочерним для `<body>`,

так как он расположен непосредственно внутри тега `<p>`, он является дочерним для этого тега.

- **Сестринский элемент.** Элементы, являющиеся дочерними для одного и того же родительского тега, называются *соседними* или элементами одного уровня. На рис. 3.4 они расположены рядом друг с другом. Теги `<head>` и `<body>` — элементы одного уровня, как и `<h1>` и `<p>`.

## 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 `<strong>`. 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 `<strong>` 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 `<strong>`, 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 `<strong>` внутри заголовка. Подход, основанный на селекторах потомков, 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 формируемый тег — справа. На рис. 3.5 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; }
```

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

Этот стиль говорит: «Нужно отформатировать все ссылки (a), которые расположены в элементах списка (li), используя шрифт Arial». Вообще, селекторы потомков могут включать более двух элементов. Ниже представлены правильные определения для тегов `<a>`, находящихся в маркированных списках (см. рис. 3.5):

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

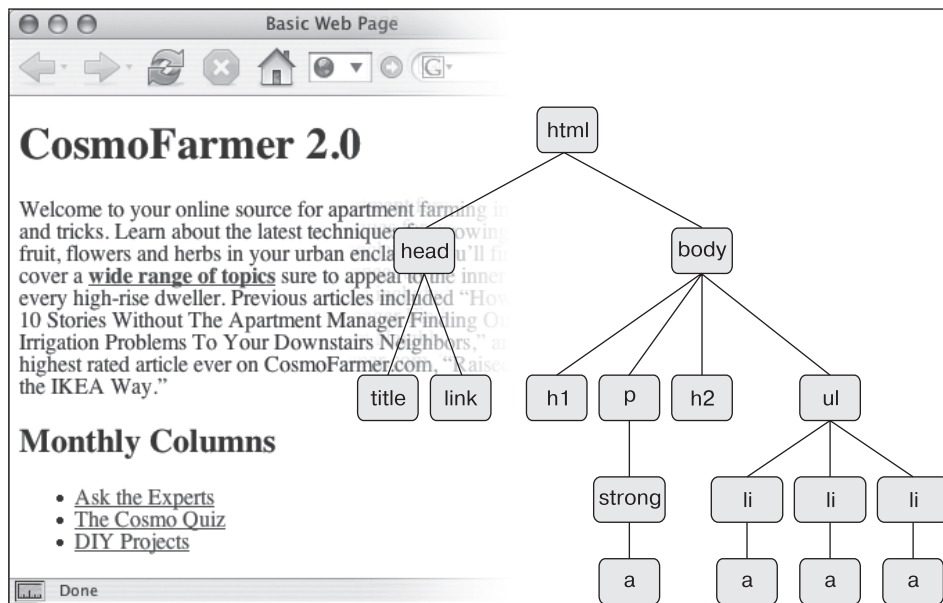


Рис. 3.5. Простейшая диаграмма в виде дерева, представляющая структуру веб-страницы

#### ПРИМЕЧАНИЕ

Несмотря на то что лучше стараться использовать как можно более короткие селекторы потомков, одна из причин, по которой удобно применять дополнительные селекторы потомков, — возможность замены нескольких различных стилей, одновременно формирующих 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 элементе введения (которое вы определили стилевым классом `intro`). Этого можно добиться с помощью следующего селектора:

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

Краткий перевод звучит так: «Нужно применить данный стиль ко всем ссылкам (`a`) — потомкам абзаца (`p`), относящегося к классу с именем `intro`».

## Создание модулей

Селекторы потомков часто используются для форматирования модуля кода, то есть коллекции HTML-элементов, выполняющих на странице конкретную функцию. Например, предположим, что на странице присутствует `<div>`-контейнер, применяемый для вывода перечня последних новостей компании. Код HTML может иметь следующий вид:

```
<div>
  <h2>Our company is great!</h2>
  <p>More information about why our company is so great</p>
  <h2>Another news item</h2>
  <p>Information about the other news item...</p>
  <h2>...and so on...</h2>
  <p>... and so on... </p>
</div>
```

Если вставить атрибут `class` в открывающий `<div>`-тег — `<div class="news">`, то вы можете создать селекторы потомков, по-разному форматирующие HTML-теги внутри раздела новостей. Например,

```
.news h2 { color: red; }
.news p { color: blue; }
```

Теперь содержимое `<h2>`-тегов внутри раздела новостей будет красным, а абзацы — синими. Можно даже создать (что часто и делается) селекторы потомков с несколькими именами классов. Предположим, например, что вы создаете страницу, предоставляющую каталог адресов сотрудников какой-нибудь организации. Вы можете заключить каждую контактную информацию в ее собственный `div`-контейнер, а затем улучшить внешний вид элементов внутри этого `div`-контейнера с помощью атрибута `class`:

```
<div class="contact">
  <p class="name">John Smith</p>
  <p class="phone">555-555-1234</p>
  <p class="address">1234 Elem St</p>
</div>
```

Затем можно создать несколько селекторов потомков для придания стиля только этим элементам контактной информации:

```
.contact .name { font-weight: bold; }
.contact .phone { color: blue ;}
.contact .address { color: red; }
```

### ПРИМЕЧАНИЕ

Иногда в таблице стилей можно увидеть следующий код:

```
p.intro
```

Может показаться, что это похоже на селектор потомка, поскольку в нем присутствует как название HTML-тега, так и имя класса, но это не так. Между `p` и `.intro` нет пробела, значит, чтобы этот стиль работал, селектор класса `intro` должен быть применен конкретно к тегу `<p>` (`<p class="intro">`). Если добавить пробел, вы получите другой эффект:

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

Это несколько иной вариант, выбирающий любой тег, стилизованный под класс `.intro`, который, в свою очередь, является потомком тега `<p>`. Иными словами, он не выбирает абзац, он выбирает другой тег внутри абзаца. В общем, чтобы максимально сохранить гибкость ваших стилей класса, лучше оставить в покое HTML-тег (иными словами, использовать только `.intro` вместо `p.intro`).

---

## 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 перешел, даже если на нее не наведен указатель мыши. Это обычный стиль непосещенных гиперссылок.

`a:visited` является ссылкой, по которой посетитель веб-страницы уже перешел. Она сохраняется в истории браузера. Можно разработать для этого типа стиль, отличный от обычного, чтобы сказать посетителю, что он уже посещал эту страницу. (Ограничения, связанные с этим селектором, будут рассмотрены во врезке в гл. 9.)

`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 этом случае вместо применения `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`.

`a:active` позволяет определить, как будет выглядеть ссылка во время выбора ее посетителем веб-страницы. Другими словами, это стиль во время кратковременного 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 утруждайте себя 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 со-

здавать 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 вопрос дизайна страниц и сайтов не стоял, 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 (о форматировании текста читайте в гл. 6, там же вы найдете подробное описание этих двух псевдоэлементов).

### ПРИМЕЧАНИЕ

---

В CSS3 синтаксис для псевдоэлементов претерпел изменения. В CSS 2.1 псевдоэлементы начинались с одинарного двоеточия:

```
:first-letter
```

В CSS3, чтобы отличить такие псевдоклассы, как `:hover`, от псевдоэлементов, добавлено еще одно двоеточие. Поэтому `:first-letter` и `:first-line` стали теперь выглядеть как `::first-letter` и `::first-line`. К счастью, чтобы сохранить поддержку более старых сайтов, браузер будет продолжать поддерживать версию псевдоэлементов с одинарным двоеточием. Это хорошо, потому что Internet Explorer 8 не понимает синтаксис с двумя двоеточиями, поэтому пока можно остановиться на одинарном двоеточии, поскольку все остальные браузеры используют их как и раньше.

---

## Дополнительные псевдоклассы и псевдоэлементы

В руководстве по языку CSS определены еще несколько мощных псевдоклассов, псевдоэлементов и селекторов. Эти селекторы очень хорошо поддерживаются во всех браузерах, кроме самых старых.

**:focus.** Псевдокласс `:focus` функционирует подобно `:hover` с той лишь разницей, что `:hover` применяется, когда посетитель перемещает указатель мыши над ссылкой, а `:focus` — когда нажимают клавишу табуляции или щелкают 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, он, как и весь стилевой эффект CSS, исчезает.

#### СОВЕТ

---

Посмотреть, какими браузерами какие CSS-селекторы поддерживаются, можно на веб-сайте [www.quirksmode.org/css/contents.html](http://wwwquirksmode.org/css/contents.html).

---

**: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 визуальнo 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`.

Селектор `:before` поддерживается браузером Explorer 8 и выше, а также другими основными браузерами (как и придуманный после него селектор `:after`). Дополнительные сведения об использовании этих селекторов можно найти во врезке в гл. 8.

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

#### ПРИМЕЧАНИЕ

---

Оба псевдоэлемента — `:before` и `:after` — похожи на `:first-line` и `:first-letter`. Как уже ранее упоминалось, в CSS3 к псевдоэлементам добавилось двойное двоеточие, поэтому `:before` и `:after` в CSS3 теперь пишутся как `::before` и `::after`. К счастью, браузеры поддерживают и более старую нотацию,



поэтому вы можете продолжать использовать `:before` и `:after`, что добавляет преимуществ при работе в Internet Explorer 8.

---

**::Selection.** Этот селектор, появившийся в CSS3, ссылается на элементы, которые посетитель выбрал на странице. Например, когда посетитель щелкает на тексте и проводит над ним указатель мыши, браузер выделяет этот текст и посетитель может затем скопировать текст. Обычно браузеры добавляют за текстом синий фон. Internet Explorer изменяет цвет текста на белый. Но вы можете управлять цветом фона и текста, определив этот селектор. Например, если нужно сделать выбранный текст белым на фиолетовом фоне, этот стиль можно добавить к таблице стилей страницы:

```
::selection {
  color: #FFFFFF;
  background-color: #993366;
}
```

Единственными свойствами, которые можно установить с помощью данного селектора, являются `color` и `background-color`, поэтому вы не можете дойти до такого безумия, чтобы изменить размер шрифта, шрифты, поля и внести другие визуальные изменения, которые точно сведут с ума посетителей вашего сайта. (Спасибо CSS за нашу защиту от нас самих!)

---

#### ПРИМЕЧАНИЕ

Версии с одинарным двоеточием для псевдоэлемента `selection` не существует, поэтому вы должны использовать двойное двоеточие. Иными словами, `::selection` работает, а `:selection` нет.

---

Этот селектор функционирует в Internet Explorer 9, Opera, Chrome и Safari, но не работает в Internet Explorer 8 или Firefox. Но вы можете добавить поддержку для Firefox путем дополнения селектора так называемым префиксом производителя:

```
::-moz-selection {
  color: #FFFFFF;
  background-color: #993366;
}
```

Чтобы это работало в Firefox и в других браузерах, в таблице стилей нужно иметь оба стиля, которые можно просто расположить друг за другом. (Подробности, касающиеся префиксов производителей и причин, по которым в них возникает потребность, изложены во врезке в гл. 7.)

Если вы действительно хотите кого-нибудь свести с ума, можете указать другой цвет фона для текста, выделенного внутри конкретного элемента. Например, чтобы сделать на розовом фоне красным текст, который находится только внутри абзацев, нужно добавить селектор `p`-элемента перед `::selection`:

```
p::selection {
  color: red;
  background-color: pink;
}
```

---

**ПРИМЕЧАНИЕ**

---

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

---

## 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 форматирова 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 способствует 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 один 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, можно 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. Хотя эта возможность сразу может показаться лишней, на самом деле она достаточно полезна.

Представьте, что вы хотите создать стиль, который выделял бы 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 с», так что вы можете использовать этот селектор для форматирования любой ссылки, начинающейся с `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 с `https://`. Чтобы создать стиль, учитывающий данную проблему, вам понадобится групповой селектор:

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

Встречаются также ситуации, когда необходимо выбрать элемент с атрибутом, заканчивающимся определенным значением. И снова ссылки подходят 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, необходим 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 прочтаете о них 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 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 фотографию 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.

## 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. 3.6 показаны 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 `<div>`, который и является родительским.

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

Для выбора дочерних элементов в CSS3 также включены некоторые весьма специфические псевдоклассы. Они позволяют точнее настроить селекторы под многие различные размещения кода HTML.

**:first-child.** Возвращаясь к аналогии с семейным деревом HTML, вспомним, что такое дочерний тег: это любой тег, непосредственно заключенный в другой тег. (Например, на рис. 3.6 теги `<h1>`, `<div>`, `<h2>` и `<ul>` являются дочерними тегами тега `<body>`.) Псевдоэлемент `:first-child` позволяет выбирать и форматировать только первый из них, независимо от того, сколько их вообще может быть.

На рис. 3.6, если нужно выбрать первый тег `<h1>` на странице, создайте следующий селектор:

```
h1:first-child
```

Этот селектор применяется к любому тегу `<h1>`, являющемуся первым дочерним тегом. На рис. 3.6 результат будет очевиден, там только один тег `<h1>`, и он является первым тегом на странице. Следовательно, он является дочерним тегом элемента `<body>`. Но `:first-child` может вызвать путаницу. Например, если вы измените тег `<h2>` внутри тега `<div>`, показанного на рис. 3.6, на тег `<h1>`, то `h1:first-child` выберет оба `<h1>`, тот, который непосредственно находится внутри тега `<body>`, и тот тег `<h1>`, который находится внутри `div`-контейнера (поскольку этот `<h1>` является первым дочерним элементом `div`).

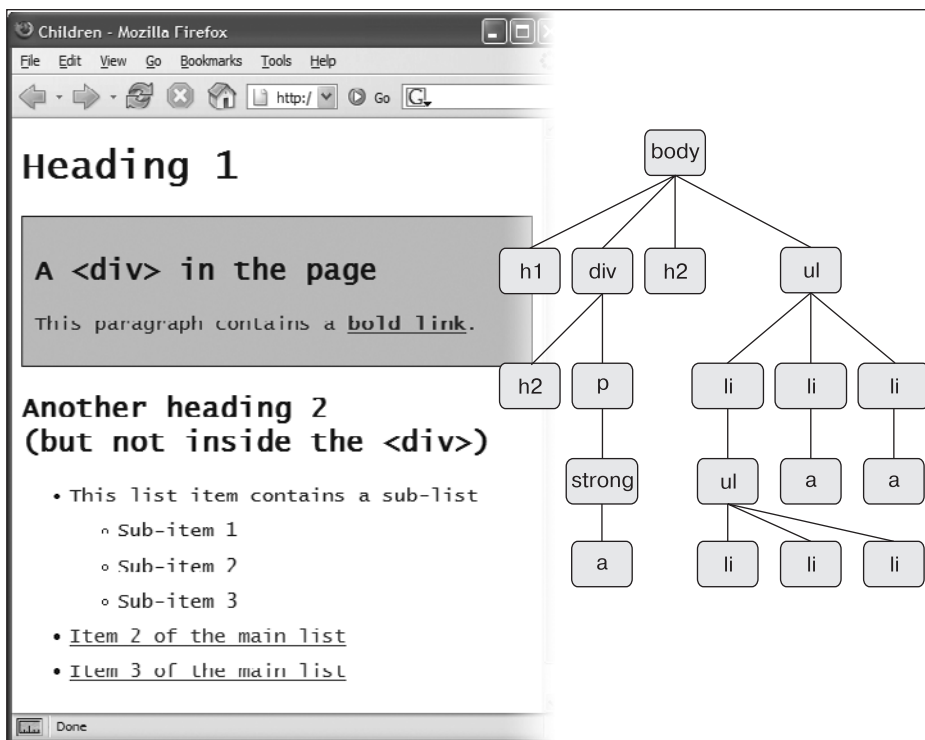


Рис. 3.6. Диаграмма в виде дерева (справа) показывает отношения между HTML-тегами (слева)

**:last-child.** Этот псевдоэлемент похож на рассмотренный ранее `:first-child`, но выбирает последний дочерний элемент. Например, для придания стиля последнему элементу списка нужно воспользоваться селектором `ul :last-child` (рис. 3.7). Имеющийся в CSS широкий диапазон селекторов дочерних элементов предоставляет вам различные способы выбора дочерних элементов. Эти селекторы хорошо подходят для выделения первого, последнего или чередующихся элементов списка.

**:nth-child.** Этот групповой псевдоэлемент селектора очень полезен. С его помощью можно легко и просто стилизовать каждую вторую строку в таблице, каждый третий элемент в списке или придать свой стиль любому сочетанию чередующихся дочерних элементов (см. рис. 3.7). Для определения, какой из дочерних элементов нужно выбирать, этот псевдоэлемент селектора требует значения.

Самым простым выбором является ключевое слово — либо `odd` (нечетный), либо `even` (четный), — позволяющее выбрать чередующиеся нечетные или четные дочерние элементы. Например, если нужно предоставить один фоновый цвет для каждой четной строки таблицы и другой фоновый цвет для каждой строки с нечетным номером, можно создать два следующих стиля:

```
tr:nth-child(odd) { background-color: #D9F0FF; }
tr:nth-child(even) { background-color: #FFFFFF; }
```

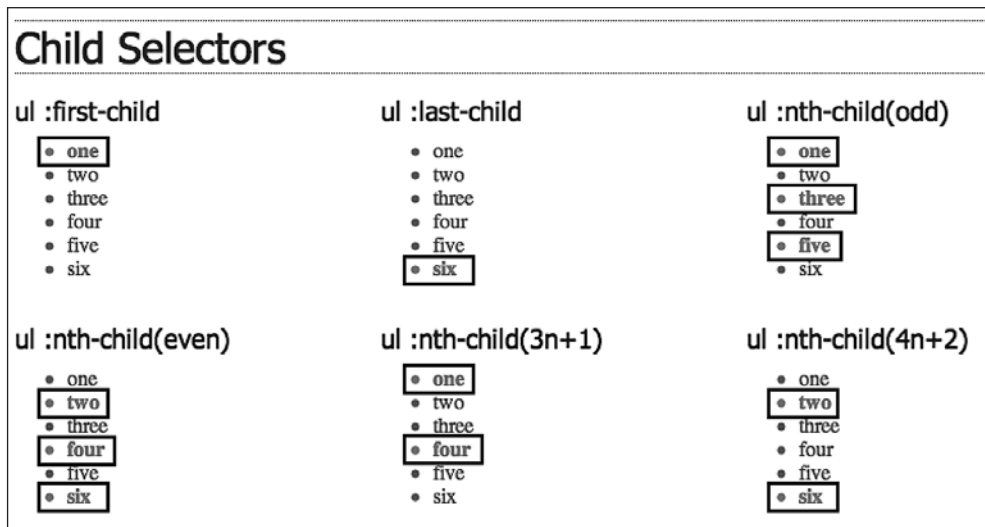


Рис. 3.7. Селекторы дочерних элементов

Теперь у вас есть действительно простой способ для раскрашивания чередующихся строк таблицы (рис. 3.8). Но у псевдоэлемента `:nth-child()` в рукаве спрятано еще немало козырей. Можно также выбрать, скажем, каждый третий дочерний элемент в серии, начиная со второго дочернего элемента. Представим, к примеру, что вам нужно выделить каждую третью ячейку таблицы (тег `<td>`) внутри строки, начиная со второй ячейки таблицы (см. рис. 3.8). Для этого необходимо применить следующий стиль:

```
tr td:nth-child(3n+2) { background-color:#900; }
```

Обычно число перед  $n$  (в данном случае 3) представляет тот дочерний элемент, который будет выделен следующим по счету. Следовательно,  $3n$  означает каждый третий элемент, а  $4n$  – каждый четвертый элемент. Знак «плюс» и следующее за ним число (+2 в данном примере) показывают, с какого элемента нужно начинать выделение, таким образом, +2 означает, что нужно начинать со второго дочернего элемента, а +5 означает, что нужно начинать выделение с пятого дочернего элемента. Следовательно, использование псевдоэлемента `:nth-child(5n + 4)` приведет к выделению каждого пятого дочернего элемента, начиная с четвертого дочернего элемента.

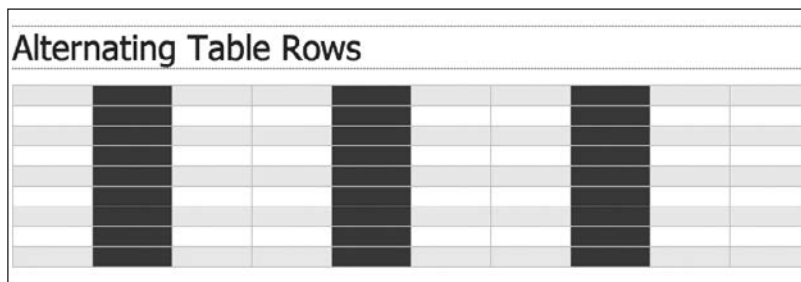


Рис. 3.8. Простой способ создания полос в таблице

Вы даже можете создавать полосы, выделяя чередующиеся столбцы, нацеливаясь на каждый второй тег `<td>` в строке или, как в данном случае, на каждый третий столбец, начиная со второго. Вот это точность!

## Селекторы типов дочерних элементов

CSS3 включает селектор, который работает во многом похоже на селектор дочерних элементов, рассмотренный в предыдущем разделе, но применяется к дочерним элементам с HTML-тегом определенного типа. Предположим, к примеру, что вам нужно определенным образом отформатировать первый абзац на боковой панели, но только на тех страницах, где эта боковая панель начинается с тега `<h2>`, и на других страницах, где она начинается с тега `<p>`. Псевдоэлемент `:first-child` для выбора этого абзаца применять нельзя, потому что в некоторых случаях абзац является вторым дочерним элементом (который следует за `<h2>`). Но тем не менее он всегда является первым абзацем (тегом `<p>`) на этой боковой панели, даже если перед ним следуют какие-нибудь другие теги, следовательно, его можно выбрать с помощью селектора типа, который называется `:first-of-type`.

### ПРИМЕЧАНИЕ

---

`:last-child`, `:first-of-type` и `:nth-child()` поддерживаются всеми современными браузерами, включая Internet Explorer 9 и выше. Но в Internet Explorer 8 они не работают.

---

**:first-of-type.** Работает так же, как и `:first-child`, но применяется к дочернему элементу, имеющему определенный тег. Предположим, например, что у вас есть боковая панель с классом `sidebar`. Для придания стиливого оформления первому абзацу этой боковой панели используется следующий селектор:

```
.sidebar p:first-of-type
```

Обратите внимание на `p` в `p:first-of-type`. Этот элемент обозначает тег, который вы собираетесь отформатировать.

**:last-of-type.** Работает так же, как и `:last-child`, но применяется к последнему экземпляру тега определенного типа. Например, если нужно на боковой панели `div` определенным образом отформатировать последний абзац, но вы не уверены в отсутствии каких-нибудь других тегов, следующих за абзацем (например, списка, заголовка или рисунка). Этот стиль имеет следующий вид:

```
.sidebar p:last-of-type
```

### ПРИМЕЧАНИЕ

---

Следует помнить, что эти селекторы типа также должны быть дочерними по отношению к конкретному тегу. Следовательно, `p:first-of-type` означает «первый дочерний элемент, являющийся тегом абзаца».

---

**:nth-of-type.** Работает так же, как и `:nth-child()`, но применяется к чередующимся дочерним элементам, имеющим определенный тег. Этот селектор может пригодиться при наличии больших абзацев текста с большим количеством фотографий. Тег `<img>` является встраиваемым тегом, поэтому у вас может быть `<p>`-тег с несколькими `<img>`-тегами внутри. И если вам захочется чередовать появление изображений то слева, то справа, как показано на рис. 3.9, то это можно сделать с помощью двух стилей:

```
img:nth-of-type(odd) { float: left; }
img:nth-of-type(even) { float: right; }
```



**Рис. 3.9.** Используя селектор `:nth-of-type()`, можно легко выбрать любые следующие изображения внутри тега, чередуя их расположение то справа, то слева

Как видите, для `:nth-of-type()` используются такие же ключевые слова (`odd` или `even`) и формулы (например,  $2n + 1$ ), как и для `:nth-child()`.

`:nth-of-type()` можно также использовать для чередующихся строк таблицы:

```
tr:nth-of-type(odd) { background-color: #D9F0FF; }
tr:nth-of-type(even) { background-color: #FFFFFF; }
```

Что касается CSS-селекторов, то здесь всегда имеется более одного способа добраться до HTML-тега, обычно более пяти! (Нужно лишь иметь в виду, что ни один из этих селекторов типов дочерних элементов не поддерживается в Internet Explorer 8 и ниже.)

## ЧАСТО ЗАДАВАЕМЫЕ ВОПРОСЫ

### Придание спискам приглядного внешнего вида

*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 этим справиться. Такая ситуация встречается на большинстве сайтов. В любом маркированном списке присутствует один или несколько пунктов — элементов списка (см. рис. 3.6). Здесь можно использовать селекторы дочерних элемен-

тов, чтобы визуально упорядочить данные в категориях (пунктах) и подкатегориях (подпунктах). Вы форматируете элементы первого уровня списка одним способом, 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



### ЧАСТО ЗАДАВАЕМЫЕ ВОПРОСЫ

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`. Затем для задания стиля дочерним `<li>`-тегам любых последующих вложенных `<ul>`-тегов воспользуйтесь таким селекто-

ром: `ul.mainList > li > ul > li`. (Селекторы таких потомков, как `ul li`, выбирают элементы списка всех неупорядоченных списков на странице: и вложенных, и невложенных.)

Вам нужно будет обратить внимание и на понятие наследования, которое будет рассмотрено в следующей главе. Как правило, конкретные CSS-свойства, примененные к одному тегу, наследуются тегами, которые находятся внутри этого тега. Поэтому, если даже вы используете селектор дочернего элемента, нацеленный на дочерний элемент одного тега, свойства могут перейти на другие теги внутри этого дочернего элемента. Одним из способов избежать такого развития событий будет применение селектора `:not()`.

## Селекторы смежных элементов одного уровня

Родительско-дочерние отношения — не единственная форма родственных связей в дереве HTML. Иногда требуется выбрать тег, относящийся к группе смежных элементов одного уровня и имеющих общего родителя. Тег, который следует сразу же за другим тегом, в HTML называется *смежным элементом того же уровня*. На рис. 3.6 тег `<div>` является смежным по отношению к `<h1>`, а тег `<p>` — по отношению к `<h2>` и т. д.

Используя селектор смежных элементов, можно, например, придать первому абзацу после каждого заголовка форматирование, отличное от следующих абзацев. Предположим, вы хотите удалить отступ, который автоматически появляется перед тегом `<p>`, чтобы между заголовком и тегом не было никакого промежутка. Или желаете придать абзацу как небольшому вводному описанию другой цвет и размер шрифта.

Селектор смежных элементов использует знак `+` для соединения одного элемента с другим. Поэтому, чтобы выбрать все первые абзацы, следующие за любым `<h2>`, используйте селектор `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 следует сразу же за смежным для него тегом `<h2>`.

Есть и другой селектор смежных элементов, который называется *общим сборным селектором смежных элементов*. Он обозначается знаком тильды (`~`) и означает следующее: «Нужно выбрать все смежные элементы этого типа». Например, если селектор `h2 + p` задает выбор отдельного `<p>`-тега, который следует сразу же за тегом `<h2>`, то селектор `h2 ~ p` определяет выбор всех смежных `<p>`-тегов, являющихся смежными элементами (то есть находящихся на одном и том же уровне) по отношению к заголовку `h2`. Честно говоря, вы можете так и не найти этому селектору достойного применения, но в CSS, в принципе, весьма разнообразный синтаксис.

## Селектор `:not()`

Селектор `:not()`, также известный как псевдокласс отрицания, позволяет выбрать что-нибудь, отличное от другого. Например, можно применить класс к абзацу — `<p class="classy">` — и создать селектор класса CSS, чтобы отформатировать этот абзац:

```
.classy { color: red; }
```

А что делать, если понадобится выбрать все абзацы, за исключением тех, что имеют класс `classy`? Именно здесь и пригодится селектор `:not()`. Чтобы показать, что не нужно выбирать, CSS-селектор помещается в круглые скобки. Например:

```
p:not(.classy) { color: blue; }
```

Этот стиль задает тексту синий цвет во всех абзацах, к которым не применялся класс `classy`.

Селектор `:not()` может быть полезен при использовании селекторов атрибутов. Например, ранее было показано, что селектор атрибутов можно использовать, чтобы подобрать все ссылки, указывающие за пределы вашего веб-сайта:

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

Как вы уже, наверное, заметили, этот селектор не выбирает конкретно все ссылки, указывающие за пределы вашего сайта, он просто выбирает все ссылки, которые используют абсолютные URL-адреса, то есть URL, которые начинаются с `http://`. Для многих сайтов это одно и то же, поскольку они используют для указания на другие страницы этого же сайта ссылки относительно документа или главной страницы сайта, а для указания ссылок на другие сайты применяются абсолютные URL-адреса. Но в некоторых случаях абсолютные URL-адреса могут задействоваться для указания на страницу внутри вашего сайта.

Например, некоторые системы управления контентом (в частности, WordPress) используют для указания на публикации блогов внутри сайта абсолютные URL-адреса. В этом случае, если нужно придать стиль ссылкам, которые ведут за пределы вашего сайта, следует усовершенствовать основной селектор атрибута, задействовав также селектор `:not()`. Предположим, например, что доменное имя вашего сайта `mysite.com`. Для выбора ссылки, которая указывает за пределы вашего сайта, нужно выбрать все абсолютные ссылки, не указывающие на домен `mysite.com`. Вот как это можно сделать:

```
a[href^="http://"]:not([href^="http://mysite.com"])
```

Если перевести на нормальный язык, этот селектор говорит: «Нужно выбрать все ссылки, чей атрибут `href` начинается с `http://`, но не те, которые начинаются с `http://mysite.com`». Если вспомнить предыдущий материал, в селекторе атрибута `^=` означает «начинается с». То же самое можно написать еще короче:

```
a[href^="http://"]:not([href*="mysite.com"])
```

В селекторе атрибута `*=` означает «содержит», тем самым любой абсолютный URL, содержащий `mysite.com`, будет исключен. А такие адреса, как `http://www.mysite.com` и `http://mysite.com`, будут включены.

В отношении селектора `:not()` действуют несколько ограничений.

- С селектором `:not()` можно использовать только простые селекторы. Другими словами, можно применять селекторы элементов (такие как `html` или `p`), универсальный селектор (`*`), классы (например `.footer`), идентификаторы ID (например, `#banner`) или псевдоклассы (`:hover`, `:checked`, `:first-child` и т. д.). Таким образом, все следующие селекторы можно считать правильными:

```
.footnote:not(div)
img:not(.portrait)
div:not(#banner)
li:not(:first-child)
```

- Нельзя использовать селекторы-потомки (такие как `div p a`), псевдоэлементы (такие как `::first-line`), групповые селекторы или комбинации (такие как смежный селектор одного уровня `h2 + p`).
- Нельзя в одной строке применять несколько `:not()`-селекторов. Например, следующий код будет неправильным:

```
a[href^="http://"]:not([href*="google.com"]):not([href="yahoo.com"])
```

Другими словами, `:not()` с селектором можно использовать только один раз.

## Обучающий урок: примеры использования селекторов

В оставшейся части этой главы мы потренируемся в создании разных селекторов. Вы 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 результа-

тов 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 добиваются требуемых результатов по созданию стиля отдельно взятой веб-страницы, то 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. 3.10). 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.

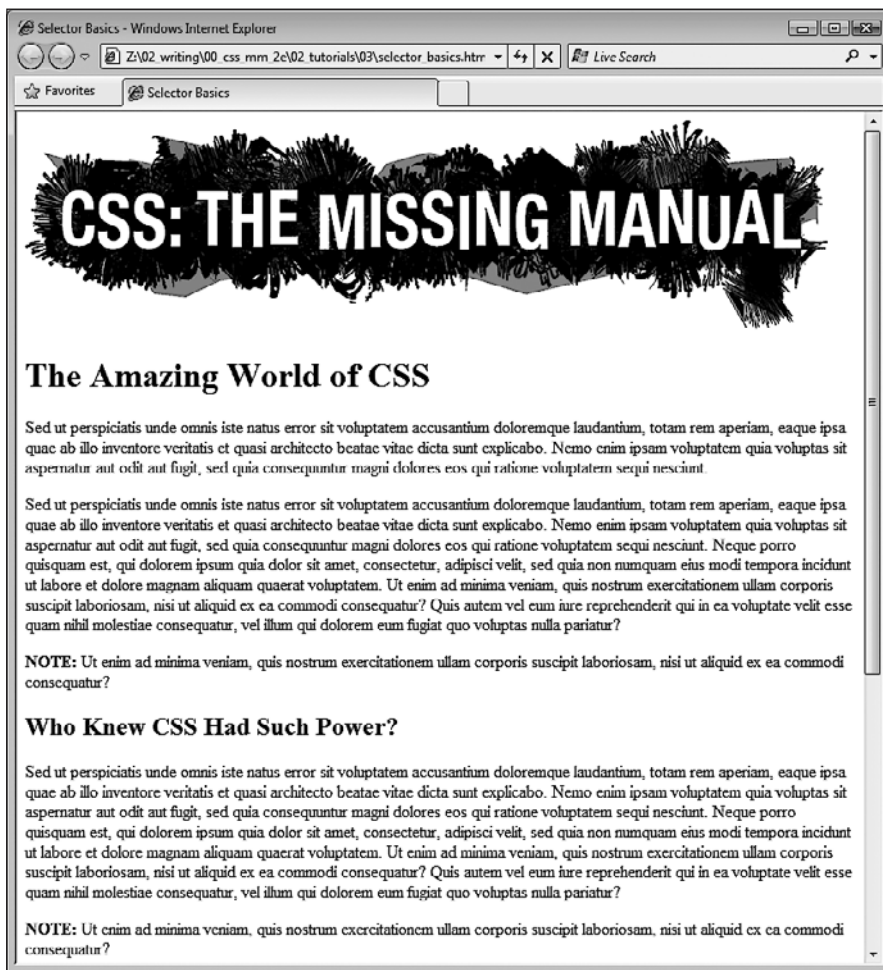


Рис. 3.10. Простой HTML-текст смотрится в браузере чересчур холодно и монотонно

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

Это открывающий и закрывающий теги внутренней таблицы стилей. Очень полезно набирать их одновременно, чтобы случайно не забыть о закрывающем.

Они вместе сообщают браузеру, что между ними находятся команды языка CSS. HTML-код теперь должен выглядеть следующим образом (код, который мы только что добавили, выделен полужирным шрифтом):

```
<title>Selector Basics</title>
<style>

</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).

---

#### ПРИМЕЧАНИЕ

Если используются типы документов HTML 4.01 или XHTML (а не HTML5), к открывающему тегу `style` нужно добавить `type="text/css"` attribute:

```
<style type="text/css">
```

---

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

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 форматированный стиль — цвет, размер, начертание шрифта, отступ:

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

Нажмите клавишу **Enter**, чтобы поместить каждое CSS-свойство на отдельной строке. Кроме того, полезно сделать отступы клавишей табуляции, чтобы улучшить визуальное восприятие кода CSS (некоторые разработчики вместо табуляции используют два пробела, а вы можете выбрать то, что вам больше нравится).

---

#### 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 в гл. 6.

---

Работа над таблицей стилей закончена.

5. 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, 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 следующее:

```
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. Щелкните на пустой строке между открывающей `{` и закрывающей `}` фигурными скобками 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;
```

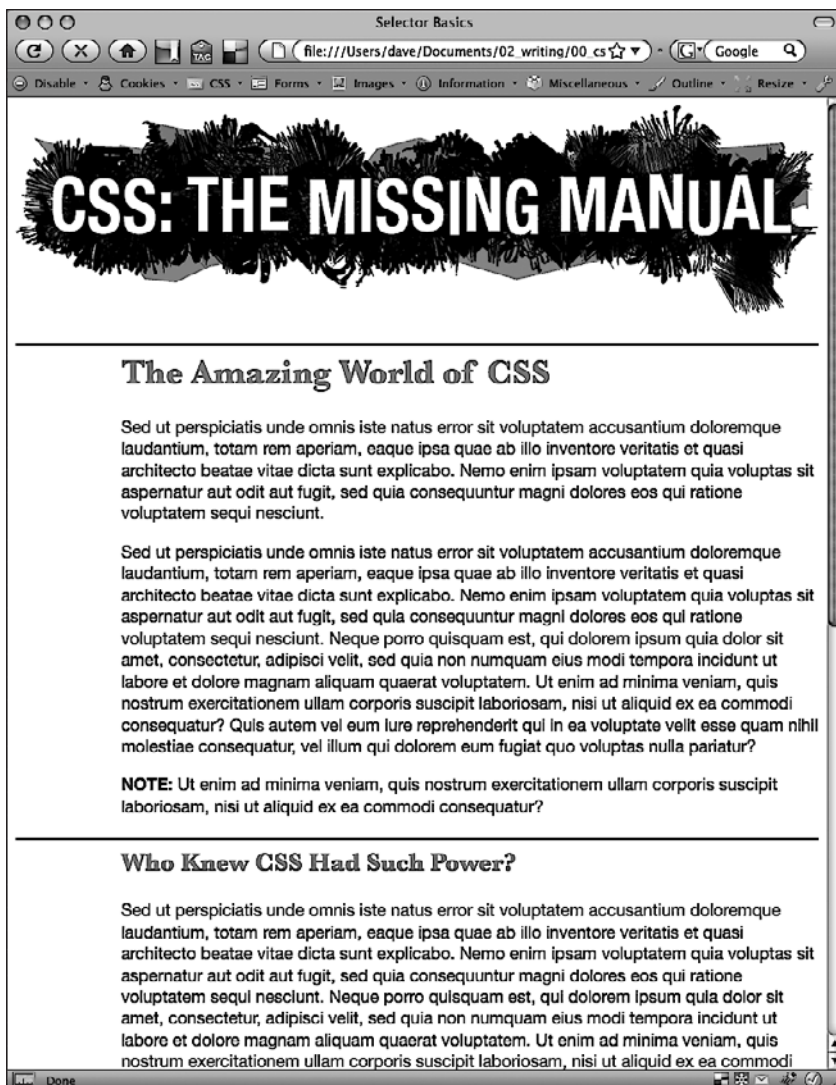
Здесь мы задаем цвет и тип шрифта для заголовков, добавляем линию границы над заголовками, устанавливаем отступы слева и сверху, используя свойство `padding`. Если говорить кратко, то это свойство добавляет дополнительное пространство от краев элемента без воздействия на фон или рамку, то есть вы убираете текст заголовка от левого и верхнего краев, не затрагивая линию рамки, которая простирается по всей странице.

4. 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 сверху (рис. 3.11).

## 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.



**Рис. 3.11.** Простой селектор типа может коренным образом преобразовать внешний вид всех вхождений формируемого тега, быстро выполнив стилизацию текста веб-страницы

- Щелкните кнопкой мыши после закрывающей фигурной скобки группового селектора `h1`, `h2`, `h3`, нажмите клавишу `Enter` и введите следующее:

```
.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.



- Щелкните на пустой строке между открывающей { и закрывающей } фигурными скобками 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 вам знакомы, однако background-color будет, скорее всего, новым. Как и вытекает из его названия, оно добавляет цвет к фону элемента. Весь стиль должен иметь следующий вид:

```
.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 возымеют никакого эффекта на веб-странице, пока стиль не будет применен к коду HTML.

- В HTML-коде веб-страницы найдите вхождение `<p>`, текст которого начинается со слова Note: и находится внутри тега `<strong>`. Чтобы применить стиливой класс к этому тегу, просто добавьте атрибут `class`, за которым должно следовать название, в данном случае `.note`.
- Щелкните кнопкой мыши сразу за буквой `p` 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 стиливого 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 мы дали классу, его можно применить 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">`.

---

- Сохраните файл и просмотрите веб-страницу в браузере. Абзац с примечанием красиво подсвечен на странице (рис. 3.12).



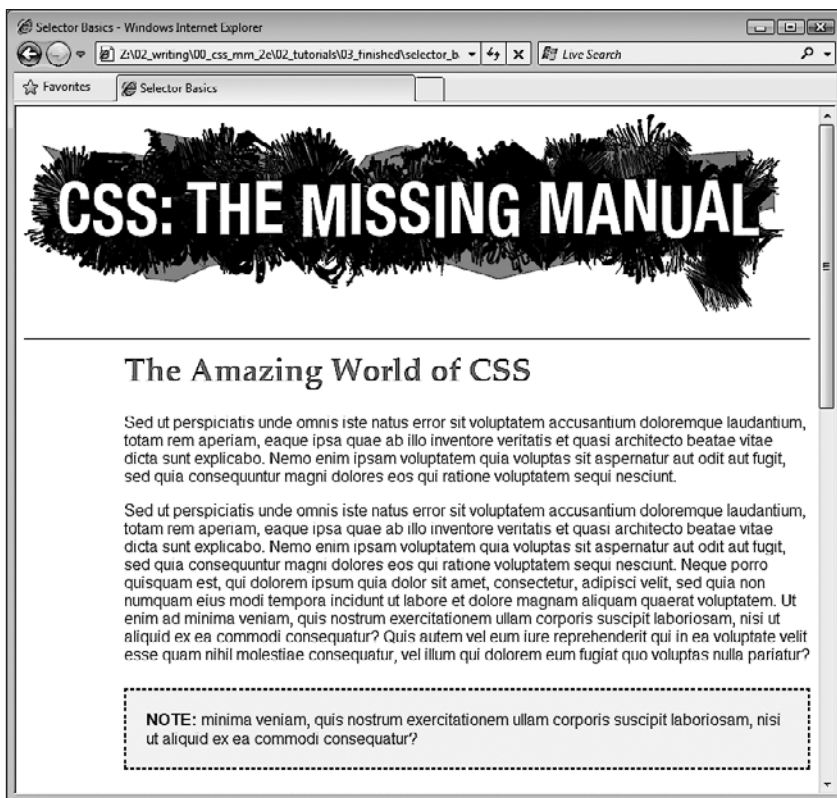


Рис. 3.12. С помощью селекторов классов вы можете выполнить точное, детальное форматирование элементов веб-страницы

#### ПРИМЕЧАНИЕ

Если ваша веб-страница не похожа на изображенную на рис. 3.12, 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 полужирным 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 `<strong>` внутри записей. Но вы наверняка забудете применить класс, если у вас много таких страниц с записями.

Лучший способ решить эту проблему — создать селектор потомков, который относится только к нужным нам тегам `<strong>`. 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 — `tag strong` — это и есть элемент, который вы хотите отформатировать. При этом стиль отформатирует `<strong>` только в том случае, если он расположен внутри другого тега, к которому применен класс `.note`. Стиль не возымеет никакого эффекта на теги `<strong>`, 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 применяются повсеместно, поэтому у вас будет возможность получить о них более полное представление.

## Создание ID-селекторов

Вы можете применить селекторы классов ко многим элементам веб-страницы. Например, ранее вы создавали класс `.note` и применили его к двум абзацам, хотя могли бы применить и к большому количеству 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. Как уже ранее упоминалось, многие веб-дизайнеры избегают использования ID-селекторов (причины будут рассмотрены чуть позже). Но это не означает, что вы не должны знать, как их применять.

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. Мы применим 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 выравнивают все посередине страницы (центрируют).

### 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; в противном случае любые другие элементы с идентификатором article получат такое же форматирование. 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 селектор одного уровня 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 следующие свойства к стилю:

```
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) управляет промежутком между строками в абзаце (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. 3.13). 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>`.

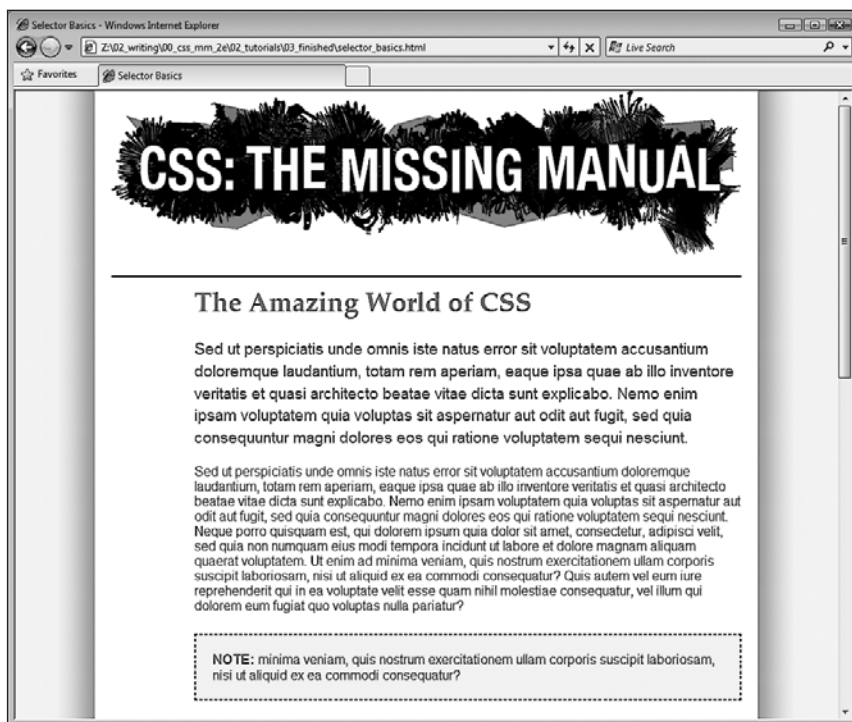


Рис. 3.13. Теперь веб-страница имеет законченный вид

#### ПРИМЕЧАНИЕ

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

#### ПРИМЕЧАНИЕ

Окончательную версию созданной в этой главе экспериментальной веб-страницы вы можете найти в папке 03\_finished 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.

# 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 него, 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, `<em>` и `<strong>` 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 прауродителями).

---

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 `<em>`, выделяющий фрагмент текста, или `<a>`, который добавляет гиперссылку. Если вы создали стиль, форматировующий данный абзац шрифтом Arial размером 24 пиксела фиолетового цвета, было бы странно, если бы внутри `<em>` отобразился прежний стиль. Вам пришлось бы создавать для форматирования еще один стиль.

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

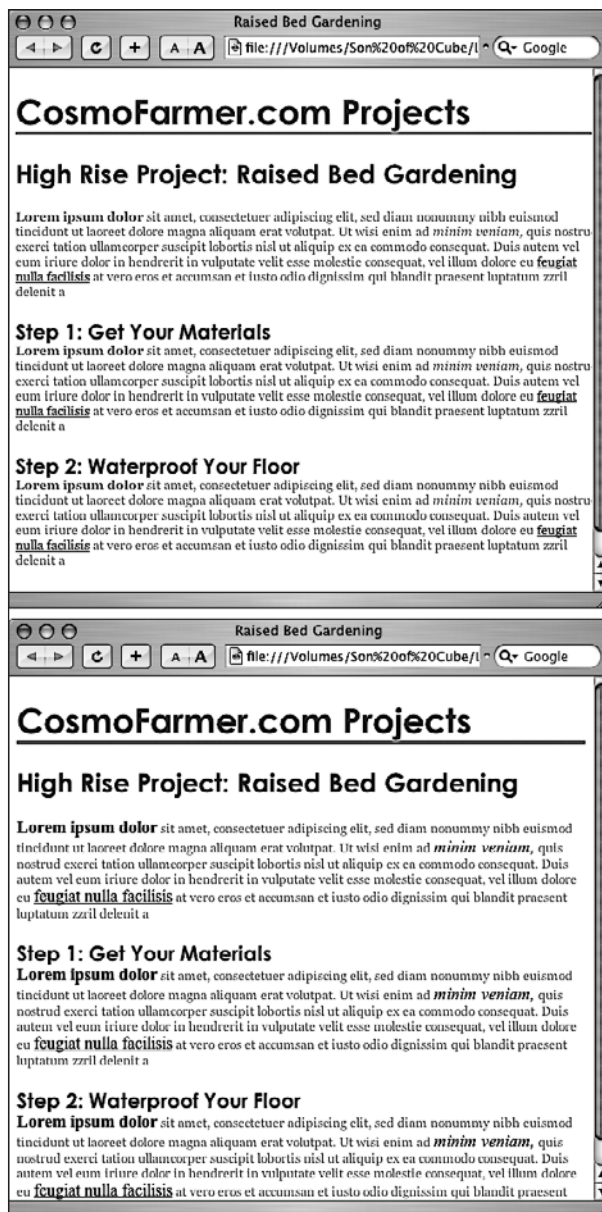


Рис 4.1. Наследование позволяет вложенным тегам копировать атрибуты окружающих их родительских тегов

Если бы наследования не существовало, то веб-страница выглядела бы так, как показано на рис. 4.1, *внизу*. 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. раздел «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 `<div>` (см. раздел «Селекторы классов: точное управление» гл. 3) для определения таких фрагментов, как шапка, навигационное меню, нижняя часть страницы, или, если вы используете элементы HTML5, можно задействовать один из новых элементов деления на разделы, например `<header>`, `<aside>`, `<footer>` или `<article>`. Применяя стиль к внешнему тегу, 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 `<div>`, `<header>`, `<article>` или к другим элементам деления на разделы. Все теги, заключенные внутри, в том числе `<p>`, `<h1>` и т. д., унаследуют этот цвет шрифта.

## 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.

---

Ниже описаны 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 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 показаны синим цветом с подчеркиванием.



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

---

**ПРИМЕЧАНИЕ**

Очень полезно устранять встроенные стили браузеров — это упростит создание сайтов, совместимых с различными типами браузеров. В гл. 5 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 `<u1>`).

---

**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. Затем сделаем 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

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>`. 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>`.

---

**ПРИМЕЧАНИЕ**

Если используются типы документов HTML 4.01 или XHTML (а не HTML5, как в данном учебнике), к открывающему тегу `style` нужно добавить атрибут `type="text/css"`:

```
<style type="text/css">
```

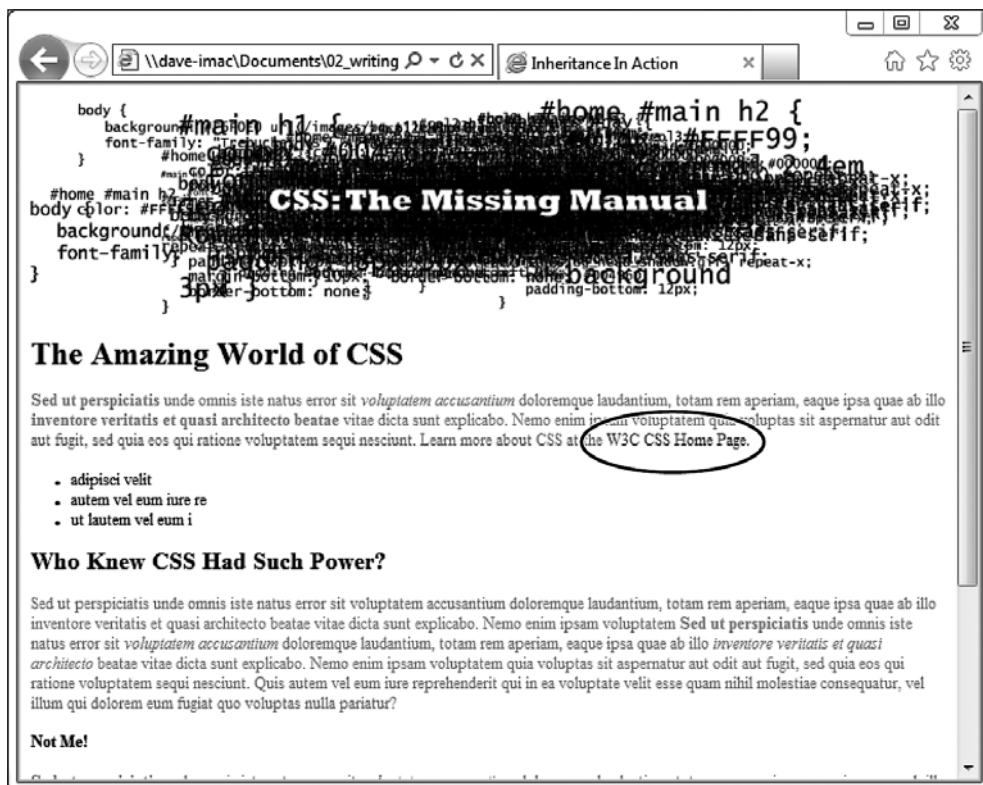
---

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. На 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. Но что это? Ссылка в конце первого абзаца осталась синей (показана в овале). Почему так получилось, вы узнаете в следующей главе

Обратите внимание на то, как этот стиль `<p>` воздействует на другие теги. Они вложены в `<p>` и также меняют цвет. Например, текст, заключенный в `<em>` и `<strong>`

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 `<em>`, `<a>` и `<strong>` 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.

Но вы увидите, что ссылка в конце первого абзаца не изменила свой цвет, оставшись, как ей и положено, синей. Как уже упоминалось, для определенных элементов у браузеров есть свои собственные стили, поэтому наследование к ним не применяется. Дополнительные сведения о подобном поведении приведены в гл. 5.

## 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;
}
```

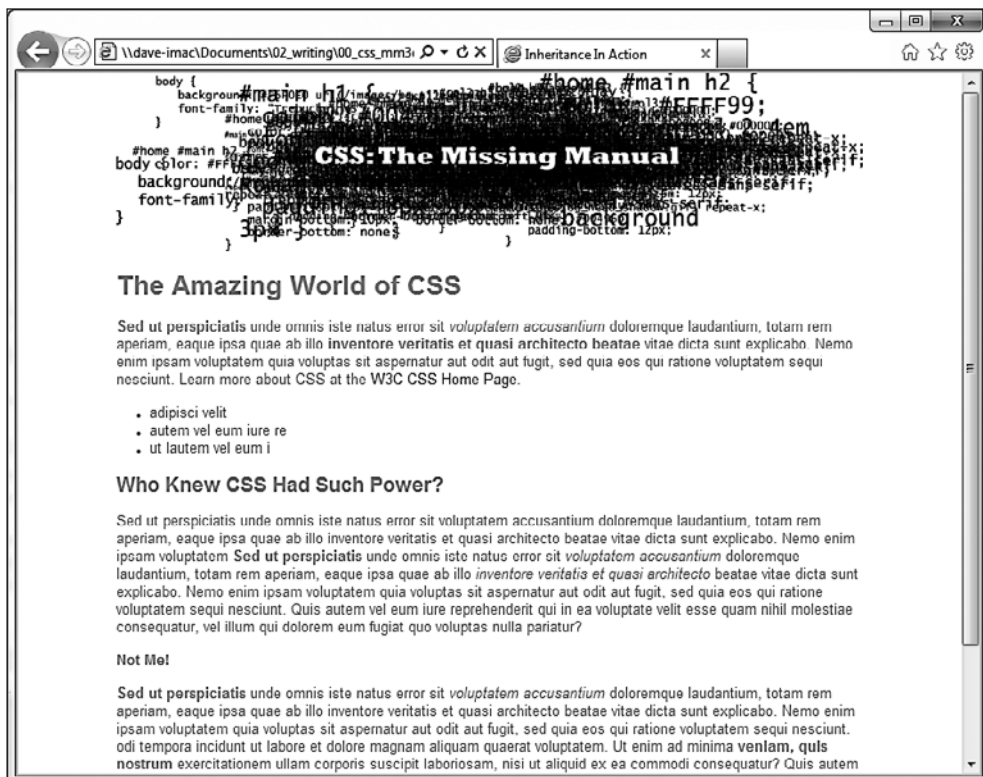
Этот класс устанавливает начертание, размер и цвет шрифта. Он также задает ширину и центрирует стиль на странице (мы рассматривали это в предыдущем обучающем примере).

4. 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">`. Соответственно, 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 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 разные классы (или ID), вы можете создавать различные стили для тега `<body>` для разных разделов сайта или разных типов страниц.

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 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:

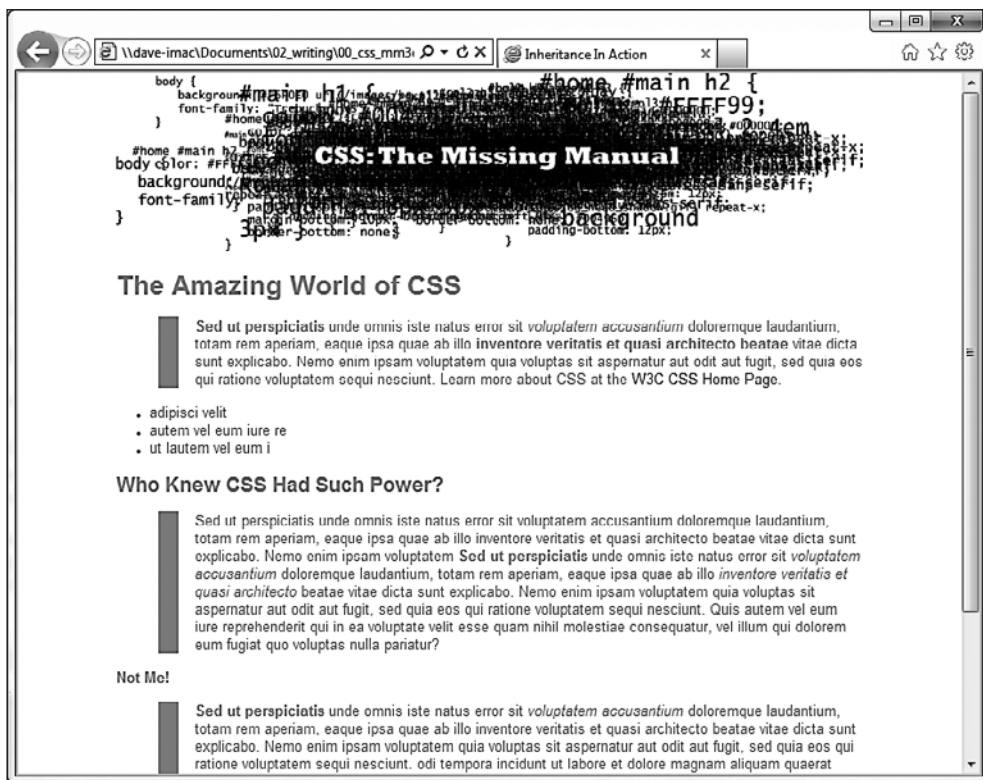
```
p {
  color: #FF6600;
  margin-left: 50px;
  padding-left: 20px;
  border-left: solid 25px #BD8100;
}
```

Свойство `margin-left` создает левый отступ размером 50 пикселей, свойство `padding` создает отступ от границы размером 20 пикселей.

4. Сохраните файл и просмотрите его в браузере.

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





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

Если вы хотите увидеть, что случилось бы, 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).

**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 сложно комбинированным. Например, тег `<body>` может передать формирующие 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 особая 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. раздел «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 изучите основы CSS.

---

## Каскадность стилей

*Каскадность* — ряд правил, определяющих, какие именно стилиевые свойства необходимы элементам веб-страницы, то есть задающих последовательность применения многократно определенных стилей. Другими словами, каскадность определяет, 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, вы применили к абзацу стиль с помощью атрибута class, а также создали стиль для тега <p>, и оба стиля применяются к этому абзацу).

## 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 стиливые 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 <strong>. CSS-код выглядит следующим образом:

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

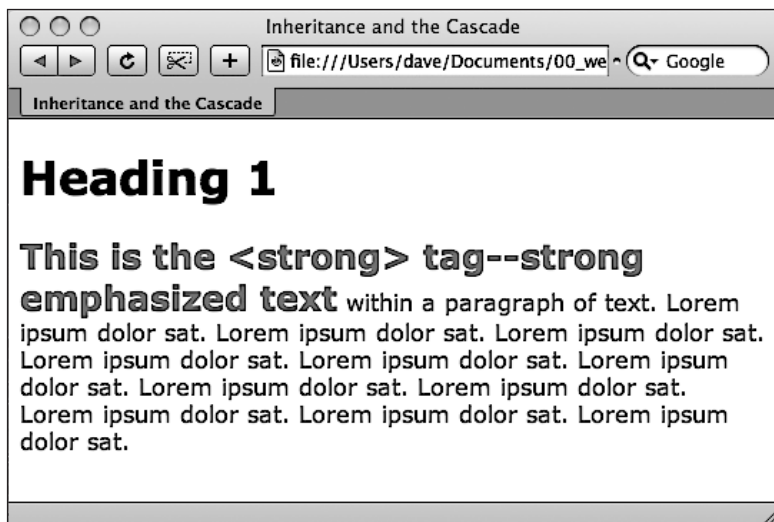


Рис. 5.1. Благодаря наследованию можно воспроизводить один тег несколькими стилями

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

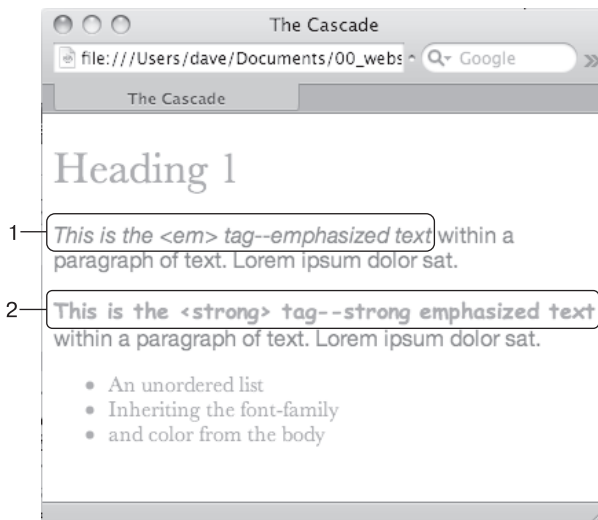
Как видно из предыдущего примера, комбинирование стилей различных тегов с применением наследования создает полный стилиевой пакет. Но что произойдет в случае конфликта унаследованных свойств CSS? Представьте себе страницу, где вы установили красный цвет шрифта для тега `body` и зеленый цвет шрифта для тега абзаца.

На рис. 5.1 `<strong>` 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 `<strong>`. Тег `<strong>` наследует стиль как тега `body`, так и тега абзаца. Так какой текст внутри тега `<strong>` – красный или зеленый? Итак, у нас есть победитель: зеленый цвет от стиля 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 `<strong>` и, можно сказать, находится по соседству с ним. Стилиевые 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).



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

На рис. 5.2 тег `<em>` наследует начертание и цвет от `<body>` и от `<p>`. Однако стили `<body>` и `<p>` имеют различные типы шрифтов и цвет, тег `<em>` в конечном счете использует тот шрифт и цвет, которые определены ближайшим тегом-предком — `<p>`. Когда стиль относится непосредственно к тегу, браузер применяет его и игнорирует свойства конфликтующих наследуемых стилей (см. рис. 5.1, позиция 1).

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 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 `<strong>`. Стиль абзаца `<p>` является определенным по отношению к `<body>`, но `<strong>` — еще более конкретный, чем все остальные. Он форматирует только теги `<strong>`, отменяя любые конфликтующие свойства, унаследованные от других тегов (см. рис. 5.2, позиция 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 стилиевой класс. Так, для одного тега абзаца `<p>` непосредственно определено три различных явных формирующих 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 стилиевой 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 несколько раз в таблице.** Это может быть групповой селектор (см. раздел «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 `<div>` будут применены оба стиля.
- **С веб-страницей связано несколько таблиц, и в каждой из них содержится одинаковое название стиля.** Стили с одинаковыми названиями могут появляться во внешней таблице стилей и во внутренней таблице стилей. Или же один и тот же стиль может появиться в нескольких внешних таблицах стилей, каждая из которых связана с одной и той же страницей.
- **Единственный тег веб-страницы может быть объектом воздействия сложных селекторов.** Это обычная ситуация, когда вы используете производные селекторы (см. раздел «Селекторы классов: точное управление» гл. 3). Допустим, на веб-странице имеется тег `<div>` (например: `<div id="mainContent">`) и внутри него 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, которые принадлежат 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» данного раздела: 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. Инструменты для облегчения работы» 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 стиливых 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/css3-selectors/#specificity](http://www.w3.org/TR/css3-selectors/#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: ``. 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. раздел «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
<code>p</code>	0	0	1	1
<code>.byline</code>	0	1	0	10
<code>p.byline</code>	0	1	1	11
<code>#banner</code>	1	0	0	100
<code>#banner p</code>	1	0	1	101
<code>#banner .byline</code>	1	1	0	110
<code>a:link</code>	0	1	1	11
<code>p:first-line</code>	0	0	2	2
<code>h2 strong</code>	0	0	2	2
<code>#wrapper #content .byline a:hover</code>	2	2	1	221

## ПРИМЕЧАНИЕ

Наследуемые свойства вообще лишены такого показателя, как значимость. Так, даже если тег унаследует, 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:

```
p .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?*

Попытки определить все входы и выходы унаследованных свойств и конфликтующие стили сбивали с толку многих исследователей. Более того, применение математических приемов для определения специфики стиля как-то не привлекает обычного веб-дизайнера, особенно при использовании огромных таблиц стилей с большим количеством селекторов потомков.

Во всех применяемых в настоящее время веб-браузерах имеется встроенная помощь в виде инспектора. Проще всего провести обследование элемента на странице и всех CSS-настроек, влияющих на его внешний вид с помощью щелчка правой кнопкой мыши (щелчка с нажатой клавишей управления на Mac) на элементе (на заголовке, ссылке, абзаце или изображении) и выборе из контекстного меню пункта Inspect Element (Просмотр кода элемента). Откроется панель (обычно в нижней части веб-страницы), показывающая HTML-код страницы с выделенным выбранным HTML-элементом.

(Чтобы добиться этого в Safari, сначала нужно установить флажок Show Develop Menu (Показывать меню Разработка в строке меню) в окне, открывающемся после выбора пунктов Preferences ► Advanced (Настройки ► Дополнения).)

В правой стороне панели вы увидите стили, примененные к элементу. Обычно это «вычисленный» стиль, итоговая сумма всех CSS-свойств, примененных к элементу путем наследования и каскадирования, или некий синтезированный стиль элемента. Чуть ниже будут показаны стилевые правила, которые применены к элементу, перечисленные в порядке от более конкретных (в верхней части списка) к менее конкретным (в нижней части списка).

Вероятнее всего, в перечне стилей вы увидите, что некоторые свойства зачеркнуты. Это свидетельствует о том, что данное свойство либо не применяется к элементу, либо оно было заменено более конкретным стилем. Чтобы просмотреть два кратких учебных пособия по использованию инструментария разработчика по анализу CSS, зайдите на сайты <https://developers.google.com/chrome-developer-tools/docs/elements-styles> и <http://webdesign.tutsplus.com/tutorials/workflowtutorials/faster-htmlcss-workflow-with-chrome-developertools/>.

Теперь представьте, что таблица стилей имеет следующий вид:

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

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

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, когда

вам абсолютно невозможно обойтись без одного или нескольких стилей, применяемых к одной странице.

#### СОВЕТ

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

### УСТРАНЕНИЕ ОШИБОК

#### 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 классом `.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 классом `nav`, будут отображены зеленовато-голубым цветом.

Обратите внимание, что вы применяете `!important` к отдельному свойству, а не ко всему стилю, поэтому нужно добавить `!important` в конце каждого свойства, которое не должно быть отменено.

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.

## Управление каскадностью

Как вы могли заметить, чем больше CSS-стилей создано, тем выше 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 приведет. Эта проблема обычно связана с механизмом каскадности. Даже когда вы абсолютно уверены в конечном результате, все равно может существовать стиль с большей значимостью.

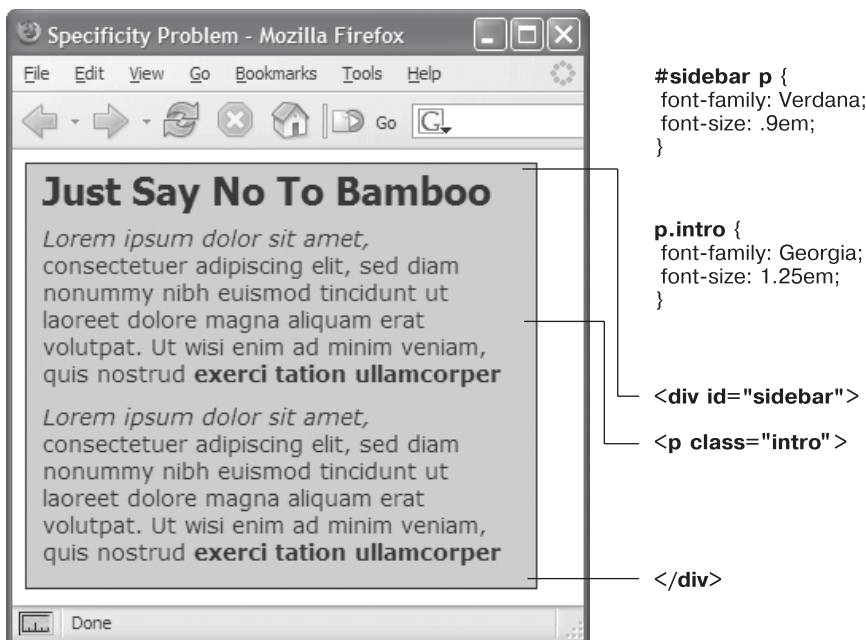
Есть несколько вариантов решения этой проблемы. Можно использовать `!important` (как описано в выделенном блоке выше), чтобы гарантировать применение конкретного свойства. Этот способ не совсем удобен, так как трудно предугадать, что вы не захотите отменить такую значимость свойства. Рассмотрим два других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

### Изменение значимости

На рис. 5.3 приведен пример, когда определенный стиль тега проигрывает в каскадной игре. К счастью, в большинстве случаев можно запросто изменить значимость одного из конфликтующих стилей и прибегнуть к `!important` для по-настоящему безысходных случаев. Здесь два стиля форматировать первый абзац. Стилиевой



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`.



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

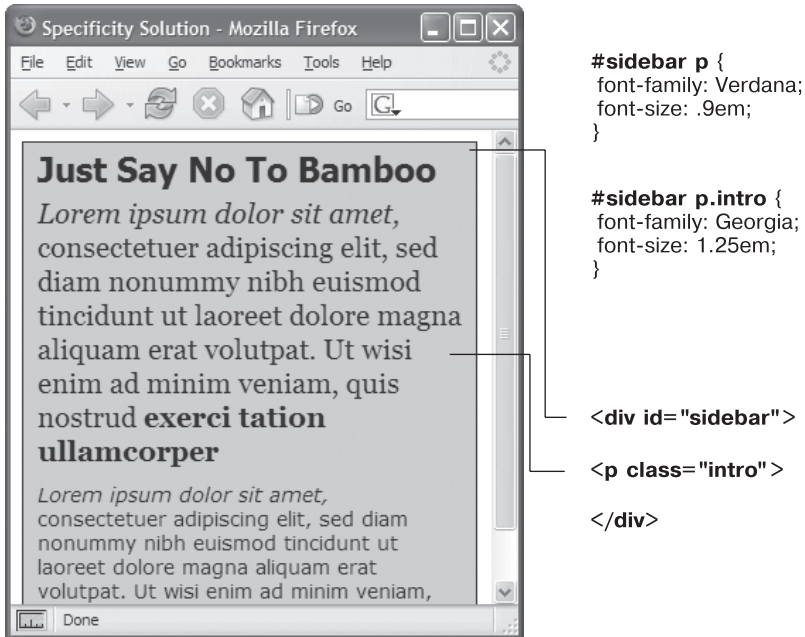
На рис. 5.3 абзац размещен внутри тега `<div>` с идентификатором `#sidebar`. Таким образом, производный селектор `#sidebar p` является более значимым по сравнению со стиливым классом `.intro` (соотношение значимостей — 101 к 10). Вывод: необходимо придать `.intro` большую значимость, добавив перед ним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

Но простое добавление дополнительных селекторов с целью «победить» свойства стиля может привести к так называемым войнам значимости, когда у вас в конечном итоге получаются таблицы стилей, содержащие очень длинные и запутанные имена стилей вроде: `#home #main #story h1`. Нужно избегать стилей такого типа и стараться, чтобы селекторы были как можно короче.

## Выборочная отмена значимости

Можно точно проработать дизайн веб-страниц, выборочно отменяя стили. Предположим, вы создали внешнюю таблицу и назвали ее `styles.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 `styles.css`, но необходимо отменить несколько атрибутов отдельных тегов (`<h1>`, `<p>` и т. д.).



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

Один из способов сделать так заключается в простом создании внутренней таблицы, содержащей стили, которые вы хотите отменить и переопределить. Предположим, в файле `styles.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 и будет иметь размер 36 пикселей (эти параметры определены во внутренней таблице стилей).

---

**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, где 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 `styles.css`. Файл `home.css` будет содержать стили `styles.css`, которые вы хотите переопределить. Для правильной работы файл `home.css` должен быть присоединен *после* `styles.css` в HTML-коде главной веб-страницы:

```
<link rel="stylesheet" href="css/ styles.css">
<link rel="stylesheet" href="css/home.css">
```

---

**СОВЕТ**

Еще один способ выполнить точную постраничную стилизацию веб-страниц основан на использовании различных имен класса 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 рассматривалась в гл. 3.

---

## Как избежать войн значимости

Как уже упоминалось, теперь многие веб-дизайнеры предпочитают вместо ID-селекторов использовать классы. Одна из причин состоит в том, что ID-селекторы обладают очень большой значимостью, в силу чего для их отмены требуется более высокая значимость. Зачастую это приводит к войнам значимости, при которых таблицы стилей загружаются с излишне пространными и сложными селекторами. Суть этой проблемы проще объяснить на примере. Предположим, что в вашей странице есть следующий фрагмент кода HTML:

```
<div id="article">
<p>A paragraph</p>
<p>Another paragraph</p>
<p class="special">A special paragraph</p>
</div>
```

Вы решили, что нужно сделать текст абзаца внутри `div`-контейнера `article` красным, и создали следующий селектор потомка:

```
#article p { color: red; }
```

Но затем вам захотелось, чтобы текст одного абзаца с классом `special` был синим. Если просто создать селектор класса, вы не получите желаемого результата.

```
.special { color: blue; }
```

Как уже говорилось, когда определяется, какое из свойств нужно применить к тегу, веб-браузер использует для разрешения стилевых конфликтов простую математическую формулу: браузеры дают ID-селектору значение 100, селектору класса — значение 10, а селектору тега — значение 1. Поскольку селектор `#article p` составлен из одного идентификатора и одного элемента (суммарный показатель

значимости —101), он заменяет собой простой стиль класса, заставляя вас изменить селектор:

```
#article .special {color: blue; }
```

К сожалению, это изменение является причиной возникновения еще двух проблем. Во-первых, селектор становится длиннее, и во-вторых, теперь этот синий цвет применяется, только когда класс `special` появляется внутри какого-нибудь элемента с идентификатором `article`. Другими словами, если вы скопируете HTML-код `<p class="special">A special paragraph</p>` и вставите его в какое-нибудь другое место страницы, текст уже не будет синим. То есть использование идентификаторов делает селекторы не только длиннее, но и бесполезнее.

А теперь посмотрим, что получится, если просто заменить все идентификаторы классами. Предыдущий код HTML приобретет следующий вид:

```
<div id="article">
<p>A paragraph</p>
<p>Another paragraph</p>
<p class="special">A special paragraph</p>
</div>
```

И код CSS можно заменить следующим:

```
.article p { color: red; }
p.special { color: blue; }
```

Первый стиль — `.article p` — является селектором потомка с показателем значимости 11. Второй стиль `p.special` также имеет показатель 11 (один тег и один класс) и означает «применить следующие свойства к любому абзацу с классом `special`». Теперь, если вырезать этот код HTML и вставить его в какое-нибудь другое место страницы, вы получите синий текст, обусловленный стилем, к чему, собственно, мы и стремились.

Это только один из примеров, но не составит особого труда найти таблицы стилей с абсурдно длинными селекторами вроде `#home #article #sidebar #legal p` и `#home #article #sidebar #legal p.special`.

По сути, использовать идентификаторы нет никакого смысла. Они не дают ничего, что нельзя было бы получить с применением простого селектора класса или селектора тега, а их высокая значимость может только привести к ненужному усложнению таблиц стилей.

---

#### ПРИМЕЧАНИЕ

Более подробные аргументы, почему нужно избегать применения селекторов идентификатора, изложены на странице <http://csswizardry.com/2011/09/when-using-ids-can-be-a-pain-in-the-class>.

---

## Начинаем с чистого листа

Как было сказано ранее, браузеры применяют к тегам 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 определяло бы 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 используют свойство `padding` для создания отступа в маркированных списках, 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 кросс-браузерного несоответствия лучше всего начинать таблицу стилей с чистого листа. Другими словами, удалить встроенное в браузер форматирование и добавить 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, div, span, object, iframe, h1, h2, h3, h4, h5, h6, p, blockquote,
pre, a, abbr, acronym, address, big, cite, code, del, dfn, em, img, ins, kbd,
q, s, samp, small, strike, strong, sub, sup, tt, var, b, u, i, center, dl, dt,
dd, ol, ul, li, fieldset, form, label, legend, table, caption, tbody, tfoot,
thead, tr, th, td, article, aside, canvas, details, embed, figure, figcaption,
footer, header, hgroup, menu, nav, output, ruby, section, summary, time, mark,
audio, video {
margin: 0;
padding: 0;
border: 0;
font-size: 100%;
vertical-align: baseline;
}
```

```
article, aside, details, figcaption, figure, footer, header, hgroup, menu,
nav, section {
display: block;
}
body {
line-height: 1.2;
}
ol {
padding-left: 1.4em;
list-style: decimal;
}
ul {
padding-left: 1.4em;
list-style: square;
}
```

```

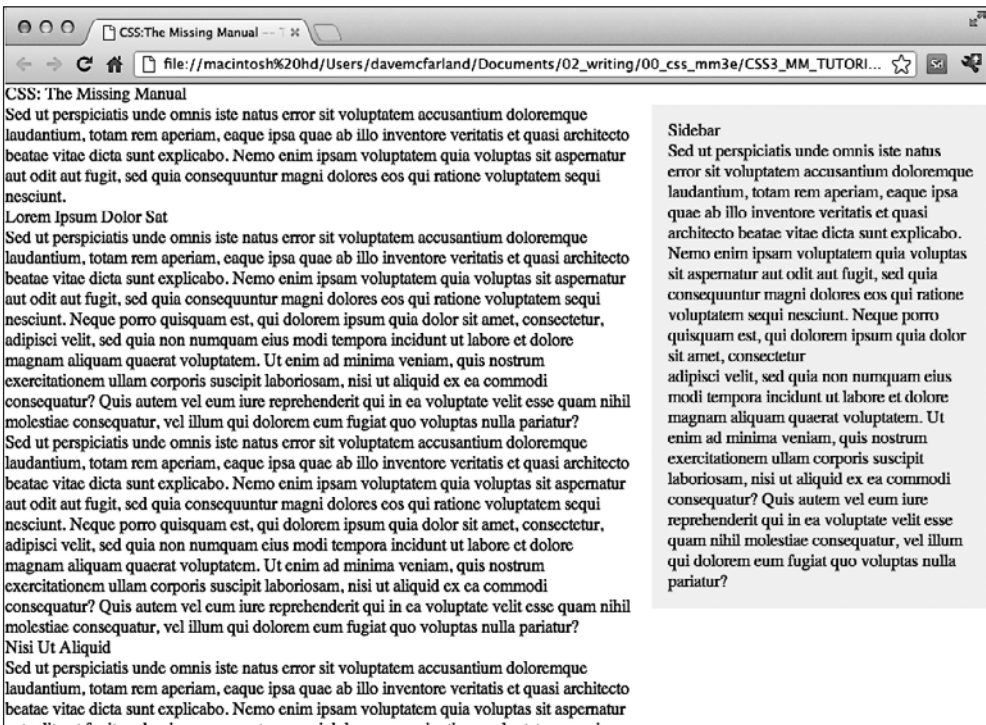
}
table {
border-collapse: collapse;
border-spacing: 0;
}

```

## ПРИМЕЧАНИЕ

Показанный выше код CSS-сброса взят из известного и авторитетного источника, составленного Эриком Мейером (Eric Meyer), который можно найти по адресу <http://meyerweb.com/eric/tools/css/reset>.

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 будут выглядеть 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.



**Рис. 5.5.** Базовый сброс стандартных стилей на этой странице устраняет небольшие различия в том, как разные браузеры отображают основные теги HTML

Второй селектор (`article`, `aside`, `details`...) является еще одним групповым селектором, помогающим устаревшим браузерам правильно отображать новые теги HTML5. Третий селектор стиля (`body`) устанавливает однообразное значение свойства `line-height` (пространства между строками в абзаце). Информация о свойстве `line-height` будет дана в следующей главе.

**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/css3](http://www.sawmac.com/css3), где есть базовый файл для сброса стандартных стилей. Просто скопируйте стили из этого файла и вставьте их в собственные таблицы стилей. Еще один подход к сбросу, обсуждаемый далее в книге, можно найти в учебных файлах гл. 17 в папке 17.

Четвертый и пятый 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 упрощает добавление рамок к ячейкам таблицы (польза от установки этого стиля будет рассмотрена далее).

## 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.

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 несколько 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 — стили классов `.main` и `.sidebar` — создают два столбца, показанные на рис. 5.5. HTML разделен на два тега `<div>`, каждый из которых имеет собственный класс. Стили классов здесь, по существу, размещают два тега `<div>` так, чтобы они отображались бок о бок в виде столбцов (как управлять макетом страницы и создавать столбцы, вы узнаете при изучении ч. 3).



Сначала добавим пару стилей, чтобы улучшить общий вид страницы и ее верхний заголовок.

4. В файле `styles.css` добавьте два стиля в нижней части таблицы стилей за последней скобкой `}` стиля `.sidebar`:

```
body {
  color: #B1967C;
  font-family: "Palatino Linotype", Baskerville, serif;
  padding-top: 115px;
  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;
  margin-bottom: 15px;
}
```

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.



**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>`



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>`.

## 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 `styles.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 создадим стиль для форматирования только тех заголовков второго уровня, которые находятся в главном столбце.

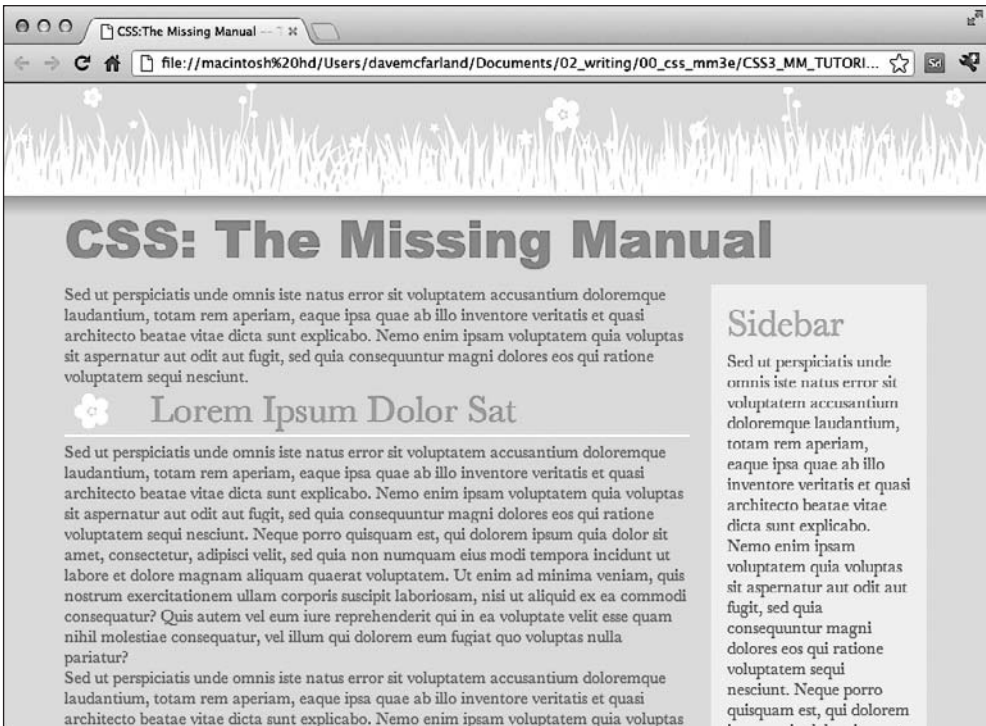
2. Вернитесь к файлу `styles.css`, щелкните кнопкой мыши сразу после окончания стиля тега `<h2>` и нажмите клавишу `Enter`, чтобы создать пустую строку. Добавьте следующий стиль:

```
.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 описанный ранее селектор потомка, формирующий все теги `<h2>`, которые появляются внутри тега с примененным к нему классом `main`. Два столбца текста на этой странице заключаются в теги `<div>` с разными названиями применяемых к ним классов. У большего, расположенного слева столбца класс `main`, поэтому такой особый стиль будет применяться только к тегам `<h2>` внутри этого `<div>`.

Рассматриваемый стиль похож на тот, который вы создали в обучающем примере в гл. 2 в шаге 17. Он добавляет подчеркивание и простой цветок к заголовку. Этот стиль также определяет оранжевый цвет для текста.

3. Сохраните таблицу стилей и снова просмотрите страницу в браузере (рис. 5.7).



**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

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.

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 заголовки второго уровня в основном столбце получают синий цвет текста от стиля `h2`, оранжевый цвет от более специфичного стиля `.main h2` побеждает.

Однако, так как 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.

В таком случае вам нужно установить, почему это произошло, и внести изменения в селекторы CSS, чтобы каскадность приводила к нужному результату.

1. Вернитесь к файлу `styles.css`.

Сейчас создадим 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 стиливой 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 `<div class = "main">`, и 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 селектором потомка и состоит из класса и имени тега. Они добавлены для создания более специфичного стиля, поэтому его свойства стиля решают любой конфликт между ним и стилем `.intro`.

Чтобы заработал стиль `.intro`, необходимо немного «укрепить» его, наделив его селектор большей значимостью.

6. Вернемся к файлу `styles.css` в редакторе и изменим название стиля с `.intro` на `p.intro`.

Убедитесь в том, что между `p` и `.intro` нет пробела. По сути, вы создали связь: `.main p` — один селектор класса и один селектор тега, а также `p.intro` — один тег и один класс. Оба они имеют показатель значимости 11, но поскольку `p.intro` появляется в таблице стилей после `.main p`, именно этот селектор

выигрывает битву и его свойства применяются к абзацу. (Чтобы преодолеть конфликт, можно создать и еще более специфический стиль — `.main .intro`.)

7. Просмотрите страницу в браузере (рис. 5.8).



**Рис. 5.8.** Даже в простой странице с небольшим набором стилей внешний вид тегов зачастую является комбинацией свойств различных стилей

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 стелевой класс не работал в первый раз.

В этой и четырех предыдущих главах мы рассмотрели основы CSS. Теперь пришло время применить полученные знания для создания настоящих полноценных дизайнов, чем мы и займемся в части 2.

## 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.** Осуществление преобразований, переходов и анимации с помощью CSS
- Глава 11.** 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-анимации, но все-таки именно содержательный текстовый материал заставляет их возвращаться на определенные сайты вновь и вновь. Посетители 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). Данная 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. Использование слишком большого количества шрифтов или типографских украшательств зачастую приводит читателя в замешательство, затрудняя понимание сути информации, изложенной на веб-странице (рис. 6.1, *a*). Применение различных типоразмеров, умелый подбор стиля и взаимодействие всего пары разных шрифтов облегчает просмотр страницы и делает ее приятной для чтения (рис. 6.1, *б*).

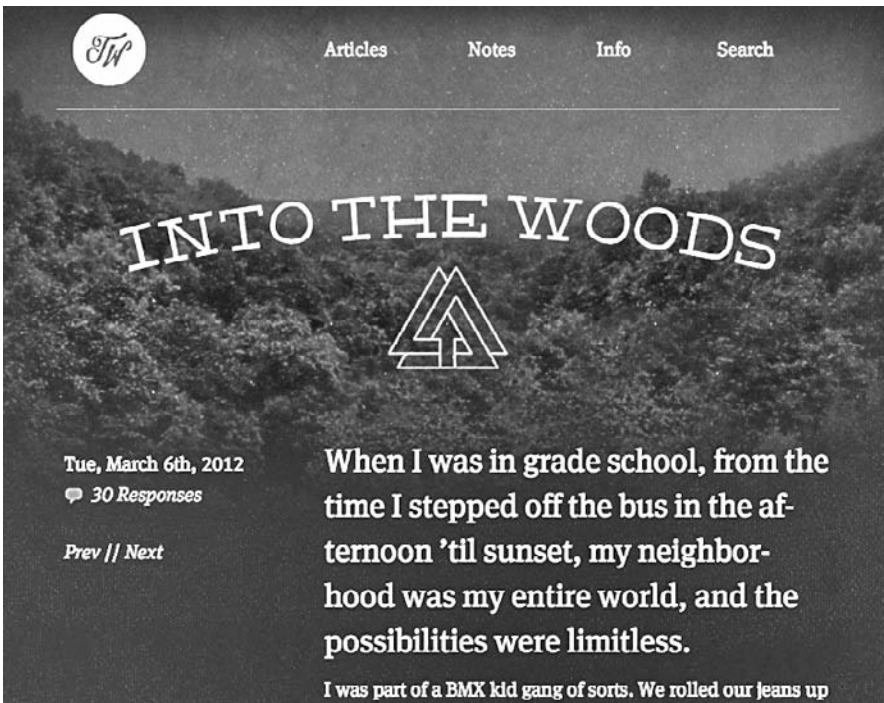
## Использование шрифтов

Первое, что вы можете сделать для достижения большей привлекательности сайта, — применить различные шрифты к заголовкам, абзацам и другим элементам. Для выбора начертания в CSS-стиле применяется свойство `font-family` и указывается тот шрифт, который вы хотите использовать. Предположим, например, что вы хотите применить для абзацев страницы шрифт Arial. Для этого можно создать стиль `p` и воспользоваться свойством `font-family`:

```
p {  
font-family: Arial;  
}
```



a



б

Рис. 6.1. Оформление сайтов очень важно для их восприятия



Изначально свойство `font-family` работает, только если у посетителей сайтов имеется такой же шрифт, установленный на их компьютерах. Иначе говоря, применительно к показанному выше примеру, если у кого-то из посетителей вашего сайта не будет на его компьютере шрифта Arial, абзацы страницы будут отображены с использованием исходного шрифта веб-браузера (обычно это какой-нибудь вариант Times New Roman). Поэтому веб-дизайнеры ограничивались небольшим набором шрифтов, поступающих предустановленными на большинстве компьютеров.

В последнее время веб-браузеры стали поддерживать веб-шрифты, то есть такие шрифты, которые браузер загружает и использует при просмотре сайта. Веб-шрифты также применяют свойство `font-family`, но требуют дополнительной CSS-команды, которая называется директивой `@font-face` и которая выдает веб-браузеру инструкцию загрузить указанный шрифт. Веб-шрифты открывают многие захватывающие возможности в области дизайна, позволяя делать выбор из огромного количества гарнитур.

Но вскоре вы увидите, что данный подход также имеет проблемы. Как веб-дизайнер вы можете остановиться на выборе проверенных и реально существующих шрифтов, то есть указать шрифты из основного набора, установленного на большинстве компьютеров, или же задать веб-шрифты, расширив свой дизайнерский кругозор (ценой дополнительной работы). Но вы не ограничены только одним из этих подходов. Многие дизайнеры используют их в сочетании, в одних случаях применяя стандартные шрифты (например, для основной части текста страницы), а в других — веб-шрифты (допустим, для создания привлекательных заголовков).

## Выбор обычного шрифта

Когда для указания шрифта применяется свойство `font-family`, посетители не обязательно должны увидеть выбранный вами шрифт, у них он либо должен быть уже установлен на компьютере, либо в случае указания веб-шрифтов временно загружен для просмотра сайта. Поскольку вы не можете знать, доступен ли предпочитаемый вами шрифт конкретному пользователю, сложилась практика указывать не только основной шрифт, но и пару резервных вариантов. Если на компьютере посетителя сайта есть шрифт, выбранный первым, он и будет виден. Но если первый шрифт не установлен, браузер просматривает список, пока не найдет указанный в нем шрифт. 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.



## 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, Helvetica, sans-serif; }`. Если есть такие примеры определений, помните, что это всего лишь отдельное свойство CSS-стиля для сокращения и упрощения написания книги и удобства чтения.

Если название шрифта состоит из нескольких слов, вам следует заключать его в кавычки:

```
font-family: "Times New Roman", Times, serif;
```

Далее представлены некоторые часто используемые сочетания обычно установленных шрифтов, сгруппированные по типу шрифта.

**Шрифты с засечками (Serif).** Шрифты с засечками (маленькими росчерками на концах основных штрихов) идеальны для больших частей текста. Распространено мнение, что засечки визуальнo связывают одну букву с другой, делая текст более читабельным. Примерами шрифтов с засечками являются Times, Times New Roman, Georgia.

Примеры данных шрифтов приведены на рис. 6.2.

"Times New Roman", Times, serif

Class aptent taciti sociosqu ad litora torquent per conub  
tempor, leo vehicula auctor gravida, sapien lacus cursus  
aliquet, magna et sodales tincidunt, metus sem porttitor  
Aenean consectetur rutrum nibh quis congue. Cras port

Georgia, "Times New Roman", Times, serif

Class aptent taciti sociosqu ad litora torquent per c  
Morbi tempor, leo vehicula auctor gravida, sapien l  
non ante. Proin aliquet, magna et sodales tincidunt,  
condimentum orci est sit amet odio. Aenean consec  
porttitor fermentum interdum.

"Hoefler Text", Garamond, Times, serif

Class aptent taciti sociosqu ad litora torquent per cor  
tempor, leo vehicula auctor gravida, sapien lacus cursi  
aliquet, magna et sodales tincidunt, metus sem porttti  
odio. Aenean consectetur rutrum nibh quis congue. C

"Palatino Linotype", Baskerville, Times, serif

Class aptent taciti sociosqu ad litora torquent per conub  
tempor, leo vehicula auctor gravida, sapien lacus cursus  
aliquet, magna et sodales tincidunt, metus sem porttitor  
Aenean consectetur rutrum nibh quis congue. Cras port

"Times New Roman", Times, serif

Class aptent taciti sociosqu ad litora torquent per conul  
gravida, sapien lacus cursus tellus, ac eleifend felis eros  
nulla, non condimentum orci est sit amet odio. Aenean

Georgia, "Times New Roman", Times, serif

Class aptent taciti sociosqu ad litora torquent per  
vehicula auctor gravida, sapien lacus cursus tellus  
tincidunt, metus sem porttitor nulla, non condime  
congue. Cras porttitor fermentum interdum.

"Hoefler Text", Garamond, Times, serif

Class aptent taciti sociosqu ad litora torquent per conul  
gravida, sapien lacus cursus tellus, ac eleifend felis eros  
nulla, non condimentum orci est sit amet odio. Aenean

"Palatino Linotype", Baskerville, Times, serif

Class aptent taciti sociosqu ad litora torquent per c  
auctor gravida, sapien lacus cursus tellus, ac eleifei  
metus sem porttitor nulla, non condimentum orci e  
porttitor fermentum interdum.

**Рис. 6.2.** Шрифты не всегда отображаются одинаково в Windows (слева) и Macintosh (справа).  
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 и Verdana.

Примеры данных шрифтов приведены на рис. 6.3.

**Arial, Helvetica, sans-serif**

Class aptent taciti sociosqu ad litora torquent per conubia nostra, lacus cursus tellus, ac eleifend felis eros non ante. Proin aliquet, magni orci est sit amet odio. Aenean consectetur rutrum nibh quis congue.

**Verdana, Arial, Helvetica, sans-serif**

Class aptent taciti sociosqu ad litora torquent per conubia auctor gravida, sapien lacus cursus tellus, ac eleifend felis sem porttitor nulla, non condimentum orci est sit amet oc fermentum interdum.

**Geneva, Arial, Helvetica, sans-serif**

Class aptent taciti sociosqu ad litora torquent per conubia r gravida, sapien lacus cursus tellus, ac eleifend felis eros non porttitor nulla, non condimentum orci est sit amet odio. Aet fermentum interdum.

**Tahoma, "Lucida Grande", Arial, sans-serif**

Class aptent taciti sociosqu ad litora torquent per conubia nostra, lacus cursus tellus, ac eleifend felis eros non ante. Proin aliquet, n orci est sit amet odio. Aenean consectetur rutrum nibh quis congue.

**"Trebuchet MS", Arial, Helvetica, sans-serif**

Class aptent taciti sociosqu ad litora torquent per conubia nostra sapien lacus cursus tellus, ac eleifend felis eros non ante. Proin i condimentum orci est sit amet odio. Aenean consectetur rutrum

**"Century Gothic", "Gill Sans", Arial, sans-serif****Arial, Helvetica, sans-serif**

Class aptent taciti sociosqu ad litora torquent per conubia tellus, ac eleifend felis eros non ante. Proin aliquet, magni consectetur rutrum nibh quis congue. Cras porttitor ferme

**Verdana, Arial, Helvetica, sans-serif**

Class aptent taciti sociosqu ad litora torquent per c sapien lacus cursus tellus, ac eleifend felis eros non condimentum orci est sit amet odio. Cras porttitor ferme

**Geneva, Arial, Helvetica, sans-serif**

Class aptent taciti sociosqu ad litora torquent per conubia tellus, ac eleifend felis eros non ante. Proin aliquet, magni consectetur rutrum nibh quis congue. Cras porttitor ferme

**Tahoma, "Lucida Grande", Arial, sans-serif**

Class aptent taciti sociosqu ad litora torquent per conubia tellus, ac eleifend felis eros non ante. Proin aliquet, magni consectetur rutrum nibh quis congue. Cras porttitor ferme

**"Trebuchet MS", Arial, Helvetica, sans-serif**

Class aptent taciti sociosqu ad litora torquent per conubi tellus, ac eleifend felis eros non ante. Proin aliquet, magi Aenean consectetur rutrum nibh quis congue. Cras portt

**"Century Gothic", "Gill Sans", Arial, sans-serif**

Class aptent taciti sociosqu ad litora torquent per conubia

**Рис. 6.3.** Шрифты без засечек в Windows (справа) и Macintosh (слева)

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.

**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 (как буквы в механических печатных машинках).

С примерами этих шрифтов можете ознакомиться на рис. 6.4. 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.

**"Courier New", Courier, monospace**

Class aptent taciti sociosqu ad litora torquent per i inceptos himenaeos. Morbi tempor, leo vehicula auctor: cursus tellus, ac eleifend felis eros non ante. Proin tincidunt, metus sem porttitor nulla, non condimentum Aenean consectetur rutrum nibh quis congue. Cras por

**"Lucida Console", Monaco, monospace**

Class aptent taciti sociosqu ad litora torquent per i inceptos himenaeos. Morbi tempor, leo vehicula auctor: cursus tellus, ac eleifend felis eros non ante. Proin tincidunt, metus sem porttitor nulla, non condimentum Aenean consectetur rutrum nibh quis congue. Cras por

**"COPPERPLATE LIGHT", "COPPERPLATE GOTHIC LIGHT", SERIF**

CLASS APTEENT TACITI SOCIOSQU AD LITORA TORQUENT PER CON HIMENAEOS. MORBI TEMPOR, LEO VEHICULA AUCTOR GRAVIDA, S ELEIFEND FELIS EROS NON ANTE. PROIN ALIQUET, MAGNA ET SOC PORTTITOR NULLA, NON CONDIMENTUM ORCI EST SIT AMET ODIO. NIBH QUIS CONGUE. CRAS PORTTITOR FERMENTUM INTERDUM.

**"Marker Felt", "Comic Sans MS", fantasy**

Class aptent taciti sociosqu ad litora torquent per conubia n auctor gravida, sapien lacus cursus tellus, ac eleifend felis eros non ante. Proin aliquet, m

**"Courier New", Courier, monospace**

Class aptent taciti sociosqu ad litora torquent Morbi tempor, leo vehicula auctor gravida, sapi non ante. Proin aliquet, magna et sodales tinci condimentum orci est sit amet odio. Aenean cons porttitor fermentum interdum.

**"Lucida Console", Monaco, monospace**

Class aptent taciti sociosqu ad litora torquent Morbi tempor, leo vehicula auctor gravida, sapi non ante. Proin aliquet, magna et sodales tinci condimentum orci est sit amet odio. Aenean cons porttitor fermentum interdum.

**"Copperplate Light", "Copperplate Gothic Light", serif**

Class aptent taciti sociosqu ad litora torquent per conubia nos auctor gravida, sapien lacus cursus tellus, ac eleifend felis erc sem porttitor nulla, non condimentum orci est sit amet odio. . fermentum interdum.

**"Marker Felt", "Comic Sans MS", fantasy**

Class aptent taciti sociosqu ad litora torquent per conubia n auctor gravida, sapien lacus cursus tellus, ac eleifend felis eros metus sem porttitor nulla, non condimentum orci est sit ame

**Рис. 6.4.** Моноширинные шрифты в Windows (справа) и Macintosh (слева)

**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 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.

## Использование веб-шрифтов

Традиционный способ использования шрифтов в CSS прост и понятен: нужно указать желаемый шрифт, воспользовавшись свойством `font-family`. Но вы ограничены теми шрифтами, которые, скорее всего, установлены на компьютерах посетителей вашего сайта. К счастью, как уже ранее упоминалось, все основные браузеры поддерживают использование веб-шрифтов. При этом браузеры фактически загружают шрифт с веб-сервера и применяют его для отображения текста на веб-странице.

CSS-часть веб-шрифтов довольно проста и требует всего двух CSS-команд:

- директивы `@font-face`, которая отвечает за сообщение веб-браузеру как имени шрифта, так и места, откуда его нужно загрузить. Вскоре вы узнаете, как эта CSS-команда работает, а сейчас просто имейте в виду, что с ее помощью браузеру сообщается о необходимости загрузить шрифт;
- свойства `font-family`, которое используется с веб-шрифтами точно так же, как и со шрифтами, уже установленными на компьютере. Другими словами, если есть директива `@font-face`, заставляющая браузер загрузить шрифт, вы можете назначить этот шрифт любому CSS-стилю с помощью свойства `font-family`.

Теоретически веб-шрифты использовать нетрудно. Но, если вдаваться в подробности, для их правильного применения нужно быть в курсе специфических требований.

---

### ПРИМЕЧАНИЕ

Довольно простой способ использования веб-шрифтов предлагает компания Google. Вскоре мы рассмотрим, в чем именно он заключается.

---

## Типы файлов шрифтов

В это сложно поверить, но Internet Explorer имел поддержку веб-шрифтов, начиная с 5-й версии (выпущенной более 12 лет назад!). К сожалению, данная поддержка

требовала использования весьма специфического и сложного способа создания шрифтового форматирования. То есть нельзя было просто взять обычный шрифт из компьютера, выложить его на веб-сервер и считать, что дело сделано. Нужно было взять обычный шрифт и преобразовать его в формат EOT (Embedded Open Type — встраиваемый формат Open Type). И такое положение вещей сохранялось вплоть до версии Internet Explorer 8.

Но для веб-шрифтов используются и другие форматы шрифтов. Некоторые из них работают в одних, но не работают в других браузерах. Чтобы указанными шрифтами могло полюбоваться как можно большее количество посетителей вашего сайта, нужно предоставить эти шрифты в различных форматах (как это сделать, вы вскоре узнаете).

В следующем перечне приведены различные типы шрифтов и перечислены браузеры, которые с ними работают.

- **EOT.** Шрифты Embedded Open Type работают только в Internet Explorer. Чтобы преобразовать обычный шрифт в EOT-формат, потребуется специальное программное средство, но сделать это можно и на таких сайтах, как FontSquirrel.
- **True Type и Open Type.** Если заглянуть в папку Fonts вашего компьютера, то там наверняка найдутся файлы шрифтов с расширениями TTF (True Type) или OTF (Open Type). Эти форматы чаще всего используются в качестве компьютерных шрифтов. Их можно применять для обработки текста и вывода его на экран, а также для веб-страниц. Эти типы шрифтов имеют широкую поддержку в большинстве веб-браузеров: Internet Explorer 9 и выше, Firefox, Chrome, Safari, Opera, iOS Safari (версии 4.2 и выше), Android и Blackberry Browser.
- **WOFF.** Файлы Web Open Font Format представляют собой самый новый формат шрифта, разработанный специально для веб-технологий. WOFF-шрифты, по сути, являются сжатой версией шрифтов TrueType или Open Type. Это означает, что WOFF-шрифты обычно имеют меньший размер файла и загружаются быстрее других шрифтов. WOFF-формат имеет также широкую поддержку со стороны браузеров, включая Internet Explorer 9 и выше, Firefox, Chrome, Safari, Opera, Blackberry Browser и iOS Safari версии 5 и выше. Наиболее заметным исключением является Android. Иными словами, если у вас используются только WOFF-шрифты, посетители, применяющие мобильные устройства Android или работающие с Internet Explorer 8, не смогут загрузить ваши шрифты и вывести их на экран.
- **SVG.** Scalable Vector Graphic формат (масштабируемой векторной графики) сам по себе не является форматом шрифта. По сути, это способ создания векторной графики (то есть такой графики, которая может масштабироваться без потери качества). Поддержка SVG-шрифтов ограничена куда существеннее. SVG-формат не поддерживается ни Internet Explorer, ни Firefox. Другой проблемой, связанной с SVG, является то, что данный формат создает файлы, зачастую в два раза длиннее, чем файлы форматов TrueType, и в три раза длиннее файлов со шрифтами формата WOFF. Единственное реальное преимущество SVG — только этот формат шрифта способны разобрать более старые версии iOS с запущенными браузерами Safari версии 4.1 или более ранних версий.

К счастью, вам не нужно выбирать только один тип шрифта, игнорируя при этом все остальные, не поддерживающие его браузеры. Совсем скоро вы узнаете, что можно (и обычно нужно) указывать несколько форматов, позволяя браузеру только тот, с которым он работает. Кроме того, вы можете загрузить шрифт, который уже был преобразован в эти четыре формата, или даже преобразовать обычный шрифт True Type в эти несколько форматов.

---

**ПРИМЕЧАНИЕ**

Отдельный файл шрифта содержит только одну плотность и один стиль для этого шрифта. Иначе говоря, если нужно получить полужирный, курсивный или полужирный и курсивный тексты, следует загрузить отдельные файлы шрифтов для каждого варианта. Некоторые шрифты, особенно имеющие забавный внешний вид, включают только один вариант и больше подходят для заголовков или текста, где не нужны курсивные или полужирные версии. Более полная информация о плотности и стилях шрифтов будет приведена чуть позже.

---

## Правовые вопросы использования веб-шрифтов

Вторым препятствием для использования веб-шрифтов являются правовые вопросы. Эти шрифты создаются с целью получения средств к существованию и продаются как отдельными разработчиками, так и компаниями. При загрузке шрифта TrueType на ваш сервер для его применения посетителями при просмотре вашего сайта любой человек может просто скачать шрифт и пользоваться им на своем сайте или на установленных на компьютере программах обработки текста или макетирования страниц. Большинству компаний, занимающихся созданием шрифтов, не нравится пиратское использование плодов их труда, поэтому многие шрифты имеют лицензии, которые однозначно запрещают их использование во Всемирной сети.

Иными словами, даже если вы купили шрифт от Adobe, его нельзя просто так начать использовать на вашем веб-сайте. Многие компании, создающие шрифты в настоящее время, предлагают различные виды лицензий (по разным ценам), чтобы разрешить использование своих шрифтов во Всемирной сети. Это касается даже шрифтов, поставляемых вместе с компьютером. Вам разрешается использовать их с программами, установленными вами на компьютер, но вам может быть не разрешено помещать файлы тех же самых шрифтов на свой веб-сервер для использования их в качестве веб-шрифтов. Если вы не знаете, разрешено ли применение шрифта во Всемирной сети, лучше воздержаться от его использования и подобрать шрифт, который в ней может использоваться.

---

**ПРИМЕЧАНИЕ**

Чтобы обойти любые правовые вопросы, можно воспользоваться такими службами шрифтов, как Google Fonts или TypeKit, коммерческой службой веб-шрифтов от компании Adobe (о которой будет рассказано в одной из следующих врезок).

---

## Поиск веб-шрифтов

При выборе веб-шрифтов приходится сталкиваться с двумя проблемами: поиском таких шрифтов, которые вполне легально можно использовать во Всемирной сети,



и подбором шрифтов, представленных всеми форматами (EOT, WOFF, TrueType и т. д.). Хотя некоторые компании, создающие шрифты, стали предлагать веб-лицензии для приобретенных вами шрифтов, существует довольно широкий ассортимент бесплатных шрифтов, доступных для использования во Всемирной сети. Вот лишь некоторые из множества источников бесплатных веб-шрифтов.

- **The League of Movable Type** ([www.theleagueofmoveabletype.com](http://www.theleagueofmoveabletype.com)). Этот сайт, представленный группой дизайнеров, был одним из первых, который предложил сетевое использование бесплатных шрифтов ручной работы. Созданный ими шрифт League Gothic широко применяется в Сети.
- **Exljbris Font Foundry** ([www.exljbris.com](http://www.exljbris.com)). Предоставляет классические бесплатные шрифты от Жоса Буйвенга: Museo, Museo Sans и Museo Slab.
- **Fontex.org** ([www.fontex.org](http://www.fontex.org)). Предоставляет широкий выбор бесплатных шрифтов, хотя не все из них лицензированы для коммерческого использования. Если вы планируете применение одного из шрифтов на коммерческом сайте (то есть не только на своем персональном веб-сайте), обязательно прочитайте лицензионное соглашение, сопровождающее загружаемый шрифт.
- **The Open Font Library** (<http://openfontlibrary.org>). Доступно более 152 бесплатных шрифтов (на момент написания книги), и все они могут использоваться на ваших веб-сайтах (только не нужно применять сразу все 152 шрифта).
- **Font Squirrel** ([www.fontsquirrel.com](http://www.fontsquirrel.com)). Весьма примечательный сайт в мире веб-шрифтов, предлагающий более 800 шрифтов. В дополнение к шрифтам Font Squirrel предоставляет то, что на этом сайте называется @font-face Kits и представляет собой коллекцию файлов, включающую шрифты во всех основных их форматах, и CSS-файл, который содержит весь необходимый @font-face-синтаксис для их загрузки (вскоре вы узнаете, как его использовать). Кроме того, Font Squirrel предлагает интерактивное средство для конвертирования шрифта TrueType или Open Type в другие форматы шрифтов, включая EOT, SVG и WOFF. Порядок использования этого средства будет рассмотрен в следующем разделе.
- **Google Fonts** ([www.google.com/webfonts](http://www.google.com/webfonts)). Компания Google предоставляет простой и бесплатный способ включения веб-шрифтов в ваши сайты. Порядок использования этой службы будет подробно рассмотрен чуть позже.

## Создание нескольких форматов шрифтов

Большинство сайтов, предлагающих бесплатные шрифты, предоставляют шрифт в единственном формате (обычно TrueType (.ttf) или Open Type (.otf)). Но TrueType и Open Type некоторыми браузерами не поддерживаются. Кроме того, самый новый формат WOFF поддерживается большинством браузеров и его файлы меньше по размеру, чем файлы TrueType. Чтобы гарантировать предоставление шрифтов как можно большему количеству браузеров, нужно как минимум использовать шрифты форматов EOT, WOFF и TrueType. А если вы обслуживаете и устройства типа iPhone с Safari версии 4.1 или более ранних версий, вам нужны шрифты SVG.

### ПРИМЕЧАНИЕ

---

Согласно сайту веб-статистики NetMarketShare, Safari 4.1 и более ранние версии представляют менее 2 % из всех мобильных веб-браузеров. Соответствующую статью можно найти по адресу [www.netmarketshare.com/browser-market-share.aspx?qprid=2&qpcustomd=1](http://www.netmarketshare.com/browser-market-share.aspx?qprid=2&qpcustomd=1).

---

К счастью, Font Squirrel предоставляет очень полезное средство, помогающее генерировать требуемые форматы шрифтов. Генератор @font-face Generator является простым способом создания не только нужных шрифтов, но также и пробного HTML-файла и простой таблицы стилей CSS.

Чтобы воспользоваться Font Squirrel @font-face Generator, выполните следующее.

1. Найдите шрифт TrueType (TTF) или Open Type (OTF).

Воспользуйтесь одним из сайтов, перечисленных в предыдущем разделе, или найдите шрифт на другом сайте. Убедитесь, что этот шрифт лицензирован для использования в качестве веб-шрифта. Если он не лицензирован или вы не уверены в его лицензировании, пропустите этот шрифт и найдите другой.

2. Перейдите к генератору @font-face Generator на странице [www.fontsquirrel.com/fontface/generator](http://www.fontsquirrel.com/fontface/generator).

Это простая страница, на которой находится всего лишь несколько настроек (рис. 6.5).

# @FONT-FACE GENERATOR

**Usage:** Click the "Add Fonts" button, check the agreement and download your fonts. If you need more fine-grain control, choose the **Expert** option.

**@font-face Kit Generator**

- 1 **Add Fonts**
- 2 **Dynamlight Regular** OTF 370 glyphs 43 KB
- 3  **BASIC**  
Straight conversion with minimal processing.  **OPTIMAL**  
Recommended settings for performance and speed.  **EXPERT...**  
You decide how best to optimize your fonts.
- 4 **Agreement:**  **Yes, the fonts I'm uploading are legally eligible for web embedding.**  
Font Squirrel offers this service in good faith. Please honor the EULAs of your fonts.

**Download Your Kit**

**Рис. 6.5.** Пока вы не станете понимать смысл своих действий, придерживайтесь варианта настройки Font Squirrel Optimal (Оптимальный)

3. Щелкните на кнопке **Add Fonts** (Добавить шрифты) (см. рис. 6.5, 1).  
Браузер откроет диалоговое окно **Select files** (Выберите файлы).
4. Выберите один или несколько шрифтов с вашего компьютера и щелкните на кнопке **Open** (Открыть).  
Браузер выложит файл или файлы на сервер Font Squirrel.
5. Выберите вариант преобразования (см. рис. 6.5, 2).
  - **Basic** (Основной) — просто преобразует шрифт в форматы EOT, WOFF и SVG.
  - **Optimal** (Оптимальный) — будет более удачным выбором, чем предыдущий вариант, поскольку он не только преобразует шрифт, но также совершит другие действия по оптимизации для повышения производительности и скорости работы со шрифтами.
  - **Expert** (Для опытных пользователей) — позволяет провести тонкую настройку всех нюансов преобразования. Например, он позволит вам создать поднабор шрифта, то есть только выбранный набор символов. Иными словами, из шрифта можно исключить определенные неиспользуемые вами символы, например точку с запятой, восклицательный знак или букву Q.

По всей вероятности, вам потребуется доступ ко всем символам, поэтому лучше остановить выбор на варианте **Optimal** (Оптимальный).

6. Установите флажок **Yes, the fonts I'm uploading are legally eligible for web embedding** (Да, выкладываемые шрифты вполне подходят для легального встраивания в веб-контент) (см. рис. 6.5, 3).

Как уже упоминалось, шрифты являются интеллектуальной собственностью, и простое выкладывание их на веб-сервер способствует пиратству в области использования программного обеспечения. Убедитесь в том, что ваши шрифты могут применяться в Сети и относятся к тому большинству, которое перечислено в данном разделе.

7. Щелкните на кнопке **Download Your Kit** (Скачайте свой комплект) (см. рис. 6.5, 4).  
В зависимости от количества конвертируемых шрифтов и их сложности процесс загрузки может занять некоторое время. Серверу Font Squirrel нужно получить шрифт и совершить над ним свои загадочные действия для генерации каждого формата шрифта. Как только все будет готово, вы скачаете папку, содержащую файлы различных форматов шрифтов, демофайл, CSS-файл и несколько других файлов разного назначения. Наиболее важными, конечно же, являются файлы шрифтов с расширениями EOT, TTF, WOFF и SVG.

Чтобы применить полученные шрифты в деле, нужно использовать их с CSS-директивой `@font-face`.

При выборе варианта настройки **Optimal** (Оптимальный) будут сгенерированы все необходимые форматы шрифтов и выполнены другие настройки, чтобы шрифты быстрее загружались и лучше выглядели на экране. Но для полного контроля процесса выберите вариант **Expert** (Для опытных пользователей). Вы увидите около десяти настроек, которыми можно воспользоваться для изменения порядка создания шрифтов.



## Использование директивы @font-face

После загрузки нужных файлов шрифтов их можно будет использовать. Сначала скопируйте файлы в то место на компьютере, где хранятся файлы для вашего веб-сайта. Многие веб-разработчики создают для этого специальную папку в корневом каталоге сайта, которая называется `fonts`, `_fonts` или `webfonts`. В качестве альтернативы, если у вас есть папка для ваших CSS-файлов, вы можете поместить файлы шрифтов в эту папку. В принципе, неважно, куда вы положите данные файлы на вашем сайте, но это поможет навести в нем порядок.

Секретом веб-шрифтов является CSS-команда под названием «директива» `@font-face`. Она называет шрифт и сообщает браузеру, где найти файл шрифта для его загрузки. Директива `@font-face` помещается в вашу таблицу стилей как обычный стиль. Предположим, например, что вы используете шрифт `League Gothic`. У вас на сайте в папке `fonts` есть файл `True Type` по имени `League_Gothic-webfont.ttf`. Нужно заставить браузер загрузить этот шрифт путем добавления директивы `@font-face` к вашей таблице стилей:

```
@font-face {
  font-family: "League Gothic";
  src: url('fonts/League_Gothic-webfont.ttf');
}
```

Первое свойство, `font-family`, вы уже видели, но здесь у него другое назначение. При использовании внутри директивы `@font-face` свойство `font-family` присваивает шрифту имя. Затем имя этого шрифта задействуется, когда данный шрифт нужно применить к стилю. Предположим, например, что требуется использовать шрифт `League Gothic` для всех абзацев на странице. Затем можно будет воспользоваться следующим стилем:

```
p {
  font-family: "League Gothic";
}
```

---

### ПРИМЕЧАНИЕ

Для каждого шрифта, который нужно использовать, применяется одна директива `@font-face`. Если у вас используются три шрифта, понадобятся три директивы `@font-face`. Лучше всего хранить их сгруппированными и помещать в верхнюю часть вашей таблицы стилей, чтобы браузер сразу же мог начать их загрузку.

---

Второй атрибут, свойство `src`, сообщает браузеру, где искать файл шрифта на сервере. Путь из таблицы стилей к файлу шрифта на системе помещается внутри кавычек и внутри функции `url()`. Путь работает точно так же, как и все другие пути к файлам HTML, например пути к изображениям, ссылкам и файлам JavaScript. Предположим, допустим, что у вас есть таблица стилей в папке по имени `_styles` и есть файл шрифта по имени `my_font.ttf` в папке по имени `_fonts`. Обе папки находятся в корневом каталоге вашего сайта. Следовательно, путь от таблицы стилей к файлу шрифта будет `../_fonts/my_font.ttf`. И тогда для этого шрифта нужно будет написать следующую директиву `@font-face`:

```
@font-face {
```

```
font-family: "My Font";
src: url('../_fonts/my_font.ttf');
}
```

Возможно, вы заметили, что в приведенном выше примере фигурирует только один файл шрифта формата TrueType. Перед тем как углубиться в подробности, скажу, что это упрощает демонстрацию общего принципа работы @font-face. Как уже говорилось, директива @font-face позволяет указать несколько файлов с различными типами шрифтов. К сожалению, из-за недостатков, свойственных браузерам, синтаксис их всех немного усложнен. Предположим, например, что на сайте нужно использовать все разновидности форматов шрифта League Gothic. Тогда показанный выше код нужно переписать следующим образом:

```
@font-face {
  font-family: 'League Gothic';
  src: url('fonts/League_Gothic-webfont.eot');
  src: url('fonts/League_Gothic-webfont.eot?#iefix') format('embedded-opentype'),
        url('fonts/League_Gothic-webfont.woff') format('woff'),
        url('fonts/League_Gothic-webfont.ttf') format('truetype'),
        url('fonts/League_Gothic-webfont.svg') format('svg');
}
```

Это выглядит как какое-то нагромождение, и, к сожалению, из-за недостатка браузера Internet Explorer код неоправданно усложнен.

Давайте в нем разберемся.

- Строка 2 сохранила свой прежний вид. Свойство font-family предоставляет имя шрифта, это то же самое имя, которое вы будете использовать, применяя шрифт к своим CSS-стилям.
- Строка 3 предназначена для Internet Explorer 9, но только при его работе в режиме совместимости (Compatibility mode), то есть в режиме, когда Internet Explorer 9 работает наподобие Internet Explorer 8. Это странное свойство было добавлено в Internet Explorer 9, чтобы веб-сайты, приспособленные под недостатки Internet Explorer 8 и более ранних версий, неплохо выглядели в Internet Explorer 9. Пользователь должен преднамеренно переключиться в режим совместимости в Internet Explorer 9, поэтому лучше, наверное, оставить эту настройку в покое.
- Строка 4 начинается со второго свойства src, которое в соответствии с правилами директивы @font-face может иметь указания на несколько шрифтов. Первым опять указан шрифт EOT, но на этот раз вы видите, что к окончанию расширения файла .eot нужно добавить строку ?#iefix. Это сделано, чтобы приспособиться к дополнительным недостаткам Internet Explorer, на этот раз дело касается Internet Explorer 6–8. Если после .eot не будет этой небольшой добавки к URL, то шрифт может неправильно отобразиться в Internet Explorer 8 или в более ранних версиях.

После URL можно также заметить новый фрагмент кода:

```
format('embedded-opentype')
```

Он показывает формат шрифта и добавляется после каждого URL для различных форматов шрифтов.

- Строки 5–7 просто определяют дополнительные форматы шрифтов. Указано всего одно свойство — `src`, разбитое на несколько строк, чтобы его было легче читать. Для каждого типа шрифта, заданного в свойстве `src`, добавляется URL, формат и запятая (для всех, за исключением последнего шрифта):

```
url('fonts/League_Gothic-webfont.woff') format('woff'),
```

---

**ПРИМЕЧАНИЕ**

В конце списка файлов для свойства `src` добавляется точка с запятой, чтобы обозначить конец свойства `src` (строка 7 в показанном выше примере). Об этой завершающей точке с запятой забывать нельзя, иначе директива `@font-face` не будет работать.

---

Даже если браузер способен понимать различные типы шрифтов (например, Chrome может использовать шрифты WOFF, TrueType и SVG), он не станет загружать все файлы шрифтов. Вместо этого по мере чтения списка типов шрифтов браузер загружает только первый шрифт, который ему понятен. Иначе говоря, если браузеру Chrome попадетс я показанный выше код, он пропустит файл с расширением EOT, поскольку этот формат шрифта ему непонятен, а загрузит файл с расширением WOFF. Затем он целиком пропустит файлы с форматами TrueType и SVG. Из этого следует, что порядок, в котором перечислены шрифты, играет важную роль. Формат WOFF в большинстве случаев более предпочтителен, поскольку его файлы меньше по размеру и загружаются быстрее, а формат SVG намного больше по размеру, поэтому нужно убедиться, что шрифты перечислены в следующем порядке: EOT, WOFF, TTF и SVG.

---

**ПРИМЕЧАНИЕ**

Показанный выше синтаксис `@font-face` стал результатом многочисленных тестирований и изысканий многих талантливых веб-профессионалов. Об эволюциях этого синтаксиса можно прочитать в следующих публикациях в блогах: <http://www.fontspring.com/blog/further-hardening-of-the-bulletproof-syntax> и <http://paulirish.com/2009/bulletproof-font-face-implementation-syntax/>.

---

## Создание стилей с использованием веб-шрифтов

Самой сложной частью использования веб-шрифтов является получение файлов шрифтов в нужном формате и настройка директивы `@font-face`. После того как все это будет готово, веб-шрифты используются точно так же, как и рассмотренные ранее шрифты, предустановленные в системе. Иначе говоря, при создании нового стиля просто применяется свойство `font-family` и предоставляется имя для шрифта, которое задействовалось в директиве `@font-face`. Например, в предыдущем коде в директиве `@font-face` новый шрифт был назван League Gothic (строка 2). Это имя и нужно использовать, применяя этот шрифт к стилю. Чтобы в содержимом всех тегов `h1` применялся шрифт League Gothic, можно создать следующий стиль:

```
h1 {  
  font-family: 'League Gothic';  
  font-weight: normal;  
}
```

Обратите внимание на употребленное здесь новое свойство `font-weight`. Обычно браузеры отображают содержимое тегов `<h1>` полужирным шрифтом. Большинство

браузеров искусственно сделают шрифт полужирным, когда требуется такая версия шрифта. В результате получается уродливое полужирное начертание. Установка для свойства `font-weight` значения `normal` заставляет браузер просто использовать шрифт League Gothic таким, какой он есть, и избежать попыток сделать его полужирным. Более подробно работа с полужирными и курсивными вариантами применительно к веб-шрифтам будет рассмотрена в следующем разделе.

Было бы неплохо также включить список резервных предустановленных шрифтов на тот случай, если браузер не сможет загрузить веб-шрифт. Здесь можно воспользоваться ранее рассмотренной технологией. Например:

```
h1 {
  font-family: 'League Gothic', Arial, sans-serif;
  font-weight: normal;
}
```

---

#### СОВЕТ

На веб-странице можно также использовать шрифты, содержащие различные символы и значки. То есть вместо создания, например, графического изображения предупреждающего значка и помещения его в текст абзаца можно воспользоваться директивой `@font-face` для загрузки шрифта, содержащего значок предупреждающего знака, и применить обыкновенную букву (отображенную в этом шрифте как значок). Для использования шрифтов, содержащих различные значки, можно обратиться к нескольким ресурсам: <http://css-tricks.com/flat-icons-icon-fonts/>, <http://css-tricks.com/html-for-icon-font-usage/> и <http://css-tricks.com/examples/IconFont/>.

---

## Работа с полужирными и курсивными вариантами

Обычные шрифты, установленные на компьютере, включают варианты стиля и плотности, поэтому при применении в HTML тега `<strong>` веб-браузер использует полужирную версию такого шрифта, а при применении к тексту тега `<em>` браузер использует курсивную версию такого шрифта. Если задействовать эти два тега в сочетании, то вы увидите полужирную курсивную версию шрифта. Это совершенно другие шрифты, содержащиеся в других файлах шрифтов. При исходном способе использования шрифтов на веб-страницах (который был рассмотрен ранее) вам не нужно волноваться насчет этих других шрифтов, потому что браузер воспользуется правильной версией автоматически.

А вот в случае с веб-шрифтами для каждого варианта шрифта вам понадобится отдельный файл. Итак, для основного текста веб-страницы нужны как минимум обычная версия шрифта, полужирная версия, курсивная версия и комбинированная полужирная и курсивная версия. И об этом нужно помнить при подборе шрифта для своего сайта, поскольку у некоторых шрифтов есть только версия с одним вариантом плотности и нет курсивной версии. Такой шрифт пригодится для заголовка, но применять его для длинных текстовых абзацев, где, скорее всего, будут встречаться фрагменты в полужирном и курсивном оформлении, нет никакого смысла. Кроме того, для каждого варианта шрифта вы должны создать отдельную директиву `@font-face`.

При работе с курсивными и полужирными версиями веб-шрифтов есть два варианта на выбор. Один из них проще реализовать, но он не работает в Internet Explorer 8 или более ранней версии (или в Internet Explorer 9 в режиме совместимости), другой требует больших трудозатрат, но функционирует и на старых версиях Internet Explorer.

## Простой способ добавления полужирного и курсивного вариантов

Самый простой способ добавления полужирного и курсивного вариантов шрифтов заключается в добавлении к директиве `@font-face` свойств `font-weight` и `font-style`. Обычно CSS-свойство `font-weight` требует от браузера показать шрифт в полужирном, (`bold`), обычном (`normal`) варианте или в одном из нескольких других вариантах плотности, а свойство `font-style` требует от браузера отобразить шрифт в курсивном (`italic`) или обычном (`normal`) варианте. Но при использовании в директиве `@font-face` свойство `font-style` требует от браузера применить шрифт, когда стиль запрашивает конкретный вариант шрифта.

Предположим, что у вас есть шрифт под названием PTSans. Вы начинаете с обычной, не полужирной и не курсивной версии шрифта. Различные форматы шрифта начинаются с PTSansRegular (PTSansRegular.eot, PTSansRegular.ttf и т. д.). В таблицу стилей нужно добавить следующую директиву `@font-face`:

```
@font-face {
  font-family: 'PTSans';
  src: url('PTSansRegular.eot');
  src: url('PTSansRegular.eot?#iefix') format('embedded-opentype'),
        url('PTSansRegular.woff') format('woff'),
        url('PTSansRegular.ttf') format('truetype'),
        url('PTSansRegular.svg') format('svg');
  font-weight: normal;
  font-style: normal;
}
```

Обратите внимание на следующее.

- Для семейства шрифтов во второй строке показанного выше кода используется общее имя — PTSans, вместо имени, указанного для файла шрифта, — PTSansRegular.
- Для свойства `font-weight` установлено значение `normal`, поскольку эта версия шрифта не является полужирной (строка 8).
- Для свойства `font-style` задано значение `normal`, поскольку это не курсивная версия шрифта (строка 9).

### ПРИМЕЧАНИЕ

---

В показанных здесь примерах кода предполагается, что файлы шрифтов PTSansRegular.eot, PTSans-Bold.eot и т. д. находятся в одной и той же папке, что и таблица стилей. Если бы использовались разные папки, пришлось бы перенастроить URL для точного указания места, где находятся файлы шрифтов по отношению к таблице стилей.

---

Теперь предположим, что у вас имеется курсивная версия шрифта, имя файла которой начинается с PTSansItalic. Его следовало бы добавить к таблице стилей:

```
@font-face {
  font-family: 'PTSans';
  src: url('PTSansItalic.eot');
  src: url('PTSansItalic.eot?#iefix') format('embedded-opentype'),
        url('PTSansItalic.woff') format('woff'),
        url('PTSansItalic.ttf') format('truetype'),
```

```

    url('PTSansItalic.svg') format('svg');
font-weight: normal;
font-style: italic;
}

```

В строке 2 используется то же самое имя `font-family` — `PTSans`. Но значение свойства `font-style` изменилось на `italic` (строка 9). Тем самым браузеру сообщается, что указанный шрифт является курсивной версией шрифта `PTSans`. Нужно также добавить подобные директивы `@font-face` и для полужирной, а также комбинированной полужирной и курсивной версий:

```

@font-face {
  font-family: 'PTSans';
  src: url('PTSansBold.eot');
  src: url('PTSansBold.eot?#iefix') format('embedded-opentype'),
        url('PTSansBold.woff') format('woff'),
        url('PTSansBold.ttf') format('truetype'),
        url('PTSansBold.svg') format('svg');
  font-weight: bold;
  font-style: normal;
}

@font-face {
  font-family: 'PTSans';
  src: url('PTSansBoldItalic.eot');
  src: url('PTSansBoldItalic.eot?#iefix') format('embedded-opentype'),
        url('PTSansBoldItalic.woff') format('woff'),
        url('PTSansBoldItalic.ttf') format('truetype'),
        url('PTSansBoldItalic.svg') format('svg');
  font-weight: bold;
  font-style: italic;
}

```

Иначе говоря, для охвата всех вариантов полужирного, курсивного и обычного текста вам нужны четыре директивы `@font-face`: обратите внимание на то, что для всех случаев используется одно и то же имя `font-family`, а изменяются только свойства `src` (чтобы указывать на разные файлы) и свойства `font-weight` и `font-style`.

Преимущество этого метода состоит в том, что к тексту можно назначить обычный шрифт, применить к коду HTML теги `<em>` и `<strong>` и позволить браузеру самому выбирать файл шрифта для загрузки и использования. В данном примере, если нужно назначить шрифт `PTSans` всем абзацам, достаточно просто добавить этот стиль к своей таблице стилей:

```

p {
  font-family: PTSans;
}

```

Затем можно будет применять HTML-разметку в тегах абзацев. Например, у вас может быть использован следующий абзац:

```

<p>When I was younger, I could remember <em>anything</em>, whether it had happened
or <strong>not</strong> -- <strong><em>Mark Twain</em></strong></p>

```

Когда веб-браузер прочитает таблицу стилей (с четырьмя директивами @font-face и стилем тега p), он станет отображать большинство абзацев с использованием шрифта PTSansRegular. Но для слова anything, заключенного в теги <em> будет применяться шрифт PTSansItalic, для слова not внутри тегов <strong> — шрифт PTSansBold, а для слов Mark Twain внутри и <strong>, и <em> — шрифт PTSansBoldItalic.

Эти директивы работают даже для заголовков. Если вы создаете стиль для форматирования всех тегов h1 с использованием шрифта PTSans, то он может иметь следующий вид:

```
h1 {  
  font-family: PTSans;  
}
```

При наличии данного стиля веб-браузер будет фактически применять полужирную версию шрифта PTSans, поскольку заголовки обычно отображаются полужирным шрифтом. (При использовании данной технологии с привлечением нескольких вариантов шрифтов уже не нужно, как раньше, добавлять фрагмент font-weight: normal:.)

К сожалению, Internet Explorer 8 и более ранние версии не понимают этого метода и будут использовать для всего текста шрифт PTSansRegular. Для тегов <em> и <strong> Internet Explorer будет создавать искусственный курсивный и искусственный полужирный варианты шрифтов, то есть он будет наклонять шрифт PTSansRegular на экране для получения курсива и делать шрифт PTSansRegular толще для получения полужирного варианта. Получающиеся в результате этого сгенерированные компьютером полужирный и курсивный варианты обычно выглядят неважно.

---

#### ПРИМЕЧАНИЕ

В учебных файлах этой главы в папке webfonts-demo присутствуют примеры использования обеих технологий включения полужирной и курсивной версий шрифтов. В файле webfonts.html используется первая из рассматриваемых технологий, а в файле webfonts-ie-safe.html — вторая.

---

## Добавление полужирного и курсивного вариантов и поддержка Internet Explorer 8

Если вы все еще поддерживаете Internet Explorer 8 (или более раннюю версию), предыдущее решение, касающееся полужирного и курсивного вариантов, не будет работать так же хорошо. Можно получить шрифты для работы в Internet Explorer 8, но это требует дополнительных усилий. Для начала нужно по-прежнему создать четыре директивы @font-face, по одной для каждого варианта шрифта. Но вместо задания для них одного и того же имени семейства font-family (например, PTSans), каждому из них дается свое уникальное имя (PTSansRegular, PTSansItalic и т. д.). Иными словами, четыре директивы @font-face нужно переписать следующим образом:

```
@font-face {  
  font-family: 'PTSansRegular';  
  src: url('PTSansRegular.eot');  
  src: url('PTSansRegular.eot?#iefix') format('embedded-opentype'),  
       url('PTSansRegular.woff') format('woff'),  
       url('PTSansRegular.ttf') format('truetype'),
```



```

        url('PTSansRegular.svg') format('svg');
    }

    @font-face {
        font-family: 'PTSansItalic';
        src: url('PTSansItalic.eot');
        src: url('PTSansItalic.eot?#iefix') format('embedded-opentype'),
            url('PTSansItalic.woff') format('woff'),
            url('PTSansItalic.ttf') format('truetype'),
            url('PTSansItalic.svg') format('svg');
    }

    @font-face {
        font-family: 'PTSansBold';
        src: url('PTSansBold.eot');
        src: url('PTSansBold.eot?#iefix') format('embedded-opentype'),
            url('PTSansBold.woff') format('woff'),
            url('PTSansBold.ttf') format('truetype'),
            url('PTSansBold.svg') format('svg');
    }

    @font-face {
        font-family: 'PTSansBoldItalic';
        src: url('PTSansBoldItalic.eot');
        src: url('PTSansBoldItalic.eot?#iefix') format('embedded-opentype'),
            url('PTSansBoldItalic.woff') format('woff'),
            url('PTSansBoldItalic.ttf') format('truetype'),
            url('PTSansBoldItalic.svg') format('svg');
    }

```

Заметьте, что у каждой директивы `@font-face` имеется собственное название семейства, которое соответствует варианту шрифта: `PTSansRegular`, `PTSansItalic`, `PTSansBold` и `PTSansBoldItalic`.

Кроме того, обратите внимание на то, что свойства `font-weight` и `font-style`, которые использовались ранее, здесь уже отсутствуют за ненадобностью.

Труднее будет обстоять дело с применением шрифта. В предыдущем примере свойство `font-family` просто применялось к стилю:

```

p {
    font-family: PTSans;
}

```

А теперь, к сожалению, вам придется применять разные имена шрифтов для различных тегов: для `p` — имя обычного шрифта, для `em` — имя курсивного шрифта, для `strong` — имя полужирного шрифта. А чтобы справиться со случаем применения полужирного курсива придется составлять селектор потомка. То есть, чтобы могли работать различные варианты шрифта `PTSans`, нужно создать четыре стиля, содержащие довольно много строк кода:

```

p {
    font-family: PTSansRegular;
    font-size: 48px;
}

```

```

    font-style: normal;
    font-weight: normal;
}

p em {
    font-family: PTSansItalic;
    font-style: normal;
    font-weight: normal;
}

p strong {
    font-family: PTSansBold;
    font-style: normal;
    font-weight: normal;
}

p strong em, p em strong {
    font-family: PTSansBoldItalic;
    font-weight: normal;
    font-style: normal;
}

```

В первую очередь обратите внимание на то, что это за четыре стиля:

- для селектора `p` используется шрифт `PTSansRegular`;
- `p em` представляет собой селектор потомка для тега `<em>`, находящегося внутри тега `<p>`; для этого стиля используется шрифт `PTSansItalic`;
- `p strong` представляет собой еще один селектор потомка, где шрифт `PTSansBold` применяется для тега `<strong>`, когда он появляется внутри абзаца;
- и наконец, мы имеем дело с групповым селектором, составленным из двух селекторов потомков. Первый селектор потомка применяется к тегу `<em>`, находящемуся внутри тега `<strong>`, который располагается внутри тега `<p>`. А второй применяется к тегу `<strong>`, находящемуся внутри тега `<em>`, который размещается внутри тега `<p>`. Нам нужны оба этих селектора, потому что можно вложить теги `<em>` внутрь тегов `<strong>` и наоборот. В результате может получиться такой код HTML:

```

<p>
  <em><strong>Hey!</strong></em>
  , I'm talking to
  <strong><em>you</em></strong>
</p>

```

Одиночный селектор `p strong em` не будет работать для `Hey!`, поскольку это слово находится в теге `<strong>`, который располагается в теге `<em>`.

#### ПРИМЕЧАНИЕ

В HTML5 теги `<b>` (для полужирного шрифта) и `<i>` (для курсива) заняли свое прежнее место. Их можно использовать только для презентационных целей, когда нужно, чтобы текст был курсивным, но без добавления к его смыслу какой-нибудь реальной выразительности. Например, курсивом часто пишут названия книг, поэтому в данном случае рекомендуется использовать тег `<i>`: `<i>CSS3: The Missing Manual</i>`.

А при использовании тега `<em>` текст будет акцентирован, что заставит программу чтения с экрана прочитать его громче, не так, как другой текст. В любом случае, если вы намерены применять теги `<b>` и `<i>`, не забудьте создать стили, использующие курсивный и полужирный варианты шрифта (волноваться об этом приходится только в том случае, если речь идет об Internet Explorer 8, для которого нужно обеспечить безопасный способ указания курсива, а первый рассмотренный ранее метод таких мер не предусматривает).

Нужно также обратить внимание на необходимость установки во всех этих стилях значения `normal` для свойств `font-weight` и `font-style`. Если этого не сделать, многие браузеры (не только Internet Explorer) будут пытаться сделать жирнее полужирный шрифт и применить курсив к курсивной версии шрифта (особенно уродливо смотрится искусственно усиленный жирный шрифт в Firefox).

Эта вторая технология поддержки полужирного и курсивного вариантов, несомненно, требует большого объема работы. Данный объем существенно возрастает, если на одном и том же сайте используется несколько шрифтов с полужирной и курсивной версиями. Какой из технологий следует воспользоваться, зависит от того, насколько важна для вас поддержка Internet Explorer 8. На момент написания данной книги Internet Explorer 8 все еще сохранял популярность, занимая от 13 ([http://gs.statcounter.com/#browser\\_versionww-monthly-201108-201208](http://gs.statcounter.com/#browser_versionww-monthly-201108-201208)) до 25 % (<http://www.netmarketshare.com/browsermarket-share.aspx?qprid=2&qpcustomd=0&qptimeframe=M>) общего количества используемых веб-браузеров.

#### ПРИМЕЧАНИЕ

Для решения проблемы поддержки полужирных и курсивных вариантов шрифтов в Internet Explorer 8 можно применить другой подход. Попробуйте воспользоваться первым методом и посмотрите, как выглядят страницы в Internet Explorer 8. Некоторые шрифты, обычно из семейства `sans-serif`, не всегда выглядят так уж плохо, когда Internet Explorer применяет к ним искусственные методы придания полужирного и курсивного начертания. Может оказаться, что разница не так уж и заметна и можно воспользоваться первым методом, не испытывая при этом особых проблем. Следует также помнить, что рынок использования Internet Explorer 8 будет сужаться по мере приобретения пользователями новых компьютеров, перехода на Chrome или другие браузеры или же обновления операционных систем.

## Google web fonts

Если инструкции по использованию веб-шрифтов, рассмотренные в предыдущем разделе, вам по каким-то причинам не подходят, есть более простой способ, хотя и с меньшим выбором шрифтов. В дополнение к таким службам, как поиск, карты, электронная почта, и множеству других предлагаемых сервисов компания Google предоставляет простую в применении службу веб-шрифтов. Вместо загрузки шрифтов, конвертирования их в нужные форматы, а затем размещения их на веб-сервере вы просто включаете одну ссылку на внешнюю таблицу стилей Google, показывающую, какой шрифт вы предпочитаете использовать. Сервер Google отправляет нужные шрифты на веб-браузер посетителя без всякой путаницы и лишних действий с вашей стороны.

Вам остается только найти требуемые шрифты на сайте Google web fonts, скопировать необходимый код (предоставляемый Google) и добавить его на свою веб-страницу, после чего создать CSS-стили, использующие данные шрифты. Сначала нужно посетить сайт Google web fonts, который находится по адресу [www.google.com/webfonts](http://www.google.com/webfonts) (рис. 6.6).



Рис. 6.6. На сайте Google web fonts перечислены шрифты, предлагаемые компанией Google

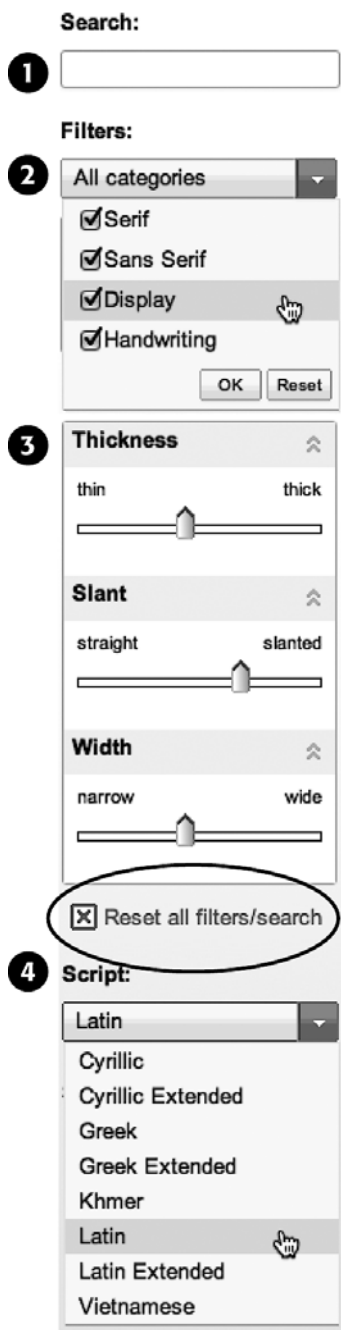
У некоторых шрифтов есть несколько стилей, например полужирный, курсив, тонкий, ультратонкий и т. д. Для просмотра всех вариантов конкретного шрифта нужно щелкнуть на ссылке Show all styles (Показать все стили) на левой боковой панели (на рис. 6.6 она обведена овалом). Для каждого шрифта имеется также по три ссылки. Первая, Quick-use (Быстрое использование) (см. рис. 6.6, 1), загружает панель использования, которую мы рассмотрим далее. Она позволяет быстро добавить код, необходимый для применения данного шрифта. Ссылка Pop out (Показать в отдельном окне) (см. рис. 6.6, 2) открывает новое окно, в котором дается более подробная информация о шрифте, а также образец, показывающий каждую букву шрифта (хороший способ составить общее представление о внешнем виде шрифта и убедиться в присутствии всех необходимых вам символов, например редко используемых или знаков пунктуации). Ссылка Add to Collection (Добавить к коллекции) (см. рис. 6.6, 3)

приводит к добавлению шрифта в вашу коллекцию. (Если нужно использовать более одного шрифта, следует все шрифты добавить в коллекцию.)

## Поиск и выбор шрифтов

Нужные шрифты выбираются путем создания коллекции. Вы просто находите понравившийся шрифт и щелкаете на кнопке **Add to Collection** (Добавить к коллекции) (см. рис. 6.6). Чтобы найти шрифт, можно спуститься вниз по главной странице веб-шрифтов и просмотреть примеры доступных шрифтов. Однако на сайте более чем 500 шрифтов и поиск нужного шрифта может занять довольно продолжительное время. Если вы уже представляете себе, как должен выглядеть искомый шрифт, например, он должен быть полужирным шрифтом без засечек, предназначенным для заголовков, воспользуйтесь вариантами фильтрации в левой части страницы (рис. 6.7).

- **Поиск по имени.** Если известно название интересующего вас шрифта, то его нужно просто набрать (или начать вводить его имя) в поле **Search** (Поиск) (см. рис. 6.7, 1).
- **Установка фильтра по категории.** Меню категорий (см. рис. 6.7, 2) позволяет показать шрифты, соответствующие одной, двум, трем или четырем категориям: **Serif** (с засечками), **Sans Serif** (без засечек), **Display** (демонстрационные) и **Handwriting** (рукописные). Чтобы убрать ненужный тип шрифта, следует просто снять соответствующий флажок, а чтобы показать нужный тип, флажок следует установить. Демонстрационные шрифты обычно бывают стильными и полужирными. Они не подходят для длинных текстовых отрывков, но могут пригодиться для коротких заголовков, выделяющихся на странице. Рукописные шрифты создают иллюзию того, что текст кем-то написан с помощью пера. Они варьируются от элегантных, похожих на стиль, используемый для свадебных приглашений, до каракулей, которые применяются в записках с требованиями выкупа типа: «Заплатите, если хотите, чтобы я вернул вашего кота!»
- **Выбор физического стиля.** Физические характеристики шрифтов можно определить с помощью трех ползунков (см. рис. 6.7, 3). Ползунок толщины (**Thickness**) позволяет подобрать шрифты, составленные из линий от очень тонких (настолько тонких, что надписи трудно читаются) до очень толстых (полужирных и плотных). Ползунок наклона (**Slant**) определяет шрифты по степени из наклона: обычно это означает использование курсивных версий шрифтов, но также относится и к рукописным шрифтам, которые обычно имеют явный наклон вправо. Применение ползунка ширины (**Width**) дает возможность подобрать шрифты зауженного либо расширенного начертания. Используя более широкие шрифты, можно поместить на одну строку всего несколько символов, но зачастую это позволяет придать заголовку более внушительный вид.
- **Выбор алфавита.** Меню гарнитуры (**Script**) (см. рис. 6.7, 4) позволяет указать шрифты для использования с другими языками. Английский язык и многие европейские языки применяют латинский алфавит (**Latin**), но если нужен, например, шрифт для русского текста, можно выбрать кириллический алфавит (**Cyrillic**). Укажите тот алфавит, который соответствует языку вашего текста.



**Рис. 6.7.** Чтобы найти шрифты, соответствующие вашему дизайну, можно провести поиск по каталогу шрифтов Google или отфильтровать список шрифтов по различным критериям

Если воспользоваться сразу всеми фильтрами, можно столкнуться с ситуацией отсутствия каких-либо результатов. В таком случае следует щелкнуть на ссылке **Reset all filters/search** (Сбросить все фильтры и критерии поиска), обведенной на рис. 6.7 овалом, чтобы вернуться к полному списку веб-шрифтов Google.

## СОВЕТ

Чтобы посмотреть демонстрацию некоторых самых лучших шрифтов от Google, зайдите на сайт <http://hellohappy.org/beautiful-web-type/>.

Как только нужные шрифты будут найдены, можно будет щелкнуть на кнопке **Add to Collection** (Добавить к коллекции) (рис. 6.8, 1). Коллекция представляет собой некое подобие корзины покупателя, в которую можно добавлять и из которой можно удалять шрифты. Чтобы увидеть шрифты, добавленные в коллекцию, щелкните на раскрывающихся стрелках (2). Шрифт можно удалить из коллекции, щелкнув на кнопке с крестиком справа от названия шрифта (3).

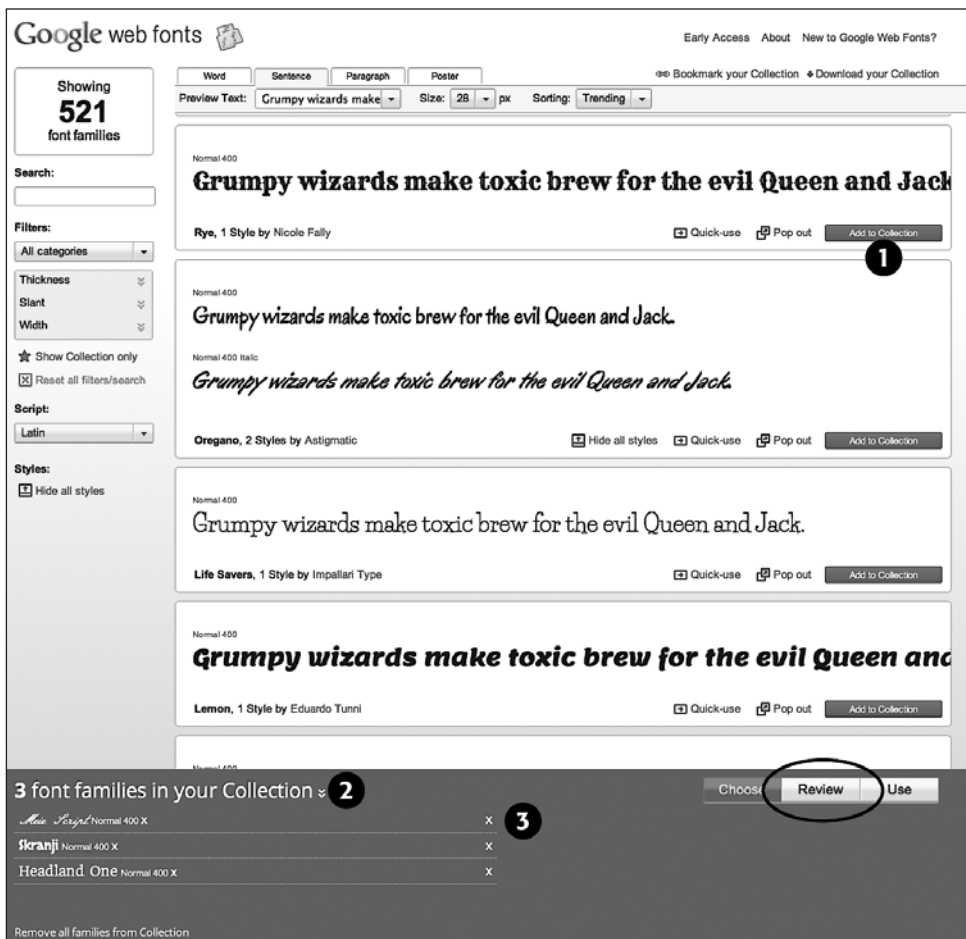


Рис. 6.8. Работа с добавленными шрифтами



Кнопка Review (Обзор), взятая на рис. 6.8 в овал, позволяет сравнивать шрифты в коллекции и извлекать более подробную информацию. Например, можно увидеть полный набор символов для каждого шрифта (то есть каждую букву и символ), протестировать шрифт, добавляя собственный текст и изменяя размер шрифта, и даже создать сравнение всех шрифтов в коллекции, наложив один на другой (если вам это действительно нужно).

## Использование шрифтов Google

После создания коллекции шрифтов настанет черед извлечения кода, необходимого для их использования.

1. Щелкните в правом нижнем углу экрана Google Web Fonts (рис. 6.9, 1) на кнопке Use (Использовать).

Откроется страница с настройками, а также с кодом, который нужно будет скопировать.

2. Выберите стиль, который нужно применить (см. рис. 6.9, 2).

Некоторые шрифты включают курсивный, полужирный и другие варианты обычного шрифта. Для основного текста страницы обычно нужны как минимум обычный, курсивный и полужирный варианты. Для заголовков можно обойтись и одним вариантом шрифта.

Вы, наверное, заметили спидометр в правой части страницы. По мере добавления новых стилей и шрифтов его указатель двигается по часовой стрелке, показывая, что загрузка шрифтов потребует больше времени. Это один из недостатков веб-шрифтов. Поскольку посетители вашего сайта вынуждены их загружать (наряду с веб-страницей, внешними таблицами стилей, графикой и другими элементами, составляющими страницу), нужно поступать разумно и не использовать слишком много шрифтов. В противном случае людям придется долго ждать, пока шрифты появятся на экране. Цифры на спидометре показывают среднее количество миллисекунд, затрачиваемых на загрузку файлов шрифтов.

3. Дополнительно ко всему остальному выберите тот набор символов, который вам нужен.

Это действие можно и не выполнять, к тому же это доступно не для всех шрифтов. Кроме того, при выборе набора символов, отличного от Latin (см. последний пункт предыдущего маркированного списка), можно увидеть и другие варианты, кроме Latin и Latin Extended. Расширенный вариант Latin Extended пригодится в том случае, если текст содержит слова на языке, использующем символы с особыми метками, например на турецком, валлийском или венгерском. Для большинства языков, имеющих письменность на латинице, например для французского и испанского, достаточно будет алфавита из набора Latin. Если расширенный алфавит Latin не нужен, то его лучше не применять, чтобы не увеличивать размер файла и время загрузки шрифта.

---

### ПРИМЕЧАНИЕ

Список дополнительных символов, доступных в Latin Extended, можно просмотреть по адресу [http://en.wikipedia.org/wiki/Latin\\_Extended-A](http://en.wikipedia.org/wiki/Latin_Extended-A).

---

4. Скопируйте код в поле Add this code to your website (Добавьте этот код к своему веб-сайту) (см. рис. 6.9, 3).

Есть три варианта.

- **Standard.** Предоставляет тег `<link>`, указывающий на внешнюю таблицу стилей (это то же самое, что и создание ссылки на любую внешнюю таблицу стилей). Но на самом деле это особая ссылка, указывающая на веб-сервер Google и предоставляющая информацию, которая необходима Google для доставки нужных шрифтов. Например:

```
<link
href='http://fonts.googleapis.com/css?family=Codystar:300,400|Gentium+Book
+Basic:400,400italic' rel='stylesheet' type='text/css'>
```

Обратите внимание, что в конце атрибута `href` перечисляются шрифты и их стили. В данном примере используются шрифты **Codystar** и **Gentium**. А Google загрузит несколько стилей: **300** и **400** для **Codystar** и **400** и **400italic** для **Gentium**. Эти числа являются способом обозначения плотности (или толщины) шрифта (см. п. 6 данного списка).

---

#### ПРИМЕЧАНИЕ

Если используется HTML5, то часть `"type='text/css'"` тега `<link>` можно опустить. В HTML5 она не нужна.

---

- **@import.** Чтобы увидеть код, необходимый для использования директивы `@import`, нужно щелкнуть на вкладке `@import` (см. рис. 6.9, 3). Преимущество данной директивы заключается в том, что ее можно добавить в начало другой таблицы стилей. Предположим, например, что у вас есть одна таблица стилей, которая предназначена для вашего сайта и на которую ссылаются все его страницы. Стандартный метод `<link>` требует добавления своего кода к каждой странице сайта. А используя директиву `@import`, можно добавить код к единственной внешней таблице стилей и закрыть вопрос.
- **JavaScript.** В этой книге данный метод не рассматривается, поскольку он требует большого объема кода, и если вы не сильны в JavaScript, то вполне можете допустить ошибку. Кроме того, этот вариант не дает весомых преимуществ по сравнению с двумя другими.

#### 5. Вставьте код в веб-страницы своего сайта.

Если используется метод `<link>`, рассмотренный в предыдущем пункте, код нужно вставить в каждую страницу, на которой вы собираетесь использовать шрифты. Если вы только приступили к процессу создания своего сайта, это сделать нетрудно, но если у вас уже много страниц, то придется потрудиться. В таком случае обратите внимание на метод `@import`: директиву `@import` можно поместить в верхней части внешней таблицы стилей вашего сайта, после чего все страницы, имеющие ссылку на эту таблицу стилей, также загрузят нужные шрифты.

---

#### ПРИМЕЧАНИЕ

Метод `@import` может оказать небольшое воздействие на производительность ваших веб-сайтов (то есть на скорость их работы).

---

#### 6. Создайте стили, использующие шрифты.

После того как шрифты загружены, их можно использовать практически так же, как и любой другой шрифт. Нужно просто создать стиль, добавить свойство

font-family и указать шрифт. Имя шрифта показано в нижней части страницы использования сайта веб-шрифтов Google (см. рис. 6.9, 4).

Если используется несколько стилей шрифта, нужно также добавить к стилю свойства font-weight и font-style. В Google обычные ключевые слова normal или bold для обозначения плотности шрифтов не используются. Вместо них применяется числовая шкала от 100 до 900. Значение 700 соответствует варианту bold, 400 — варианту normal, а остальные числа обозначают другие варианты толщины. Предположим, например, что вам нужно применить обычную курсивную версию шрифта Gentium Book Basic к тегу <em>. Для этого можно создать следующий стиль:

```
em {
font-family: "Gentium Book Basic". serif;
font-weight: 400;
font-style: italic;}
```

The screenshot shows the Google Web Fonts interface with four numbered steps:

- 1. Choose the styles you want:** Under the 'Gentium Book Basic' font family, 'Normal 400' and 'Bold 700' are selected. A 'Page Load' gauge shows 158ms. A tip states: 'Using many font styles can slow down your webpage, so only select the font styles that you actually need on your webpage.'
- 2. Choose the character sets you want:** 'Latin' is selected. A tip states: 'If you choose only the languages that you need, you'll help prevent slowness on your webpage.'
- 3. Add this code to your website:** A code block shows the link href: `<link href="http://fonts.googleapis.com/css?family=Codystar:300,400|GentiumBook+Basic:400,400italic" rel="stylesheet" type="text/css">`
- 4. Integrate the fonts into your CSS:** A code block shows the CSS rules: `font-family: 'Codystar', cursive;` and `font-family: 'Gentium Book Basic', serif;`

At the bottom right, there is a 'Use' button highlighted with a circled '1'.

**Рис. 6.9.** Когда будете готовы к применению шрифтов Google, добавленных в вашу коллекцию, щелкните на кнопке Use (Использовать) в нижнем правом углу (1), на нужном вам стиле (2) и на методе, который хотите использовать для прикрепления этих шрифтов к странице. Наиболее распространенный метод заключается в создании ссылки на таблицу стилей, которая загрузит шрифты с серверов Google

## ИНФОРМАЦИЯ ДЛЯ НОВИЧКОВ

### TypeKit как альтернатива Google

Из-за технических и правовых требований к использованию веб-шрифтов некоторые компании взяли на себя все связанные с этим трудности. И Google web fonts не единственный пример. Такие *службы шрифтов* позволяют выбирать из больших коллекций шрифтов, находящихся на собственных веб-серверах. Иными словами, вы не помещаете шрифты на свой сервер, а просто ссылаетесь на их серверы, используя либо фрагмент CSS, либо код JavaScript. Эти службы берут заботу по отправке нужных шрифтов браузерам ваших посетителей на себя (например, EOT для Internet Explorer 8 и более ранних версий).

Коммерческая служба Adobe, которая называется TypeKit, также предоставляет широкий выбор шрифтов, но за плату. Поскольку эта служба является частью

компании Adobe (которая вдобавок ко всему производимому программному обеспечению занимается также созданием шрифтов), вы получаете доступ к широкому диапазону профессионально созданных шрифтов. С помощью TypeKit создаются отдельные наборы или коллекции шрифтов, которые назначаются веб-сайту. Затем к каждой странице вашего сайта нужно добавить фрагмент кода JavaScript. Этот код устанавливает подключение к серверам TypeKit и доставляет запрошенные вами шрифты посетителям сайта. TypeKit является коммерческой службой, но она предлагает бесплатную пробную версию, имеющую некоторые ограничения. В зависимости от количества шрифтов, к которым нужно получить доступ, и ежемесячного количества посетителей вашего сайта конечная плата может составить от \$24 до 99 в год.

## 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.colorlovers.com](http://www.colorlovers.com).

## Шестнадцатеричное представление цвета

Самой старой системой задания цвета, используемой веб-дизайнерами, является шестнадцатеричная (Hexadecimal) нотация. 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 двузначных чисел содержит по два одинаковых символа. Например, можно привести #6600FF к виду #60F или #FFFFFF к #FFF.

---

**RGB**

Можно 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. Кроме того, большинство браузеров поддерживает 147 SVG-цветов (которые также называются цветами X11), поэтому, если вам действительно хочется что-то представить в более выгодном цвете, начните использовать такие цвета, как льяной, шоколадный, хаки и белый дым. Перечень этих цветов можно найти по адресу [https://developer.mozilla.org/en-US/docs/CSS/color\\_value](https://developer.mozilla.org/en-US/docs/CSS/color_value).

---

**RGBA**

Чтобы придать странице выразительность, стоит рассмотреть более современные методы задания цвета. RGBA означает Red — красный, Green — зеленый, Blue — синий, Alpha — степень смешивания с фоном. Этот способ задания цвета работает так же, как и RGB, но с добавлением альфа-канала. То есть можно задать уровень прозрачности, превратив цвет из сплошного в прозрачный (рис. 6.10). Для RGB-цветов добавляется еще одно число с диапазоном значений от 0 до 1. Значение 0 превращает цвет в невидимый, а 1 выводит цвет совершенно непрозрачным (то есть через него ничего нельзя увидеть):

```
color: rgba(255, 100, 50, .5);
```

Накладывая текст, цвет которого задан с помощью RGBA, на фоновые изображения, можно создать интересные визуальные эффекты. Например, можно создавать изображения, немного просматриваемые через цвет текста (путем использования такого высокого значения, как .9) или сильно просматриваемые через него (путем применения такого низкого значения, как .1). RGBA-цвета можно использовать с любым CSS-свойством, применяющим значение цвета, например, как на рис. 6.10 — с цветом фона поля поиска в правом верхнем углу этого изображения или с цветом кнопок навигации: Home (Домой), Archives (Архивы), Authors (Авторы) и т. д.

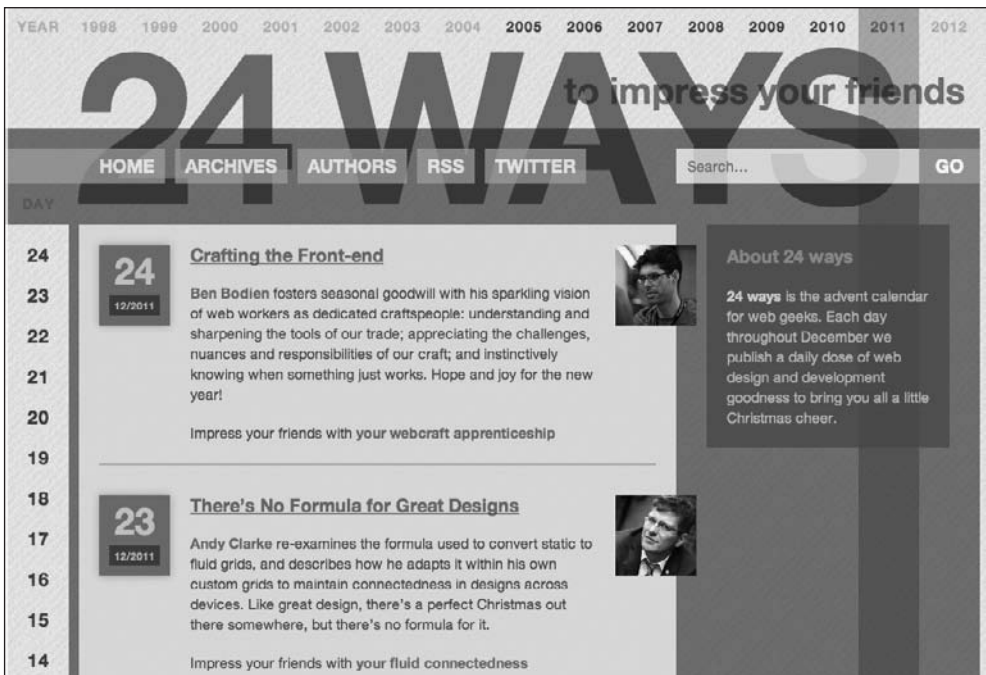


Рис. 6.10. Применением RGBA-цветов

---

**СОВЕТ**

RGBA особенно хорошо работает со свойствами `text-shadow` и `box-shadow`, которые будут рассмотрены далее. С помощью RGBA можно создавать более тонкие эффекты отбрасываемых теней, в результате чего основная часть фона становится видна сквозь тень.

А в чем же недостатки этого метода? Internet Explorer 8 и более ранние версии не понимают RGBA-цветов. Один из выходов заключается в задании сначала непрозрачного цвета с использованием шестнадцатеричной нотации, а затем второго свойства цвета с помощью RGBA-цветов:

```
color: rgb(255,100,50); /* Для IE8 */
color: rgba(255,100,50,.5); /* Для более современных браузеров */
```

Первую строку интерпретируют все браузеры, а вторая строка отменяет первую, но только для тех браузеров, которые понимают, что такое RGBA-цвета. Иными словами, Internet Explorer 8 использует первое объявление цвета и игнорирует второе, а Internet Explorer 9 и другие браузеры применяют RGBA-цвета. Вы просто не получите в Internet Explorer 8 эффекта прозрачности. Кроме того, эта технология не работает в Internet Explorer 7, который просто игнорирует обе строки и применяет черный цвет. К счастью, доля Internet Explorer 7 стремится к нулю, поэтому, наверное, о нем не стоит волноваться.

---

**ПРИМЕЧАНИЕ**

Если требуется сохранить эффект прозрачности для Internet Explorer 7 и 8, можно сделать и так. Но для этого понадобится применить немного волшебства, присущего только Internet Explorer. Научиться тому, как это делается, можно на сайте <http://css-tricks.com/rgba-browser-support>.

---

## HSL и HSLA

HSL означает Hue — тон, Saturation — насыщенность и Lightness — освещенность (иногда также используется термин Luminance — светимость). Это еще один способ задания цвета. Он не поддерживается Internet Explorer 8 или более ранними версиями, но работает во всех остальных браузерах. Если вы используете RGB- или Hex-цвета, то синтаксис HSL может показаться немного необычным. Так выглядит задание ярко-красного цвета:

```
color: hsl(0, 100%, 50%);
```

Внутри конструкции `hsl( )` задаются три значения. Первое обозначает оттенок на круге оттенков и указывается в градусах от 0 до 360. Если вы помните очередность следования цветов в радуге — красный, оранжевый, желтый, зеленый, голубой, синий и фиолетовый, — то поймете основную идею того значения, через которое выражен цвет. Красный — 0 (а также 360, поскольку это один полный оборот круга), желтый — приблизительно 50, оранжевый — приблизительно 100, зеленый — 150 и т. д. Каждый цвет занимает примерно 51°.

Вторым значением является насыщенность, то есть насколько чистым является цвет. Насыщенность можно указать в процентах от 0 до 100. Значение 0 % означает полное отсутствие насыщенности, или тусклый серый оттенок. Фактически неважно, какой тон указан, 0 % всегда даст один и тот же серый тон. Значение 100 % задает чистый цвет, яркий и живой.

Третье значение определяет освещенность, указанную в процентах от 0 (полностью черный) до 100 (полностью белый). Если нужно получить чистый цвет, следует воспользоваться значением 50 %.

Предполагалось, что метод HSL будет более интуитивно понятен, чем RGB или десятичный, но, если вы считаете, что его трудно освоить, можете его не применять. Вместо этого воспользуйтесь такими программами, как Fireworks или Photoshop, или интерактивным средством выбора цвета, которые существенно упрощают выбор RGB- или Hex-значения.

### ПРИМЕЧАНИЕ

---

Если вас заинтересовало использование HSL, то по адресу [www.workwithcolor.com/hsl-color-schemes-01.htm](http://www.workwithcolor.com/hsl-color-schemes-01.htm) можно найти простое в использовании интерактивное средство выбора цвета.

---

Точно так же, как у RGB есть сопутствующий формат RGBA, HSL поддерживает прозрачность с помощью формата HSLA. Он работает, как и ранее рассмотренный формат RGBA. Значение прозрачности указывается в диапазоне от 0 (невидимый) до 1 (полностью непрозрачный). Следующее задание свойства приведет к созданию ярко-красного цвета с 50%-ной прозрачностью:

```
color: hsla(0, 100%, 50%, .5);
```

Примеры в данной книге больше ориентированы на применение RGBA, но если HSL вам более понятен, то используйте его.

## Установка размера шрифта

Установка определенного 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 указываются 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`), пиксели, 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 — пиксели, ключевые слова, `em`, проценты — можно использовать для определения размеров шрифтов текста веб-страниц, отображаемых браузером на экране монитора. Далее рассказывается, как они работают.

## Пиксели

Переменные для значений пикселей являются самыми легкими для понимания, поскольку не зависят от параметров настроек браузера. Когда вы определяете, например, размер шрифта равным 36 пиксел для заголовка `<h1>`, браузер отображает текст высотой 36 пикселей. Дизайнеры выбирают эти единицы измерения потому, что они обеспечивают постоянные совместимые параметры размеров текста на различных типах компьютеров и браузеров (см. далее врезку, где описано ограничение для размеров, устанавливаемых в пикселях).

Чтобы задать для свойства `font-size` значение в пикселях, введите число с сокращением данной единицы измерения `px`:

```
font-size: 36px;
```

### ПРИМЕЧАНИЕ

Не добавляйте пробел между значением свойства и единицей измерения. Например, правильно `36px`, а не `36 px`.

## ИНФОРМАЦИЯ ДЛЯ НОВИЧКОВ

### Пиксели и Retina-дисплей

Когда компания Apple представила свой iPhone с Retina-дисплеем, владельцы iPhone радовались яркости и четкости изображений. Retina-дисплей компании Apple предо-

ставляет более четкое изображение за счет размещения большего количества пикселей на квадратном дюйме. В от время как в обычных компьютерных дисплеях этот



### ИНФОРМАЦИЯ ДЛЯ НОВИЧКОВ

показатель варьируется в пределах 72–100 пикселей на дюйм, новые Retina-дисплеи могут похвастаться 224 пикселями на дюйм.

С тех пор Apple добавила Retina-дисплеи на такие устройства, как iPad и ноутбуки. Другие изготовители планшетных компьютеров, например Amazon, пытаются занять эту нишу, предлагая существенно больше пикселей на дюйм, чем на старых экранах. Что это все означает для веб-дизайнеров? Существенное увеличение объема работы. Далее мы еще вернемся к этой теме. Из одной из врезок вы узнаете, что эти экраны существенно влияют на изображения, и вам придется потрудиться, чтобы изображения на дисплеях с уплотненными пикселями выглядели достойно.

Что же касается пикселей, упомянутых выше, то браузеры на устройствах, которые оснащены Retina-дисплеями, фактически умножают значение в пикселях на два. То есть если указать для текста значение 16 пикселей, веб-браузеры, подстраиваемые под Retina-дисплей, фактически будут использовать на экране для прорисовки текста 32 пикселя. Это хорошая новость. Если браузер для отображения текста применял бы только 16 из своих супермелких пикселей, то никто не смог бы прочитать то, что вы хотели сказать.

Иными словами, хотя дисплеи с повышенной плотностью уже входят в компьютерный мир мобильных и настольных устройств, они не приведут к внезапному превращению изображения, которое задано в пикселях, в нечто отображаемое в микроскопических размерах.

## 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 пикселей — это *основной (базовый) размер шрифта текста*.

Путешественники по просторам Интернета могут корректировать настройки браузеров, увеличивая или уменьшая базовый размер шрифта, но для изменения базового размера шрифта нужно изменять исходные настройки браузера, с чем многие просто не захотят связываться.

### ПРИМЕЧАНИЕ

У большинства браузеров имеется функция увеличения, укрупняющая или уменьшающая текст, графику и всю страницу. Эти настройки по мере увеличения изображения всей страницы на самом деле не изменяют базовый размер текста. Комбинация клавиш Ctrl++ (⌘++) на большинстве браузеров приводит к увеличению, а комбинация Ctrl+- (⌘--) — к уменьшению изображения. Удержание нажатой клавиши Ctrl (Control) и использование колесика мыши также позволяет увеличивать и уменьшать изображение страницы.

Когда вы изменяете текст с помощью CSS, браузер берет базовый размер шрифта текста (будь то первоначальный, высотой 16 пикселей, 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 пикселей (принимая во внимание, что вы не изменяли базовый размер шрифта в своем браузере), `x-small` соответствует 10 пикселям, `small` — 13, `large` — 18, `x-large` — 24, а `xx-large` — 32 пикселям.

Поскольку имеется ограниченный набор ключевых слов — всего семь вариантов, — если требуется большая свобода действий по масштабированию, необходимо 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 пикселей. 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 пикселей.

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. 6.11 есть текст в теге `<div>`, шрифт которого установлен размером 200 %. Это в два раза больше базового размера и составляет 32 пикселя. Все вложенные теги наследуют его и используют для вычисления своих размеров шрифтов. Другими словами, для тегов, вложенных в `<div>`, размер 100 % равен 32 пикселям. Так, текст заголовка `<h1>`, находящегося внутри тега `<div>`, составляет 100 % и отображается в два раза больше базового для этой веб-страницы, то есть высотой 32 пикселя.

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. Процентное значение эквивалентно em, умноженному на 100: .5em — 50 % и т. д.

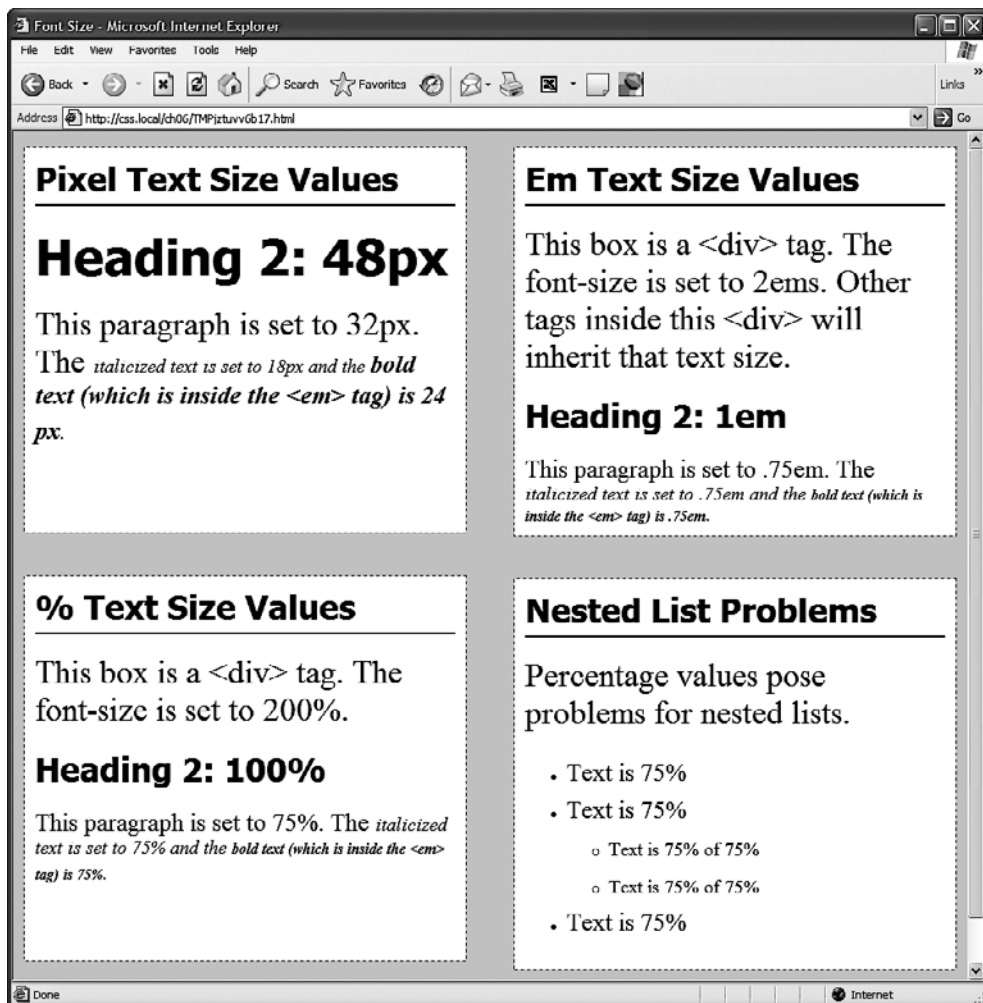


Рис. 6.11. Три самых распространенными и наиболее часто применяемыми единицами измерения для установки размеров шрифтов являются пиксели, em и проценты

Например, этот CSS-код имеет тот же эффект, что и `font-size: 200%;`:

```
font-size: 2em;
```

#### ПРИМЕЧАНИЕ

Как и при использовании пикселей, 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, а не 2.5ems.

Принцип работы механизма наследования с `em` точно такой же, как и с процентами. Взгляните, например, на верхнюю правую часть рис. 6.11. 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 пиксела.

## ОБХОДНОЙ ПРИЕМ

### Разбор вложенных конструкций

Унаследованные значения размеров шрифтов могут вызвать проблемы для вложенных (многоуровневых) списков (см. рис. 6.11). 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 решить эту проблему, создайте дополнительный стиль с селектором потомков (см. раздел «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 % от базового размера текста.

Есть еще один способ предотвращения этого эффекта комнаты кривых зеркал, сжимающего текст. В CSS3 включена новая единица измерения под названием `rem`. Это название означает Root EM, то есть его значение основано на размере текста корневого (`root`) элемента. В большинстве случаев это просто относится к базовому размеру текста, поэтому значение `.75em`, показанное на рис. 6.11, можно заменить на следующее значение:

```
font-size: .75rem;
```

Этот стиль задает размер шрифта равным `.75` от базового размера текста, а не текущего размера шрифта (как в случае применения единиц измерения `em`). Корневым элементом в HTML считается элемент `html`, который можно найти в начале веб-страницы. При использовании `rem`-значений можно установить базовый размер текста элемента `html`, а затем использовать единицу измерения `rem` для установки размера текста относительно этого базового размера. Например, можно установить базовый размер текста равным 20 пиксела:

```
html {
  font-size: 20px;
}
```

А затем воспользоваться единицей измерения `rem` для создания размера шрифтов относительно этого 20-пиксельного базового размера текста, тогда, чтобы задать для всех абзацев размер текста, равный 15 пиксела, нужно добавить следующий стиль:

```
p {
  font-size: .75rem;
}
```

Но нужно иметь в виду, что такую единицу, как `rem`, понимают на данный момент все основные браузеры, кроме браузера Internet Explorer 8 и более ранних версий.

---

**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 акцентировать важные сообщения, фрагменты текста, привлечь внимание.

---

## 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. 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 `<em>` и `<i>`, курсивом (шрифтом с наклонным начертанием), а текст, находящийся в тегах `<strong>`, `<b>`, `<th>` (table header — «заголовок таблицы»), тегах заголовков (`<h1>`, `<h2>` и т. д.) — полужирным. Вы можете управлять этими параметрами. Можно 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 имеет 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. Единственный способ применения числовых значений касается рассмотренных ранее веб-шрифтов. Фактически

числовые значения используются Google web fonts исключительно для задания плотности шрифта.

#### ПРИМЕЧАНИЕ

---

При применении веб-шрифтов задание для текста полужирного и курсивного начертания требуют выполнения других ранее рассмотренных действий.

---

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 `<strong>` (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 — используя `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, воспользуйтесь значением 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. 6.12) 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;
```

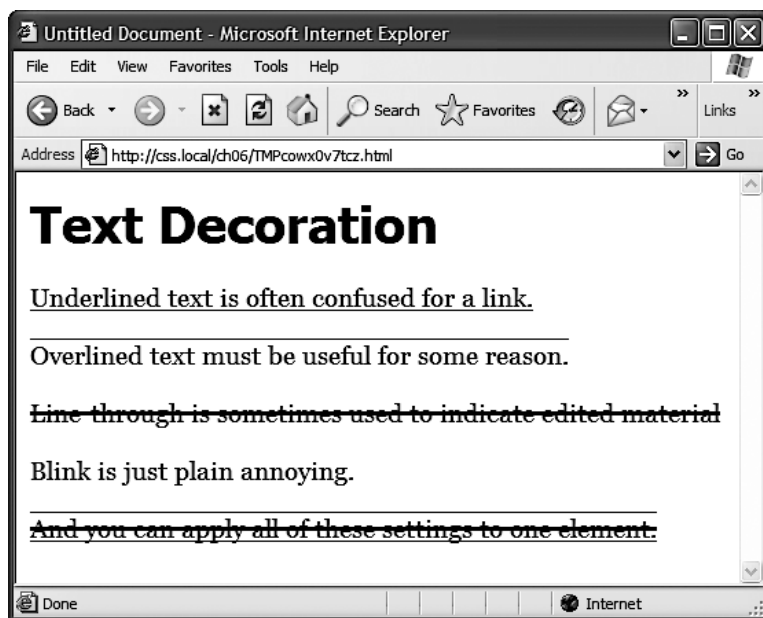


Рис. 6.12. Результат применения свойства `text-decoration`

Вы можете также объединять несколько ключевых слов 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. 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. Использование же мерцания `blink` на самом деле скорее выдаст в вас начинающего дизайнера (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. раздел «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.

Отменить все элементы украшения 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. раздел «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. 6.13). 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. Чтобы это сделать, используйте отрицательные значения свойства:

```
letter-spacing: -1px;
```

Положительные значения свойства делают промежуток между буквами больше:

```
letter-spacing: .7em;
```

Аналогично можно изменить межсловный интервал, используя свойство `word-spacing`. Оно делает промежуток между словами шире или уже, не затрагивая промежутков между буквами внутри самих слов:

```
word-spacing: 2px;
```

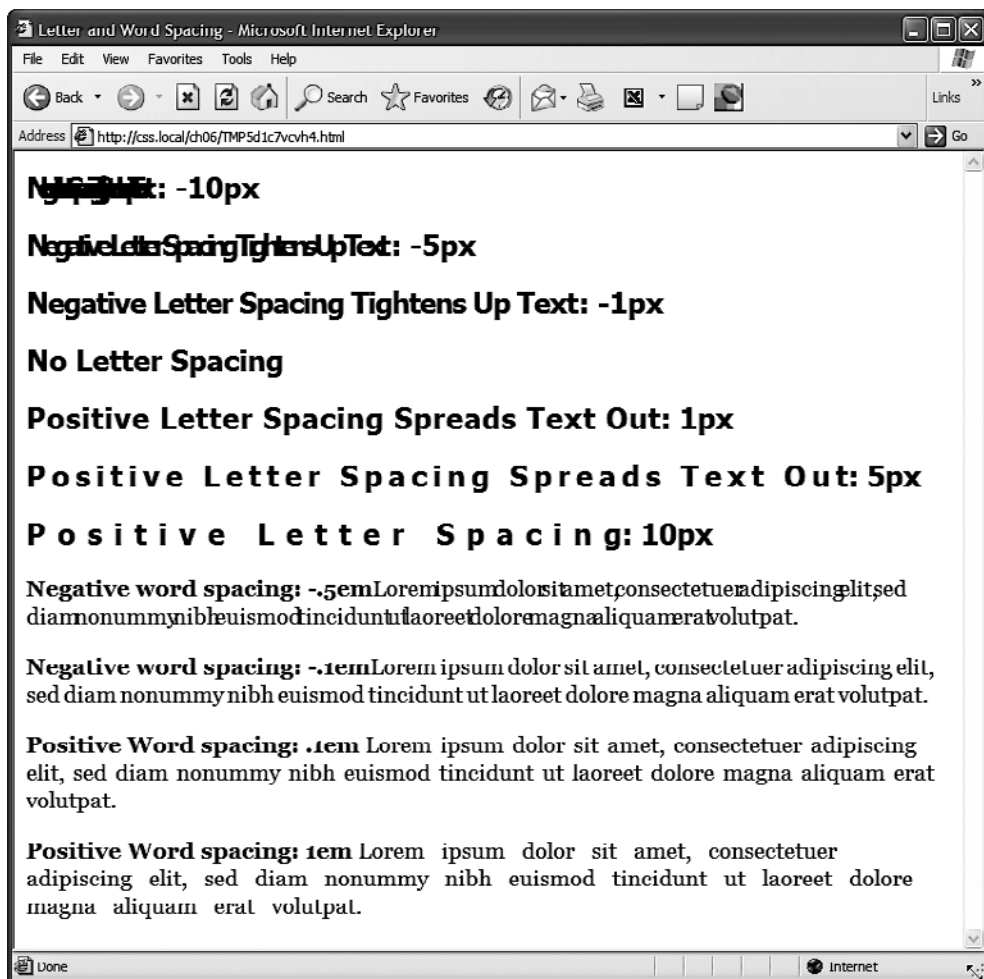
С этими свойствами можно использовать любые единицы измерений, применимые к тексту: пиксели, `em`, проценты (с положительными или отрицательными значениями).

Если вы хотите, чтобы текст хорошо читался, применяйте небольшие значения интервалов. В противном случае буквы и слова будут перекрываться, накладываться друг на друга. Чтобы сделать представление содержимого сайта четким и понятным, аккуратно используйте межсимвольные и межсловные интервалы.

## Добавление текстовых теней

В CSS3 имеется свойство, позволяющее добавлять к тексту тени для придания глубины и выразительности заголовкам, спискам и абзацам (рис. 6.14).





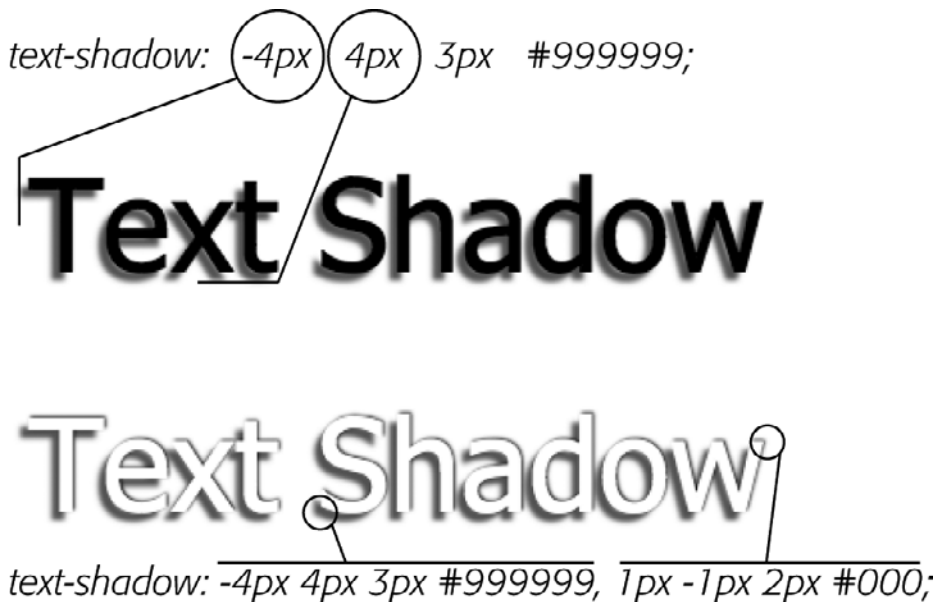
**Рис. 6.13.** Слишком большое или слишком маленькое значение межсимвольного или межсловного интервалов может сделать текст трудночитаемым или вообще нечитаемым

Свойство `text-shadow` требует задания трех параметров: горизонтального смещения (насколько левее или правее текста должна появиться тень), вертикального смещения (насколько выше или ниже текста будет тень), степень размытости тени и цвет отбрасываемой тени. Например, свойство `text-shadow`, создающее эффект, который показан в верхней части рис. 6.14, задается следующим образом:

```
text-shadow: -4px 4px 3px #999999;
```

Первое значение — `-4px` — означает «показать тень на 4 пиксела левее текста». (Положительное значение приведет к показу тени правее текста.) Второе значение — `4px` — задает отображение тени на 4 пиксела ниже текста. (Отрицательное значение приведет к показу тени над текстом.) Значение `3px` определяет степень размытости тени. Значение `0px` (без размытости) приводит к отбрасыванию четкой

тени, и чем больше будет значение, тем более размытой будет тень. И наконец, последнее значение определяет цвет отбрасываемой тени.



**Рис. 6.14.** Текстовые тени — отличный способ придания глубины заголовкам и другому тексту. Но свойство `text-shadow` не работает в Internet Explorer 9 и более ранних версиях

Для достижения более сложных эффектов можно даже создать несколько теней (см. нижнее изображение на рис. 6.14): нужно просто добавить запятую, а после нее дополнительные значения отбрасываемой тени:

```
text-shadow: -4px 4px 3px #666, 1px -1px 2px #000;
```

Количество добавляемых таким образом теней ничем не ограничивается (кроме вашего хорошего вкуса). К сожалению, этот эффект не работает в Internet Explorer 9 или более ранних версиях. Но он работает во всех других современных браузерах (даже в Internet Explorer 10). Иными словами, чтобы текст лучше читался, полагаться на этот эффект не стоит. Изображение в нижней части рис. 6.14 показывает, что не нужно делать: использовать белый цвет текста, обеспечивая его читаемость исключительно за счет отбрасываемой тени, определяющей очертания текста. В Internet Explorer белый текст на белом фоне просто не будет виден.

#### ПРИМЕЧАНИЕ

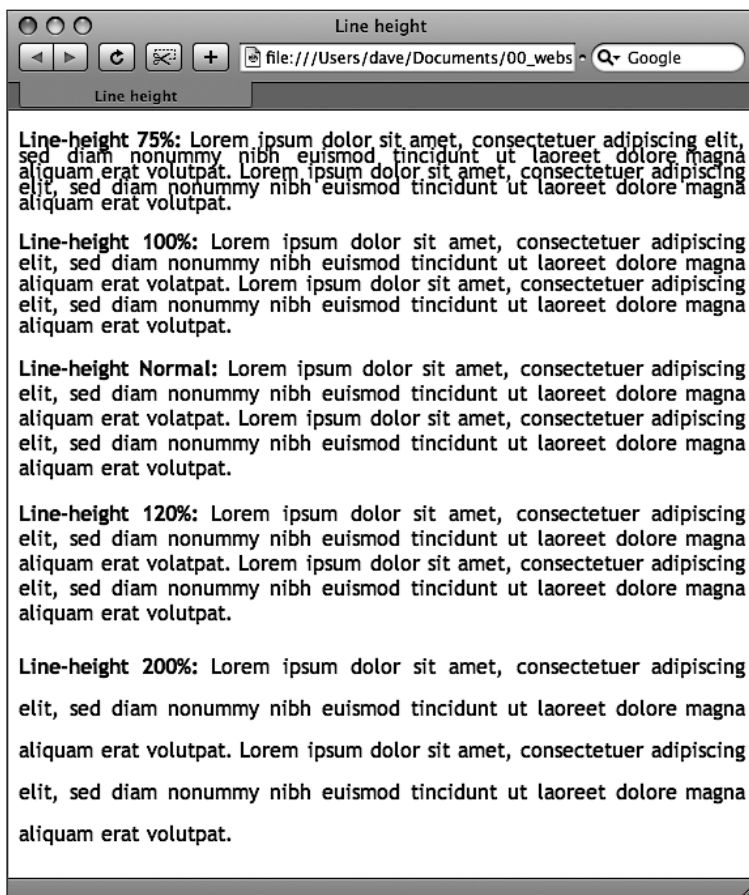
Красивые приемы использования текстовых теней можно увидеть на сайте <http://webexpedition18.com/articles/css3-text-shadow-property/>.

## Форматирование абзацев текста

В CSS есть свойства, которые используются для форматирования не отдельно взятых слов, а фрагментов, блоков текста. Иначе говоря, их можно применять к целым абзацам, заголовкам и т. д.

## Установка межстрочного интервала

В дополнение к настройке интервала между словами и буквами (междусловный и межсимвольный интервалы) CSS позволяет устанавливать межстрочный интервал (интерлиньяж) — расстояние, промежуток между базовыми линиями двух соседних строк текста, — используя свойство `line-height`. Чем выше межстрочный интервал, тем больше промежуток между отдельными строками (рис. 6.15).



**Рис. 6.15.** Свойство установки межстрочного интервала `line-height` позволяет расположить строки абзаца ближе друг к другу (*вверху*) или расположить их с меньшим промежутком, растянуть (*внизу*)

Стандартный межстрочный интервал эквивалентен 120 %. Таким образом, на рис. 6.15, *вверху*, мы видим, что меньшее процентное значение сжимает строки, а большее значение распределяет их более обособленно (см. рис. 6.15, *внизу*).

**Межстрочный интервал в пикселах, em, процентах.** Как и в свойстве размера шрифта `font-size`, вы можете использовать пиксели, `em` или проценты для установки размера межстрочного интервала с помощью свойства `line-height`:

```
line-height: 150%;
```

Вообще, процентные значения или `em` лучше, нежели пиксели: размер, установленный в этих единицах измерения, напрямую зависит от параметров шрифта и автоматически корректируется пропорционально изменению свойства `font-size`. Так, если вы зададите межстрочный интервал равным 10 пикселям и затем измените на гораздо больший (например, на 36 пикселей), 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 составляет 120 %. Таким образом, когда вы хотите уменьшить высоту строк, промежутков между ними, используйте меньшее значение. Соответственно, чтобы увеличить межстрочный интервал, то есть положить строки дальше друг от друга, примените 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 пикселей. Межстрочный интервал, установленный в размере 150 %, в итоге равняется 18 пикселям. Таким образом, браузер добавляет пустой промежуток размером 6 пикселей 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>` равным 150 %. Все теги веб-страницы унаследуют это значение. Однако наследуется не процентное значение, а *рассчитанное значение* межстрочного интервала. Допустим, основной размер шрифта задан равным 10 пикселям. 150 % от 10 пикселей составляет 15. Все теги унаследуют межстрочный интервал размером 15 пикселей. Так, если на веб-странице есть абзац текста со шрифтом высотой 36 пикселей, его межстрочный интервал размером 15 пикселей будет намного меньше самого текста, а строки сольются.

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 пикселях от стиля `body`, просто умножает размер своего шрифта на этот

коэффициент. Так, в вышеупомянутом примере, где абзац текста отображен размером шрифта 36 пикселей, межстрочный интервал составит  $1,5 \cdot 36 = 54$  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 центрировать 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, получают 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

Многочисленный повторный набор стилиевых 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`. Рассмотрим, например, следующее объявление:

```
font: italic bold small-caps 18px/150%
    Arial, Helvetica, sans-serif;
```

Оно приводит к созданию полужирного, курсивного шрифта с малыми прописными буквами размером 18 пикселей, 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

### ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

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 курсивым шрифтом с малыми пропис-

ными буквами и межстрочным интервалом 2000 % (совсем не обязательно это повторять). Затем создали стилевой класс под названием, скажем, `.special-Paragraph` 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. Обычно отрицательное значение абзацного отступа используется вместе с указанием значения поля, чтобы отрицательный абзацный отступ не выходил за левую сторону страницы, столбца или прямоугольной структуры.

Вы можете также применить 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, 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 разделе «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 пиксели 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 разделе «Начинаем с чистого листа» гл. 5.

---

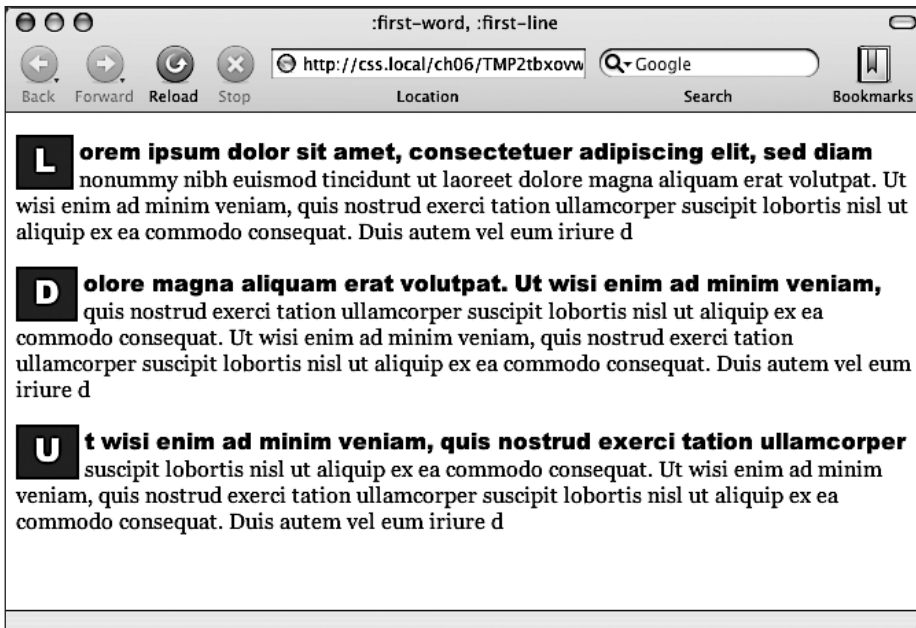
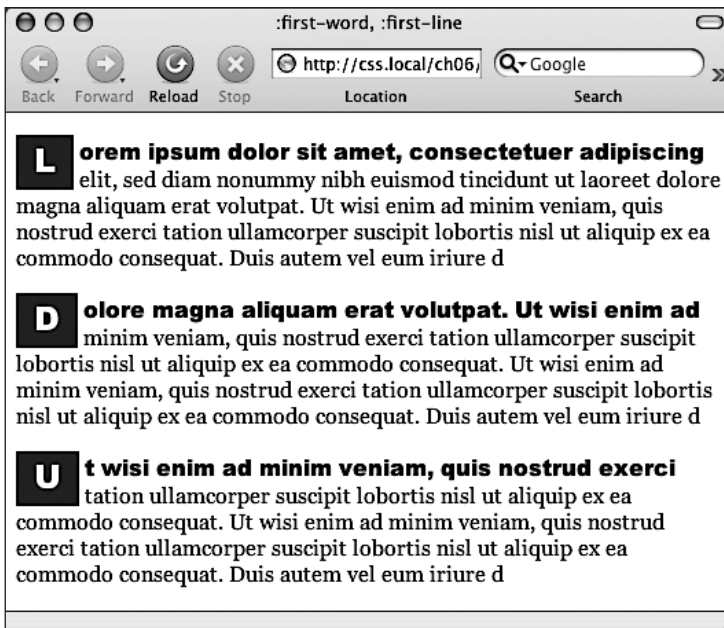
Для создания специальных эффектов можно назначить *отрицательное* значение верхнему или нижнему абзацному полю. Например, верхняя граница, установленная равной 10 пикселям, приподнимает абзац выше на 10 пикселей, возможно, даже визуальнo накладывая его на вышестоящий элемент веб-страницы.

## Форматирование первой буквы, первой строки абзаца

CSS дает возможность форматирования абзаца с использованием псевдоэлементов `:first-line` и `:first-letter` (рис. 6.16). 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;
}
```





**Рис. 6.16.** Псевдоэлемент `:first-letter` форматирует первый символ стилизуемого элемента — заглавные буквы абзацев

Для избирательной стилизации, скажем форматирования только первого символа определенного абзаца, можно применить стилевой класс, например, `.intro:`



```
<p class="intro">Text for the introductory paragraph goes here...</p>
```

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 браузеры Chrome и Safari не распознают свойство `text-transform`, когда оно используется с псевдоэлементом `:first-line`. Другими словами, вы не можете применять в Chrome и 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 predetermined 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. 6.17). Все эти варианты можно выбрать, используя CSS-свойство `list-style-type`:

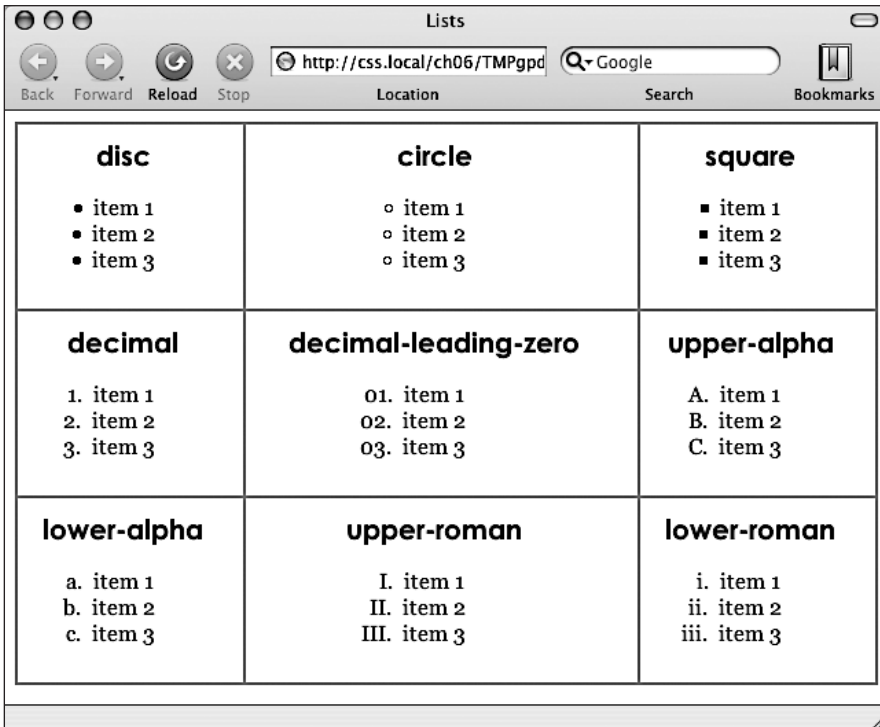
```
list-style-type: square;
```

или

```
list-style-type: upper-alpha;
```

Если нужно учесть чьи-нибудь особые предпочтения, числа можно заменить буквами греческого алфавита —  $\alpha$ ,  $\beta$ ,  $\gamma$ , воспользовавшись настройкой `lower-greek`. Существует множество других схем нумерации, включая армянский, грузинский, катакана и другие региональные варианты. Информацию о них можно найти по адресу <https://developer.mozilla.org/en-US/docs/CSS/list-style-type>.

Чаще всего это свойство используют при определении стилей, формирующих `<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.



**Рис. 6.17.** CSS предоставляет различные способы маркировки нумерованных и нумерованных списков, начиная с набора геометрических фигур и заканчивая разными системами счисления

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 стиливой класс `.circle`, который изменяет тип маркера на круглый:

```
li {list-style-type: square; }
.circle { list-style-type: circle; }
```

Теперь примените класс к элементам списка через один для чередования квадратных и круглых маркеров:

```
<ul>
  <li>Item 1</li>
  <li class="circle">Item 2</li>
  <li>Item 3</li>
  <li class="circle">Item 4</li>
</ul>
```

Или, используя рассмотренный ранее селектор `nth-of-type`, можно вообще избавиться от имени класса:

```
li {list-style-type: square; }
li:nth-of-type(odd) { list-style-type: circle; }
```

Иногда может понадобиться вообще скрыть маркеры, в том числе тогда, когда вы захотите использовать 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» раздела «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. 6.18, *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 (рис. 6.18, *справа*). Значение `inside` обеспечит списку максимальную ширину на странице:

```
list-style-position: outside;
```

или

```
list-style-position: inside;
```

### СОВЕТ

---

Вы можете изменить промежуток между маркером и текстом путем увеличения или уменьшения значения свойства `padding-left` (см. раздел «Управление размерами полей и отступов» гл. 7). Чтобы использовать это свойство, нужно создать стиль для тега элементов списка `<li>`. Данный способ работает только в том случае, если свойство `list-style-position` определено со значением `outside` (или вообще отсутствует).

---

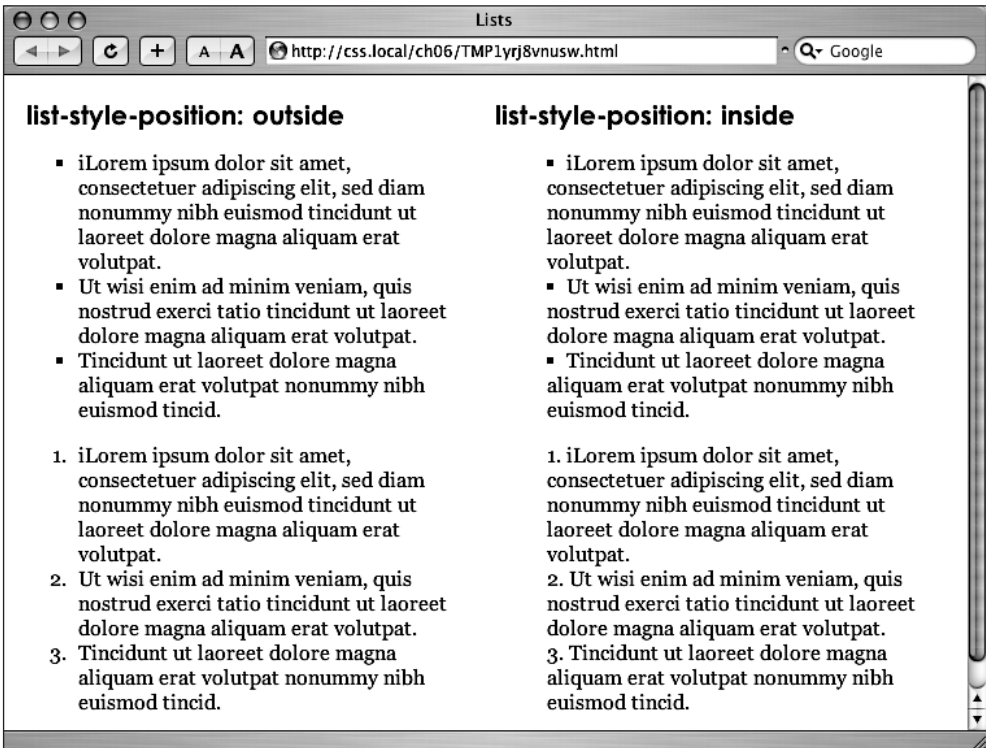
Кроме того, если вам не нравится, что браузеры делают для списков отступ, смещая все содержимое вправо, можете переопределить стиль, установив значения свойств `margin-left` и `padding-left` равными 0. Чтобы удалить отступ, можно создать такой групповой селектор:

```
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.

Вы должны только 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.



**Рис. 6.18.** Используя свойство `list-style-position`, вы можете управлять позиционированием маркеров и чисел в списках

## Графические маркеры

Если вам недостаточно стандартных маркеров квадратной и круглой формы, можете создать собственные. Используя программу редактирования графики, например Photoshop или Fireworks, можно быстро сделать красочные и интересные маркеры. В качестве образцов можете рассмотреть примеры коллекции рисунков и шрифтов символов (таких как Webdings).

### СОВЕТ

Для получения списка многочисленных сайтов с бесплатными значками и маркерами посетите страницу [www.cssjuice.com/38-freeicon-checkpoints/](http://www.cssjuice.com/38-freeicon-checkpoints/).

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, предворяющих 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 отображались 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 показываться 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 нему стилевой класс. Например: `<li><span class="regularList">Item 1</span></li>`. Сейчас маркеры отображаются полужирным красным шрифтом, а текст — обычным черным цветом. К сожалению, придется заключать в тег каждый элемент списка.

Более совершенный 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>`. Если вы хотите сделать обычные маркеры рядом с элементами списка красными, добавьте к таблице стилей следующий CSS:

```
ul li {
  list-style-type: none;
}
ul li:before {
  content: counter(item, disc) " ";
  color: red;
}
```

А если нужно сделать красными элементы нумерованного списка, можно добавить следующий CSS:

```
ol li {
  list-style-type: none;
  counter-increment: item;
}
ol li:before {
  content: counter(item) ". ";
  color: red;
}
```

Дополнительные сведения о том, как придать стиль нумерованному списку, можно найти по адресу [www.456bereastreet.com/archive/2011105/styling\\_ordered\\_list\\_numbers/](http://www.456bereastreet.com/archive/2011105/styling_ordered_list_numbers/).

**ПРИМЕЧАНИЕ**

Задавая 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 разделе «Добавление фоновых изображений» гл. 8, когда мы начнем использовать графику.

Свойство `list-style-image` позволяет применять 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 узнаете 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. Метод стенографии для набора стилей, форматирования текста» в разделе «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 `Об`.

### Настройка параметров страницы

Начнем с таблицы стилей и добавим директиву `@font-face`, чтобы загрузить некоторые веб-шрифты для форматирования основного текста страницы.

1. Запустите браузер и откройте файл `text.html` (рис. 6.19). 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>
</style>
```

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.

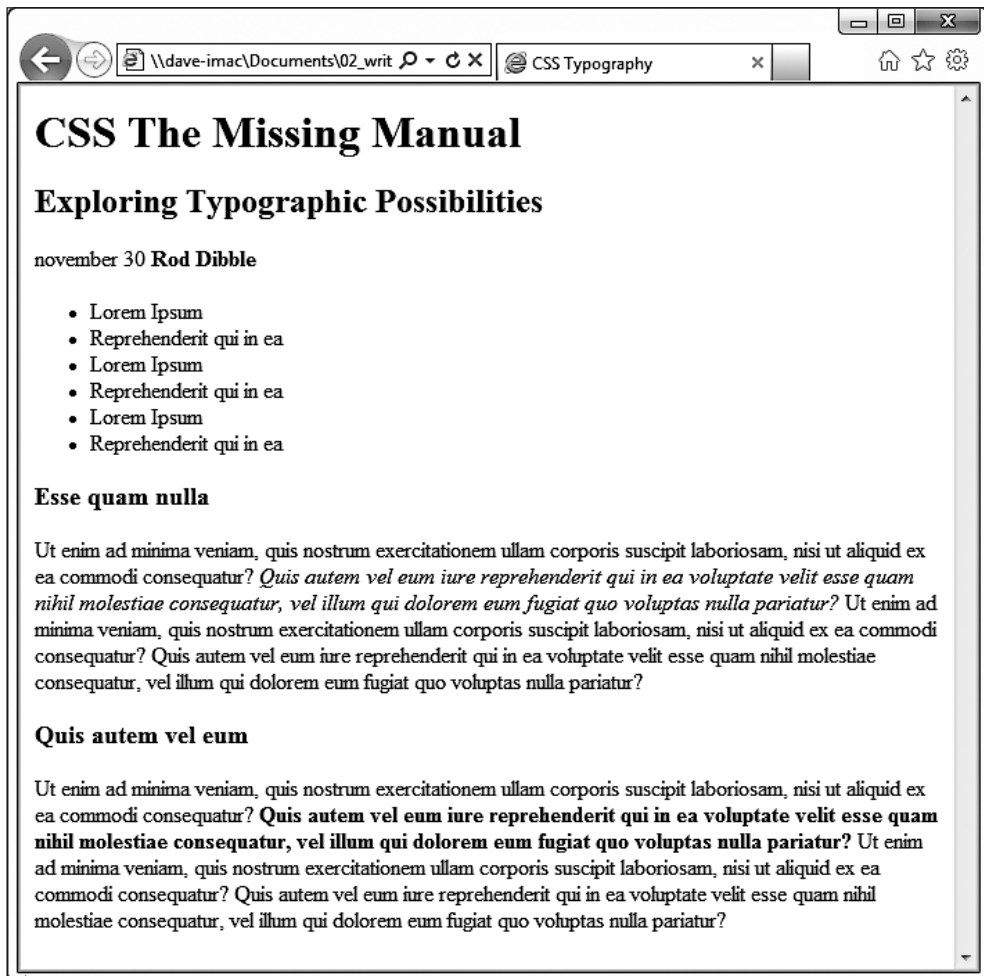


Рис. 6.19. Работа над любой веб-страницей начинается с обычного HTML-содержимого

4. Откройте файл `reset.css`. Скопируйте весь код оттуда и вставьте его между открывающим и закрывающим тегами `<style>`, добавленными на 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.

Затем нужно добавить необходимые директивы `@font-face` для загрузки четырех веб-шрифтов. Вообще-то все они являются одним и тем же шрифтом PTSans, но включают полужирную, курсивную и полужирно-курсивную версии.



5. Добавьте к странице следующий код:

```
@font-face {
  font-family: 'PTSans';
  src: url('fonts/PTSansRegular.eot');
  src: url('fonts/PTSansRegular.eot?#iefix') format('embedded-opentype'),
        url('fonts/PTSansRegular.woff') format('woff'),
        url('fonts/PTSansRegular.ttf') format('truetype'),
        url('fonts/PTSansRegular.svg') format('svg');
  font-weight: normal;
  font-style: normal;
}

@font-face {
  font-family: 'PTSans';
  src: url('fonts/PTSansItalic.eot');
  src: url('fonts/PTSansItalic.eot?#iefix') format('embedded-opentype'),
        url('fonts/PTSansItalic.woff') format('woff'),
        url('fonts/PTSansItalic.ttf') format('truetype'),
        url('fonts/PTSansItalic.svg') format('svg');
  font-weight: normal;
  font-style: italic;
}

@font-face {
  font-family: 'PTSans';
  src: url('fonts/PTSansBold.eot');
  src: url('fonts/PTSansBold.eot?#iefix') format('embedded-opentype'),
        url('fonts/PTSansBold.woff') format('woff'),
        url('fonts/PTSansBold.ttf') format('truetype'),
        url('fonts/PTSansBold.svg') format('svg');
  font-weight: bold;
  font-style: normal;
}

@font-face {
  font-family: 'PTSans';
  src: url('fonts/PTSansBoldItalic.eot');
  src: url('fonts/PTSansBoldItalic.eot?#iefix') format('embedded-opentype'),
        url('fonts/PTSansBoldItalic.woff') format('woff'),
        url('fonts/PTSansBoldItalic.ttf') format('truetype'),
        url('fonts/PTSansBoldItalic.svg') format('svg');
  font-weight: bold;
  font-style: italic;
}
```

---

#### ПРИМЕЧАНИЕ

Да, кода довольно много. Если не хочется его набирать, в примерах к этой главе есть файл `at-font-face.css`. Его можно открыть, скопировать содержащийся в нем код и вставить его в свою таблицу стилей.

Таким образом, мы создали новое семейство шрифтов по имени PTSans, которое вы сможете использовать в любых создаваемых вами новых стилях.

Для добавления полужирно-курсивных вариантов задействуется метод, рассмотренный ранее. Суть его в том, что в местах применения обычного тега полужирного шрифта (например, в заголовке или при использовании тега `<strong>`) браузер будет автоматически переключаться на полужирную версию шрифта. (Этот метод не работает в Internet Explorer 8, где полужирный и курсив создаются искусственно, вместо переключения на полужирную или курсивную версию шрифта.)

Затем вы создадите стиль, определяющий общие свойства для всего текста на странице.

- После четырех директив `@font-face` нажмите **Enter** и наберите `html {`.

Это основной селектор тега, применяемый к тегу `<html>`. В гл. 4 мы выяснили, что другие теги наследуют свойства этого тега. Можно настроить основные текстовые характеристики, такие как шрифт, цвет и размер шрифта для их использования последующими тегами в качестве их исходных настроек.

- Снова нажмите **Enter** и добавьте следующие два свойства:

```
font-family: PTSans, Arial, sans-serif;
font-size: 62.5%;
```

Эти две инструкции устанавливают шрифт PTSans (или Arial, если браузер не может загрузить PTSans) и значение для размера шрифта 62,5 %.

---

#### ПРИМЕЧАНИЕ

Зачем нужно устанавливать базовый размер шрифта веб-страницы равным 62,5 %? Оказывается, если умножить 62,5 % на 16 пикселей (стандартный размер шрифта текста большинства браузеров), получится 10 пикселей. 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  $1,5 \cdot 10$ , или 15 пикселям, 2 em — 20 пикселям и т. д. Считать очень просто, умножая размер в em на 10. Читайте об этой и других методиках, касающихся размеров шрифтов, по адресу <http://clagnut.com/blog/348/>.

---

- Завершите определение текущего стиля, нажав клавишу **Enter** и набрав закрывающую фигурную скобку для указания окончания стиля.

На данном этапе стиль должен выглядеть следующим образом:

```
html {
  font-family: PTSans, Arial, sans-serif;
  font-size: 62.5%;
}
```

Ваша таблица стилей закончена.

- 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 7. 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 после закрывающей скобки 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 классом `main`. Если вы просмотрите код HTML веб-страницы, то увидите, что там есть тег `<div>` с классом `main` (`<div class="main">`). Как вы узнаете позже, при создании дизайнов, основанных на CSS, достаточно распространено группирование HTML-тегов внутри тегов `<div>`. Вы сможете размещать отдельные теги `<div>` для создания столбцов и других составных разметок страниц. Распространено также применение 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: rgb(252,102,0);
font-family: "Arial Black", Arial, Helvetica, sans-serif;
font-size: 4em;
```

Вы только что изменили цвет заголовков `<h1>`, а также их шрифт. На этот раз вместо использования веб-шрифта указан шрифт, который может быть установлен на компьютере посетителя. Шрифт `Arial Black` есть на многих компьютерах, но если у какого-нибудь посетителя он установлен не будет, браузер воспользуется шрифтом `Arial` или `Helvetica`, или просто общим шрифтом без засечек. Кроме того, вы задали размер шрифта равным `4em`, что для большинства браузеров составляет 40 пикселей (если, конечно, посетитель не изменял параметры шрифтов в настройках своего браузера). Это все благодаря размеру `62,5%`, определенному ранее в шаге 7. Таким образом, базовый размер шрифта стал составлять 10 пикселей в высоту, а `4 · 10` задает размер 40 пикселей. Затем к заголовку будет добавлена текстовая тень.

3. Завершите текущий стиль, нажав `Enter` и добавив код, выделенный ниже полужирным шрифтом (не забудьте поставить закрывающую скобку):

```
n h1 {
color: rgb(252,102,0);
font-family: "Arial Black", Arial, Helvetica, sans-serif;
font-size: 4em;
text-shadow: 4px 3px 6px rgba(0,0,0,.5);
}
```

Здесь добавлена текстовая тень, смещенная на 4 пиксела вправо, на 3 пиксела ниже с размытостью в 6 пикселей. Кроме того, используется RGBA-цвет, устанавливающий для тени черный цвет и 50%-ную прозрачность.

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. Установите указатель после закрывающей скобки стиля `.main h1`, нажмите **Enter** и добавьте следующие стили:

```
.main h2 {  
  font: bold 3.5em "Hoefler Text", Garamond, Times, serif;  
  border-bottom: 1px solid rgb(200,200,200);  
  margin-top: 25px;  
}
```

Здесь есть еще один селектор потомка, который относится только к тегам `<h2>`, находящимся внутри другого тега с классом `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 настроить другие заголовки.

6. Добавьте новый стиль под тем, который вы создавали в предыдущем шаге:

```
.main h3 {  
  color: rgb(255,102,0);  
  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 улучшим 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 станут легче читать (но только если вы используете латиницу).

Данный стиль также увеличивает размер шрифта до 1,5 em (15 пикселей 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 совершенный тип селектора.

8. Добавьте следующий стиль к таблице стилей:

```
.main p:first-line {
  font-weight: bold;
  color: rgb(153,153,153);
}
```

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. 6.20.

## Форматирование списков

На этой странице есть один маркированный список. Переместим его вверх к правому краю страницы и сделаем так, чтобы текст, идущий за этим списком, обертывался вокруг него. С помощью CSS сделать это достаточно легко.

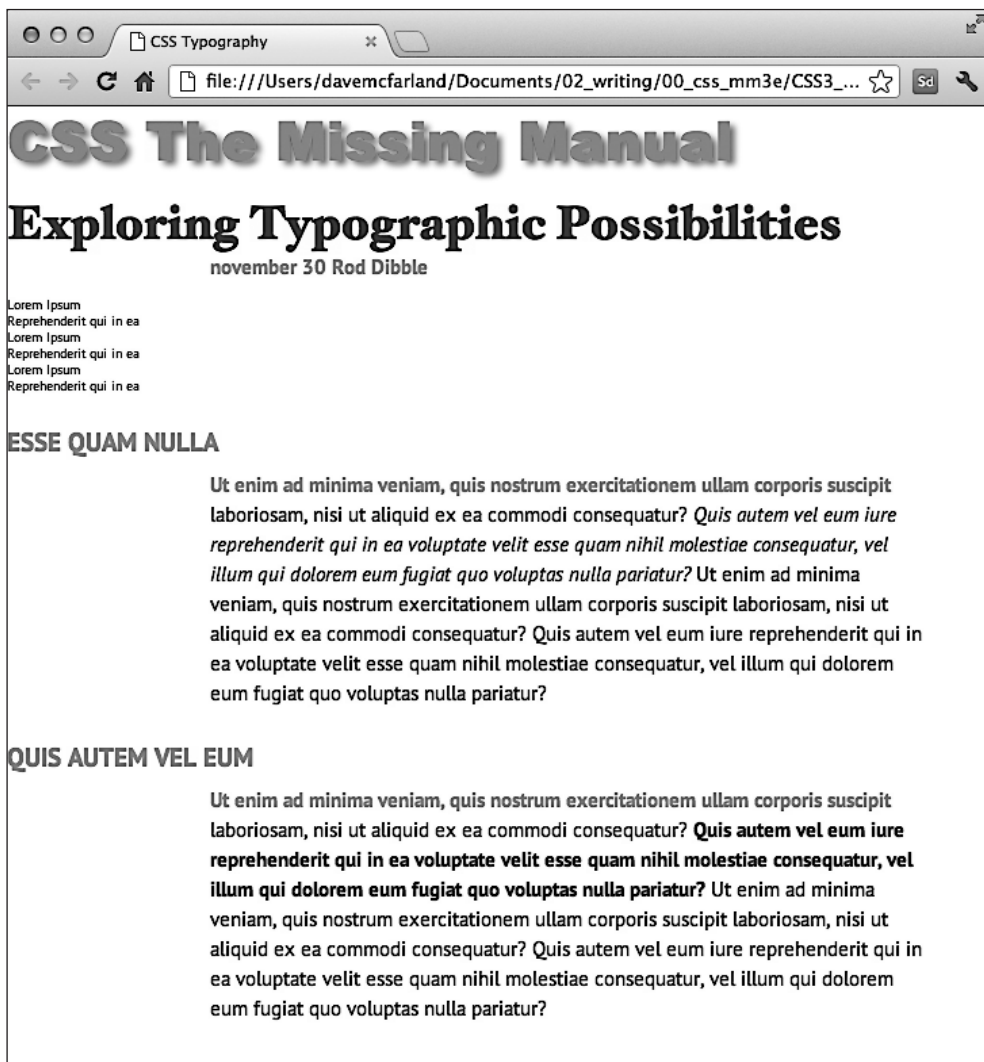
1. Вернитесь к файлу `text.html` в HTML-редакторе. Добавьте следующий стиль в конце внутренней таблицы стилей:

```
.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, этот стиль устанавливает поле шириной 50 пикселей вверху списка, 0 — справа, 25 — внизу и 50 пикселей — слева.

Свойство `width` задает для всего списка ширину 150 пикселей. 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 обертывается вокруг левой стороны списка. Это замечательный метод, а более подробно о плавающих элементах вы узнаете в следующей главе.



**Рис. 6.20.** Внешний вид веб-страницы преобразуется: параметры заголовков, абзацев и основного текста приведены в порядок

Далее определим внешний вид отдельных пунктов списка.

2. Добавьте еще один стиль во внутренней таблице в файле `text.html`:

```
.main li {
  color: rgb(32,126,191);
  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 добавить 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. 6.21.

## 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>
```

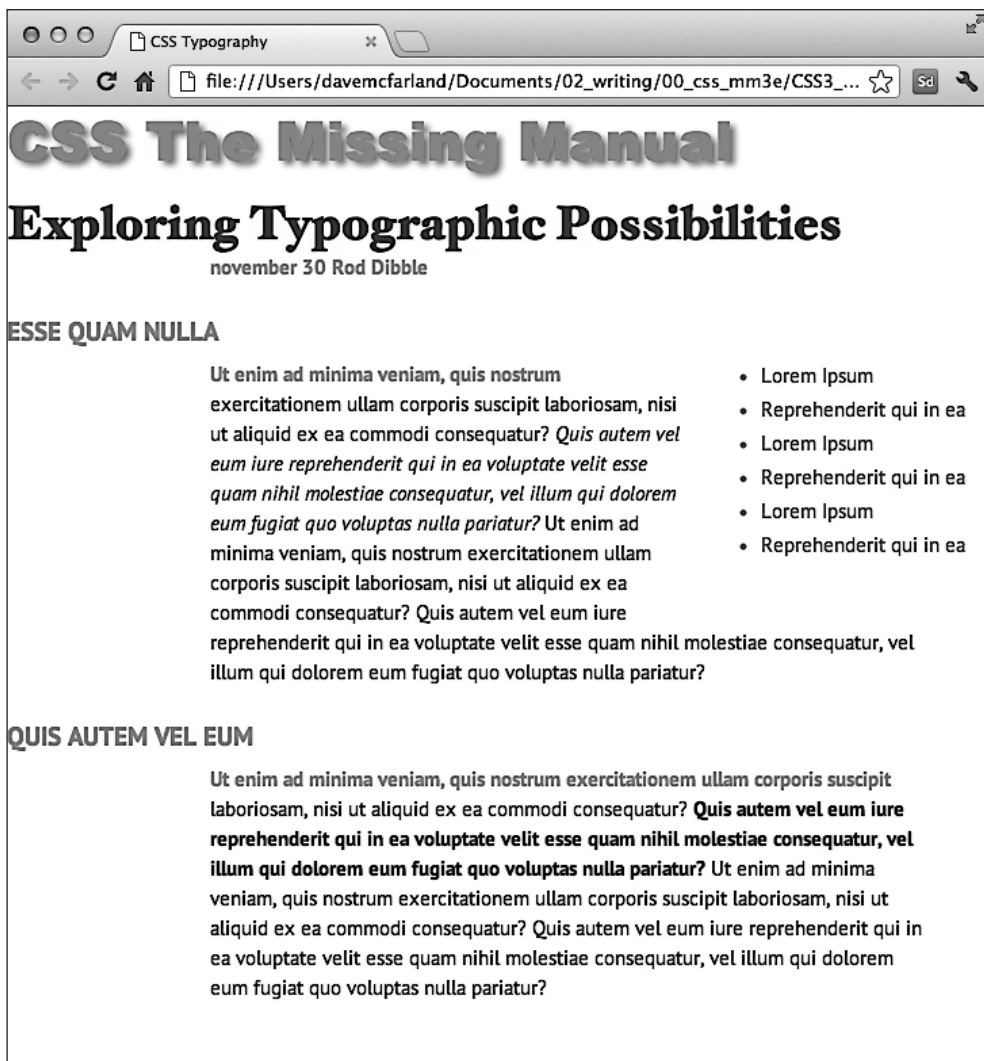
Теперь осталось создать стиливой класс, переопределяющий общие свойства форматирования абзацев на этой странице.

2. Во внутренней таблице стилей рядом с верхней частью страницы добавьте стиль для этого абзаца:

```
.main .byline {
  font-size: 1.6em;
  margin: 5px 0 25px 50px;
}
```

Этот стиль настраивает 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.





**Рис. 6.21.** Свойство float дает интересные возможности при проектировании дизайна. В данном случае маркированный список перемещается к правому краю страницы

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: rgb(32,126,191);
  text-transform: uppercase;
  margin-left: 5px;
}
```

Если вы просмотрите код HTML в шаге 1, то увидите, что имя — Rod Dibble — находится внутри тега `<strong>`. Тег `<strong>` 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, селектор потомка относится к тегу `<strong>`, но только тогда, когда он появляется внутри другого тега с классом `.byline`, и только если они все находятся внутри еще одного тега с классом `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 `<div>`, чтобы они выглядели лучше. Закончим, 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 `html` в верхней части внутренней таблицы стилей и добавьте одно новое свойство (изменения выделены полужирным):

```
html {
  font-size: 62.5%;
  font-family: Arial, Helvetica, sans-serif;
  background: rgb(225,238,253) 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 отмечу, что введенная строка изменяет цвет фона страницы на светло-голубой и добавляет темно-синюю полосу в верхнюю часть страницы.

Далее преобразим основной тег `<div>`.

3. Добавьте еще один стиль между стилем `html` 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 `html`, 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 `<div>`, перед другими стилями, которые форматируют теги внутри этого `<div>`, кажется логичным.

Свойство `width` устанавливает общую ширину этого `<div>` (и содержимого внутри него), по существу превращая страницу в документ шириной 740 пикселей. Значения свойства `margin` здесь — `0 auto` — добавляют 0 пикселей пространства над и под `<div>` и устанавливают для правого и левого полей параметр `auto`, центрирующий этот тег посередине окна браузера. Свойство `padding` добавляет пространство внутри раздела, отталкивая содержимое внутри `<div>` от линии границы. Наконец, мы также поместили изображение в фоне `<div>`.

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 `<strong>`, так что данный селектор потомка 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 пикселей — таков размер шрифта данного стиля.

Один интересный прием позволяет осуществить свойство `margin-right`, для которого установлено 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».

---

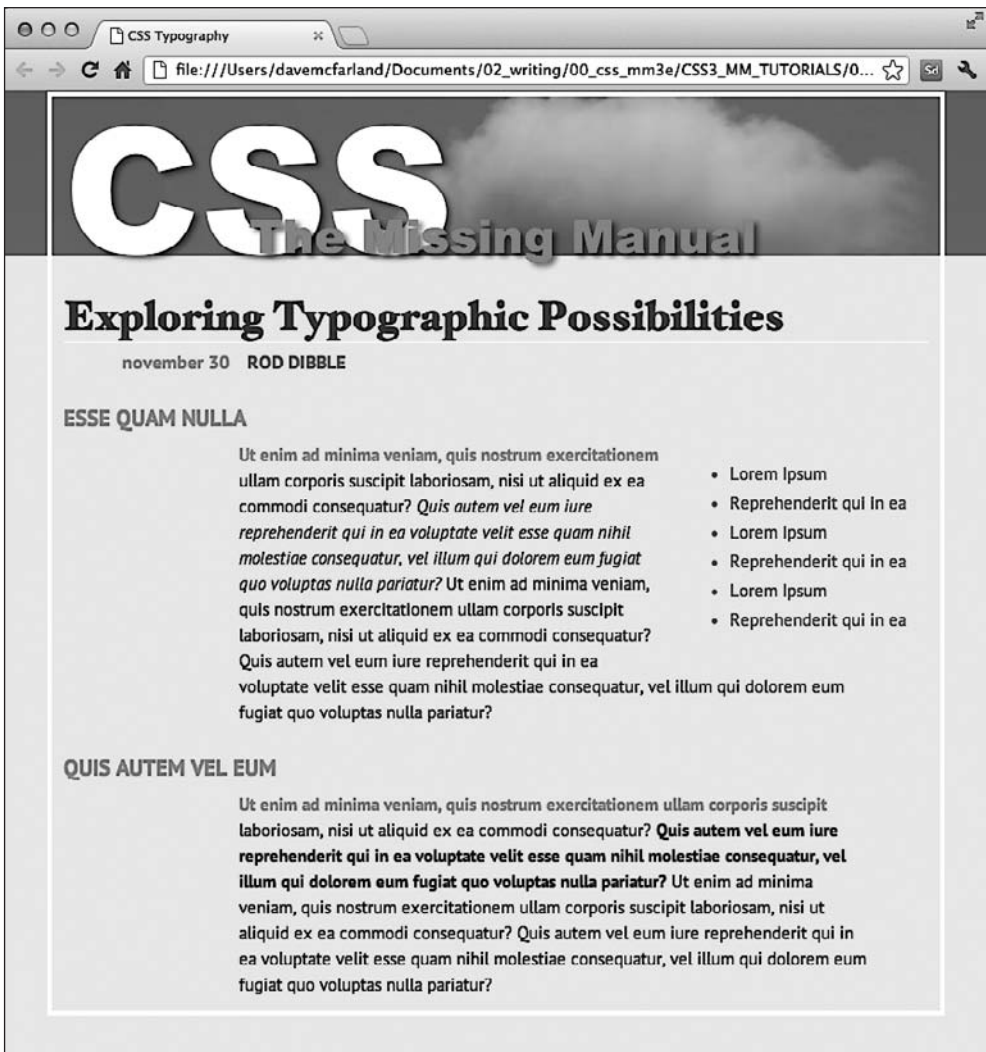
#### ПРИМЕЧАНИЕ

Использование отрицательных полей разрешено в CSS (хотя это и сложная работа). Их даже применяют для некоторых весьма интересных разметок CSS.

---

5. Сохраните файл и просмотрите его в браузере.

Веб-страница должна иметь вид, представленный на рис. 6.22. Теперь вы можете сравнить получившийся файл с законченным вариантом `text.html`, который находится в папке `06_finished` учебного материала.



**Рис. 6.22.** 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 собой 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.

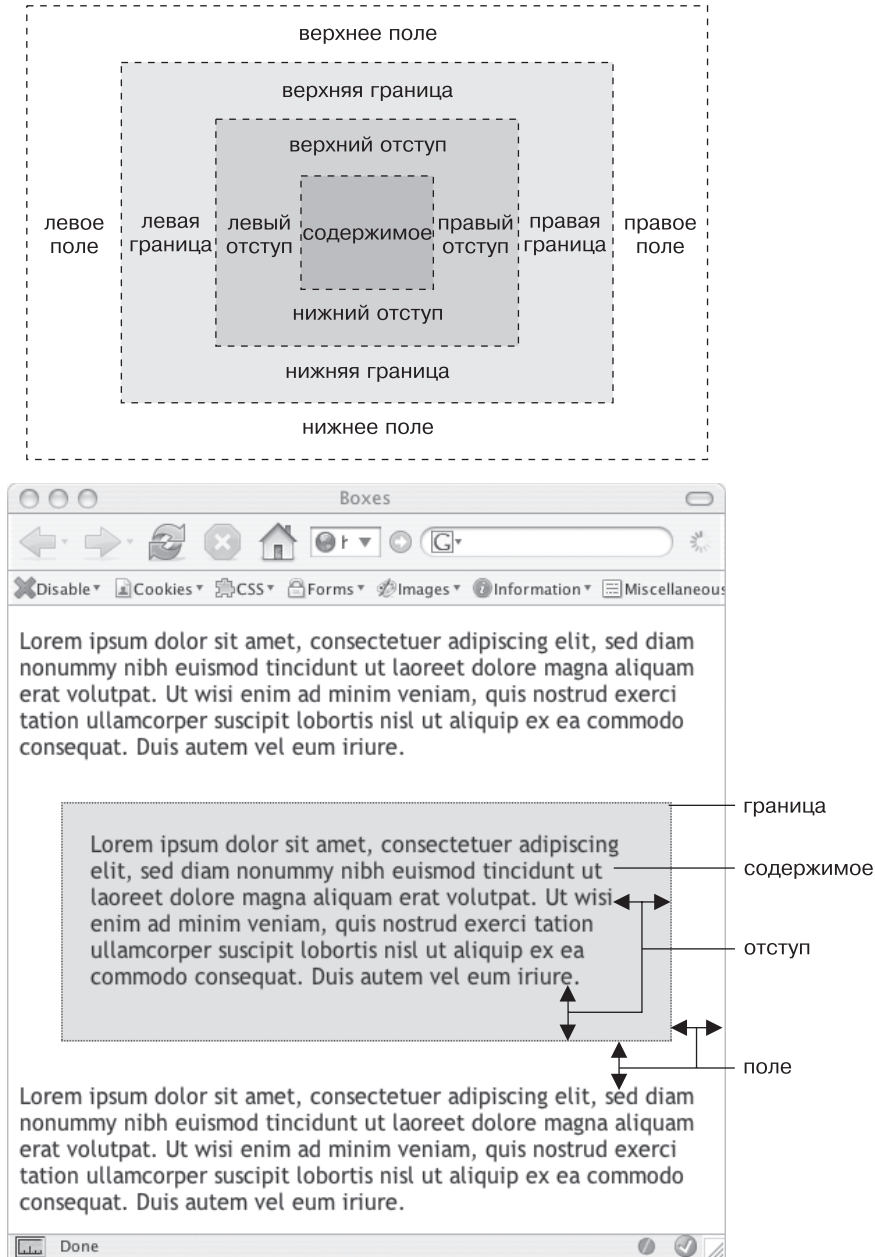
Все то разнообразие свойств, в которое заключено содержимое, образует блок, или контейнер.

`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`.

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.



**Рис. 7.1.** Блочную структуру элемента образует содержимое тега (например, несколько предложений текста), а также отступы, границы и поля

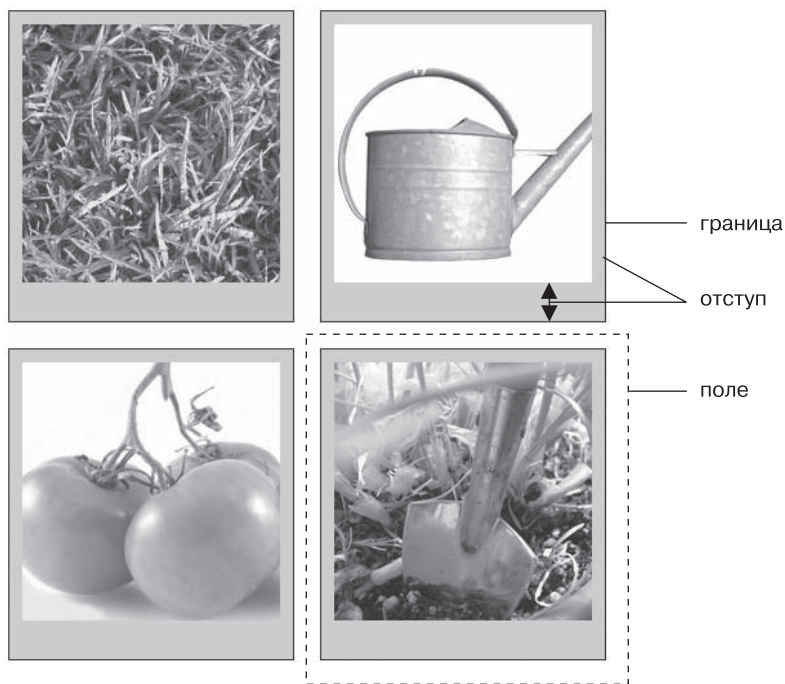
Для заданного тега можно применить любые комбинации. Вы можете установить только поле или добавить границы и отступы и т. д. Если вы сами явно не устанавливаете какие-то из этих свойств, то браузер предопределит их в своих настройках по умолчанию. Как правило, к тегам веб-страниц не добавляют ни отступов, 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 пикселей, то есть промежуток, отделяющий две соседние фотографии, составляет 20 пикселей



На рис. 7.2 благодаря отступам изображения отделены друг от друга и для них установлен серый цвет фона. Вы можете задавать границы и поля для каждой стороны изображения независимо друг от друга. Обратите внимание, что для нижних краев (оснований) фотографий установлены большие 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 различие этих свойств. Отступ добавляет промежуток между содержимым и границей элемента и предотвращает эффект заключения содержимого элемента в рамку. Он также включает область фона, поэтому пространство, занимаемое отступом, может быть свободно от содержимого (текста или фотографии), но может быть заполнено фоновым цветом или изображением. А 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%;
```

Пиксели и `em` используются и работают точно так же, как при форматировании текста (см. раздел «Установка размера шрифта» гл. 6). Поле 20 пикселей 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. Поля, отступы в процентах» далее).

## 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 формиру-

емые 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



### ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

в каждый конкретный момент времени вычисляется на основании текущей ширины окна. Допустим, оно составляет 760 пикселей. При этом левое поле, равное 10 %, добавит промежуток 76 пикселей с левого края стилизуемого элемента. Но если вы измените размеры окна браузера, размер промежутка левого поля тоже изменится. Уменьшение до 600 пикселей изменит размер на 60 пикселей (10 % от 600 пикселей).

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, меню имеет ширину 300 пикселей. 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 пикселям.

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:

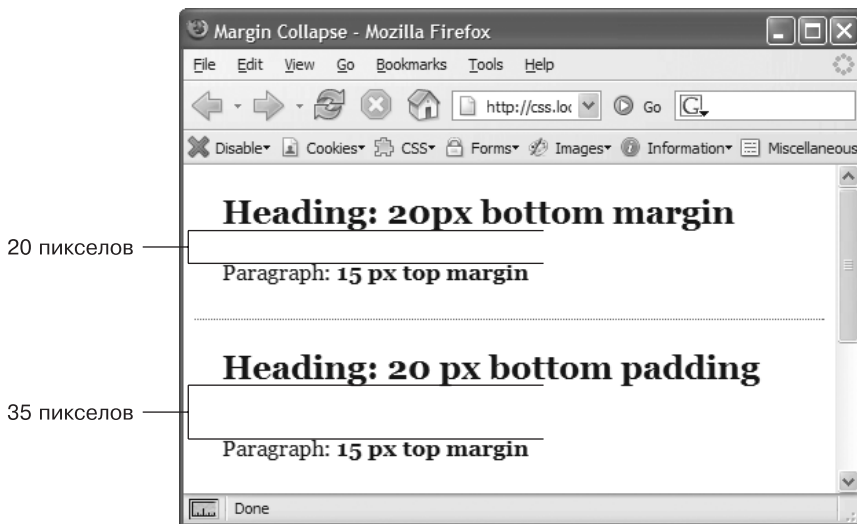
```
padding: 10px;
```

## 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. Точно так же, если верхние и нижние поля (или отступы) изменяются, а правые и левые остаются прежними, можно воспользоваться тремя значениями. Например, объявление `margin: 0 2em 1em`; установит размер верхнего поля равным 0, левого и правого полей — 2 ems, а нижнего поля — 1 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, браузер использует большее из них (рис. 7.3, *вверху*). Предположим, значение нижнего поля маркированного списка установлено равным 30 пикселям, а значение верхнего поля следующего за ним абзаца составляет 20 пикселей. Вместо того чтобы сложить два значения, получив общий промежуток в размере 50 пикселей между списком и абзацем, браузер применяет *наибольшее из двух значений* — в данном случае 30 пикселей. Если вас это не устраивает, используйте вместо полей верхний или нижний отступ (см. рис. 7.3, *внизу*).

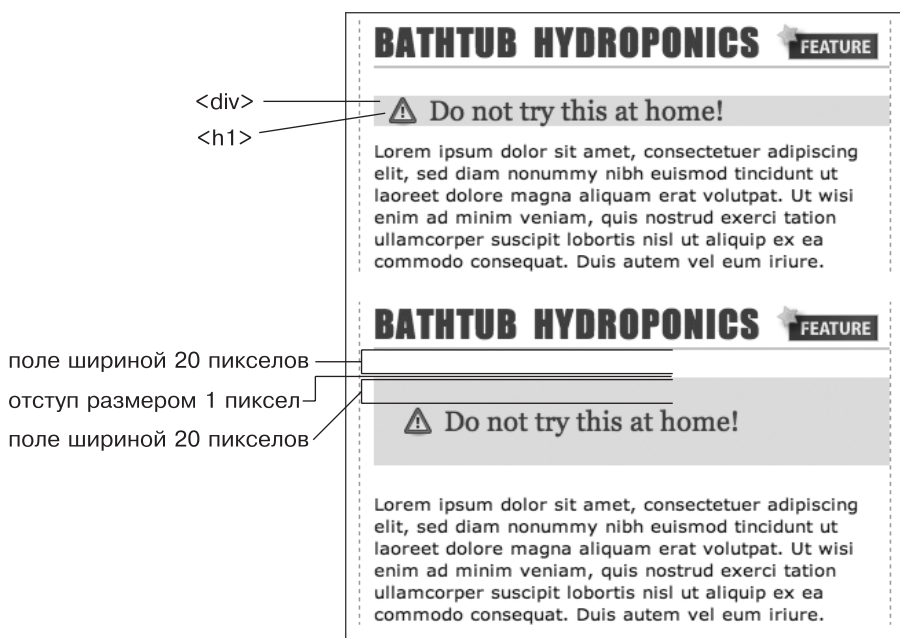


**Рис. 7.3.** В случае соприкосновения двух вертикальных полей меньшее из них игнорируется

На рис. 7.3, несмотря на то что и верхний заголовок имеет нижнее поле в размере 20 пикселей, а у расположенного ниже абзаца текста верхнее составляет 15 пикселей, браузер добавляет промежуток между ними, равный всего 20 пикселям. Чтобы получить тот промежуток, который вы хотели (35 пикселей), используйте вместо полей отступы, как показано в нижней части рисунка. Здесь для заголовка установлен нижний отступ, равный 20 пикселям. Он складывается с верхним полем абзаца, равным 15 пикселям, и получается общий промежуток в размере 35 пикселей.

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 `<div>`). Верхнее и нижнее поля устанавливаются в размере 20 пикселей, 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 `<div>`, устанавливаете для заголовка поле в размере 10 пикселей. Вы, наверное, думаете, что добавили 10-пиксельный промежуток между заголовком и верхним и нижним краем блока `<div>`, но вы не правы (рис. 7.4, *слева*). Вместо этого поле появляется *над* блоком `<div>`. В данном случае не имеет значения, какого размера поле применяется к заголовку, — поле все равно окажется над `<div>`.



**Рис. 7.4.** Как бы ни соприкасались вертикальные поля, в любом случае произойдет конфликт

Чтобы решить эту проблему, добавьте небольшой отступ или границу вокруг элемента-контейнера (в данном случае 1 пиксел нижнего отступа для тега `<div>`).

#### 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 `<div>`. Поскольку *между* этими двумя полями располагаются граница и отступ, поля больше не соприкасаются и заголовок имеет небольшой отделивающий 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 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 пикселей? 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 (с боковой панели или из 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» раздела «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 пикселям. 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 добавить к тексту границы, вы узнаете из следующего раздела). 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 нижнюю границу так, чтобы она находилась ближе к тексту или любому другому содержанию, 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 границу 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 линейных 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 *линейные* (*inline*), которым соответствуют два различных типа тегов.

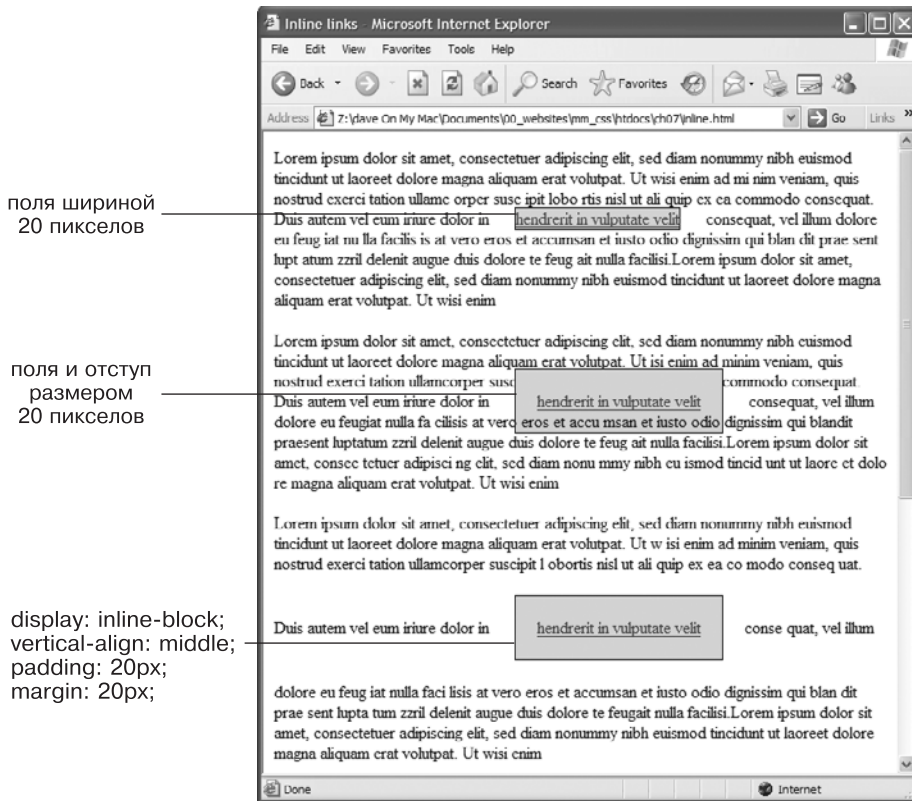
В *блочных* элементах создается разрыв строки перед тегом и после него. Например, тег `<p>` создает блок, отделенный от тегов, расположенных выше и ниже его. Другими примерами являются заголовки, теги `<div>`, таблицы, списки и элементы списков.

*Линейные* (*inline*) элементы не создают разрывов строк ни до, ни после самих тегов. Они отображаются на одной строке с содержимым рядом стоящих тегов. Тег `<strong>` — линейный. Слово, отформатированное с его помощью, будет расположено на одной строке с текстом, который заключен в другие линейные теги, например `<em>`. Было бы очень странно, если бы отдельное слово в середине абзаца, выделенное `<strong>`, вдруг появилось на отдельной строке.

Другими примерами линейных тегов являются `<img>` — для добавления изображений, `<a>` — для создания ссылок, различные теги для полей форм и т. д.

В большинстве случаев CSS работает с линейными и блочными элементами одинаково. Можно применять шрифты, цвет, фоновые параметры, границы к обоим типам элементов. Однако поля и отступы линейных элементов браузеры обрабатывают по-другому. Если добавить поля или отступы слева или справа линейного

элемента, то посредством установки верхнего или нижнего отступа или поля увеличить высоту линейного элемента не удастся. В верхнем абзаце на рис. 7.6 линейному элементу назначено форматирование с применением границ, фонового цвета и полей по 20 пикселей со всех четырех сторон. Но браузер добавляет пустые промежутки только с левой и правой сторон элемента.



**Рис. 7.6.** Добавление к линейному элементу верхнего, нижнего полей 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 линейного 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 линейного элемента, используйте инструкцию `display: inline-block` (см рис. 7.6, *внизу*). Она оставит элемент линейным, 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 7, но только для нормальных линейных элементов, например ссылок, тегов `<strong>`, `<em>` и `<span>`. Кроме того,

вам следует добавить инструкцию `vertical-align:middle`, чтобы Internet Explorer 7 отображал линейный блок таким же образом, как и остальные браузеры.

---

**ПРИМЕЧАНИЕ**

Есть одно исключение из этого правила: если поля или отступы применяются к линейным элементам `<img>`, элементы не изменяют своей высоты. Браузеры корректно изменяют высоту контейнера элемента-изображения, чтобы подогнать ваши отступы и поля.

---

Иногда требуется, чтобы линейные 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 линейный 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 линейный: `display: inline;`

Или, наоборот, вы можете сделать так, чтобы линейные 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.

---

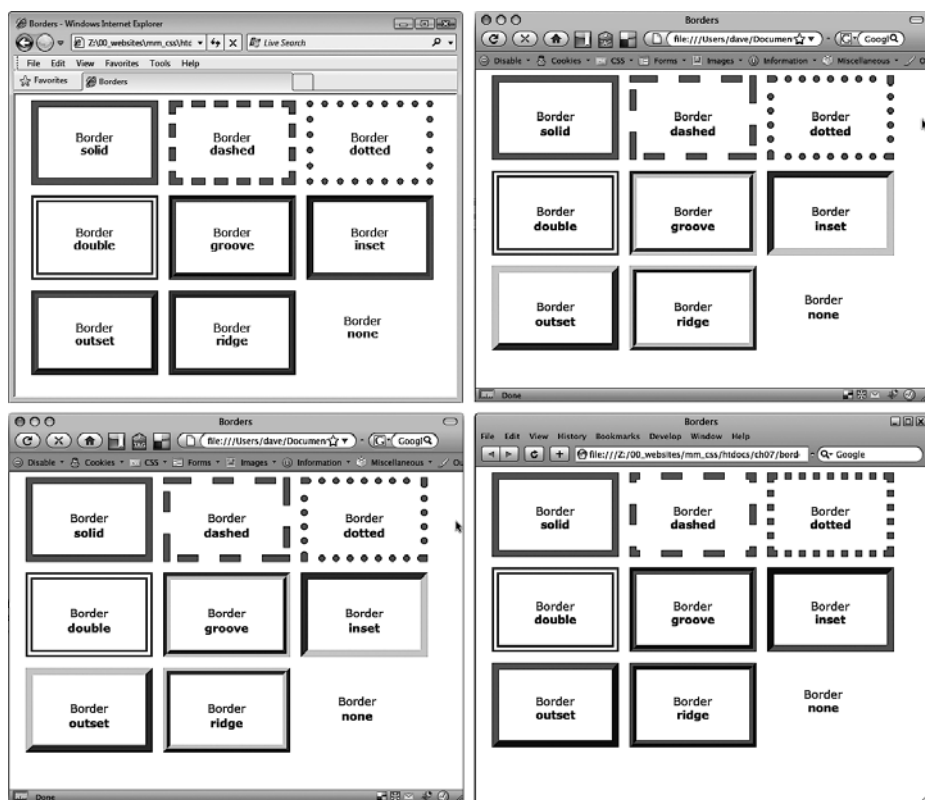
## Добавление границ

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 любой 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 (или RGBA). 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 — пиксели.

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. раздел «Форматирование абзацев текста» гл. 6).



**Рис. 7.7.** Внешний вид стилей границ в следующих браузерах: Internet Explorer 9 (вверху слева), Chrome для Windows (вверху справа), Firefox для Macintosh (внизу слева) и Safari для Macintosh (внизу справа)



## 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: цвета, ширины (толщины) и стиля для каждой из четырех границ. Наиболее простое и понятное свойство — border, которое просто добавляет границы с заданными параметрами:

```
border: 4px solid #FF0000;
```

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 #FF0000; работает так же, как border: #FF0000 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 разделе «Придание тексту цветового оформления» гл. 6. При желании вы можете окрасить фон веб-страницы в яркий зеленый цвет, указав следующий код:

```
body { background-color: rgb(109,218,63); }
```

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. Кроме того, к фону любого элемента можно добавить цветовой градиент.

---

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 боковая панель.

И не забывайте о ранее рассмотренном методе задания цвета RGBA. С его помощью можно сделать фон полупрозрачным, позволяя просматриваться находящимся за ним цветам, текстурам или изображениям, принадлежащим другим объектам. Например, можно сделать желто-коричневый фон страницы. Затем, предположим,

что возникло желание придать находящемуся внутри тегу `<div>` более светлый оттенок желто-коричневого цвета. Вместо задания для фона элемента `div` сплошного цвета можно добавить белый цвет, а затем управлять прозрачностью этого цвета, чтобы через него просвечивались различные оттенки желто-коричневого цвета:

```
body {  
  background-color: rgb(247,226,155);  
}  
.special-div {  
  background-color: rgba(255,255,255,.75);  
}
```

---

**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.

---

## Создание скругленных углов

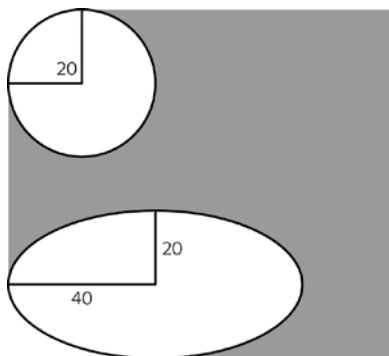
Как уже ранее упоминалось, веб-браузеры рассматривают все элементы как строго прямоугольные блоки. Это становится очевидным при установке границы вокруг абзаца или `div`-элемента. К счастью, есть возможность сгладить острые углы таких прямоугольников, добавив к стилям скругленные углы (рис. 7.8). В CSS3 имеется свойство `border-radius`, позволяющее дизайнерам добавлять скругления к одному или нескольким углам элемента. В самом простом варианте свойство `border-radius` получает одно значение, которое затем применяется ко всем четырем углам элемента:

```
.specialBox {  
  background-color: red;  
  border-radius: 20px;  
}
```



**Рис. 7.8.** В CSS3 позволяет скруглять углы любого элемента. Чтобы увидеть любое из этих удивительных закруглений, элемент должен иметь цветной фон или границу

Браузер использует предоставленное значение радиуса для создания круговой траектории в каждом углу элемента. Как показано на рис. 7.9, значение, равное расстоянию от центра окружности до ее края, является ее радиусом. В качестве единиц измерения чаще всего используются пиксели и `em`, но можно также применять и проценты (хотя они ведут себя немного неожиданно).



**Рис. 7.9.** Можно создавать не только скругленные (*вверху*), но и эллиптические углы (*внизу*), предоставляя либо одно значение — `border-radius: 20px`, либо комбинацию из двух значений, разделенных символом слеша, — `border-radius: 40px/20px`;

При использовании единственного значения браузер рисует закругления одинакового радиуса для каждого угла элемента. Например, для верхнего левого изображения на рис. 7.8 используется следующее объявление:

```
border-radius: 30px;
```

Но предоставлением одного и того же значения для каждого угла настройки не ограничиваются. Для каждого угла можно предоставить отдельные значения, задав четыре параметра. Например, у верхнего правого блока на рис. 7.8 четыре разных угла. Объявление имеет следующий вид:

```
border-radius: 0 30px 10px 5px;
```

Сначала задается числовое значение для левого верхнего угла блока, а затем по часовой стрелке — для всех остальных углов. То есть первое значение (0 в примере на рис. 7.8) применяется к левому верхнему, второе (30px) — к правому верхнему, третье (10px) — к правому нижнему, четвертое (5px) — к левому нижнему углу. Можно также задать только два значения, тогда первое число будет применено к левому верхнему и правому нижнему углам, а второе к правому верхнему и левому нижнему углам.

Кроме рассматриваемых до сих пор абсолютно круглых углов (то есть представляющих собой часть окружности), можно также задавать эллиптические углы, подобные тем, что показаны в двух нижних примерах на рис. 7.8. Для эллиптической формы угла требуется два значения радиуса: первое для радиуса от центра до левого или правого края, а второе для определения расстояния от центра до верхнего или нижнего края. Например, чтобы добавить углы, показанные в левом нижнем углу на рис. 7.8, нужно создать следующее объявление:

```
border-radius: 40px/20px;
```

Значение 40px задает горизонтальный радиус, а значение 20px — вертикальный радиус. Слеш между ними оповещает браузер о создании эллиптического угла. Можно сделать так, что у каждого из четырех углов будут разные вытянутые формы, предоставив четыре набора следующих значений:

```
border-radius: 40px/20px 10px/30px 20px/40px 10px/20px;
```

Можно даже смешивать эллиптические и скругленные углы:

```
border-radius: 10px 10px/30px 20px/40px 10px;
```

И наконец, если нужно пойти по более длинному пути, то для определения формы каждого угла можно воспользоваться отдельными свойствами. Например:

```
border-radius: 1em 2em 1.5em .75em;
```

можно также написать следующим образом:

```
border-top-left-radius: 1em;  
border-top-right-radius: 2em;  
border-bottom-right-radius: 1.5em;  
border-bottom-left-radius: .75em;
```

Если использовать одно процентное значение, то, скорее всего, у вас получится эллиптический угол. Дело в том, что браузер вычисляет горизонтальный радиус, используя процентное вычисление от ширины элемента, а для вычисления вертикального радиуса применяется процентное вычисление от высоты этого элемента. Поэтому, если создать объявление `border-radius: 20%` для элемента, не представляющего собой правильный квадрат, браузер вычислит эллиптический угол, похожий, скорее всего, на тот, который получается при объявлении `border-radius: 20px/40px;`.

#### ПРИМЕЧАНИЕ

Internet Explorer 8 и более ранние версии не понимают свойство `border-radius`, поэтому при его объявлении будут показаны прямые углы. Кроме того, версия 3.2 браузера Safari для iOS и версия 2.1 Android Browser требуют префикса производителя `-webkit-`, например: `-webkit-border-radius: 20px;`. Дополнительная информация дана в следующей врезке.

### ИНФОРМАЦИЯ ДЛЯ НОВИЧКОВ

#### Префиксы производителей

Правила CSS постоянно развиваются. Команда W3C (World Wide Web Consortium) разрабатывает новые CSS-свойства, поддержка которых затем добавляется в новые веб-браузеры. Но иногда создатели браузеров сами придумывают новые CSS-свойства, которые представляются им интересными, и добавляют их к своим браузерам. А иногда специалисты W3C придумывают новые CSS-свойства, которые производители браузеров не спешат добавить в свои браузеры.

В течение периода разработки новых CSS-свойств и стандартов производители браузеров проявляют осторожность. Они не хотят окончательно связывать себя обязательствами по поддержке тех CSS-свойств, которые могут измениться. Точно так же при проведении экспериментов с CSS-свойствами собственной разработки производители браузеров не смеют утверждать, что они придумали согласованный стандарт. Чтобы пометить CSS-свойство как экспе-

## ИНФОРМАЦИЯ ДЛЯ НОВИЧКОВ

риментальное или еще не до конца согласованное, производители браузеров используют префикс, который ставится перед названием свойства. Каждый из основных производителей браузеров использует собственный префикс:

- `-webkit-` — применяется Chrome, Safari и другими браузерами на базе WebKit;
- `-moz-` — используется Mozilla Firefox;
- `-o-` — применяется Opera;
- `-ms-` — используется Microsoft Internet Explorer.

Когда работа над свойством еще не завершена, браузер может поддерживать версию с префиксом производителя. Например, когда было впервые предложено свойство `border-radius`, Firefox поддерживал свойство `-moz-border-radius`, а Safari поддерживал свойство `-webkit-border-radius`.

Когда свойству требуется префикс производителя, то для получения одного и того же эффекта обычно используется несколько строк кода, по одной для каждого производителя, а в завершение — версия без префикса:

```
-moz-box-sizing: border-box;
-webkit-box-sizing: border-box;
box-sizing: border-box;
```

Обычно, если W3C CSS Working Group принимает свойство и в достаточной степени завершает его детализацию, производители отказываются от префикса. Например, сегодня все основные браузеры просто поддерживают свойство `border-radius`, которому теперь не требуется префикс производителя. В книге будут встречаться CSS3-свойства с не до конца отработанными особенностями, для использования которых может понадобиться префикс производителя. Необходимость применения таких префиксов будет особо оговариваться, наряду с рассмотрением порядка их использования с тем или иным свойством.

## Добавление теней

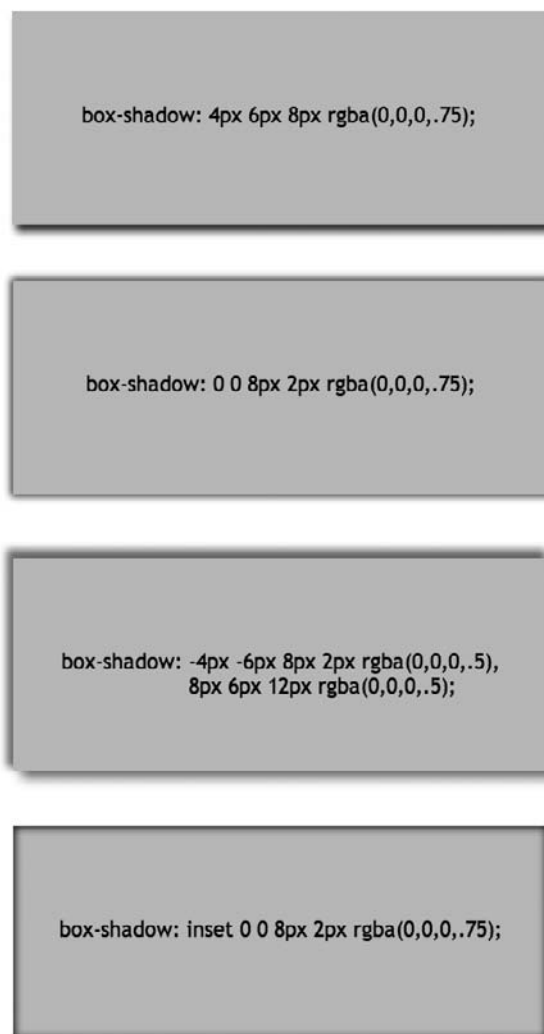
Как уже упоминалось, чтобы сделать текст более рельефным, к нему можно добавить еле заметную (или весьма заметную) тень. Для добавления теней к блоку, обрамляющему элемент, в CSS3 включено свойство `box-shadow`, позволяющее, например, блоку `<div>` казаться парящим над страницей (рис. 7.10).

Это свойство работает в большинстве современных браузеров, включая Internet Explorer 9. К сожалению, Internet Explorer 9 рисует тени заметно тоньше, чем другие браузеры. Кроме того, Internet Explorer 8 и более ранние версии просто игнорируют это свойство и не рисуют тени для элементов.

По сравнению со свойством `text-shadow`, у этого свойства есть несколько дополнительных настроек. Например, можно сделать так, чтобы тень появлялась внутри блока, как показано в нижней части рис. 7.10.

Основной синтаксис свойства `box-shadow` показан на рис. 7.11. Первое значение задает горизонтальное смещение, то есть это значение приводит к перемещению тени влево или вправо от элемента. Положительное число приводит к перемещению тени вправо (верхний блок на рис. 7.10), а отрицательное число — влево. Основными позициями тени является положение либо слева (при отрицательном значении, как показано на рисунке), либо справа (при положительном значении), либо выше верхнего края (при отрицательном значении), либо ниже нижнего края (при положительном значении) элемента с ее размытием с помощью указанного радиуса.





**Рис. 7.10.** CSS3-свойство `box-shadow` позволяет добавлять тени элементам

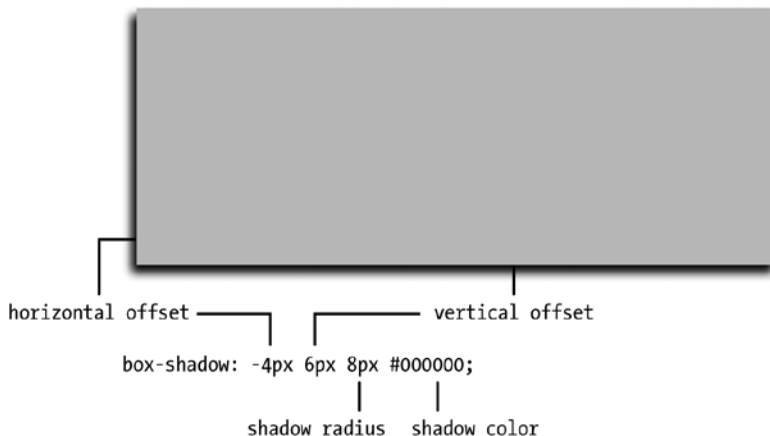
Второе значение задает вертикальное смещение — позицию тени либо над элементом, либо под ним. При положительном значении тень помещается ниже нижнего края блока (верхний блок на рис. 7.10), а при отрицательном значении тень помещается над верхним краем блока.

**ПРИМЕЧАНИЕ**

Для задания значений тени нужно использовать пиксели или `em`. Проценты работать не будут.

Третье значение определяет радиус размытия тени. Оно определяет степень размытости и ширины тени. Значение 0 не придает эффекта размытости, то есть

получается тень с четкими краями. Чем выше значение, тем более размытой и тусклой становится тень.



**Рис. 7.11.** Основной синтаксис свойства `box-shadow`

И наконец, последнее значение указывает цвет отбрасываемой тени. Можно воспользоваться любым цветовым значением CSS, но RGBA-значения смотрятся намного лучше, потому что позволяют управлять прозрачностью цвета, чтобы сделать тень полупрозрачной и больше похожей на настоящую.

У свойства `box-shadow` есть два дополнительных значения: ключевое слово `inset` и значение расширения (`spread`). Ключевое слово `inset` заставляет браузер рисовать тень внутри блока (нижний блок на рис. 7.10). Для создания внутренней тени нужно просто добавить `inset` в качестве первого значения свойства `box-shadow`:

```
box-shadow: inset 4px 4px 8px rgba(0,0,0,.75);
```

В качестве четвертого значения (между радиусом размытия тени и ее цветом) можно также добавить значение расширения. Это приведет к расширению тени на указанное значение. Иначе говоря, если добавить значение расширения, равное `10px`, браузер расширит тень на 10 пикселей в каждом направлении (тень получится на 20 пикселей шире и на 20 пикселей выше). Это значение также задает то место, где применяется радиус размытия. То есть, когда добавляется значение расширения, размытие тени не начинается до тех пор, пока не будет применено значение расширения. В частности, это пригодится в том случае, когда нужно добавить тень вокруг всего элемента для создания эффекта, который во многих программах редактирования изображений называется свечением (`glow`).

#### ПРИМЕЧАНИЕ

Браузер Android и устаревшие версии Safari для iPhone для работы свойства `box-shadow` требуют использования префикса производителя, рассмотренного в предыдущей врезке. То есть, чтобы заставить тени блоков работать в этих, а также в новых браузерах, нужно добавить к стилю два объявления:

```
-webkit-box-shadow: 2px 2px 10px #000000;
```

```
box-shadow: 2px 2px 10px #000000;
```

Например, во втором сверху блоке на рис. 7.10 горизонтальное и вертикальное смещения установлены в 0, для радиуса тени задано значение 8px, а для расширения — значение 2px. Значение расширения выталкивает тень на 2 пиксела наружу относительно всех четырех сторон блока, а затем радиус тени, равной 8 пикселям, расширяет размытие еще на 8 пикселей. Значение расширения можно даже использовать для создания вокруг существующей границы второй, по-другому расцвеченной тени. Вот как выглядит пример кода для этого эффекта:

```
border: 10px solid rgb(100,255,30);  
box-shadow: 0 0 0 10px rgb(0,33,255);
```

И наконец, можно даже применить к стилю несколько теней (второй блок снизу на рис. 7.10). Нужно просто поставить запятую после первого набора установок тени, а затем добавить еще одну тень:

```
box-shadow: 10px 5px 8px #FF00FF,  
            -5px -10px 20px 5px rgb(0,33,255);
```

Количество теней не ограничено (или ограничено только здравым смыслом).

---

#### ПРИМЕЧАНИЕ

К сожалению, браузеры рисуют тени множеством вариантов. Internet Explorer 9, например, создает тень заметно более тонкую, чем другие браузеры. Чтобы визуально сравнить тени блоков, рисуемые браузерами, зайдите на сайт по адресу <http://thany.nl/apps/boxshadows/>.

---

## 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, боковая панель. 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 панелей или галерей эскизов.

Разработка стилей с этими свойствами не составляет сложностей. Просто наберите их со значением в любой единице измерения CSS, которые мы изучили. Например:

```
width: 300px;  
width: 30%;  
height: 20em;
```

Пиксели 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 пиксела, а если вы установите ширину равной 2 `em`, она составит 2 · 24, или 48 пикселей. Если вы не определите в стиле размер шрифта текста, то он будет взят из унаследованных параметров.

## ПРИМЕЧАНИЕ

В CSS3 представлены дополнительные единицы измерения под названием `rem`, которые основаны на значении свойства `font-size` элемента `html`. Эти единицы измерения работают во всех основных браузерах, за исключением Internet Explorer 8 и более ранних версий. Эта единица измерения уже рассматривалась в данной книге.

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 пикселей (возможно, для создания столбца), то ширина данного заголовка составит 150 пикселей. Процентные значения в свойстве высоты `height` работают точно так же, но расчет базируется на высоте элемента-контейнера, а не на его ширине.

## Вычисление фактических размеров блочных элементов

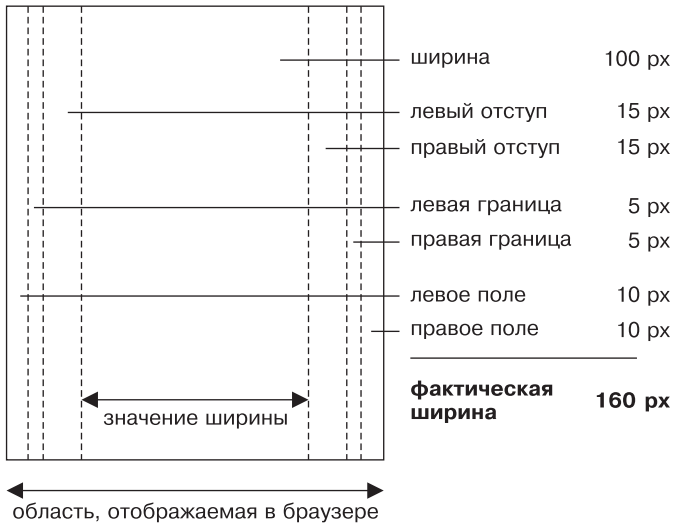
Свойства `width` и `height` на первый взгляд кажутся довольно простыми и понятными, однако есть несколько нюансов, вводящих начинающих веб-дизайнеров в заблуждение. Прежде всего, существует различие между значениями ширины и высоты, которые вы явно указываете при написании стилей, и размером пространства, 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. 7.12.

Допустим, вы назначили следующие свойства:

```
width: 100px;
padding: 15px;
border-width: 5px;
margin: 10px;
```

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 пикселей). Конечно, придется выполнить несложные арифметические операции, чтобы выяснить общий точный размер. В представленном выше примере для размещения стилизуемого элемента браузером отводится пространство шириной

160 пикселей: 20 пикселей для левого и правого полей, 10 для левой и правой границ, 30 для левого и правого отступа и 100 пикселей в качестве ширины основного содержимого. Версии браузера Internet Explorer старше 6-й неправильно работают с этими параметрами, поэтому для них требуется сделать кое-какие доработки веб-страницы.



**Рис. 7.12.** Вычисление фактической ширины блока стилизуемого элемента выполняется путем сложения ее составляющих

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 пикселями. Но если вы не уверены полностью в точных размерах содержимого внутри тега, то можете столкнуться с некоторыми нежелательными результатами (рис. 7.13). В этом примере блок с цитатой, который используется для того, чтобы акцентировать внимание на интересном отрывке из статьи, имеет ширину и высоту по 100 пикселей. Когда в блок добавляется больше текста, чем может уместиться в такую высоту, его содержимое 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.

Другими словами, свойство `height` полезно для контроля высоты элемента `<div>`, 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.



**Рис. 7.13.** Когда устанавливается высота элемента (например, высота `div`-элемента правой боковой панели, как в данном примере), содержимое которого выше самого элемента, браузеры просто позволяют содержимому выйти за нижнюю границу элемента

## СОВЕТ

Область баннеров на странице является еще одним подходящим кандидатом для установки высоты. Обычно баннер имеет ограниченное содержимое: логотип, поле поиска, может быть, какие-нибудь навигационные кнопки. Зачастую у баннеров остается довольно много пустого пространства (помогающего привлечь внимание посетителя к ключевым элементам навигационной панели), поэтому указание высоты для баннера обычно не вызывает проблем.

## Переопределение ширины блока с помощью `box-sizing`

Как уже упоминалось, веб-браузеры традиционно вычисляют ширину элемента, складывая значения свойств `border`, `padding` и `width`. Это не только заставит вас проводить математические вычисления (этого только не хватало!), чтобы определить реальную ширину отображаемого элемента, но может вызвать некоторые другие проблемы. Особенно это касается случаев создания плавающих разметок с использованием процентных отношений. Подробности создания плавающих разметок нам еще предстоит изучить, но если говорить вкратце, CSS позволяет с помощью свойства `float` помещать элементы бок о бок, что дает возможность создавать разметки, состоящие из нескольких столбцов.

Когда для нескольких столбцов используется процентное отношение, могут возникать довольно странные проблемы. Предположим, что есть два столбца (на самом деле два таких тега, как `<div>`) и нужно, чтобы каждый занимал 50 % ширины

окна. Соответственно, для двух столбцов устанавливается ширина 50 %, но на тот момент, когда добавляется отступ или рамка к одному из столбцов, вы увеличиваете его ширину, которая становится больше 50 % (если точнее, она составляет 50 % плюс значение левого и правого отступов и значение ширины левой и правой частей). В большинстве случаев это заставит второй столбец опуститься ниже первого.

К счастью, CSS3 предлагает свойство, позволяющее изменить порядок вычисления браузером экранной ширины (и высоты) элемента. Свойство `box-sizing` предоставляет три варианта.

- `content-box` устанавливает ранее рассмотренный способ, с помощью которого браузеры всегда определяют экранную ширину и высоту элемента. То есть браузер добавляет ширину рамки и толщину отступа к значениям, установленным для свойств ширины и высоты, чтобы определить экранную ширину и высоту заданного тегом элемента. Поскольку это поведение является исходным, указывать что-нибудь для `content-box` не нужно.
- `padding-box` сообщает браузеру, что при установке для стиля свойства ширины или высоты они должны включать отступы как часть своего значения. Например, предположим, что есть элемент с отступами слева и справа, равными 20 пикселям, ширина которого установлена равной 100 пикселям. Браузер будет рассматривать часть, приходящуюся на отступы, частью этого 100-пиксельного значения. То есть ширина области содержимого будет равна всего лишь 60 пикселям (100 – 20 [левый отступ] – 20 [правый отступ]).
- `border-box` сообщает браузеру о необходимости включения в качестве составляющей части значений свойств ширины и высоты толщину как отступа, так и рамки. Эта настройка решает проблему использования значений, выраженных в процентном отношении, о которой говорилось выше. То есть, когда для свойства `box-sizing` установлено значение `border-box`, при задании ширины элемента, равной 50 %, этот элемент будет занимать до 50 % пространства, даже если к нему будут добавлены отступы и рамки.

Если вам не нравится стандартный способ вычисления браузером элементов ширины и высоты, воспользуйтесь значением `border-box`. (Если, конечно, у вас нет какой-нибудь особой причины, по которой вы хотите включить в расчеты отступ, но не желаете включать туда еще и рамку.) Чтобы воспользоваться свойством `box-sizing`, просто предоставьте ему одно из трех значений из списка. Например:

```
box-sizing: border-box;
```

Кроме того, Firefox (по крайней мере на момент написания книги) еще не поддерживал стандартное название свойства, поэтому нужно воспользоваться ранее упоминавшейся версией с префиксом производителя, это же требование остается в силе и для устаревших версий Safari для iOS и Android (версии 3 и ниже). Чтобы обеспечить работу свойства `box-sizing` во всех этих браузерах, нужно создать три объявления:

```
-moz-box-sizing: border-box;  
-webkit-box-sizing: border-box;  
box-sizing: border-box;
```

Многие веб-разработчики настолько оценили пользу настройки `border-box`, что создали универсальный селектор стиля, применяемый к каждому элементу на странице:

```
* {  
-moz-box-sizing: border-box;  
-webkit-box-sizing: border-box;  
box-sizing: border-box;  
}
```

В гл. 13 будет показано, что особая польза от применения `border-box` будет при создании веб-дизайна, зависящего от использования CSS3, который меняет размеры с целью вписаться в размеры различных устройств (таких как iPhone, планшетный компьютер и настольный монитор).

---

#### ПРИМЕЧАНИЕ

Свойство `box-sizing` работает даже в Internet Explorer 8 и более старших версиях, то есть оно поддерживается более чем 95 % используемых браузеров. К сожалению, Internet Explorer 7 не понимает этого свойства, поэтому при использовании настройки `border-box` Internet Explorer 7 ее проигнорирует и нарисует по сравнению с другими браузерами более широкие (и более высокие) элементы. Если вам необходимо сохранить поддержку Internet Explorer 7, это свойство лучше не использовать.

---

## 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. 7.13 показано, что в 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, которое выходит за пределы блочного элемента.

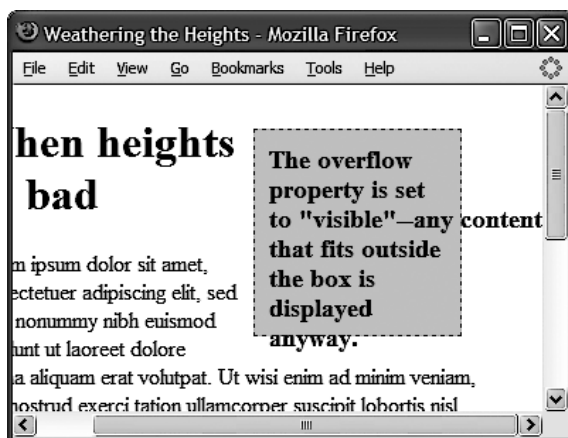
`Visible` — это значение, принимаемое браузером по умолчанию. Указание этого ключевого слова имеет тот же эффект, что и отсутствие установки свойства `overflow` (рис. 7.14, *вверху*).

`Scroll` — позволяет добавить полосы прокрутки (см. рис. 7.14, *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 (см. рис. 7.14, *внизу*). Это значение небезопасно, поскольку может привести к тому, что часть содержимого будет не видна. Но иногда оно применяется при создании плавающих разметок.





**Рис. 7.14.** Свойство `overflow` предоставляет три простых способа отображения текста, размеры которого не позволяют браузеру показать его внутри блочного элемента

## Задание максимальных и минимальных значений высоты и ширины

Если вы еще не поняли, хочу вас заверить, что CSS предлагает множество гибких решений. Кроме стандартных свойств `width` и `height`, можно воспользоваться еще четырьмя вариантами.

- `max-width` задает максимальную ширину элемента. Этот элемент может быть уже установленного предела, но он не может быть шире его. Данный вариант пригодится в том случае, когда ваша страница должна менять свои размеры, чтобы поместиться на дисплеях разной ширины. Но при этом она не должна становиться настолько широкой, чтобы это затрудняло ее чтение на слишком больших мониторах. Предположим, например, что к странице добавлен следующий стиль:

```
body {  
  max-width: 1200px;  
}
```

Этот стиль позволяет странице переформатировать текст, чтобы поместиться по ширине на небольших дисплеях, встречающихся у смартфонов или у планшетных компьютеров. Но на действительно больших настольных мониторах страница не должна быть шире 1200 пикселей, то есть страница не может разрастаться в ширину, при которой ее уже невозможно читать.

- `max-height` работает во многом похоже на `max-width`, за исключением того, что `max-height` применяется к высоте элемента. Но, как уже упоминалось, с высотой элемента лучше все же не связываться.
- `min-width` устанавливает минимальную ширину элемента. Элемент может растянуться шире значения минимальной ширины, но никогда не может стать уже этого значения. Если, например, вы заметили, что при изменении размеров окна вашего браузера элементы становятся настолько узкими, что разметка разваливается, можно с помощью следующего кода установить минимум:

```
body {  
  min-width: 760px;  
}
```

Если посетитель уменьшит окно своего браузера, оставив только 500 пикселей ширины, браузер добавит полосу прокрутки, не делая элементы на странице слишком узкими.

- `min-height` работает точно так же, как и свойство `min-width`, но применительно к высоте. Это свойство может решить проблему, показанную на рис. 7.13. Используя минимальную высоту, вы заставляете веб-браузер сделать элемент по крайней мере имеющим определенную высоту. Если содержимое элемента выше, то браузер сделает выше весь элемент.

Свойства минимума и максимума можно также применять вместе. Например, если нужно гарантировать минимальную ширину страницы 760 пикселей,

но не дать ей расширяться более чем до 1200 пикселей, можно создать следующий стиль:

```
body {  
    min-width: 760px;  
    max-width: 1200px;  
}
```

## 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. 7.15, *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 боковую панель с относящейся к тексту информацией и управляющие ссылки к одной из сторон веб-страницы.

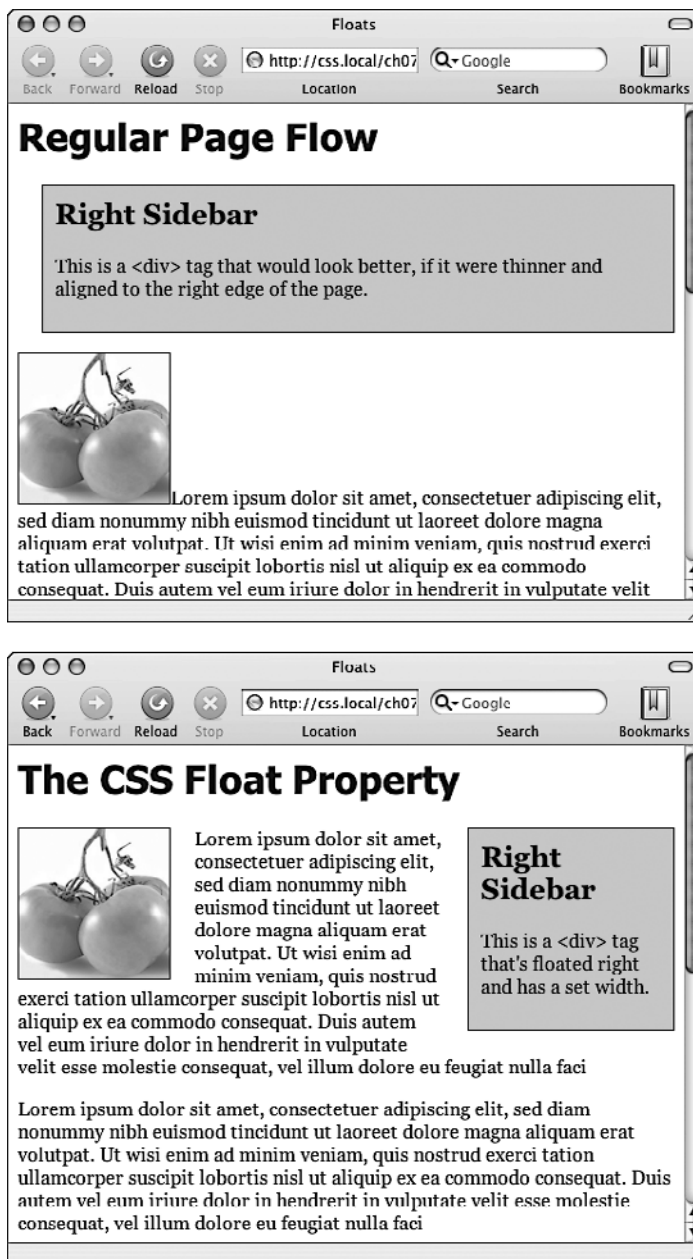
Свойство `float`, позволяющее избавиться от однообразия, является одним из самых мощных и полезных средств, которые предлагает CSS. Широкий диапазон его применения включает не только простое позиционирование, перемещение изображений к одной из сторон абзаца, но и обеспечивает полный контроль по управлению 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. 13), 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;
```



**Рис. 7.15.** Стандартная последовательность отображения HTML (вверху) и вид страницы при использовании свойства float (внизу)

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 пикселей, 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 пикселей, а не к окну браузера.

Вы можете применять свойство `float` к линейным элементам, например `<img>`. Надо сказать, использование выравнивания по левому или правому краю для фотографий — самый обычный, даже наиболее распространенный способ применения `float`. Браузер обрабатывает линейные элементы точно так же, как блочные. Поэтому, используя отступы и поля в линейных элементах, вы избежите проблем, которые обычно возникают в такой ситуации.

Вы можете также использовать `float` для блочных элементов, таких как заголовок или абзац текста (или один из новых элементов HTML5, например `<article>`, `<section>` или `<aside>`). Общая методика применения свойства к тегу `<div>`, содержащему другие HTML-теги, заключается в создании своего рода блока-контейнера. Таким способом вы можете создавать боковые панели, 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.

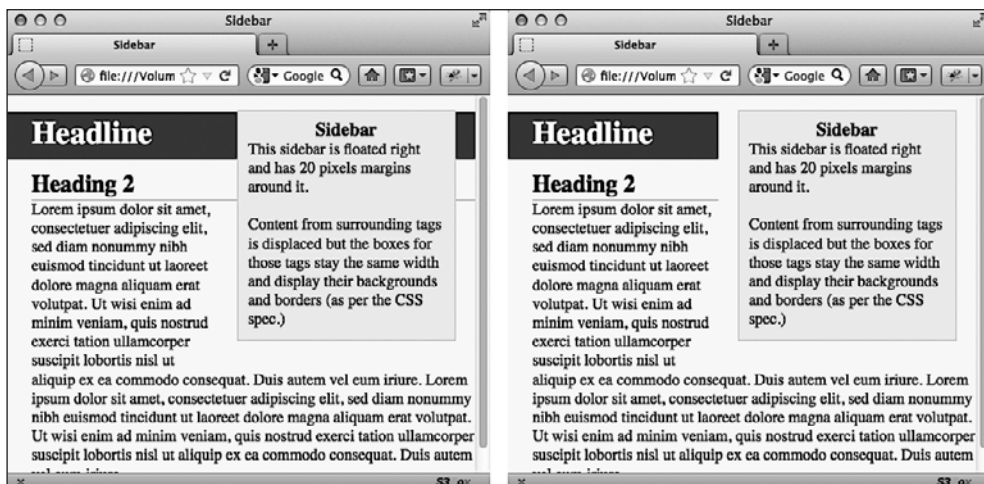
Допустим, вы создали веб-страницу, в которой имеется тег заголовка `<h1>`, а за ним идет тег абзаца `<p>`. Внутри `<p>` вы вставили фотографию с помощью тега `<img>`. Теперь, если вы установите выравнивание фотографии по правому краю, заголовок `<h1>` и часть содержимого абзаца `<p>` будут по-прежнему 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 боковая панель, примыкающая к правому краю веб-страницы. Содержимое страницы, расположенное под ним, как ему и положено, приподнимается выше и обтекает панель.

Однако, если у этого содержимого есть фоновые параметры или границы, они отображаются, фактически проступая под боковой панелью (рис. 7.16, *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. 7.16, *справа*). С помощью CSS вы можете сделать и это.



**Рис. 7.16.** 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 выше) отменяет отображение продолжающихся фона или границ под плавающими элементами.

#### ПРИМЕЧАНИЕ

Этот метод работает не во всех браузерах. Смотрите врезку далее.

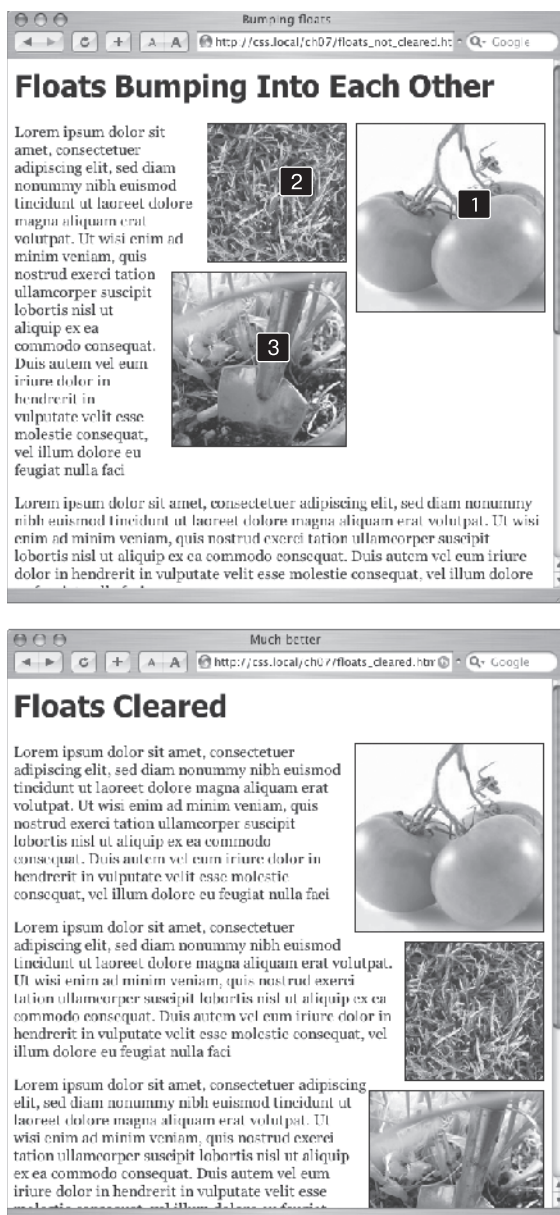
Другой подход состоит в добавлении границ вокруг плавающего элемента. Если вы создаете достаточно толстую границу цвета фона страницы, то не будете даже догадываться о существовании лишних границ, находящихся под плавающим элементом, так как они располагаются поверх и скрывают аналогичные лишние атрибуты.

## 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 *высокая* боковая панель, примыкающая 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 боковой панелью. Вам же нужно, чтобы для примечания с авторскими правами было отменено обтекание, установленное свойством плавающей боковой панели.

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. 7.17, *вверху*). В данном случае плавающие

элементы не должны отображаться рядом друг с другом. Для решения этой проблемы в CSS предусмотрено свойство `clear`.



**Рис. 7.17.** 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 элемент



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 стиливой класс, который выполняет это:

```
.copyright {  
  clear: both;  
}
```

На рис. 7.17 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. 7.17, *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. 13, там вы познакомитесь с использованием обтекания в более сложных вариантах.

## Обучающий урок: поля, фоновые параметры, границы

В этом обучающем уроке мы исследуем элементы блочной модели CSS, потренируемся в настройке параметров интервалов, промежутков объектов веб-страниц,

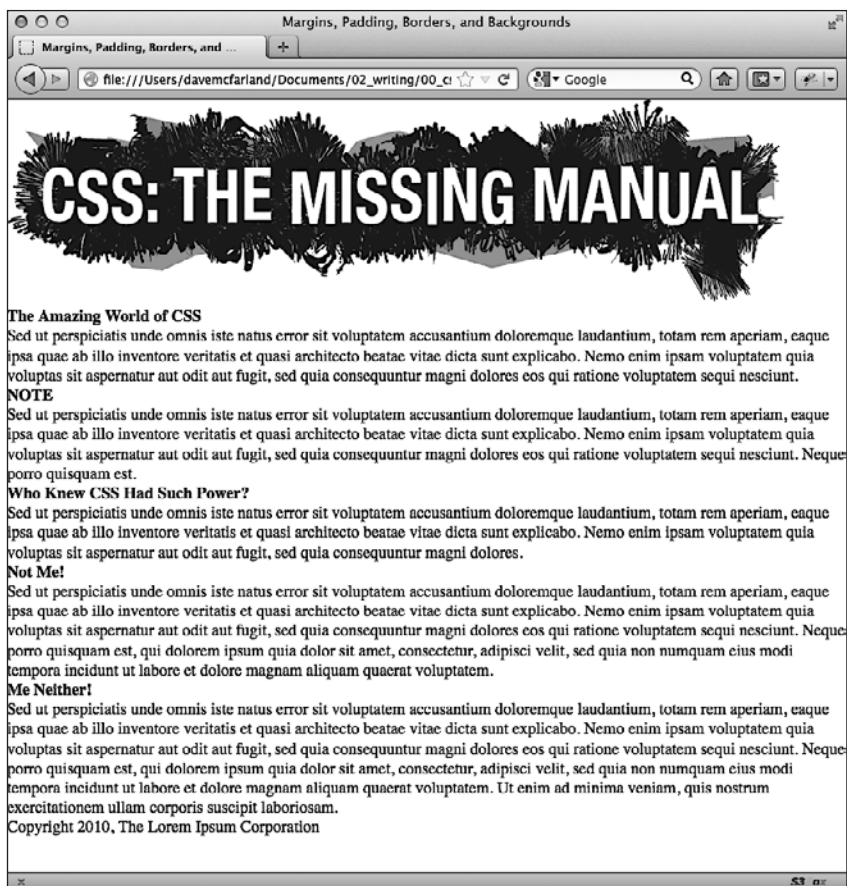


применим к элементам красочные границы-рамки, поуправляем размерами и обтеканием элементов веб-страниц.

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. 7.18).



**Рис. 7.18.** 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 набор стилей, который рассматривался в разделе «Начинаем с чистого листа» гл. 5. Этот набор стилей в основном удаляет все поля, отступы, установленные значения для размера

шрифта, начертания шрифта из наиболее значимых элементов уровня блока и устраняет множество кросс-браузерных проблем с отображением элементов, с которыми вы могли столкнуться из-за этих свойств.

Как минимум вы должны всегда включать этот набор стилей во все создаваемые таблицы стилей. Наверное, наиболее важными свойствами в нем являются `margin` и `padding`, установленные в первом стиле. Существует достаточно кросс-браузерных 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: rgb(253,248,171);
}
```

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>`.

---

#### ПРИМЕЧАНИЕ

Возможно, вы привыкли вместо RGB-модели задания цвета использовать шестнадцатеричную модель (в которой цвет представляется, например, так: `#FDF8AB`). Чтобы осуществлять преобразования между этими двумя моделями, можно воспользоваться онлайн-конвертером [www.colorhexa.com/](http://www.colorhexa.com/). Но использование модели RGB предпочтительнее, поскольку RGBA-формат этой модели, имеющий дополнительный параметр прозрачности, настолько полезен, что лучше остановиться на одной цветовой модели (RGB), а не смешивать две (RGB и шестнадцатеричную).

---

3. Добавьте другой стиль во внутреннюю таблицу стилей.

```
body {
  background-color: rgb(255,255,255);
  border: 3px solid rgb(75,75,75);
}
```

Этот стиль добавляет белый цвет фона для тега `<body>` и темно-серую трехпиксельную границу. Поскольку тег `<body>` расположен внутри тега `<html>`, браузер считает, что он находится «поверх» тега `<html>` и белый фон покрывает 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 2, затем удалите стиль для тега `<html>` и просмотрите страницу снова. Странная, но полезная особенность CSS.

---

4. Измените стиль тела (`<body>`), который вы только что создали, добавив пять новых свойств (изменения выделены полужирным шрифтом):

```
body {
  background-color: rgb(255,255,255);
```

```
border: 3px solid rgb(75,75,75);
width: 760px;
margin-top: 20px;
margin-left: auto;
margin-right: auto;
padding: 15px;
}
```

Свойство `width` ограничивает тело страницы 760 пикселями по ширине: если окно браузера посетителя окажется шире чем 760 пикселей, он увидит фоновый цвет стиля `<html>` и блок шириной 760 пикселей с белым фоном тега `<body>`.

Свойство `margin-top` добавляет 20 пикселей 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: «Разбейся в этом сам», и поскольку данное значение применяется как к левому, так и к правому полю, 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, чтобы предотвратить прикосновение содержимого тега `<body>` к линии границы, отступ шириной 15 пикселей добавлен к внутренней части содержимого с помощью свойства `padding`. Другими словами, для изображения и текста был задан отступ 15 пикселей от всех четырех краев. Затем мы добавим свечение вокруг блока, воспользовавшись свойством `box-shadow`.

5. Отредактируйте только что созданный стиль элемента `body`, добавив к нему еще одно свойство после `border`, но перед `width` (изменения выделены полужирным шрифтом):

```
body {
background-color: rgb(255,255,255);
border: 3px solid rgb(75,75,75);
box-shadow: 0 0 15px 5px rgba(44,82,100,.75);
width: 760px;
margin-top: 20px;
margin-left: auto;
margin-right: auto;
padding: 15px;
}
```

Этот стиль добавляет блоку свечение путем создания 15-пиксельной тени помещенной непосредственно за блоком (то, что параметры тени начинаются с `0 0`, означает, что она не имеет смещения вправо-влево или вверх-вниз, и представляет собой просто фон). Значение `5px` определяет расширение тени, выводя ее на 5 пикселей дальше всех четырех углов. И наконец, значение `rgba` устанавливает темно-синий цвет с показателем непрозрачности 75 % (то есть через него можно видеть желтый фон).

В вашей таблице стилей уже много всего, поэтому пришло время проверить, как выглядит страница.

- Сохраните файл и просмотрите веб-страницу в браузере.

Вы должны увидеть белый блок с изображением, кусок текста и серый контур с синеватым свечением, плавающие на желтом фоне (рис. 7.19). Теперь следует уделить внимание тексту, чем мы и займемся далее.

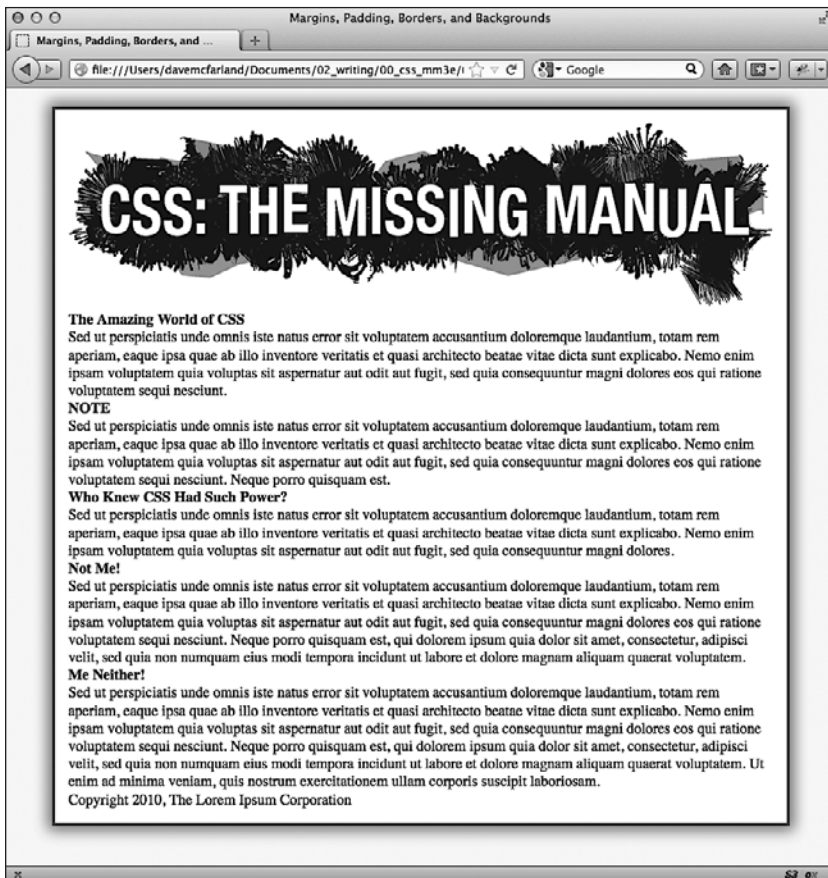


Рис. 7.19. Устанавливая значение auto для левого и правого полей любого элемента, имеющего заданную ширину, вы центрируете этот элемент

## 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>` вверху страницы.

- Вернитесь к файлу `sidebar.html` в HTML-редакторе. Установите курсор после закрывающей скобки селектора `body`, нажмите `Enter` и добавьте следующий стиль:

```
h1 {
  font-size: 2.75em;
```

```
font-family: Georgia, "Times New Roman", Times, serif;
font-weight: normal;
text-align: center;
letter-spacing: 1px;
color: rgb(133,161,16);
text-transform: uppercase;
}
```

Этот стиль использует множество свойств для форматирования текста, которые мы обсуждали в предыдущей главе. Верхний заголовок имеет высоту 2.75 em (44 пиксела в большинстве браузеров) и набран прописными буквами. К нему применяется шрифт Georgia зеленого цвета, есть небольшой промежуток между буквами. Свойство `text-align` обеспечивает центровку текста в середине блока. Очень интересным получается добавление фонового цвета для выделения заголовка.

---

#### СОВЕТ

Сохраняйте страницу и просматривайте ее в браузере после каждого шага из этого примера. Таким образом вы получите хорошее понимание того, как приведенные здесь свойства CSS воздействуют на элементы, которые они форматировуют.

---

2. Добавьте одно новое свойство к стилевому тегу `h1`, чтобы он выглядел следующим образом (изменения выделены полужирным шрифтом):

```
h1 {
font-size: 2.75em;
font-family: Georgia, "Times New Roman", Times, serif;
font-weight: normal;

text-align: center;
letter-spacing: 1px;
color: rgb(133,161,16);
text-transform: uppercase;
background-color: rgb(226,235,180);
}
```

Если вы посмотрите 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: 2.75em;
font-family: Georgia, "Times New Roman", Times, serif;
font-weight: normal;
text-align: center;
letter-spacing: 1px;
background-color: rgb(226,235,180);
}
```

```

color: rgb(133,161,16);
text-transform: uppercase;
background-color: rgb(226,235,180);
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 пикселей добавляется над текстом, 15 пикселей — справа, 2 пикселя — внизу и 15 пикселей — слева.

Есть еще одна проблема с заголовком: из-за отступов, которые добавлены к тегу `<body>` (см. шаг 4 предыдущего задания), заголовок отодвинут на 15 пикселей 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 одно последнее свойство к стилю тега `h1`, чтобы он выглядел следующим образом (изменения выделены полужирным шрифтом):

```

h1 {
font-size: 2.75em;
font-family: Georgia, "Times New Roman", Times, serif;
font-weight: normal;
text-align: center;
letter-spacing: 1px;
color: rgb(133,161,16);
text-transform: uppercase;
background-color: rgb(226,235,180);
padding: 5px 15px 2px 15px;
margin: 0 -15px 20px -15px;
}

```

Здесь сокращенное свойство `margin` устанавливает 0 пикселей для верхнего поля, минус 15 пикселей для правого, 20 пикселей для нижнего и минус 15 пикселей для левого. Нижнее поле просто добавляет немного пространства между заголовком и абзацем, который идет 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 пикселей влево и 15 пикселей вправо, расширяясь и вытягиваясь над отступами тега `<body>`.

5. Теперь немного отформатируем тег `<h2>`. Добавьте следующий стиль после стиля для `h1`:

```

h2 {
font-size: 1.5em;
font-family: "Arial Narrow", Arial, Helvetica, sans-serif;
color: rgb(249,107,24);
border-top: 2px dotted rgb(141,165,22);
border-bottom: 2px dotted rgb(141,165,22);
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 пикселей над заголовком и 5 пикселей под ним.

6. Сохраните файл и просмотрите страницу в браузере.

Заголовки выглядят хорошо (рис. 7.20). Далее создадим боковую панель на правой стороне страницы.

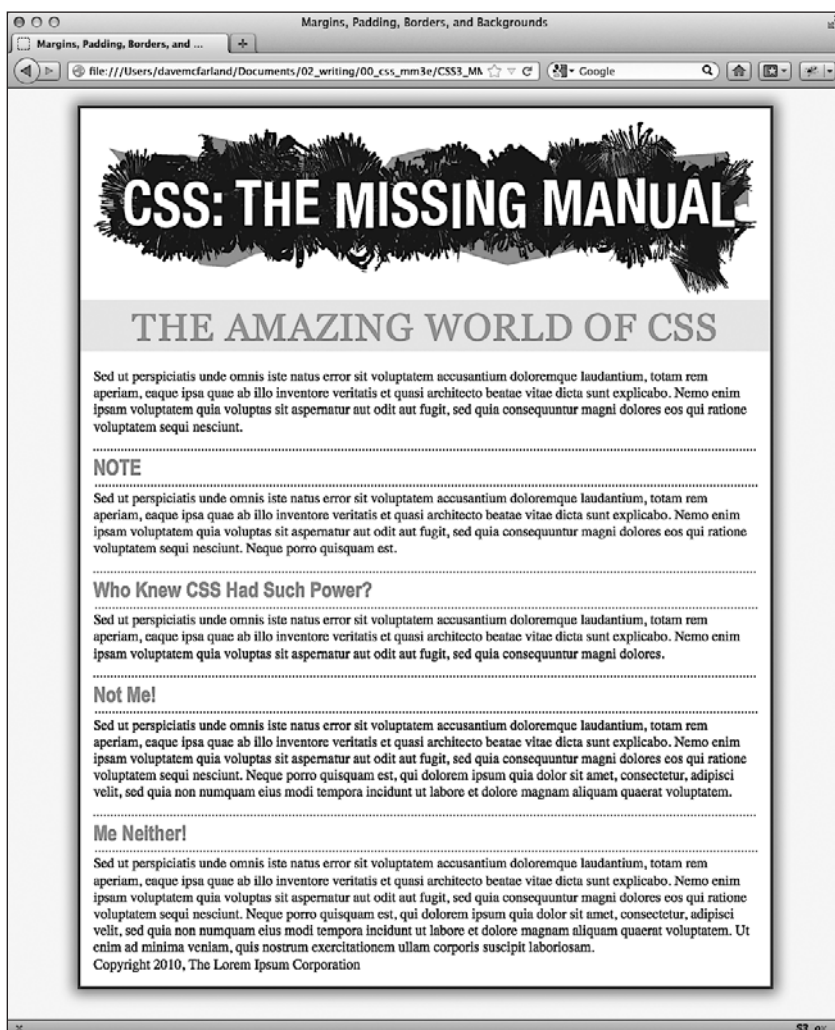


Рис. 7.20. С помощью нескольких стилей вы можете изменить фоновый цвет веб-страницы, добавить поля, регулировать интервалы между заголовками и абзацами



## Создание боковой панели

Боковые панели — обычные элементы, применяемые в большинстве печатных изданий: в журналах, книгах, газетах. Они отделяют и подсвечивают небольшие блоки информации, например перечень ресурсов, тематичный анекдот. Но, чтобы боковые панели были достаточно эффективными и полезными, они не должны прерывать поток основного содержимого. Даже название «боковая панель» 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 боковую панель. Для этого прекрасно подходит тег `<div>`. С его помощью можно заключить любой объем HTML-кода в свой собственный отдельный блок.

2. Прокрутите страницу вниз к коду HTML и щелкните кнопкой мыши перед первым тегом `<h2>` (с заголовком NOTE). Наберите `<div class="sidebar">` и нажмите **Enter**.

Этот HTML-код отмечает начало блока боковой панели и применяет к нему стилевой класс. Мы создадим сам стилевой класс `.sidebar`, но сначала нужно определить завершение блока боковой панели, 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. Прокрутите страницу вверх к таблице стилей и добавьте 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 пиксели, 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 пикселей пространства вокруг панели.

Если вы просмотрите веб-страницу в браузере, то увидите, что форма и положение блока боковой панели определены, но есть одна проблема: границы тегов `<h2>` отображаются под самим блоком. Несмотря на то что плавающая боковая панель смещает текст заголовков, границы остаются на том же месте. Они выступают в качестве фона плавающей боковой панели. 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: rgb(250,235,199);
  padding: 10px 20px;
}
```

Это свойство добавляет светло-оранжевый цвет к боковой панели и оттесняет текст от границ боковой панели, чтобы он не касался границ, которые вы собираетесь добавить.

6. 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: rgb(250,235,199);
  padding: 10px 20px;
  border: 1px dotted rgb(252,101,18);
  border-top: 20px solid rgb(252,101,18);
}
```

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 пикселей. Этот способ позволяет использовать всего две строки кода CSS, а не четыре (`border-top`, `border-bottom`, `border-left` и `border-right`).

Затем мы добавим скругленные углы и тени, чтобы выделить эту боковую панель по-настоящему.

7. Добавьте еще два свойства к стилю `.sidebar`, придав ему следующий вид (изменения выделены полужирным шрифтом):

```
.sidebar {
  width: 30%;
  float: right;
  margin: 10px;
  background-color: rgb(250,235,199);
  padding: 10px 20px;
  border: 1px dotted rgb(252,101,18);
}
```

```
border-top: 20px solid rgb(252,101,18);
border-radius: 10px;
box-shadow: 5px 5px 10px rgba(0,0,0,.5);
}
```

Свойство `border-radius` позволяет создать скругленные углы. В данном случае 10-пиксельная установка предоставляет заметное скругление. Свойство `box-shadow` добавляет тень, отбрасываемую вниз и вправо от блока, придавая ему вид парящего над страницей. Теперь мы близки к завершению работы.

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 мы создали в шаге 4). Границы отвлекают внимание, а верхнее поле слишком отталкивает заголовок вниз от верхней части боковой панели. К счастью, вы можете использовать селектор потомка, чтобы переопределить эти свойства.

8. После стиля `.sidebar` во внутренней таблице стилей добавьте селектор потомка:

```
.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.

9. Найдите стиль `h2` и добавьте свойство `overflow`:

```
h2 {
font-size: 1.5em;
font-family: "Arial Narrow", Arial, Helvetica, sans-serif;
color: rgb(249,107,24);
border-top: 2px dotted rgb(141,165,22);
border-bottom: 2px dotted rgb(141,165,22);
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.

10. Сохраните файл и просмотрите веб-страницу в браузере. Она должна иметь вид, представленный на рис. 7.21.

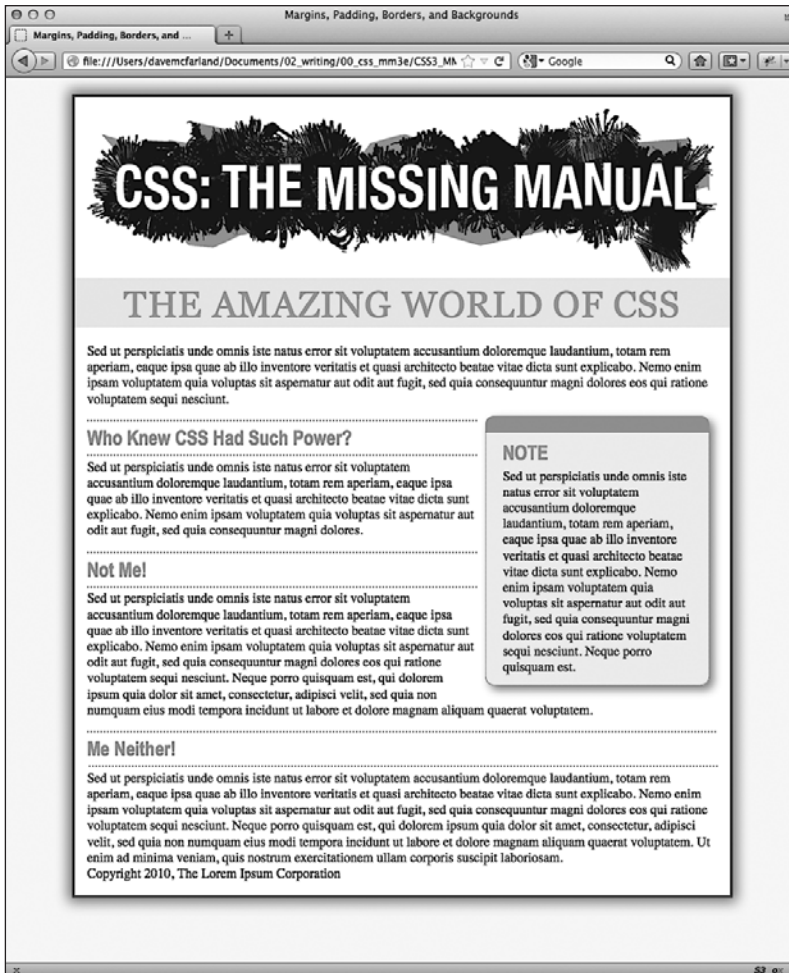


Рис. 7.21. Небольшой набор CSS-стилей придает непривлекательному HTML-коду элегантный вид

## Двигаемся дальше

Чтобы проверить новые знания и навыки, попробуйте выполнить следующее практическое задание самостоятельно. Создайте для тега `<p>` стиль, который смог бы украсить 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 разделе «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 подробно рассмотрены некоторые оригинальные методы применения изображений средствами языка CSS. На самом деле лучший способ узнать о возможностях графики и научиться ее использовать — 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 на 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 применяемых в сочетании с изображениями.

- `Border` — можно использовать одно из множества свойств границ (см. раздел «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 боковой панели, — 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 в одном фрагменте веб-страницы. Если у вас есть галерея фотографий, можно заключить все внутри тега `<div>` с классом `gallery` и затем создать стиль только для изображений, расположенных внутри этого блока, так: `.gallery img`.

## Добавление фоновых изображений

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 применял атрибут `background` тега `<body>`. CSS делает все то же самое, но гораздо лучше.

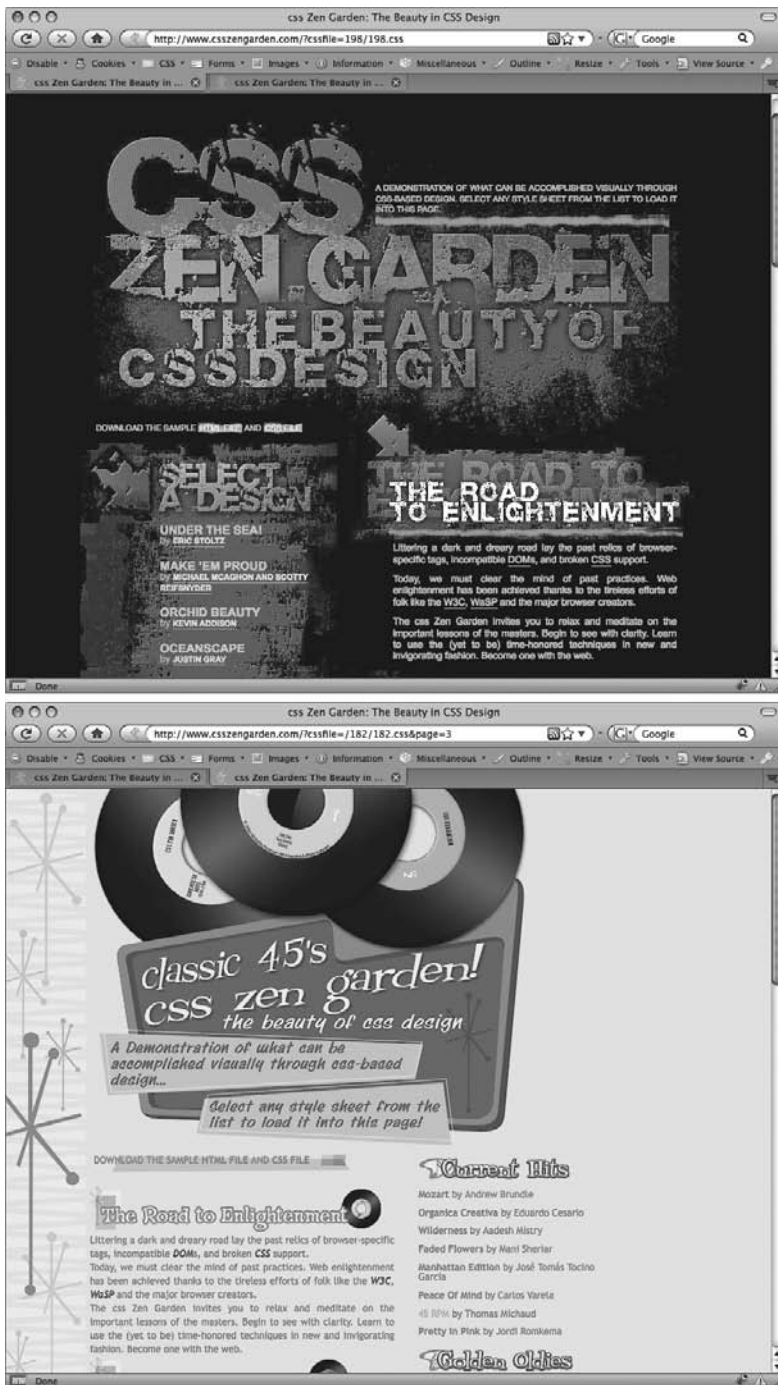


Рис. 8.1. Сайт csszengarden.com демонстрирует мощьность каскадных таблиц стилей, показывая, как с помощью CSS вы можете превратить один и тот же HTML-файл в две совершенно разные веб-страницы

## 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 уменьшает 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 видимым 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, которое 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. То есть лучше использовать PNG8, а не 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 (также называемой *альфа-прозрачностью*). 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.



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);
}
```

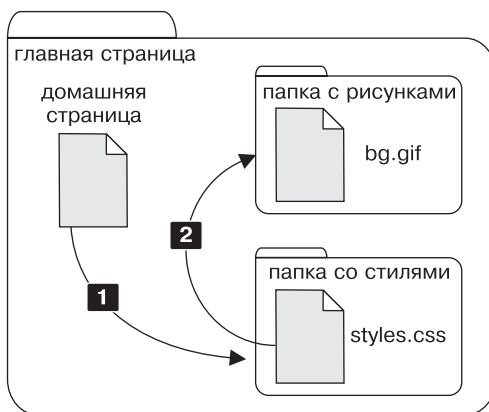
Свойство принимает единственное значение: ключевое слово `url`, за которым следует путь к файлу рисунка, заключенный в круглые скобки. Вы можете использовать абсолютный URL, например, так:

```
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 в следующей врезке — «Информация для новичков. Типы URL», — 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 имеется 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 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 следующих 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 этот способ трудно применять, 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` (Файл ▶ Открыть),

## ИНФОРМАЦИЯ ДЛЯ НОВИЧКОВ

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

Если свойство `background-image` используется само по себе, фоновый рисунок многократно повторяется в виде мозаики, заполняя всю веб-страницу (в современном стандарте HTML эта недоработка исключена). К счастью, можно применить свойство `background-repeat`, чтобы 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 и сверху вниз, до полного заполнения всего пространства веб-страницы (рис. 8.3).



**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

- `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).
- `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, той же боковой панели).



**Рис. 8.4.** Пользуясь возможностью управления повтором фоновых изображений, можно создать графический фон для заголовков (*слева*) и боковых панелей (*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

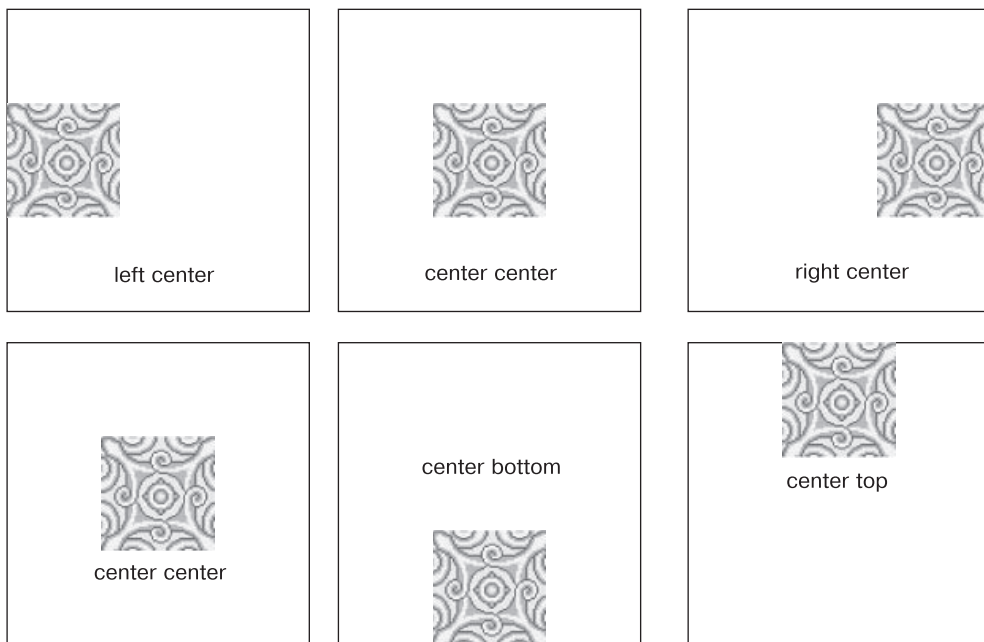
Можно работать с двумя наборами ключевых слов: один из них имеет три параметра управления горизонтальным позиционированием: `left`, `center`, `right`, а другой — три параметра вертикального позиционирования: `top`, `center`, `bottom` (рис. 8.5).

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

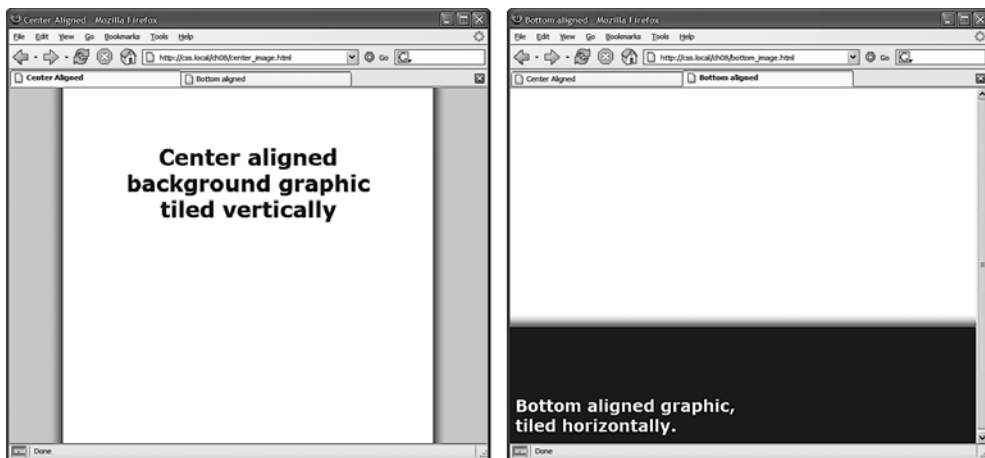
Если вы решили применить повторяющееся фоновое изображение (установив для свойства `background-repeat` одно из значений, описанных ранее), то начальную точку или координату первого отображаемого фонового рисунка определит свойство `background-position`. Например, если вы примените параметр `repeat`, 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 выполнить следующий стиль:

```
body {
  background-image: url(background.jpg);
  background-repeat: repeat-y;
  background-position: center top;
}
```



**Рис. 8.6.** При повторении фонового изображения вертикально (*слева*) или горизонтально (*справа*) пользуйтесь свойством `background-position`

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*).

На рис. 8.6, *слева*, фоновое изображение представляет собой широкий белый блок с тенью 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>` сверху, 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 Internet Explorer 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 изображе-

ния. Например, если вы используете вертикальную позицию `bottom`, изображение не всегда появляется в нижней части окна браузера. Это случается,

## ОШИБКИ БРАУЗЕРОВ

когда основное содержимое веб-страницы меньше по высоте.

Если веб-страница имеет всего несколько абзацев текста и отображается на большом мониторе, то

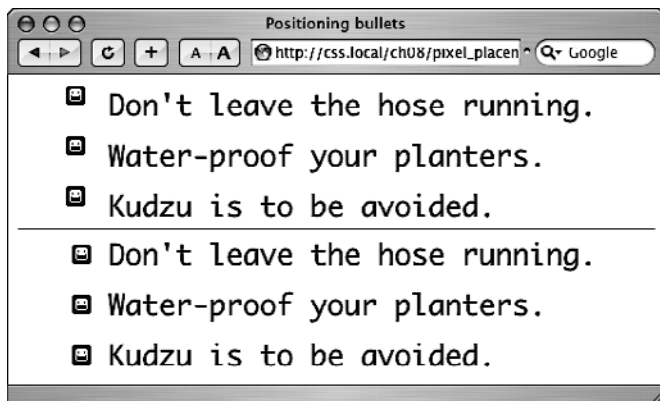
Firefox принимает за нижний край окна браузера `bottom` нижний край последнего абзаца текста веб-страницы. Если вы столкнетесь с такой ситуацией, то просто добавьте следующий стиль: `html { height: 100 %; }`.

## 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 пикселей правее и на 8 пикселей 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 применять 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. Так можно добиться интересного эффекта. Например, можно позиционировать элемент на расстояние, пропорциональное его ширине. Допустим, когда вам нужно переместить рисунок на расстояние три четверти размера элемента от заголовка, но вы не знаете ширину.

Как и в случае с пикселями 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  $y$  задается 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).

---

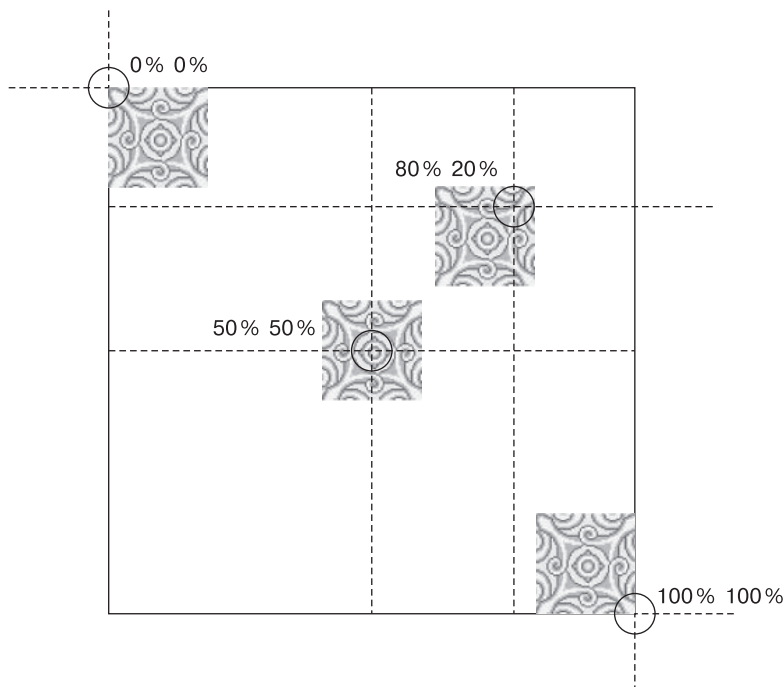
### ПРИМЕЧАНИЕ

Размещение изображения по вертикали в фоне страницы с помощью процентов не обязательно поместит его в правильное место, если содержимое страницы не заполняет всю высоту окна браузера.

---

Как и в случае с пикселями и `em`, можно назначать отрицательные процентные значения, но результаты трудно предугадать. Вы можете также комбинировать и подбирать значения в пикселях, `em` и процентах одновременно. Например, чтобы поместить изображение на расстоянии 5 пикселей от левого края элемента, но в центре элемента по высоте, применяют свойство со следующими параметрами:

```
background-position: 5px 50%;
```



**Рис. 8.8.** Используя проценты, сначала определите начальную «якорную» точку (координату) в изображении

#### 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>`. Например, для вставки таких важных фрагментов, как логотип сайта или схема проезда к интернет-магазину.

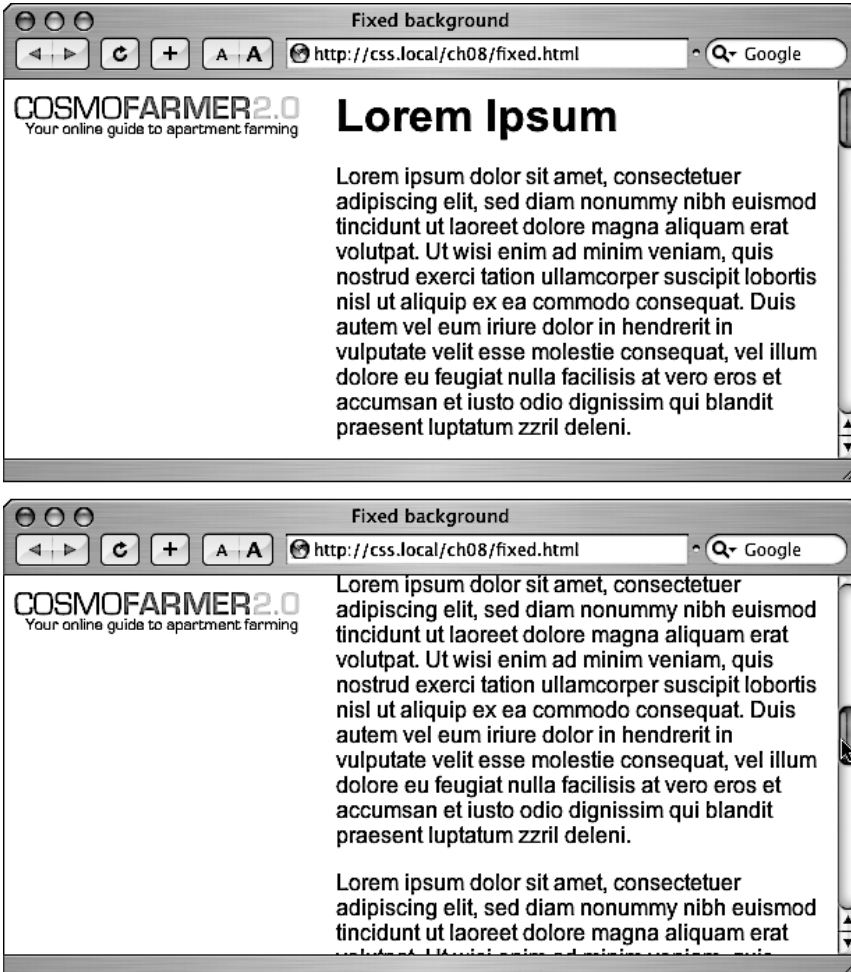
## Фиксация изображения на месте

Когда посетитель прокручивает содержимое веб-страницы так, чтобы увидеть ее остальную 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;
}
```



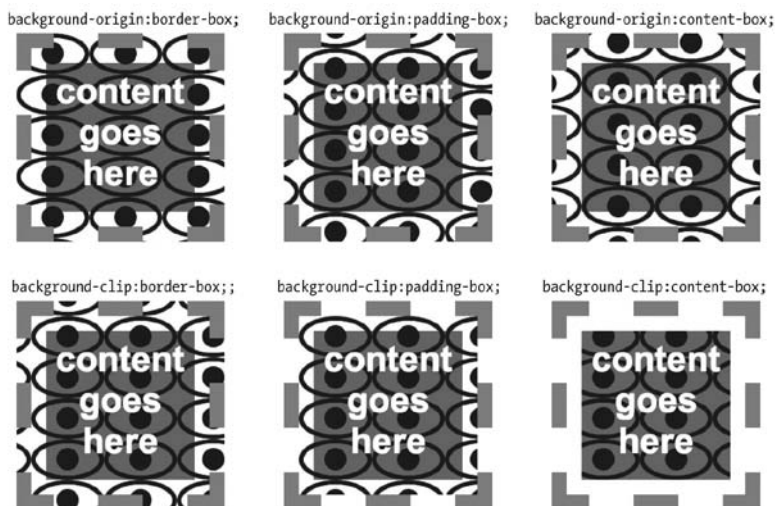
**Рис. 8.9.** Для закрепления логотипа используйте свойство `background-attachment` с параметром `fixed`

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.

## Определение начальной позиции фонового изображения и порядка его отсечения

В CSS3 добавлена возможность, которая позволяет сообщить браузеру, где фоновое изображение должно начаться по отношению к рамке, отступу и содержимому элемента. Например, когда изображение используется для повторяющегося заполнения, оно начинается в верхнем левом углу отступа элемента (верхнее среднее изображение на рис. 8.10). Но начальную позицию изображения можно изменить, воспользовавшись свойством `background-origin`. Этому свойству задается одно из трех значений:

- `border-box` — изображение помещается в верхний левый угол области, которая выделена для рамки (верхнее изображение слева на рис. 8.10);
- `padding-box` — изображение помещается в верхний левый угол области, которая отведена под отступы (верхнее изображение посередине на рис. 8.10). Это обычное место, куда браузер помещает фоновое изображение;
- `content-box` — изображение помещается в верхний правый угол той области, которая отведена под содержимое (верхнее изображение справа на рис. 8.10).



**Рис. 8.10.** Применение свойств `background-origin` и `background-clip` к фоновому цвету

Разумеется, эти настройки не имеют смысла, когда вокруг элемента нет ни отступа, ни рамки. Кроме того, эффект может быть практически незаметен, особенно если у заполняющего повторяющегося изображения не видно стыков.

### ПРИМЕЧАНИЕ

Свойства `background-origin` и `background-clip` в Internet Explorer 8 и более ранних версиях не работают. Поскольку Internet Explorer все еще пользуется популярностью, применяйте это свойство с оглядкой.

Однако данный эффект может быть весьма заметен, если изображение не повторяется. Например, если вы выберете настройку `no-repeat` и фоновое изображение появится только один раз, его можно поместить в верхний левый угол области (внутри любых отступов):

```
background-image: url(logo.png);
background-repeat: no-repeat;
background-origin: content-box;
```

Кроме того, свойство `background-origin` может эффективно использоваться с другим новым свойством — `background-clip`. Свойство `background-clip` ограничивает область появления фонового изображения. Обычно фоновые изображения заполняют всю область элемента, включая пространство за рамкой и отступами (нижнее левое изображение на рис. 8.10). Но область появления изображения можно указать с помощью трех значений.

- `border-box` — позволяет изображению появляться позади содержимого полностью за любыми границами. Это можно заметить только при наличии границ с видимыми разрывами, например при использовании пунктирной линии (нижнее левое изображение на рис. 8.10). Именно это и делают браузеры, поэтому применять эту настройку нет никакого смысла.
- `padding-box` — дает возможность ограничить любое фоновое изображение областью отступов и содержимого элемента (нижнее среднее изображение на рис. 8.10). Эта настройка пригодится при наличии границы, составленной из пунктиров или точек и нежелании видеть изображения в разрывах границы.
- `content-box` — позволяет ограничить фоновое изображение областью содержимого элемента (нижнее правое изображение на рис. 8.10).

Сочетая свойства `background-origin` и `background-clip`, можно заставить изображение начинаться в верхнем левом углу области содержимого и появляться только за содержимым:

```
background-origin: content-box;
background-clip: content-box;
```

## Масштабирование фоновых изображений

В отличие от HTML-тега `<img>`, размеры фонового изображения можно изменить. Тег размещает фоновое изображение с тем размером, который был задан при его создании. А в CSS3 добавлено свойство `background-size`, позволяющее управлять размером фонового изображения. Чтобы установить размер, можно выбирать либо значения, либо ключевые слова.

- Чтобы установить размер изображения, нужно задать его высоту и ширину. Для этого можно воспользоваться абсолютными значениями, выраженными в пикселах:

```
background-size: 100px 200px;
```

Этот код устанавливает для фонового изображения ширину 100 пикселей и высоту 200 пикселей. Можно также указать только значения ширины или высоты, задав для оставшегося значения режим определения `auto`:

```
background-size: 100px auto;
```

В таком случае фоновое изображения будет иметь ширину 100 пикселей, а браузер автоматически установит его высоту, сохраняя пропорции изображения (чтобы не возникло искажений). Можно также использовать процентные значения. Если нужно масштабировать изображение для точного попадания в фон, можно для обеих настроек использовать значение 100 % (изображение слева на рис. 8.11):

```
background-size: 100% 100%;
```

- Ключевое слово `contain` приводит к изменению размеров изображения. Оно заставляет его поместиться в область, сообразуясь с соотношением сторон (посередине изображение на рис. 8.11). В зависимости от формы изображения и элемента изображение растягивается, чтобы поместиться либо по ширине, либо по высоте элемента.

```
background-size: contain;
```

- Ключевое слово `cover` заставляет ширину изображения соответствовать ширине элемента, а высоту изображения соответствовать высоте элемента. Обычно это приводит к искажению изображения путем либо сжатия, либо растяжения с тем, чтобы оно поместилось в размеры элемента (изображение справа на рис. 8.11).

```
background-size: cover;
```

Использование свойства `background-size` практически всегда приводит к изменению размеров исходного изображения: если изображение меньше элементов, браузер увеличивает его масштаб, что зачастую выражается заметным дублированием пикселей и ухудшением качества изображения (что видно по фоновым изображениям на рис. 8.11).



Рис. 8.11. Использование свойства `background-size`

Свойство `background-size` является единственным способом изменения размеров фонового изображения. Это свойство понимает большинство браузеров, но следует иметь в виду, что такой все еще широко используемый веб-браузер, как Internet Explorer 8, его не понимает, поэтому используйте его с оглядкой (или без нее, если Internet Explorer 8 вас не волнует).

#### СОВЕТ

Свойство `background-size` может оказаться особенно удобным при работе с элементами, размер которых определяется с применением процентных показателей, допустим при использовании восприимчивых конструкций, рассматриваемых в гл. 14. Например, если поместить изображение в фон

баннера, занимающего 960 пикселей при просмотре на мониторе настольного компьютера, но сжать его до 480 пикселей при просмотре на телефоне, можно поместить в баннер большое изображение и воспользоваться следующей настройкой:

```
background-size: 100% auto;
```

Эта настройка заставит веб-браузер изменить размер изображения таким образом, чтобы оно поместилось в сжатый баннер.

---

## Сокращенный вариант свойства background

Как видно 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-image`, `background-position`, `background-repeat`, `background-attachment` и `background-color`. Следующий стиль устанавливает фон белого цвета с едва заметным неповторяющимся отпечатком, закрепленным по центру веб-страницы:

```
body {  
  background: url(bullseye.gif) center center no-repeat fixed #FFF;  
}
```

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 задали только само фоновое изображение:

```
background: url(image/bullseye.gif)
```

Это эквивалентно следующему стилю:

```
background: url(image/bullseye.gif) scroll left top repeat transparent;
```

Поведение, когда при отсутствии определения значения происходит возвращение к значениям, по умолчанию может привести к весьма неожиданным результатам. Предположим, например, что к стилю добавляются два объявления:

```
background-color: yellow;  
background: url(image/bullseye.gif) no-repeat;
```

Возможно, вы ожидали увидеть изображение бычьего глаза на желтом фоне. Но вы его не увидите, потому что при встрече свойства `background` без указания цвета, браузер переключает значение `background-color` на `transparent` (невидимый). Чтобы выйти из этой сложной ситуации нужно указать свойство `background-color` вторым:



```
background: url(image/bullseye.gif) no-repeat;  
background-color: yellow;
```

Кроме того, когда к одному и тому же элементу применяется сразу несколько свойств, можно в конечном итоге случайно уничтожить фоновые изображения. Предположим, например, что нужно добавить фоновое изображение к каждому абзацу страницы, для чего создается следующий стиль:

```
p {  
  background: url(icon.png) left top no-repeat rgb(0,30,0);  
}
```

Затем принимается решение, что после заголовка второго уровня каждый первый абзац должен иметь синий фон, и создается следующий стиль:

```
h2 + p {  
  background: blue;  
}
```

В этом втором стиле используется сокращенная запись, сбрасывающая все остальные свойства фона к их значениям по умолчанию. Что касается изображения, то по умолчанию оно вообще не используется, поэтому вместо простого добавления к абзацу синего фона с оставлением изображения на месте, этот стиль вообще удаляет изображение!

То есть сокращенная запись свойства `background` может сэкономить время на написании кода, но также может и принести проблемы, о чем не нужно забывать.

#### ПРИМЕЧАНИЕ

В сокращенной записи свойств фона можно использовать и самые новые свойства, введенные в CSS3, — `background-size`, `background-origin` и `background-clip`, но стоит ли это делать? Во-первых, поскольку Internet Explorer 8 не понимает эти свойства, их добавление к объявлению фона приведет к тому, что Internet Explorer 8 пропустит другие, понимаемые им свойства (то есть он проигнорирует все ваши фоновые настройки). Кроме того, свойства нужно перечислять в определенном порядке и нет такого браузера (на момент написания книги), который мог бы в одном объявлении обработать вместе значения `background-position` и `background-size`. Чтобы решить эту проблему, можно создать объявление для общепринятых свойств в форме сокращенной записи — изображения, позиции, повторения, размещения и цвета, а затем после сокращенной формы объявления добавить отдельные объявления для новых свойств CSS3.

### 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 мно-

жество платных сайтов, содержащих коллекции готовых фотографий и иллюстраций, но наряду с ними также есть довольно много бесплатного материала. Образцы замечательных фотографий можно найти на сайте Morgue File ([www.morguefile.com](http://www.morguefile.com)). Stock.xchng ([www.sxc.hu](http://www.sxc.hu)) — еще один отличный фотопесчур. Сайт Open Photo (<http://openphoto.net/gallery/browse.html>) предлагает коллекции

## ЧАСТО ЗАДАВАЕМЫЕ ВОПРОСЫ

изображений, предоставляя возможность их применения с некоторыми ограничениями. 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](http://www.flickr.com/creativecommons)) и Picasa Web Albums (<http://picasaweb.google.com>). За фотографии на бесплатных сайтах не нужно платить, но не все они могут использоваться в коммерческих проектах. Обязательно 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 в вашем распоряжении также много сайтов. Сайт под названием Some Random Dude бесплатно предлагает набор из 121 значка: [www.somerandomdude.net/srd-projects/sanscons](http://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](http://www.colourlovers.com/patterns)), Pattern4u ([www.kollermedia.at/pattern4u](http://www.kollermedia.at/pattern4u)) или Squidfingers (<http://squidfingers.com/patterns>). Можете также создать собственный черепичный фон с помощью следующих инструментов: BgPatterns (<http://bgpatterns.com>), Stripe Generator 2.0 ([www.stripegenerator.com](http://www.stripegenerator.com)) или PatternCooler ([www.patterncooler.com](http://www.patterncooler.com)).

## Использование нескольких фоновых изображений

Хотя вполне достаточно и одного фоновое изображения, имеется возможность выкладки слоями сразу нескольких изображений. Предположим, например, что нужно добавить фоновое изображение к боковой панели, чтобы придать ей видимость свитка (рис. 8.12). Если просто поместить в качестве фона одно изображение, то сначала оно может работать (верхнее изображение слева на рис. 8.12), но если добавить к боковой панели слишком много текста, внешний вид ухудшится (верхнее изображение справа на рис. 8.12). Причина в том, что у изображения один размер и оно не будет становиться больше или меньше, чтобы подстроиться под размер боковой панели.

К счастью, CSS3 допускает добавление к фону элемента нескольких изображений. В случае со свитком можно использовать три фоновых изображения: одно для верхней части свитка, другое для нижней его части и третье для его текстовой области. Это последнее изображение представляет собой бесшовную плитку, поэтому при увеличении высоты боковой панели изображение просто заполняет новое пустое пространство плитками.

Разумеется, несколько изображений можно применять и для решения менее сложных задач. Предположим, например, что к фону элемента нужно добавить красочное сильно текстурированное изображение, а также к нему нужно добавить двухцветный логотип. Как сказано во врезке «Информация для новичков. GIF-, JPEG- и PNG-файлы: веб-графика» в разделе «Добавление фоновых изображений» данной главы, для многоцветного изображения больше подходит формат JPEG, а для областей, содержащих чистые цвета, к которым можно отнести логотип, больше подходит формат PNG8. В данном случае можно объединить преимущества, используя JPEG для многоцветного фона и PNG8 для логотипа.



**Рис. 8.12.** Использование нескольких фоновых изображений

Для создания нескольких фоновых изображений нужно просто воспользоваться свойством `background-image` со списком изображений с запятой в качестве разделителя:

```
background-image: url(scrollTop.jpg),
                  url(scrollBottom.jpg),
                  url(scrollMiddle.jpg);
```

---

**ПРИМЕЧАНИЕ**

В данном примере кода каждый URL-адрес находится на отдельной строке, но это не является обязательным требованием. Все это можно набрать и одной (но очень длинной) строкой. Однако многие разработчики считают, что код будет легче читаться, если ссылку на каждое изображение поместить в отдельной строке и воспользоваться пробелами (или табуляцией) для отступа строк, расположив ссылки друг под другом. Нужно лишь помнить о необходимости поставить запятую после каждой ссылки, за исключением последней, для которой обычно требуется точка с запятой, завершающая объявление.

---

Поскольку фоновые изображения обычно выстраиваются в плитку, зачастую необходимо также включать и объявление свойства `background-repeat` (если этого не сделать, изображения начнут выстраиваться в плитку одно над другим, закрывая собой все нижние изображения). Можно добавить еще и другие свойства фона, состоящие из простого набора значений, для которого в качестве разделителя используется запятая:

```
background-repeat: no-repeat,
                  no-repeat,
                  repeat-y;
```

При использовании нескольких значений, подобных этим, первое значение (в данном примере `no-repeat`) применяется для первого изображения, указанного в свойстве `background-image (scrollTop.png)`; второе используется со вторым изображением и т. д. Поскольку во всем этом можно быстро запутаться, многие веб-разработчики задействуют для указания нескольких изображений сокращенный метод записи:

```
background: url(scrollTop.jpg) center top no-repeat,
            url(scrollBottom.jpg) center bottom no-repeat,
            url(scrollMiddle.jpg) center top repeat-y;
```

---

**ПРИМЕЧАНИЕ**

Несколько фоновых изображений выкладываются друг на друга, как слои в программе редактирования изображений. Какое из изображений появится в верхнем слое, определяется порядком перечисления фоновых изображений. Изображение, указанное первым, появляется в верхнем слое элемента, второе — во втором слое, и последнее появляется в нижнем слое. В предыдущем примере кода верхняя часть свитка (`scrollTop.jpg`) находится выше его нижней части (`scrollBottom.jpg`), которая, в свою очередь, располагается выше текстовой области свитка (`scrollMiddle.jpg`).

---

Учтите, что такой прием создания нескольких фоновых изображений в Internet Explorer 8 не работает. Чтобы узнать о способе использования нескольких фоновых изображений в Internet Explorer 8, изучите следующую врезку

### **ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ**

#### **Имитация нескольких фоновых изображений**

Браузер Internet Explorer 8 используется все еще довольно широко (из-за широкого применения Windows XP).

Но поскольку Internet Explorer 8 не поддерживает несколько фоновых изображений, может сложиться

## ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

впечатление, что их применения нужно избегать. Но есть все же пара способов имитации нескольких фоновых изображений. Самый простой из них заключается в добавлении отдельных фоновых изображений к перекрывающимся элементам. Например, можно получить такой же эффект свитка, который показан на рис. 8.12, добавляя среднюю, текстовую область свитка к фону тега `<div>`, с помощью которого создается эта область боковой панели. Затем добавляется верхняя часть свитка к фону первого элемента боковой панели, в данном случае это тег `<h2>` с текстом `Announcements`. И наконец, к нижней части последнего элемента боковой панели — в данном случае это маркированный список — добавляется изображение нижней части свитка. И заголовок, и маркированный список находятся внутри `div`-контейнера боковой панели, и располагаются в верхней части `<div>`.

Во втором приеме используется содержимое, сгенерированное с помощью CSS (см. раздел «Различные свойства» в приложении 1). Генерируемое содержимое является способом добавления содержимого до и (или) после элемента. Можно добавить текст, но можно также добавить и изображение. Сгенерированное содержимое помещается над элементом, поэтому в примере свитка можно добавить

верхнюю часть свитка к псевдоклассу `:before`, а нижнюю часть свитка — к псевдоклассу `:after`:

```
.sidebar {
    background: url(scrollMiddle.jpg)
    center
    top repeat-y;
}
.sidebar:before {
    content: url(scrollTop.jpg);
}
.sidebar:after {
    content: url(scrollBottom.jpg);
}
```

Для перемещения изображений, чтобы они не находились в верхней или в нижней части элемента, нужно изучить использование абсолютного позиционирования, что позволит перемещать сгенерированный контекст. Более подробно вопрос абсолютного позиционирования рассмотрен в гл. 15.

Введенное в CSS3 применение нескольких фоновых изображений допускает любое количество таких изображений. При использовании сгенерированного содержимого вы ограничены всего лишь тремя изображениями: фоном элемента и его псевдоклассами `before` и `after`. Метод сгенерированного содержимого будет задействован в обучающем уроке этой главы.

## Использование градиентных фонов

Градиенты — плавный переход цвета от синего к красному или от черного к белому, входят в состав основных элементов любой программы редактирования изображений. Создание тонких переходов от одного схожего цвета к другому придает изображению своеобразную расплывчатость. Apple использует градиенты при изображении кнопок и других элементов пользовательского интерфейса своего программного обеспечения OS X и iPhone. Раньше для создания огромного файла с градиентом для последующего применения приходилось использовать программу Photoshop. Теперь задачу создания градиента можно возложить на браузер.

В CSS3 поддерживаются фоновые градиенты, представляющие собой фоновые изображения, создаваемые веб-браузером на лету. Фактически для создания градиента применяется обычное свойство `background-image`. Существует несколько типов градиентов на выбор.

## Линейные градиенты

Самым простым является линейный градиент. Он распространяется по прямой от одного конца элемента к другому, демонстрируя плавный переход от одного цвета

к другому (рис. 8.13). Нужно только указать его начало (край элемента), начальный и конечный цвет. Например, для создания градиента, который начинается с черного цвета с левого края элемента и плавно переходит в белый цвет на противоположном правом краю элемента, нужно указать следующий код:

```
background-image: linear-gradient(left,black,white);
```



**Рис. 8.13.** Линейные градиенты позволяют отказаться от устаревшего метода добавления градиентов к фону элемента: создание градиента в виде графического изображения в Photoshop или Fireworks с последующим использованием свойства `background-image` для помещения этого графического изображения в качестве фона элемента

#### ПРИМЕЧАНИЕ

Как уже упоминалось во врезке «Информация для новичков. Префиксы производителей» в разделе «Создание скругленных углов» гл. 7, для фоновых изображений можно просто воспользоваться значением `linear-gradient()`. Для его нормальной работы большинству браузеров требуется указание префикса производителя, например `-webkit-linear-gradient`.

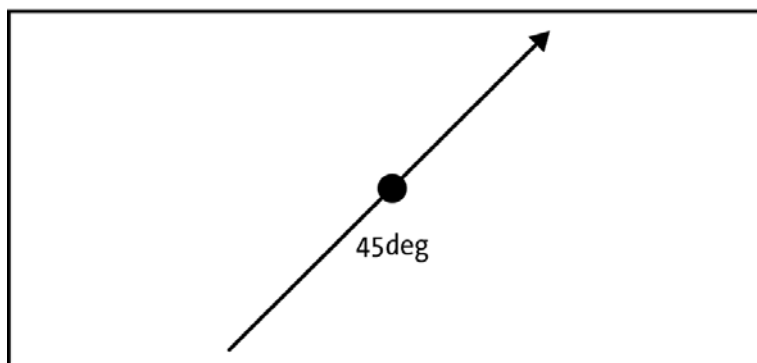
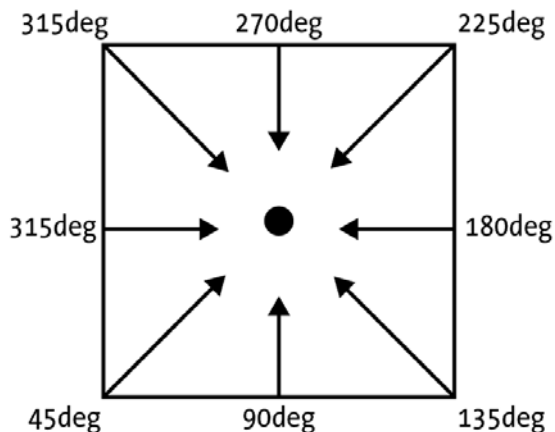
Градиент можно создать, используя одно из цветовых значений CSS, например ключевые слова вида `white` или `black`, шестнадцатеричные значения вида `#000000`, RGB-значения вида `rgb(0,0,0)` и т. д. Для указания стартовой позиции можно применить ключевые слова `left`, `right`, `top` или `bottom`. Фактически, если нужен градиент, изменяющий фоновый цвет элемента по диагонали, ключевые слова можно комбинировать. Например, чтобы нарисовать градиент, изменяющий цвет от белого к черному от верхнего левого угла к нижнему правому углу, нужно воспользоваться следующим кодом:

```
background-image: linear-gradient(top left,black,white);
```

Но вы не ограничены применением ключевых слов. Можно задать угол, определяющий направление градиента. Углы записываются в диапазоне от 0 до 360, а за ними указывается ключевое слово `deg`: `0deg` (рис. 8.14). Для получения нужного эффекта следует воспользоваться методом проб и ошибок, но все это работает примерно следующим образом: `0deg` означает, что градиент начинается в левой части и перемещается в правую часть, а при указании `45deg` он начинается в нижнем

левом углу и перемещается под углом  $45^\circ$  в верхний правый угол. Например, для градиента, созданного в нижнем левом изображении на рис. 8.13, используется следующее объявление:

```
background-image: linear-gradient(45deg, rgb(0,0,0), rgb(204,204,204));
```



**Рис. 8.14.** Значение в градусах, указанное для линейного градиента, используется для создания линии, проходящей через середину воображаемой точки центра элемента. Градиент рисуется вдоль этой линии. Значение определяет как точку старта, так и наклон градиента

При абсолютно квадратном элементе значение `45deg` представляет нижний левый угол и приводит к тому же самому результату, что и использование ключевого слова `bottom left`. То есть этот код:

```
background-image: linear-gradient(45deg, rgb(0,0,0), rgb(204,204,204));
```

аналогичен этому:

```
background-image: linear-gradient(bottom left, rgb(0,0,0), rgb(204,204,204));
```



Но если изображение имеет прямоугольную форму, эти два объявления не приведут к созданию одинаковых градиентов. Причина в том, что браузер проводит воображаемую линию через центр элемента. Указанный угол является углом наклона этой линии. Например, в широком прямоугольном элементе в нижней части рис. 8.14 линия, нарисованная под углом  $45^\circ$  через центр, не проходит из нижнего левого в верхний правый угол элемента. Она проходит где-то между левой и средней частью нижнего края элемента в точку между средней и правой частью его верхнего края.

## Цветовые опорные точки

В рассматриваемых выше градиентах использовалось только два цвета, но их можно добавить любое количество. Дополнительные цвета называются цветовыми опорными точками и позволяют фону осуществлять плавный переход от одного цвета ко второму, затем от второго к третьему и т. д., пока градиент не будет завершен последним цветом. Для задания цветовых опорных точек нужно просто добавить к градиенту дополнительные цветовые значения. Например, нижнее изображение справа на рис. 8.13 состоит из трех цветов и градиент переходит от белого к черному и от черного к белому цвету:

```
background-image: linear-gradient(left, black, white, black);
```

Разумеется, можно использовать любые цветовые значения (выраженные ключевыми словами, шестнадцатеричными числами или схемой RGB) и даже перемешивать их друг с другом, следовательно, предыдущую строку кода можно даже записать следующим образом:

```
background-image: linear-gradient(left, black, rgb(255,255,255), #FFFFFF, HSL(0,0,0%));
```

Веб-браузеры распределяют цвета равномерно, поэтому в данном примере в самой крайней левой точке фон черный, в середине фон белый, а в самой крайней правой точке он снова черный. Но можно разместить разные цветовые опорные точки и в более конкретно указанных местах фона, добавив после цвета второе значение.

---

### СОВЕТ

Если добавить к элементу с градиентным фоном свойство `background-position`, можно получить весьма необычные эффекты. По сути, можно изменить то, что браузером рассматривается в качестве стартовой точки градиента. Попробуйте и посмотрите, что получится.

---

Предположим, например, что нужно получить в начале градиента темно-красный цвет, затем быстро перейти к ярко-оранжевому примерно на 10 % длины элемента, оставляя оранжевый цвет на большей части элемента, а затем быстро вернуться к темно-красному цвету, как в верхнем изображении на рис. 8.15 (конечно, в книге черно-белое изображение, но общий смысл понятен). Для этого нужно использовать четыре цветовые опорные точки — красную, оранжевую,

оранжевую и красную, и указать где должны располагаться две оранжевые опорные точки:

```
background-image: linear-gradient(left, #900, #FC0 10%, #FC0 90%, #900);
```



**Рис. 8.15.** Градиенты с разными значениями опорных точек

Обратите внимание на то, что значение 10 % применяется ко второму цвету (оранжевому): тем самым браузеру сообщается, что он должен достичь этого цвета к 10 % длины элемента. Точно так же значение 90 % показывает, что браузеру нужно оставить оранжевый цвет до достижения 90 % длины элемента, а затем приступить к плавному переходу к темно-красному цвету на правом краю элемента.

**ПРИМЕЧАНИЕ**

Для указания позиций цветowych опорных точек не обязательно задавать проценты. Можно также использовать пиксели или `em`. Но указание процентных значений является более гибким решением, приспособившимся к изменениям ширины и высоты элемента. Одним из исключений является рассматриваемый далее повторяющийся линейный градиент. При создании этого типа плиточных градиентов очень хорошо подойдут пиксельные значения.

Как для первого, так и для последнего цвета не нужно указывать какие-либо значения положения, поскольку браузер предполагает, что первый цвет начинается с 0 %, а последний заканчивается на 100 %. Но если нужно сохранять начальный цвет неизменным на некоторой протяженности элемента, можно после первого цвета указать значение этой протяженности. Например, нижнее изображение на рис. 8.15 создано с помощью следующего объявления:

```
background-image: linear-gradient(left, #900 20%, #FC0 30%, #FC0 70%, #900 80%);
```

Обратите внимание, что у первого цвета — `#900` — также есть позиция 20 %. Это означает, что первые 20 % протяженности элемента (слева направо) будут иметь фон из сплошного красного цвета. Затем, от 20%- до 30%-ной точки градиент плавно перейдет от красного к оранжевому цвету.

## Префиксы производителей и поддержка Internet Explorer

Как уже упоминалось во врезке «Информация для новичков. Префиксы производителей» в гл. 7, многие из новейших CSS-свойств требуют указания перед именем свойства префикса производителя: `-webkit-` для Chrome и Safari, `-moz-` для Firefox и `-o-` для Opera. Это же требование распространяется и на градиенты. К сожалению, при этом приходится создавать объявление для каждого браузера. То есть, чтобы заставить работать с различными браузерами показанный ранее простой градиент, нужно создать следующий код:

```
background-image: -webkit-linear-gradient(left,black,white);
background-image: -moz-linear-gradient(left,black,white);
background-image: -o-linear-gradient(left,black,white);
background-image: linear-gradient(left,black,white);
```

Это увеличивает объем работы в четыре раза. Что еще хуже, градиенты не поддерживаются в Internet Explorer 9 и более ранних версиях. Если выбирать использование градиентов, для Internet Explorer 9 и более ранних версий нужно применять резервный цвет. Подберите сплошной цвет, соответствующий общему тону вашего градиента, и объявите его первым, а за ним разместите все объявления градиентов:

```
background-color: #FC0;
background-image: -webkit-linear-gradient(left, #900, #FC0, #900);
background-image: -moz-linear-gradient(left, #900, #FC0, #900);
background-image: -o-linear-gradient(left, #900, #FC0, #900);
background-image: linear-gradient(left, #900, #FC0, #900);
```

Internet Explorer 9 применит фоновый цвет и пропустит все остальные, непонятные ему объявления. Другие браузеры применяют фоновый цвет, но также создадут градиент, который накроет фоновый цвет. Если используются RGBA-цвета с некоторой степенью прозрачности, то вам не захочется, чтобы сквозь них просматривался фоновый цвет. В таком случае примените сокращенную форму записи свойства `background`, и свойство `background-color` будет отменено (благодаря особому поведению сокращенной формы записи свойства `background`, рассмотренной в разделе «Сокращенный вариант свойства `background`» данной главы). Тогда для объявления цвета в формате RGBA можно будет воспользоваться следующим кодом:

```
background-color: #FC0;
background: -webkit-linear-gradient(left, rgba(153,0,0,.5), #FC0,
    rgba(153,0,0,.5));
background: -moz-linear-gradient(left, rgba(153,0,0,.5),
    #FC0, rgba(153,0,0,.5));
background: -o-linear-gradient(left, rgba(153,0,0,.5), #FC0,
    rgba(153,0,0,.5));
background: linear-gradient(left, rgba(153,0,0,.5), #FC0, rgba(153,0,0,.5));
```

---

#### ПРИМЕЧАНИЕ

Как видите, для создания градиентов используется большой объем кода. Кроме того, устаревшие версии WebKit (браузеры Safari на операционных системах Mac и iOS, а также браузер Android) используют для линейных градиентов совершенно другой синтаксис. Поэтому вполне вероятно, что вам захочется воспользоваться рассматриваемым чуть позже интерактивным средством создания градиентов. Оно в целом позволит существенно упростить процесс создания градиентов.

---

## Повторяющиеся линейные градиенты

Обычно линейный градиент заполняет весь элемент с первого цвета с одного края элемента до последнего цвета с его противоположного края. Но при желании выполнить шаблонные градиенты можно создать их повторяющийся вариант. По сути, при этом определяется градиент с указанными опорными точками, браузер рисует градиент, а затем повторяет этот шаблон, распространяя его в виде плиток по фону элемента. Например, чтобы получить повторяющийся градиент, как в изображении, показанном слева на рис. 8.16, можно написать следующий код:

```
background-image: repeating-linear-gradient(bottom left, #900 20px, #FC0 30px,
    #900 40px)
```

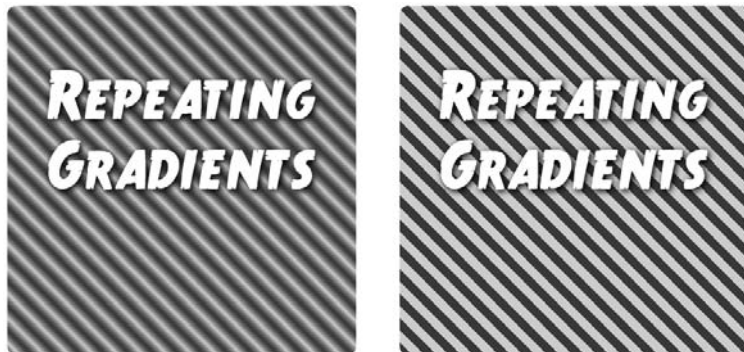
В подобных случаях для опорных точек лучше всего подойдут пиксельные значения. Получается, что браузер рисует градиент, начинающийся в нижнем левом углу с 20 пикселей темно-красного цвета, после этого до 30 пикселей происходит переход к оранжевому цвету, а затем до 40 пикселей выполняется обратный переход к темно-красному цвету. После того как этот градиент будет нарисован, браузер просто выкладывает его в виде плитки в фоне элемента.

---

#### ПРИМЕЧАНИЕ

Для экспериментов с градиентами имеются широчайшие возможности. Чтобы посмотреть только лишь на малую часть тех удивительных результатов, которых можно достичь в ходе таких экспериментов, посетите галерею шаблонов CSS3 по адресам <http://lea.verou.me/css3patterns/>,

[www.standardista.com/cssgradients/](http://www.standardista.com/cssgradients/) и оцените удивительные возможности использования градиентов для создания флагов со всего мира по адресу [www.standardista.com/CSS3gradients/flags.html](http://www.standardista.com/CSS3gradients/flags.html).



**Рис. 8.16.** Используя повторяющиеся линейные градиенты CSS3, можно создать полосатые шаблоны

Можно также применять повторяющиеся градиенты для создания однотонных полос без каких-либо излишеств и переходов между цветами. Например, изображение, показанное справа на рис. 8.16, создано с помощью следующего объявления:

```
background-image: repeating-linear-gradient(45deg, #900 0, #900 10px, #FC0 10px, #FC0 20px);
```

Оно начинается с темно-красного цвета (#900) в точке 0 и переходит снова к красному цвету в точке 10px. Поскольку переход задается между одинаковыми цветами, браузер рисует его в виде сплошного цвета. Затем на 10 пикселях задается переход к оранжевому цвету (#FC0). Поскольку это та же самая точка, в которой заканчивается красный цвет, никакого плавного перехода не происходит, шаблон не содержит перехода от красного к оранжевому цвету. И наконец, переход к такому же оранжевому цвету осуществляется до 20 пикселей, что создает еще одну сплошную линию. Поскольку это повторяющийся линейный градиент, браузер просто заполняет шаблоном фон элемента.

А теперь плохие новости. Повторяющиеся градиенты работают в большинстве браузеров, но только не в старых версиях Safari (до 5.1 или iOS 5) или в Internet Explorer 9 и более ранних версиях. Поэтому желательно сделать запасной вариант фонового цвета; нужно также обеспечить добавление всех объявлений с префиксами производителей, чтобы первый повторяющийся градиент принял следующий вид:

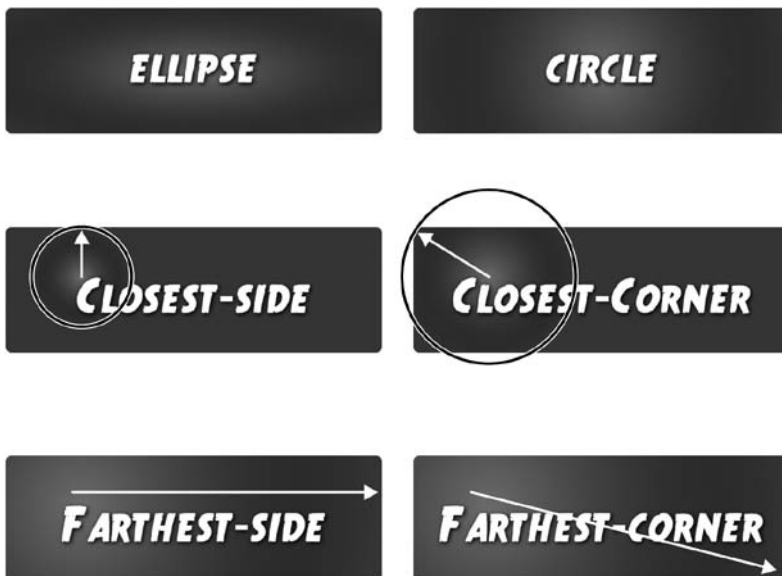
```
background-color: #FC0;
background: -webkit-repeating-linear-gradient(bottom left, #900 20px, #FC0 30px, #900 40px)
background: -moz-repeating-linear-gradient(bottom left, #900 20px, #FC0 30px, #900 40px)
```

```
background: -o-repeating-linear-gradient(bottom left, #900 20px, #FC0 30px,
#900 40px)
background: repeating-linear-gradient(bottom left, #900 20px, #FC0 30px, #900
40px)
```

## Радиальные градиенты

В CSS также есть способ создания радиальных градиентов, то есть градиентов, расходящихся наружу по круговой или эллиптической схеме (рис. 8.17). Синтаксис похож на тот, который применяется для линейных градиентов, нужно только предоставить начальный цвет (цвет в середине градиента) и конечный цвет (цвет в конце градиента). Например, левое изображение слева на рис. 8.17 выполнено с помощью следующего кода:

```
background-image: radial-gradient(red, blue);
```



**Рис. 8.17.** Радиальные градиенты бывают эллиптические (верхнее изображение слева) или круговые (верхнее изображение справа). Позицией градиента в фоне элемента, а также его размерами можно управлять

Он создает эллиптическую форму, помещающуюся в высоту и ширину элемента с центром градиента (где начинается красный цвет) в центре элемента.

Можно также создать круговые градиенты, добавив перед указанием цветов ключевое слово `circle`. Например, верхнее изображение справа на рис. 8.17 создано с помощью следующего кода:

```
background-image: radial-gradient(circle, red, blue);
```

Обычно браузер рисует центр радиального градиента в центре элемента, но центр градиента можно позиционировать с помощью таких же ключевых слов позиционирования и значений, которые используются для свойства `background-position`.

Например, посередине слева изображение на рис. 8.17 является радиальным градиентом, который начинается с отступом на 20 % от левого края элемента и с отступом на 40 % от верхнего края элемента. Значения позиционирования указываются перед ключевыми словами, определяющими форму и цвета:

```
background-image: radial-gradient(20% 40%, circle, red, blue);
```

Чтобы указать размер градиента, можно воспользоваться одним из четырех ключевых слов.

- `closest-side` — предписывает браузеру создать градиент, распространяющийся из центра только до ближайшей к центру стороне элемента. Например, в среднем левом изображении на рис. 8.17 ближайшей к центру градиента стороной является верхний край элемента, поэтому радиус окружности простирается от центра до этого края. Тем самым весь градиент остается внутри элемента. Код для создания этого градиента имеет следующий вид:

```
background-image: radial-gradient(20% 40%, circle closest-side, red, blue);
```

Когда ключевое слово `closest-side` применяется к эллиптическому градиенту, для вычисления  $x$  и  $y$  радиусов эллипса используются обе ближайшие стороны (то есть верхняя или нижняя и левая или правая).

- `closest-corner` — приводит к вычислению ширины градиента из его центра до ближайшего угла элемента (изображение справа посередине на рис. 8.17). Это может означать выход градиента за пределы элемента. Например, окружность, нарисованная в верхней части среднего правого изображения на рис. 8.17 показывает фактический размер градиента. Часть градиента выходит за границы элемента:

```
background-image: radial-gradient(20% 40%, circle closest-corner, red, blue);
```

- `farthest-side` — приводит к вычислению радиуса окружности от ее середины до самой дальней стороны элемента. В случае эллиптического градиента это расстояние от центра до любой самой дальней верхней или нижней стороны и до любой самой дальней левой или правой стороны. Код, с помощью которого создано нижнее левое изображение на рис. 8.17, имеет следующий вид:

```
background-image: radial-gradient(20% 40%, circle farthest-side, red, blue);
```

- `farthest-corner` — приводит к вычислению радиуса окружности от ее центра до самого дальнего угла элемента. Код, с помощью которого создано нижнее правое изображение на рис. 8.17, имеет следующий вид:

```
background-image: radial-gradient(20% 40%, circle farthest-corner, red, blue);
```

Как и в случае использования линейных градиентов, можно применять несколько цветовых опорных точек и указывать для них конкретные места. Предположим, например, что нужно создать круговой градиент, составленный из темно-красного, оранжевого и желтого цветов. Нужно, чтобы красный цвет сохранялся на некотором протяжении до его перехода в оранжевый, а затем оранжевый цвет сохранялся на некотором протяжении до перехода в желтый, появляющийся в конце градиента. Чтобы указать места появления того или иного цвета, можно воспользоваться процентными значениями:

```
background-image: radial-gradient(20% 40%, circle, red 20%, orange 80%, yellow);
```



Разумеется, как и в случае линейного градиента, можно воспользоваться допустимыми в CSS цветовыми значениями и переписать этот код следующим образом:

```
background-image: radial-gradient(20% 40%, circle, rgb(255,0,0) 20%,  
rgb(255,165,0) 80%, #FFFF00);
```

Как и в случае с линейными градиентами, Internet Explorer 9 и более ранние версии при использовании радиальных градиентов также не будут понимать, о чем идет речь, то же самое касается версий Safari, предшествующих версии 5.1 (и iOS 5). Нужно добавить резервный фоновый цвет, а также все префиксы производителей. Например:

```
background-color: red;  
background: -webkit-radial-gradient(20% 40%, circle, red 20%, orange 80%, yellow);  
background: -moz-radial-gradient(20% 40%, circle, red 20%, orange 80%, yellow);  
background: -o-radial-gradient(20% 40%, circle, red 20%, orange 80%, yellow);  
background: radial-gradient(20% 40%, circle, red 20%, orange 80%, yellow);
```

## Повторяющиеся радиальные градиенты

Как и в случае с линейными градиентами, радиальные градиенты также можно создать повторяющимися, что подойдет для изображения подобия бычьего глаза или для гипнотизирования посетителей вашего сайта. Чтобы браузер знал размер отдельно взятого радиального градиента и смог его повторить, нужно обязательно добавить к различным цветовым опорным точкам процентные, пиксельные или em-значения. Например:

```
background-image: repeating-radial-gradient(circle, red 20px, orange 30px,  
yellow 40px, red 50px);
```

Учтите, что для создания повторяющихся радиальных градиентов без резких переходов, градиент нужно завершать тем же цветом, с которого он начинался (в данном примере это красный цвет). Тем самым будет обеспечено плавное возвращение цвета к начальному. Если этого не сделать (например, если последним цветом в показанном выше коде сделать желтый), то получится резкая граница там, где заканчивается последний цвет и в повторяющемся градиенте начинается первый цвет.

### СОВЕТ

---

Поскольку веб-браузеры считают линейные и радиальные градиенты просто фоновым изображением, для них можно использовать и другие свойства фоновых изображений, такие как `background-size`, `background-position` и т. д. Кроме того, в одном и том же стиле можно применять несколько градиентов, перечислив их через запятую, точно так же, как ранее это делалось с несколькими фоновыми изображениями. На самом деле в одном и том же элементе можно смешивать изображения с градиентами.

---

## Применение градиентов, создаваемых с помощью Colorzilla

Сложность градиентов и их ограниченная поддержка браузерами могут склонить вас к решению не использовать их в своих таблицах CSS. К счастью, есть

интерактивное средство, позволяющее создавать большинство типов градиентов (за исключением повторяющихся) буквально одним щелчком. Это средство называется Ultimate CSS Gradient Generator ([www.colorzilla.com/gradient-editor/](http://www.colorzilla.com/gradient-editor/)), и показано на рис. 8.18. Слева, ближе к верхнему краю страницы, вы найдете галерею уже созданных градиентов. Можете выбрать один из градиентов в качестве отправной точки и настроить его цвета и цветовые опорные точки в соответствии с собственными представлениями о будущем градиенте или же создайте свой собственный градиент.

**Ultimate CSS Gradient Generator**  
A powerful Photoshop-like CSS gradient editor from ColorZilla.

For Firefox For Chrome Gradient Generator

**1** Presets

Name: Custom

**3** **5**

**7** Stops

Opacity:  Location:  %

**4** Color:  Location: 40 %

Adjustments

Sponsor

**8** CSS

```
background: rgb(240,183,161); /* Old browsers */
/* IE9 SVG, needs conditional override of
'filter' to 'none' */
background:
url(data:image/svg+xml;base64,PD94bWwgdmVyc2lvdj
background: -moz-linear-gradient(left,
rgb(240,183,161) 0%, rgb(140,51,16,1) 50%,
rgb(117,34,1,1) 51%, rgb(191,110,78,1)
100%); /* FF3.6+ */
background: -webkit-gradient(linear, left top,
right top, color-stop(0%,rgb(240,183,161)),
color-stop(50%,rgb(140,51,16,1)), color-
stop(51%,rgb(117,34,1,1)), color-
stop(100%,rgb(191,110,78,1))); /*
Chrome, Safari4+ */
background: -webkit-linear-gradient(left,
rgb(240,183,161,1) 0%,rgb(140,51,16,1)
50%,rgb(117,34,1,1) 51%,rgb(191,110,78,1)
100%); /* Chrome10+,Safari5.1+ */
background: -o-linear-gradient(left,
rgb(240,183,161,1) 0%,rgb(140,51,16,1)
50%,rgb(117,34,1,1) 51%,rgb(191,110,78,1)
100%); /* Opera 11.10+ */
background: -ms-linear-gradient(left,
rgb(240,183,161,1) 0%,rgb(140,51,16,1)
50%,rgb(117,34,1,1) 51%,rgb(191,110,78,1)
100%); /* IE10+ */
```

Color format:   Comments  IE9 Support

Permalink

Link to, save or share the current gradient using its unique link.

**Рис. 8.18.** Самым простым способом начала работы с градиентами является использование такого интерактивного средства, как Ultimate CSS Gradient Generator

Хотя это интерактивное средство представлено всего лишь одной веб-страницей, у него имеется множество настроек. Рассмотрим один из способов его применения.

1. Начните с предварительной установки, выбрав один из градиентов в окне Presets (Предустановки) (см. рис. 8.18, 1).

Это делать необязательно, но создание градиента может пойти значительно быстрее, если начать с чего-нибудь похожего на желаемый результат.

2. В меню Orientation (Ориентация) укажите тип градиента (см. рис. 8.18, 2).

Выбор в этом средстве ограничен радиальными и линейными градиентами, следующими сверху вниз, слева направо, из верхнего левого в нижний правый угол, из нижнего левого в верхний правый угол или по диагонали. Но после создания кода можно заменить ключевое слово угла (top left, например) значением градуса (допустим, 65deg). Средство также создает эллиптические радиальные градиенты. Если нужен круговой градиент, следует просто найти ключевое слово ellipse и заменить его в итоговом коде CSS ключевым словом circle.

Но создавать повторяющиеся линейные или повторяющиеся радиальные градиенты это средство не позволяет.

3. Для изменения цвета или позиции цветовой опорной точки щелкните на ней, чтобы появилась возможность выбора (см. рис. 8.18, 3).

Если вам не подойдут предустановленные цвета, нужно будет отредактировать каждую цветовую опорную точку.

4. Для изменения цвета цветовой опорной точки щелкните на поле Color (Цвет) (см. рис. 8.18, 4).

Появится окно выбора цвета. Можно щелкнуть на нужном цвете или, если известно значение цвета в RGB, в шестнадцатеричном виде или в схеме HSL, ввести это значение. Чтобы выбрать новый цвет и закрыть это окно, щелкните на кнопке ОК.

5. Дополнительно в поле Location (Расположение) справа от поля выбора цвета можно установить позицию цветовой опорной точки (см. рис. 8.18, 5).

Наберите процентное значение или, что еще проще, перетащите цветовую опорную точку на панели градации (см. рис. 8.18, 3).

---

#### ПРИМЕЧАНИЕ

Чтобы удалить цветовую опорную точку, выберите ее на панели градации (см. рис. 8.18, 3) и щелкните на кнопке delete (Удалить), расположенной ниже панели.

---

6. Добавьте дополнительные цветовые опорные точки, щелкнув ниже панели градации в том месте, где нужно добавить такую точку.

После добавления новой цветовой опорной точки измените ее цвет и расположение в соответствии со своими предпочтениями (шаги 4 и 5).

7. Можно также изменить прозрачность градиента, что сделает его начало или конец немного или полностью прозрачными. Выберите одну из опорных точек выше панели градации (см. рис. 8.18, 6), а затем настройте ее прозрачность и позицию (см. рис. 8.18, 7).

---

#### ПРИМЕЧАНИЕ

Ultimate CSS Gradient Generator также пытается создать градиент, каким-то образом работающий в Internet Explorer 9 и более ранних версиях. Установите флажок IE в верхней правой части области предварительного просмотра (Preview) (см. рис. 8.18), чтобы включить эффект, использующий

собственное свойство Internet Explorer под названием `filters` для получения градиента. К сожалению, Internet Explorer может имитировать только линейные градиенты, которые простираются слева направо или сверху вниз и не работает ни с радиальными градиентами, ни с градиентами, устанавливаемыми под определенным углом.

8. После создания градиента укажите на область CSS и щелкните на появившейся кнопке `copy` (скопировать) (см. рис. 8.18, 8).

Код CSS будет скопирован в буфер обмена вашего компьютера. Затем можно будет перейти в избранный вами текстовый редактор и вставить код в тот стиль, в который нужно добавить фоновый градиент.

Для имитации градиента для Internet Explorer 9 и более ранних версий Ultimate CSS Gradient Generator использует свойство `IE-only` (Только Internet Explorer). Фактически предоставляются два альтернативных варианта: один для Internet Explorer 9 и один для более ранних версий. Вариант, предназначенный для Internet Explorer 9, для достижения требуемого результата использует другую технологию вывода изображений — SVG (Scalable Vector Graphics, масштабируемая векторная графика), применение которой представляется весьма неплохой идеей. А в том варианте, который предназначен для более ранних версий Internet Explorer, задействуется фильтр (`filter`), который не всегда может правильно имитировать градиент и замедляет производительность работы Internet Explorer. По этой причине лучше будет проигнорировать Internet Explorer 8 и более ранние версии. Для этого нужно сделать следующее.

1. Установить флажок `IE9 Support` (Поддержка Internet Explorer 9) в нижнем правом углу, ниже области кода CSS (см. рис. 8.18). После этого будет добавлен код для создания SVG-градиента.
2. Скопировать код CSS (шаг 8 в предыдущем списке действий) и вставить его в свой CSS.
3. Найти код фильтра Internet Explorer и удалить его.

Последняя строка скопированного кода должна иметь примерно следующий вид:

```
filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#3f4c6b',
endColorstr='#7ed6d3',GradientType=1 ); /* Internet Explorer 6-8 не показывают
горизонтальный градиент */
```

Если вам действительно нужна поддержка Internet Explorer 8, не используйте этот код, а протестируйте градиент в Internet Explorer 8, чтобы убедиться в его правильном отображении.

4. И наконец, добавьте свойство `background-color` в тот промежуток, где находился удаленный код фильтра Internet Explorer; этот цвет будет появляться в фоне для Internet Explorer 8 и более ранних версий.

## Обучающий урок 1: совершенствуем изображения

Фотогалерея — отличный пример привлекательной веб-страницы. Этот обучающий урок 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 рассмотренное ранее свойство `box-shadow` 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

Мы будем работать над веб-страницей вымышленного сайта `CosmoFarmer.com` (рис. 8.19). У нас уже имеется присоединенная внешняя таблица стилей, создающая модель веб-страницы и обеспечивающая ей простейший дизайн.

1. Откройте файл `image.html` из папки `08\image_ex` в браузере.

На рис. 8.19 форматирование веб-страницы выполнено исключительно средствами языка HTML. Имеются недостатки. В частности, изображения занимают на странице слишком много места (см. рис. 8.19, *вверху*). С помощью кода CSS (см. рис. 8.19, *внизу*) 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, используемую 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: 10px;
```

Свойство `float` смещает изображение к правой стороне веб-страницы, приподнимая текст вверх и создавая обтекание фотографии с левой стороны. 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, цвет фона и отступы. Законченный вариант стиля должен иметь следующий вид:

```
img.figure {  
  float: right;  
  margin-left: 10px;  
  margin-bottom: 10px;  
  border: 1px solid #666;  
  background-color: #CCC;  
  padding: 10px;  
}
```

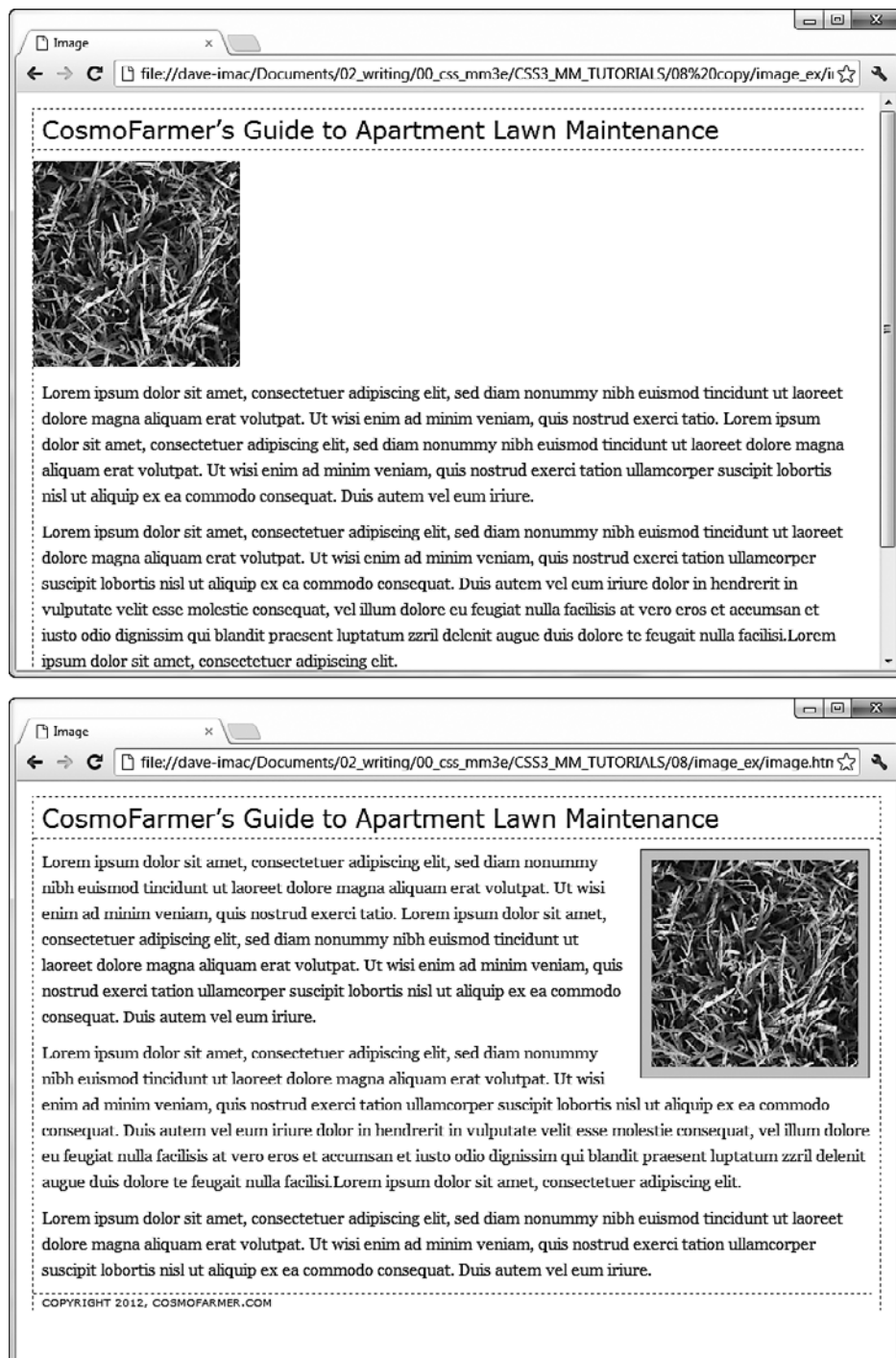


Рис. 8.19. Два варианта веб-страницы: до (вверху) и после (внизу) CSS-стилизации



Если вы сохраните и просмотрите веб-страницу в браузере прямо сейчас, то не будет видно никаких изменений, поскольку стилевой класс не возьмется 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:

```

```

Теперь изображение приобретет все форматизирующие параметры, определенные для стилевого класса `.figure`.

7. Просмотрите веб-страницу в браузере. Она должна выглядеть так, как страница, представленная на рис. 8.19, *внизу*.

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 данной обучающей программы мы добавим под фотографией текстовую подпись.

Нередко требуется добавить к изображению подрисовочную подпись с подробной информацией об объекте, месте съемки и т. д. При этом вы наверняка захотите, чтобы комментарий был связан с изображением, чтобы рядом стоящий текст обтекал надпись так же, как и само изображение. Лучший способ — заключить изображение и текст в контейнер, 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.

К счастью, в HTML5 именно для решения нашей задачи включены два новых тега: тег `<figure>` предназначен для заключения в него изображения, из которого нужно сделать иллюстрацию. В дополнение к нему может использоваться тег `<figcaption>`, который включает в себя подпись под изображением. Сначала нужно немного подправить HTML-код.

#### ПРИМЕЧАНИЕ

---

Чтобы заставить Internet Explorer 8 и более ранние версии распознавать теги `<figure>` и `<figcaption>`, нужно включить JavaScript-файл HTML5shiv. См. врезку «Обходной прием. Как заставить Internet Explorer 8 понимать HTML5» в разделе «Дополнительные теги в HTML5» гл. 1.

---

8. Вернитесь в текстовый редактор к файлу `image.html`. Найдите в коде тег `<img>` и сначала удалите `class="figure"`, который был добавлен при выполнении шага 6, а затем добавьте перед этим тегом тег `<figure>`. Он отметит собой начало



контейнера. Теперь будет добавлена подпись, и тег `<figure>` будет закрыт, чтобы обозначить конец контейнера.

9. После тега `<img>` добавьте код, выделенный ниже полужирным шрифтом, чтобы HTML-код приобрел следующий вид:

```
<figure>
```

```

```

```
<figcaption>Figure 1: Creeping Bentgrass is best suited for outdoor use and should be avoided by the indoor farmer.
```

```
</figcaption>
```

```
</figure>
```

10. Прокрутите файл до стиля `.figure`, созданного ранее, и удалите этот стиль.

Для элемента `figure` будет добавлен новый стиль.

11. Добавьте следующий стиль к файлу `styles.css`:

```
figure {
  float: right;
  width: 222px;
  margin: 15px 10px 5px 10px;
}
```

В предыдущем уроке свойство `float: right` уже использовалось, а свойство `margin` добавит вокруг всех четырех сторон тега `<figure>` небольшое пустое пространство. Возникает вопрос: а для чего применяется свойство `width`? Хотя у фотографии и есть установленная ширина (200 пикселей), ее нет у абзаца подписи. Если не указать ширину, абзац заставит тег `<figure>` стать шире фотографии. В данном случае нужно, чтобы подпись была такой же по ширине, как фотография и ее рамка.

Значение 222 пикселя получилось в результате небольших математических вычислений общей области, занимаемой фотографией на странице: хотя сама фотография занимает в ширину 200 пикселей, у нее есть по 10 пикселей отступов влево и вправо, а также у изображения имеется рамка, занимающая слева и справа по 1 пикселу, что в целом и составляет ширину фотографии от одной границы к другой (см. подраздел «Вычисление фактических размеров блочных элементов» раздела «Определение параметров высоты и ширины» гл. 7). Затем мы добавим форматирование к изображению.

12. Добавьте к файлу `styles.css` следующий стиль:

```
figure img {
  border: 1px solid #666;
  background-color: #CCC;
  padding: 10px;
}
```

Этот селектор потомка воздействует на любой тег `<img>`, который находится внутри тега `<figure>`. Поскольку здесь используется селектор потомка, класс к тегу `<img>` добавлять не нужно. Затем добавим стиль к подписи.

13. Добавьте к таблице стилей `styles.css` следующий стиль:

```
figcaption {
  font: bold 1em/normal Verdana, Arial, Helvetica, sans-serif;
  color: #333;
  text-align: center;
}
```

В этом стиле используются некоторые свойства, рассмотренные в гл. 6, с помощью которых осуществляется выравнивание по центру, задание полужирного шрифта и серого цвета для подписи, выполняемой шрифтом Verdana. К счастью сокращенная форма записи свойства `font` в первой строке позволяет свернуть четыре разных свойства в одно объявление стиля.

Чтобы выделить подпись еще больше, добавим к ней фоновый цвет и рамку.

14. Добавьте к стилю `figcaption` три свойства:

```
figcaption {
  font: bold 1em/normal Verdana, Arial, Helvetica, sans-serif;
  color: #333;
  text-align: center;
  background-color: #e6f3ff;
  border: 1px dashed #666;
  padding: 5px;
}
```

Свойства `background-color`, `border` и `padding` предназначены для создания расцвеченного прямоугольника, охватывающего подпись. А теперь настало время посмотреть на результаты труда.

15. Сохраните файлы `image.html` и `styles.css` и просмотрите файл `image.html` в веб-браузере.
16. Теперь можно понять одну из причин, по которой разработку дизайна проще вести с использованием внешней таблицы стилей — приходится работать только с одним файлом, а не с двумя и сохранять только его. Страница должна иметь вид, показанный на рис. 8.20. (Полную версию этой страницы можно найти в папке `08_finished\01_image_ex.`)

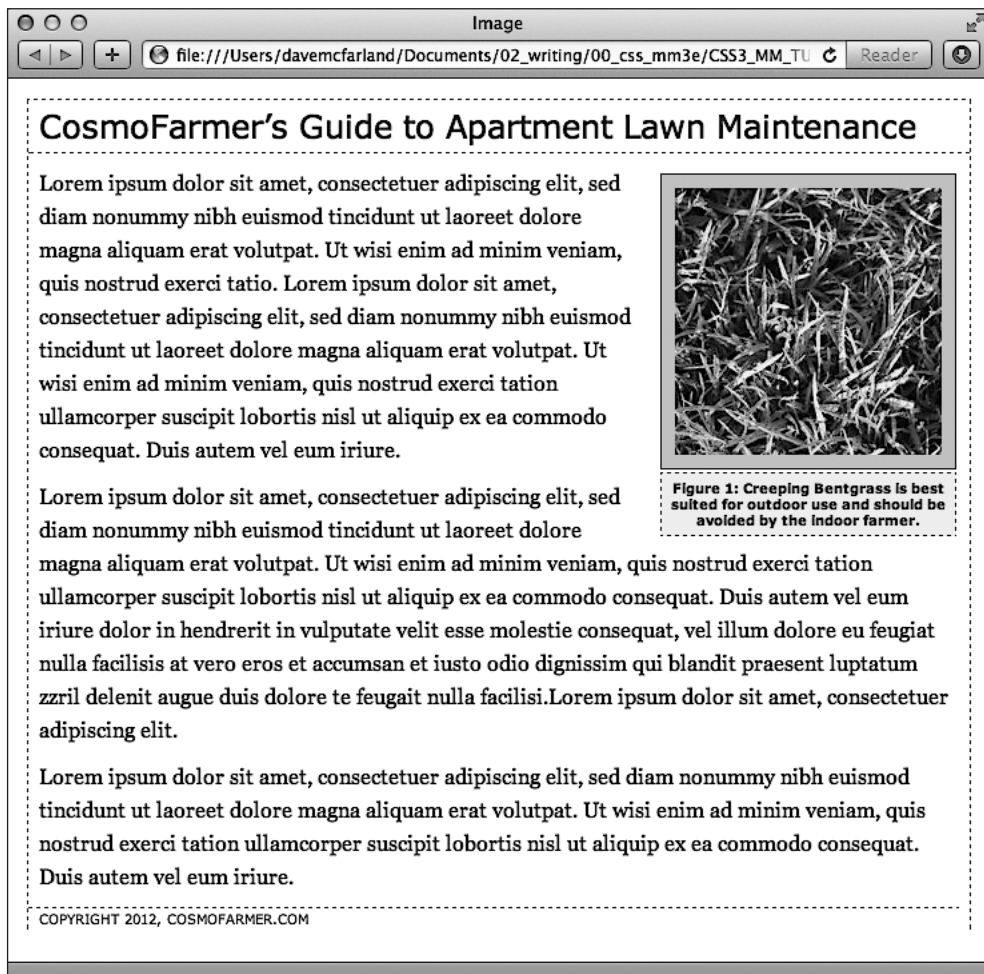
## Обучающий урок 2: создание фотогалереи

Раньше веб-дизайнеры создавали фотогалереи на основе HTML-тега таблицы `<table>`, помещая изображения в ячейки, образуемые строками и столбцами. Но сейчас вы можете достичь 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\02\gallery_ex.`

Посмотрите на HTML-код, который использован для создания фотогалереи. Веб-страница содержит шесть фотографий и подписей к ним. Каждая фотография и подпись к ней содержится в контейнере `<figure>`. Сама фотография находится в теге `<img>`, а подпись к ней — в теге `<figcaption>`.

```
<figure>
  
<figcaption>Figure 6: The dandelion: scourge of the apartment farmer.
</figcaption>
</figure>
```



**Рис. 8.20.** Использование в качестве контейнера тега `<figure>`, перемещения вправо и небольших стилевых настроек упрощает добавление подписей к фотографиям

#### ПРИМЕЧАНИЕ

Поскольку здесь используются теги HTML5, чтобы заставить Internet Explorer 8 и более ранние версии распознавать теги `<figure>` и `<figcaption>`, нужно включить JavaScript-файл HTML5shiv. См. врезку «Обходной прием. Как заставить Internet Explorer 8 понимать HTML5» в разделе «Дополнительные теги в HTML5» гл. 1.

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>
figure img {
  border: 1px solid #666;
  background-color: #FFF;
  padding: 4px;
}

figcaption {
  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. В первом стиле для выбора только тех изображений, которые находятся внутри тегов `<figure>`, используется селектор потомка.

Теперь поместим фотографии рядом друг с другом.

---

#### ПРИМЕЧАНИЕ

При вставке внутренней таблицы стилей убедитесь в том, что вы поместили ее в заголовок веб-страницы после тега `link` и перед условным комментарием IE, с помощью которого прикрепляется JavaScript-файл HTML5shiv.

---

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 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 пикселей на фотографию, 8 пикселей на левый и правый отступы и 2 пикселя на левую и правую границы-рамки.

---

#### СОВЕТ

В нашей экспериментальной фотогалерее все изображения имеют одинаковую ширину. На практике же размеры различаются. В следующей врезке 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.

---

## ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

### Если ширина одного изображения отличается от остальных

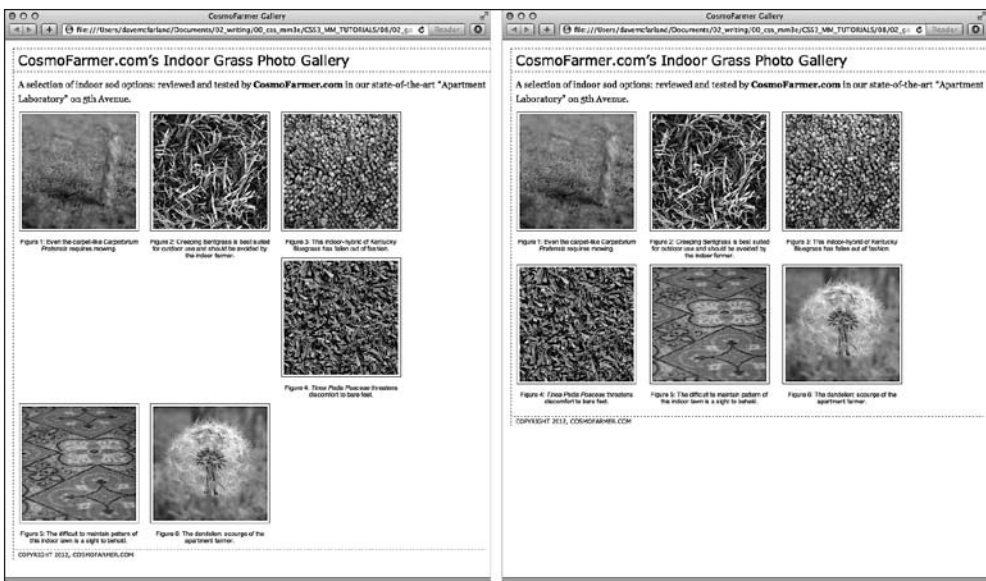
Создать фотогалерею, которая похожа на галерею от CosmoFarmer, очень просто. В описываемом случае все фотографии имеют одинаковую ширину. Но как следует поступить, если у вас фото различных размеров? Одно из решений — сделать стиль для каждой отдельно взятой ширины и применить его к тегу `<figure>` отдельного изображения (это небольшая работа, с таким же успехом можно отредактировать фотографии, приведя их ширину к одному размеру).

Например: `<figure class="w300">`.

Теперь создайте стилевой класс, например `.w300`, и определите ширину изображения (в данном случае 300) плюс 10 на отступы и границы: `.w300 { width: 310 };`.

Теперь создайте стилевой класс, например `.w300`, и определите ширину изображения (в данном случае 300) плюс 10 на отступы и границы: `.w300 { width: 310 };`.

5. Сохраните файл и просмотрите веб-страницу `gallery.html` в браузере. Она должна быть похожа на страницу, представленную на рис. 8.21, а.



а

б

**Рис. 8.21.** Плавающие элементы, расположенные рядом друг с другом, — один из способов имитации столбцов и строк таблицы

На рис. 8.21 метод работает неправильно, если элементы имеют различную высоту (см. рис. 8.21, а). Использование свойства `display: inline-block` — еще один способ заставить элементы располагаться в ряд (см. рис. 8.21, б), без ненужного переноса изображений вниз, показанного слева.

Измените ширину окна браузера так, чтобы оно стало уже или шире, и наблюдайте, 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 гл. 13). К счастью, существует простое решение этой проблемы.

- Вернитесь к файлу `gallery.html` в HTML-редакторе. Найдите стиль `figure`. Удалите свойство `float:left` и добавьте свойства `display: inline-block; vertical-align: top`. Стиль должен приобрести следующий окончательный вид:

```
figure {  
  display: inline-block;  
  vertical-align: top;  
  width: 210px;  
  margin: 0 10px 10px 10px;  
}
```

Свойство `display: inline-block` (см. подраздел «Горизонтальные панели навигации» раздела «Создание панелей навигации» гл. 9) рассматривает каждую пару «изображение/подпись» в качестве блока (у которого есть высота и ширина), но также и в качестве линейного элемента (то есть блоки могут выстраиваться в ряд). Кроме того, свойство `vertical-align` со значением `top` гарантирует выравнивание каждого тега `<figure>` по верхнему краю других тегов `<figure>`, имеющихся в данном ряду.

- Сохраните файл и воспользуйтесь предварительным просмотром веб-страницы в браузере. Посмотрите на рис. 8.21, б.

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 ее можно сделать еще выразительнее. Добавление теневых эффектов каждой фотографии придаст веб-странице иллюзию глубины и обеспечит реалистичность трехмерного пространства. Однако не спешите запускать программу для редактирования фотографий типа Photoshop. Добавить теневые эффекты любому изображению веб-страницы можно автоматически средствами CSS. Перед появлением CSS3 (и в предыдущих изданиях этой книги) приходилось проходить через сложный процесс добавления теней к изображениям данной галереи путем создания дополнительных `div`-контейнеров вокруг изображений и использования двух фоновых изображений, и все равно тени не были полностью похожи на настоящие. К счастью в CSS3 все делается настолько просто, что весь урок удалось свести всего к трем шагам.

- Откройте в HTML-редакторе файл `styles.css`, над которым вы работали в предыдущей части практического урока.

Изменение коснется созданного ранее стиля `img`.

- Добавьте в конец стиля `figure img` свойство `box-shadow: 2px 2px 4px rgba(0,0,0,.5);`, чтобы придать ему следующий вид (изменения выделены полужирным шрифтом):

```
figure img {  
  border: 1px solid #666;  
  background-color: #FFF;
```



```
padding: 4px;
box-shadow: 2px 2px 4px rgba(0,0,0,.5);
}
```

Свойство `box-shadow` было рассмотрено в разделе «Добавление теней» гл. 7. Здесь добавляется тень, распространяющаяся на 2 пиксела вправо от изображения и на 2 пиксела ниже его, которая расширяется наружу на 4 пиксела. Используя цветовую схему RGBA (см. подраздел «RGBA» раздела «Придание тексту цветового оформления» гл. 6), можно установить для тени черный цвет с 50%-ной прозрачностью.

3. Сохраните файл и просмотрите веб-страницу в браузере. Она должна иметь вид, представленный на рис. 8.22.



**Рис. 8.22.** Добавление теневого эффекта к фотографиям придает веб-странице объемность и улучшает визуальное восприятие любой фотогалереи



Каждое изображение отображается со своей собственной тенью, и вам не пришлось даже запускать Photoshop!

Законченную версию веб-страницы этой обучающей программы вы сможете найти в папке `08_finished\gallery_ex` учебного материала.

## Обучающий урок 3: использование фоновых изображений

CSS-свойство `background-image` — секретное оружие современного веб-дизайна. Оно превращает скучный текст в великолепный графический образ (рис. 8.23). Вы можете использовать его для добавления фоновых изображений к любому HTML-тегу. Дизайнер, который сумеет создать, ограничен только вашим воображением. Пример теневого эффекта, приведенного в предыдущем обучающем уроке, — всего лишь один из способов креативного использования фоновых изображений. Основное 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 данном обучающем уроке.

### Добавление на веб-страницу фонового изображения

Что бы это ни было — сложный замысловатый узор, логотип компании или полноэкранный фотография, — изображения очень часто используются в качестве фоновых рисунков веб-страниц. И не случайно, фактически это и есть основное применение свойства `background-image`.

1. Откройте файл `images.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 (см. рис. 8.23, *a*). 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;  
}
```

Первая строка стиля определяет само фоновое изображение `bg_page.png`, которое мы хотим вывести на веб-странице. Изображение находится в папке `images`. Оно представляет собой градиент, начинающийся светло-синим цветом сверху 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 появление этой неприятной картины, 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.

Возможно, вы подумали, что намного лучше было бы использовать не градиентное изображение, а линейный градиент. Да, в некоторых случаях лучше. Но, когда линейный градиент добавляется к фону страницы, веб-браузер должен перерисовать этот линейный градиент при каждом изменении размеров окна браузера, необоснованно снижая скорость вывода страницы и создавая впечатление о низкой скорости работы с ней. Кроме того, поскольку линейные градиенты в Internet Explorer 9 и более ранних версиях без дополнительного кода не работают, в данном случае мы остановимся на применении фонового изображения.

#### ПРИМЕЧАНИЕ

---

Используя сокращенный вариант свойства, вы можете превратить три строки кода, показанные в шаге 2, в одну строку:

```
background: url(images/bg_page.png) repeat-x #FFF;
```

---

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 украсить текст, задав для него другой фоновый цвет.

4. Вернитесь к текстовому редактору и файлу `bg_images.html`. Добавьте другой стиль для тега `<div>`, в котором находится содержимое страницы:

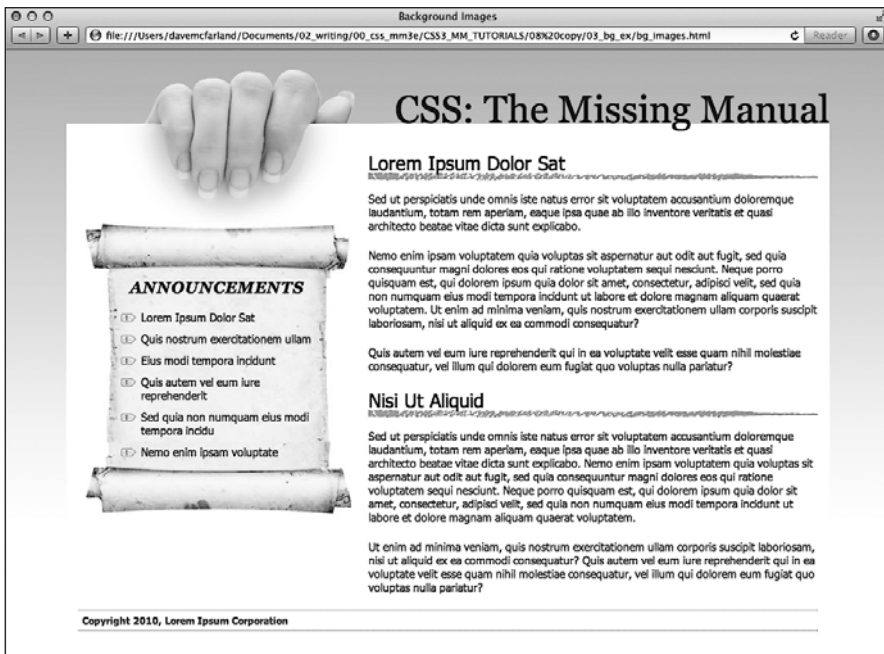
```
.wrapper {  
  background-color: #FFF;  
}
```

У данного тега `<div>` есть фиксированная ширина, он расположен посередине страницы и содержит весь ее текст. Этот стиль задает ему белый фоновый цвет, но с помощью изображения вы можете сделать его лучше.

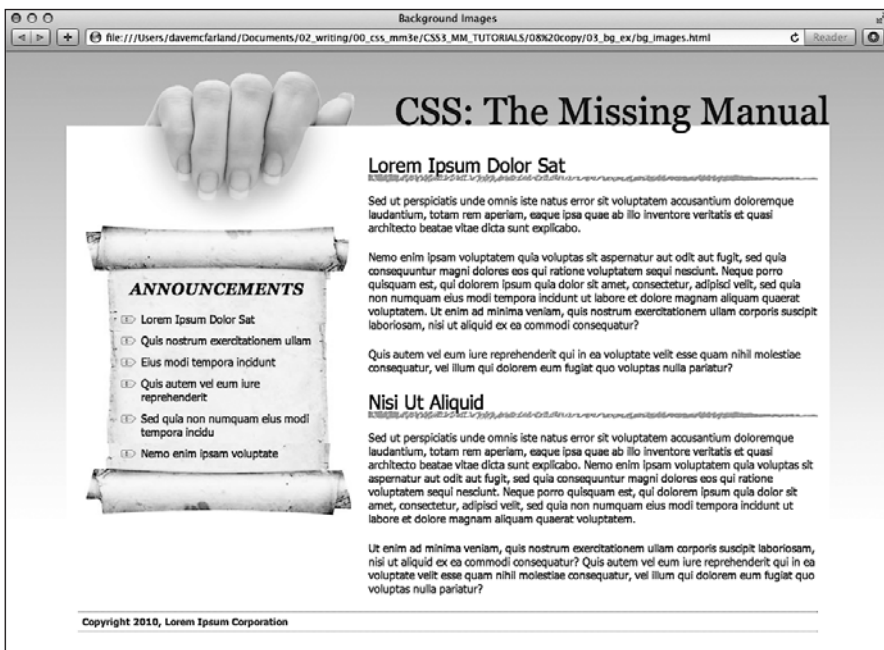
5. Отредактируйте стиль, который вы создали в шаге 4, добавив фоновое изображение:

```
.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 `<div>`; 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 будто держит эту страницу. Очень здорово. Но единственная проблема заключается в том, что текст находится слишком высоко сверху и прикрывает изображение. Далее вы разместите на странице большой заголовок и левую боковую панель.



a



b

Рис. 8.23. С помощью фоновых изображений вы можете превратить упорядоченную веб-страницу (a) в еще более привлекательный сайт (б)

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, который содержит 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. 8.24.

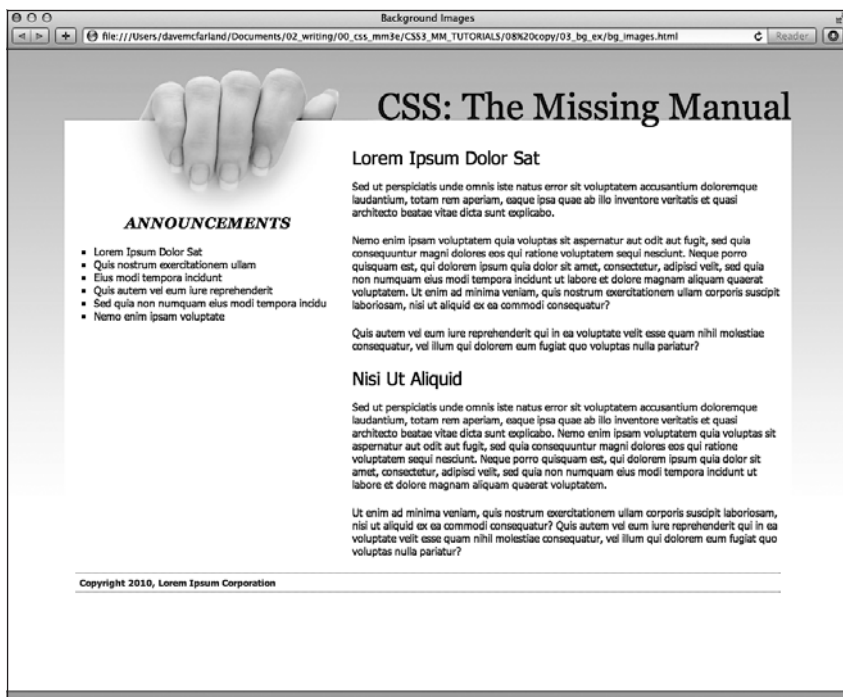


Рис. 8.24. CSS позволяет комбинировать цвет фона и фоновое изображение, что оказывается весьма полезным в настоящем примере

## Замена границ изображениями

Свойство `border` (см. раздел «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 фонового рисунка — это очень просто. В данном обучающем уроке мы заменим линии подчеркивания под тегами заголовков `<h2>` собственным рисунком, похожим на рукописную линию.

1. Вернитесь к файлу `bg_images.html` в текстовом редакторе. Добавьте стиль для тега `<h2>` внутри основного `<div>`:

```
.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 классом `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 пикселей 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 боковой панели, сделаем его менее угловатым и плоским, а также лучшим 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 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 `images.html` в HTML-редакторе. Добавьте стиль для форматирования списка пунктов левой боковой панели.

```
.announcement li {
  list-style: none;
}
```

Маркированный список находится внутри тега `<div>` с классом `announcement`, так что этот селектор потомка воздействует только на пункты списка (теги `<li>`) внутри этого `<div>`. Этот стиль удаляет маркеры. Теперь добавим наш рисунок.

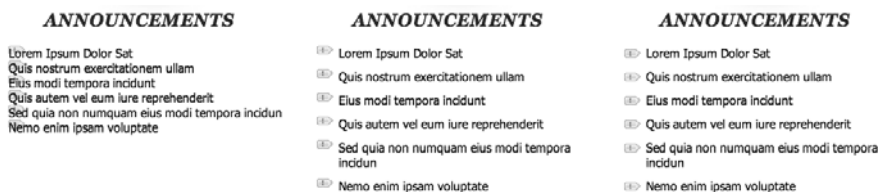
#### ПРИМЕЧАНИЕ

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. 8.25, *слева*). Небольшие отступы и поля исправят эту проблему.



**Рис. 8.25.** Заменить стандартные маркеры списков 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. 8.25, *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 помещает 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 оригинальные маркеры вместо скучных черных кружков (см. рис. 8.25, *справа*).

## Придание боковой панели характерных особенностей

На данный момент боковая панель выглядит довольно неплохо. Текст приятно отформатирован, маркеры отлично смотрятся, но сама боковая панель теряется на белом фоне. Добавление фонового изображения позволит ее заметно выделить.

Вы можете использовать одно изображение в виде свитка, показанное на рис. 8.23, в фоне тега `<div>`. А 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 (см. верхнее изображение слева на рис. 8.12), 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 (см. нижнее изображение справа на рис. 8.12). Хорошо, что эту маленькую хитрость не так сложно реализовать — вам понадобится три разных изображения и три разных стиля.

Как уже говорилось в разделе «Использование нескольких фоновых изображений» данной главы, существует несколько способов достижения данного эффекта. Можно сделать фоновую часть свитка (там, где появляется текст) в виде повторяющегося фонового изображения в `div`-элементе, затем поместить фоновое изображение для верхней части свитка в верхнюю часть `div`-элемента, а фоновое изображение нижней части свитка в нижнюю часть `div`-элемента. Фактически именно этот метод и использовался в предыдущих изданиях книги. Но сейчас большинство браузеров поддерживают появившуюся в CSS3 возможность объявления нескольких фоновых изображений, которой можно воспользоваться взамен старого метода. Получится намного проще, и не нужно будет создавать три разных стиля.

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» данного раздела) и добавьте одно свойство:

```
.announcement {
background: url(images/scroll_top.jpg) no-repeat center top,
           url(images/scroll_bottom.jpg) no-repeat center bottom,
```



```

        url(images/scroll_middle.jpg) repeat-y center top;
margin-top: 115px;
}

```

2. Обновите стиль `.announcement`, добавив верхний и нижний отступы. Внесите изменение, выделенное полужирным шрифтом:

```

.announcement {
background: url(images/scroll_top.jpg) no-repeat center top,
           url(images/scroll_bottom.jpg) no-repeat center bottom,
           url(images/scroll_middle.jpg) repeat-y center top;
padding: 70px 0 60px 0;
margin-top: 115px;
}

```

Еще одна проблема проявляется в том, что маркированный список выдается как с левой, так и с правой стороны изображения боковой панели. Сделаем так, чтобы он соответствовал этому изображению, добавив немного левых и правых полей.

3. 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.

4. Сохраните файл и просмотрите его в браузере.

Страница должна выглядеть как правое изображение на рис 8.23 на всех браузерах, кроме Internet Explorer 8. Этот браузер несколько фоновых изображений не поддерживает.

## Получение свитка, появляющегося в Internet Explorer 8

Хотя Internet Explorer 8 не поддерживает несколько фоновых изображений, такую поддержку можно имитировать, особенно когда нужно только два или три фоновых изображения. Используя сгенерированное содержимое (см. раздел «Различные свойства» в приложении 1), можно добавить текст или изображения перед и после любого элемента на странице. В данном случае верхнюю часть свитка можно добавить перед `div`-элементом, а его нижнюю — после этого элемента.

Если поддержка Internet Explorer 8 вас особо не волнует, этот шаг можно пропустить, но если вам хочется посмотреть, как эта работа делается в Internet Explorer 8 и в последующих версиях, а также в других текущих браузерах, читайте дальше.

1. Отредактируйте стиль `.announcement`, оставив только одно фоновое изображение (среднюю часть свитка) и убрав отступы:

```
.announcement {  
  background: url(images/scroll_middle.jpg) repeat-y center top;  
  margin-top: 115px;  
}
```

Отступы не понадобятся, потому что другие изображения займут пространство выше и ниже `div`-элемента. Затем будет добавлено верхнее изображение свитка в качестве элемента, предшествующего `div`.

2. Добавьте ниже стиля `.announcement` следующий стиль:

```
.announcement:before {  
  content: url(images/scroll_top.jpg);  
}
```

Как уже упоминалось выше, псевдоэлемент `:before` позволяет добавлять содержимое перед тегом. Это может быть текст или (как в данном случае) изображение. Теперь перейдем к нижней части свитка.

3. Добавьте ниже только что добавленного стиля `:before` следующий стиль:

```
.announcement:after {  
  content: url(images/scroll_bottom.jpg);  
}
```

Этот код добавит после `div`-элемента нижнюю часть свитка. Но в действительности это изображение установится не после `div`-элемента, а поверх него, и `div` расширится, чтобы его уместить. Если посмотреть на страницу, будет немного виден фон элемента `div` (свиток), показанный ниже скрученной нижней части свитка. Исправить положение поможет дополнительная небольшая магия CSS.

4. Отредактируйте только что созданный стиль, добавив в него еще две строки кода (выделенные полужирным шрифтом):

```
.announcement:after {  
  content: url(images/scroll_bottom.jpg);  
  position: relative;  
  bottom: -5px;  
}
```

Здесь мы немного забегаем вперед, поскольку о свойстве `position` не будет серьезных упоминаний вплоть до гл. 15. Но если коротко объяснять, вы смещаете нижнюю часть свитка на 5 пикселей вниз, чтобы прикрыть фоновое изображение `div`-элемента.

Теперь страница должна выглядеть как правое изображение на рис. 8.23 и быть похожа на ту, что создавалась в предыдущей части урока с помощью нескольких изображений. Обе страницы, `bg_images.html` и `bg_images_ie8.html` можно найти в папке `08_finished/03_bg_ex`.

# 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 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 распознают также псевдоклассы `:focus`. Ссылки получают фокус, когда посетители, недолюбливающие мышь, используют клавиатуру, чтобы перейти на ссылки с помощью клавиши табуляции. Этот псевдокласс также полезно применять с текстовыми полями формы, что будет показано в уроке по стилизации формы в гл. 11.

Предположим, вы хотите изменить цвет текста непосещенной ссылки с синего цвета 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, а затем отформатируйте тег `<a>`, создав общий селектор `a`. Далее вы сможете применять конкретные состояния ссылок, например `a:visited`, для изменения цвета или настройки какого-либо конкретного состояния.

### НАМЕК ДЛЯ СМЫШЛЕННЫХ

#### Ограничения на посещения

Из-за проблем конфиденциальности браузеры начали накладывать некоторые ограничения на CSS-свойства, применяемые к псевдоклассу `:visited`. Оказалось, что злоумышленники научились с помощью JavaScript считывать изменения в стиле `:visited` ссылок, чтобы определить, какие сайты посещались. Например, путем загрузки нового фонового изображения для ссылок на посещенные сайты можно определить, был ли посетитель на сайтах `Paypal.com`, `eBay.com`,

`BankofAmerica.com` и т. д. Из-за потенциальной возможности возникновения проблем наложены ограничения на стиливое изменение цвета, фоновое цвета и цвета рамки ссылок на посещенные сайты, исключая случаи заблаговременного придания этим ссылкам при их обычном состоянии цвета, фоновое цвета или цвета рамки. То есть теперь уже нельзя рассчитывать на внесение изменений с помощью псевдокласса `:visited`.

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. раздел «Стилизация ссылок» данной главы). Для начала изучим 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, над которым наведен указатель мыши, могут повлиять на другие элементы, находящиеся рядом. Например, если вы увеличиваете размер шрифта текстовой ссылки, над которой 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 миллисекунды, 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, 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** («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 заданным 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 выше, представляют собой простейшие стили ссылок `<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 желаете выделить (например, сайты ваших друзей, партнеров, спонсоров). Возможно, вы захотите их отформатировать таким обра-

зом, чтобы посетители заранее знали, что это какие-то особенные ссылки, и наверняка захотели бы перейти по ним. В данном случае можете применить стиливой класс:

```
<a href="http://www.hydroponicsonline.com" class="sponsor">Visit this great resource</a>
```

Для стилизации этой ссылки другим способом создадим следующий стиль:

```
a.sponsor { font-family: Arial, sans-serif; }
a.external:link { color: #F60; }
a.external:visited { color: #900; }
a.external:hover { color: #F33; }
a.external:active {color: #B2F511; }
```

Определения только имени стиливого класса, без указания названия тега а, будет тоже достаточно:

```
.sponsor { font-family: Arial, sans-serif; }
.external:link { color: #F60; }
.external:visited { color: #900; }
.external:hover { color: #F33; }
.external:active {color: #B2F511; }
```

Теперь такое форматирование приобретут лишь ссылки, принадлежащие стиливому классу 'external'.

#### ПРИМЕЧАНИЕ

В приведенных примерах в демонстративных целях я изменил только цвет ссылок. На практике для форматирования вы можете использовать любые CSS-свойства (за исключением ссылок, отвечающих псевдоклассу :visited, как объяснялось в предыдущей врезке «Намек для смысленых. Ограничения на посещения»). 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 в основные разделы вашего сайта.

Ссылки представляют собой главную панель навигации, и вы хотите придать им характерный вид. Просто заключите эти ссылки в HTML5-тег <nav>. Теперь появилась возможность произвести выборку и отформатировать только эти ссылки:

```
nav a { font-family: Arial, sans-serif; }
nav a:link { color: #F60; }
nav a:visited { color: #900; }
nav a:hover { color: #F33; }
nav a:active {color: #B2F511; }
```

Вместо этого можно воспользоваться тегом <div> (если вы еще не вполне готовы к применению HTML5-тегов) и добавить класс к этому div-тегу: <div class="mainNav">:

```
.mainNav a { font-family: Arial, sans-serif; }
.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 разделе «Использование селекторов потомков» гл. 17).

#### СОВЕТ

---

Применение 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 class="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 вашим воображением. Единственное, что нужно учесть, — 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 поймет, что они могут щелкнуть на ней. Можно также воспользоваться линейным градиентом для добавления к ссылке текстуры и глубины. Ссылки, расположенные в объемных фрагментах текста, должны четко выделяться. Этого можно добиться использованием текста полужирного начертания с сохранением традиционного подчеркивания, 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 визуальнo 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`, Internet Explorer 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;}

Естественно, полное удаление подчеркивания может смутить посетителей сайта. Если вы не предусмотрите каких-то 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.



**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 фоновое изображение (6)

- **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;
}
```

- **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.

Большого 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`.

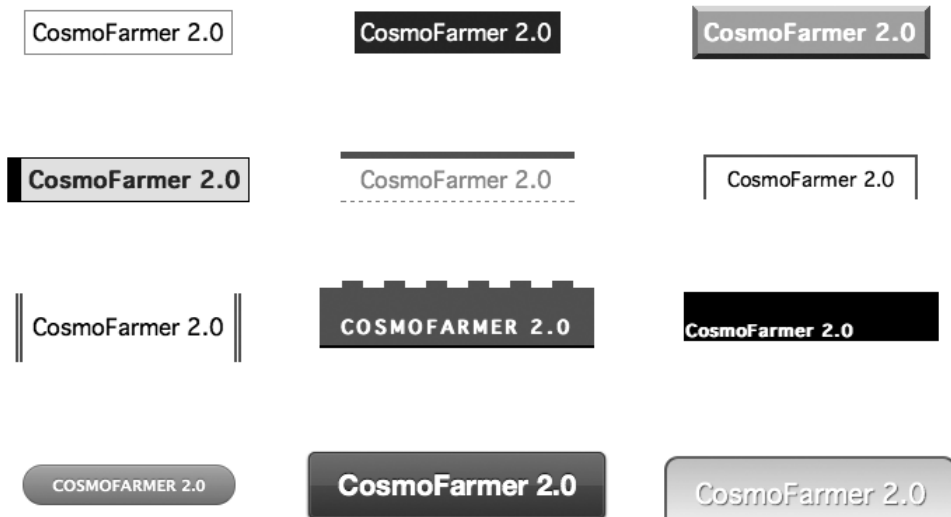
- **Использование фонового изображения.** Можете больше преобразить вид ссылок, добавив фоновый рисунок. Например, на рис. 9.1, 6 показана граница в виде рукописной линии.

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;
}
```

## 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 разнообразно-стей 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*), создайте 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.

Чтобы превратить ссылку в кнопку, на которой хочется щелкнуть, следует использовать свойства `border`, `background` и `padding`. Может оказаться достаточно простым добавление рамки (верхнее изображение слева на рис. 9.2). Или же можно создать более впечатляющие кнопки, сочетая различные рамки или варьируя цвета, стили и ширину в свойстве `background-color`. А для тех браузеров, которые поддерживают линейные градиенты, текстовые и отбрасываемые блочными элементами тени, а также скругленные углы рамок с указанием радиусов, можно добиться результатов, имеющих весьма привлекательный вид (нижний ряд на рис. 9.2).

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 на 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.

По-настоящему добавить глубину кнопке можно, используя линейные градиенты и скругленные углы, блочные и текстовые тени. Например, нижняя средняя кнопка на рис. 9.2 создана с помощью следующего кода:

1. `background-color: #ee432e;`
2. `background-image: -webkit-linear-gradient(top, #ee432e 0%, #c63929 50%, #b51700 50%, #891100 100%);`
3. `background-image: -moz-linear-gradient(top, #ee432e 0%, #c63929 50%, #b51700 50%, #891100 100%);`
4. `background-image: -o-linear-gradient(top, #ee432e 0%, #c63929 50%, #b51700`

- ```
50%, #891100 100%);
5. background-image: linear-gradient(top, #ee432e 0%, #c63929 50%, #b51700
50%, #891100 100%);
6. border: 1px solid #951100;
7. border-radius: 5px;
8. box-shadow: inset 0px 0px 0px 1px rgba(255, 115, 100, 0.4), 0 1px 3px
#333333;
9. padding: 12px 20px 14px 20px;
10. text-decoration: none;
11. color: #fff;
12. font: bold 20px/1 "helvetica neue", helvetica, arial, sans-serif;
13. text-align: center;
14. text-shadow: 0px -1px 1px rgba(0, 0, 0, 0.8);
```

В строке 1 устанавливается темно-красный цвет фона; этот цвет предоставляется для браузеров, не понимающих линейные градиенты (Internet Explorer 9 и более ранние версии, а также другие браузеры). В строках 2–5 задается линейный градиент для различных браузеров. В строке 6 добавляется простая рамка, а в строке 7 создаются скругленные углы, после чего в строке 8 добавляется тонкая внутренняя тень и вторая отбрасываемая тень ниже кнопки, то есть применяются две отбрасываемые тени, что вполне допустимо. В строке 9 добавляется отступ, обеспечивающий появление пустого пространства вокруг текста кнопки, а в строке 10 удаляется подчеркивание, которое обычно появляется под ссылками. В последних строках устанавливаются свойства для текста в кнопке, включая цвет, шрифт, выравнивание и текстовую тень.

#### СОВЕТ

---

Создание привлекательных кнопок часто требует целого пакета свойств CSS (что можно заключить из приведенного выше кода). Если в создании всего этого кода требуется помощь, обратитесь к генератору кнопок CSS3 по адресу <http://css3button.net>. Чтобы посмотреть на другие впечатляющие CSS3-кнопки, зайдите на сайты <http://hellohappy.org/css3-buttons/> и <http://webdesignerwall.com/tutorials/css3-gradient-buttons>.

---

## 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 совершенные методы применения изображений для создания графических кнопок и эффектов ролловеров<sup>1</sup> описаны в обучающем уроке этой главы).

---

<sup>1</sup> Ролловер — это элемент веб-страницы, изменяющий свой вид в зависимости от внешнего воздействия. — *Примеч. пер.*

### Normal Link

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

### No Repeating Images

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.


### Image in Button



### Image + Background



### External Link

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod laoreet dolore  kasdfklj dskfj ksajdf kljsad fkljs adfklj sadklfj lskadjf kljs dflkj sakdfljksjd

**Рис. 9.3.** Даже простой рисунок может оживить ссылку и сделать ее более понятной для восприятия: значок земного шара (внизу слева) — один из способов обозначения внешних ссылок

Чтобы вспомнить свойство `background-image` и все, что с ним связано, вернитесь к разделу «Добавление фоновых изображений» гл. 8. Пока же напомним 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, *вверху справа*). Если вы не используете едва заметный узор, похожий на градиентную заливку, не забудьте выключить повторное отображение фонового рисунка: `background-repeat: no-repeat;`
- **Управляйте позиционированием с помощью свойства `background-position`.** Чтобы точно разместить фоновое изображение, используйте свойство `background-position` (см. раздел «Позиционирование фоновых изображений» гл. 8). Если необходимо позиционировать изображение по правому краю ссылки, но при этом центрировать его вертикально на строке, то применяйте следующий CSS-код: `background-position: right center;`

Для еще более точного размещения изображений задействуйте 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 применять 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» раздела «Позиционирование фоновых изображений» гл. 8).

- **Добавляйте отступы с помощью свойства `padding`.** Если вы используете для выделения ссылки изображение или значок, нужно добавить отступ с той стороны, где рисунок примыкает к тексту. Например, в левом центральном фрагменте на рис. 9.3 ссылка имеет левый отступ шириной 30 пикселей для предотвращения наложения текста (слово Home) на изображение дома, в то же время в нижнем левом изображении небольшой правый отступ обеспечивает пространство для отделения текста ссылки от значков земного шара.

## ПРИМЕЧАНИЕ

Поскольку тег `<a>` является линейным элементом, добавление верхнего или нижнего отступов (или полей) не окажет никакого эффекта. Причина такого поведения описана в подразделе «Отображение линейных и блочных элементов» раздела «Управление размерами полей и отступов» гл. 7. Однако можно преобразовать ссылку в блочный элемент (`display: block`) или воспользоваться настройкой линейного элемента `inline-block` (`display: inline-block`), чтобы применить к нему верхние и нижние отступы и поля. Оба этих метода будут рассмотрены позже.

- **Пользуйтесь псевдоклассами.** Не забывайте о псевдоклассах `:hover` и `:visited`. Они помогут создать для ссылок замечательные динамические эффекты и обеспечить 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.

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 далее.

## 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 содержанию. Например, 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.



**Рис. 9.4.** Используя CSS, вы можете превратить обычные теги `<ul>` в то, что захочется, например в вертикальное или горизонтальное управляющее меню (панель навигации)

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). 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 подразделу.

## 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 выше), вы выполните большую часть работы. Но вы должны знать, что существует еще несколько дополнительных хитростей, обеспечивающих правильное отображение панели навигации.

1. **Отообразим ссылку в виде блочного элемента.** Поскольку тег `<a>` является линейным элементом, ссылка будет занимать по ширине пространство, равное ширине содержимого тега. Кнопки со ссылками в виде текста различной ширины (например, `Home` (Домой) и `Our Products` (Наши продукты)) 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 воздействия на линейные 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.



**Рис. 9.5.** Всего четырьмя простыми действиями можно превратить маркированный список ссылок в привлекательную панель навигации

- 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 боковую панель, примыкающую к левому или правому краю веб-страницы. Боковая панель имеет свою ширину, и помещенный в нее маркированный список со ссылками-кнопками автоматически приобретает эту ширину.

Теперь, когда мы закончили всю эту малопродуктивную 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 данной 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 подразделе описываются два распространенных подхода в создании из списка горизонтальной панели навигации. Первый метод основывается на использовании свойства `display: inline-block` — он прост, но обычно приводит к маленьким расстояниям между кнопками. Этот вопрос можно решить, но для этого придется возиться с кодом HTML. Если нужна горизонтальная навигационная панель с соприкасающимися кнопками, обратитесь к методу с применением плавающих элементов `<ul>`.

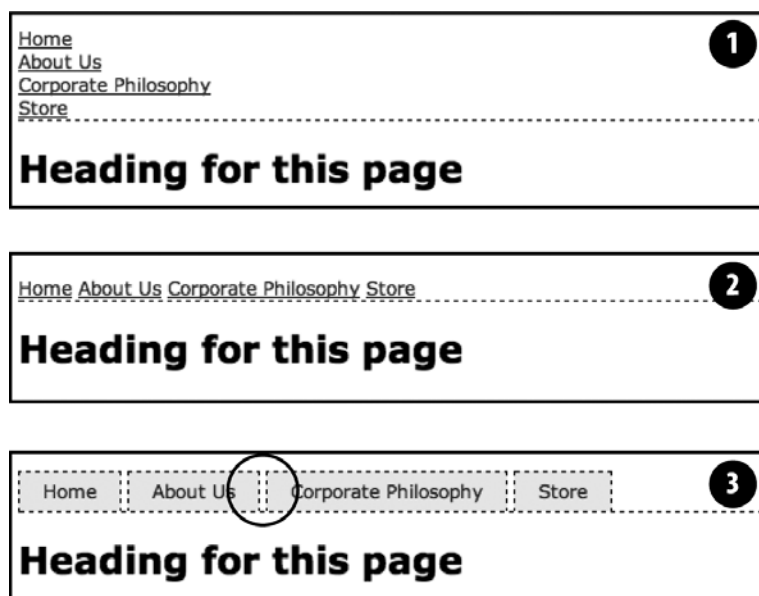
Какой бы метод вы ни использовали, сначала удалите маркеры и левые отступы тегов `<ul>`, как показано на рис. 9.6, 1.

**Использование `display:inline` и `display:inline-block`.** Самый простой метод создания горизонтальной панели навигации заключается в изменении для элементов списка значения `block` (блочный элемент) свойства `display` на `inline` (линейный элемент). Это делается средствами CSS.

1. Создаем стиль для маркированного списка, чтобы удалить отступы, поля и маркеры.

```
ul.nav {
  margin-left: 0px;
  padding-left: 0px;
  list-style: none;
  border-bottom: 1px dashed #000;
}
```

В данном случае добавляется нижняя граница, которая появится ниже кнопок (см. рис. 9.6, 1).



**Рис. 9.6.** Создание горизонтального меню из списка ссылок потребует выполнения всего нескольких действий. Но использование метода `inline` и `inline-block` приводит к заметным промежуткам между кнопками (обведено кружком)

2. Преобразуем пункты списка в линейные элементы. Линейные 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).

```
.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.

---

**ПРИМЕЧАНИЕ**

В показанном выше стиле можно также использовать `inline-block`, но только если не обращать внимания на Internet Explorer 7, который допускает применение `inline-block` только к таким линейным элементам, как ссылки, но не к таким блочным элементам, как тег `<li>`.

---

3. Стилизуем ссылки. Можно убрать подчеркивание под ссылками и вместо этого добавить вокруг них окаймляющую границу. Чтобы обеспечить визуальную глубину и реалистичность настоящих кнопок, нужно 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:

```
.nav a {
  display: inline-block;
  border: 1px dashed #000;
  border-bottom: none;
  padding: 5px 15px 5px 15px;
  background-color: #EAEAEA;
  text-decoration: none;
  color: #333;
}
```

Сначала нужно установить для свойства ссылок `display` значение `inline-block`. Тогда верхние и нижние отступы и поля будут соответствовать требованиям (как уже упоминалось, обычно линейные элементы игнорируют верхние и нижние отступы и поля, а также значения ширины и высоты). Затем можно будет указать стиль кнопок в соответствии с вашим вкусом. Здесь к ссылкам добавляется граница и удаляется нижняя граница, чтобы исключить дублирование нижней границы, применяемой к маркированному списку.

---

**СОВЕТ**

Определив ширину ссылок, все кнопки можно сделать одинакового размера.

---

Чтобы эта навигационная панель появлялась в центре страницы, к стилю тега `<ul>` нужно добавить объявление `text-align: center;`. У данной технологии, по сравнению с рассматриваемой далее плавающей технологией, есть одно преимущество — при использовании `inline` и `inline-block` панель навигации можно центрировать, чего нельзя сделать с `float`.

Однако можно заметить, что кнопки не соприкасаются (см. область, обозначенную кружком, на рис. 9.6). Это связано с тем, как веб-браузеры рассматривают пустое пространство между `<li>`-тегами. Возьмем, например, следующий фрагмент кода HTML:

```
<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>
```

Символ возврата каретки после одного тега `</li>` и перед следующим тегом `<li>` веб-браузеры рассматривают как пустое пространство. Удалить его можно одним из двух способов.

- Поместить закрывающий тег `</li>` и открывающий тег `<li>` на одной и той же строке:

```
<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>
```

Обычно так код не записывают, и программы типа Dreamweaver так бы не сделали. Чтобы удалить пустое пространство, нужно зайти в код и внести изменения вручную.

- Добавить к элементам списка правое поле с отрицательным значением. Например, в стиль `list` в показанном выше шаге 2 можно внести следующее изменение:

```
.nav li {
  display: inline;
  margin-right: -5px;
}
```

Отрицательное значение поля приводит к накладке следующего элемента списка на 5 пикселей, закрывая просвет между кнопками. Проблема при использовании данной технологии заключается в том, что конкретное применяемое значение варьируется в зависимости от размера текста — 5 пикселей могут сработать, а могут и нет, поэтому для получения нужного значения следует провести эксперимент.

## 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. Основное руковод-

ство по процессу создания можно найти по адресу <http://line25.com/tutorials/howto-create-a-pure-css-dropdown-menu>. Как создать изящное CSS-меню, применяющее CSS-переходы для появления и исчезновения раскрывающегося меню, можно узнать по адресу [www.red-team-design.com/css3-animated-dropdown-menu](http://www.red-team-design.com/css3-animated-dropdown-menu).

Если вы не хотите создавать меню самостоятельно или, возможно, просто ограничены во времени, то можете воспользоваться бесплатным генератором Pure CSS Menu — мастером, реализованным в виде



**ЧАСТО ЗАДАВАЕМЫЕ ВОПРОСЫ**

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, подмену исчезает мгновенно, без задержки, — остается надеяться, что у посетителей сайта достаточно хорошая реакция.

Есть и другой подход: задействуйте CSS-код для стилизации кнопок-ссылок и JavaScript-код для управления работой подмену.

В качестве очень простого меню JavaScript попробуйте воспользоваться дополнительным модулем jQuery Navigation, который находится по адресу <https://github.com/dansdom/plugins-navigation>.

**Используем плавающие элементы для горизонтальной навигационной панели.** Еще одной более популярной технологией являются плавающие элементы списка. Этот метод также позволяет поместить ссылки бок о бок. Кроме того, данная технология не страдает от проблемы пустого пространства, присущей методу inline.

**ПРИМЕЧАНИЕ**

Панели навигации на основе плавающих элементов трудно центрировать горизонтально посередине веб-страницы. Если требуется такое выравнивание, то лучше пользоваться методом с линейными 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:

```
.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 боковой панели, в которую заключены кнопки).

2. Добавим в стиль ссылок свойство `display: block`. Ссылки — это линейные 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.
4. Определим ширину. Если необходимо, чтобы все кнопки панели управления имели одинаковую ширину, установите параметры тега `<a>`. Точное значение ширины зависит от максимальной длины текста каждой из кнопок. Очевидно, что для ссылки *Corporate Philosophy* (Корпоративная философия) 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>`).

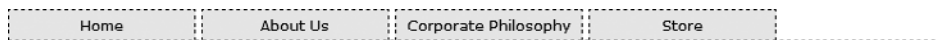
```
ul.nav {
  overflow: hidden;
}..
```

Ниже даны описания стилей, требуемых для создания панели навигации, изображенной на рис. 9.7. Обратите внимание, что кнопки имеют одинаковую ширину и текст на них центрирован.

```
..nav {
  margin-left: 0px;
  padding-left: 0px;
  list-style: none;
  border-bottom: 3px solid rgb(204,204,204);
  overflow: hidden;
}

.nav li {
  float: left;
}

.nav a {
  width: 12em;
  display: block;
  border: 3px solid rgb(204,204,204);
  border-bottom: none;
  border-radius: 5px 5px 0 0 ;
  padding: 10px;
  margin-right: 5px;
  background-color: rgb(95,95,95);
  background-image: -webkit-linear-gradient(rgb(175,175,175), rgb(95,95,95));
  background-image: -moz-linear-gradient(rgb(175,175,175), rgb(95,95,95));
  background-image: -o-linear-gradient(rgb(175,175,175), rgb(95,95,95));
  background-image: linear-gradient(rgb(175,175,175), rgb(95,95,95));
  text-decoration: none;
  color: white;
  text-align: center;
  font-family: Arial, Helvetice, sans-serif;
  font-weight: bold;
}
```



## Heading for this page

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, однако ваш сайт обязательно 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 также размещены инструменты, которые облегчат работу.

Подробная информация по превращению простых списков в необычные, неординарные элементы управ-

ления 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/>.

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).

## 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 браузер не загрузил 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-спрайтами (sprites), при котором используется

единственное изображение для создания различных состояний одной навигационной кнопки.

---

**ПРИМЕЧАНИЕ**

Эволюционировавший метод CSS-спрайтов 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 <http://coding.smashingmagazine.com/2009/04/27/the-mystery-of-css-sprites-techniques-tools-and-tutorials/>.

---

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-ролlover при наведении указателя мыши — снизу.

---

**ПРИМЕЧАНИЕ**

В Fireworks CS6 есть встроенное средство для создания CSS-спрайтов. Имеется также множество других вспомогательных интерактивных средств для их создания, например SpritePad (<http://spritepad.wearekiss.com>).

---

2. Измерим расстояние от верхнего края получившегося комбинированного изображения до верхней границы каждого последующего изображения. На рис. 9.8, *вверху*, верхняя граница сменного изображения-ролловера имеет смещение от верхнего края общего изображения в 39 пикселей.
3. Создадим CSS-стиль для ссылки в обычном ненажатом состоянии. Поместим изображение в качестве фонового в левый верхний угол стилизуемой ссылки (см. рис. 9.8, *посередине*).

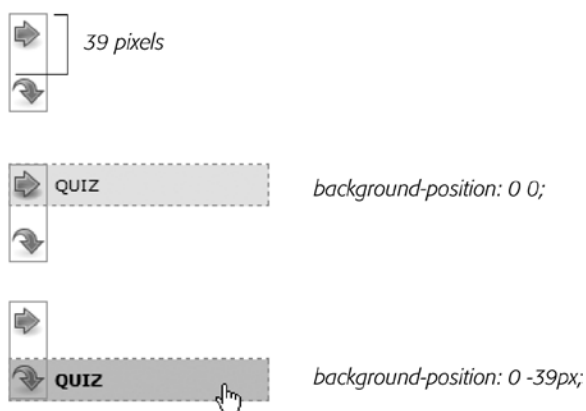
Стиль может выглядеть следующим образом:

```
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-ролlover окажется видимым (см. рис. 9.8, *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.



**Рис. 9.8.** Используя метод CSS-спрайтов, можно избежать задержки во время загрузки браузером изображения-роллера в первый раз, включив все изображения различных состояний кнопки-ссылки в единственный файл рисунка

#### ПРИМЕЧАНИЕ

Некоторые сайты пошли в использовании этого метода еще дальше. Yahoo!, Amazon и Google (а также многие другие) 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](http://www.flickr.com/photos/mezzoblue/3217540317). Сайты с миллионами посетителей в день активно используют сложные спрайты, поскольку веб-серверу проще и быстрее отправить один (хотя и больший по размеру) файл изображения, чем десяток меньших по размеру файлов.

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-v2.duoh.com/blog/comments/the\\_xhtml\\_css\\_template\\_phase\\_of\\_my\\_new\\_blog\\_part\\_2/](http://veerle-v2.duoh.com/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.

## 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.

## 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/css3/">` 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']`. Если вы хотите, например, при сборке всех элементов добавить значок с земным шаром рядом с внешними, но не внутренними ссылками вашего сайта, создайте два следующих 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 указывающих 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 Internet Explorer 8 и более ранних версий не понимает селектор `:not()`.

---

## 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 применяется 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, форматизирующего 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, заставляют 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 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 дает указание выбрать все ссылки, значение атрибута `href` которых заканчивается на `.pdf`. Вы можете создать аналогичные стили и для других типов файлов:

```
a[href$='.zip'] /* ZIP-архив */  
a[href$='.doc'] /* Word-документ */
```

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.

1. Запустите браузер и откройте файл веб-страницы `links.html` в папке `09\links`.

Эта страница содержит множество ссылок (обведены на рис. 9.9), которые указывают на другие веб-страницы текущего или иных сайтов, а также обозначают адрес электронной почты. Сначала изменим цвет ссылок данной страницы.

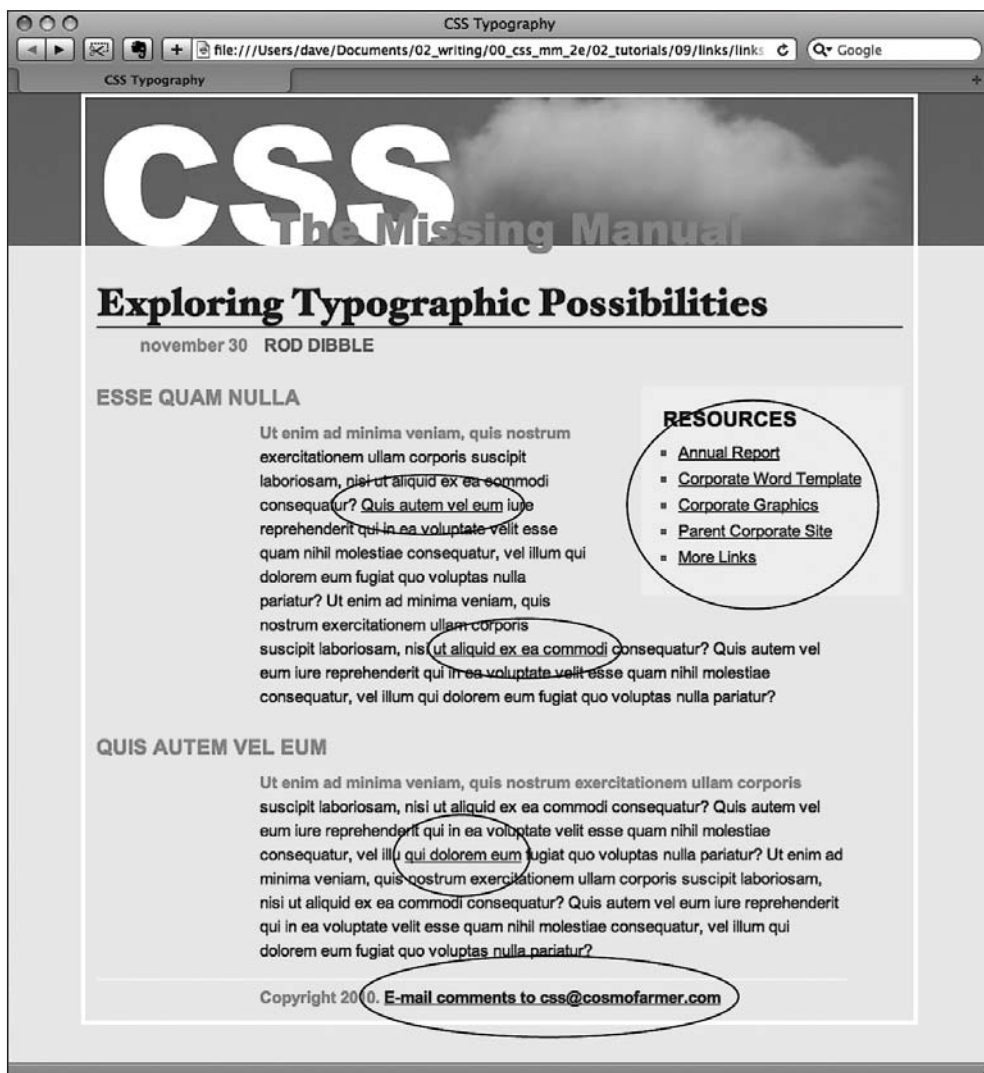


Рис. 9.9. Простейшая веб-страница со стандартной для браузеров стилизацией

- Откройте файл `links.html` в HTML-редакторе и поместите указатель между открывающим и закрывающим тегами `<style>`.

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, чтобы было легче писать код и просматривать страницу. Когда все будет готово, желательно переместить стили во внешнюю таблицу стилей.

---

- 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.

- 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 выделяться 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. 9.9 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.

- 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.

- Добавьте свойство `border`, при этом стиль должен выглядеть следующим образом:

```
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. 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-ролlover `hover` (наведения), чтобы фон изменял цвет при наведении на ссылку указателя мыши.

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-роллвера позволяет посетителям узнать, что выполняется какое-то действие (рис. 9.10).



**Рис. 9.10.** 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

## Добавление для ссылки фонового изображения

Ссылка на адрес электронной почты внизу веб-страницы ничем не отличается от других ссылок на странице (рис. 9.11, *вверху*). Поскольку `mailto` указывает на адрес электронной почты, при щелчке на ней кнопкой мыши посетитель не пе-

рейдет на другую веб-страницу, а запустит почтовую программу. Чтобы обеспечить визуальное выделение этой ссылки, добавим небольшой значок почтового конверта.



**Рис. 9.11.** 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 потомка:

```
a[href^="mailto:"] {
  color: #666666;
  border: none;
  background: url(images/email.gif) no-repeat left center;
}
```

Здесь используется расширенный селектор атрибута, выбирающий любые ссылки, начинающиеся с `mailto:` (то есть он выбирает ссылки на адреса 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. 9.11, *посередине*).

2. Добавьте в только что созданный стиль атрибута левый отступ размером 20 пикселей:

```
padding-left: 20px;
```

Помните, что отступ добавляет промежуток между содержимым и границей элемента, поэтому установка небольшого левого отступа смещает текст ссылки на 20 пикселей, 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 пикселей. Должен получиться следующий окончательный вариант:

```
a[href^="mailto:"] {
  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. 9.11, *внизу*). Таким образом, ссылка воспринимается посетителем как сочетание значка с текстом.

## 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 мы работаем, правосторонняя боковая панель **Resources** (Ресурсы) содержит различные типы ссылок, которые вы выделите значками, используя отдельный значок для каждого типа. Для начала настроим 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 классом `resources`, селектор потомка `.resources a` воздействует только на эти ссылки. Этот стиль избавляется от подчеркивания, которое было добавлено в общем стиле ранее.

Далее добавим значок рядом с внешними ссылками.

2. Добавьте другой стиль в конец внутренней таблицы стилей веб-страницы `links.html`:

```
.resources a[href^='http://'] {
  background: url(images/globe.png) no-repeat right top;
}
```

В этом стиле селектора потомка используется расширенный селектор атрибута. В основном он воздействует на любую ссылку, которая начинается с `http://` (но только на ту, которая находится внутри элемента с классом `resources`). Как и стиль ссылки на адрес электронной почты, который вы создали ранее, этот стиль добавляет фоновое изображение. Он помещает изображение на правую сторону ссылки.

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.

- 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.

- Добавьте еще три стиля во внутренней таблице стилей:

```
.resources a[href$='.pdf'] {
  background: url(images/acrobat.png) no-repeat right top;
}

.resources a[href$='.zip'] {
  background: url(images/zip.png) no-repeat right top;
}

.resources a[href$='.doc'] {
  background: url(images/word.png) no-repeat right top;
}
```

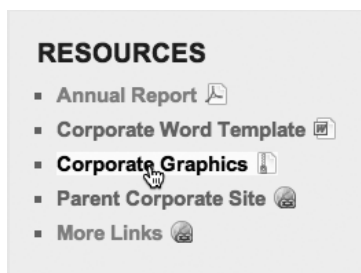
Эти три новых стиля проверяют, как заканчивается атрибут `href`, идентифицируют ссылки как файлы Adobe Acrobat (`.pdf`), ZIP-архивы (`.zip`) или документы Word (`.doc`) и назначают различные значки в каждом конкретном случае.

- Наконец, добавьте состояние `hover` (наведения указателя мыши) для ссылок на ресурсы:

```
.resources a:hover {
```

```
color: #000;
background-color: rgba(255,255,255,.8);
}
```

Этот стиль изменяет цвет текста и добавляет цвет фона (рис. 9.12).



**Рис. 9.12.** Используя расширенные селекторы атрибутов, вы можете легко идентифицировать и стилизовать различные типы ссылок веб-страницы

Окончательную версию этого урока вы найдете в файле 09\_finished/links/links.html.

## Обучающий урок 2: создание панели навигации

На этом практическом занятии мы превратим простой список ссылок во впечатляющую панель навигации с эффектом-ролloverом 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. 9.13, 1. Первым шагом будет добавление HTML-кода, обеспечивающего целенаправленное форматирование ссылок списка средствами CSS.

2. Найдите открывающий тег `<ul>` и добавьте в него `class = "mainNav"`, чтобы он выглядел следующим образом:

```
<ul class="mainNav">
```

Атрибут `class` назначает этот список главной областью навигации, или панелью управления. Мы будем использовать его для создания селекторов потомков, избирательно формирующих только ссылки этого списка (а не все ссылки веб-страницы).

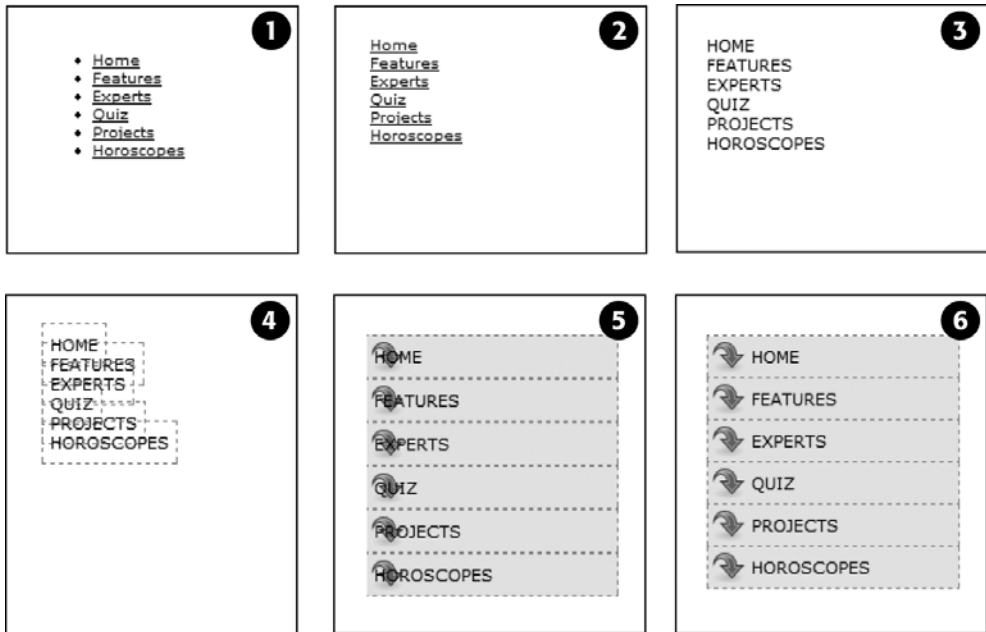
3. Добавьте во внутреннюю таблицу после `body` новый стиль:

```
.mainNav {
margin: 0;
padding: 0;
```



```
list-style: none;
}
```

Этот стиль будет применен лишь к тегу `<ul>` с классом `mainNav`. Он удаляет отступ и маркеры, которые браузер добавляет в маркированные списки, как показано на рис. 9.13, 2. Теперь приступим к форматированию ссылок.



**Рис. 9.13.** 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

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. 9.13, 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. 9.13, 4): границы перекрываются между собой, а ссылки имеют различную ширину. Это происходит потому, что тег `<a>` является линейным элементом

и ширина ссылки равна длине ее текста. Кроме того, верхний и нижний отступы не обеспечивают увеличения высоты линейным элементам, поэтому границы накладываются друг на друга (о линейных элементах читайте в разделе «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.

7. Во внутренней таблице найдите стиль `.mainNav` и добавьте в него свойство `width: 175px;`.

```
.mainNav {
  margin: 0;
  padding: 0;
  list-style: none;
  width: 175px;
}
```

При ширине списка 175 пикселей 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, боковая панель), для которого уже определена ширина, поэтому можете пропустить этот шаг (о том, как создавать боковые панели, 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. 9.13, 5). Здесь тоже нужно кое-что исправить: текст ссылки накладывается на значок, а линии границ между кнопками имеют толщину, равную 2 пикселям (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 `.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. 9.13, б).

## Добавление ролловеров и выделение текущего раздела сайта стилем выбранных ссылок

Пришло время усовершенствовать панель навигации и придать ей некоторую интерактивность. Сначала обеспечим кнопкам главной панели управления эффект ролловера. Они должны изменять свой вид, акцентируя внимание посетителя на том, какую кнопку он собирается нажать.

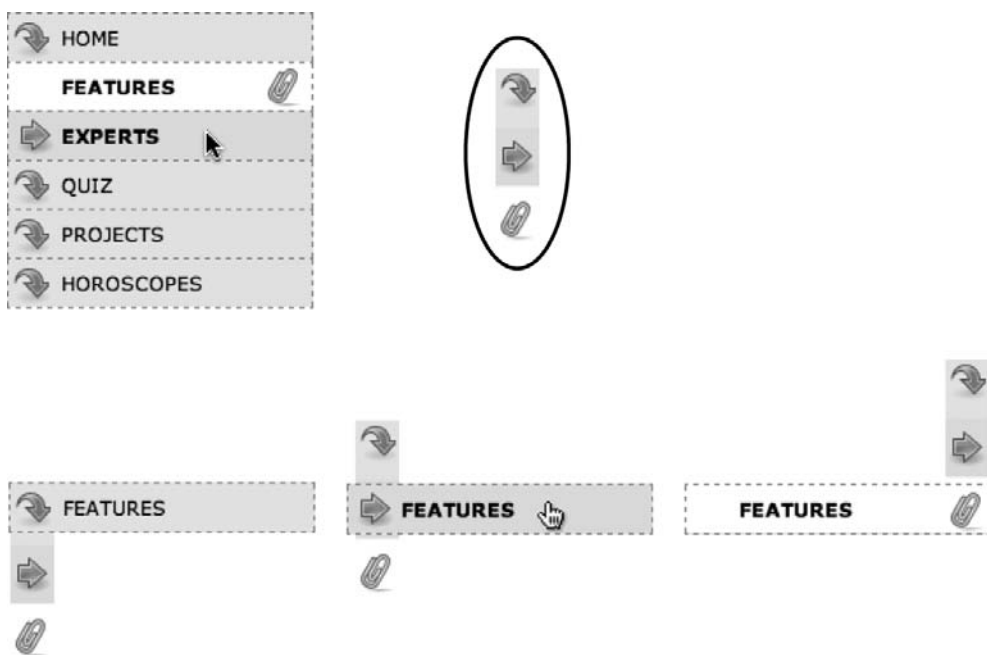
Неплохо было бы дать посетителю знать, в каком разделе (на какой странице) сайта он находится. Мы можем обеспечить созданной панели навигации автоматическую интерактивность. Она будет просто изменять свой стиль в соответствии с выбранным разделом страницы. Звучит совсем просто, но в действительности это потребует некоторой разметки и настроек, как вы увидите в следующих 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-спрайтов, упоминавшийся ранее. То же самое изображение применяется в шаге 8 в прошлом разделе — оно в действительности содержит три различных значка (рис. 9.14). В этом случае изображение центрируется внутри кнопки, отображая средний значок файла.



**Рис. 9.14.** С помощью несложного CSS-кода можно создать интерактивный эффект ролловера для кнопок панели навигации, а также автоматически подсветить раздел сайта текущей страницы

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).

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="current">Horoscopes</a>`.

2. Найдите тег `<body>` и добавьте в него класс `class="home"`:

```
<body class="home">
```

Теперь, когда мы знаем, какому разделу сайта принадлежит текущая открытая страница, можно использовать селектор потомка для создания конкретного CSS-стиля, который будет применен к веб-странице раздела **Features** (Особенности). Далее мы должны маркировать разделы, на которые указывает каждая ссылка-кнопка, что достигается путем добавления классов.

3. Найдите в HTML-коде панели навигации ссылку **Home** (Домой) и добавьте в нее класс `class="homeLink"`, чтобы тег выглядел следующим образом:

```
<a href="/index.html" class="homeLink">Home</a>
```

Этот класс однозначно определяет ссылку, предоставляя возможность создания стиля, который будет применен только к ней.

Естественно, нам потребуется добавить классы для всех ссылок панели навигации.

4. Повторите шаг 3 для каждой из ссылок, используя следующие классы: `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** (Особенности).

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. Мы снова используем метод CSS Styles 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. 9.14). 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 классом `homeLink`, которая также заключена внутри тега `<body>` с классом `home`. Если вы измените класс страницы, например, на `quiz`, с кнопки-ссылки раздела **Home (Домой)** исчезнет выделение.

Просмотрите веб-страницу в браузере, чтобы увидеть результат: теперь ссылка **Home (Домой)** 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 (Викторина)** приобретет такое же форматирование, каким была выделена ссылка раздела **Features (Особенности)**.

Теперь пришло время проверить результаты работы.

#### ПРИМЕЧАНИЕ

---

Этот длинный селектор является примером группового селектора, который мы обсуждали в гл. 3.

#### 7. Измените атрибут `class` тега `<body>` на `feature` следующим образом:

```
<body class="feature">
```

Воспользовавшись предварительным просмотром веб-страницы, вы увидите, что ссылка **Features (Особенности)** теперь выделена белым цветом фона и значком скрепки (см. рис. 9.14). Весь секрет в том, что нужно изменить атрибут `class` тега `<body>`, чтобы указать, к какому разделу сайта принадлежит страница. Например, для страницы гороскопа измените атрибут класса `<body>` на `class="horoscope"`.

#### ПРИМЕЧАНИЕ

---

Готовы продолжить стилизацию дальше? Попробуйте 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 (Домой)** (можете использовать файл `home.png` в папке с рисунками `images`).

---

Полную версию данной навигационной панели можно увидеть в файле `09_finished\nav_bar\nav_bar_vertical.html`.

## Переход от вертикальной панели навигации к горизонтальной

Предположим, вы хотите создать горизонтальную панель навигации в верхней части веб-страницы. Никаких проблем — большую 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. 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 раскрываю 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.

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. 9.15, 1). Нижняя граница, созданная нами в шаге 10 выше, 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 размещены рядом друг с другом, их левые и правые границы сливаются между собой, образуя линии толщиной 2 пиксела. Сейчас мы решим эти проблемы.





**Рис. 9.15.** 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` а 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. 9.15, 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. 9.15, 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. 13).

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;
}
```

Код `overflow: hidden` заставляет расширяться неупорядоченный список. Почему свойство работает? Об этом вы узнаете из гл. 12. Наконец, значок скрепки, выровненный по правому краю нажатой кнопки **Quiz**, выглядит неплохо (см. рис. 9.15, 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 и замечательно выглядит (см. рис. 9.15, 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 Осуществление преобразований, переходов и анимации с помощью CSS

За весьма короткую историю Интернета у разработчиков было несколько вариантов добавления анимации к своим веб-сайтам. Самая простая и примитивная анимация в рамках одного изображения предоставлялась в GIF-формате. Программы Adobe Flash позволяли создавать сложную анимацию и даже игры и веб-приложения, но изучить эту технологию весьма непросто, и она не позволяет взаимодействовать с другим HTML-кодом на вашей странице, представляющим изображения, заголовки и абзацы, из которых составлена основа веб-содержимого. JavaScript дает возможность анимировать все, что имеется на веб-странице, но ценой изучения всех тонкостей этого языка программирования.

К счастью, CSS3 предоставляет способ перемещения, преобразования и анимации любого HTML-элемента, имеющегося на странице без обращения к любой из других упомянутых здесь технологий.

## Преобразования

В CSS3 представлены несколько свойств, связанных с преобразованиями элемента веб-страницы, будь то вращение, масштабирование, перемещение этого элемента или его перекашивание вдоль горизонтальной или вертикальной оси (которое называется наклоном). Преобразование можно использовать, например, для придания небольшого наклона (вращения) панели навигации или для укрупнения изображения вдвое при проходе над ним указателя мыши. Можно даже сочетать несколько преобразований для получения весьма впечатляющих визуальных эффектов.

Основным CSS-свойством для получения любого из этих изменений является `transform`. Оно используется с предоставлением типа желаемого преобразования и добавлением значения, указывающего на степень преобразования элемента. Например, для вращения элемента предоставляется ключевое слово `rotate`, за которым следует количество градусов поворота:

```
transform: rotate(10deg);
```

Показанное выше объявление приведет к вращению элемента на  $10^\circ$  по часовой стрелке.

CSS-преобразования появились относительно недавно, и поэтому они поддерживаются не всеми браузерами. Двухмерные (2D) преобразования поддерживают Internet Explorer 9, Safari, Chrome, Firefox и Opera, но не поддерживают Internet Explorer 8 и более ранние версии. Более того, для всех широко используемых на данный момент браузеров нужно применять префиксы производителей. Поэтому, чтобы показанный выше код заработал, его нужно переписать следующим образом:

```
-webkit-transform: rotate(10deg);  
-moz-transform: rotate(10deg);  
-o-transform: rotate(10deg);  
-ms-transform: rotate(10deg);  
transform: rotate(10deg);
```

#### ПРИМЕЧАНИЕ

---

Ранее в данной книге уже встречались некоторые другие свойства CSS3, требующие использования префиксов производителей, например `-webkit-linear-gradient`. Но до сих пор не было префикса вида `-ms-`, который предназначался бы для продукции компании Microsoft и был бы нацелен на использование Internet Explorer. Internet Explorer 9 не понимает многие из свойств CSS3, поэтому для него обычно не требуется никаких префиксов производителя. Internet Explorer 10 (который, возможно, станет доступен ко времени чтения данной книги) не собирается применять префикс производителя для большинства свойств, рассматриваемых в книге. А Internet Explorer 9 понимает свойство `transform`, поэтому для работы данного свойства в этом браузере нужно воспользоваться свойством `-ms-transform`.

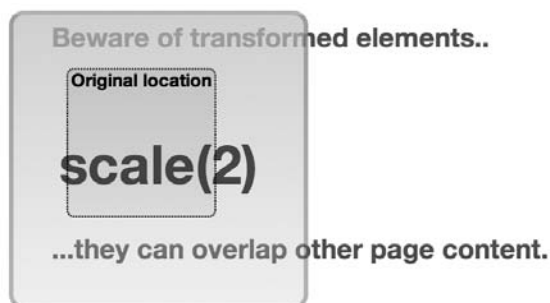
---

У CSS-преобразований есть одна странность: они не касаются окружающих элементов. То есть, если повернуть элемент на  $45^\circ$ , он может наложиться на те элементы, которые находятся выше, ниже его или по бокам. Сначала веб-браузеры выделяют элементу то пространство, которое он занимал бы при обычных обстоятельствах (до преобразования), а затем они занимаются преобразованием элемента (его вращением, увеличением или наклоном). Этот процесс становится очевидным при увеличении размеров элемента путем использования функции преобразования `scale` (см. далее подраздел «Масштабирование»). При увеличении размеров элемента в два раза браузер увеличивает преобразуемый элемент, но при этом не убирает с пути окружающее его содержимое, что обычно приводит к частичному перекрытию содержимого страницы (рис. 10.1). То есть веб-браузер сохраняет все остальные части страницы в том виде, в котором они бы появились, если бы элемент не был увеличен. Показанное на рис. 10.1 увеличение `<div>`-элемента (большой квадрат) приводит к перекрытию им заголовков выше и ниже этого элемента. Квадрат посередине, выделенный точками, представляет `<div>`-элемент до его увеличения в два раза.

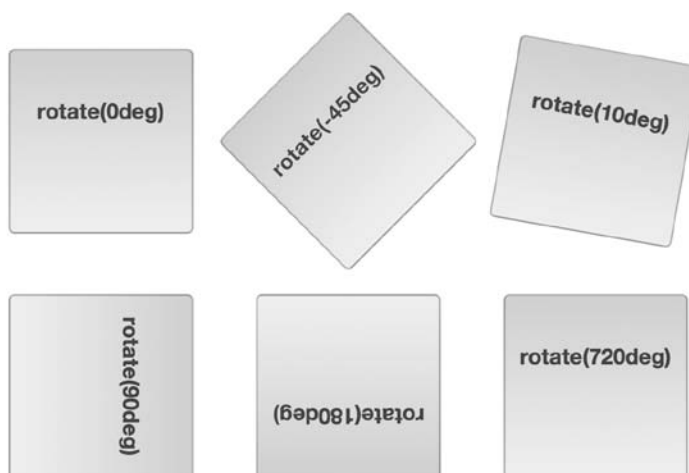
## Вращение

Разобраться в работе функции `rotate` свойства `transform` довольно просто: ей предоставляется угол от  $0$  до  $360^\circ$ , а браузер вращает указанный элемент по кругу на указанное количество градусов (рис. 10.2). Чтобы задать значение угла, используется число, за которым следует сокращение `deg`. Например, для вращения элемента на  $180^\circ$ , добавьте следующее объявление:

```
transform: rotate(180deg);
```



**Рис. 10.1.** Трансформируемые элементы просто игнорируются другими окружающими их элементами



**Рис. 10.2.** Функция rotate позволяет заставить любой элемент страницы — div, button, banner, photo или footer — вращаться либо на небольшой, либо на совершенно немыслимый угол

#### ПРИМЕЧАНИЕ

Для экономии пространства в приводимые здесь примеры не включаются префиксы производителей, но при помещении данного кода в таблицу стилей нужно добавлять к нему также свойства `-webkit-transform`, `-moz-transform`, `-o-transform` и `-ms-transform`.

Для вращения элемента против часовой стрелки можно применять отрицательные числовые значения. Например, средний элемент в верхней части рис. 10.2 повернут на  $45^\circ$  против часовой стрелки благодаря следующему объявлению:

```
transform: rotate(-45deg);
```

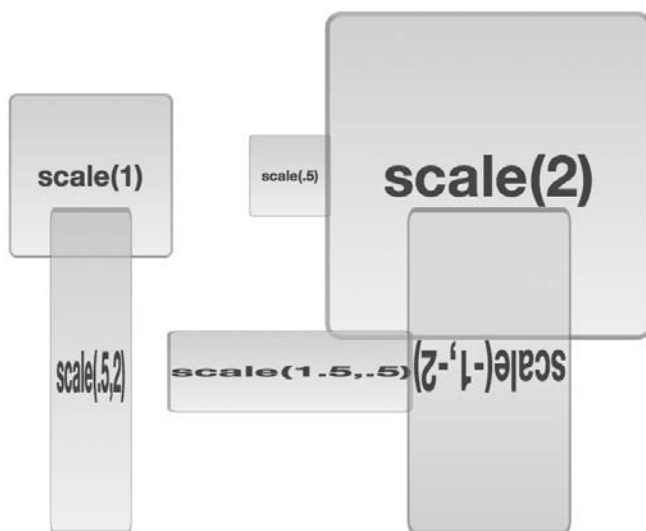
Значение `0deg` не придает никакого вращения, значение `360deg` придает одно вращение на полный оборот, а значение `720deg` — два полных вращения. Разумеется, внешний вид элемента, которому было придано одинарное, двойное или тройное вращение, останется таким же, как и у элемента, не подвергавшегося никакому вращению (верхнее левое и нижнее правое изображения на рис. 10.2), поэтому использование

значения с числом 360 и кратным ему может вызвать удивление. В CSS3 предоставляется механизм анимирования изменений в CSS-свойствах. Так, например, можно заставить кнопку прокрутиться четыре раза, когда указатель мыши пользователя проходит над ней, путем начального вращения на 0deg и добавления стиля `:hover` для этой кнопки с вращением 1440deg. Как это делается, будет вскоре показано.

## Масштабирование

Элемент можно также увеличить или уменьшить в размерах, воспользовавшись для этого функцией `scale` (рис. 10.3). Например, чтобы увеличить элемент вдвое, нужно добавить следующее объявление:

```
transform: scale(2);
```



**Рис. 10.3.** Для увеличения размера любого элемента на странице используется функция `scale`

Следует напомнить, что при увеличении элемента браузер не убирает другие элементы с его пути. Фактически при увеличении элемента он, скорее всего, перекроет другие элементы (посмотрите, к примеру, на правый верхний квадрат на рис. 10.3).

Число, представляемое в скобках, является коэффициентом масштабирования — числом, на которое умножаются текущие размеры элемента. Например, 1 говорит об отсутствии масштабирования, .5 указывает на половину текущего размера, а 4 — на учетверение текущего размера. То есть числа между 0 и 1 приводят к уменьшению, а числа больше 1 — к увеличению элемента (значение 0 фактически делает элемент невидимым на странице).

На это число происходит масштабирование элемента и всего, что в нем находится. Например, если масштабировать `<div>`-контейнер с коэффициентом 2, то вдвое шире и выше станет не только сам контейнер, но также и текст внутри него. Это же касается и находящихся внутри изображений.

Конечно, можно удивиться: а зачем вообще все это нужно. В конце концов, можно ведь просто увеличить или уменьшить ширину и высоту элемента, используя CSS-свойства `width` и `height`, а размер текста можно увеличивать или уменьшать с помощью свойства `font-size`. Чаще всего масштабирование используется для визуальных изменений элемента на странице в динамическом режиме. Например, проход указателя мыши над кнопкой может тут же привести к увеличению размера этой кнопки. Этого эффекта можно добиться с помощью состояния `:hover`.

Предположим, например, что на странице есть ссылка, к которой применен класс `.button`. Для форматирования этой ссылки в виде кнопки можно создать следующий простой стиль:

```
.button {
  font: .9em Arial, Helvetica, sans-serif;
  border-radius: .5em;
  background-color: rgb(34,255,23);
  border: 1px solid rgba(0,0,0,.5);
  padding: .5em;
}
```

Чтобы выделить это кнопку, можно сделать ее немного больше при проходе над ней указателя мыши посетителя:

```
.button:hover {
  -webkit-transform: scale(1.2);
  -moz-transform: scale(1.2);
  -o-transform: scale(1.2);
  -ms-transform: scale(1.2);
  transform: scale(1.2);
}
```

Когда указатель выходит за пределы кнопки, преобразование удаляется и кнопка возвращается к своему обычному размеру. Способ анимирования этого изменения размеров с помощью CSS-преобразований будет рассмотрен в подразделе «Переходы».

## СОВЕТ

---

Подобный замысел можно применить для изображений. Показать галерею из небольших изображений, а затем добавить к этим изображениям стиль `:hover`, чтобы при проходе над ними указателя мыши посетителя они становились больше. Чтобы все это выглядело вполне достойно, в HTML вставляется финальная, увеличенная версия изображения, но его размер уменьшается либо с помощью CSS, либо с использованием свойств `width` и `height` тега `<img>`.

---

Можно даже проводить раздельное горизонтальное и вертикальное масштабирование. Для этого внутри скобок нужно представить два значения, разделенных запятой. Первое число будет относиться к горизонтальному, а второе — к вертикальному масштабированию. Например, чтобы сделать элемент вдвое меньше по ширине, но вдвое выше, используется следующее объявление:

```
transform: scale(.5,2);
```

Эффект можно увидеть в левом нижнем изображении рис. 10.3.

В CSS3 также предоставлены отдельные функции для горизонтального и вертикального масштабирования: `scaleX` проводит масштабирование по горизонталь-

ной оси, а `scaleY` — по вертикальной. Например, чтобы элемент стал вдвое выше без изменения его ширины, можно воспользоваться следующим объявлением:

```
transform: scaleY(2);
```

#### ПРИМЕЧАНИЕ

Для экономии пространства в этом примере опять показана только лишь версия свойства `transform`, в которой префикс не используется. Чтобы объявление работало во всех браузерах, нужно добавить соответствующие префиксы: `-webkit`, `-moz` и т. д.

Однако чтобы элемент стал в три с половиной раза шире, а не выше или ниже, следует воспользоваться объявлением:

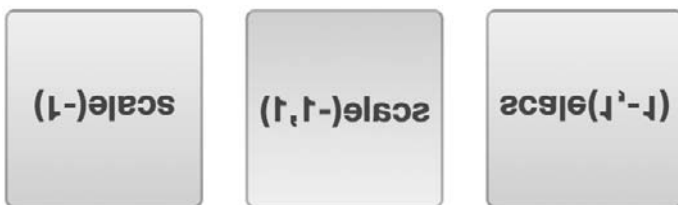
```
transform: scaleX(3.5);
```

Есть еще один визуальный трюк, предлагаемый функцией масштабирования: возможность перевернуть элемент вокруг его вертикальной или горизонтальной центральной оси. Никто не знает достоверно, какой из разделов математики был использован консорциумом W3C, чтобы придумать эту систему, но если для масштабирования применить отрицательное значение, то элемент будет перевернут. Например, чтобы перевернуть элемент вокруг его обеих центральных осей, нужно применить следующее объявление:

```
transform: scale(-1);
```

Получится изображение, показанное на рис. 10.4, *слева*. Можно также перевернуть элемент вокруг одной центральной оси. На правом изображении на рис. 10.4 изображение перевернуто только вокруг его горизонтальной центральной оси. Среднее изображение получено за счет поворота изображения вокруг его вертикальной оси:

```
transform: scale(-1,1);
```



**Рис. 10.4.** Чтобы запутать или обескуражить посетителей вашего сайта, в CSS есть масса разных способов. Левый квадрат перевернут вокруг горизонтальной центральной оси, средний — вокруг вертикальной, а правый — вокруг обеих осей

Похоже на эффект зеркального бокового отображения элемента или на просмотр элемента на просвет сзади после его поворота. Весьма забавно! Конечно, если нужно создать сайт как дань памяти Леонардо да Винчи ([www.mos.org/sln/Leonardo/LeonardoRighttoLeft.html](http://www.mos.org/sln/Leonardo/LeonardoRighttoLeft.html)), то вполне может подойти зеркальное отображение всего имеющегося на странице текста!



## Перемещение

Функция `translate` свойства `transform` просто перемещает элемент из его текущей позиции на некоторое расстояние вправо или влево и вверх или вниз. Пользы от этой функции как таковой немного. Как выяснится далее, когда веб-браузер перемещает элемент, он не перестраивает содержимое страницы, а раскладывает его так, как будто элемент никуда не перемещался. Соответственно, когда с использованием функции `translate` перемещается `div` или другой тег, браузер оставляет пустое пространство там, где этот тег появился бы при обычных обстоятельствах, а затем рисует элемент в его новой позиции (рис. 10.5). Зачастую в результате перемещения элемента таким образом остается пустое место: незаполненное пространство между двумя заголовками является тем местом, где должен был появиться обрамленный тег `<div>` при обычных обстоятельствах.

Если нужно лишь разместить элемент на странице, можно просто воспользоваться абсолютным или относительным позиционированием, рассматриваемым в гл. 15.

Но перемещение пригодится, когда нужно изобразить небольшое движение в ответ на проход указателя мыши или на щелчок кнопкой. Например, во многих конструкциях пользовательского интерфейса при щелчке на кнопке эта кнопка смещается немного вниз и влево, имитируя внешний вид реальной трехмерной кнопки, нажатой на клавиатуре. Применительно к ссылке этот эффект можно имитировать с помощью функции `translate` и состояния `:active`:

```
.button:active {
  -webkit-translate(1px,2px);
  -moz-translate(1px,2px);
  -o-translate(1px,2px);
  -ms-translate(1px,2px);
  translate(1px,2px);
}
```

**Beware of transformed elements...**

**...they can overlap other page content.**

`translate(200px,150px)`

**Рис. 10.5.** Функция `translate` перемещает элемент на указанное количество пикселей, em или процентов от его обычного места на странице

Функции `translate` передаются два значения: первое определяет величину горизонтального, а второе — вертикального перемещения. В данном примере щелчок на элементе с классом `.button` вызывает перемещение этого элемента на 1 пиксел вправо и на 2 пиксела вниз. Чтобы элемент переместился влево, нужно для первого значения использовать отрицательное число, применение отрицательного числа в качестве второго значения приведет к перемещению элемента вверх.

Но можно применять не только пиксельные значения. Будут работать любые значения, указывающие длину в CSS, — px, em, % и т. д.

В CSS3 предоставляются также две дополнительные функции для перемещения элемента только влево или вправо — `translateX` и только вверх или вниз — `translateY`. Например, для перемещения элемента вверх на `.5em` используется функция `translateY`:

```
transform: translateY(-.5em);
```

Очень интересный эффект от функции `translate` можно получить при ее совместном использовании с CSS-переходами. Это сочетание позволяет анимировать перемещение элемента, показав его путешествующим по экрану. Вскоре вы увидите, как это делается.

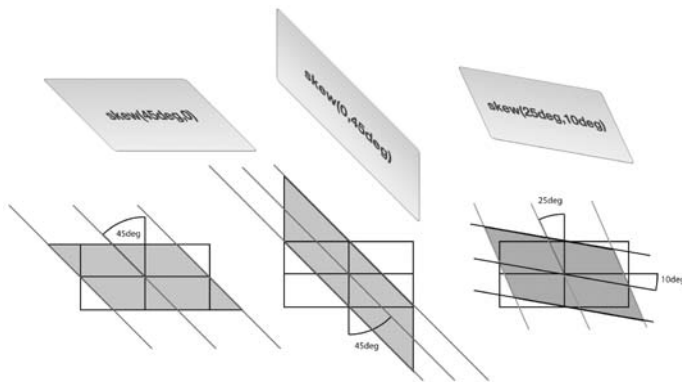
## Наклон

Наклон элемента можно осуществить по его горизонтальной и вертикальной осям: это придает элементу трехмерное представление (рис. 10.6). Например, для наклона всех вертикальных линий влево на  $45^\circ$  (как в первом изображении на рис. 10.6), нужно написать следующий код:

```
transform: skew(45deg, 0);
```

Чтобы придать элементу такой же наклон, но по оси  $Y$  (среднее изображение на рис. 10.6), нужно написать следующий код:

```
transform: skew(0,45deg);
```



**Рис. 10.6.** Функция `skew` является одним из способов имитации трехмерного отображения

Можно наклонить элемент по двум осям одновременно. Например, следующий код приводит к появлению третьего изображения, показанного на рис. 10.6:

```
transform: skew(25deg,10deg);
```

Первое значение задает угол от `0deg` до `360deg`, действующий в направлении против часовой стрелки от верхней части элемента. Например, в первом изображении

на рис. 10.6  $45^\circ$  представлены линией, нарисованной из центра элемента и повернутой на  $45^\circ$  против часовой стрелки (вниз и влево).

---

**ПРИМЕЧАНИЕ**

Не забудьте в окончательном виде кода CSS добавить префиксные версии свойства transform для различных браузеров: -webkit-transform, -moz-transform и т. д.

---

Вторым значением является угол от 0deg до 360deg. Но этот угол действует в направлении по часовой стрелке с правой стороны элемента. Среднее изображение на рис. 10.6 является примером  $45^\circ$ -градусного наклона по всем горизонтальным линиям.

---

**ПРИМЕЧАНИЕ**

Чтобы получить визуальное представление о CSS-преобразованиях, зайдите на сайт с интерактивным средством визуализации по адресу <http://westciv.com/tools/transforms/index.html>.

---

Как и в случае с translate и scale, в CSS3 предлагаются отдельные функции для осей X и Y: skewX и skewY.

---

**ПРИМЕЧАНИЕ**

В CSS3 есть еще один метод перемещения под названием matrix. Он представляет собой числовой массив. Этот метод не всегда понятен интуитивно. Наиболее понятное объяснение можно найти на веб-сайте <http://dev.opera.com/articles/view/understanding-the-css-transforms-matrix/>.

---

## Множественные преобразования

Однако вы не ограничены только одним преобразованием. Изображение можно одновременно масштабировать и наклонять, вращать и перемещать или использовать любые из четырех различных преобразований. Нужно просто добавить через запятую к свойству transform дополнительные функции. Например, повернуть элемент на  $45^\circ$  и увеличить его размер вдвое можно с помощью следующего объявления:

```
transform: rotate(45deg) scale(2);
```

А вот пример всех четырех преобразований, применяемых одновременно:

```
transform: skew(45deg,0deg) scale(.5) translate(400px,500px) rotate(90deg);
```

Браузер будет применять все эффекты в порядке следования функций. Например, во втором примере элемент сначала будет наклонен, затем масштабирован, после чего будет перемещен и, наконец, подвергнется вращению. Порядок не играет роли, если только не используется перемещение. Поскольку при перемещении изменяется местоположение элемента, то, если, например, поместить перемещение перед вращением, браузер сначала переместит элемент, а затем применит к нему вращение. Так как сначала элемент перемещается, точка, вокруг которой он будет вращаться, изменится. С другой стороны, если сначала будет применено вращение, то затем элемент, уже подвергшийся вращению, будет перемещен на определенное расстояние относительно его центра (который теперь будет находиться в новом месте).

## Исходная точка

Обычно, когда к элементу применяется преобразование, в качестве точки начала преобразования браузер использует центр элемента. Например, при вращении элемента браузер поворачивает его вокруг центральной точки (рис. 10.7, *слева*).

Но в CSS3 разрешается изменять точку преобразования, используя свойство `transform-origin`. Это свойство работает точно так же, как и ранее рассмотренное свойство `background-position`. Для него можно указывать ключевые слова, абсолютные значения и относительные значения в `em` и процентах.

Например, чтобы повернуть `div`-контейнер вокруг его левой верхней точки (см. рис. 10.7, *посередине*), можно воспользоваться ключевыми словами `left` и `top`:

```
transform-origin: left top;
```

Можно также использовать пиксельные значения:

```
transform-origin: 0 0;
```

или проценты:

```
transform-origin: 0% 0%;
```

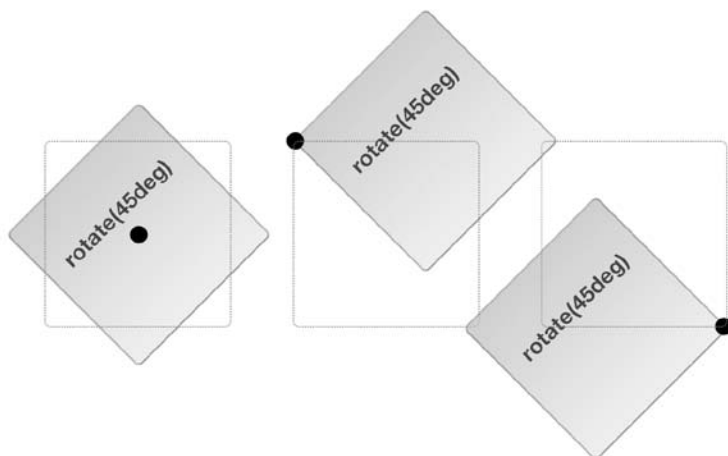
Точно так же для вращения элемента вокруг его нижнего правого угла (см. рис. 10.7, *справа*), используются ключевые слова `right` и `bottom`:

```
transform-origin: right bottom;
```

Что эквивалентно следующему объявлению:

```
transform-origin: 100% 100%;
```

При использовании пикселей, `em` или процентных значений, первое число означает горизонтальную, а второе — вертикальную позицию.



**Рис. 10.7.** Установка значения для свойства `transform-origin` изменяет точку элемента, в которой будет применено преобразование. Пунктиром обозначен элемент, каким бы он был без вращения (его обычная позиция на странице)

#### ПРИМЕЧАНИЕ

Свойство `transform-origin` не влияет на элементы, которые подвергаются только перемещению с помощью функции `translate`.

## ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

### Трехмерные преобразования

В CSS3 имеется также более сложный тип преобразования. Трехмерные преобразования (3D transforms) позволяют имитировать трехмерное пространство на плоском экране монитора, планшетного компьютера или телефона.

Краткое введение в трехмерные преобразования дано на сайтах <http://coding.smashingmagazine.com/2012/01/06/adventures-in-the-third-dimension-css-3-d-transforms/> и <http://blogs.msdn.com/b/ie/archive/2012/02/02/css3-3d-transforms-in-ie10.aspx>. Посмотреть удачные примеры трехмерных преобразований в действии можно на следующих сайтах.

- Один из первых примеров демонстрации возможностей трехмерных преобразований — страница Apple's Morphing Power Cubes ([www.webkit.org/blog-files/3d-transforms/morphing-cubes.html](http://www.webkit.org/blog-files/3d-transforms/morphing-cubes.html)) — по-

казывает вращающийся куб, который может превращаться во вращающийся набор плиток.

- Еще одна впечатляющая демонстрация от Apple — галерея картинок Horizontal 3D (<https://developer.apple.com/safaridemos/showcase/gallery/>) — позволяет быстро просматривать похожую на карусель серию фотографий.
- Snow Stack: используйте клавиши управления курсором (с левой и правой стрелками), чтобы перемещаться по бесконечной галерее фотографий с возможностью выбора: <http://www.satine.org/research/webkit/snowleopard/snowstack.html>.
- Веб-сайт 2011 BeerCamp (<http://2011.beercamp.com>) предоставляет новаторский способ навигации путем перемещения страниц в пространстве вперед и назад.

## Переходы

Преобразованиями, конечно, можно позабавить посетителей (особенно функцией `rotate`), но в сочетании с переходами CSS3 они по-настоящему оживят вашу страницу. Переход является простой анимацией от одного набора CSS-свойств к другому через определенный промежуток времени. Например, можно задать вращение баннера на 360° в течение двух секунд. Чтобы переход заработал, нужно следующее.

- **Два стиля.** Один стиль должен представлять начальный вид элемента, например красную кнопку перехода, а второй — его конечный вид: синюю кнопку перехода. О процессе анимации изменения между двумя состояниями позаботится веб-браузер (например, об изменении цвета кнопки с красного на синий).
- **Свойство `transition`.** В CSS3 добавляется свойство `transition` — секретный ингредиент, позволяющий осуществить анимацию. Обычно свойство `transition` применяется к исходному стилю, который определяет внешний вид элемента до начала анимации.
- **Инициатор.** Инициатор представляет собой действие, вызывающее изменение от одного стиля к другому. В CSS для запуска анимации можно использовать несколько псевдоклассов. Наиболее часто применяется псевдокласс `:hover`. С его помощью можно анимировать изменение, происходящее от обычного появления элемента и до его внешнего вида, который появляется, когда посетитель проводит над ним указатель мыши. Как это делается, вы увидите совсем скоро. Кроме того, у вас в активе такие псевдоклассы, как `:active` (связанный

со щелчком кнопкой мыши на элементе), `:target` (связанный с элементом, ставшим целью перехода по ссылке) и `:focus` (связанный с переходом на ссылку с помощью клавиши табуляции, или со щелчком по полю формы, или с переходом на это поле с помощью клавиши табуляции). Кроме того, для динамической смены стиля любого тега можно воспользоваться JavaScript (см. врезку «Информация для опытных пользователей. Использование JavaScript для запуска переходов» в подразделе «Распределение скорости выполнения перехода по времени» данного раздела).

Когда инициатор больше не применяется, то есть когда, например, посетитель убирает указатель мыши с кнопки навигации, браузер возвращает тег обратно к его прежнему стилю и анимирует весь этот процесс. Иными словами, вам нужно лишь установить переход для элемента один раз, а анимацию от одного стиля к другому и обратно к исходному стилю браузер возьмет на себя.

Веб-браузер не может анимировать каждое отдельно взятое свойство CSS, но у вас в распоряжении остается все же весьма длинный список свойств, на которых можно остановить свой выбор. Кроме таких преобразований, как `rotate`, `scale`, `translate` и `skew`, о которых только что шла речь, можно также анимировать такие свойства, как `color`, `background-color`, `border-color`, `border-width`, `font-size`, `height`, `width`, `letter-spacing`, `line-height`, `margin`, `opacity`, `padding`, `word-spacing`, свойства позиционирования `top`, `left`, `right` и `bottom`, которые будут рассмотрены в гл. 15, а также многие другие свойства. Их полный список можно найти по адресу [www.w3.org/TR/css3-transitions/#animatable-properties](http://www.w3.org/TR/css3-transitions/#animatable-properties).

---

#### ПРИМЕЧАНИЕ

Появившиеся в CSS3 переходы работают в большинстве браузеров. Но когда речь заходит об Internet Explorer, с сожалением приходится заметить, что переходы этот браузер понимает, только начиная с версии 10. Поэтому лучше использовать переходы только как средство придания выразительности. При этом в большинстве случаев все будет выглядеть вполне приемлемо: Internet Explorer 9 и более ранние версии смогут обеспечить переключение между двумя стилями (например, показать стиль для `hover`) без анимации данного изменения.

---

## Добавление перехода

В основе CSS-переходов лежат четыре свойства, который управляют тем, какие свойства анимировать, сколько времени займет анимация, какой тип анимации будет использован и какой будет необязательная задержка перед началом анимации. Рассмотрим простой пример. Предположим, что нужно анимировать изменение фонового цвета кнопки навигации с оранжевого на синий при проходе над кнопкой указателя мыши посетителя. Начать нужно с двух стилей, необходимых для переключения между этими двумя цветами. Можно, например, применить к ссылке класс `.navButton`, а затем создать следующие два стиля:

```
.navButton {
    background-color: orange;
}

.navButton:hover {
    background-color: blue;
}
```

Эти стили будут работать в любом браузере, при проходе указателя мыши над кнопкой навигации фоновый цвет этой кнопки изменится с оранжевого на синий. Но это изменение произойдет мгновенно. Чтобы цвет изменился в режиме анимации за одну секунду, нужно к стилю `.navButton` добавить два новых свойства:

```
.navButton {  
  background-color: orange;  
  transition-property: background-color;  
  transition-duration: 1s;  
}  
  
.navButton:hover {  
  background-color: blue;  
}
```

Первое свойство — `transition-property` — указывает на анимируемое свойство. Можно указать одно свойство (как в показанном выше примере), воспользоваться ключевым словом `all` для анимирования всех изменяемых CSS-свойств или применить список с запятой в качестве разделителя для указания более чем одного свойства (но не всех свойств). Предположим, например, что вы создаете стиль `:hover`, чтобы изменялись сразу цвет текста, фоновый цвет и цвет границы. Все эти три свойства указываются в виде следующего списка:

```
transition-property: color, background-color, border-color;
```

или, чтобы упростить код, указывается ключевое слово `all`:

```
transition-property: all;
```

Во многих случаях при анимировании каждого CSS-изменения лучше использовать ключевое слово `all`, в результате чего создается привлекательный визуальный эффект.

Чтобы указать продолжительность анимации, используется свойство `transition-duration`. Ему передается или значение в секундах, или значение в миллисекундах (тысячных долях секунды). Например, чтобы переход занимал полсекунды, можно использовать либо:

```
transition-duration: .5s;
```

либо

```
transition-duration: 500ms;
```

Можно даже указать отдельную продолжительность для каждого анимируемого свойства. Например, когда посетитель проводит указатель мыши над кнопкой, может потребоваться, чтобы цвет текста изменялся быстрее, цвет фона изменялся немного медленнее, а цвет границы изменялся заметно медленнее. Для этого нужно перечислить анимируемые свойства, используя свойство `transition`, а затем перечислить показатели продолжительности с помощью свойства `transition-duration`:

```
transition-property: color, background-color, border-color;
```

```
transition-duration: .25s, .75s, 2s;
```

Порядок перечисления показателей продолжительности должен соответствовать порядку перечисления свойств. Следовательно, в показанном выше примере `.25s` относится к свойству `color`, `.75s` — к свойству `background-color`, а `2s` — к свойству `border-color`. Если поменять свойства местами, их показатели продолжительности переходов изменятся.

К сожалению, как и в случае применения многих других свойств CSS3, для работы свойства `transition` требуется указать префиксы производителей. То есть показанный выше пример `.navButton` во многих браузерах не будет работать, пока не будут добавлены соответствующие префиксы производителей, поэтому данный стиль нужно переписать следующим образом:

```
.navButton {  
  background-color: orange;  
  -webkit-transition-property: background-color;  
  -moz-transition-property: background-color;  
  -o-transition-property: background-color;  
  transition-property: background-color;  
  -webkit-transition-duration: 1s;  
  -moz-transition-duration: 1s;  
  -o-transition-duration: 1s;  
  transition-duration: 1s;  
}
```

Префикс производителя `-ms-` применять не нужно, поскольку Internet Explorer 9 и более ранние версии свойства `transition` не понимают, а Internet Explorer 10 понимает версию этого свойства без указания префикса.

#### ПРИМЕЧАНИЕ

---

В процессе написания книги текущие версии Firefox и Opera перестали нуждаться в указании префиксов производителей для свойства `transition`. Но чтобы гарантировать работу анимации в более ранних версиях этих браузеров, лучше пока продолжить использование префиксов `-moz-` и `-o-`.

---

## Распределение скорости выполнения перехода по времени

Чтобы анимированный переход заработал, нужно только установить значения для свойств `transition-property` и `transition-duration`. Но с помощью свойства `transition-timing-function` можно также контролировать и скорость хода анимации. В предназначении этого свойства нетрудно и запутаться: оно управляет не продолжительностью анимации (для этого есть свойство `transition-duration`), а скоростью хода анимации. Например, можно начать анимацию медленно, а затем быстро ее завершить, создавая эффект незаметного в начале и быстрого в конце изменения фонового цвета.

Свойство `transition-timing-function` может получать одно из пяти ключевых слов: `linear`, `ease`, `ease-in`, `ease-out` и `ease-in-out`. Если функцию регулирования скорости не задавать, браузер будет использовать метод `ease`, при котором анимация начинается медленно, ускоряется к середине и замедляется к концу, предоставляя более естественное изменение. Вариант `linear` дает равномерное изменение на протяжении всего периода анимации. Ни один из вариантов не имеет заметных



преимуществ перед другими, они просто предлагают различную общую картину, и для определения своих предпочтений нужно просмотреть все варианты.

Чтобы воспользоваться этим свойством, нужно просто добавить свойство `transition-timing-function` и метод, который вам хочется использовать:

```
transition-timing-function: ease-in-out;
```

Разумеется, как и в случае со всеми свойствами переходов, нужно также добавить префиксы производителей. Поэтому, чтобы заставить эту строку кода работать на максимально возможном количестве браузеров, ее нужно переписать следующим образом:

```
-webkit-transition-timing-function: ease-in-out;  
-moz-transition-timing-function: ease-in-out;  
-o-transition-timing-function: ease-in-out;  
transition-timing-function: ease-in-out;
```

#### ПРИМЕЧАНИЕ

---

В случае с различными функциями регулирования скорости перехода лучше один раз увидеть, чем сто раз услышать. Зайдите на сайт [www.the-art-of-web.com/css/timing-function/](http://www.the-art-of-web.com/css/timing-function/), чтобы увидеть великолепно преподнесенное сравнение пяти методов регулирования скорости.

---


Для свойства `transition-timing` можно также применять то, что называется кубической кривой Безье. Эта кривая определяет график хода анимации по времени (рис. 10.8). Если вам приходилось работать с такими программами обработки изображений, как Adobe Illustrator, то вы, наверное, знакомы с кривыми Безье. Кривизной линии можно управлять путем настройки двух контрольных точек: чем круче линия, тем быстрее анимация, а более пологая линия означает более медленный ход анимации. Например, кривая Безье, нарисованная на рис. 10.8, начинается с крутого участка (анимация начинается в быстром темпе), затем переходит к середине в более пологую фазу (анимация замедляется), а затем кривизна снова возрастает (анимация быстро подходит к своей завершающей стадии). Для создания анимации такого профиля нужно добавить следующую строку кода:

```
transition-timing-function: cubic-bezier(.20, .96, .74, .07);
```

Вам совершенно не обязательно забивать себе голову кубическими кривыми Безье. Лучше воспользоваться одним из многочисленных интерактивных средств для создания и тестирования различных функций распределения скорости хода анимации. Одним из лучших считается средство *Seasar*, разработанное Мэтью Лейном: <http://matthewlein.com/ceaser/>. Можно задать очень медленное начало и быстрое завершение или наоборот. Интерактивное средство *Caesar*, изображенное на рис. 10.7, превращает создание кубических кривых Безье в сущий пустяк: нужно просто для изменения кривизны перетаскивать нижнюю левую и верхнюю правую контрольные точки. Чем круче линия, тем быстрее идет анимация на ее участке.

В данном примере линия сначала имеет относительно пологую форму, то есть вначале анимация будет идти медленно, а затем до самого конца линия резко уходит вверх, то есть в конце времени перехода анимация резко ускорится.

Так же, как и в случае со свойством `transition-duration`, к различным свойствам можно применять различные функции распределения хода анимации по времени.



**Ceaser** CSS EASING ANIMATION TOOL

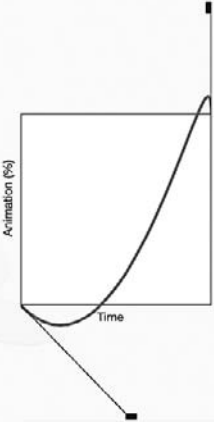
[Tweet](#)

1. Choose an easing type and test it out with a few effects.  
 2. If you don't quite like the easing, grab a handle and fix it.  
 3. When you're happy, snag your code and off you go.

Now that we can use CSS transitions in all the modern browsers, let's make them pretty. I love the classic Penner equations with Flash and jQuery, so I included most of those. If you're anything like me, you probably thought this about the default easing options: "ease-in, ease-out...yawn." The mysterious cubic-bezier has a lot of potential, but was cumbersome to use. Until now. Also, touch-device friendly!

*\*If you are anything like me, we should be friends @matthowlin*


**Note:** Bugfixes have landed, so the newest Webkit now supports values above 1 and below 0. For the time being, I am including fallback code for older Webkit that is clamped between 0 and 1 when needed.



Easing:

Duration:

Effect: Left Width Height Opacity



**Code snippets, short and long-hand:**

```

-webkit-transition: all 500ms cubic-bezier(0.580, 0, 1.000, 1); /* older webkit */
-webkit-transition: all 500ms cubic-bezier(0.580, -0.600, 1.000, 1.560);
-moz-transition: all 500ms cubic-bezier(0.580, -0.600, 1.000, 1.560);
-ms-transition: all 500ms cubic-bezier(0.580, -0.600, 1.000, 1.560);
-o-transition: all 500ms cubic-bezier(0.580, -0.600, 1.000, 1.560);
transition: all 500ms cubic-bezier(0.580, -0.600, 1.000, 1.560); /* custom */

-webkit-transition-timing-function: cubic-bezier(0.580, 0, 1.000, 1); /* older webkit */
-webkit-transition-timing-function: cubic-bezier(0.580, -0.600, 1.000, 1.560);
-moz-transition-timing-function: cubic-bezier(0.580, -0.600, 1.000, 1.560);
-ms-transition-timing-function: cubic-bezier(0.580, -0.600, 1.000, 1.560);
-o-transition-timing-function: cubic-bezier(0.580, -0.600, 1.000, 1.560);
transition-timing-function: cubic-bezier(0.580, -0.600, 1.000, 1.560); /* custom */

```

If this saves you time, or blows your mind, consider making a [Donation](#) to keep these projects alive.

**Рис. 10.8.** Создание кубической кривой Безье позволяет создавать широкий диапазон интересных функций распределения скорости выполнения анимации по времени

## ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

### Использование JavaScript для запуска переходов

CSS3-переходы являются простыми в использовании средствами анимации, встроенными непосредственно в веб-браузеры (по крайней мере, в большинстве веб-браузеров). Сложные эффекты можно создавать,

просто определяя для этого набор исходных и конечных CSS-свойств и возлагая все остальное на веб-браузер. К сожалению, для запуска этих переходов у вас в распоряжении имеется всего лишь несколько CSS-селекторов.

**ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ**

Чаще всего для изменения элемента при проходе над ним указателя мыши используется псевдокласс `:hover`. Для анимации элемента формы при щелчке на нем кнопкой мыши можно также воспользоваться псевдоклассом `:focus` (например, можно заставить область многострочного ввода при получении фокуса изменять свою высоту, а затем сжиматься до нескольких строк при щелчке за пределами этой области).

Но в CSS нет, например, псевдокласса `:click`, поэтому при желании запустить переход по щелчку на элементе, сделать это средствами CSS не получится. Не удастся также запустить анимацию какого-нибудь элемента при проходе указателя мыши над другим элементом, если использовать только CSS. Но переходы CSS работают при любом изменении CSS-свойства, даже если это делается с помощью JavaScript.

JavaScript является языком программирования, позволяющим добавлять интерактивность веб-

страницам, создавать динамические пользовательские интерфейсы и осуществлять множество других полезных действий. Но JavaScript можно также использовать для простого добавления или удаления класса элемента или изменения какого угодно свойства CSS. Используя JavaScript, можно добавить класс к изображению, когда посетитель щелкает на ссылке «Показать увеличенный вариант изображения».

Этот новый класс просто увеличивает масштаб изображения (с помощью свойства `scale-transform`). Добавляя к изображению переход, вы тут же получите анимацию. JavaScript является довольно обширной темой, заслуживающей отдельной книги. Но если для начала вы захотите освоить простейший способ использования CSS-переходов, запускаемых с помощью JavaScript, прочитайте следующее краткое руководство: [www.netmagazine.com/tutorials/get-more-out-your-css-transitions-jquery](http://www.netmagazine.com/tutorials/get-more-out-your-css-transitions-jquery).

## Задержка начала перехода

И наконец, можно задержать время начала анимации перехода, воспользовавшись свойством `transition-delay`. Например, если вы хотите подождать полсекунды, прежде чем начать анимацию, можно добавить следующий код:

```
transition-delay: .5s;
```

Разумеется, это свойство требует наличия префиксов производителей, поэтому, чтобы заставить его работать в большинстве браузеров, нужно написать следующий код:

```
-webkit-transition-delay: .5s;  
-moz-transition-delay: .5s;  
-o-transition-delay: .5s;  
transition-delay: .5s;
```

Чаще всего задержка перехода для всех свойств вам не понадобится, поскольку она снижает эффект интерактивности. В конце концов, жестоко заставлять посетителей ждать целую секунду, пока они увидят изменения в кнопке при наведении на нее указателя мыши. Но если анимируются сразу несколько свойств, может потребоваться выждать с изменением одного свойства, пока не закончится анимация перехода других свойств. Предположим, например, что у вас есть кнопка, цвет фона и текста которой нужно изменить, а затем неожиданно изменить цвет ее границы после завершения изменения первых двух свойств. Это можно сделать с помощью следующего кода:

```
.navButton {
  color: black;
  background-color: #FF6603;
  border: 5px solid #660034;
  transition-property: color, background-color, border-color;
  transition-duration: 1s, 1s, .5s;
  transition-delay: 0, 0, 1s;
}

.navButton:hover {
  color: white;
  background-color: #660034;
  border-color: #FF6603;
}
```

Как и в случае применения свойства `transition-duration`, для каждого свойства можно указать свое время задержки. Порядок перечисления показателей времени должен соответствовать порядку перечисления этих свойств для `transition-property`. Например, в предыдущем коде задержки для переходов цвета текста и цвета фона отсутствуют, но зато есть односекундная задержка до начала изменения цвета границы.

---

#### СОВЕТ

Обычно свойства перехода помещаются в начальный (`.navButton` в предыдущем примере), а не в конечный стиль (`.navButton:hover`). Но при использовании раскрывающегося меню CSS возникают некоторые сложности. Проблема в том, что при задании раскрывающихся меню с помощью CSS они слишком быстро пропадают при случайном выходе указателя мыши за их пределы. Но с помощью свойства `transition-delay` можно заставить меню быстро появляться и медленно исчезать. Для этого добавьте в исходный стиль следующий код:

```
transition-delay: 5s;
```

Затем добавьте задержку к стилю `:hover`:

```
transition-delay: 0;
```

Каким бы странным этот код ни показался, он заставляет переход по `:hover` происходить немедленно, без задержки. А вот возвращение к обычному стилю (при котором меню исчезает) занимает 5 секунд. За это время посетитель успеет вернуть свою непослушную мышь обратно на меню, пока оно не исчезло.

---

## Краткая запись свойства `transition`

Запись всех свойств по отдельности — `transition-property`, `transition-duration`, `transition-timing-function` — может просто утомить. Особенно если потребуется создать и версию каждого из них с префиксом производителя. К счастью, есть более быстрый способ задания переходов — свойство `transition`.

Это свойство связывает все другие свойства в одно. Нужно просто перечислить через запятую свойство, продолжительность, функцию распределения скорости по времени и задержку. Например, для анимации всех CSS-свойств на 1 секунду, используя функцию `ease-in` и полусекундную задержку, нужно написать следующий код:

```
transition: all 1s ease-in .5s;
```

Список должен состоять из ключевого слова `all` или какого-нибудь одного CSS-свойства, а также из продолжительности, а функцию распределения скорости по времени и задержку можно не указывать. По умолчанию в качестве функции распределения задается `ease-in`, а задержка не используется. То есть, если нужно просто анимировать переход всех CSS-свойств на 1 секунду, нужно написать следующий код:

```
transition: all 1s;
```

Если нужно анимировать изменение фонового цвета, включите в список именно это свойство:

```
transition: background-color 1s;
```

Здесь можно включить в список только одно CSS-свойство, поэтому, если нужно анимировать сразу несколько CSS-свойств (но не все), можно воспользоваться их списком с запятой в качестве разделителя, где элементами будут являться разделенные пробелами свойства переходов. Возьмем один из предыдущих примеров, где свойство `border-color` анимировалось отдельно от цвета текста и цвета фона. Его код можно переписать следующим образом:

```
transition: color 1s, background-color 1s, border-color .5s 1s;
```

Чтобы код было легче читать, многие веб-разработчики помещают каждый переход на отдельной строке:

```
transition: color 1s,  
           background-color 1s,  
           border-color .5s 1s;
```

Нужно только не забывать разделять переходы запятыми, а всю связку завершить точкой с запятой. Разумеется, нужно будет предоставить также и версии с префиксами производителей:

```
-webkit-transition: color 1s,  
                  background-color 1s,  
                  border-color .5s 1s;  
-moz-transition: color 1s,  
                background-color 1s,  
                border-color .5s 1s;  
-o-transition: color 1s,  
              background-color 1s,  
              border-color .5s 1s;  
transition: color 1s,  
           background-color 1s,  
           border-color .5s 1s;
```

## Анимация

В CSS3 предоставляется еще один, более богатый свойствами механизм создания анимации. С помощью CSS-переходов можно анимировать только переход от одного набора CSS-свойств к другому. Анимация, предусмотренная в CSS3, позволяет анимировать переходы от одного набора свойств к другому, затем к третьему

и т. д. Кроме того, можно задать повторяющуюся анимацию, приостановить ее выполнение при проходе указателя мыши над элементом и даже повернуть ее вспять, когда анимация дойдет до конца.

Анимация в CSS3 сложнее перехода, но у нее есть дополнительное преимущество в том, что ей не нужен обязательный инициатор. Наряду с тем, что анимацию можно добавить к состоянию `:hover` для ее проигрывания при прохождении указателя мыши над элементом, запустить анимацию можно также при загрузке страницы. Тем самым можно привлечь внимание к баннеру или логотипу, анимированному на той странице, с которой посетитель заходит на сайт.

#### ПРИМЕЧАНИЕ

Анимация в CSS3, как и переходы, не работает в Internet Explorer 9 и более ранних версиях. Но эти эффекты поддерживаются большинством других современных браузеров.

При создании анимации первым делом создаются ключевые кадры (keyframes). Ключевой кадр в анимации — это отдельный кадр анимации, определяющий внешний вид сцены. Предположим, первый ключевой кадр имеет изображение мяча на одной половине футбольного поля. Добавив второй ключевой кадр, можно задать конечную точку анимации, например мяч в воротах на другой половине поля. После этого веб-браузер предоставит анимацию между данными двумя ключевыми кадрами, прорисовывая все промежуточные фазы, отображающие перемещение мяча по полю на его пути к голу в воротах.

Если вы подумали, что в переходы заложена такая же идея, то так оно и есть. В переходе определяются два стиля, а на браузер возлагается задача анимировать изменения от одного стиля к другому. В этом смысле каждый из этих стилей можно представить в качестве ключевых кадров. Но анимации в CSS3 позволяют определять множество ключевых кадров и создавать более сложные эффекты анимации: например, футбольный мяч, перемещающийся с одной стороны поля к игроку, к другому игроку, а затем в ворота.

Создание анимации проходит в два приема.

1. **Определение анимации.** Включает определение ключевых кадров со списком анимируемых CSS-свойств.
2. **Применение анимации к элементу.** После определения анимации ее можно назначить любому количеству элементов страницы. Можно даже задать для каждого элемента разные распределения скорости выполнения, задержки и другие свойства анимации. То есть одну и ту же анимацию на странице можно использовать с немного разными настройками несколько раз.

## Определение ключевых кадров

Первым делом нужно настроить ключевые кадры. Реализующий их синтаксис может показаться немного странным, но его основная структура имеет следующий вид:

```
@keyframes имяАнимации {
  from {
    /* Здесь перечисляются CSS-свойства */
  }
  to {
```

```
    /* Здесь перечисляются CSS-свойства */  
  }  
}
```

Сначала идет ключевое слово `@keyframes`. За ним следует имя, которым вы называете анимацию. Позже это имя будет использоваться при применении анимации к элементу страницы, поэтому оно должно быть описательным, например `fadeOut` или `fadeIn`.

#### ПРИМЕЧАНИЕ

---

Само понятие `@keyframes` является не CSS-свойством, а так называемым правилом. В CSS есть и другие правила, например инструкция `@import` для загрузки внешней таблицы стилей из другой таблицы стилей и `@media` для определения стилей для различных типов аппаратуры представления информации, допустим принтера или экранов различного размера и разрешения.

---

Затем добавляются как минимум два ключевых кадра. В показанном выше примере ключевые слова `from` и `to` используются для создания начального ключевого кадра (`from`) и конечного ключевого кадра (`to`). Внутри каждого ключевого кадра находится одно или несколько CSS-свойств, как будто создается стиль. Фактически каждый ключевой кадр можно представить в виде простого CSS-стиля, содержащего одно или несколько свойств CSS. Предположим, к примеру, что нужно создать анимацию постепенного появления элемента в поле зрения. Можно начать со значения `opacity`, равного 0 (невидимый), и закончить значением этого свойства, равным 1 (полностью видимый):

```
@keyframes fadeIn{  
  from {  
    opacity: 0;  
  }  
  
  to {  
    opacity: 1;  
  }  
}
```

Но вы не ограничены использованием только двух ключевых кадров. С помощью процентных значений можно определить несколько ключевых кадров. Процентное значение указывает место на общей продолжительности анимации, где должно произойти изменение. Предположим, к примеру, что нужно создать эффект изменения фона элемента с желтого на синий, а потом на красный.

Для этого можно написать следующий код:

```
@keyframes backgroundGlow {  
  from {  
    background-color: yellow;  
  }  
  
  50% {  
    background-color: blue;  
  }  
}
```

```
to {  
  background-color: red;  
}  
}
```

В данном случае синий фон появится в середине анимации. Если нужно, чтобы желтый цвет присутствовал в фоне дольше, а синим становился после трех четвертей времени анимации, используйте не 50%, а 75 %. Таким же образом можно продолжать добавлять дополнительные ключевые кадры (например, на 25, 66 % и т. д.).

---

**ПРИМЕЧАНИЕ**

Ключевое слово `from` можно заменить значением 0 %, а ключевое слово `to` — значением 100 %.

---

Но одним свойством CSS можно не ограничиваться. Внутри ключевого кадра может находиться любое количество пригодных к анимации свойств — `background-color`, `opacity`, `width`, `height` и т. д.:

```
@keyframes growAndGlow {  
  from {  
    background-color: yellow;  
  }  
  
  50% {  
    transform: scale(1.5);  
    background-color: blue;  
  }  
  
  to {  
    transform: scale(3);  
    background-color: red;  
  }  
}
```

В этом примере наряду с трехкратным изменением цвета фона происходит увеличение масштаба элемента, размеры которого в итоге становятся втрое больше первоначальных.

Можно также усложнить использование процентных значений, добавив несколько таких значений к одному набору CSS-свойств. Это может пригодиться в нескольких случаях: при необходимости довести анимацию до некоторого момента, сделать паузу, а затем продолжить анимацию. Предположим, например, что вам нужно, чтобы сначала у содержимого тега `<div>` был желтый цвет фона. Затем нужно, чтобы цвет сменился на синий, некоторое время оставался синим, а затем сменился на красный. То есть нужна пауза в середине анимации, при которой цвет не менялся, а затем необходима новая смена цвета. Эта задача решается с помощью следующего кода:

```
@keyframes glow {  
  from {  
    background-color: yellow;  
  }  
}
```



```
25%, 75% {  
    background-color: blue;  
}  
  
to {  
    background-color: red;  
}  
}
```

Обратите внимание на процентные значения 25%, 75% в строке 5. Они означают, что на 25 % от всего хода анимации фоновый цвет элемента станет синим. Но он будет оставаться синим до 75 %, то есть 25 % от всего хода анимации. От отметки 25 % до отметки 75 % фон будет сохранять чисто-синий цвет, перед тем как к окончанию анимации превратиться в красный. Если продолжительность этой анимации будет 4 секунды, то в средние 2-й секунды фон элемента будет иметь чисто-синий цвет.

Процентное значение можно также использовать при необходимости применения одного и того же набора CSS-свойств для различных частей анимации. Предположим, например, что нужно провести еще одну анимацию фонового цвета, но на этот раз менять цвет от желтого к синему, затем к оранжевому, после чего к синему, к оранжевому и к красному. Синий и оранжевый цвета фигурируют в этом перечне дважды, поэтому вместо многократного упоминания об этих свойствах фонового цвета можно воспользоваться следующим кодом:

```
@keyframes glow {  
    from {  
        background-color: yellow;  
    }  
  
    20%, 60% {  
        background-color: blue;  
    }  
  
    40%, 80% {  
        background-color: orange;  
    }  
  
    to {  
        background-color: red;  
    }  
}
```

В данном случае фоновый цвет превратится в синий на отметке 20 %, в оранжевый — на отметке 40 %, затем опять в синий на отметке 60 %, и перед тем как стать в конце анимации красным, он еще один последний раз на отметке 80 % превратится в оранжевый.

Один из недостатков анимации CSS3 состоит в том, что ей требуются префиксы производителей (как и большинству других свойств CSS3). То есть вам нужно повторить определения ключевых кадров для каждого браузера. Поэтому для за-

пуска следующей анимации постепенного появления элемента на как можно большем количестве браузеров, придется написать следующий код:

```
@-webkit-keyframes fadeIn {
  from {
    opacity: 0;
  }

  to {
    opacity: 1;
  }
}

@-moz-keyframes fadeIn{
  from {
    opacity: 0;
  }

  to {
    opacity: 1;
  }
}

@-o-keyframes fadeIn{
  from {
    opacity: 0;
  }

  to {
    opacity: 1;
  }
}

@keyframes fadeIn{
  from {
    opacity: 0;
  }

  to {
    opacity: 1;
  }
}
```

Обратите внимание на два дефиса: один между символом @ и префиксом производителя и второй между префиксом производителя и словом keyframes.

---

**ПРИМЕЧАНИЕ**

Префикс производителя -ms- для Internet Explorer в данном случае не нужен. Internet Explorer 9 и более ранние версии анимацию совсем не воспринимают, а Internet Explorer 10 поддерживает синтаксис @keyframes без префикса.

---

## Применение анимации

Когда определение набора ключевых кадров будет завершено, анимация будет готова к применению, и, чтобы заставить ее работать, ее нужно применить к какому-нибудь элементу страницы. Анимацию можно добавить к любому стилю любого элемента страницы. Если просто добавить анимацию к стилю, который тут же применяется к элементу, например к стилю тега `h1`, анимация будет применена при загрузке страницы. Этот прием можно использовать для добавления на страницу вводной анимации, которая увеличивает масштаб логотипа, помещая его в верхний левый угол страницы, или создает свечение конкретного блока информации, привлекая к нему внимание.

Кроме того, анимацию можно применить к одному из псевдоклассов, включая `:hover`, `:active`, `:target` или `:focus`, чтобы, например, запустить анимацию при проходе указателя мыши посетителя над ссылкой или при щелчке на поле формы. И наконец, анимацию можно применить к стилю класса, и воспользоваться кодом JavaScript для динамического применения этого класса в ответ на щелчок посетителя на кнопке или на каком-нибудь другом элементе страницы.

В CSS3 предоставляется несколько свойств, связанных с анимацией и позволяющих управлять способом и временем проигрывания анимации (а также сокращенная версия, охватывающая все отдельные свойства). Как минимум, чтобы заставить анимацию выполняться, нужно указать имя, которое было дано исходной анимации (в правиле `@keyframes`), и продолжительность анимации.

Рассмотрим простой пример. Предположим, нужно, чтобы при загрузке страницы в поле зрения появился `div`-контейнер с важным объявлением. Этому `div`-контейнеру был назначен класс с именем `announcement`:

```
-<div class="announcement">.
```

1. Создайте анимацию постепенного появления, задав правило `@keyframes`:

```
@keyframes fadeIn {
  from { opacity: 0; }
  to { opacity: 1; }
}
```

---

### СОВЕТ

Если анимируется всего лишь одно свойство, то его будет проще прочитать, если поместить весь ключевой кадр в одной строке:

```
from { opacity: 0; }
```

Но если анимируется множество свойств, то читать (и набирать) проще будет при распределении кода по нескольким отдельным строкам:

```
from {
  opacity: 0;
  color: red;
  width: 50%;
}
```

---

2. Примените эту анимацию к стилю, предназначенному для тега `<div>`:

```
.announcement {
  animation-name: fadeIn;
  animation-duration: 1s;
}
```

Свойство `animation-name` просто сообщает браузеру, какую анимацию нужно применить. Ему указывается то самое имя, которое предоставлялось анимации при выполнении шага 1. Свойство `animation-duration` устанавливает время, которое займет анимация от старта до финиша. В данном примере перечисляются только два свойства анимации, но вы можете (и, наверное, захотите) поместить в стиль и другие, не связанные с анимацией свойства. Например, в реальности к нашему стилю `.announcement` будут, наверное, добавлены такие свойства, как `width`, `background-color`, `border` и т. д.

---

**ПРИМЕЧАНИЕ**

С технической точки зрения помещать имя анимации в кавычки (`'fadeIn'`) не требуется, и в этом примере так не делается, но помещение в кавычки поможет избежать конфликтов, возникающих при использовании в качестве имени анимации какого-нибудь ключевого слова CSS.

---

Как и в случае использования правила `@keyframes`, каждое из свойств анимации требует префиксов, определяющих производителя, поэтому показанный выше стиль `.announcement` для работы на как можно большем количестве браузеров нужно переписать следующим образом:

```
.announcement {
  animation-name: fadeIn;
  animation-duration: 1s;
  -webkit-animation-name: fadeIn;
  -webkit-animation-duration: 1s;
  -moz-animation-name: fadeIn;
  -moz-animation-duration: 1s;
  -o-animation-name: fadeIn;
  -o-animation-duration: 1s;
  animation-name: fadeIn;
  animation-duration: 1s;
}
```

Возможно, покажется несколько неудобным определять анимацию с помощью правила `@keyframes` в одном месте, а затем применять ее в другом месте (в стиле), но после задания анимация может применяться несколько раз в любом количестве стилей. Например, можно создать общий тип анимации появления элемента на экране и применить этот тип к различным элементам. Более того, можно управлять анимацией для каждого стиля совершенно независимо, например заставить заголовков появляться в течение половины секунды, а другие элементы страницы — в течение 5 секунд.

Кроме того, к одному и тому же элементу можно применить более одной анимации. Предположим, что вы создаете одну анимацию с именем `fadeIn`, чтобы анимировать появление элемента, и еще одну анимацию с именем `blink`, чтобы получить мигающий цвет фона. Чтобы применить к элементу обе анимации, нужно указать список имен с запятой в качестве разделителя:

```
animation-name: fadeIn, blink;
```

Чтобы задать этим анимациям разные параметры времени, нужно указать список этих параметров с запятой в качестве разделителя:

```
animation-name: fadeIn, blink;  
animation-duration: 1s, 3s;
```

Порядок применения параметров времени совпадает с порядком указания имен анимаций. Например, первая анимация получает время, указанное в списке первым. В показанном выше примере продолжительность `fadeIn` составляет 1 секунду, а продолжительность `blink` — 3 секунды.

Можно также применить некоторые другие полезные свойства анимации, рассматриваемые далее.

## Распределение скорости выполнения анимации по времени

Как показано ранее, свойство `animation-duration` позволяет управлять продолжительностью анимации. Как и в случае с переходами (`transitions`), для задания продолжительности можно использовать миллисекунды (например, `750ms`) или секунды (например, `.75s`).

Как и в случае с переходами, можно указывать функцию распределения скорости анимации по времени, которая управляет этой скоростью на заданном времени продолжительности анимации. Например, анимацию можно начинать медленно и завершать быстро, используя для этого кубическую кривую Безье или одно из встроенных ключевых слов, определяющих метод: `linear`, `ease`, `ease-in`, `ease-out`, `ease-in-out`. Все это работает точно так же, как и в ранее рассмотренном случае с переходами.

Свойство `animation-timing-function` можно использовать для управления всей анимацией или только для конкретных ключевых кадров. Например, чтобы применить функцию `ease-out` для представленной ранее анимации `fadeIn` (шаг 1 в предыдущей последовательности действий), добавьте функцию распределения скорости по времени к стилю `.announcement` (шаг 2 в предыдущей последовательности действий):

```
.announcement {  
  animation-name: fadeIn;  
  animation-duration: 1s;  
  animation-timing-function: ease-out;  
}
```

Но можно также управлять функцией распределения скорости по времени между ключевыми кадрами. Предположим, например, что создается анимация с тремя ключевыми кадрами и с тремя различными цветами фона. Веб-браузер будет осуществлять анимацию при превращении одного цвета в другой, а затем в третий. Возможно, вам захочется замедлить превращение первого цвета во второй с помощью кубической кривой Безье, а затем равномерно продолжить анимацию от середины до ее окончания. Это можно сделать путем добавления двух функций распределения скорости по времени, одной для первого ключевого кадра (она будет управлять анимацией от ключевого кадра 1 к ключевому кадру 2), и второй — для второго ключевого кадра для управления анимацией от второго ключевого кадра к третьему:

```
@keyframes growAndGlow {  
  from {  
    background-color: yellow;  
    animation-timing-function: cubic-bezier(1, .03, 1, .115);  
  }  
  
  50% {
```

```
    transform: scale(1.5);
    background-color: blue;
    animation-timing-function: linear;
  }

  to {
    transform: scale(3);
    background-color: red;
  }
}
```

С помощью соответствующего свойства можно задержать начало анимации. Это свойство работает точно так же, как и свойство `transition-delay`, использующееся для переходов, и просто задает до начала анимации период ожидания на указанное количество миллисекунд или секунд. Например, если нужно подождать 1 секунду, прежде чем запустить постепенное появление на экране `div`-контейнера с классом `announcement`, можно переписать стиль `.announcement` следующим образом:

```
.announcement {
  animation-name: fadeIn;
  animation-duration: 1s;
  animation-delay: 1s;
}
```

Добавление задержки к анимации является неплохим способом привлечения внимания и добавления элемента неожиданности странице.

---

#### ПРИМЕЧАНИЕ

Свойства `animation-timing-function` и `animation-delay` требуют префиксов производителей, поэтому не забудьте добавить к своим анимируемым стилям соответствующие префиксы, чтобы получить `-webkit-animation-timing-function`, `-moz-animation-timing-function` и т. д.

---

## Завершение анимации

С помощью CSS можно контролировать еще несколько аспектов анимации, включая ее повторение, направление ее выполнения, если она выполняется более одного раза, а также порядка форматирования браузером элемента по завершении анимации.

Переходы являются анимациями, выполняемыми однократно, например указатель мыши проходит над кнопкой и она увеличивается в размерах. А вот анимации благодаря свойству `animation-iteration-count` могут запускаться один, два и более раз или непрерывно. Если нужно запустить анимацию 10 раз (возможно, в виде десятикратного появления и исчезновения элемента), добавьте к анимируемому стилю следующий код:

```
animation-iteration-count: 10;
```

Обычно браузеры проигрывают анимацию только один раз, и если это все, что вам нужно, то свойство подсчета итераций можно оставить в покое. Если же необходимо, чтобы анимация повторялась непрерывно, для этого свойству `animation-iteration-count` передается одно ключевое слово: `infinite`. Следовательно, для запуска анимации `fadeIn` для `div`-контейнера с классом `announcement` бесконечное количество раз, можно создать следующий стиль:

```
.announcement {  
  animation-name: fadeIn;  
  animation-duration: .25s;  
  animation-iteration-count: infinite;  
}
```

Но такое решение, несомненно, утомит ваших пользователей, поэтому лучше от этого отказаться. Тем не менее подобным стилем можно воспользоваться для простого «пульсирующего» эффекта, при котором вызывается мягкое свечение кнопки **Подпишитесь сегодня** (путем анимации ранее рассмотренного свойства `box-shadow`).

Обычно при многократном запуске анимации веб-браузер запускает анимацию буквально с самого начала. Поэтому, если анимируется превращение желтого фоновой цвета в синий и это повторяется дважды, браузер покажет, как желтый блок превращается в синий, а затем внезапно вернется желтый блок, который превратится в синий еще раз. Этот эффект может слишком сильно резать глаз. Эффект будет выглядеть лучше, если через некоторое время после анимации браузер просто воспроизведет эффект в обратном порядке. Именно так работают переходы: например, когда указатель мыши проходит над элементом, браузер анимирует переход из обычного состояния к состоянию, заданному при проходе над элементом. Когда указатель мыши выходит за пределы элемента, браузер просто воспроизводит анимацию в обратном порядке, возвращая внешний вид элемента в обычное состояние. Чтобы анимация при нечетных проигрываниях шла в прямом, а при нечетных — в обратном порядке, используется свойство `animation-direction` и ключевое слово `alternate`. Например, чтобы элемент постепенно исчезал, а затем снова появлялся, можно создать анимацию `fadeOut`:

```
@keyframes fadeOut {  
  from { opacity: 1; }  
  to { opacity: 0; }  
}
```

А затем можно воспроизвести эту анимацию дважды, меняя при втором воспроизведении ее направление на противоположное:

```
.fade {  
  animation-name: fadeOut;  
  animation-duration: 2s;  
  animation-iteration-count: 2;  
  animation-direction: alternate;  
}
```

Этот код заставляет веб-браузер запустить анимацию `fadeOut` в отношении любого элемента, имеющего класс `fade`. Анимация должна продолжаться 2 секунды, а затем повторяться. Поскольку свойство `animation-direction` имеет значение `alternate`, анимация в первый раз приведет к исчезновению элемента (его свойство непрозрачности от полностью непрозрачного (значение `opacity` равно 1) постепенно поменяется на 0 — полностью прозрачное), а затем при втором проигрывании все пойдет вспять (значение `opacity` будет изменяться от 0 до 1) и элемент снова появится на экране.

---

#### СОВЕТ

Чтобы анимация воспроизводилась несколько раз, но в конечном итоге возвращала элементу его первоначальный вид, используйте четное количество итераций и установите для свойства `animation-direction` значение `alternate`.

---

Неважно, сколько раз вы заставите браузер воспроизводить анимацию, но как только она завершится, браузер отобразит элемент в его первоначальном виде. Предположим, например, что изображение анимируется таким образом, чтобы его размер медленно увеличивался до достижения двойного размера. Как только анимация будет завершена, веб-браузер тут же вернет изображению его первоначальный размер, создавая слишком резкий визуальный эффект. К счастью, браузер можно заставить сохранить элементу тот вид, который у него был по завершении анимации. Для этого нужно установить для свойства `animation-fill-mode` значение `forwards`.

```
animation-fill-mode: forwards;
```

Это свойство применяется к анимируемому элементу вместе с такими свойствами, как `animation-name`, `animation-duration`, и другими свойствами, имеющими отношение к анимации.

## Краткая запись свойства `animation`

Как видите, существует множество свойств анимации, и набор вручную всех этих свойств, да еще и со всеми версиями, включающими префиксы производителей, может привести к болевым ощущениям, вызванным кистевым туннельным синдромом. Поскольку вам по-прежнему нужно пользоваться версиями с префиксами производителей, путь к упрощению ситуации лежит через применение краткой записи свойства `animation`. В одном этом свойстве сочетаются такие свойства, как `animation-name`, `animation-duration`, `animation-timing-function`, `animation-iteration-count`, `animation-direction`, `animation-delay` и `animation-fill-mode`. Можно, например, взять следующий код:

```
.fade {
  animation-name: fadeOut;
  animation-duration: 2s;
  animation-timing-function: ease-in-out;
  animation-iteration-count: 2;
  animation-direction: alternate;
  animation-delay: 5s;
  animation-fill-mode: forwards;
}
```

и придать ему следующий вид:

```
.fade {
  animation: fadeOut 2s ease-in-out 2 alternate 5s forwards;
}
```

Всего одна строка кода вместо семи! Значения свойств нужно перечислять в ранее упомянутом порядке: имя, продолжительность, функция распределения скорости по времени, количество раз, направление, задержка и режим заполнения. Кроме того, нужно поставить между значениями запяты. Обязательными являются только имя и продолжительность. Все остальные значения являются необязательными.

Если к элементу нужно применить несколько анимаций, следует воспользоваться списком их свойств с применением запятой в качестве разделителя. Например, чтобы применить две анимации (скажем, `fadeOut` и `glow`) к элементам, выбираемым стилем `.fade`, напишите следующий код:



```
.fade {  
  animation: fadeOut 2s ease-in-out 2 alternate 5s forwards,  
           glow 5s;  
}
```

Разумеется, при реальном использовании этого свойства нужны еще и версии с префиксами производителей:

```
.fade {  
  -webkit-animation: fadeOut 2s ease-in-out 2 alternate 5s forwards,  
                   glow 5s;  
  -moz-animation: fadeOut 2s ease-in-out 2 alternate 5s forwards,  
                 glow 5s;  
  -o-animation: fadeOut 2s ease-in-out 2 alternate 5s forwards,  
                glow 5s;  
  animation: fadeOut 2s ease-in-out 2 alternate 5s forwards,  
            glow 5s;  
}
```

Выбирайте краткую форму записи, она намного лаконичнее и благосклоннее по отношению к вашим пальцам и клавиатуре.

## Приостановка анимации

В CSS3 имеется еще одно свойство анимации — `animation-play-state`. Оно предназначено для управления проигрыванием анимации. Этому свойству передается одно из двух ключевых слов — `running` или `paused`. Чтобы приостановить анимацию, нужно просто применить к стилю следующее объявление:

```
animation-play-state: paused;
```

Но фактически применить это свойство к CSS можно только с помощью псевдокласса. Как и в случае с переходами, для приостановки анимации нужен инициатор. Один из вариантов предусматривает приостановку анимации при проходе над подвергающимся анимации элементом указателя мыши. В следующем примере используется стиль для класса `fade`:

```
.fade {  
  animation: fadeOut 2s ease-in-out 2 alternate 5s forwards,  
           glow 5s;  
}
```

Этот код запускает две анимации — `fadeOut` и `glow` — для любого элемента, к которому применен класс `fade`. Предположим, что нужно позволить посетителям приостанавливать данную анимацию при проходе над ней указателя мыши. Для этого нужно только добавить еще один стиль:

```
.fade:hover {  
  animation-play-state: paused;  
}
```

Разумеется, вам понадобятся и все версии с префиксами производителей. Более эффективным способом приостановки анимации будет динамическое применение свойства `animation-play-state` к элементу с помощью JavaScript. Можно будет соз-

дать сложную анимацию и добавить кнопку **Пауза**, по щелчку на которой анимация будет приостановлена. Дополнительные сведения о JavaScript и CSS-анимациях даны во врезке «Информация для опытных пользователей. Использование JavaScript для запуска переходов» в подразделе «Распределение скорости выполнения перехода по времени» раздела «Переходы».

## Анимация при проходе указателя мыши над элементом

Все ранее показанные анимации запускаются при загрузках страницы. Но у вас есть еще несколько вариантов запуска CSS-анимации, включая несколько псевдоклассов и использование JavaScript. Чаще всего для анимации применяется псевдокласс `:hover`. С помощью этого псевдокласса можно запустить анимацию при проходе указателя мыши посетителя над любым элементом, например можно придать логотипу элемент динамичности, заставляя его исчезать со страницы и появляться на ней снова. Чтобы применить анимацию к элементу при проходе над ним указателя мыши посетителя, нужно сначала создать анимацию с помощью правила `@keyframes` (шаг 1 в предыдущей последовательности действий). Затем для анимируемого элемента следует создать псевдокласс `:hover`. В стиле для этого псевдокласса просто добавляются свойства анимации (шаг 2 в предыдущей последовательности действий). Теперь анимация запускается, только когда посетитель проводит указатель мыши над элементом.

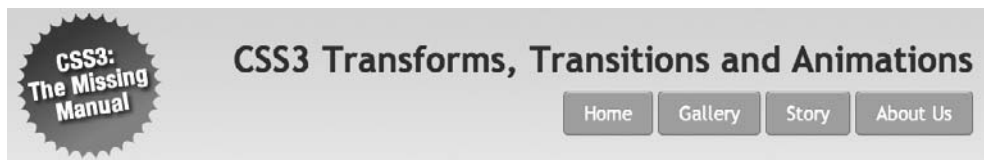
## Обучающий урок

В этом уроке к баннеру будут добавлены преобразования, анимация и переходы.

Сначала нужно загрузить с сайта [www.sawmac.com/css3](http://www.sawmac.com/css3) файлы урока. Перейдите по ссылке и загрузите ZIP-архив с файлами (подробное описание процесса разархивации файлов содержится на указанном сайте). Файлы этого урока содержатся в папке 10.

1. Откройте в текстовом редакторе файл `banner.html`, который находится в каталоге 10.

Страница, находящаяся в этом файле, включает баннер с изображением логотипа, заголовок и набор навигационных кнопок (рис. 10.9). Сначала будет добавлено преобразование, увеличивающее масштаб изображения кнопки при проходе над ней указателя мыши посетителя.



**Рис. 10.9.** Обычный статический баннер, ожидающий оживления с помощью анимации, преобразований и переходов

2. Найдите открывающий и закрывающий теги `<style>` ближе к началу страницы и добавьте в таблицу стилей следующее правило:

```
nav a:hover {
  -webkit-transform: scale(1.2);
  -moz-transform: scale(1.2);
```

```

-o-transform: scale(1.2);
-ms-transform: scale(1.2);
transform: scale(1.2);
}

```

Кнопки на странице находятся в HTML5-элементе `nav`. Данный селектор потомков нацелен на содержимое тега `<nav>` в состоянии прохода над ним указателя мыши (то есть на момент прохода указателя мыши посетителя над ссылкой). Этот стиль применяет функцию `scale`, которая делает кнопку немного крупнее. К сожалению, чтобы код работал в Safari, Chrome, Firefox, Opera и Internet Explorer 9, а также в будущих браузерах, не использующих префиксов производителей, придется применять пять строк кода.

#### ПРИМЕЧАНИЕ

Чтобы пошаговым действиям этого урока было легче следовать, стили помещены во внутреннюю таблицу стилей в веб-странице. Но если хотите, можете добавить этот код к внешней таблице стилей, которая уже применяется к этой странице. (Она называется `styles.css` и находится в той же папке, что и веб-страница.)

3. Сохраните страницу и просмотрите ее в веб-браузере. Проведите указатель мыши над ссылками ниже заголовка.

Когда указатель мыши будет находиться над кнопкой, она станет больше, выделяясь на странице (рис. 10.10). Таким образом, добавляя к кнопке эффект масштабирования в состоянии прохода над ней указателя мыши, можно заставить ее выделиться на странице. Увеличенная кнопка никак не влияет на окружающее ее содержимое (например, она не расталкивает соседние кнопки). В этом заключается уникальность CSS-преобразований.



Рис. 10.10. Выделение кнопки

#### ПРИМЕЧАНИЕ

Internet Explorer 8 и более ранние версии не разбираются в преобразованиях, поэтому на данных браузерах никакого эффекта не будет.

Неплохо, но было бы еще лучше, если бы ко всему этому добавить анимацию.

4. Добавьте перед только что созданным стилем `nav a:hover` еще один стиль:

```

nav a {
  -webkit-transition: all .5s;
  -moz-transition: all .5s;
  -o-transition: all .5s;
  transition: all .5s;
}

```

Здесь опять не обойтись без вариантов с префиксами производителей. (Internet Explorer 9 не понимает преобразований, а поскольку Internet Explorer 10 понимает версию без указания префикса, отпадает надобность в использовании свойства `-ms-transition`.) Этот код предписывает веб-браузеру анимировать все изменения CSS-свойств тега `<a>`, находящегося внутри элемента `<nav>`, и задает продолжительность анимации `.5s`. Чтобы еще больше оживить ситуацию, добавим к состоянию `hover` фоновый цвет.

5. Найдите стиль `nav a: hover` и добавьте к нему объявление `background-color: red`, чтобы стиль приобрел следующий окончательный вид:

```
nav a: hover {
  background-color: red;
  -webkit-transform: scale(1.2);
  -moz-transform: scale(1.2);
  -o-transform: scale(1.2);
  -ms-transform: scale(1.2);
  transform: scale(1.2);
}
```

Если сейчас сохранить страницу и просмотреть ее в браузере, можно увидеть, что кнопки не только становятся крупнее при прохождении над ними указателя мыши, но и приобретают красный фон. Но теперь нужно добавить для переходов разные параметры распределения скорости переходов по времени, чтобы кнопки сначала становились крупнее, а затем приобретали красный фон.

6. Измените стиль `nav a` таким образом, чтобы он включал два перехода: один для преобразования и другой для фонового цвета (изменения выделены полужирным шрифтом):

```
nav a {
  -webkit-transition: -webkit-transform .5s,
                    background-color 1s ease-in .5s;
  -moz-transition: -moz-transform .5s,
                  background-color 1s ease-in .5s;
  -o-transition: -o-transform .5s,
                background-color 1s ease-in .5s;
  transition: transform .5s,
             background-color 1s ease-in .5s;
}
```

Первая часть — `transform .5s` — сообщает браузеру о том, что нужно анимировать любые изменения в свойстве `transform` за полсекунды. Вторая часть — `background-color 1s ease-in .5s` — показывает, что нужно анимировать изменение цвета фона за 1 секунду, используя для распределения скорости перехода по времени метод `ease-in`, но до начала перехода взять паузу на полсекунды. Параметр `.5s` в конце играет важную роль, поскольку он соответствует параметру `.5s` в анимации `transform`. То есть браузер будет ждать, пока не завершится переход `transform`, и только потом приступит к изменению цвета фона.

7. Сохраните страницу и просмотрите ее в браузере.

Проведите указатель мыши над кнопкой. Сначала кнопка станет крупнее. Оставьте указатель над кнопкой, и затем она станет красной. Можно, конечно, перенастроить

порядок перехода, например удалить задержку для перехода `background-color`, и изменение цвета начнется одновременно с изменением размера. Но, так как изменение цвета длится в течение 1 секунды, оно еще будет продолжаться, когда кнопка уже достигнет своего окончательного размера.

## Добавление анимации

Ну а теперь настало время испытать в деле CSS3-анимацию. Начнем с простого вращения и масштабирования изображения логотипа. Первым шагом в создании CSS-анимации станет использование правила `@keyframes` для настройки ключевых кадров анимации. Этот пример начинается с простой анимации, состоящей всего лишь из двух ключевых кадров.

### СОВЕТ

---

Поскольку CSS-анимация требует большого объема кода, лучше будет начать с использования только одного префикса производителя для наиболее часто используемого браузера. Протестируйте страницу в этом браузере, и после доведения ее до совершенства добавьте код для других браузеров.

---

1. Добавьте к внутренней таблице стилей страницы следующий стиль, разместив его после стиля `nav a:hover`:

```
@-webkit-keyframes logo {
  from {
    -webkit-transform: rotate(0) scale(.5);
  }

  to {
    -webkit-transform: rotate(-720deg) scale(1);
  }
}
```

В этом примере используется синтаксис WebKit, следовательно, он будет работать в Chrome и Safari. Если вы в качестве основного пользуетесь браузером Firefox, замените фрагмент `@-webkit-keyframes` фрагментом `@-moz-keyframes`. (Для Opera используйте `@-o-keyframes`, а для Internet Explorer 10 уберите префиксы.)

Данный код сообщает браузеру, что он должен сначала показать элемент без вращения — `rotate(0)`, а также в половинном размере — `scale(.5)`. Затем будут анимированы изменения от состояния `from` до состояния `to` — в данном случае это вращение на  $-720^\circ$ , что означает три оборота против часовой стрелки и возвращение элемента к его нормальному размеру. То есть эта анимация будет вращать элемент и сделает его крупнее.

Теперь нужно применить анимацию к какому-нибудь элементу страницы.

2. Создайте после только что добавленного стиля еще один стиль:

```
.logo {
  -webkit-animation: logo 1s;
}
```

К логотипу на баннере применен класс `logo`, следовательно, этот селектор класса применит анимацию к нему. Свойству `animation` можно передать множество значений, но обязательными являются только имя и продолжительность. Здесь указывается анимация, созданная в последнем действии — `logo` — и веб-браузеру указывается ее продолжительность, равная 1 секунде.

3. Сохраните файл и просмотрите его в Chrome или Safari (или в том браузере, для которого предназначен используемый префикс производителя).

Логотип должен прокрутиться, увеличиться в размерах и остановиться. Если этого не произойдет, перепроверьте код и убедитесь, что просматриваете страницу в соответствующем браузере. Если все идет по плану, эта анимация должна выглядеть весьма привлекательно, но было бы еще лучше, если бы логотип появлялся из-за пределов страницы, а затем останавливался в предназначенном для него месте баннера. Для этого нужно просто анимировать позицию элемента на странице.

4. Отредактируйте правило `@keyframe`, придав ему следующий вид (добавления выделены полужирным шрифтом):

```
@-webkit-keyframes logo {
  from {
    -webkit-transform: rotate(0) scale(.5);
    left: 120%;
  }

  to {
    -webkit-transform: rotate(-720deg) scale(1);
    left: 0;
  }
}
```

Здесь начальная позиция логотипа находится правее баннера и кнопок навигации на краю экрана (вообще-то, левый край логотипа помещается на 1.2-кратную ширину баннера). Чтобы этот код сработал, нужно воспользоваться CSS-свойством `position`. Данное свойство более подробно будет рассмотрено в гл. 15, а сейчас нужно только понять, что CSS дает вам возможность позиционировать любой элемент в любом месте окна браузера и даже за его пределами.

Завершающий ключевой кадр помещает логотип в позицию 0, где он обычно должен появляться на странице, то есть в левой части баннера.

5. Сохраните страницу и просмотрите ее в Chrome или Safari.

Выглядит неплохо. Но можно сделать еще лучше. Добавьте еще один ключевой кадр, чтобы логотип перекатывался на свое место, а затем вращался в обратном направлении по мере приобретения более крупного размера.

6. Измените код, добавленный в шаге 4, придав ему следующий вид:

```
@-webkit-keyframes logo {
  from {
    -webkit-transform: rotate(0) scale(.5);
```

```

    left: 120%
  }

  50% {
    -webkit-transform: rotate(-720deg) scale(.5);
    left: 0;
  }

  to {
    -webkit-transform: rotate(0) scale(1);
  }
}

```

Теперь у вас три ключевых кадра: один в начале, второй точно посередине анимации и третий — в ее конце. Браузер будет анимировать свойства по мере их изменения от кадра 1 к кадру 2 и к кадру 3.

Первый ключевой кадр такой же, как и созданный в шаге 4: логотип не прокручен, имеет половинный размер и помещен за пределами правой стороны. Второй ключевой кадр сохраняет логотип в том же размере, три раза его прокручивает и перемещает в левую сторону баннера. И наконец, когда логотип встанет на свое место, браузер увеличит его масштаб, вращая назад с позиции  $-720^\circ$  в позицию  $0^\circ$ , то есть логотип теперь будет вращаться три раза по часовой стрелке.

#### 7. Сохраните страницу и просмотрите ее в Chrome или Safari.

Логотип должен прокатиться с правой стороны баннера в левую, остановиться, увеличиться в размерах и повернуться по часовой стрелке. Если этот код не сработает, перепроверьте его на соответствие коду, показанному выше. К сожалению, анимация проигрывается немного быстрее желаемого. Но это легко исправить.

#### 8. Отредактируйте стиль `.logo` так, чтобы анимация продолжалась не одну, а три секунды:

```

.logo {
  -webkit-animation: logo 3s;
}

```

А теперь приступим к самой трудоемкой части — заставим анимацию работать в других браузерах, добавив все версии с префиксами производителей. Сначала нужно дополнить правило `@keyframes`.

#### 9. Добавьте к правилу `@keyframes` версии с префиксами производителей, чтобы код приобрел следующий вид:

```

@-webkit-keyframes logo {
  from {
    -webkit-transform: rotate(0) scale(.5);
    left: 120%
  }
}

```

```
50% {
  -webkit-transform: rotate(-720deg) scale(.5);
  left: 0;
}

to {
  -webkit-transform: rotate(0) scale(1);
}

}

@-moz-keyframes logo {
  from {
    -moz-transform: rotate(0) scale(.5);
    left: 120%
  }

  50% {
    -moz-transform: rotate(-720deg) scale(.5);
    left: 0;
  }

  to {
    -moz-transform: rotate(0) scale(1);
  }

}

@-o-keyframes logo {
  from {
    -o-transform: rotate(0) scale(.5);
    left: 120%
  }

  50% {
    -o-transform: rotate(-720deg) scale(.5);
    left: 0;
  }

  to {
    -o-transform: rotate(0) scale(1);
  }

}

@keyframes logo {
  from {
    transform: rotate(0) scale(.5);
    left: 120%
  }

  50% {
```



```

    transform: rotate(-720deg) scale(.5);
    left: 0;
  }

  to {
    transform: rotate(0) scale(1);
  }
}

```

Код имеет довольно большой объем, но, к сожалению, пока для использования CSS-анимации без этого не обойтись. Когда-нибудь в прекрасном будущем все браузеры станут применять единое правило `@keyframes` без всяких префиксов и такое же не имеющее префиксов свойство `transform`. Но пока придется набирать весь этот код.

К счастью, добавления к правилу `@keyframes` являются самой тяжелой частью работы. Добавление анимации к стилю `.logo` потребует чуть меньшего объема работы.

10. Отредактируйте стиль `.logo`, добавив к нему три дополнительных строки:

```

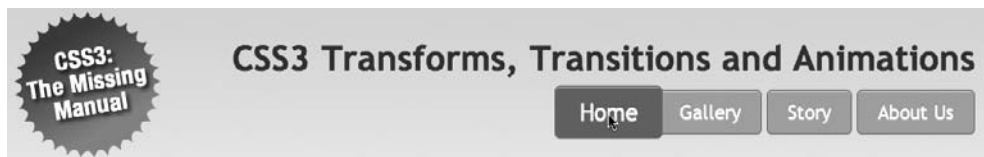
.logo {
  -webkit-animation: logo 3s;
  -moz-animation: logo 3s;
  -o-animation: logo 3s;
  animation: logo 3s;
}

```

11. Сохраните страницу и проверьте ее работу в Firefox, Chrome, Safari и Opera.

Логотип должен прокатываться по странице и увеличиваться во всех браузерах (рис. 10.11). Разумеется, в Internet Explorer 9 и более ранних версиях никакой анимации не будет. Но логотип все же будет помещен в нужное место на странице, не нарушая ее внешнего вида. Окончательную версию урока можно найти в папке `10_finished`.

Чтобы продолжить усовершенствование страницы, добавьте анимацию, заставляющую одну из кнопок, над которой установлен указатель мыши, светиться. (Подсказка: создайте анимацию, осуществляющую циклический переход через различные тени блока. Для добавления тени, размещающейся внутри блока, воспользуйтесь значением `inset`. Затем добавьте анимацию к стилю `nav a: hover`, чтобы она проигрывалась только при проходе указателя мыши над кнопкой.)



**Рис. 10.11.** Книжки способны на многое, но, к сожалению, они не могут показать ту впечатляющую анимацию, которая только что была создана. Так должна выглядеть страница в ходе воспроизведения анимации и при проходе указателя мыши над кнопкой Home (Домой)

# 11 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, применяя 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 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. 11.1).

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. 11.2.

```
<table>
```

```
  <caption align="bottom">
```

```
    Table 1: CosmoFarmer.com's Indoor Mower Roundup
```

```
  </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>
  </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>`. Вообще-то, чем меньше HTML-кода тем лучше, и все эти теги вам не нужны: вполне можно обойтись только тегами `<table>`, `<tr>` и `<td>` (и в большинстве случаев также тегом `<th>`). Но различные табличные теги предоставляют множество полезных зацепок для привязки к ним CSS-стилей. Заголовков 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>`, а тег `<colgroup>` можно применять в качестве простого средства установки ширины столбца таблицы. 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/](http://www.456bereastreet.com/archive/200410/bring_on_the_tables/).

---

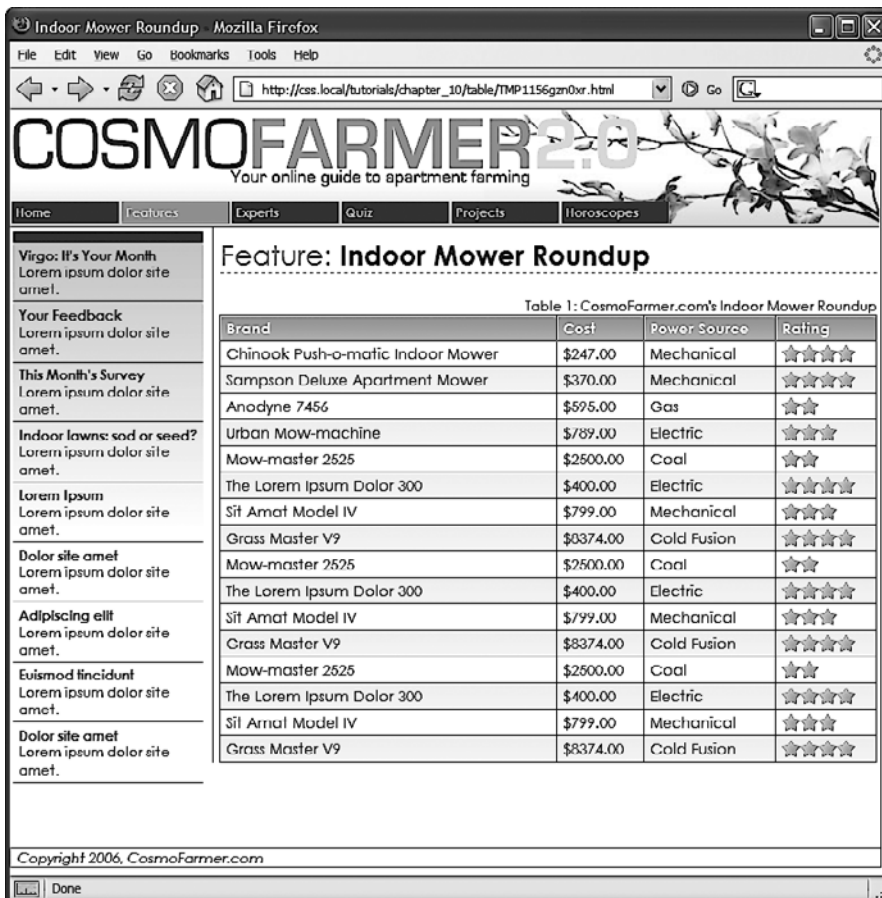


Рис. 11.1. Благодаря CSS в этой таблице, содержащей информацию о домашних газонокосилках, шрифты изменяются на более привлекательные, создаются границы и меняются фоновые цвета, но вся лежащая в основе структура создана с помощью HTML

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.

Table 1: CosmoFarmer.com's Indoor Mower Roundup

Рис. 11.2. Заголовок Price говорит, что вы найдете стоимость каждой из газонокосилок в ячейках снизу. Реальные данные хранятся в таблице в тегах <td>

## Создание стилей для таблиц

Вы можете использовать многие свойства CSS, о которых уже прочитали, чтобы добавить привлекательность 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 особо 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. Применение 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 разделе «Управление размерами полей и отступов» гл. 7, отступ — это расстояние между границей элемента и его содержимым. Вы можете применять отступы, чтобы обеспечить немного свободного места между краями абзацев с текстом и их границами. Когда речь идет о таблицах, границы — это *края* ячеек и при использовании отступов добавляется свободное место вокруг содержимого этой ячейки таблицы (см. рис. 11.2). Это работает так же, как и старый 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 указать отступ размером 10 пикселей для всех ячеек таблицы, следует воспользоваться таким стилем:

```
td, th { padding: 10px; }
```

Вы можете также контролировать отступы от каждой из четырех границ ячейки. Чтобы добавить 10 пикселей места сверху для каждой ячейки с данными в таблице, 3 пикселя снизу и по 5 пикселей 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 линейных и блочных элементов» раздела «Управление размерами полей и отступов» гл. 7). Чтобы получить больше информации, посетите страницу [http://developer.mozilla.org/en/docs/Images,\\_Tables,\\_and\\_Mysterious\\_Gaps](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. 11.3). Это унаследованное свойство. Если вы хотите выровнять содержимое всех ячеек таблицы по правому краю, создайте следующий стиль:

```
table { text-align: right; }
```

left	center	right	justified
<b>Brand</b>	<b>Price</b>	<b>Power Source</b>	<b>Mini-Review</b>
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.

Table 1: CosmoFarmer.com's Indoor Mower Roundup

**Рис. 11.3.** 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; }` выравнивает и заголовки таблицы.

Свойство CSS `text-align` по отношению к ячейкам таблицы работает так же, как и атрибут `align` тега `<td>`. Но использования HTML-атрибута `align` следует избегать, поскольку в HTML5 он не работает, а CSS позволяет 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. 11.4). Вы можете изменить это с помощью свойства `vertical-align`. Примените 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 (см. рис. 11.4). Скорее всего, вы даже не заметите тонкостей работы параметра `baseline`. В отличие от `text-align` свойство `vertical-align` не унаследовано, поэтому вы можете задействовать его только в стилях, которые применяются прямо в тегах `<th>` и `<td>`.

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.	top
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.	baseline
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.	middle
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.	bottom

Table 1: CosmoFarmer.com's Indoor Mower Roundup

**Рис. 11.4.** Свойство CSS `vertical-align` — эквивалент ныне устаревшего 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

## СОВЕТ

Итак, изученное форматирование таблиц применяется сразу ко всем вашим таблицам. Когда же вы хотите применить для каждой таблицы 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 селекторов потомков 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.

## Создание границ

Свойство CSS `border` (см. раздел «Добавление границ» гл. 7) работает с таблицами так же хорошо, как и с другими элементами, но вам нужно кое-что помнить. Во-первых, применение 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. 11.5. Чтобы правильно управлять границами, вам необходимо понять принцип отрисовки ячеек таблицы веб-браузерами и разобраться с CSS-свойством `border-collapse`.

**Управление промежутком между ячейками таблицы.** Если не указано другое, браузеры расставляют в таблице интервалы между ячейками около 2 пикселей. Этот промежуток действительно заметен, когда вы задаете границы для ячеек.

В CSS доступно свойство `border-spacing`, которое можно использовать, чтобы контролировать размер этого промежутка. Это свойство нужно применять к самой таблице, а если требуется удалить постое пространство, обычно добавляемое браузером между ячейками, установите для свойства `border-spacing` значение 0:

```
table {
  border-spacing: 0;
}
```

Но если вы предпочитаете иметь пустое пространство между ячейками, то его можно добавить:

```
table {
  border-spacing: 2px;
}
```

**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. 11.5, *посередине*). Лучший способ избавиться от этого (а также 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 (см. рис. 11.5, *внизу*). Используйте значение `collapse` в стилях для форматирования таблиц:

```
table { border-collapse: collapse; }
```

---

#### ПРИМЕЧАНИЕ

Если установить для свойства `border-collapse` значение `collapse`, свойство `border-spacing` работать не будет.

---

- **Скругленные углы.** Чтобы добавить к ячейкам таблиц (но не к самим таблицам) скругленные углы, можно воспользоваться свойством `border-radius`. Например, если нужно добавить ячейкам таблицы контуры и скругленные углы, можно создать следующий стиль:

```
td {
  border: 1px solid black;
  border-radius: 5px;
}
```

Учтите, что при установке для свойства `border-collapse` значения `collapse` браузеры проигнорируют установку для ячеек таблицы свойства `border-radius` и просто нарисуют обычные прямые углы.

---

#### ПРИМЕЧАНИЕ

Таблицам и ячейкам можно даже назначать свойство `box-shadow`. Если применить это свойство к таблице, тень появится за пределами всей таблицы, а если его задавать отдельным ячейкам, то собственная отбрасываемая тень будет у каждой 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.

Table 1: CosmoFamer.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: CosmoFamer.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: CosmoFamer.com's Indoor Mower Roundup

**Рис. 11.5.** Браузеры обычно вставляют пробелы между всеми ячейками в таблице

## Применение стилей к строкам и столбцам

Добавление чередования затемненных и незатемненных строк в таблицу, как на рис. 11.6, — 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 счастью, CSS предлагает способ, позволяющий это сделать. Используя ранее рассмотренный селектор `nth-of-type`, можно добавить четным и нечетным строкам разный фон:

```
tr:nth-of-type(odd) { background-color: red; }
tr:nth-of-type(even) { background-color: blue; }
```

Если не нужно применять один и тот же фон к чередующимся строкам всех таблиц, следует просто добавить к целевой таблице имя класса (например, `products` для таблицы с информацией о товарах), а затем воспользоваться селектором потока:

```
.products tr:nth-of-type(odd) { background-color: red; }
.products tr:nth-of-type(even) { background-color: blue; }
```

Вы также не ограничены цветами. Можно использовать фоновые изображения (см. раздел «Добавление фоновых изображений» гл. 8) или даже линейные градиенты (см. раздел «Линейные градиенты» гл. 8) для создания более привлекательного вида, например легкого затемнения строки таблицы с заголовками, как на

рис. 11.6 (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, применяя для каждой определенный стиль и вид, например `<td class="price">`.



Рис. 11.6. Благодаря чередующемуся изменению фонового цвета строк намного проще следить за всеми данными из строки

#### ПРИМЕЧАНИЕ

Селектор `nth-of-type` в Internet Explorer 8 и более ранних версиях не поддерживается.

В процессе форматирования столбцов нужно 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.

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, можете пропустить все атрибуты HTML и просто применить к столбцам стиль, используя стиль, назначенный тегу `<col>`. Скажем, у вас есть следующий кусок кода HTML: `<col class="price">`. Вы можете добавить этот стиль к таблице стилей и установить ширину каждой ячейки в данном столбце равной 200 пикселям:

```
.price { width: 200px; }
```

Чтобы выделить столбец, вы можете воспользоваться свойствами фона. И снова считайте, что у вас есть тег `<col>`, к которому применен класс `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. Передавая информацию в форму, вы можете подписаться на новости, поискать какие-либо товары в базе данных, ввести изменения в профиль на “Фейсбуке” или заказать набор конструкторов «Звездные войны», о котором уже давно мечтали. Совсем не обязательно, чтобы ваши формы имели такой же вид, 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 применять практически любое из описанных в этой книге свойств и для элемента типа «форма». Результаты тем не менее могут быть различными (рис. 11.7).

Internet Explorer 9 (см. рис. 11.7, *вверху*) не поддерживает скругленные углы или тени текстовой области (Text area) (см. текстовую область на панели Borders (Границы) справа). Он также не понимает линейный градиент, примененный к кнопке Submit (Отправить) на средней панели Backgrounds (Фон). Различия между Chrome (второй сверху), Firefox (второй снизу) и Safari (внизу) не столь заметны, но между ними все же есть несколько визуальных несоответствий. Лучше всего тщательно прорабатывать свои формы и не ожидать их одинакового внешнего вида в каждом браузере.

В следующем разделе будут рассмотрены свойства, которые лучше всего работают с тегами формы, а также перечислены браузеры, которые правильно интерпретируют эти свойства.

### 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. 11.7), 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 в форме, вы можете добиться того, что пользователь просто не будет обращать на него внимания и будет его пропускать (см. примеры на рис. 11.7, второй снизу справа 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.

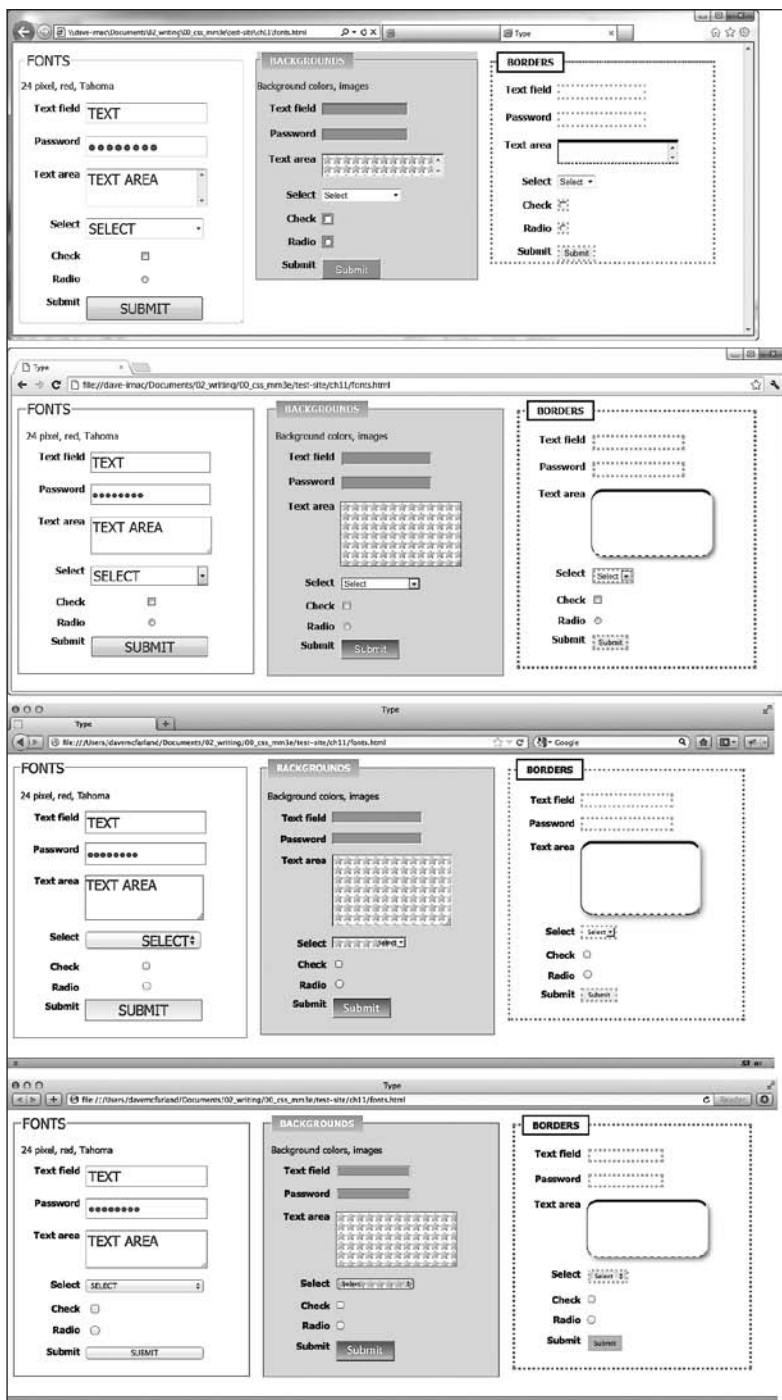


Рис. 11.7. Возможности форматирования полей ввода с помощью стилей варьируются от браузера к браузеру

## Элементы HTML-форм

Большой выбор HTML-тегов позволяет вам создавать формы. Некоторые из них форматировать проще (например, текстовые поля), некоторые — сложнее (переключатели). 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 среднюю панель верхнего изображения на рис. 11.7). При использовании отступов появляются пробелы между верхними границами тегов `<fieldset>` и их содержимым. Хотя 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="password">` и `<text-area>` 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, фоновые цвета и изображения. Можно задать ширину этих полей с помощью CSS-свойства `width`. Однако свойство `height` поддерживает только тег `<textarea>`.

**Кнопки.** Кнопки в форме, такие как `<input type="submit">`, позволяют вашим пользователям передавать данные в форму, очищать содержимое формы или предпринимать какие-либо другие действия. Многие браузеры позволяют развить фантазию в форматировании текста, рамок, оформления фона, теней и скругленных углов. Вы также можете выравнивать текст на кнопке по левому или правому краю либо посередине с помощью свойства `text-align`. Особенно хорошо смотрятся на кнопках линейные градиенты.

**Раскрывающиеся списки.** Списки, созданные с помощью тега `<select>`, также можно форматировать, используя стили. Большинство браузеров позволяют вам устанавливать фоновый цвет, изображение и границы. Но фоновые цвета и изображения не всегда добавляются к раскрывающимся спискам, когда они раскрываются для показа всех вариантов выбора.

**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 пикселей вполне подходит для текстового поля, скорее всего, вам не захочется, чтобы поле для установки флажков было таким же большим, поэтому вы не сможете применить тег `<input>` для изменения ширины. Самым лучшим, учитывая кросс-браузерность, 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.

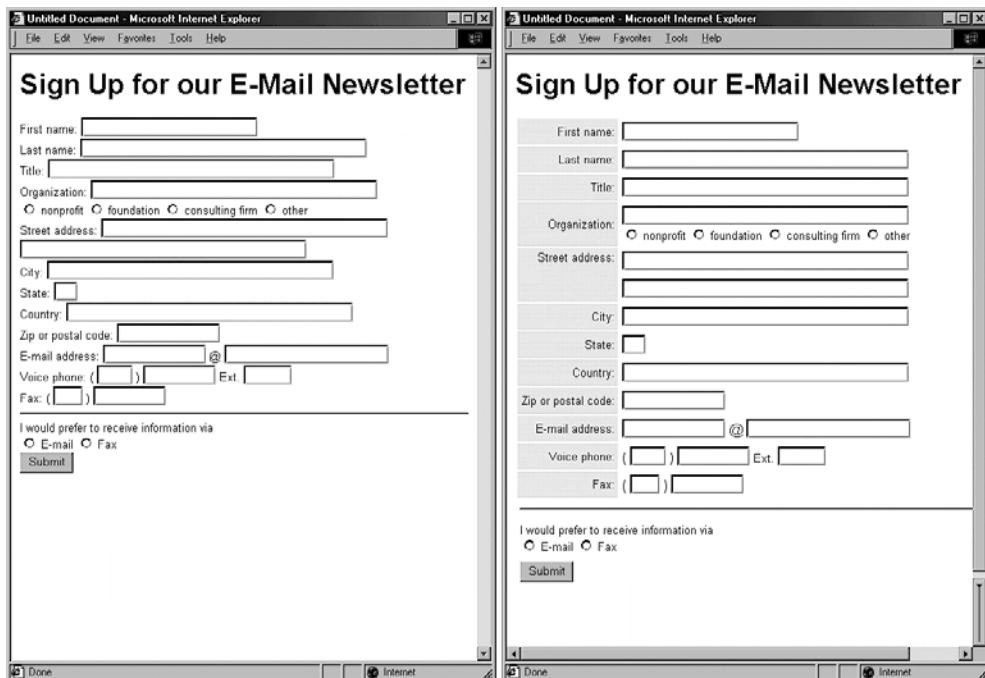
Селекторы атрибутов понимают все используемые в настоящее время браузеры, даже Internet Explorer 7, поэтому применять их для придания стиля своим формам можно вполне свободно.

## 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. 11.8, *слева*). Формы обычно выглядят лучше, когда запросы и поля ввода расположены в виде столбца (см. рис. 11.8, *справа*).

Вы можете добиться этого несколькими путями. Наиболее простой способ — таблицы HTML. Хотя поля ввода и надпись — это не просто набор данных для таблицы, они хорошо приспособляются к расположению в формате столбцов и строк. Просто разместите 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. 11.8 (как преимущество — меньше кода на HTML). Рассмотрим простой вариант этого.

1. Поместите 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 применять для создания такого вопроса? Вам следует обратиться к тегу `<span>`: `<span>Какой ваш любимый цвет?</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. 11.8 это были бы метки для First name (Имя), Last name (Фамилия) и т. д., но не теги `<label>` для переключателей).



**Рис. 11.8.** Иногда элементы формы не очень хорошо компонуются с текстом, что приводит к тому, что они располагаются зигзагом (*слева*). Решением данной проблемы может стать размещение элементов в столбец (*справа*)

## ВНИМАНИЕ

Посетите страницу [www.htmldog.com/guides/htmladvanced/forms/](http://www.htmldog.com/guides/htmladvanced/forms/), чтобы получить быстрый обзор тега `<label>`.

### 2. Установите значение `inline-block` для свойства `display` и ширину.

Обычно теги `<label>` и `<span>` являются линейными элементами, игнорирующими многие настройки, доступные блочным элементам, включая `width`, `height` и `text-align`. Но если превратить надпись (`label`) в линейный блок (`inline-block`), она по-прежнему будет располагаться рядом с полем формы (оправдывая свою линейную — `inline` — часть свойства). Значение `width` предоставит достаточное пространство, позволяющее поместиться всей надписи в одну строку, если это возможно. Можно создать стиль класса, который будет иметь следующий вид:

```
.label {
  display: inline-block;
  width: 20em;
}
```

Настройки `width` и `inline-block` превращают надписи с классом `label` в небольшие, равные по размеру блоки и предоставляет четкий левый край, по которому происходит выравнивание всех полей формы.



3. Настройте стиль. Есть еще несколько улучшений, которые вы можете использовать для завершения работы. Вам нужно установить значение `top` для свойства `vertical-align`, чтобы верхняя кромка текста надписи выровнялась с верхней кромкой поля формы. Нужно также выровнять текст надписи по правому краю, чтобы каждая надпись появлялась рядом с каждым полем формы. И наконец, добавив немного правого поля, можно создать небольшой промежуток между надписями и полями формы.

```
.label {
display: inline-block;
width: 20em;
vertical-align: top;
text-align: right;
margin-right: 15px;
}
```

В итоге получится аккуратная форма. 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.

## ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

### Псевдоклассы формы

Существует несколько псевдоклассов, предназначенных для стилизации формы. Псевдокласс `:focus` позволяет создать селектор, изменяющий внешний вид текстового поля при щелчке на нем кнопкой мыши или при переходе на него с помощью клавиши табуляции (это называется получением фокуса). Этим можно воспользоваться для изменения размера, цвета фона, шрифта и других применяемых к полю свойств CSS. Практический пример будет показан в следующем обучающем уроке.

Псевдокласс `:checked` работает с переключателями и флажками. Он предназначен для стилизации этих элементов, но обычно веб-браузеры в отношении внешнего вида этих полей ведут себя слишком сдержанно и большинство CSS-свойств к ним не применяются. Но, если хотите, чтобы они попробовали внести какие-то изменения, можно просто создать стиль с именем `:checked` и добавить к нему какие-нибудь свойства CSS. Все указанное в этом стиле будет применяться только к переключателям и флажкам формы. Если нужно придать стиль конкретному флажку при его установке, можно создать стиль класса:

```
.special:checked
```

А затем применить этот класс только к этому флажку:

```
<input type="checkbox" class="special" ...
```

Элементы формы можно также делать активными и неактивными. Неактивное поле изменить невозможно: например, нельзя ввести текст в неактивное текстовое поле или изменить состояние неактивного флажка. Изменить состояние элемента формы (допустим, с активного на неактивное) можно только с помощью JavaScript, поэтому для получения пользы от псевдоклассов `:enabled` или `:disabled` нужно научиться работать с JavaScript. И тем не менее эти псевдоклассы могут оказаться весьма полезными.

Предположим, например, что у вас есть форма заказа с адресными полями «Выставление счета» и «Доставка». Если рядом с адресом доставки поставить флажок «Такой же, как и адрес выставления счета», посетитель может установить этот флажок. Используя JavaScript, вы сможете сделать неактивным поле адреса доставки, а чтобы в него нельзя было случайно что-нибудь ввести, стиль его цвета можно изменить на серый:



### ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

```
:disabled {
  background-color: #333;
}
```

Существуют и другие псевдоклассы, предназначенные для работы с некоторыми специальными свойствами формы HTML5, такими как встроенная проверка пра-

вильности заполнения формы или с другими специальными свойствами. Чтобы узнать об этих псевдоклассах, посетите веб-страницу <http://html5doctor.com/css3-pseudo-classes-and-html5-forms/>. Но имейте в виду, что многие веб-браузеры пока не поддерживают свойства формы, введенные в HTML5.

## 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 разделе «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 применяемый 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/css3/](http://www.sawmac.com/css3/). Файлы для этого учебного урока находятся в папке 11\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. 11.9). 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.

2. Откройте файл `table.html` в текстовом редакторе.

Для начала создадим стиль, который задает ширину таблицы и используемый шрифт. К этой таблице применим класс `inventory`, так что вы можете применить селектор класса 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%;
  margin-top: 25px;
}
```

Если вы не установите ширину таблицы, она растянется по размерам содержимого. В данном случае мы зададим ширину, равную 100 %, чтобы таблица растянулась по всей ширине ее контейнера (в данном случае это сам тег `<body>`

с установленной шириной и автоматическими левыми и правыми полями для его центровки в окне браузера). При изменении шрифта в теге <table> используется наследование, чтобы изменился шрифт и во всех остальных тегах внутри страницы — <caption>, <th>, <td> и т. д. И наконец, свойство margin-top сдвигает таблицу вниз от находящегося выше нее заголовка.

	Table 1: Current Inventory	<caption>
Product	Price	Rating
Quam Lorem	\$19.95	★★★★★
pus Velit	\$14.55	★★★
Ipsum Dolor Sat	Priceless	★★★★★
elis Fringilla	\$29.95	★★
iem Pharetra	\$75.99	★★★
icibus Elit	\$82.00	★
dipiscing Vitae	\$1.95	★★★★★
Orci Ante	\$17.95	★★★★★
itis Non Adipiscing	\$44.00	★★★★★

| id="product">    <col id="price"><col id="rating">

(шапка таблицы)  
 (ячейки таблицы)

Table 1: Current Inventory		
PRODUCT	PRICE	RATING
Quam Lorem	\$19.95	★★★★★
pus Velit	\$14.55	★★★
Ipsum Dolor Sat	Priceless	★★★★★
elis Fringilla	\$29.95	★★
iem Pharetra	\$75.99	★★★
icibus Elit	\$82.00	★
dipiscing Vitae	\$1.95	★★★★★
n Orci Ante	\$17.95	★★★★★
itis Non Adipiscing	\$44.00	★★★★★

**Рис. 11.9.** 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)

Далее создадим стиль для заголовка таблицы.

4. Добавьте еще один стиль после только что созданного для таблицы:

```
.inventory caption {
  text-align: right;
  font-size: 1.3em;
  margin-bottom: 10px;
}
```

Этот селектор потомка влияет только на тег <caption>, который появляется внутри другого тега с классом 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. Нижнее поле добавляет немного пространства между заголовком и таблицей.

Когда вы читаете одну из строк в таблице, очень легко сбиться. Необходим хороший визуальный ориентир. Если мы добавим границы вокруг ячеек, они визуально выделяют 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. 11.10, 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. 11.10.** 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%;
margin-top: 25px;
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 пикселям. 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 `.inventory td, .inventory th`; он будет применяться 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 применять 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 отступов, теги `<th>` сохраняют 7 пикселей каждого из отступов, заданных 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 к фону линейный градиент и изменив цвет шрифта:

```
.inventory th {
  text-transform: uppercase;
  text-align: left;
```

```
padding-top: 5px;
padding-bottom: 4px;
background: rgb(229,76,16);
background: -webkit-linear-gradient(rgb(229,76,16), rgb(173,54,8));
background: -moz-linear-gradient(rgb(229,76,16), rgb(173,54,8));
background: -o-linear-gradient(rgb(229,76,16), rgb(173,54,8));
background: linear-gradient(rgb(229,76,16), rgb(173,54,8));
color: white;
}
```

Сначала к фону добавляется простой цвет в формате RGB, благодаря этому Internet Explorer 9 и более ранние версии (которые не работают с градиентами) получают по крайней мере сплошной цвет. В следующих трех строках добавляются версии задания линейных градиентов с префиксами производителей, а во второй строке снизу линейный градиент задан с использованием официального синтаксиса. И наконец, цвет текста меняется на белый.

Когда в таблице хранится слишком много данных, расположенных во множестве строк и столбцов, иногда бывает трудно определить, к какой строке какие данные относятся. К счастью, упростить указание на чередующиеся строки можно с помощью селектора `nth-of-type`.

10. Добавьте еще один стиль во внутреннюю таблицу стилей вашей веб-страницы:

```
.inventory tr:nth-of-type(even){
  background-color: rgba(255,255,255,.1);
}

.inventory tr:nth-of-type(odd){
  background-color: rgba(229,76,16,.1);
}
```

Как уже говорилось, селектор `nth-of-type` используется для выбора дочерних элементов, отвечающих той или иной числовой схеме, например каждый пятый абзац. В данном случае первый стиль выбирает каждый четный тег `<tr>`, а второй — каждый нечетный тег `<tr>`.

И наконец, нужно подогнать ширину ячеек, которые находятся ниже, в столбцах **Price** (Цена) и **Rating** (Рейтинг). Один из способов заключается в методичном добавлении к этим ячейкам имен классов и создании стиля класса с установленной шириной ячейки. Но лучше будет воспользоваться тегом `<col>`, что позволит назначить класс или идентификатор всей совокупности ячеек столбца. На рис. 11.9 показано, что эти два столбца имеют идентификаторы `price` и `rating`. С помощью одного группового селектора можно упростить задание ширины для этих двух столбцов.

11. Добавьте еще один стиль к внутренней таблице стилей страницы:

```
#price, #rating {
  width: 15%;
}
```

Откройте страницу в браузере. Она должна выглядеть так же, как и страница, изображенная на рис. 11.9, *внизу*. Вы также можете найти готовый пример в папке `11_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 `11\form`, то увидите, что на ней есть простая форма для подписки на вымышленном сайте `CosmoFarmer.com` (рис. 11.11). 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. 11.11.** Тег `<table>` обычно используется для размещения вопросов в форме. Можно также применять 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: 1.2em;
  color: white;
  font-family:Tahoma, Geneva, sans-serif;
}
```

У этой формы есть класс — `subForm`, поэтому новый стиль изменяет размер текста, цвет и шрифт для всего текста между открывающим и закрывающим тегами `<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 {
display: inline-block;
width: 200px;
vertical-align: top;
}
```

Этот селектор потомка используется для всех элементов класса `.label`, расположенных в форме. Стилль превращает надписи из линейных элементов (которые не воспринимают значения ширины) в линейно-блочные элементы. Настройка ширины устанавливает для области надписей ширину 200 пикселей, а настройка вертикального выравнивания гарантирует выравнивание текста надписей по верхнему краю соседствующих с ними полей формы. В результате после применения этого стиля для каждого вопроса из формы вы получите выровненный по ширине столбец. Чтобы оценить полученный результат, вам необходимо применить созданный класс к соответствующим элементам формы.

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 в форме.

5. Повторите шаг 4 для следующих фрагментов HTML: `<label for="email">`, `<label for="refer">`, `<label for="comments">`.

В форме есть еще один дополнительный вопрос — **Rate your apartment farming skills** (Оцените ваш опыт). 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** (Оцените ваш опыт) и поместите его в тег `<span>`, использующий класс `label`:

```
<span class="label">Rate your apartment farming skills</span>
```

Теперь вопросы размещены в отдельном столбце (рис. 11.12, *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:

```
.subform .label {
display: inline-block;
width: 200px;
vertical-align: top;
text-align: right;
margin-right: 10px;
font-weight: bold;
color: rgba(255,255,255,.5);
}
```

Просмотрите страницу в браузере. Форма должна выглядеть так же, как и форма, изображенная на рис. 11.12, *внизу*.

**Рис. 11.12.** Выделив вопросы в форме полужирным шрифтом, а также выровняв их в соответствии с остальными элементами формы (внизу), мы получили форму, которая выглядит намного лучше

Форма выглядит уже намного лучше, но кнопка **Subscribe** (Подписаться), размещенная у левой границы, смотрится не на своем месте. Выравниваем ее, как и остальные элементы формы.

8. Добавьте еще один стиль во внутреннюю таблицу стилей.

```
.subform input[type="submit"] {
  margin-left: 220px;
}
```

Поскольку кнопки отправки создаются путем добавления к тегу `<input>` атрибута `type="submit"`, указать на них можно с помощью простого использования селектора атрибута. Тогда не придется создавать стиль класса и применять его к кнопкам отправки.

9. Измените только что созданный стиль кнопки **Subscribe** (Подписаться), добавив дополнительные свойства:



```
.subform input[type="submit"] {
  margin-left: 220px;
  padding: 10px 25px;
  font-size: 1em;
  color: white;
}
```

Свойство `padding` добавляет пространство между текстом внутри кнопки и ее краями, а свойства `font-size` и `color` влияют на текст кнопки.

Теперь можно проявить изобретательность и добавить на кнопку градиент.

#### 10. Отредактируйте стиль кнопки **Submit** (Отправить) еще раз:

```
.subform input[type="submit"] {
  margin-left: 220px;
  padding: 10px 25px;
  font-size: 1em;
  color: white;
  background: rgb(0,102,153);
  background: -webkit-linear-gradient(rgba(255,255,255,.1) 40%,
    rgba(255,255,255,.5));
  background: -moz-linear-gradient(rgba(255,255,255,.1) 40%,
    rgba(255,255,255,.5));
  background: -o-linear-gradient(rgba(255,255,255,.1) 40%,
    rgba(255,255,255,.5));
  background: linear-gradient(rgba(255,255,255,.1) 40%, rgba(255,255,255,.5));
}
```

Здесь установлен стандартный цвет фона (для Internet Explorer 9 и более ранних версий), а затем установлен линейный градиент. В нем используются две цветовые опорные точки. Первый цвет простирается от верхнего края до 40% внутрь кнопки, затем начинается градиент (градиенты и цветовые опорные точки рассматриваются в разделе «Использование градиентных фонов» гл. 8).

И наконец, подправьте внешний вид еще раз. Удалите стандартную рамку, скруглите углы и задайте эффект сияния вокруг блока.

#### 11. Отредактируйте стиль кнопки **Submit** (Отправить) в последний раз (изменения выделены полужирным шрифтом):

```
.subform input[type="submit"] {
  margin-left: 220px;
  padding: 10px 25px;
  font-size: 1em;
  color: white;
  background: rgb(0,102,153);
  background: -webkit-linear-gradient(rgba(255,255,255,.1) 40%,
rgba(255,255,255,.5));
  background: -moz-linear-gradient(rgba(255,255,255,.1) 40%,
rgba(255,255,255,.5));
  background: -o-linear-gradient(rgba(255,255,255,.1) 40%,
rgba(255,255,255,.5));
  background: linear-gradient(rgba(255,255,255,.1) 40%, rgba(255,255,255,.5));
}
```

```
border: none;
border-radius: 5px;
box-shadow: 0 0 4px white;
}
```

Задание для свойства `border` значения `none` удаляет рамку, которую браузер обычно устанавливает вокруг кнопки, а свойство `border-radius` приводит к скруглению углов кнопки. И наконец, добавление отбрасываемой тени без горизонтального или вертикального смещения (имеется в виду часть `0 0`), позволяет получить эффект свечения элемента, которое выглядит как легкая белая подсветка, исходящая из-под кнопки.

#### ПРИМЕЧАНИЕ

Усовершенствование конструкции кнопки можно продолжить. Попробуйте создать для нее стиль-ролlover — `.subform input[type="submit"]:hover` — и измените фоновый цвет. Используя информацию предыдущей главы, вы можете даже анимировать этот переход!

Теперь текстовые надписи и кнопка **Subscribe** (Подписаться) смотрятся великолепно, но зачем на этом останавливаться? Настало время оживить поля формы. Начнем с изменения их шрифта и фонового цвета.

12. Добавьте стиль для размещенного в форме списка:

```
.subform select {
  font-size: 1.2em;
}
```

Этот стиль немного укрощает текст. Можно выбрать шрифт, добавить цвет фона и внести другие изменения. Но некоторые браузеры (например, Safari) не позволяют вносить существенные изменения в стиль раскрывающихся списков, поэтому все внесенные в них стилевые изменения нужно тестировать.

А теперь пришла пора внести изменения в текстовые поля.

13. Создайте новый групповой селектор для трех полей ввода в форме:

```
.subform input[type="text"], .subform textarea {
  border-radius: 5px;
  border: none;
  background-color: rgba(255,255,255,.5);
  color: rgba(255,255,255,1);
  font-size: 1.2em;
  box-shadow: inset 0 0 10px rgba(255,255,255,.75);
}
```

Этот групповой стиль выбирает все элементы ввода, имеющие тип `text`, а также многострочные текстовые поля (тег `<textarea>`). В нем добавляются различные свойства, с которыми вы уже должны быть знакомы, такие как `border-radius`, `background-color`, `font-size` и `box-shadow`. Текстовые поля выглядят мелко, поэтому для них устанавливается ширина и к ним добавляются небольшие отступы.

14. Отредактируйте только что созданный стиль, изменив ширину и применив отступы (изменения выделены полужирным шрифтом):

```
.subform input[type="text"], .subform textarea {
  font-size: 1.2em;
  border-radius: 5px;
  border: none;
  background-color: rgba(255,255,255,.5);
  color: rgba(255,255,255,1);
  font-size: 1.2em;
  box-shadow: inset 0 0 10px rgba(255,255,255,.75);
  width: 500px;
  padding: 5px;
}
```

Вы можете сделать форму более удобной для пользователей, выделив активные элементы с помощью специального псевдокласса `:focus` (см. раздел «Выборка стилизуемых ссылок» гл. 9). Мы добавим его в следующем пункте.

15. В конце внутренней таблицы стилей добавьте стиль для раскрывающегося списка и трех полей для ввода текста:

```
.subform input[type="text"]:focus, .subform textarea:focus {
  background-color: white;
  color: black;
}
```

Посмотрите страницу в браузере. Сейчас она должна выглядеть так, как на рис. 11.13.

The image shows a 'Reader Subscription Form' on a dark background. The form has the following elements:

- A title 'Reader Subscription Form' with a dotted line underneath.
- A text input field labeled 'What is your name?'.
- A text input field labeled 'What is your email address?'.
- A section 'Rate your apartment farming skills' with three radio buttons: 'Novice', 'Intermediate', and 'Advanced'.
- A dropdown menu labeled 'Where did you hear about us?' with the text 'Select One'.
- A large text area labeled 'Any additional comments?'.
- A 'Subscribe' button at the bottom.

**Рис. 11.13.** Используя псевдокласс `:focus`, вы можете сделать форму более удобной для пользователя: будет подсвечиваться активное поле ввода. Здесь можно увидеть, что все готово к вводу текста в поле комментариев, поскольку оно имеет белый цвет фона

Готовую версию созданной в этом уроке формы вы можете найти в папке `11_finished\form`.

## ЧАСТЬ 3

# Макет страницы

**Глава 12.** Введение в разметку CSS

**Глава 13.** Разметка страницы на основе плавающих элементов

**Глава 14.** Адаптивный веб-дизайн

**Глава 15.** Позиционирование элементов на веб-странице

# 12 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 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, из следующих трех глав 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? На чем они просматривают ваш сайт: на телефоне с операционной системой Android или на iPad? 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  $640 \times 480$  пикселей до чудовищных 30-дюймовых экранов с разрешением примерно  $5\,000\,000 \times 4\,300\,000$  пикселей. Не говоря уже о небольших дисплеях мобильных телефонов и экранах планшетных компьютеров, имеющих размер от небольшого до среднего.

Веб-разметка предполагает несколько основных подходов 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. Кроме того, владельцам смартфонов для получения нужного им содержимого приходится прибегать к сужению и расширению выводимой на экран информации. Исполни-

зование непостоянной ширины (которая увеличивается или уменьшается, чтобы соответствовать окнам браузера) делает управление дизайном страницы серьезным испытанием, но при этом эффективнее используется окно браузера. Одно из последних новшеств — *адаптивный веб-дизайн* — старается решить проблему сильно отличающихся по ширине экранов, но за счет усложнения.

- **Фиксированная ширина.** Многие дизайнеры предпочитают плотно распределять страницу по ширине, как на странице на рис. 12.1, *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 на менее 1000 пикселей, позволяя окну, полосам прокрутки и другим частям браузера подходить по размерам для экранов с разрешением 1024 × 768. Очень распространенной является ширина 960 пикселей. Фиксированная ширина используется во многих веб-сайтах, но положение дел изменяется благодаря широкому внедрению смартфонов и планшетных компьютеров, экран которых зачастую имеет меньшую ширину, чем среднестатистический дизайн веб-страниц с фиксированной шириной.

---

**ПРИМЕЧАНИЕ**

Чтобы увидеть примеры дизайнов с фиксированной шириной, нацеленных на большие мониторы, зайдите на сайты [www.alistapart.com](http://www.alistapart.com), [www.espn.com](http://www.espn.com) или [www.nytimes.com](http://www.nytimes.com).

---

- **Непостоянная ширина.** Иногда легче плыть по течению вместо того, чтобы бороться с ним. Свободный дизайн приспособляется так, чтобы соответствовать любой ширине браузера путем задания процентного значения ширины вместо абсолютного значения в пикселях. Ваша страница становится шире либо уже, когда посетитель изменяет размеры окна (см. рис. 12.1, *посередине*). Хотя 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, не подходящими для их нормального прочтения.

---

**ПРИМЕЧАНИЕ**

Чтобы увидеть примеры дизайнов с непостоянной шириной, зайдите на сайт <http://maps.google.com>.

---

- **Адаптивный веб-дизайн** (Responsive Web Design, RWD). Эта новая технология, отстаиваемая веб-дизайнером Этаном Маркоттом (Ethan Marcotte), предлагает другой способ решения проблемы, возникшей из-за большого разнообразия размеров экранов, имеющихся в распоряжении веб-браузеров смартфонов, планшетных и настольных компьютеров. Вместо предоставления единого макета

для всех устройств адаптивный веб-дизайн проводит коррекцию ширины для различных веб-браузеров путем изменения их представлений. Например, адаптивная веб-страница может ужаться с широкого, состоящего из нескольких колонок макета для мониторов настольных компьютеров до макета, содержащего всего одну колонку для смартфонов. Таким образом, адаптивный веб-дизайн сильно напоминает макеты с непостоянной шириной — конструкции, использующие процентные отношения с целью расширения или сужения в ответ на задаваемую ширину окна браузера. Но в новом веб-дизайне технология пошла дальше путем использования более сложного кода CSS, так называемых уточнений носителей данных (Media Queries) для отправки различных дизайнерских решений для браузеров, располагающих экранами разной ширины, что позволяет создавать существенно отличающийся по внешнему виду макеты в зависимости от устройств, на которых просматривается страница.

#### ПРИМЕЧАНИЕ

---

Свойства `max-width` и `min-width` предлагают компромисс между фиксированной и свободной шириной.

---

Факт наличия у посетителей вашего сайта мониторов с разным разрешением можно просто проигнорировать, и задать для страницы единую, неизменяющуюся ширину (см. рис. 12.1, *вверху*). Можно создать дизайн с непостоянной шириной, элементы которого перетекают, чтобы занять ширину того или иного окна (см. рис. 12.1, *по середине*), Адаптивный дизайн предусматривает настоящее перерформатирование в зависимости от ширины окна браузера. Например, как показано в нижней части рис. 12.1, по мере увеличения окна браузера сайт газеты Boston Globe переходит от использования одной колонки (слева), к двум (посередине), а затем и к трем колонкам (справа).

В обучающих уроках в конце этой главы вы создадите страницы с фиксированной и свободной шириной. Техника адаптивного веб-дизайна будет подробно рассмотрена в гл. 14.

## Как работает CSS-разметка

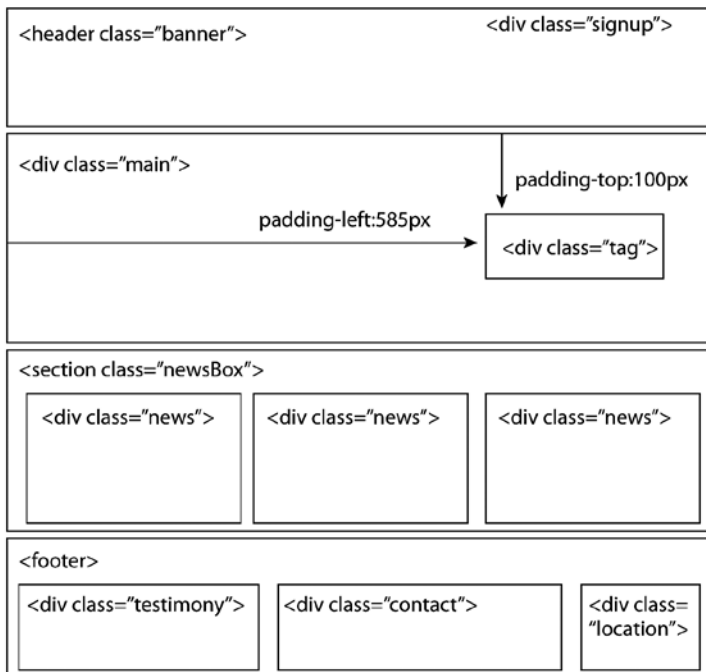
Как говорилось в гл. 1, на заре развития Интернета ограничения HTML вынудили дизайнеров придумывать новые способы оформления сайтов. Самым распространенным инструментом был тег `<table>`, который, как изначально предполагалось, должен был служить для отображения информации в виде электронной таблицы, составленной из строк и столбцов с данными. Разработчики использовали HTML-таблицы, чтобы создать своего рода основу для организации содержимого страницы (рис. 12.2). 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. 12.1. Справиться с неопределенностью ширины окон веб-браузеров и шрифтов браузеров можно несколькими способами

В заслугу HTML-таблицам можно поставить следующее: они предоставляют хорошо организованную структуру с унифицированными отдельными ячейками. В CSS-дизайне имеются отдельные теги с содержимым, которое нужно разместить в одной из областей страницы. Путем группировки содержимого во многих отдельных контейнерах и позиционирования этих контейнеров можно создать сложные дизайнерские решения, состоящие из столбцов и строк. До появления HTML5 в вашем распоряжении имелся только один тег для достижения данной цели, это <div>-тег.





**Рис. 12.2.** 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 `<div>`

## Тег <div>

Разметка приводит к расположению содержимого в различных областях страницы. В CSS для организации содержимого обычно применяется тег <div>. Как вы читали в разделе «Написание HTML-кода для CSS» гл. 1, тег <div> — это HTML-элемент, у которого нет никаких собственных свойств форматирования (помимо того факта, что браузеры рассматривают этот тег как блок с концом строки до и после него). Вместо этого он используется для обозначения логического группирования элементов, или *распределения*, на странице.

В тег <div> обычно заключается фрагмент HTML-кода, где все элементы объединены общим смыслом. Такой элемент, как навигационная панель (см. рис. 12.2), обычно занимает вершину страницы, так что есть смысл задавать тег <div> и вокруг него. Кроме того, придется определить тег <div> для всех основных областей вашей страницы, таких как баннер, область с главным содержимым, боковая панель, нижний колонтитул и т. д. Есть возможность указать тег <div> вокруг одного или нескольких дополнительных отделений. Одна из распространенных методик состоит в упаковке HTML-кода внутри тега <body> в тег <div>. 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 расположить все содержимое посередине экрана и добавить фоновый цвет или изображение для главного столбца страницы.

Как только будут определены все <div>-теги, добавьте к каждому из них либо класс, либо идентификатор, чтобы было удобнее создавать для каждого <div>-контейнера отдельный стиль. Тег <div> для области баннера страницы может иметь следующий вид: <div class="banner">. Можно обойтись только одними идентификаторами, но один и тот же идентификатор можно использовать на странице только один, поэтому при наличии элемента, появляющегося несколько раз, следует применить класс. Если, например, есть несколько div-контейнеров, позиционирующих фотографии и подписи под ними, можно заключить такие теги в div-контейнер и добавить следующий класс: <div class="photo">. (Чтобы лучше понять, когда следует использовать div-контейнер, прочитайте врезку «Информация для новичков. HTML-теги <div> и <span>» в разделе «Селекторы классов: точное управление» гл. 3.) Кроме того, когда дело касается каскада (гл. 5), идентификаторы проявляют всю свою мощь и легко отменяют другие стили, зачастую заставляя вас создавать длинные селекторы вроде #home #banner #nav просто для того, чтобы отменить ранее созданные селекторы с использованием идентификаторов. Поэтому многие веб-дизайнеры избегают применения идентификаторов, отдавая предпочтение классам.

Как только будут расставлены теги <div>, добавьте к каждому из них либо класс, либо идентификатор (ID), и тогда вы сможете создавать стили CSS для размещения блоков на странице, используя плавающие элементы (см. гл. 13).

### ИНФОРМАЦИЯ ДЛЯ НОВИЧКОВ

#### Находим разумный баланс

Хоть разделы div крайне необходимы CSS-разметкам, не бросайтесь активно применять их на вашей стра-

нице. Распространенное заблуждение состоит в том, что вы должны указывать теги <div> для всех элемен-

## ИНФОРМАЦИЯ ДЛЯ НОВИЧКОВ

тов страницы. Скажем, ваша главная навигационная панель — неупорядоченный список ссылок (как, например, та, что описана в разделе «Создание панелей навигации» гл. 9). Поскольку это важный элемент, вы, возможно, пожелаете указать `<div>` вокруг него: `<div id="mainNav"><ul>...</ul></div>`.

Однако нет никаких причин добавлять тег `<div>`, когда `<ul>` настолько же удобен. Пока тег `<ul>` содержит ссылки главной навигационной панели, вы можете просто добавить ваш класс к этому тегу: `<ul class="mainNav">`.

Кроме того, нет необходимости использовать `<div>`, когда под рукой есть другой, более подходящий элемент HTML. Например, вы захотели процитировать кусок текста — создать блок, выровненный по правому краю и содержащий важную цитату из содержимого страницы. В этом случае вы можете пропустить `<div>` и просто использовать тег `<blockquote>`, позиционируя его с помощью свойства `float`, которое

рассматривалось в разделе «Управление обтеканием содержимого плавающих элементов» гл. 7.

Но это не значит, что нужно бояться использовать теги `<div>`. Добавив их на парочку больше, вы не увеличите размер файла и не замедлите загрузку страницы. Если `<div>` поможет вам решить проблему и ни один из других тегов не подходит, естественно, применяйте `<div>`. Кроме того, `<div>` — это единственный способ сгруппировать в единое целое несколько различных тегов HTML. На самом деле не так редко можно увидеть один тег `<div>`, 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 `<div>` имеет смысл для проектирования дизайна, то смело применяйте их. Или же, если вы вплотную занялись HTML5, попробуйте воспользоваться элементами секционирования, рассматриваемыми в следующем разделе.

## Элементы секционирования в HTML5

До появления HTML5 для организации содержимого дизайнеры использовали в основном тег `div`. В HTML5 были представлены многие другие теги, предназначенные для группировки содержимого конкретного типа. Например, тег `<article>` предназначен для содержимого, составляющего одну самостоятельную композицию, например пост блога. Тег `<header>` — для предоставления заголовка для навигационных целей и для другого вводного материала страницы или ее раздела. Есть также тег `<footer>`, предназначенный для содержания завершающей информации, такой как сведения об авторских правах, имя создателя веб-сайта, дата публикации страницы и т. д. Короче говоря, в HTML5 есть теги, которые берут на себя работу тега `<div>` для определенных типов содержимого. (Краткое введение в элементы секционирования в HTML5 можно найти по адресу [www.adobe.com/devnet/dreamweaver/articles/understanding-html5-semantic.html](http://www.adobe.com/devnet/dreamweaver/articles/understanding-html5-semantic.html).)

### ПРИМЕЧАНИЕ

Если вы нацелились на использование HTML5, не забывайте использовать HTML5-прокладку, рассмотренную во врезке «Обходной прием. Как заставить Internet Explorer 8 понимать HTML5» раздела «Дополнительные теги в HTML5» гл. 1, чтобы заставить Internet Explorer 8 и более ранние версии проводить форматирование с помощью этих новых элементов.

Как уже упоминалось в гл. 1, новые элементы HTML5 предназначены для добавления семантики к веб-странице. То есть имя тега, например `<header>`, сообщает идентификационную информацию о содержимом этого тега. Тег `<figure>` предназначен для содержимого рисунка. То есть, если нужно поместить внутри тега со-

держимое конкретного типа, посмотрите, какой из тегов отвечает данным требованиям. Если содержимое является нижним колонтитулом вашей страницы, то вместо простого старого `<div>`-контейнера следует использовать контейнер `<footer>`. В мире HTML5 `<div>`-тег по-прежнему существует и абсолютно пригоден к применению. Но теперь он в основном используется для группировки содержимого по стилистическим соображениям, например, если нужно добавить границу вокруг группы тегов или переместить эти теги к левому краю страницы, используя свойство `float`. (Как все это делается, будет рассмотрено в главе 13.)

Все это говорит о том, что, кроме демонстрации превосходства над конкурентами на очередной встрече веб-разработчиков, от использования HTML в данный момент нет никаких особых преимуществ. В будущем программные средства смогут просматривать ваш HTML и идентифицировать отдельные статьи на основе присутствия в нем тега `<article>`, или задействовать теги HTML5 каким-либо другим образом, чтобы улучшить анализ вашей веб-страницы. Или же, если вы хотите предусмотреть для своих страниц долгую жизнь и подготовиться к тем временам, когда HTML5 начнет править миром, можете приступить к использованию HTML5 уже сегодня. Но если вы оттачивали свое мастерство на применении `<div>`-тегов и не желаете связываться с применением JavaScript, чтобы заставить HTML5 работать в Internet Explorer 8 (см. предыдущее примечание), то пока можете по-прежнему использовать только `<div>`-тег.

## Технологии разметки CSS

В большинстве веб-страниц для разметки используется CSS-свойство `float` (другие варианты CSS-разметки рассмотрены в следующей врезке с информацией для опытных пользователей). 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 на левую, либо на правую сторону. Тот же принцип применяется и к другим тегам: устанавливая для них ширину и перемещая влево или вправо, вы можете создать столбец (текст, следующий за элементом, обтекает плавающий элемент, как будто тот является еще одним столбцом). Используя свойство `float` с множеством тегов `<div>` или других тегов, вы получаете возможность создавать многостолбцовые разметки. Применяя эту методику дальше, вы будете быстро создавать сложные разметки, помещая одни плавающие теги `<div>` внутри других.

Еще одна CSS-технология — абсолютное позиционирование — позволяет поместить элемент в любом месте страницы с точностью до одного пиксела. Вы можете расположить 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 технологии. Как вы узнаете из гл. 14, выполнить разметку страницы с помощью абсолютного позиционирования можно, но этот способ лучше подходит для небольших задач вроде позиционирования логотипа в конкретном месте на странице.

Если пока все эти понятия вы воспринимаете слишком абстрактно, переживать не стоит — работа с использованием всех этих технологий будет рассмотрена в следующих трех главах.

## 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, видео, 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 большое значение будет иметь фотография с приветливыми сотрудниками, как и цитата из отзыва одного из довольных клиентов. Поскольку оба этих элемента важны для страницы, вы должны сделать их видными и неотразимыми.

### Сначала нужно подумать о мобильных устройствах (Mobile First)

В связи с ростом количества смартфонов и планшетных компьютеров разработчикам пришлось призадуматься о сокращении содержимого до ключевых сообщений и фактов. Это движение под названием Mobile First связано с ограниченным размером экрана смартфонов, а также с ограниченным вниманием людей, находящихся в движении. Конструкции Mobile First касаются начального вида вашего содержимого, а также избавления от его излишнего зашумления, включая дополнительную информацию, которая прекрасно помещается на больших экранах настольных систем, но создает помеху на экранах значительно меньшего размера и отвлекает от основной информации, которую вы надеялись донести до посетителя.

Учтите, если ваш сайт будет посещать существенное количество людей, использующих смартфоны или небольшие планшетные компьютеры, не каждый из них захочет прокручивать длинную информационную страницу (или сужать или расширять поле зрения, чтобы увидеть все, что доступно на странице). Вместо попытки заполнить каждый оставшийся квадратный сантиметр на 32-дюймовом настольном мониторе, подумайте об упрощении своего содержимого, чтобы его усвоение давалось просто, легко и непосредственно.

#### ПРИМЕЧАНИЕ

Понятие Mobile First было придумано Люком Врублевским. В действительности им был написан фантастически короткий трактат на эту тему, который стоит прочитать: [www.abookapart.com/products/mobile-first](http://www.abookapart.com/products/mobile-first).

### ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

#### Другие возможности макетирования, предлагаемые в CSS3

Сначала для создания многоколоночных макетов веб-дизайнеры использовали таблицы. Затем появилось CSS-свойство `float`, благодаря чему был предоставлен другой менее объемный по коду метод макетирования веб-страницы. И хотя большинство разработчиков все еще используют свойство `float` для достижения своих дизайнерских целей, в ближайшее время все может измениться.

Рабочая группа W3C настойчиво приводит к окончательному виду разработки некоторых других технологий CSS-верстки. Например, CSS-модуль многоколоночного макетирования — *multicolumn* — предоставляет способ получения длинной колонки текста и ее отображения в нескольких смежных колонках. Разговор о нем пойдет в обучающем уроке следующей главы.

Модуль гибкого макетирования блоков — *flexible box*, предоставляет еще один способ выстраивания блоков содержимого — вертикально, горизонтально, а также в различных направлениях и порядках следования. Этот модуль имеет хорошую поддержку

во многих современных браузерах. Он будет рассмотрен во врезке «Информация для опытных пользователей». На подходе более совершенные способы разметки» в подразделе «Создание столбцов на всю высоту» раздела «Решение проблем плавающих элементов» гл. 13.

И наконец, CSS-модуль решетки — *grid* — является, наверное, наиболее амбициозной попыткой изменить способ макетирования веб-страниц. Этот способ позволяет разбить страницу на набор строк и столбцов, затем аккуратно прикрепить элементы страницы к этой решетке. Он предназначен для упрощения создания веб-приложений и больше похож на создание программ для настольного компьютера. CSS *grid* довольно сложный модуль и (на момент написания данной книги) еще не поддерживается какими-либо из поставляемых браузеров, хотя поддерживается в бета-версии Internet Explorer 10. Дополнительные сведения о нем можно получить по адресу <http://blogs.msdn.com/b/ie/archive/2011/04/14/ie10-platformpreview-and-css-features-for-adaptive-layouts.aspx>.

## Начните с наброска

После решения вопроса с содержимым можно подумать о его визуальной организации. Некоторые разработчики начинают с ручного программирования на HTML: создания `div`-контейнеров, добавления тегов `<header>`, `<article>`, `<footer>` и т. д. Это вполне разумный подход, часто называемый разработкой в браузере, поскольку он дает вам главный старт для создания вашего сайта путем перехода непосредственно к HTML.

Но перед переходом к HTML нужно хотя бы набросать план размещения содержимого. Здесь не нужно ничего особенного, сгодятся бумага и карандаш. Поскольку разработчик собирается поместить содержимое в блоки (div и другие HTML-теги) и расположить эти блоки на странице, простой набросок группы блоков с прорисовкой столбцов и т. д. является быстрым и простым способом проработки различных макетов страницы. Можно быстро понять, куда должно помещаться содержимое, насколько объемным оно будет и каким нужно сделать общий цветовой тон (светлым или темным).

## СОВЕТ

---

Yahoo! предлагает свободный доступ к средству Stencil Kit (<http://developer.yahoo.com/ypatterns/wireframes>), которое может использоваться в Illustrator, Visio, OmniGraffle и других программах работы с графикой для создания макетов веб-страниц. Предоставляемые элементы пользовательского интерфейса, такие как кнопки, поля формы, окна и кнопки навигации, позволяют составить набросок макета страницы путем простого перетаскивания значков.

---

При наличии опыта работы с программами обработки графики типа Photoshop, Illustrator или Fireworks, вы можете воспользоваться ими для создания визуального представления. Если вы несильны в работе с такими программами, простой рисунок блоков для обозначения различных мест расположения элементов страницы поможет уточнить замысел намечаемого макета страницы. Намного проще будет изменить двухколоночный дизайн на четырехколоночный путем изменения размеров блоков в программе Illustrator, чем переписыванием кода HTML и CSS.

Но при использовании программ работы с графикой не стоит тратить слишком много времени на улучшение графического дизайна. Не тратьте для создания в Photoshop или Illustrator внешнего вида, получаемого при использовании CSS-свойств, слишком много усилий, для его переделки нужно будет просто воспользоваться кодом CSS. То есть постарайтесь уточнить предварительный вид своего сайта на бумаге, а затем переходите к текстовому редактору для создания HTML, в который помещается содержимое, и к использованию CSS-свойств, изученных благодаря данной книге, для тестирования различных стилей.

## Идентифицируйте разделы

После того как вы создали визуальный макет, нужно подумать о том, как выполнить 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. 12.2 есть немало элементов, которые выглядят как небольшие разделы: наиболее крупным является раздел с тремя объявлениями внизу (помечены буквой А на рис. 12.2). 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. 12.2), вы знаете, что нужно каж-



дый 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. 12.2 вы видите набор тегов `<div>`, которые обеспечивают структуру страницы. Разделы `div`-контейнер `news` и тег `footer` являются контейнерами для своих собственных наборов `<div>`-контейнеров. Хотя 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 по его правую сторону, который бы сформировал 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. 12.2 четыре элемента — тег `<header>`, `div`-контейнер `main`, тег `<section>` и тег `<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.

## Помните о фоновых изображениях

Вы, несомненно, видели черепичные изображения, заполняющие фон веб-страницы, или градиенты, которые добавляют эффект глубины для баннера. Свойство `background-image` предоставляет еще один способ добавления фотографий 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. 12.2 изображение руки по центру (*B*) на самом деле является обычным фоновым изображением. Это облегчает размещение другого тега `<div>` 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 `<div>`. Добавление небольшого отступа справа отодвинет текст в этом разделе от фотографии.

### ПРИМЕЧАНИЕ

---

Есть минусы использования фотографий в качестве фонов тегов `<div>` (или других тегов HTML). 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`.

---



## Кусочки пазла

Часто то, что выглядит как одно целое, на самом деле состоит из нескольких частей. Вы можете увидеть простой пример этого в нижней части изображения веб-страницы на рис. 12.2. Здесь есть четыре расположенных друг за другом тега `<div>`, 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 пазла.

## 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. 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. 15, 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. Помните, что отступы и поля представляют собой обычное пустое пространство и, используя их, вы можете двигать элементы по странице. Например, подзаголовок (С на рис. 12.2) размещается простой установкой верхнего и левого отступа для его родительского тега. Как вы можете видеть на схеме в нижней половине рис. 12.2, подзаголовок помещен внутри другого тега `<div>` (`<div class="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.

# 13 Разметка страницы на основе плавающих элементов

Разметка, основанная на плавающих элементах, использует преимущества свойства `float` для организации элементов бок о бок, создавая на веб-странице столбцы. Как было описано в разделе «Управление обтеканием содержимого плавающих элементов» гл. 7, вы можете использовать это свойство с целью 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 боковую панель:

```
.sidebar {  
  float: left;  
  width: 170px;  
}
```

---

## ПРИМЕЧАНИЕ

Значение `none` отменяет любое перемещение и определяет элемент как обычный (неплавающий). 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. раздел «Управление каскадностью» гл. 5) с `float: none`, вы можете предохранить этот элемент от перемещения.

---

На рис. 13.1 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.



← левое поле

**Рис. 13.1.** Используйте свойство `float` для разметки веб-страницы с множеством столбцов

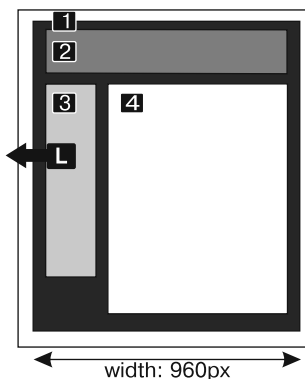
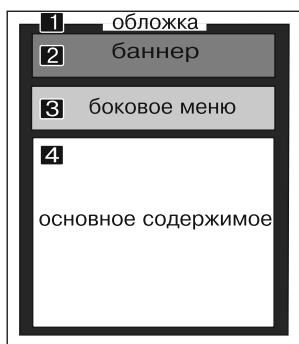
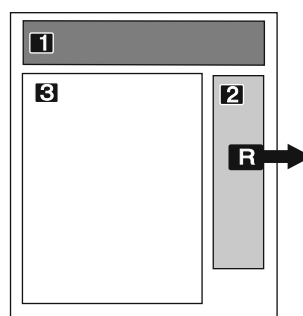
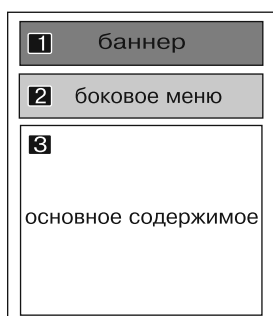
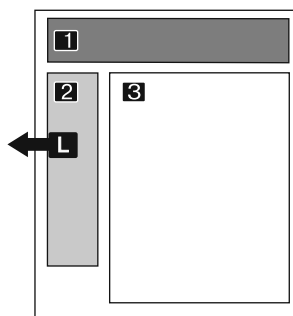
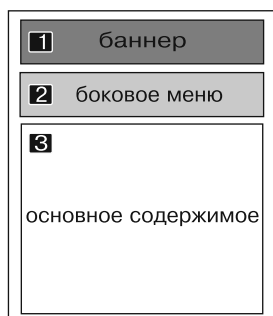
Простой дизайн с двумя столбцами, как на рис. 13.1, требует выполнения всего нескольких действий.

1. **Укажите вокруг каждого столбца тег `<div>` с атрибутом класса или ID.** На рис. 13.1 заголовки новостей, перечисленные в левой части, заключены в один раздел (`<div class = "news">`), а главное содержимое страницы — в другой (`<div class = "main">`).
2. **Переместите тег `<div>` с боковой панелью 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. 13.2 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 боковой панели, и, наконец, тег `<div>` для главного содержимого. Справа вы видите фактическую разметку страницы. Боковая панель идет *перед* главным содержимым в HTML, так что она может переместиться или влево (*вверху и внизу*), или вправо (*посередине*).

Порядок описания в HTML

CSS-разметка



**Рис. 13.2.** Создание разметки с двумя столбцами — лишь вопрос перемещения тега `<div>` влево (вверху). Чтобы заставить боковую панель переместиться в правую сторону страницы (посередине), просто измените CSS-стиль боковой панели на `float: right`

3. **Установите ширину для плавающей боковой панели.** 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. в разделе «Типы разметок веб-страницы» обо всех «за» и «против» различных единиц измерения). Если у боковой панели ширина 20 %, а окно браузера — 700 пикселей в ширину, то ширина боковой панели будет 140 пикселей. Если же ваш посетитель изменит размер окна до 1000 пикселей, то боковая панель вырастет до 200 пикселей. Боковые панели с фиксированной шириной легче проектировать, так как не нужно рассматривать все различные значения ширины, до которых боковая панель могла бы растянуться. Однако проценты позволяют вам поддерживать одинаковые пропорции между двумя столбцами, что может быть визуально привлекательнее. Кроме того, проценты делают ваши конструкции гибче, поскольку общие пропорции страницы могут подстраиваться под размер экрана, что играет важную роль при создании адаптивных веб-конструкций, речь о которых пойдет в следующей главе.

#### ПРИМЕЧАНИЕ

Когда дизайн всей страницы указан с фиксированной шириной, значения ширины для боковой панели 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 боковая панель короче другого содержимого на странице, то текст из главного столбца обтекает панель снизу. Добавление левого края, большего или равного ширине боковой панели, выравнивает главное содержимое страницы, создавая иллюзию второго столбца:

```
.main { margin-left: 180px ; }
```

Удобно делать левый край чуть больше по ширине, чем боковая панель: это добавляет промежуток между двумя элементами. Если для установки ширины боковой панели 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.

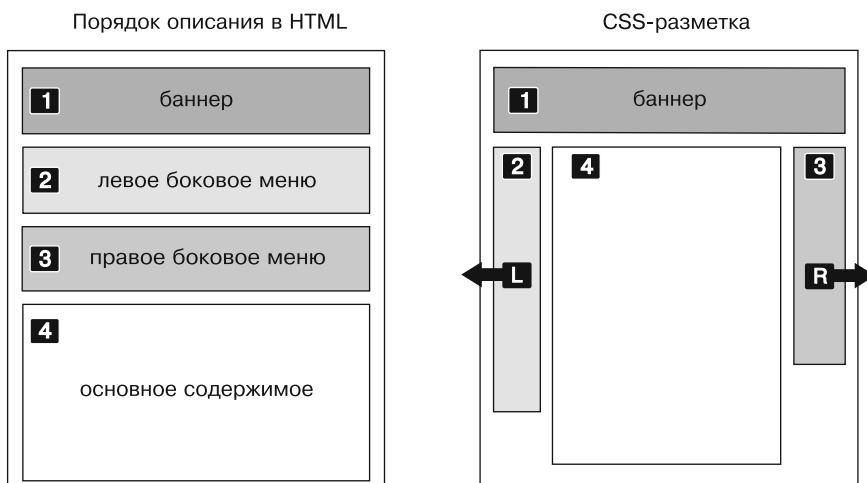
## Использование плавающих элементов в разметках

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 `<div>` (например, `<div class = "wrap">`), а затем создать стиль для этого нового элемента-контейнера, для которого определена ширина, скажем, 960 пикселей (см. рис. 13.2, *внизу*). Эта установка ширины удерживает все внутри контейнера.

#### ПРИМЕЧАНИЕ

Существует также вариант создания страницы с фиксированной шириной без применения дополнительного тега `<div>`, 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. 13.3). 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 новой правой боковой панели, он не обтекал боковую панель.



**Рис. 13.3.** При дизайне с тремя столбцами вы перемещаете левую и правую панели 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

#### Нужно ли снова изобретать 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 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 (<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 настройками форматирования, такими как стиль шрифта и изображения.

Есть также немало генераторов разметок — эти онлайн-инструменты позволяют по вашему желанию настроить такие основные параметры, как количество столбцов, тип разметки и т. д. *Gridinator* (<http://gridinator.com>) предоставляет простое средство для создания сложных многоколоночных grid-систем (см. врезку «Информация для опытных пользователей. Простой способ создания нескольких колонок» в подразделе «Отмена и установка перемещения для элементов» раздела «Решение проблем плавающих элементов» данной главы). После этого можно будет загрузить HTML- и CSS-файлы с созданным для вас кодом.

## Перемещение всех столбцов

В полной мере можно заставить перемещаться каждый столбец, а не только левую и правую боковые панели. Вы можете переместить левую боковую панель и средний столбец влево, а правую боковую панель — вправо, как показано на рис. 13.4, *вверху*. Этот подход позволяет размещать более трех столбцов на странице. Вы можете перемещать четыре или более столбца, пока есть место для всех плавающих элементов, чтобы они находились бок о бок.

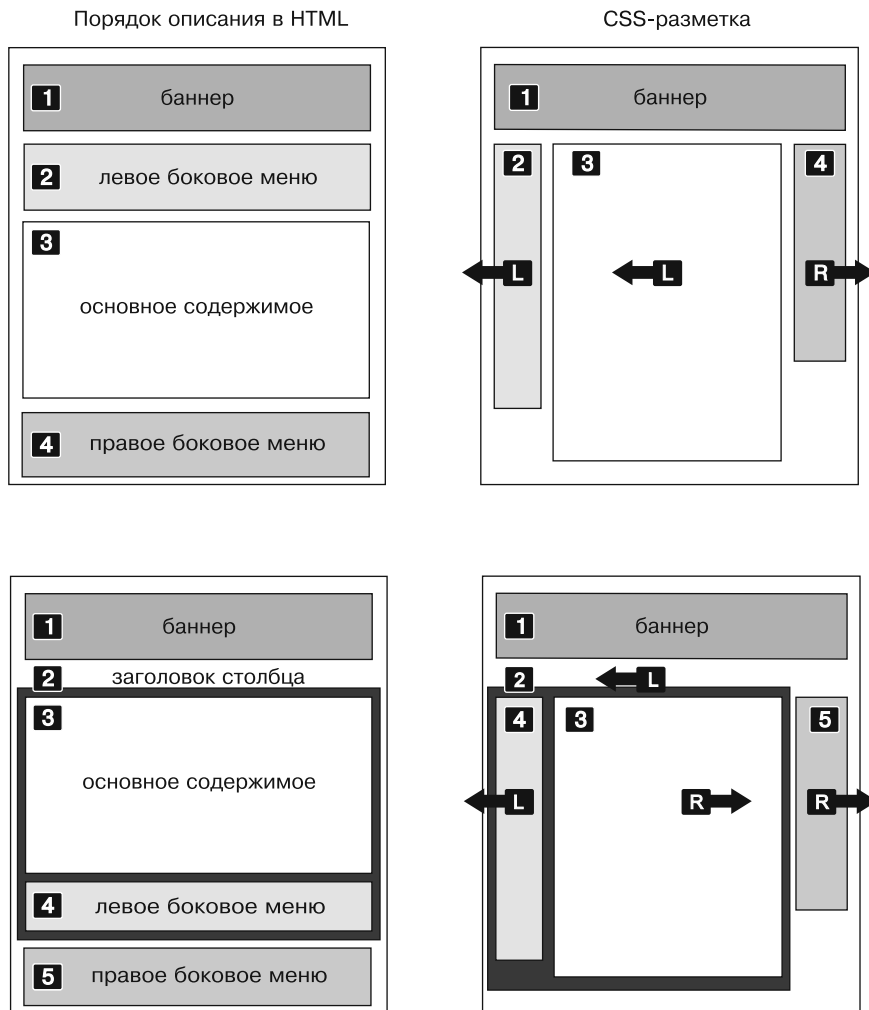
Когда вы перемещаете все столбцы в дизайне, вы должны обратить пристальное внимание на ширину каждого столбца. Если совокупная ширина всех столбцов меньше доступного места, например, когда окно браузера меньше или столбцы помещены внутри другого тега `<div>` с установленной шириной, то последний столбец опускается вниз (как решать проблему перемещения плавающих элементов, вы можете прочитать в подразделе «Предотвращение выпадений плавающих элементов» раздела «Решение проблем плавающих элементов» этой главы).

Вдобавок, перемещая не только боковые панели, вы сможете изменить порядок своих разделов `div` в HTML. Возьмите, например, левую диаграмму на рис. 13.3, 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* боковой панели.

Порядок тегов `<div>` в HTML может показаться чем-то не очень важным, пока вы не попытаетесь просмотреть веб-страницу *без* CSS, что имеет место во многих альтернативных браузерах, включая экранных дикторов, которые читают содержание страницы вслух для слабовидящих посетителей. Без CSS весь материал боковой панели (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 боковой панели 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.

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 большее 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.

В левой верхней разметке на рис. 13.4 HTML-код с основным содержимым находится между левой и правой боковыми панелями, что лучше, чем размещать его *после* этих блоков. Вы можете даже поместить описание главного содержимого перед HTML-кодом для боковых панелей, определяя для него и левой боковой панели один тег `<div>` и перемещая его влево. Затем внутри этого тега `<div>` нужно переместить основное содержимое вправо, а левую боковую панель — влево (см. рис. 13.4, *внизу*). Вуаля! HTML-код главного столбца идет перед остальными тегами `<div>`.



**Рис. 13.4.** 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

## Плавающие элементы внутри плавающих элементов

Нижняя разметка на рис. 13.4 иллюстрирует другую полезную методику — перемещение элементов *внутри* плавающих элементов. Предположим, что разделов главного содержимого (3) и левой боковой панели (4) не существует, а были оставлены только «обертка» столбца (2) и правая боковая панель (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.



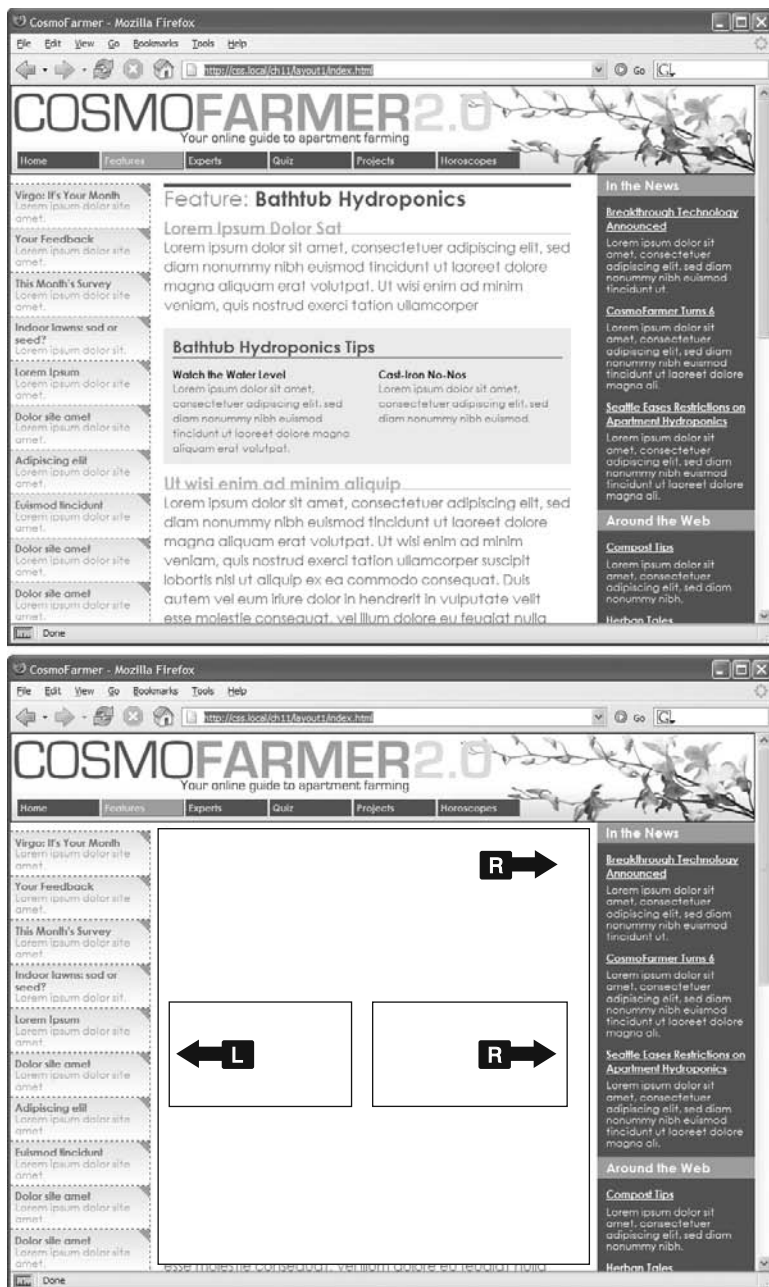


Рис. 13.5. Пример разметки с плавающими элементами внутри плавающих элементов

Хотя эта разметка немного смущает, она бывает полезной во многих случаях. Во-первых, она позволяет добавлять столбцы внутри других столбцов. На разметке на рис. 13.5, *вверху*, показан маленький блок для подсказок в среднем столбце,

внутри которого также есть два столбца. Вкладывая одни плавающие элементы внутри 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 поможет при решении проблем выпадений плавающих элементов (см. подраздел «Предотвращение выпадений плавающих элементов» раздела «Решение проблем плавающих элементов» этой главы) и другие проблемы, возникающие, когда столбцы становятся слишком широкими.

Рассмотрим страницу на рис. 13.5, *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. 13.5, *внизу*, видно, что неважно, в каком направлении перемещается контейнер (в этом случае — вправо), — вы просто передвигаете два дополнительных столбца влево и вправо.

## ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

### Определение порядка в grid-системе

Графические дизайнеры использовали решетки (grid) для предоставления исходного порядка в своих макетах задолго до появления Интернета. Grid-система является простым набором строк и столбцов, помогающим выровнять элементы в макете. Например, в 12-колоночной grid-системе страница делится на 12 столбцов одинакового размера, обычно с небольшим межстолбцовым промежутком. Но наличие grid-системы не означает, что вашей странице нужны 12 узких столбцов содержимого. Вместо этого вы просто делите 12 столбцов (которые часто называют grid-элементами) на группы. Например, чтобы заполнить все 12 столбцов, в трехколоночном дизайне у вас может быть левая боковая панель шириной в три grid-элемента, основная колонка шириной в шесть grid-элементов и правая боковая панель шириной в три grid-элемента. Эта 12-колоночная grid-система обладает гибкостью, позволяющей вам пробовать создавать и другие макеты при сохранении общей и последовательной измерительной схемы. Например, у вас может быть три столбца одинаковой ширины (по 4 grid-элемента в каждом).

В большинстве CSS grid-систем страница делится на строки, состоящие из столбцов. Для определения строки с помощью `<div>` обычно существует некая основанная на классе номенклатура:

```
<div class="row">
```

А внутри строки помещаются дополнительные теги, зачастую `<div>`-теги, но вы можете использовать любые блочные элементы, например элементы секционирования HTML5. Каждый из этих тегов включает класс, показывающий его ширину в grid-элементах. Например, трехколоночный макет может иметь следующий код:

```
<div class="span3"> </div>
<div class="span6"> </div>
<div class="span3"> </div>
```

Имя класса зависит от grid-системы. Две популярные структуры, основанные на grid-системах, — Foundation (<http://foundation.zurb.com/docs/grid.php>) и Twitter Bootstrap (<http://twitter.github.com/bootstrap/scaffolding.html#grid>) — включают полные решения для grid-систем на основе CSS. Для создания столбцов различной ширины используется их исходный CSS-код и их соглашение об именах. Более простой подход описан в блоге-посте Криса Койера (Chris Coyier) на сайте CSS-Tricks.com, где он объясняет порядок создания своей собственной простой grid-системы: <http://css-tricks.com/dont-overthinkit-grids/>.

## Решение проблем плавающих элементов

Когда вы начнете активно работать с CSS, вероятно, как и многие веб-разработчики до вас, столкнетесь с некоторыми сложностями при работе с перемещаемыми (плавающими) элементами. В этом разделе описаны некоторые распространенные проблемы и пути их решения. Если же у вас когда-либо возникнет проблема, которая не описана здесь, то всегда можете спросить о ней на одном из онлайн-форумов, перечисленных в приложении 2.

## Отмена и установка перемещения для элементов

Перемещаемые элементы — мощные средства проектирования, поскольку позволяют содержимому обтекать их. Перемещение фотографии позволяет тексту, находящемуся под ней, продвигнуться вверх и «обернуться» вокруг изображения (см. рис. 13.1). 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 плавающей боковой панелью, нижний колонтитул может передвинуться вверх, оказавшись рядом с левым плавающим столбцом (рис. 13.6, *слева*). Чтобы заставить нижний колонтитул остаться внизу под боковой панелью, вы можете использовать свойство `clear` (см. раздел «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 обтекания:

```
#footer { clear: both; }
```

Другая проблема появляется, когда вы перемещаете один или несколько элементов внутри непереключаемого 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. 13.7 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. 13.7, *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. 13.6. Нижний колонтитул должен оставаться внизу страницы, независимо от перемещаемых элементов. Чтобы добиться этого, используйте свойство clear

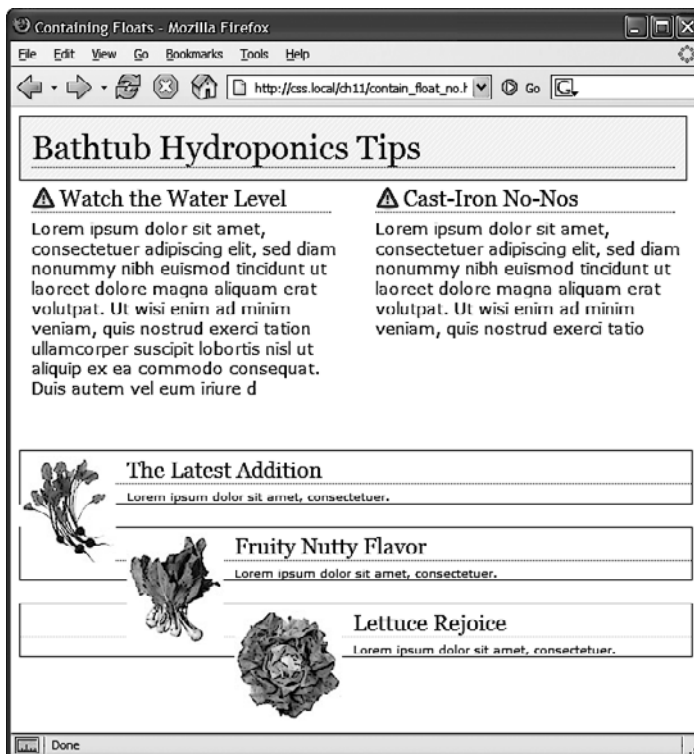


Рис. 13.7. Перемещаемый элемент может выйти за пределы содержащего его блока. Если в блоке определены фоновый цвет или граница, то выходящие элементы могут выглядеть так, как будто даже не являются частью блока (вверху). Кроме того, перемещаемый элемент может врезаться в другие элементы, создавая «ступенчатый» 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/](http://www.complexspiral.com/publications/containing-floats/).

Есть много способов избавиться от проблем, возникающих с перемещаемыми элементами. Хорошо, когда под рукой есть не одно, а несколько решений, поэтому перечислю наиболее популярные из них.

- **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 `<div>`, содержащем перемещаемый элемент (то есть прямо перед закрывающим тегом `</div>`). 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 — `<br>` (HTML) или `<br/>` (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.

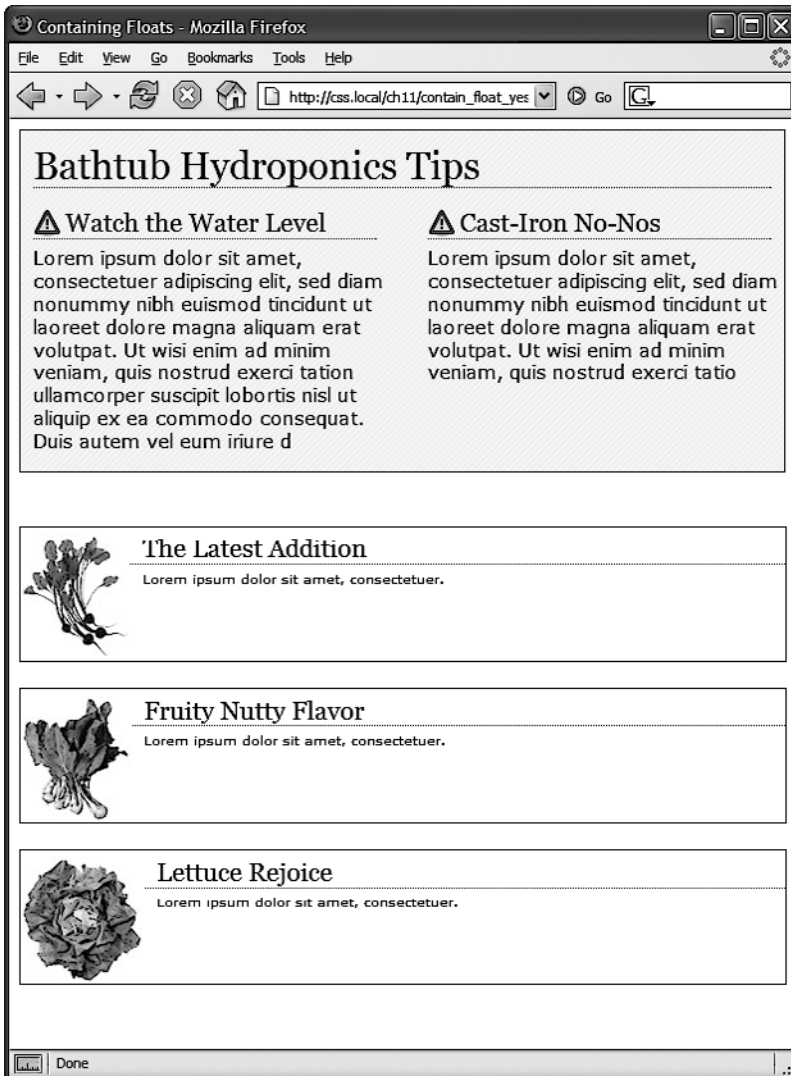
- **Сделайте перемещаемым элемент-контейнер.** Более легкий путь состоит в том, чтобы просто сделать перемещаемым тег `<div>`, который содержит плавающие элементы. Перемещаемый контейнер `<div>` расширяется так, чтобы полностью вмещать любые плавающие элементы. На рис. 13.8, *вверху*, тег `<div>`, содержащий 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;
```

Свойство `overflow:hidden` — одно из странностей CSS. Оно принуждает контейнер расширяться и содержать плавающие элементы. 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 раскрывающиеся меню внутри другого тега, и, когда появляются раскрывающиеся элементы, кажется, что они находятся за пределами элемента-контейнера. Если это так, применяйте какой-либо другой способ из описанных на этих страницах.



**Рис. 13.8.** Есть несколько способов сохранить плавающие элементы внутри содержащего их блока. Можно сделать плавающим сам блок (*вверху*) или использовать специальную комбинацию CSS (*внизу*)

- **Воспользуйтесь методом Micro Clear Fix.** Эта технология, созданная Николасом Галлахером, позволяет добавлять к тегу, который содержит плавающие элементы, всего несколько стилей и имя класса. Она является самой новой в длинной эволюции методов, применяющих псевдокласс `:after`. Чтобы воспользоваться ею, нужно добавить к таблице стилей два различных стиля, один из которых будет применяться всеми современными браузерами, а другой – Internet Explorer 7 и более ранними версиями. Все это имеет следующий вид:

```
.clear:after {
```



```
content: " ";
display: table;
clear: both;
}

.clear {
zoom:1;
}
```

Последний стиль заставляет Internet Explorer 6 и 7 «получить разметку», как описано во врезке «Информация для опытных пользователей. Получили разметку?» в подразделе «Предотвращение выпадения плавающего элемента с помощью свойства `box-sizing`» данного раздела. Если поддержка этих браузеров вас не интересует, можно удалить этот стиль из своей таблицы стилей.

После добавления этих стилей к таблице стилей, к `div`-элементу, содержащему исчезающие плавающие элементы, нужно просто добавить класс: `<div class="clear">`. Если используются элементы секционирования HTML5, например `<article>` или `<footer>`, класс нужно добавить и к ним: `<article class="clear">`. Посмотрите на нижнее изображение на рис. 13.8. Эта технология работает очень надежно, но, в отличие от предыдущих двух технологий, вам потребуется наличие на странице дополнительного кода HTML (атрибута `class`).

#### ПРИМЕЧАНИЕ

Использование свойства `zoom` приводит к тому, что страница не проходит проверку CSS-кода на корректность. Чтобы решить эту проблему, вы можете поместить данное выражение (наряду с любыми другими специальными стилями для Internet Explorer) во внешнюю таблицу стилей и присоединить ее к вашим веб-страницам, используя любую методику, описанную в гл. 17.

## ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

### Простой способ создания нескольких колонок

В CSS3 представлен модуль многоколоночной разметки: он позволяет делить один элемент (например, заполненный текст `div`-элемент) на три, четыре столбца или более. Этот модуль предоставляет CSS-свойства для определения количества столбцов, пустых пространств между ними и добавления прямых линий между столбцами. Многоколоночность предназначена для имитации газетных или журнальных колонок, где один длинный рассказ простирается от верхней к нижней части одной колонки, продолжается в верхней части второй колонки, опускается по этой колонке вниз и т. д. То есть она не предназначена для помещения в один ряд колонок не связанного между собой содержимого (например, боковой панели со ссылками и основной колонки, содержащей статью). Кроме того, для каждой отдельной

колонки нельзя указать ширину, отличную от ширины других колонок, все они будут одинаковой ширины.

Разметка в несколько колонок хорошо смотрится в журналах, у которых установлена определенная высота, и вы можете видеть всю страницу, что упрощает чтение внизу колонки и переход к верхней части следующей колонки. Но высота монитора посетителя вам неизвестна, длинная колонка текста может заставить посетителя осуществлять прокрутку, чтобы добраться до ее нижней части, затем прокручивать страницу вверх, чтобы прочитать следующую колонку. То есть, если не будет полной уверенности, что колонки у вас достаточно короткие, вы можете заставить людей прокручивать страницу вниз-вверх, чтобы прочитать ее содержимое.

## ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

Тем не менее технология создания нескольких колонок хорошо подойдет для длинных маркированных списков с короткими пунктами. Вместо одного слишком длинного маркированного списка можно будет распределить пункты по странице, расположив их в нескольких колонках, экономя драгоценное вертикальное пространство. Основной синтаксис очень прост: для задания количества колонок можно воспользоваться свойством `column-count`, для задания разрывов между ними — свойством `column-gap`, а для рисования линий между колонками — свойством `column-rule`. Чтобы установить стиль, размер и цвет линий между колонками, используется точно такой же синтаксис, как и для границ.

Эти свойства применяются к тегу, содержащему элементы, которые нужно разбить на колонки. Предположим, например, что имеется несколько абзацев текста, расположенных в теге `<div>`. При применении этих свойств к такому `div`-контейнеру абзацы будут простираются на несколько колонок.

Если, например, этому контейнеру присвоен класс по имени `multicol`, то для получения трехколоночной

конструкции с разрывом в 1 em и точечными линиями между колонками можно создать следующий стиль:

```
.multicol {
  column-count: 3;
  column-gap: 1em;
  column-rule: 1px dotted black;
}
```

Следует напомнить, что Internet Explorer 9 и более ранние версии не разбираются в свойствах задания нескольких колонок. Кроме того, нужно использовать префиксы производителей для Chrome и Safari (`-webkit-column-count`, `-webkit-column-gap` и `-webkit-column-rule`) и для Firefox (`-moz-column-count`, `-moz-column-gap` и `-moz-column-rule`). А вот Opera понимает версию этих свойств без префиксов.

Существуют и другие свойства для задания нескольких колонок, о которых можно прочитать на официальной странице W3C: [www.w3.org/TR/css3-multicol/](http://www.w3.org/TR/css3-multicol/). Кроме того, есть простое введение в создание нескольких колонок по адресу <http://dev.opera.com/articles/view/css3-multi-column-layout/> и интерактивное средство для создания и предварительного просмотра нескольких колонок по адресу [www.aaronlumsden.com/multicol/](http://www.aaronlumsden.com/multicol/).

## 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 панелей на рис. 13.9, *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 содержанием. 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 овалом областях на рис. 13.9, *внизу*.

### ПРИМЕЧАНИЕ

Новая гибкая блочная модель решает проблему разной высоты столбцов в строке. К сожалению, поддержка этой модели в браузерах пока что оставляет желать лучшего. Обратите внимание на следующую далее врезку «Информация для опытных пользователей». На подходе более совершенные способы разметки».





**Рис. 13.9.** Левая и правая боковые панели (вверху) показывают, как четкие фоновые цвета помогают визуально разделять области страницы. Когда фон боковой панели резко прерывается (внизу), появляется пустое пространство, что выглядит непривлекательно

Но, как и у большинства проблем, связанных с CSS, здесь существует несколько обходных путей. Наиболее действенным из них является метод искусственных столбцов, впервые предложенный CSS-гигантом Дэном Седерхольмом. Секрет состоит в том, чтобы добавить фоновые изображения к тегу, который включает в себя «приземистую» боковую панель и другие столбцы на странице. Скажем, в вашем HTML-коде есть два тега `<div>`, в которых определяется содержимое для левой боковой панели и главного текста страницы:

```
<div class="sidebar">Sidebar content here</div>
<div class="main">Main content for page, this column has a lot of text
and is much taller than the sidebar.</div>
```

Тег `<div>` боковой панели перемещается к левому краю страницы, и ширина его равна 170 пикселям. Поскольку на боковой панели немного текста, ее содержимое короче, чем основное содержимое. Предположим, что вы задаете этот `<div>`-контейнер таким образом:

```
<div class="wrapper">
<div class="sidebar">Sidebar content here</div>
<div class="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 него, так что, даже если содержимое `div`-контейнера с классом `main` очень длинное, охватывающий блок `div` с классом `wrapper` будет такой же высоты. В этом все волшебство: создайте стиль для охватывающего тега `<div>` с фоновым изображением, ширина которого равна ширине боковой панели, подобрав нужный фоновый цвет. Таким образом, если фоновое изображение повторяется вертикально, оно формирует сплошную полосу высотой, равной высоте `div`-контейнера (см. рис. 13.9, *вверху*).

```
.wrapper { background: url (images/col_bg.gif) repeat-y left top; }
```

Браузеры показывают это фоновое изображение прямо под боковой панелью, создавая иллюзию того, что у панели есть фоновый цвет.

---

#### 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. Нужно добавить два охватывающих блока:

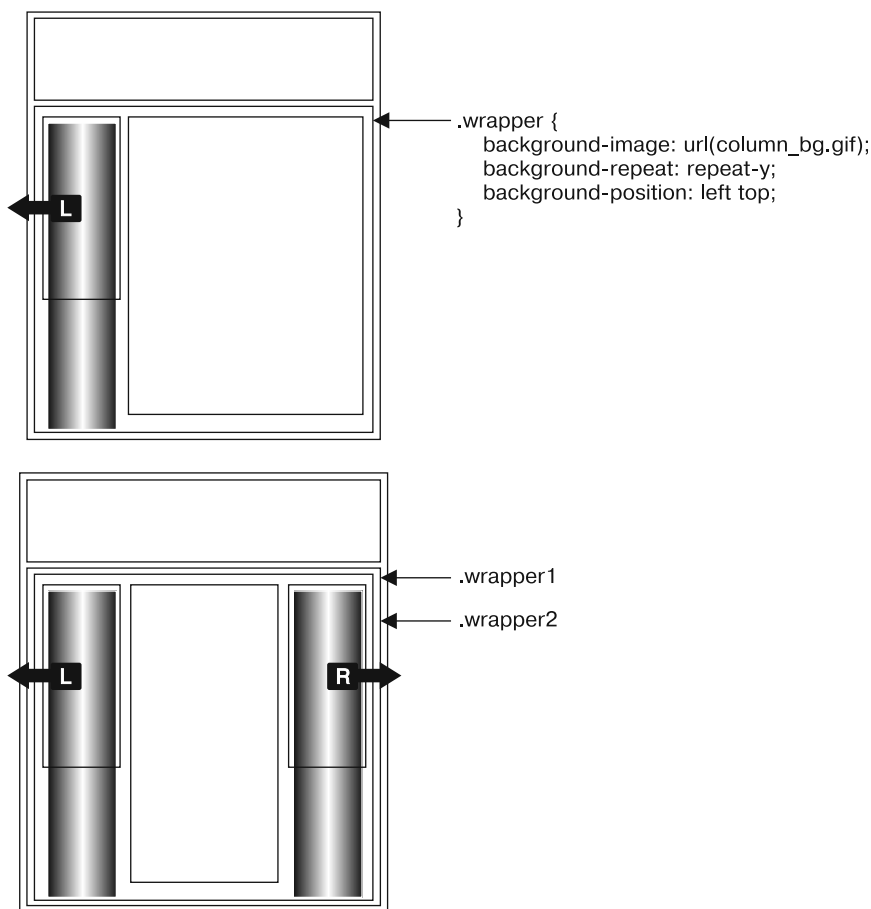
```
<div class="wrapper1">
  <div class="wrapper2">
    <div class="sidebar1">Sidebar content here</div>
    <div class="sidebar2">Second sidebar</div>
    <div class="main">Main content for page, this column has a lot of text
      and is much taller than the two sidebars.</div>
  </div>
</div>
```

## ПРИМЕЧАНИЕ

Если «упаковка» и каждый столбец имеют фиксированную ширину, вы можете создать видимость искусственного столбца для левой и правой панели всего с одним изображением и охватывающим блоком. Просто сделайте рисунок таким же широким, как «упаковка», и задайте для левой стороны изображения цвет и ширину левой боковой панели, а для правой стороны — цвет и ширину правой боковой панели. Центральная часть должна соответствовать фоновому цвету центрального столбца.

Если первая боковая панель появляется на левой стороне страницы, а вторая боковая панель — на правой стороне, то создаются два стиля. Примените один стиль к первому охватывающему тегу `<div>`, чтобы добавить фон к левой боковой панели; другой же стиль примените ко второму охватывающему тегу `<div>`, добавив фон к правой боковой панели (рис. 13.10, *внизу*).

```
.wrapper1 { background: url(images/col1_bg.gif) repeat-y left top; }
.wrapper2 { background: url(images/col2_bg.gif) repeat-y right top; }
```



**Рис. 13.10.** Для получения фоновых изображений на всю высоту плавающих столбцов нужно обратиться к некоторым дополнительным тегам `<div>`. Добавляя к ним фоновые цвета, вы создаете видимость столбцов одинаковой высоты

При добавлении фонового изображения к правому столбцу убедитесь, что вы размещаете изображение вверх справа во второй «упаковке» так, что оно свободно простирается под второй боковой панелью на правой стороне страницы.

Основная проблема технологии искусственных столбцов связана с тем, что ее очень трудно заставить работать, когда у всех столбцов ширина задана в процентах. Если ширина боковых панелей задана в процентах от ширины окна браузера, они могут быть уже или шире в зависимости от характеристик монитора посетителя. Технология искусственных столбцов требует помещения графики в элемент, являющийся контейнером. Эта графика имеет определенную ширину и не будет масштабироваться при изменениях ширины окна браузера, а соответственно, и ширины столбцов.

Одним из удачных обходных маневров является использование имеющихся в CSS3 линейных градиентов для добавления градиента к элементу-контейнеру. Градиент становится для столбцов цветом фона внутри элемента-контейнера. Теперь можно подумать: «Мне, конечно, нравится радуга, но я хочу, чтобы у каждого столбца был обычный цвет, а не градиент с цветами, перетекающими от одного к другому». Но можно ведь получить и чистые цвета.

Как уже говорилось, линейные градиенты позволяют устанавливать цветовые опорные точки, то есть места, где новый цвет определяется в качестве части градиента. Если первая цветовая опорная точка, например, белая и вторая цветовая опорная точка белая, градиент будет переходить от белого цвета к белому. То есть цвет не будет изменяться, в результате получится один сплошной цвет. Кроме того, можно установить цветовую опорную точку в том же месте в качестве другой такой точки, позволяющей второму цвету взять начало сразу же после первого без какого-либо градиентного эффекта.

Предположим, что есть конструкция из трех столбцов. Первый столбец имеет ширину 25 %, второй — 50 %, и третий — 25 %. Нужно чтобы у первого столбца фоновым цветом был красный, у второго — белый, а у третьего — синий. Для этого сделайте следующее.

1. Заключите все три столбца в элемент-контейнер:

```
<div class="wrapper">  
  <div class="sidebar1"> ... сюда помещается содержимое ...</div>  
  <div class="main"> ... сюда помещается содержимое ...</div>  
  <div class="sidebar2"> ... сюда помещается содержимое ...</div>  
</div>
```

Контейнер является тем самым элементом, к которому добавляется градиент. Кроме того, если все три столбца внутри контейнера являются плавающими, то для содержания всех этих плавающих элементов нужно будет воспользоваться одним из методов, рассмотренных ранее в разделе «Решение проблем плавающих элементов».

2. Добавьте линейный градиент с цветовыми опорными точками, соответствующими ширине столбцов:

```
.wrapper {  
  background-image: linear-gradient(left,  
    red 0%,  
    red 25%,  
    white 25%,
```

```

white 75%,
blue 75%,
blue 100%);
}

```

Красный цвет займет пространство от 0 (от левого края контейнера) до 25 %. Так как используется один и тот же цвет, градиента не будет. Затем на отметке 25 %, там же, где закончился красный цвет, начинается белый цвет, поэтому градиента опять не будет. Белый цвет займет пространство до 75 % отметки и будет просто сплошным белым цветом. Затем точно там же, где заканчивается белый цвет, начнется синий цвет, который займет пространство до 100 % отметки (до правого края контейнера). То есть получатся три столбца со сплошными фоновыми цветами, ширина которых будет изменяться по мере изменения ширины окна браузера. Отлично.

Разумеется, нужно включить также и версии свойств с префиксами производителей.

### 3. Добавьте версии свойств с префиксами производителей:

```

.wrapper {
background-image: -webkit-linear-gradient(left,
red 0%,
red 25%,
white 25%,
white 75%,
blue 75%,
blue 100%);

background-image: -moz-linear-gradient(left,
red 0%,
red 25%,
white 25%,
white 75%,
blue 75%,
blue 100%);

background-image: -o-linear-gradient(left,
red 0%,
red 25%,
white 25%,
white 75%,
blue 75%,
blue 100%);

background-image: linear-gradient(left,
red 0%,
red 25%,
white 25%,
white 75%,
blue 75%,
blue 100%);
}

```

Недостаток использования линейного градиента заключается в том, что он работает только с чистыми цветами (и, разумеется, с градиентами, если вы того пожелаете), поэтому вы не можете применять в качестве фона графику или границы вокруг каждого столбца. Кроме того, Internet Explorer 9 и более ранние версии вообще не понимают, что такое градиенты.

#### ПРИМЕЧАНИЕ

Есть еще несколько способов заставить столбцы появляться с одинаковой высотой. Обзор различных технологий можно найти в блоге CSS-эксперта Криса Койера по адресу <http://css-tricks.com/fluid-width-equal-height-columns/>. Его рассуждения насчет метода гибких блоков не следует принимать во внимание, поскольку статья о них была написана до того, как консорциум W3C выпустил финальную версию синтаксиса flexbox.

### ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

#### На подходе более совершенные способы разметки

В CSS3 включен ряд многообещающих новых методов управления потоком содержимого веб-страницы. В большинстве браузеров (за исключением Internet Explorer 9 и более ранних версий) уже можно использовать модуль многоколоночной разметки, направляя текст по нескольким колонкам, как в газетных и журнальных полосах (см. предыдущую врезку «Информация для опытных пользователей. Простой способ создания нескольких колонок»).

Кроме этого, новый модуль гибких блоков *flexible box* предлагает способы реализации большинства тех задач, попытка решения которых связывалась с применением плавающих элементов, например создание смежных столбцов разной ширины. Но flexbox (как часто называют этот модуль) способен на большее, включая управление порядком отображения столбцов независимо от их порядка следования в HTML или изменение их отображения со «слева направо» на «справа налево». Flexbox имеет довольно простой синтаксис, предоставляя множество средств для управления разметкой. Этот модуль даже поддерживается большинством браузеров.

К сожалению, рабочая группа CSS внесла в синтаксис flexbox существенные изменения. Были изменены названия свойств и добавлены новые свойства, поэтому реализация модуля в браузерах на момент написания

данной книги полностью устарела и использование старого синтаксиса не рекомендуется. Предсказывалось, что даже Internet Explorer 10, который при написании этих строк еще не был выпущен, поддерживает старый, вышедший из употребления синтаксис flexbox. То есть для надежного применения flexbox на веб-сайтах должно пройти какое-то время. Для получения дополнительной информации о модуле flexbox нужно обратиться к источнику по адресу <http://www.w3.org/TR/css3-flexbox/>. Следует также избегать любых руководств по flexbox, написанных до 2012 года, поскольку во всех них используется старый синтаксис. Чтобы научиться определять разницу между старым и новым flexbox, прочитайте статью по адресу <http://css-tricks.com/old-flexbox-and-new-flexbox/>.

Кроме того, на горизонте уже появился довольно сложный модуль разметки *grid* ([www.w3.org/TR/css3-grid-layout/](http://www.w3.org/TR/css3-grid-layout/)). Он призван упростить процесс определения исходной структуры строк и столбцов, по которой можно выравнивать элементы страницы. Хотя на создание этого модуля был затрачен большой объем работы, на конец 2012 года модуль еще не был готов и браузеры его не поддерживают.

И наконец, некоторые улучшения текстовой разметки были предложены компанией Adobe (<http://html.adobe.com/webstandards/>).

## Предотвращение выпадений плавающих элементов

Может случиться так, что внезапно один из столбцов просто опустится под другими (рис. 13.11, *верху*). Видно, что есть много места для всех столбцов, чтобы они

отлично сосуществовали бок о бок, но этого не получается. Можно сказать, что произошло выпадение плавающего элемента.



**Рис. 13.11.** Когда ширина плавающих элементов хоть на волосок шире 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*)

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 плавающий элемент может опуститься вниз. Кроме того, не забывайте о блочной модели в CSS: как обсуждалось в подразделе «Вычисление фактических размеров блочных элементов» раздела «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. 13.11. Как вы можете видеть на верхнем изображении, эти три столбца не согласуются. Рассмотрим, из чего состоят элементы приведенной страницы.



- **Охватывающий блок.** Охватывающий тег `<div>` с фиксированной шириной включает в себе весь дизайн. Его ширина равна 760 пикселям, таким образом, все три столбца не могут быть в сумме шире, чем это значение.
- **Первая боковая панель (с левой стороны).** Ее ширина — 150 пикселей, но у нее также есть отступы по 10 пикселей, что делает общую ширину панели равной 170 пикселям (150 пикселей ширины панели + 10 пикселей левого отступа + 10 пикселей правого отступа).
- **Главное содержимое.** Имеет ширину 390 пикселей, а также включает по 1 пикселу обеих границ и по 15 пикселей левого и правого полей, что делает полную ширину равной 422 пикселям (390 пикселей ширины содержимого + 1 пиксел левой границы + 1 пиксел правой границы + 15 пикселей левого поля + 15 пикселей правого поля).
- **Вторая боковая панель (с правой стороны).** Ширина этого элемента равна 150 пикселям. Он также включает в себя по 10 пикселей левого и правого отступов; в результате получается 170 пикселей, как и для первой боковой панели.

Фактическая ширина всех добавленных элементов составляет 762 пикселя. Это на 2 пикселя больше, чем ширина охватывающего тега `<div>`. На рис. 13.11, *посередине*, показана рамка вокруг тега `<div>` для главного содержимого, которая обозначает его полную ширину вместе с полями. Всего двух лишних пикселей ширины (обведены кружком) достаточно для того, чтобы заставить столбец опуститься вниз.

Решение заключается в удалении области размером 2 пикселя из любого элемента. Для этого измените левое и правое поля главного содержимого с 15 до 14 пикселей, что даст вам дополнительное место, необходимое для размещения всех трех столбцов, которые теперь будут находиться рядом.

Подведем итог: выпадение плавающих элементов вызывается недостатком места для содержания всех столбцов.

Другой способ облегчения данных математических вычислений заключается в отказе от установки границ или отступов для `div`-контейнеров или элементов, содержащих столбцы. Тогда при установке трех столбцов на 20, 60 и 20 % ширины соответственно вы будете знать, что они поместятся рядом друг с другом, поскольку вместе они составляют 100 % и нарушающие это положение отступы или границы отсутствуют. Если нужно применить отступы, их можно просто добавить к элементам, находящимся внутри столбцов: например, дать одинаковые левые и правые отступы заголовкам, абзацам и другим элементам внутри `div`-контейнера. Это потребует дополнительной работы, но предотвратит потенциальное выпадение плавающих элементов, вызванное превышением суммарной ширины столбцов значения 100 %.

Если нужны еще и границы, можно применить другую технологию вложенных `div`-контейнеров:

```
<div class="column1">
  <div class="innerColumn">
    ... сюда помещается содержимое...
  </div>
</div>
```

Затем задается ширина внешнего `div`-контейнера, в данном случае с классом `column1`, и добавляются отступы и границы к внутреннему `div`-контейнеру, который



в показанном выше коде имеет класс `innerColumn`. Устанавливать ширину внутреннего `div`-контейнера не нужно, он заполнит всю ширину внешнего столбца автоматически.

## Предотвращение выпадения плавающего элемента с помощью свойства `box-sizing`

Основной причиной выпадения плавающих элементов является странный способ, который используется браузерами для вычисления фактической экранной ширины элемента. Например, вы устанавливаете ширину 100 пикселей, но браузер рисует элемент, занимая ширину 122 пиксела, потому что вы добавили также 10 пикселей на левый и правый отступы и однопиксельную границу вокруг всего элемента. К счастью, есть CSS-свойство, которое позволяет избавиться от столь неприятного способа расчета.

CSS-свойство `box-sizing` позволяет заставить браузер применить другую модель расчета фактической экранной ширины элемента. Этому свойству можно задать одно из трех значений.

- `content-box` — приводит к обычной работе браузера: ширина элемента складывается из толщины левой и правой границ, левого и правого отступов и значения CSS-свойства `width`.

`box-sizing: content-box;`

- `padding-box` — заставляет браузер включать левый и правый отступ в общий расчет ширины элемента. То есть экранная ширина элемента складывается из значения CSS-свойства `width`, а также из левого и правого отступов. Любые границы вокруг элемента в нее не включаются.

`box-sizing: padding-box;`

- `border-box` — включает отступы, границы и значение CSS-свойства `width`. В общем, это то, что вам нужно. При задании этого значения расчеты упрощаются, что помогает избежать выпадения плавающих элементов, особенно при использовании ширины, заданной в процентном отношении в сочетании с пиксельными единицами измерения толщины границ и отступов:

`box-sizing: border-box;`

Приятно также, что такую схему поддерживает большинство браузеров, включая Internet Explorer 8. Для Firefox понадобится префикс производителя — `-moz-box-sizing`, — но все остальные браузеры способны принимать обычную версию без префикса. Это свойство можно смело применять для подавляющего большинства интернет-пользователей. Этим инструкциям не будут следовать только Internet Explorer 7 и более ранние версии. Они прибегают к определению визуальной ширины так, как это делается в браузерах в обычных условиях, поэтому в Internet Explorer 7 будут получаться более широкие элементы, что, возможно, при использовании плавающей разметки приведет к выпадению плавающих элементов. Чтобы понять, нужно ли вам беспокоиться о поддержке Internet Explorer 7, прочитайте врезку «Часто задаваемые вопросы. Стоит ли волноваться насчет Internet Explorer 6, 7 или 8?» в разделе «Важность doctype» гл. 1.

Некоторые веб-разработчики предлагают устанавливать для всех элементов значение `border-box`, чтобы все теги измерялись одинаково. Для этого используется селектор `*`, выбирающий каждый элемент на странице, который нужно поместить в верхней части таблицы стилей вместе с перезапуском CSS:

```
* {
  -moz-box-sizing: border-box;
  box-sizing: border-box;
}
```

## ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

### Получили разметку?

Как вы уже, вероятно, поняли, в Internet Explorer 6 и 7 проявляется много ошибок при отображении страниц, созданных с помощью CSS. Некий базовый CSS-стиль, который прекрасно выглядит в Internet Explorer 8, Firefox или Safari, разрушается в Internet Explorer версий 6 и ниже. К счастью, популярность Internet Explorer 6 и 7 уменьшается и они быстро предаются забвению. Но если их все же нужно поддерживать, пригодится следующая уловка. Оказываясь, от многих недостатков браузеров Internet Explorer можно избавиться путем включения специального, присущего только Internet Explorer свойства, известного как `layout` (разметка). Оно не относится непосредственно к CSS и при этом ничего не делает с правилами HTML. Это просто концепция, встроенная проектировщиками в Internet Explorer (версии 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. Например, чтобы гарантировать, что внутри элемента находятся только плавающие элементы, к CSS можно добавить следующий стиль:

```
.clear { zoom: 1; }
```

Затем добавьте этот класс к любому элементу, содержащему плавающие элементы. Суть в том, что этот стиль

не приводит к реальному масштабированию тех элементов, на которые он ссылается. Хотя присущее только Internet Explorer свойство `zoom` позволяет изменять масштаб элемента страницы (используя JavaScript), оно также включает разметку в Internet Explorer 6 и 7. По причинам, известным только Microsoft, включение разметки заставляет Internet Explorer 6 и 7 содержать внутри тега-контейнера любые плавающие элементы. 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 Internet Explorer, «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». Проверьте правильность кода веб-страниц» в разделе «Важность doctype» гл. 1).

## Обучающий урок. Многоколоночная разметка

В этом обучающем уроке мы рассмотрим, как создавать разметку на несколько столбцов, основанную на плавающих элементах. Кроме того, вы научитесь создавать свободный дизайн на три столбца, дизайн с фиксированной шириной, а также применять некоторые другие методы достижения аналогичного результата.

Чтобы приступить, загрузите файлы с сайта [www.sawmac.com/css3/](http://www.sawmac.com/css3/). Как это сделать, описано в конце гл. 2. Файлы для этого урока находятся в папке 13.

## 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: `<!-- first 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 левой боковой панели.

2. Добавьте открывающий тег `<aside>` для боковой панели: `<aside class = "sidebar1">`. Затем нажмите клавишу **Enter**, чтобы создать новую строку.

Здесь используется HTML5-тег `<aside>`, но для создания данного столбца за просто можно воспользоваться и тегом `<div>`. (На странице имеется ссылка на файл JavaScript, который позволяет Internet Explorer 8 и более ранним версиям применять CSS к элементам HTML5; зачем это нужно, объяснялось во врезке «Обходной прием. Как заставить Internet Explorer 8 понимать HTML5» в разделе «Дополнительные теги в HTML5» гл. 1.)

Если бы вы создавали веб-страницу с нуля, то в этом пункте пришлось бы добавлять HTML-код для боковой панели 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 `sidebar1.txt`, скопируйте его содержимое, а затем вернитесь к файлу `start.html`. Вставьте скопированный HTML-код после тега `<aside>`, который вы создали в шаге 2 (или своего тега `<div>`, если вы решили использовать именно этот тег).

Боковая панель почти готова. Осталось лишь закрыть тег `<aside>`.

4. Сразу же после кода, который вы только что вставили, введите `</aside>`.

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 `<article class = "main">`.

Этот раздел будет хранить главное содержимое страницы. Нужный HTML-код вы также возьмете в другом файле.

6. Откройте файл `main.txt`, скопируйте его содержимое, вернитесь к файлу `start.html` и вставьте скопированный код после тега `<article>`, который вы только что создали. Добавьте закрывающий тег `</article>` точно так же, как в шаге 4. Сохраните HTML-файл.

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. Откройте в текстовом редакторе файл `styles.css`.

Поскольку веб-страница использует внешнюю страницу стилей, новые стили будут добавляться к этой таблице. Теперь вы работаете с двумя файлами, файлом HTML и файлом CSS, поэтому перед просмотром работы в веб-браузере нужно убедиться в том, что сохранены оба файла.

2. Перейдите в конец CSS-файла, там вы увидите комментарий `/* add tutorial styles below here */` (то есть ниже следует добавлять стили, разрабатываемые в рамках урока). Добавьте ниже этого комментария следующий стиль:

```
.sidebar1 {  
  float: left;  
  width: 20%;  
}
```

Он перемещает боковую панель в левую сторону страницы, задает ей ширину 20%. Свойство `width` играет в этом стиле важную роль: 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. Здесь ширина задана в процентном отношении, стало быть, она определяется шириной данного контейнера. В нашем случае контейнером является тег `<body>`, который заполняет всю ширину окна браузера. Поэтому экранная ширина боковой панели будет зависеть от ширины окна того браузера, который используется посетителем.

3. Сохраните файлы HTML и CSS, и просмотрите файл `start.html` в веб-браузере.

Боковая панель теперь представляет собой столбец, выровненный по левому краю. Когда текст главного столбца достигает основания боковой панели, он обтекает вокруг основания боковой панели, как показано на рис. 13.12. Хотя это и типично для плавающих элементов, это не то, что нам сейчас нужно. Чтобы основной текст содержимого появился в виде отдельного столбца, следует добавить достаточное по размеру левое поле. Это позволит структурировать основную текст за пределами боковой панели.

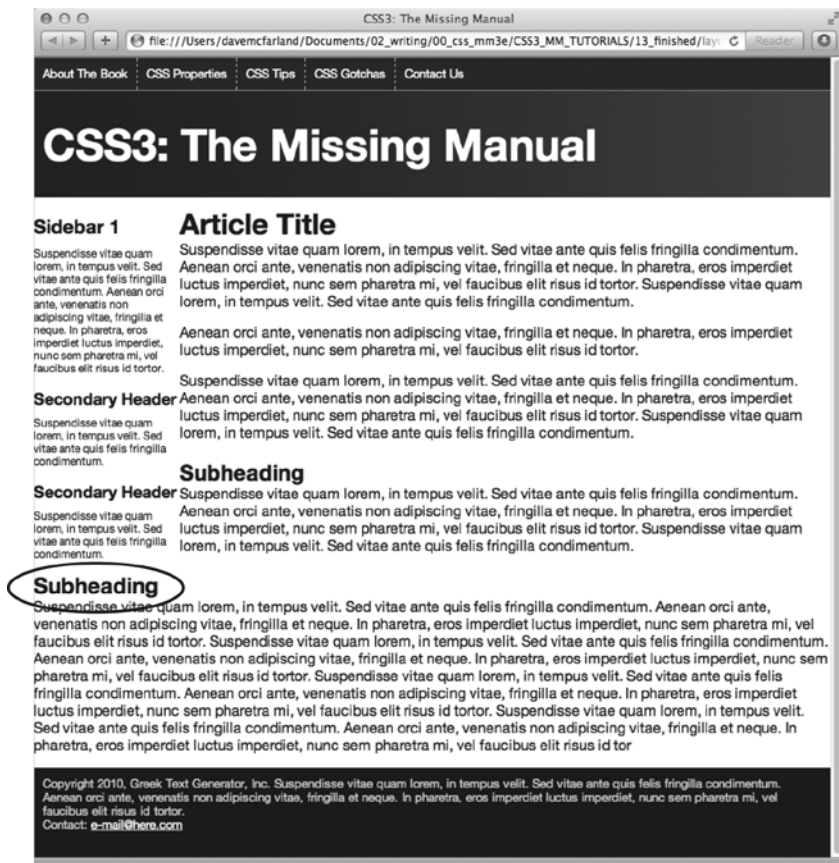
4. Создайте стиль для второго столбца:

```
.main {  
  margin-left: 22%;  
}
```

Поскольку ширина боковой панели составляет 20%, поле размером 22% отодвигает основное содержимое на дополнительные 2%, создавая промежуток

между этими двумя столбцами. Это дополнительное пустое место не только облегчает чтение текста, но и улучшает внешний вид страницы.

Посмотрите теперь на страницу и увидите, что получилась разметка с двумя столбцами.



**Рис. 13.12.** Плавающий элемент в действительности не создает столбец на странице. Он просто смещает любое содержимое, которое обтекает его, до того места, где он заканчивается (обведено овалом). После этого содержимое вновь переходит на свое место под элементом

## Добавление еще одного столбца

Как вы могли видеть, дизайн с двумя столбцами создать несложно. Добавив третий столбец, вы сможете предоставить 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 `sidebar2.txt`. Скопируйте из него весь HTML-код, а затем вернитесь к файлу `start.html`.

HTML-код для следующего столбца помещается после основного текста внутри тега `<article>`.

2. Найдите ближе к концу файла HTML-комментарий `<!-- second sidebar goes here -->` (то есть сюда помещается вторая боковая панель). Щелкните на пустой строке ниже этого комментария.

Зачастую, когда для структурирования страницы используется много `div`-контейнеров, найти нужный закрывающий `</div>`-тег бывает нелегко. Вот почему HTML-комментарии, такие как этот, могут помочь идентифицировать и отслеживать HTML-код на вашей странице.

3. Введите `<aside class="sidebar2">`, нажмите **Enter** и вставьте HTML-код, который вы скопировали в шаге 1. Вновь нажмите клавишу **Enter** и добавьте закрывающий тег `</aside>`. Сохраните HTML-файл.

Закрывая тег `<div>`, вы заканчиваете HTML-код третьего столбца страницы. Теперь приступим к разработке стиля для этого столбца.

4. Откройте в текстовом редакторе файл `styles.css`. Под стилем `.main`, который вы создали в шаге 4 в предыдущем подразделе, добавьте новый стиль во внутреннюю таблицу стилей:

```
.sidebar2 {  
  float: right;  
  width: 20%;  
}
```

Задав такой стиль, вы передвигаете столбец в правую часть страницы, чтобы по обе стороны главного содержимого располагались боковые панели.

5. Сохраните все файлы и просмотрите файл `start.html` в веб-браузере.

Теперь вы должны увидеть нечто странное. Вторая боковая панель появляется ниже основного содержимого и даже набегаёт на нижний колонтитул. Проблема связана с порядком следования исходного кода HTML. Когда задается плавающий элемент, то его обтекает и появляется после него только тот HTML, который следует в коде за этим плавающим элементом. То есть, пока HTML-код второй боковой панели следует за основным содержимым, он появляется *рядом* с основным содержимым, а после него.

Справиться со сложившейся ситуацией можно двумя способами. Можно переместить HTML-код второй боковой панели (второй элемент `<aside>`), поставив его перед HTML-кодом основного содержимого (перед элементом `<article>`). Тогда первая боковая панель переплывет влево, а вторая — вправо, а основное содержимое просто уйдет вверх между ними.

Можно просто сделать плавающим основное содержимое. Если установить его ширину таким образом, чтобы ширина всех трех столбцов не превышала 100 %, все три столбца будут располагаться рядом. Для данного примера будет применен второй вариант.

6. Отредактируйте стиль `.main` следующим образом:

```
.main {  
  float: left;  
  width: 60%;  
}
```

Если сохранить CSS-файл и просмотреть файл `start.html` в браузере, вы увидите, что все съехало! Спокойно: это нижний колонтитул попытался охватить все

плавающие элементы и создал тем самым неразбериху. Как уже ранее говорилось, когда элемент превращается в плавающий и его обтекает другой элемент, фоновый цвет и границы этого элемента фактически расширяются под плавающий элемент. Странно, все вроде правильно, а результат вызывает разочарование. Вполне очевидно, что для решения проблемы нужно, чтобы нижний колонтитул опустился ниже плавающих элементов.

7. После стиля `.sidebar2` добавьте еще один стиль:

```
footer {
  clear: both;
}
```

Как уже упоминалось, свойство `clear` может заставить элемент опуститься ниже столбцов. В данном случае оно выталкивает нижний колонтитул ниже столбцов (рис. 13.13).

## Добавление разрядки

Три столбца у вас уже есть, но текст выглядит слишком скученно. Эти три столбца практически касаются друг друга, и текст на правой боковой панели слишком близко прижимается к краю окна браузера. Исправить ситуацию помогут небольшие отступы.

1. Добавьте отступы к стилям `.sidebar1`, `.main` и `.sidebar2`, придав им следующий вид:

```
.sidebar1 {
  float: left;
  width: 20%;
  padding: 0 20px 0 10px;
}

.main {
  float: left;
  width: 60%;
  padding: 0 20px 0 20px;
}

.sidebar2 {
  float: right;
  width: 20%;
  padding: 0 10px 0 20px;
}
```

Здесь используется сокращенная форма записи свойства `padding`. Числа представляют верхний, правый, левый и нижний отступы. То есть в стиль `.sidebar1` добавлен верхний нулевой отступ, 20 пикселей отступа справа, нижний нулевой отступ и 10 пикселей отступа слева.

Если сохранить файл `styles.css` и просмотреть в браузере файл `start.html`, станет заметна еще одна проблема — выпадение плавающего элемента. В результате добавления отступов каждый столбец стал шире, а поскольку общая ширина столбцов (20 % + 60 % + 20 %) уже составила в сумме 100 %, дополнительные



отступы заставили третий столбец опуститься под первые два столбца. Это нужно срочно исправить!

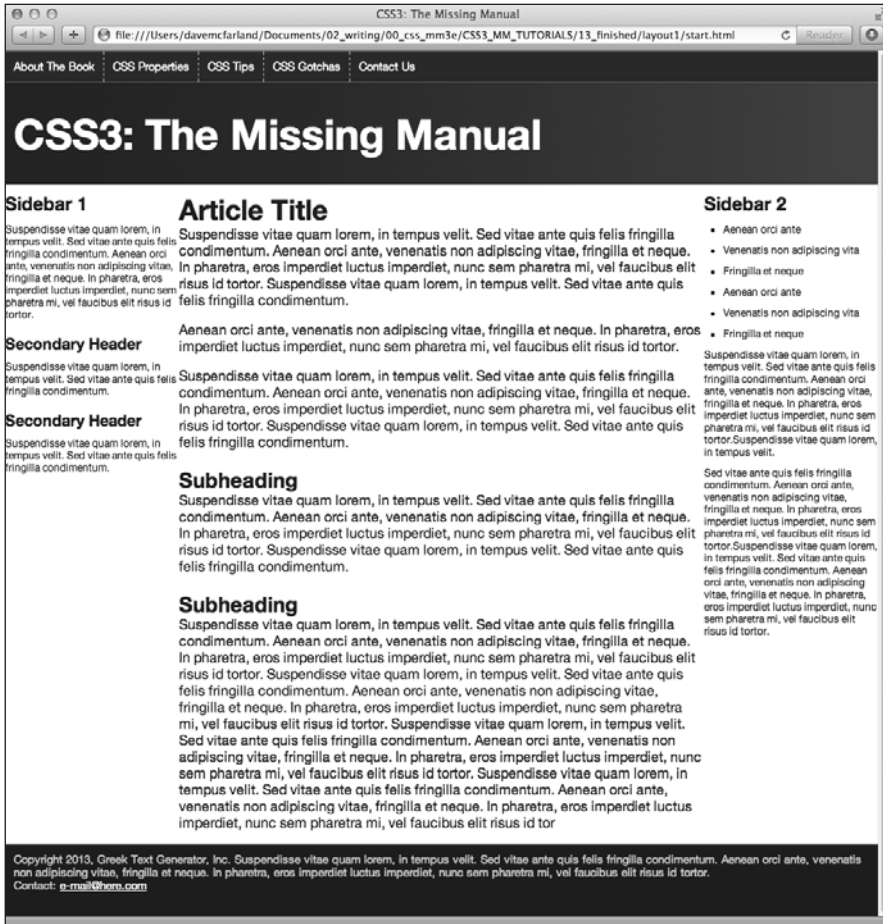


Рис. 13.13. Используя плавающие элементы, можно заставить все три элемента выстроиться рядом в три столбца

Эту проблему можно решить несколькими способами. Во-первых, можно убрать отступы из этих стилей и добавить их ко всем внутренним элементам. То есть, добавить 10 пикселей правого и левого отступов к тегам `<h2>`, `<h3>`, `<p>` и `<ul>`. Это довольно трудоемкий процесс.

Во-вторых, можно убрать отступы из стилей в CSS-файла, а затем в файле `start.html` добавить в каждый столбец `<div>`-контейнер следующего вида:

```
<aside class="sidebar1">
  <div class="innerColumn">
    ... сюда помещается содержимое...
  </div>
```



```
</aside>
```

Затем в файле `styles.css` нужно просто создать стиль для добавления отступов:

```
.innerColumn {
  padding: 0 20px 0 10px;
}
```

Поскольку в стиле `.innerColumn` ширина не задается, Контейнер просто разрастается, чтобы занять столбец, а отступы сдвигают все, что находится внутри этого контейнера (заголовки, абзацы и т. д.) внутрь на 10 пикселей. Недостаток такого подхода состоит в необходимости добавления дополнительного кода.

Есть еще один, более простой, гибкий и широко поддерживаемый веб-браузерами способ.

## 2. Добавьте к таблице стилей еще один стиль:

```
* {
  -moz-box-sizing: border-box;
  box-sizing: border-box;
}}
```

В этом стиле используется универсальный селектор. Благодаря ему свойство `box-sizing` применяется к каждому элементу страницы. Установка для этого свойства значения `border-box` заставляет веб-браузеры использовать размер отступов и границ в качестве части CSS-значения `width`. То есть отступы не добавляются к установленному ранее CSS-значению ширины. Тем самым предотвращаются любые выпадения плавающих элементов, поскольку столбцы теперь составляют только 100 % ширины окна браузера.

Единственным недостатком этой технологии является то, что свойство `box-sizing` не распознается Internet Explorer 7 и более ранними версиями, поэтому в этих браузерах будет происходить выпадение плавающих элементов. Чтобы понять, нужно ли переживать за постоянно сокращающееся количество приверженцев этих браузеров, прочитайте врезку «Часто задаваемые вопросы. Стоит ли волноваться насчет Internet Explorer 6, 7 или 8?» в разделе «Важность doctype» гл. 1. Если нужно вести разработку с учетом этих браузеров, воспользуйтесь технологией дополнительного `div`-контейнера, которая рассматривалась в шаге 1.

И наконец, добавьте линии границы, чтобы отделить столбцы друг от друга.

## 3. Отредактируйте стиль `.main`, добавив к нему левую и правую границы:

```
.main {
  float: left;
  width: 60%;
  padding: 0 20px;
  border-left: dashed 1px rgb(153,153,153);
  border-right: dashed 1px rgb(153,153,153);
}
```

Эти свойства приведут к добавлению двух прямых линий по одной с каждой стороны раздела основного содержимого. Если теперь просмотреть страницу в веб-браузере, она должна быть похожа на рис. 13.14.

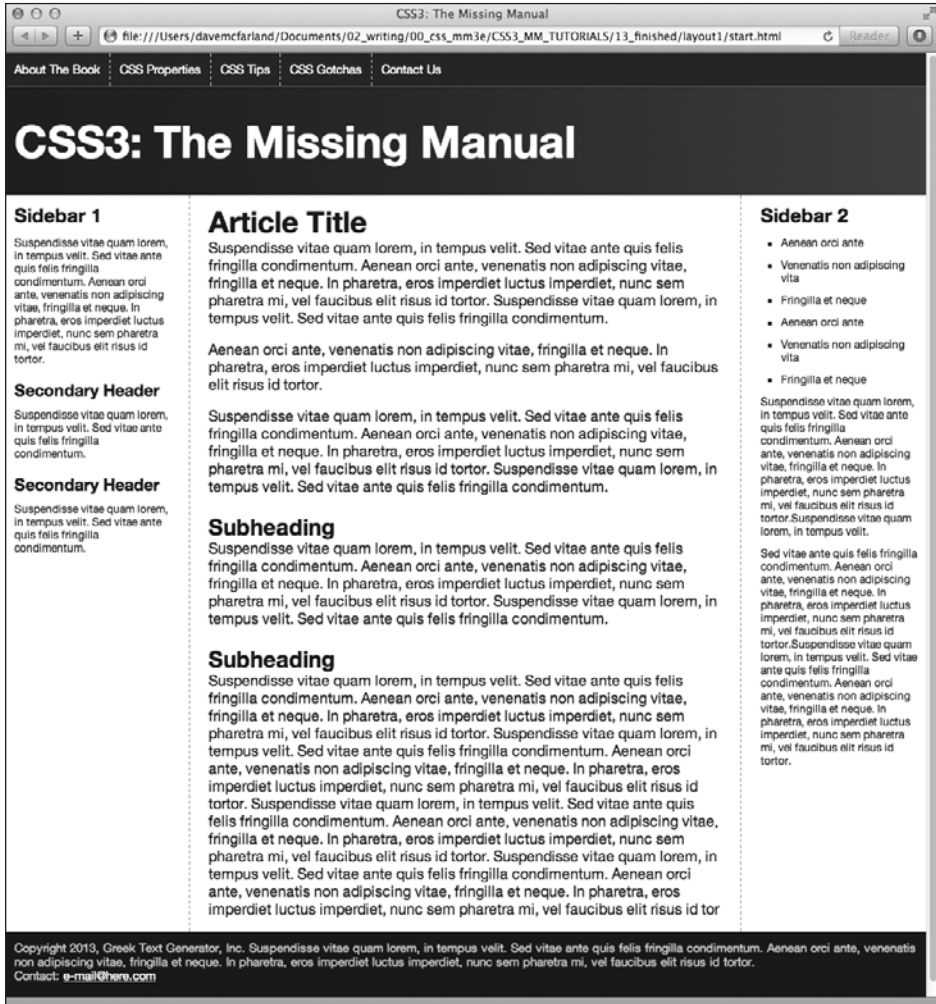


Рис. 13.14. Добавление отступов к столбцам и линий границ между ними создает четкое разграничение столбцов

## Фиксация ширины

В настоящее время у страницы свободный дизайн (см. раздел «Типы разметок веб-страницы» гл. 12), 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. Вернитесь в текстовый редактор к редактированию файла `start.html`. Сразу же за открывающим тегом `<body>` добавьте новый тег `<div>`:

```
<body>
<div class="pageWrapper">
```

Вы заключаете всю страницу в блок, который будет использоваться для управления шириной страницы. Вы должны убедиться, что этот тег закрыт.

2. Добавьте закрывающий тег `</div>` 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. Сохраните HTML-файл, и перейдите к редактированию файла `styles.css`. Добавьте еще один стиль:

```
.pageWrapper {
  width: 960px;
}
```

Если сохранить файлы CSS и HTML и просмотреть файл `start.html` в браузере, вы увидите, что страница действительно заключена в пространство шириной 960 пикселей. Если сделать ширину окна браузера меньше 960 пикселей, появятся полосы прокрутки.

Но вам, разумеется, не нужно устанавливать точную ширину. Если хотите, чтобы страница поместилась в более узком (скажем, имеющем ширину 760 пикселей) окне браузера, лучше избегать установки конкретной ширины. Настоящая проблема заключается в том, что страницу становится трудно читать в слишком широком окне браузера. Другой подход заключается в использовании свойства `max-width`, которое не дает `div`-контейнеру становиться больше определенной величины, но не препятствует более узкому значению его ширины для помещения на небольших экранах. Раз уж вы за это взялись, нужно также отцентровать `div`-контейнер в окне браузера.

4. Измените только что созданный в файле `styles.css` стиль `.pageWrapper`, чтобы он приобрел следующий вид:

```
.pageWrapper {
  max-width: 1200px;
  margin: 0 auto;
}
```

Свойство `max-width` (которое работает даже в Internet Explorer 7) предоставляет свободную разметку, но только до определенного предела. В данном случае, когда окно браузера шире 1200 пикселей, `div`-контейнер не будет становиться шире. Для полей здесь установлено значение `0 auto`, обеспечивающее нулевые поля для верхней и нижней сторон и автоматические поля для левой и правой сторон. Такая автоустановка заставляет браузер автоматически определять размер поля

путем равномерного деления пространства между левой и правой сторонами и центрируя таким образом `div`-контейнер в окне браузера. Теперь страница должна приобрести вид, показанный на рис. 13.15.

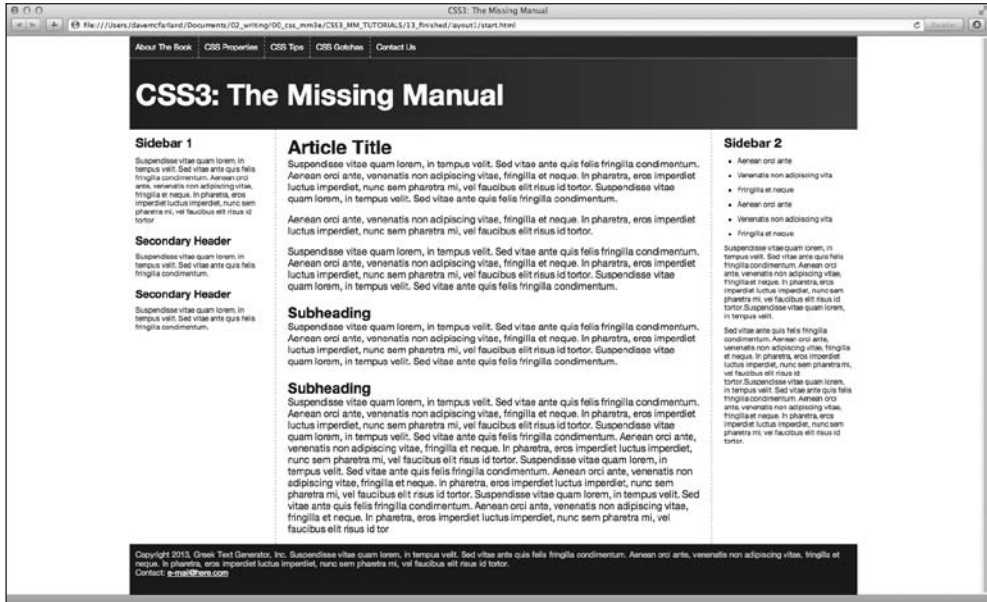
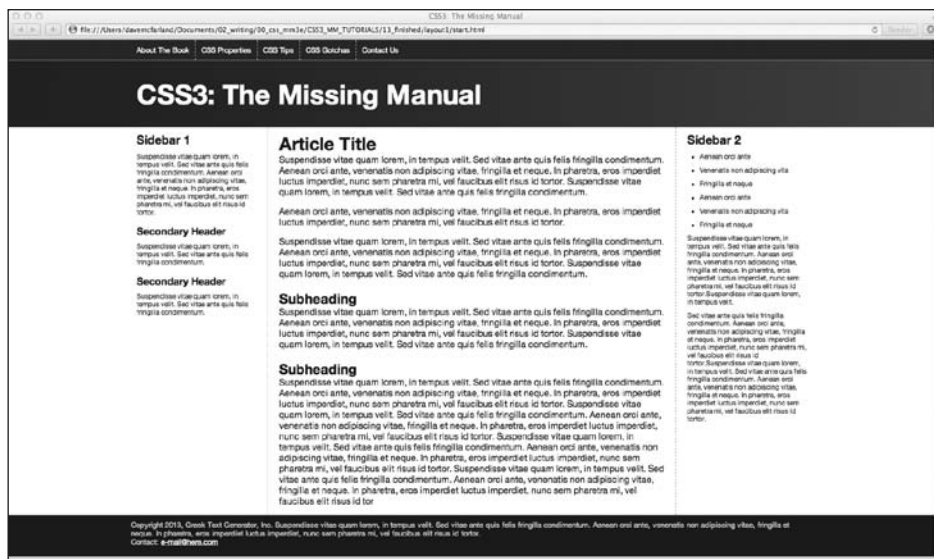


Рис. 13.15. Использование CSS-свойства `max-width` вместо `width` позволяет предоставлять гибкий дизайн, помещающийся в окна браузеров различной ширины, но при этом не приобретающий излишнюю ширину, затрудняющую чтение на слишком широких мониторах

## Смешанный свободный и фиксированный дизайн

Страница выглядит вполне привлекательно, но она могла бы выглядеть лучше, если бы черный фон для навигационной панели вверху и для колонтитула внизу, а также темно-фиолетовый градиент в баннере распространялись на ширину всей страницы (рис. 13.16). Поскольку навигационная панель, баннер и нижний колонтитул находятся внутри `div`-контейнера `pageWrapper`, эти фоновые цвета останавливаются, когда окно браузера шире 1200 пикселей. Вместо этого нужно оставить часть страницы, теги с фоновыми цветами, свободной, при этом ограничив ширину тегов с основным содержанием. Дополнительные колонки можно добавлять простым добавлением дополнительных HTML-контейнеров (`<div>`, `<article>`, `<aside>` и т. д.) и приданием им свойства плавающих влево или вправо. Нужно только не забыть подобрать их ширину, чтобы хватило пространства для их смежного размещения.

Фоновый цвет для навигационной панели применяется к элементу `<nav>` в HTML, фиолетовый градиент применяется к элементу `<header>`, черный фон в нижнем колонтитуле применяется к элементу `<footer>`. Чтобы фон каждого из этих элементов распространился по ширине страницы, они не должны быть ограничены `div`-контейнером `pageWrapper`. Поэтому сначала нужно удалить этот стиль.



**Рис. 13.16.** Создание многоколоночного дизайна ничуть не сложнее создания плавающих HTML-тегов: три столбца, три плавающих элемента

1. Удалите в файле `styles.css` недавно созданный стиль `.pagewrapper`.

`div`-контейнер `pagewrapper` в HTML можно оставить. Этот дополнительный HTML-код не станет обузой, и вы можете захотеть воспользоваться им позже для применения других стилей.

Страница возвращается к своей полностью свободной форме. Теперь нужно ограничить только навигационную панель, текст баннера, текст нижнего колонтитула и основное содержимое, чтобы они не превышали по ширине 1200 пикселей. Для этого нужно немного углубиться в код HTML и посмотреть, с какими элементами следует поработать.

Посмотрите на код HTML в файле `start.html` и найдите тег `<nav>`. Внутри этого тега вы увидите кнопки навигации, созданные с помощью простого неупорядоченного списка. Именно это нам и нужно. Для этого списка можно установить максимальную ширину 1200 пикселей и отцентрировать его на странице, позволив тегу `<nav>` (и его черному фону) для распространения на всю ширину окна браузера. То же самое касается и текста баннера CSS3: The Missing Manual внутри тега `<h1>`. Для него также можно установить максимальную ширину и поля. В нижнем колонтитуле можно найти для управления тег `<p>`.

2. Добавьте в файл `styles.css` еще один стиль:

```
nav ul, header h1, footer p {
  max-width: 1200px;
  margin: 0 auto;
}
```

Этот групповой селектор нацелен на текст панели навигации, баннера и нижнего колонтитула, но не на те элементы, в которых он содержится. Теперь, если сохра-

нить файл CSS и просмотреть в браузере файл `start.html`, вы увидите, что элементы навигации, заголовок и нижний колонтитул не становятся шире 1200 пикселей. Но основное содержимое по-прежнему заполняет все пространство окна браузера. Чтобы исправить ситуацию, нужно заключить три столбца в еще один `div`-контейнер для создания группы, размером и центровкой которой можно управлять.

#### ПРИМЕЧАНИЕ

---

Другой вариант заключается в том, чтобы вставить еще один тег `<div>` в тег `<header>`, в него заключить теги `<nav>` и `<h1>`, а другой тег `<div>` вставить в тег `<footer>`. Можно позволить `<header>` и `<footer>` сохранять полную ширину, ограничивая при этом ширину этих вложенных `div`-контейнеров.

---

3. Откройте в текстовом редакторе файл `start.html`. Непосредственно перед комментарием с представлением первой боковой панели `<!-- first sidebar goes here -->` (то есть сюда помещается первая боковая панель) добавьте `<div class="contentWrapper">`:

```
<div class="contentWrapper">
<!-- first sidebar goes here -->
```

Теперь этот `div`-контейнер нужно закрыть.

4. Перейдите в конец HTML-файла. Между закрывающим тегом `</aside>` и открывающим тегом `<footer>` добавьте закрывающий тег `</div>`, чтобы код HTML приобрел следующий вид:

```
</aside>
</div>
<footer>
```

И наконец, можно добавить это новое имя класса к стилю, созданному в шаге 2.

5. Добавьте новый класс к групповому селектору:

```
nav ul, header h1, footer p, .contentWrapper {
  max-width: 1200px;
  margin: 0 auto;
}
```

Полная версия этого урока находится в папке `13_finished`.

# 14 Адаптивный веб-дизайн

Веб-дизайнерам всегда приходилось решать задачу разработки под различные размеры экрана, от экранов ноутбуков шириной 760 пикселей до огромных широкоэкранных дисплеев. С ростом числа смартфонов и планшетных компьютеров разработка под широкий диапазон экранов с различной шириной и высотой стала еще актуальнее. Некоторые компании зашли так далеко, что стали создавать отдельные веб-сайты, предназначенные только для мобильных устройств (верхние изображения на рис. 14.1). Но при дефиците времени, средств и технических знаний для разработки двух сайтов и программы для веб-сервера, предоставляющей нужный сайт тому или иному устройству, веб-сайты исключительно для мобильных устройств вам вряд ли по плечу.

К счастью, есть еще один, более простой подход, позволяющий создать один сайт, адаптируемый к устройствам с различной шириной экрана (нижние изображения на рис. 14.1). Эта технология, получившая название «адаптивный веб-дизайн», использует различные приемы, заставляющие страницу изменять разметку на основе ширины экрана браузера. Например, на смартфоне страницу можно выложить одной, легкой для чтения колонкой, помещающейся на узком экране (нижнее левое изображение на рис. 14.1), а на более широких мониторах поддерживать разметку в несколько колонок (нижнее правое изображение на рис. 14.1).

## Основы адаптивного веб-дизайна

Читать веб-страницу, состоящую из четырех колонок, на экране телефона, имеющем ширину 320 пикселей, очень трудно. Не менее трудно читать одну колонку текста, растянутую на 2560 пикселей большого монитора настольного компьютера. Адаптивный веб-дизайн (Responsive web design, RWD, этот термин введен Итаном Маркоттом) является попыткой решить эту проблему. Он позволяет изменять всю разметку страницы на основе ширины окна браузера (наряду с другими факторами), допуская создание наиболее легко читаемых представлений для каждого устройства при отсутствии необходимости в создании нескольких версий одного и того же веб-сайта. RWD не является единой технологией или методом. В нем собраны воедино несколько методов CSS и HTML для создания веб-страниц, чья разметка адаптируется к различным экранам.





Рис. 14.1. Многие крупные компании, например Amazon и Target, создают мобильные версии своих сайтов, оптимизированные под дисплеи портативных устройств типа iPhone

ПРИМЕЧАНИЕ

Подходы к RWD Итан Маркотт излагает в своей книге Responsive Web Design издательства A Book Apart. Можно также прочитать статью, посвященную RWD, этого же автора по адресу <http://www.alistapart.com/articles/responsive-web-design>.

В RWD объединены три основные идеи: гибкие сетки (grid) для разметки, гибкая среда для изображений, а также видео- и медиазапросы CSS, предназначенные для создания различных стилей для экранов различной ширины. Гибкие сетки позволяют избавиться от разметок с фиксированной шириной. Поскольку у экранов смартфонов довольно широкий диапазон ширины экрана, создавать страницы с фиксированной шириной нет никакого смысла. Нужна страница, способная расширяться и сужаться, чтобы поместиться на экране конкретного устройства (это



концепция свободного дизайна, рассмотренная в подразделе «Смешанный свободный и фиксированный дизайн» обучающего урока главы 13. Создание гибкой среды дает возможность подбирать масштаб для изображений и видеоматериалов, чтобы они поместились на соответствующем экране — большие фотографии на больших мониторах, фотографии поменьше на небольших экранах и т. д.

И наконец, медиазапросы являются CSS-технологией, позволяющей на основе созданных условий отправлять браузеру различные стили. Например, для экрана, имеющего ширину меньше 480 пикселей, можно отправлять один набор стилей, а для экрана шириной 760 пикселей — другой набор стилей. Но одной шириной дело не ограничивается: можно разрабатывать стили, применяемые только к планшетным устройствам при альбомном просмотре или к устройствам с высокой плотностью пикселей (например, с дисплеем Retina на iPhone и iPad).

## Настройка веб-страницы для RWD

Если у вас есть смартфон, например iPhone или телефон с оперативной системой Android, откройте веб-браузер и зайдите на страницу [www.nytimes.com](http://www.nytimes.com). Должно появиться изображение, подобное показанному на рис. 14.2 (если только New York Times с момента написания книги не сделал свой сайт адаптивным).



**Рис. 14.2.** Веб-сайты, разработанные для браузеров настольных компьютеров, такие как сайт New York Times, на телефонах выглядят слишком мелкими

Это крупноформатный многоколоночный дизайн, втиснутый в небольшое пространство экрана телефона. Поскольку производители телефонов понимают, что большинство веб-сайтов созданы для экранов настольных компьютеров, они заставили свои браузеры вести себя немного непривычно для вас. Мобильные браузеры не отображают страницу на все 100 %; если бы они это сделали, то страница шириной 960 пикселей не поместилась бы на экране и вы бы увидели только часть этой

страницы. Затем, чтобы увидеть всю страницу, вам пришлось бы перемещать поле просмотра в разных направлениях. Вместо этого, чтобы страница поместилась на экране, телефонные браузеры уменьшают масштаб. Конкретный коэффициент уменьшения варьируется в зависимости от характеристики конкретного телефона. Например, Safari на iPhone работает так, будто экран действительно имеет ширину 980 пикселей, и уменьшает страницу, чтобы она поместилась в этих 980 пикселях.

Впрочем, подобное поведение мобильных браузеров позволяет неплохо справиться с большинством сайтов, но с адаптивным веб-дизайном оно сочетается не очень хорошо. Поскольку адаптивные сайты предназначены для получения хорошего представления на смартфонах, уменьшать масштаб изображения на дисплее не нужно, так как текст станет слишком мелким для чтения. К счастью, есть довольно простой способ отмены такого поведения в браузерах мобильных устройств. Нужно просто к разделу `<head>` веб-страницы добавить следующий код (самое подходящее место для этого — непосредственно перед тегом `<title>`):

```
<meta name="viewport" content="width=device-width">
```

Метатеги HTML предоставляют дополнительную информацию о содержимом страницы и могут дать браузерам дополнительные инструкции о способах отображения страницы на дисплее. В данном случае `viewport` обозначает экран браузера, а для атрибута `content` устанавливается ширина экрана браузера, равная ширине экрана смартфона. То есть браузерам мобильных устройств, склонным к уменьшению масштаба, предписывается этого не делать, настроив ширину экрана на текущую ширину дисплея смартфона.

---

#### ПРИМЕЧАНИЕ

Кроме использования метатега `viewport`, есть еще один способ заставить смартфон не предпринимать попыток сжатия вашей страницы, а вывести вместо этого ее в 100%-ном размере. Разработчики из CSS Working Group добавили к CSS правило `@viewport`, которое позволяет делать все то же самое, что и с метатегом `viewport`, но в вашей таблице стилей. Благодаря этому можно отказаться от добавления тега `<meta>` к каждому HTML-файлу сайта и просто добавить одно правило `@viewport` к своей таблице стилей:

```
@viewport { width: device-width; }
```

Это правило нужно добавить в самом начале таблицы стилей до объявления самих стилей. К сожалению, в настоящее время правило `@viewport` еще не работает на всех браузерах и требует для тех браузеров, которые его не понимают, добавлять префикс производителя. Дополнительные сведения о правиле `@viewport` можно получить на сайтах <http://dev.opera.com/articles/view/an-introduction-to-meta-viewport-and-viewport/> и <http://dev.w3.org/csswg/css-device-adapt/>.

---

## Медиазапросы

В CSS3 введено такое понятие, как *медиазапросы*. Они позволяют назначать стили страницам на основе ширины и высоты окна целевого браузера. Используя данный метод, можно создавать пользовательские стили для браузеров мобильных телефонов, планшетных и настольных компьютеров и тем самым настраивать представление сайта таким образом, чтобы он выглядел наилучшим образом на каждом типе устройства.

Смысл адаптивного дизайна состоит в том, чтобы дать посетителям вашего сайта наиболее читаемое и привлекательное представление. Обычно это означает настройку дизайна под наилучший вид на окнах браузеров различной ширины. Многие

разработчики ориентируются на три целевых экрана, относящихся к трем наиболее распространенным устройствам просмотра веб-содержимого: смартфонам, планшетным и настольным компьютерам. При условии существования широкого разнообразия размеров экранов этих устройств (у вас могут быть небольшие телефоны, большие телефоны, 7-дюймовые планшетные компьютеры, 10-дюймовые планшетные компьютеры и т. д.) единой ширины для всех этих устройств не существует. Нужно просто иметь в виду, что целью является хороший внешний вид страницы при различной ширине экрана. Можно просто протестировать различные дизайнерские решения при окнах различной ширины, чтобы увидеть, когда дизайн в четыре колонки должен превращаться в дизайн в две или в одну колонку.

## Стратегии использования медиазапросов

Хотя для принятия решения о внесении изменений в дизайн, позволяющего придать ему наиболее удачный внешний вид на различных устройствах, рекомендуется использовать метод проб и ошибок, существует несколько общих стратегий медиазапросов.

- **Настройка количества колонок (столбцов).** Несколько расположенных в ряд колонок неплохо смотрятся на больших мониторах (и даже на планшетных компьютерах в альбомном режиме), но не подходят для телефонов. Кроме того, четырех колонок, наверное, будет многовато для большинства планшетных компьютеров в книжном режиме экрана, поэтому сведение страницы к двум или трем колонкам, по всей видимости, вполне подходит для медиазапросов, нацеленных на планшетные компьютеры. Исключение плавающих элементов в стилях медиазапросов, предназначенных для планшетных компьютеров, позволяет ставить контейнеры с содержимым страницы друг на друга. Эта технология будет рассмотрена в обучающем уроке этой главы.
- **Гибкие параметры ширины.** Дизайн с фиксированными параметрами ширины можно использовать для браузеров настольных компьютеров. Именно так годами и поступали дизайнеры, но для более узких экранов планшетных компьютеров и телефонов элементы с фиксированным дизайном не поместятся в окно. Страница шириной 960 пикселей будет слишком большой для имеющихся у телефонов 320 или 480 пикселей. Для телефонов и планшетных компьютеров более удачным подходом будет установка параметров ширины `div`-контейнеров с содержимым на `Auto` или на `100%`. Эти установки превратят дизайн вашей страницы из фиксированного в свободный или гибкий. Иными словами, независимо от ширины экрана телефона, `div`-контейнеры поместятся на нем на все 100%. Если человек держит iPhone в книжном режиме (при ширине экрана 320 пикселей), а затем быстро поворачивает телефон горизонтально (изменяя ширину экрана до значения 480 пикселей), `div`-контейнеры, настроенные на `Auto` или на `100%` просто изменят свой размер, чтобы поместиться в новом пространстве.
- **Сжатие пустых пространств.** Обширные пустоты между заголовками, графикой и другими элементами страницы дают свободное пространство для дизайнерских решений на 23-дюймовом мониторе, но приводят к разбросанности и нерационально использованному пространству на небольших телефонных экранах, заставляя посетителей чаще прибегать к прокрутке. Сужение полей и отступов

позволяет поместить на таких небольших экранах больше полезной информации.

- **Настройка размеров шрифтов.** Контраст между большими жирными заголовками и набранным мелким шрифтом основным текстом неплохо выглядит на мониторах настольных компьютеров, но на портативных устройствах излишне крупные заголовки труднее читаются и совершенно необоснованно занимают полезное пространство. А вот некоторое укрупнение на телефонах шрифта основного текста зачастую облегчает его чтение. Иначе говоря, создавая стили медиазапросов, обращайтесь внимание на размеры шрифтов.
- **Изменение навигационных меню.** У вас может быть красиво оформленная горизонтальная панель навигации, которая занимает всю верхнюю часть веб-страницы и состоит из десятка кнопок, направляющих посетителей на просмотр разных разделов вашего сайта. К сожалению, по мере сужения окна браузера эти кнопки могут не поместиться на экран. Они разобьются на нем на две, три и более строки. Пример такого явления будет показан в обучающем уроке данной главы. Возможно, в том, что навигационная панель займет не одну, а несколько строк экрана, не будет ничего страшного, но ведь эта панель может занять в верхней части страницы слишком много места, заставляя пользователей задействовать прокрутку, чтобы добраться до первых строк реального информационного наполнения.

К сожалению, в CSS не предлагается простых и понятных решений этой проблемы. На многих сайтах для динамического превращения навигационного меню в раскрывающееся HTML-меню используется JavaScript. В таком случае это меню занимает весьма небольшое пространство экрана (чтобы этому научиться, обратитесь по адресу <http://css-tricks.com/convert-menu-to-dropdown/>). Но есть и другие решения. Обзор различных подходов к преодолению этой проблемы, применяемых на некоторых сайтах, дан на страницах <http://bradfrostweb.com/blog/web/responsive-nav-patterns/> и <http://bradfrostweb.com/blog/web/complex-navigation-patterns-for-responsive-design/>.

- **Скрытие содержимого на портативных устройствах.** Многие разработчики скрывают содержимое в мобильных версиях сайтов. На мониторе настольного компьютера просмотр нескольких колонок и сотен строк текста дается довольно легко, а вот слишком много информации на телефоне может показаться совершенно излишним. CSS можно использовать для того, чтобы просто скрыть содержимое, которое, на ваш взгляд, не нужно показывать пользователям мобильных устройств, для чего требуется установить для CSS-свойства `display` значение `none`. Но все же нужно иметь в виду, что, скрывая содержимое, вы отстраняете посетителя от той информации, которая предоставляется на вашем сайте. Для тех, кто ранее посещал ваш сайт, используя настольный компьютер, а теперь посещает его же с помощью телефона, будет крайне неприятно увидеть, что совсем недавно просматриваемая ими важная информация теперь куда-то исчезла. Кроме того, даже если вы скроете содержимое с помощью CSS, сам код HTML никуда не денется, заставляя мобильный телефон впустую тратить время и трафик на загрузку неиспользуемого HTML.

- **Использование фонового изображения.** Если поместить на экран 960-пиксельный баннер, то ни один телефон не покажет его без сжатия. Можно, конечно, предоставить достаточно небольшое изображение, способное поместиться на экране телефона, или же воспользоваться вместо него фоновыми изображениями CSS. Можно, например, создать div-контейнер и добавить к нему следующий класс: `<div class="logo">`. Затем в таблице стилей для браузера настольного компьютера установить ширину и высоту div-контейнера, соответствующую размеру большого логотипа, используя свойство `background-image` для вставки изображения в фон. Например:

```
.logo {  
  width: 960px;  
  height: 120px;  
  background-image: url(images/large_logo.png)  
}
```

Затем можно поместить в эту таблицу стилей еще один стиль, используемый для мобильных телефонов, который изменяет размеры div-контейнера и использует другое фоновое изображение:

```
.logo {  
  width: 320px;  
  height: 60px;  
  background-image: url(images/small_logo.png)  
}
```

В разделе «Изменяемые изображения» данной главы вы научитесь масштабировать изображения, вставляемые в HTML с помощью тега `<img>`, чтобы они помещались в окна браузеров различной ширины.

#### ПРИМЕЧАНИЕ

---

Обзор различных стратегий, которыми можно воспользоваться для изменения раскладки страницы для различных устройств, можно найти по адресу [www.lukew.com/ff/entry.asp?1514](http://www.lukew.com/ff/entry.asp?1514).

---

## Создание контрольных точек

Медиазапросы позволяют отправлять браузерам различные стили на основе ширины экранов этих браузеров. Например, браузеру можно сказать следующее: «Если твой экран шире 480 пикселей, применяй эти стили» или «Если твой экран шире 480 пикселей, но уже 769 пикселей, применяй эти стили». Различные значения ширины, указываемые вами, — 480, 769 и т. д. — в адаптивном дизайне часто называют контрольными точками (breakpoints). А с каких вообще значений нужно начинать разбивать дизайн на контрольные точки?

Проще всего это определить, если взять готовый дизайн для настольного компьютера и открыть страницу в веб-браузере. Захватите щелчком мыши точку регулирования размера окна и медленно уменьшайте ширину окна. В определенный момент страница приобретет совершенно неприглядный вид. Например, станет тесно ее четырем колонкам. Точка, в которой дизайн теряет приемлемый внешний вид, становится хорошим кандидатом на контрольную точку, то есть этот размер вполне подходит для определения нового медиазапроса и для загрузки новых стилей, чтобы удалить одну или две колонки.

Нередко создаются три набора медиазапросов для трех различных контрольных точек — один для смартфонов, другой для планшетных компьютеров, а третий для настольных мониторов. Конкретные значения контрольных точек зависят от конкретного дизайна (а также от конкретного устройства), но чаще всего отправной точкой служит экран, имеющий ширину меньше 480 пикселей, который получает один набор стилей, экран шириной между 481 и 768 пикселями получает второй набор стилей, а все, что по ширине больше 768 пикселей, получает дизайн, предназначенный для настольных компьютеров. Но все это в конечном счете остается на ваше усмотрение. Некоторые дизайнеры допускают расширение зоны планшетных компьютеров до 1024 пикселей, а стили для настольных компьютеров начинают отправлять тем браузерам, чья ширина окна превышает 1024 пиксела.

Некоторые дизайнеры даже доходят до определения четырех или пяти контрольных точек, чтобы их творения хорошо смотрелись на более широком диапазоне экранов. Подробности определения порядка создания этих контрольных точек с использованием медиазапросов будут рассмотрены в подразделе «Создание медиазапросов» данного раздела.

## Для чего сначала разрабатывать дизайн: для настольных или для мобильных систем?

Следует рассмотреть еще один вопрос: с расчетом на какое устройство нужно приступать к разработке дизайна? Вам не нужно создавать три отдельных набора стилей по одному для каждого из устройств, на которые вы нацелены. Вы можете и должны сначала создать исходный дизайн, то есть дизайн, работающий без медиазапросов. Затем можно создать стили медиазапросов для замены исходных стилей и переформатирования страницы под конкретную ширину экрана. Для этого существуют два основных подхода.

- **Предпочтение настольным системам.** Дизайн сайта можно разрабатывать с прицелом на настольные системы. Введите все требуемые колонки. Отработайте дизайн до совершенства, чтобы он хорошо выглядел на большом мониторе. Теперь это будет ваш исходный дизайн. Все стили можно убрать во внешнюю таблицу стилей и привязать эту таблицу к страницам вашего сайта обычным образом.

Затем добавьте медиазапросы для планшетных и телефонных устройств. Стили этих медиазапросов будут подстраивать дизайн для настольных систем под новые условия — удалять колонки, уменьшать шрифт заголовков и т. д. Преимущество такого подхода в том, что браузеры, не понимающие медиазапросов, получают основные стили для настольных систем. Большинство телефонов понимают медиазапросы, но устаревающие браузеры настольных систем, например Internet Explorer 8 и более ранние версии, их не понимают. Следовательно, подход, при котором предпочтение отдается настольным системам, означает, что Internet Explorer 8 получит нужные ему стили для создания полноценной презентации на настольных системах.

- **Предпочтение мобильным системам.** Можно сделать все наоборот и сначала разработать дизайн для мобильных систем. В таком случае в обычную внешнюю таблицу стилей помещаются основные стили, предназначенные для небольшого

экрана, а потом в медиазапросах дизайн дорабатывается для планшетных и настольных устройств путем добавления колонок и других корректировок для больших экранов.

Какой бы метод вы ни выбрали, нужно использовать обычную внешнюю таблицу стилей, привязанную к веб-странице обычным образом. В эту таблицу стилей включаются все стили, являющиеся общими применительно к различным устройствам. Например, для всех версий сайта требуются одинаковая цветовая палитра и одинаковые шрифты. Можно также использовать одинаковые стили для ссылок, изображений и других HTML-элементов. Иными словами, вам не нужно создавать для каждого устройства три абсолютно отдельных набора стилей, начните с одного набора, применяемого ко всем браузерам как телефонов, так и планшетных и настольных систем, а затем уточните дизайн для устройств, на которые нацелены медиазапросы.

## Создание медиазапросов

Запрос представляет собой вопрос, заданный веб-браузеру: «Равна ли ширина экрана 480 пикселям?» Если ответ положительный, браузер запускает таблицу стилей именно для устройства с данной шириной экрана (предоставляемая вами таблица стилей, рассмотренная ранее). Код, выполняющий эту задачу, очень похож на код для любой внешней таблицы стилей:

```
<link href="css/small.css" rel="stylesheet" media="(width: 480px)">
```

К этой стандартной ссылке на таблицу стилей добавился только еще один атрибут `media`, устанавливающий условия, при которых браузер использует указанную таблицу. В показанном выше примере браузер загружает внешнюю таблицу стилей `small.css`, когда кто-нибудь просматривает ваш сайт с помощью браузера, ширина окна которого составляет 480 пикселей. Скобки вокруг запроса — `(width: 480px)` — являются обязательным элементом. Если их не поставить, браузер проигнорирует запрос.

### ПРИМЕЧАНИЕ

---

Медиазапросы понятны большинству браузеров мобильных и настольных устройств, но непонятны Internet Explorer 8 и более ранним версиям. Устаревшие версии Internet Explorer можно заставить понимать ваши медиазапросы, если добавить к тегу `<head>` документа немного кода JavaScript. Нужно загрузить файл `respond.js` с адреса <http://tinycloud.com/7w49a6z>, поместить этот файл в ваш сайт, а затем дать на него ссылку на странице, воспользовавшись тегом `<script>`. Например:

```
<!--[if lte IE 8]>  
<script src="respond.min.js"></script>  
<![endif]-->
```

Этот небольшой маневр заставит Internet Explorer 8, 7 и 6 понимать медиазапросы.

---

480 пикселей — слишком точное значение. А что, если кто-нибудь пользуется телефонами с менее широким экраном, скажем, с шириной 300 пикселей? Лучше все же применять в медиазапросе диапазон значений. Например, может понадобиться задействовать конкретный стиль для экранов, ширина которых меньше или равна 480 пикселям. Это можно сделать с помощью следующего кода:

```
<link href="css/small.css" rel="stylesheet" media="(max-width:480px)">
```

Запись `(max-width:480px)` эквивалентна высказыванию «для экранов, имеющих ширину до 480 пикселей». Поэтому стили внутри файла `small.css` применяются, например, к экранам шириной 480, 320 и 200 пикселей.



Есть также и вариант `min-width`, определяющий, имеет ли браузер как минимум указанную ширину экрана. Этот вариант применяется при нацеливании на устройство, которое больше мобильного телефона или планшетного компьютера. Например, чтобы применить стили к экрану шире 768 пикселей, что превосходит ширину экрана многих планшетных компьютеров, нужно использовать следующий код:

```
<link href="css/large.css" rel="stylesheet" media="(min-width:769px)">
```

Чтобы воспользоваться этой таблицей стилей, окно браузера должно быть как минимум шириной 769 пикселей, что на один пиксел шире, чем 768 пикселей, составляющих ширину экрана некоторых планшетных устройств.

И наконец, можно установить как максимальное, так и минимальное значения ширины экрана целевых устройств, чтобы в них попадали браузеры устройств между телефонами и настольными компьютерами. Например, чтобы создать набор стилей для планшетного компьютера, имеющего экран шириной 768 пикселей, можно воспользоваться следующим кодом CSS:

```
<link href="css/medium.css" rel="stylesheet" media="(min-width:481px) and (max-width:768px)">
```

Иными словами, экран браузера должен быть шириной как минимум 481 и как максимум 768 пикселей. Этот файл `medium.css` не будет применяться к смартфонам с шириной экрана 320 пикселей, а также к браузерам настольных систем с шириной экрана 1024 пиксела.

---

#### ПРИМЕЧАНИЕ

Медиазапросы в CSS3 способны не только на простую проверку ширины окна браузера. Текущие стандарты медиазапросов утверждают, что можно проверять высоту и ориентацию (то есть в каком формате находится экран смартфона посетителя: в книжном или альбомном) или даже определять, какого типа экран устройства — цветной или монохромный. Есть еще ряд характеристик, проверяемых с помощью медиазапросов, но запросы с такими характеристиками поддерживаются не всеми браузерами. Получить дополнительные сведения о медиазапросах можно на веб-сайте W3C: [www.w3.org/TR/css3-mediaqueries](http://www.w3.org/TR/css3-mediaqueries).

---

## Включение запросов в таблицу стилей

Показанная выше технология является одним из способов использования медиазапросов с тегом `<link>` для загрузки различных таблиц стилей для экранов разной ширины. Но медиазапросы можно также добавлять и внутри единой таблицы стилей. Возможно, вам захочется сделать это, чтобы не пришлось, например, добавлять к HTML-файлу несколько тегов `<link>`, или вам захочется хранить все стили медиазапросов вместе в основной таблице стилей.

Есть два способа добавления медиазапросов к таблице стилей.

- **Использование директивы `@import`.** Эта технология уже рассматривалась в гл. 2. Директива `@import` позволяет загружать дополнительные внешние таблицы стилей либо во внутреннюю, либо во внешнюю таблицу стилей. Директиву `@import` можно также использовать с медиазапросом. Предположим, например, что нужно загрузить внешнюю таблицу стилей под названием `small.css`, содержащую стили для дисплеев шириной 320 или меньше пикселей. Для этого добавьте директиву `@import` непосредственно к таблице стилей:



```
@import url(css/small.css) (max-width:320px);
```

К сожалению, как уже ранее отмечалось, Internet Explorer 8 и более ранние версии понимают медиазапросы только при использовании программы JavaScript под названием `respond.js`. Эта программа не работает с медиазапросами, применяющими технологию `@import`, поэтому при использовании этой директивы и медиазапроса для загрузки таблицы стилей, предназначенной для браузеров настольных систем, Internet Explorer 8 и более ранние версии не загрузят эту таблицу стилей. Но эта проблема так остро не стоит, поскольку нет никакой необходимости задействовать медиазапросы для браузеров настольных систем.

#### ПРИМЕЧАНИЕ

Как уже упоминалось в гл. 2, директивы `@import` должны помещаться в начало таблицы стилей. Они не могут идти после каких-либо стилей. В результате могут возникать проблемы с каскадностью, при которых стили, определенные во внешней таблице стилей и загруженные с помощью директивы `@import`, будут отменяться более поздними стилями в таблице стилей. (Наиболее полно эта проблема рассмотрена в гл. 5.) Этой проблемы можно избежать при наличии всего лишь одной внешней таблицы стилей, которая содержит только директивы `@import`. Первая из них приведет к загрузке основной таблицы стилей, предназначенной для всех устройств, а вторая и третья приведут к загрузке таблиц стилей с использованием медиазапросов:

```
@import url(css/base.css); /* нет медиазапроса, применять ко всем */
@import url(css/medium.css) (min-width:481px) and (max-width:768);
@import url(css/small.css) (max-width: 480px);
```

○ **Встраивание медиазапроса в таблицу стилей.** Медиазапрос можно также встроить непосредственно в таблицу стилей:

```
@media (max-width: 480px) {
    body {
        /* сюда помещаются свойства стиля */
    }
    .style1 {
        /* сюда помещаются свойства стиля */
    }
}
```

Директива `@media` работает как своеобразный контейнер для всех стилей, соответствующих запросу. Следовательно, в данном примере стили `body` и `.style1` применяются только к тем устройствам, ширина экрана которых не превышает 480 пикселей. Использование встраиваемых директив `@media` позволяет собрать все ваши стили в одну таблицу стилей. Неплохо будет начинать внешние таблицы стилей со стилей, которые не содержатся в медиазапросах, отдавая сначала предпочтение либо стилям для настольных, либо стилям для мобильных устройств, а затем добавляя медиазапросы для всех остальных устройств.

## Основная структура таблицы стилей

Можно сказать, что существует множество различных способов использования медиазапросов, которые нацелены на различные устройства: с приоритетом, отдаваемым настольным системам, с приоритетом, отдаваемым мобильным системам, в отдельных таблицах стилей, в единой таблице стилей и т. д. По мере накопления опыта вы поймете, какой из способов больше подойдет для вашего проекта. Но сначала нужно всегда создавать единую внешнюю таблицу стилей, включая в нее стили для дисплеев настольных систем, а затем добавляя медиазапросы с изменениями этого дизайна основного уровня под планшетные устройства и телефоны. Схематически структура для такого файла должна иметь следующий вид:

```
/* Сюда помещаются стили, перезапускающие исходные настройки браузера */
/* Сюда помещаются стили для браузеров настольных устройств и основные стили
для всех устройств */
body {
    /* свойства для тела документа */
}

/* только для дисплеев средней ширины */
@media (min-width: 481px) and (max-width:768px) {
    body {
        /* свойства, применяемые только к браузерам планшетных устройств */
    }
}

/* только для дисплеев малой ширины */
@media (max-width:480px) {
    body {
        /* свойства, применяемые только к браузерам телефонов */
    }
}
}
```

Вы увидите эту структуру в действии в обучающем уроке данной главы, а начальную основную таблицу стилей можно найти в файле `desktop_first.css`, который находится в папке 14 с уроком этой главы ([www.sawmac.com/css3](http://www.sawmac.com/css3)). Следует помнить: даже притом, что в этот файл включены медиазапросы, он содержит обыкновенную таблицу стилей, которая привязывается к веб-странице обычным образом. Например, если сохранить этот файл в его текущем виде, нужно добавить к тегу `<head>` веб-страницы следующий код HTML:

```
<link href="styles.css" rel="stylesheet">
```

## Предпочтение мобильным системам

Если вы решили отдать предпочтение мобильным системам, то сначала нужно создать набор стилей, нацеленный на браузеры мобильных систем, а затем добавить медиазапросы для изменения дизайна под браузеры планшетных и настольных устройств. Основная структура таблицы стилей при таком подходе будет иметь следующий вид:

```
/* Сюда помещаются стили, перезапускающие исходные настройки браузера */
/* Сюда помещаются стили для браузеров мобильных устройств и основные стили
для всех устройств */
body {
  /* свойства для тела документа */
}

/* только для дисплеев средней ширины */
@media (min-width: 481px) and (max-width:768px) {
  body {
    /* свойства, применяемые только к браузерам планшетных устройств */
  }
}

/* только для дисплеев большой ширины */
@media (min-width:769px) {
  body {
    /* свойства, применяемые только к браузерам настольных устройств */
  }
}
```

Если следовать этому решению, нужно помнить, что Internet Explorer 8 и более ранние версии не понимают медиазапросов и требуют использования программы JavaScript для применения стилей, предназначенных для больших дисплеев. В противном случае для сайта будут применяться только стили, предназначенные для браузеров мобильных устройств (см. второе примечание в подразделе «Создание медиазапросов» данного раздела).

## Гибкие сетки

Не стоит поддаваться соблазну разработки раскладок с фиксированной шириной 320 пикселей, типичной для iPhone и некоторых телефонов с операционной системой Android, с шириной 720 пикселей для компьютеров типа iPad в книжном (высоком) режиме работы и с шириной 1000 пикселей для мониторов настольных систем. Хотя устройства типа iPhone приобрели широкую популярность, на них свет клином не сошелся. Есть еще и телефоны с операционной системой Android, имеющие всевозможные формы и размеры, а стало быть, и ширину. Не сомневаюсь, что вам попадались и многие другие устройства с необычными размерами, да и планшетные устройства имеют весьма разнообразную ширину экрана. То есть универсальной ширины для смартфонов и планшетных устройств просто нет, поэтому лучше всего создавать страницы с гибкой шириной.

Основным компонентом адаптивного веб-дизайна являются гибкие сетки. Они — не что иное, как свободная разметка с непостоянной шириной, рассматривавшаяся в гл. 12, и при которой общая ширина страницы изменяется, чтобы страница могла поместиться на экранах разной ширины. В большинстве случаев это означает, что для ширины устанавливается значение 100%. Но для дисплеев настольных систем, возможно, потребуется использовать свойство `max-width`, чтобы страница не становилась абсурдно широкой на больших мониторах настольных систем. Кроме того, процентная

основа должна быть в дизайне и у отдельных колонок, заменяя собой единицы измерения в фиксированных пикселах или em. Отдельные колонки также должны становиться шире или уже, чтобы вписываться в изменяющуюся ширину страницы.

#### СОВЕТ

---

Как уже упоминалось во врезке «Информация для опытных пользователей. Определение порядка в grid-системе» в подразделе «Плавающие элементы внутри плавающих элементов» раздела «Использование плавающих элементов в разметках» гл. 13, в свободном доступе есть множество заранее собранных в пакеты grid-систем, представляющих собой HTML- и CSS-код, облегчающий создание многоколоночных разметок. А с созданием свободной разметки вы уже познакомились в предыдущей главе.

---

Предположим, например, что нужно создать дизайн на основе двух колонок, где первая колонка имеет ширину, равную одной трети ширины страницы, а вторая — равную двум третям ширины страницы. Можно начать с создания простого кода HTML:

```
<div class="columns">
  <div class="one-third">
    ... сюда помещается содержимое...
  </div>
  <div class="two-thirds">
    ... сюда помещается содержимое...
  </div>
</div>
```

Затем можно создать несколько CSS-стилей для создания свободной разметки:

```
.columns {
  width: auto; /* то же самое что и 100 % */
  max-width: 1200px;
}

.columns:after {
  content: "";
  display: table;
  clear: both;
}

.one-third {
  float: left;
  width: 33%;
}

.two-thirds {
  float: left;
  width: 67%;
}
```

Первый стиль — `.columns` — устанавливает ширину div-контейнера, содержащего колонки. Задание для свойства `width` значения `auto` — это то же самое, что и установка для него значения `100%`, поскольку свойство `max-width` будет сдерживать блок от приобретения слишком большой ширины. Второй стиль — `.columns:after` — помогает

содержать две плавающие колонки (подробности того, как это работает, можно найти в разделе «Решение проблем плавающих элементов» гл. 13). И наконец, последние два стиля просто устанавливают ширину двух `div`-контейнеров в 33 % (в ширину одной трети его контейнера) и в 67 % (две трети) и перемещают их влево, чтобы они появлялись рядом.

## Важность порядка следования исходного HTML-кода

Когда вы держите телефон вертикально, то при условии сохранения читаемости страницы там просто не хватает места не только для трех, а даже для двух колонок в строке. Поэтому многие разработчики для отображения страницы в мобильном телефоне просто выкладывают ее в одной большой колонке. Для этого нужно удалить из создаваемых колонок все плавающие элементы. Например, если создается дизайн, состоящий из трех колонок, для дисплея настольной системы и в нем используются плавающие элементы для позиционирования колонок рядом друг с другом, нужно просто установить для свойства `float` таких элементов значение `none`. Тогда они будут отображать HTML в обычном порядке — один тег блочного уровня над другим. Это придает порядку следования исходного HTML-кода особую важность. Например, у страницы могут быть две боковые панели, первая — с перечнем ссылок на родственные сайты, а вторая — с рекламой товаров, продаваемых вашей компанией. При этом основная информация, за которой, собственно, и приходят на вашу страницу, содержится в средней колонке. Один из способов раскладки всего этого по колонкам заключается в смещении первой боковой панели в качестве плавающего элемента влево, а второй боковой панели — вправо, позволяя основной колонке просто обтекать две остальные колонки и находиться в центре.

В понятиях HTML это будет означать, что сначала появляются два `div`-контейнера боковых панелей, а затем следует тег, содержащий основную информацию. Если приспособить эту страницу под мобильные устройства, удалив указание на то, что боковые панели являются плавающими элементами, то получится, что две боковые панели появятся перед основным содержимым. Вашей аудитории придется прокручивать информацию вниз по странице, чтобы пройти рекламу и ссылки и добраться до нужного ей содержимого.

С точки зрения пользователей было бы лучше поместить контейнер с основным содержимым до боковых панелей. Как уже упоминалось в разделе «Использование плавающих элементов в разметках» гл. 13, этот метод может потребовать добавления дополнительных контейнеров и превращения всех элементов в плавающие, включая и `div`-контейнер с основным содержимым.

Короче говоря, нужно соблюдать соответствующий порядок следования HTML-кода, прежде чем создавать раскладку для многоколоночного дизайна, предназначенного для настольных систем. Проще всего понять, что происходит, — это просмотреть страницу вообще без применения к ней CSS. Тогда вы увидите все `div`-контейнеры и другие блочные элементы поставленными друг на друга и сможете понять, как будет выглядеть страница в виде одной колонки на телефоне.

---

### ПРИМЕЧАНИЕ

Как можно создать дизайн, состоящий из трех колонок и предназначенный для настольных систем при условии сохранения наиболее важной информации в верхней части страницы, было показано в обучающем уроке предыдущей главы.

---

## Сброс блочной модели

Как уже объяснялось в подразделе «Предотвращение выпадений плавающих элементов» раздела «Решение проблем плавающих элементов» гл. 13, при использовании для указания ширины процентного отношения возникает угроза выпадения плавающих элементов, когда общая ширина колонок, размещаемых в одном ряду, превышает 100 %, что заставляет последнюю колонку опускаться под другие колонки. Из-за способа, который применяется браузерами для вычисления ширины элементов, добавление толщины границы, окружающей `div`-контейнер, или его внутренних отступов приводит к тому, что ширина `div`-контейнера на экране становится больше той ширины, которая указана в CSS. Например, если для одной колонки задана ширина 33 %, а для другой — 67 % (как в примере, показанном выше), они должны вполне уместиться рядом друг с другом, поскольку их общая ширина составляет 100 %. Но если к колонке добавлена граница толщиной 1 пиксел, то ее общая ширина станет  $100\% + 2$  пиксела (учитываются левая и правая границы). Теперь эта колонка будет слишком большой, чтобы поместиться в окне, и вторая колонка выпадет под первую.

Есть несколько способов преодоления этого затруднительного положения, и все они были рассмотрены в подразделе «Предотвращение выпадений плавающих элементов» раздела «Решение проблем плавающих элементов» гл. 13. Но наиболее очевидным решением станет предписание веб-браузеру включать в общую ширину и высоту элемента границы и отступы в качестве части его вычислений в рамках блочной модели. То есть браузер можно заставить рассматривать границы и отступы как составную часть CSS-свойства `width`, чтобы добавление дополнительных отступов или границ не приводило к увеличению ширины (или высоты) элемента. Поскольку такой подход желательнее применить ко всем элементам, лучше воспользоваться универсальным селектором, позволяющим сбросить исходные параметры блочной модели для всех элементов, которые имеются на странице:

```
* {  
  -moz-box-sizing: border-box;  
  box-sizing: border-box;  
}
```

В Firefox для этого свойства все еще применяется префикс производителя, поэтому здесь нужна версия `-moz-box-sizing`. А вот все остальные браузеры, включая Internet Explorer 8 и более поздние версии, понимают версию без такого префикса. Лучше всего поместить этот стиль в CSS-код, используемый для сброса исходных установок страницы.

---

### ПРИМЕЧАНИЕ

Internet Explorer 7 не понимает свойство `box-sizing`, поэтому данный прием в нем работать не будет. Internet Explorer 7 быстро выходит из употребления, но если вам все же нужно поддерживать этот браузер, существует несколько способов предотвращения выпадения плавающих элементов при использовании разметки на основе процентных соотношений (см. подраздел «Предотвращение выпадений плавающих элементов» раздела «Решение проблем плавающих элементов» гл. 13).

---

## Преобразование фиксированной ширины в гибкие сетки

При разработке совершенно нового дизайна представление процентных отношений не должно даваться слишком трудно. В конце концов, если нужны четыре колонки одинаковой ширины, для каждой из них устанавливается значение 25 %:

```
width: 25%;
```

Но если приходится иметь дело с дизайном веб-сайта с фиксированной шириной и нужно преобразовать его в свободный дизайн, ситуация усложняется. Для начала представим, что вы разработали страницу с фиксированной шириной 960 пикселей. Эта страница либо заключена в тег `<div>`, для которого задана ширина 960 пикселей, либо такая ширина просто установлена для тега `<body>`. В любом случае теперь нужно, чтобы этот контейнер был полностью свободным. Для этого достаточно просто изменить

```
width: 960px;
```

на

```
width: auto;
```

Установка ширины элемента на `auto` — практически то же самое, что и задание для ширины значения `100%`: ширина этого элемента сравняется с шириной его контейнера.

Затем нужно преобразовать параметры ширины колонок со значения в пикселях в значение в процентах. Чтобы облегчить вычисление, главный специалист по адаптивному веб-дизайну Итан Маркотт придумал замечательную формулу:  $\text{цель} / \text{контекст} = \text{результат}$ . В переводе на простой язык это означает: «возьмите ширину элемента, подвергаемого преобразованию (в пикселях), и разделите ее на ширину контейнера этого элемента (в пикселях)». В результате получится дробное значение, которое нужно перевести в проценты.

Рассмотрим пример. Предположим, что внутри этой страницы шириной 960 пикселей есть две колонки: боковая панель шириной 180 пикселей и основная колонка шириной 780 пикселей. В CSS-файле имеется следующий код:

```
.sidebar {
  float: left;
  width: 180px;
}

.main {
  float: left;
  width: 780px;
}
```

Конечно, в нем присутствует множество другого кода CSS, задающего границы, фоновые цвета и т. д., но в данном случае нас интересуют только параметры ширины. Приступая к преобразованию боковой панели, нужно взять ее ширину, равную 180 пикселям, и разделить ее на ширину контейнера этой боковой панели, равную 960 пикселям. Получившийся результат 0,1875 нужно умножить на 100 и получить процентное значение: 18,75 %. Точно так же для основной колонки 780 делится на 960, и получается 0,8125. При умножении этого результата на 100 получаем 81,25 %. То есть изменения параметров ширины этих стилей придадут им следующий вид:

```
.sidebar {
  float: left;
  width: 18.75%;
}
```

```
.main {  
  float: left;  
  width: 81.25%;  
}
```

Эти новые значения не нужно округлять. То есть не превращайте 18,75 % в 19 %, поскольку это, скорее всего, приведет к тому, что колонки больше не будут стоять рядом и мы столкнемся с выпадением плавающего элемента. Браузеры хорошо справляются с десятичными значениями, и вы спокойно можете использовать любые значения, возвращаемые калькулятором. Вполне допустима, скажем, ширина, заданная значением 25.48488%.

То же самое применимо и к колонкам, находящимся внутри других колонок. Предположим, например, что в показанной выше основной колонке имеется один раздел, состоящий из двух плавающих div-контейнеров и создающий две дополнительные колонки внутри основной колонки. Оба div-контейнера имеют одинаковую ширину, равную 390 пикселям, поэтому для вычисления их ширины в процентном отношении берется их текущая ширина в пикселях (390) и делится на ширину их контейнера, которая в данном примере относится к ширине основной колонки и равна 780 пикселям. Результат получается 0,5. Умножение на 100 дает результат 50 %. (Но, по сути, эти вычисления не нужны. Ведь у вас имеются две стоящие рядом колонки одинаковых размеров, и вы знаете, что каждая из них занимает половину доступного пространства, или 50 %.)

После переделки дизайна и вычисления процентных значений нужно помнить, что общая ширина всех колонок в одном ряду не должна превышать 100 %.

#### СОВЕТ

Ту же самую формулу можно применить для преобразования размеров в пикселях в размеры в em. Предположим, что ширина текстового абзаца составляет 18 пикселей (цель). Исходный размер обычного текста (контекст) составляет 16 пикселей. При делении 18 на 16 получаем новый размер в em: 1.125em.

### ИНФОРМАЦИЯ ДЛЯ НОВИЧКОВ

#### Тестирование адаптивного дизайна

Поскольку адаптивный дизайн предназначен для ответа экранам различных устройств просмотра, нужно найти способ просмотра страниц на экранах с различной шириной. Проще всего протестировать медиазапросы путем просмотра страницы в браузере настольного компьютера при изменении экрана браузера. Перетащите границу окна, сделав его тоньше, и посмотрите, что получится на каждой контрольной точке или на каждом установленном вами медиазапросе. Этот прием работает неплохо, но не все браузеры позволяют уменьшать масштаб окна до ширины 320 пикселей, характерной для некоторых телефонов.

Есть также несколько веб-средств, позволяющих просматривать страницы в окнах разных размеров. Сайт [responsivepx](http://responsivepx.com) (<http://responsivepx.com>) дает возможность вводить URL-адрес страницы в Интернете, затем указывать различные параметры ширины и высоты экрана. Этот сайт открывает страницу в `<iframe>`-теге установленной ширины. Браузеры применяют любые медиазапросы, имеющие отношение к этой ширине экрана, поэтому вам предоставляется возможность просматривать результат выполнения ваших медиазапросов.

Средство [Responsinator](http://www.responsinator.com) ([www.responsinator.com](http://www.responsinator.com)) поставляется с некоторыми предустановленными



## ИНФОРМАЦИЯ ДЛЯ НОВИЧКОВ

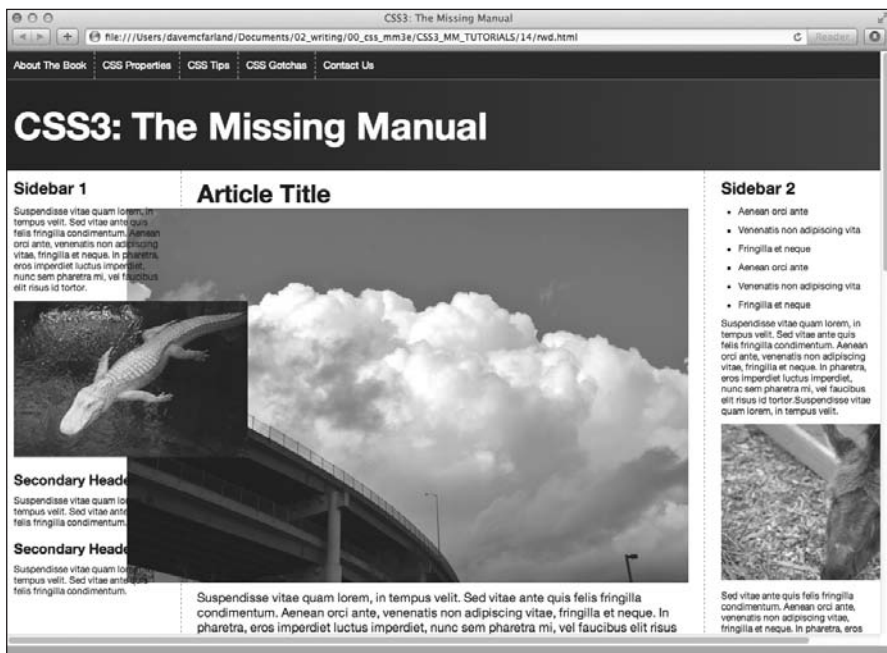
параметрами ширины окна, соответствующими таким популярным устройствам, как iPhone, Samsung Galaxy, iPad, Kindle и т. д. Нужно просто ввести URL-адрес вашего сайта, и его страница появится внутри имитируемых экранов. Responsinator работает только с файлами, выложенными на веб-сервер в Интернете.

Если у вас есть смартфон или планшетный компьютер, страницу можно просмотреть непосредственно из вашего компьютера на этих устройствах, используя разработанное компанией Adobe средство Edge

Inspect (<http://html.adobe.com/edge/inspect/>). Это довольно сложное средство предоставляет замечательную возможность просматривать ваш рабочий дизайн в процессе его разработки. Можно также просто выложить дизайн на веб-сервер, активизировать телефон и посетить эту страницу, чтобы посмотреть на ее внешний вид. Разумеется, пока вы не забудете десяток смартфонов, вы не узнаете, какое впечатление создается от вашей веб-страницы конкретно у тех людей, которые пользуются мобильными браузерами.

## Изменчивые изображения

Гибкая разметка позволяет создавать дизайн, работающий в широком спектре окон браузеров различной ширины, однако при вставке на страницы изображений у вас возникнет проблема. Хотя колонки в гибком дизайне по мере уменьшения окна сжимаются, изображений обычно это не касается. Это может привести к выходу графики за предназначенные для нее границы и к тому, что она уже не станет вписываться в ширину колонки (рис. 14.3).



**Рис. 14.3.** Когда колонка становится уже, чем находящееся внутри нее изображение, это изображение выходит за границы колонки, перекрывая обычно другие колонки и содержимое страницы

К счастью, существует способ придания гибкости и изображениям. Он требует выполнения двух шагов: создания нового CSS-стиля и внесения ряда изменений в ваш HTML-код.

1. Сначала нужно добавить в таблицу стилей следующий стиль:

```
img { max-width: 100%; }
```

Он установит для максимального размера любого изображения значение 100 % от ширины контейнера этого изображения. То есть изображение не сможет стать больше колонки, div-контейнера или любого HTML-элемента, внутри которого оно находится.

Но этого еще недостаточно, чтобы сделать изображение гибким. Обычно при вставке тега `<img>` для изображения добавляются параметры его высоты и ширины. Именно эти размеры используются браузером при выводе изображения. Когда указано значение для свойства `max-width`, изображение не станет шире колонки, но его высота по-прежнему будет точно соответствовать значению, установленному в коде HTML. То есть изображение подстроится под ширину колонки, а его высота не изменится, что приведет к искажению этого изображения. Решение вполне очевидно: нужно просто убрать из кода HTML атрибуты `width` и `height`.

2. Найдите на странице все теги `<img>` и удалите из них атрибуты `height` и `width`. То есть превратите этот код HTML:

```

```

в этот:

```

```

У многих текстовых редакторов есть режим поиска и замены, который может ускорить поиск и удаление этих атрибутов.

Разумеется, этот подход предполагает, что все ваши изображения будут заполнять колонку, внутри которой они находятся. Во многих случаях вам потребуется, чтобы изображения были меньше этого размера, возьмем, например, фотографию, которая уплывает к правому краю основной колонки, а текст ее обтекает. Для работы с изображениями, размер которых задается по-разному, можно создать разные классы с различными установками значения для свойства `max-width` и применить эти классы к конкретным тегам `<img>` в коде HTML.

Предположим, например, что нужно, чтобы изображение уплыло к левому краю колонки и его размер составлял 40 % от ширины колонки. Сначала создадим стиль класса:

```
.imgSmallLeft {  
  float: left;  
  max-width: 40%;  
}
```

Затем применим этот класс к тегу `<img>`:

```

```

Более гибким подходом будет разбить размер и задание плавающего элемента по разным классам:

```
.imgSmall {  
    max-width: 40%;  
}  
  
.imgLeft {  
    float: left;  
}
```

А затем применить к изображению оба класса:

```

```

Путем использования двух классов можно применить класс `imgSmall` к любым изображениям, даже к тем, которые вы решили прижать к правому краю или вообще не делать плавающими.

#### ПРИМЕЧАНИЕ

---

Изменить размер фоновых изображений можно с помощью свойства `background-size`. Но это свойство не работает в Internet Explorer 8 и более ранних версиях.

---

## Недостатки изменчивых изображений

У изменчивых изображений есть две проблемы. Во-первых, поскольку изображение будет укрупняться, чтобы поместиться в 100 % ширины колонки (или в то процентное отношение, которое будет задано), возможно, это изображение будет растянуто свыше его исходной ширины. Предположим, например, что вы вставили в боковую панель изображение шириной 200 пикселей. Когда страница, на которой находится это изображение, отображается на очень широком мониторе, браузер может сделать ширину этой боковой панели равной 300 пикселям. Изображение будет соответствовать по ширине этой колонке и также станет растянутым до 300 пикселей. Поскольку эта ширина превышает ширину файла изображения, это изображение появится в искаженном и пикселизованном виде. Иными словами, нужно убедиться в том, что сам файл изображения имеет размер по ширине, позволяющий этому изображению поместиться в самом широком варианте масштабирования колонки.

Во-вторых, при просмотре страницы на телефоне ширина колонки и ширина изображения могут ужаться до ширины, существенно меньшей, чем на браузере настольной системы. Это означает, что изображения на экране будут сильно уменьшены. Хотя качество изображения от этого не пострадает, вы заставите пользователей мобильных телефонов загружать файлы изображений, имеющие размер намного больше необходимого. Тем самым будет неоправданный расход трафика на большие файлы изображений, в которых нет никакой необходимости. К сожалению, пока эта проблема не имеет полноценного решения, и фактически сторонники адаптивного веб-дизайна считают ее одной из наиболее существенных проблем RWD.

К числу других людей, работающих над проблемой изменчивых изображений, относится группа сообщества W3C, которая сформирована для попытки решения этой проблемы средствами HTML. Об их успехах можно прочитать на сайтах <http://picture.responsiveimages.org> и <http://www.w3.org/community/respimg/>. А пока можно проигнорировать все волнения насчет больших файлов изображений для мобильных систем и ждать, пока не появится решение этой проблемы. Вполне возможно, что одно из таких решений, пока вы читаете эту книгу, уже появилось. Или же можно воспользоваться весьма удачным, но непростым решением, которое называется

«адаптивные изображения». Для отправки изображений подходящего к каждому устройству размера в нем используются JavaScript и PHP. То есть браузеры небольших мобильных устройств для своих небольших экранов получают изображения, меньшие по объему, а браузеры настольных систем загружают более крупные изображения. Шире этот вопрос рассмотрен на сайте <http://adaptive-images.com>.

## Видео и Flash-графика

При использовании HTML5-тега `video` или встроенного Flash-содержимого для масштабирования таких элементов вместе с их контейнерами можно также воспользоваться приемом со свойством `max-width`. Добавьте к своей таблице стилей следующий стиль:

```
img, video, embed, object {
  max-width: 100%;
}
```

К сожалению, этот стиль не справляется с видео, которое вставлено с помощью `iframe`-тегов (а это наиболее часто используемый способ добавления на страницу видео с YouTube или Vimeo). Для встроенных видео с YouTube воспользуйтесь JavaScript-программой Дэйва Руперта и Криса Койера FitVids: <http://fitvidsjs.com>. Этот небольшой по объему код JavaScript гарантирует для большинства встроенных видео масштабирование в обе стороны для того, чтобы видео поместилось в свой контейнер.

### ИНФОРМАЦИЯ ДЛЯ НОВИЧКОВ

#### Когда пиксел не является таковым?

Многим пиксел представляется в виде отдельной точки на экране или на мониторе. И это так и есть. Но благодаря новым дисплеям с высокой плотностью пикселей, таким как разработанные компанией Apple Retina-дисплеи, теперь о пикселах нужно иметь двойное представление. Устройства типа iPhone 5 имеют разрешение 1136×640 пикселей. В iPhone это довольно большое количество пикселей втиснуто в область, меньшую по размеру, чем у большинства других экранов. В каждый дюйм экрана втиснуто 326 пикселей. У мониторов настольных систем обычно приходится около 100 пикселей на дюйм. То есть у экрана iPhone в три раза больше пикселей на дюйм, чем у мониторов многих настольных систем. В результате получаются очень четкие и резкие изображения.

Но из-за этого у веб-дизайнеров возникает проблема. Если для текста задан размер 16 пикселей, позволяющий ему хорошо смотреться на мониторе настольной системы (его высота получается около 0,16 дюйма), то он будет нечитаем в телефоне с экраном высокой плотности, поскольку его высота окажется 0,04 дюйма.

К счастью, браузеры с дисплеями с высокой плотностью пикселей так поступать не будут. Они делают так, чтобы отдельный пиксел CSS занимал на экране такого дисплея несколько пикселей и высота текста размером 16 пикселей на самом деле отображалась с использованием более 16 пикселей. Телефоны и другие устройства с дисплеями, имеющими высокую плотность пикселей, делают различие между пикселом устройства, то есть точкой на экране, и пикселом CSS.

Пиксел CSS вычисляется на основе пиксельной плотности экрана и его расстояния от того, кто смотрит на экран. Поскольку телефон держат ближе к лицу, чем монитор, элементы на его экране крупнее элементов с таким же размером на мониторе с диагональю 28 дюймов.

На iPhone один пиксел CSS фактически представлен четырьмя пикселями устройства. Поэтому текст высотой 16 пикселей на устройстве фактически занимает высоту 32 пикселя. У разных устройств, например у телефонов Android, разная плотность пикселей, и поэтому для определения количества пикселей экрана, приходящихся на один пиксел CSS, используются разные вычисления.

## Обучающий урок по адаптивному веб-дизайну

В этом уроке мы возьмем разметку, созданную в предыдущей главе (с добавлением нескольких изображений), и придадим ей свойства адаптивного веб-дизайна. Будет продемонстрирован подход, при котором предпочтение отдано разметке для настольных устройств, то есть исходные CSS-стили лучше всего будут работать в браузерах настольных систем, а затем к ним будут добавлены медиазапросы, приспособливающие внешний вид под экраны средней величины (подобные экранам планшетных устройств) и под небольшие экраны (подобные экранам телефонов).

Для начала нужно загрузить файлы урока с веб-сайта [www.sawmac.com/css3/](http://www.sawmac.com/css3/). Щелкните на ссылке на урок и загрузите файлы. Все файлы находятся в ZIP-архиве, поэтому сначала их нужно будет разархивировать. Файлы для этого урока находятся в папке 14.

### Изменение порядка следования исходного кода HTML

При создании раскладки под небольшой экран вы превращаете свой дизайн, состоящий из трех колонок, в дизайн с одной колонкой, где все блоки с содержимым ставятся друг на друга. Проблема кода HTML той страницы, которая была создана в предыдущем обучающем уроке, заключается в том, что сначала там следует левая боковая панель, поэтому при переводе дизайна в одноколоночный эта боковая панель появится перед основным содержимым.

Лучше пусть посетители сайта получают сначала основное содержимое, а затем, прокручивая страницу, увидят дополнительную информацию, находящуюся на бывших боковых панелях. Чтобы получить такой результат, нужно добавить немного кода HTML и переместить часть уже существующего кода. Для этого будет использоваться технология, рассмотренная в подразделе «Предотвращение выпадений плавающих элементов» раздела «Решение проблем плавающих элементов» гл. 13 (см. рис. 13.11): установка контейнера с основным содержимым перед первой боковой панелью с последующим заключением основного содержимого с боковой панели в новый `<div>`-тег. Этот новый `div`-контейнер нужно сделать плавающим элементом, прижимающимся к левому краю его контейнера, а основное содержимое прижать к правому краю его контейнера, прижав боковую панель к левому краю этого же контейнера. Такая расстановка позволит поддерживать единую визуальную раскладку для дисплеев настольных систем — сначала будет следовать слева боковая панель, в середине будет находиться основное содержимое, а справа — вторая боковая панель, но затем для телефонного дизайна основное содержимое будет следовать выше двух бывших боковых панелей.

1. Откройте файл `rwd.html`, который находится в папке 14.

Эта страница точно соответствует завершенной странице из урока предыдущей главы. Сначала нужно переместить основное содержимое вверх.

2. Найдите HTML-комментарий `<!-- main content goes here -->` (сюда помещается основное содержимое) и выберите его и весь следующий ниже код HTML, включая закрывающий тег `</article>`. То есть выберите этот комментарий и все, что ниже его, до комментария `<!-- second sidebar goes here -->` (сюда помещается вторая боковая панель).

3. Вырежьте текст и поместите его в буфер обмена с помощью команды меню **Edit ▶ Cut** (Редактировать ▶ Вырезать) (или воспользуйтесь аналогичным методом, предоставляемым вашим текстовым редактором).

Затем этот код будет вставлен перед кодом первой боковой панели.

4. Найдите ближе к началу файла тег `<div class="contentWrapper">`.

Добавьте ниже его, но перед комментарием `<!-- first sidebar goes here -->` (сюда помещается первая боковая панель) пустую строку и воспользуйтесь командой меню **Edit ▶ Paste** (Редактировать ▶ Вставить), чтобы вставить основное содержимое перед кодом боковой панели.

Теперь нужно будет добавить `<div>`-тег, чтобы заключить в него основное содержимое и первую боковую панель.

5. После тега `<div class="contentWrapper">` добавьте тег `<div class="columnWrapper">`.

Код HTML должен приобрести следующий вид:

```
<div class="contentWrapper">
<div class="columnWrapper">
<!-- main content goes here -->
```

Затем нужно закрыть этот `div`-контейнер.

6. Найдите закрывающий тег `</aside>`, принадлежащий коду первой боковой панели. Он находится непосредственно перед комментарием `<!-- second sidebar goes here -->` (сюда помещается вторая боковая панель). Добавьте тег `</div>` после тега `</aside>`, чтобы этот HTML приобрел следующий вид:

```
</aside>
</div>
<!-- second sidebar goes here -->
```

Пока это будут все изменения, вносимые в код HTML. Если вы запутались или хотите проверить свою работу, файл, содержащий все эти изменения кода HTML, под названием `new-source-order.html` можно найти в папке 14.

Если просмотреть эту страницу в веб-браузере, можно увидеть разметку в три колонки, но с основным содержимым слева и первой боковой панелью в центре (рис. 14.4). Эту проблему можно решить с помощью дополнительного кода CSS.

7. Откройте файл `styles.css`, который находится в папке 14.

В этом файле содержится код CSS, созданный при изучении предыдущей главы. Сначала нужно добавить новый стиль для контейнера колонок, сделать его плавающим и прижать к левому краю, чтобы он располагался рядом с правой боковой панелью.

8. Добавьте ближе к концу файла перед стилем `.sidebar` следующий стиль:

```
.columnWrapper {
float: left;
width: 80%;
}
```

Установленная здесь ширина составлена из ширины левой боковой панели (20 %) и основной колонки (60 %). На самом деле здесь создается раскладка, состоящая из двух колонок: первую колонку представляет вновь созданный тег

<div>, а вторую — правая боковая панель. Контейнер основного содержимого и левая боковая панель, по сути, являются двумя колонками внутри div-контейнера, в который заключена эта колонка, то есть колонки внутри колонки, как уже объяснялось в подразделе «Предотвращение выпадений плавающих элементов» раздела «Решение проблем плавающих элементов» гл. 13 и показывалось на рис. 13.11.

Далее нужно будет настроить плавающие элементы и параметры ширины основного содержимого и первой боковой панели.



**Рис. 14.4.** Страница по-прежнему имеет дизайн, состоящий из трех колонок, но теперь основное содержимое находится в левой части окна, а левая боковая панель располагается в центре

9. Измените значение ширины в стиле `.sidebar` на 25%. Стиль должен приобрести следующий вид:

```
.sidebar1 {
  float: left;
  width: 25%;
  padding: 0 20px 0 10px;
}
```

Изначально эта боковая панель занимала 20 % всей ширины страницы, но теперь, когда она находится внутри div-контейнера, заключающего в себе колонки, нужно подогнать ширину колонок боковой панели и основного содержимого, чтобы они поместились в 80 % от ширины страницы, выделенных контейнеру колонок. То есть указанное процентное отношение имеет значение не для всей страницы, а для контейнера колонок, который занимает всего 80 % ширины страницы.

Чтобы вычислить новое процентное отношение, нужно взять старое процентное отношение — 20 % — и разделить его на ширину контейнера — 80 %, а затем умножить результат на 100. Результат деления 20 на 80 равен 0,25. Умножение этого результата на 100 дает значение 25 %. Та же технология должна быть применена и к изменению ширины основного содержимого.

10. Найдите стиль `.main` и измените настройку `float` на `right`, а `width` на 75%, придав стилю следующий вид:



```
.main {
  float: right;
  width: 75%;
  padding: 0 20px;
  border-left: dashed 1px rgb(153,153,153);
  border-right: dashed 1px rgb(153,153,153);
}
```

Этот элемент прижимается к правому краю, поэтому он появляется справа от первой боковой панели, в результате чего основное содержимое снова помещается в центре. Ширина вычисляется путем деления старого показателя ширины элемента на показатель ширины его контейнера:  $60 / 80 = 0,75$ , или 75 %. Если сохранить файлы и просмотреть файл `rwd.html` в веб-браузере, он должен выглядеть так же, как показано на рис. 14.5.

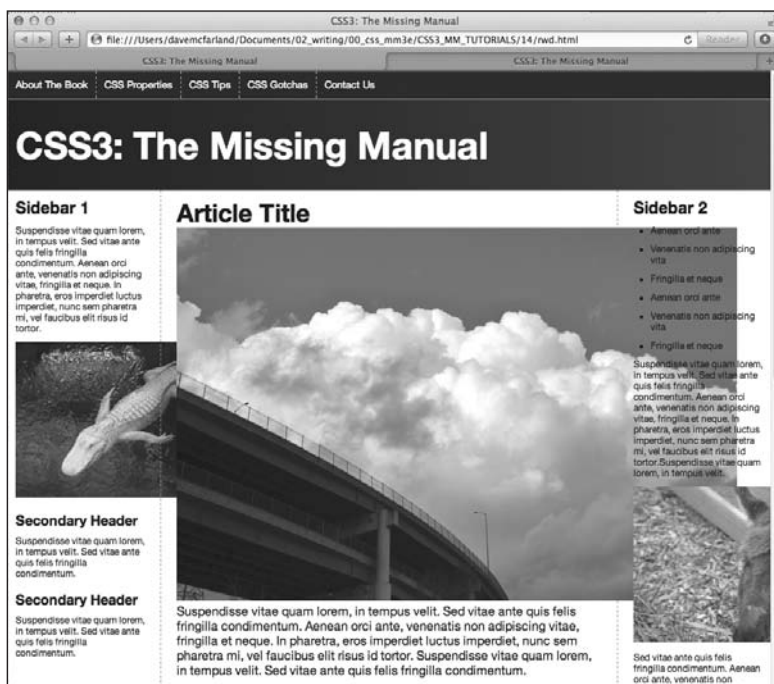


Рис. 14.5. Возвращение к исходному состоянию: к дизайну из трех колонок

Таким образом, мы вернулись к дизайну из трех колонок. Но теперь порядок следования исходного кода HTML-страницы больше подходит для создания телефонной раскладки, имеющей одну колонку. Однако изображения не помещаются в отведенное для них пространство и, как здесь показано, «повисают» за пределами колонок. Далее мы займемся устранением этого недостатка.

#### ПРИМЕЧАНИЕ

Конкретно этот дизайн не станет шире 1200 пикселей благодаря свойству `max-width`, для которого установлено значение в групповом селекторе `nav ul`, `header h1`, `footer p`, `.contentWrapper`. Если нужно, чтобы этот дизайн заполнил окно браузера независимо от его ширины, просто удалите объявления `max-width 1200px`; из этого группового селектора.



## Изменчивые изображения

Вы успешно справились с созданием свободной разметки. Если теперь просмотреть страницу и изменить размеры окна, можно увидеть, что по мере сужения окна колонки становятся меньше. К сожалению, некоторые изображения шире колонок и выбиваются из общего дизайна. Как уже ранее упоминалось, составной частью адаптивного веб-дизайна является еще и придание изображениям возможности адаптации к изменениям ширины окна браузера. Для реализации такой возможности нужно немного дополнить код CSS и убрать часть кода HTML.

1. Откройте в текстовом редакторе файл `styles.css`. Внизу таблицы стилей добавьте стиль для тега `img`:

```
img {
  max-width: 100%;
}
```

Этот стиль задает для изображения максимальную ширину, равную 100 % от ширины контейнера. Следовательно, для колонки, имеющей на экране ширину 200 пикселей, изображение будет шириной 200 пикселей. Если посетитель изменит размеры окна браузера и колонки станут меньше, изображение также уменьшится, чтобы поместиться в колонке.

Но все это заработает при условии удаления свойств изображения `width` и `height` из кода HTML.

2. Удалите для каждого из четырех тегов `<img>`, имеющихся в файле `rwd.html`, атрибуты `height` и `width`.

У нас есть четыре изображения: `clouds.jpg`, `jellyfishy.jpg`, `gator.jpg` и `mule.jpg`. Удалите из их тегов атрибуты `width` и `height`, например код

```

```

должен превратиться в такой:

```

```

Если сохранить файлы CSS и HTML и просмотреть страницу в веб-браузере, можно будет увидеть, что по мере того, как страница сужается, изображения меняют свой размер. Но два изображения в основной колонке все же слишком большие. Использование объявления `max-width: 100%` для тега `<img>` приводит к созданию гибких изображений, но во многих случаях вам не нужно, чтобы изображения становились такими же по ширине, как и сама колонка. В случае с двумя изображениями в основной колонке они будут в целом лучше смотреться, если займут только половину размера основной колонки. Этого можно добиться путем применения к этим тегам определенных классов и добавления стилей.

3. Добавьте к файлу `styles.css` к концу таблицы стилей следующие стили:

```
img.half {
  max-width: 50%;
}
```

```
img.left {
  float: left;
  margin: 0 10px 10px 0;
}
```

```
img.right {
  float: right;
  margin: 0 0 10px 10px;
}
```

Первый стиль устанавливает для свойства `max-width` значение, равное только 50 %, что соответствует половине ширины колонки. Остальные два стиля позволяют прижать плавающий элемент изображения к левому или к правому краю колонки, давая тексту обтекать изображения. Чтобы воспользоваться этими стилями классов, к тегам изображений нужно добавить немного кода HTML.

- Найдите в файле `rwd.html` тег `<img>` для изображения `clouds.jpg` — `` — и добавьте код `class="half right"`, чтобы получился следующий код:

```

```

Добавление к элементу более одного класса является довольно устоявшейся и полезной практикой. Таким образом можно объединять простые, разбитые на модули стили для создания более сложных вариантов дизайна. В данном случае класс `half` изменяет размер изображения, а класс `right` делает это изображение плавающим и прижимает его к правому краю. Простым изменением `right` на `left` можно переместить изображение к левому краю колонки, сохраняя свойство размера, заданное в стиле `half`.

- Добавьте для изображения `jellyfish.jpg` атрибут `class="half left"`, придав коду HTML следующий вид:

```

```

Теперь изображения получили нужные размеры, и страница должна приобрести внешний вид, показанный на рис. 14.6. Если нужно, можно будет создать дополнительные стили для других размеров и даже указать размеры в пикселях, позволяя конкретным изображениям изменять размеры до их полной ширины, но не более.



Рис. 14.6. Создание адаптивных изображений

Таким образом, создание адаптивных изображений не составляет особого труда. Нужно просто убрать из HTML атрибуты высоты и ширины и установить значения `max-width`. Если изображениям нужны другие размеры, следует создать селекторы классов с требуемыми значениями свойств `max-width` и применить эти классы к тегам `<img>`.

## Добавление стилей для экранов планшетных устройств

Сейчас дизайн работает одинаково для всех устройств: настольных, планшетных компьютеров и мобильных телефонов. Это гибкий дизайн, который уменьшается или увеличивается, чтобы поместиться в окне браузера. Но в определенный момент боковые панели становятся настолько узкими, что их уже не прочитать. Первый добавляемый вами медиазапрос будет нацелен на экраны, ширина которых находится между 480 и 768 пикселями, с перемещением правой боковой панели в нижнюю часть страницы. В результате дизайн будет преобразовываться из трех колонок в дизайн из легче читаемых двух колонок.

1. Откройте в текстовом редакторе файл `styles.css`. Перейдите в конец файла и добавьте следующий код:

```
@media (min-width: 481px) and (max-width:768px) {
```

Это первый медиазапрос. Он нацелен на экраны, имеющие ширину как минимум 481 пиксел, но не более 768 пикселей. В следующей части этого обучающего урока будет создан медиазапрос для устройств, ширина экранов которых не превышает 480 пикселей. Поэтому данный медиазапрос исключает такие устройства, но также исключает и устройства с шириной экрана, превышающей 768 пикселей.

Иными словами, любые стили, помещаемые в данном разделе таблицы стилей, не будут применяться к браузерам настольных систем (пока вы не уменьшите их окно до слишком узкого), а также не будут применяться к браузерам большинства телефонов.

Первое, что нужно сделать, — удалить свойство `float` из стиля правой боковой панели.

2. Внутри медиазапроса, созданного при выполнении предыдущего шага, добавьте следующий стиль:

```
.sidebar2 {
  float: none;
  width: auto;
}
}
```

Этот код удаляет свойство `float` с боковой панели и устанавливает для ее ширины значение `auto` (отменяющее ширину 20 % из стиля `.sidebar2`, ранее определенного в данной таблице). Но это не заставит боковую панель опуститься ниже двух других колонок. Поскольку `div`-контейнер колонок является плавающим элементом, прижатым к левому краю, вторая боковая панель будет обтекать этот контейнер (подробности поведения плавающих элементов даны в разделе «Использование

плавающих элементов в разметках» гл. 13). Вам нужно с помощью свойства `clear` освободить боковую панель от соседства с этими плавающими элементами.

3. Отредактируйте стиль, созданный при выполнении предыдущего шага, добавив к нему код, выделенный полужирным шрифтом:

```
.sidebar2 {  
  float: none;  
  width: auto;  
  clear: both;  
  border-top: 2px solid black;  
  padding-top: 10px;  
}
```

Свойство `clear` (рассмотренное в подразделе «Отмена правил обтекания» раздела «Управление обтеканием содержимого плавающих элементов» гл. 7) заставляет эту боковую панель опуститься ниже двух других колонок. Кроме того, здесь добавляется верхняя граница для еще большего визуального отделения этого контейнера от верхних колонок.

Теперь если просмотреть страницу, вы увидите, что две колонки не распространяются на всю ширину страницы. Причина в том, что ранее, в шаге 8 последовательности действий для контейнера этих колонок, описанной в подразделе «Изменение порядка следования исходного кода HTML», была установлена ширина 80 % от ширины страницы. Вам нужно сбросить этот стиль внутри медиазапроса для планшетных устройств.

4. Добавьте после стиля `.sidebar2` следующий стиль:

```
.columnWrapper {  
  width: auto;  
}
```

Убедитесь в том, что этот стиль находится внутри медиазапроса. Он сбрасывает ширину контейнера колонок в `auto` (это то же самое, что и 100%), поэтому он распространяется на всю ширину страницы. Поскольку теперь у нас только две колонки, правую боковую панель, применявшуюся к колонке основного содержимого, можно удалить.

5. Добавьте еще один, последний стиль после стиля `.columnWrapper`. Весь медиазапрос должен приобрести следующий вид (изменение выделено полужирным шрифтом):

```
@media (min-width: 481px) and (max-width:768px) {  
  .sidebar2 {  
    float: none;  
    width: auto;  
    clear: both;  
    border-top: 2px solid black;  
    padding-top: 10px;  
  }  
  
  .columnWrapper {  
    width: auto;  
  }  
  
  .main {
```

```
border-right: none;
}
}
```

6. Сохраните файл `styles.css` и просмотрите файл `rwd.html` в веб-браузере. Уменьшайте ширину окна браузера до появления только двух колонок.

Страница должна иметь внешний вид, показанный на рис. 14.7.

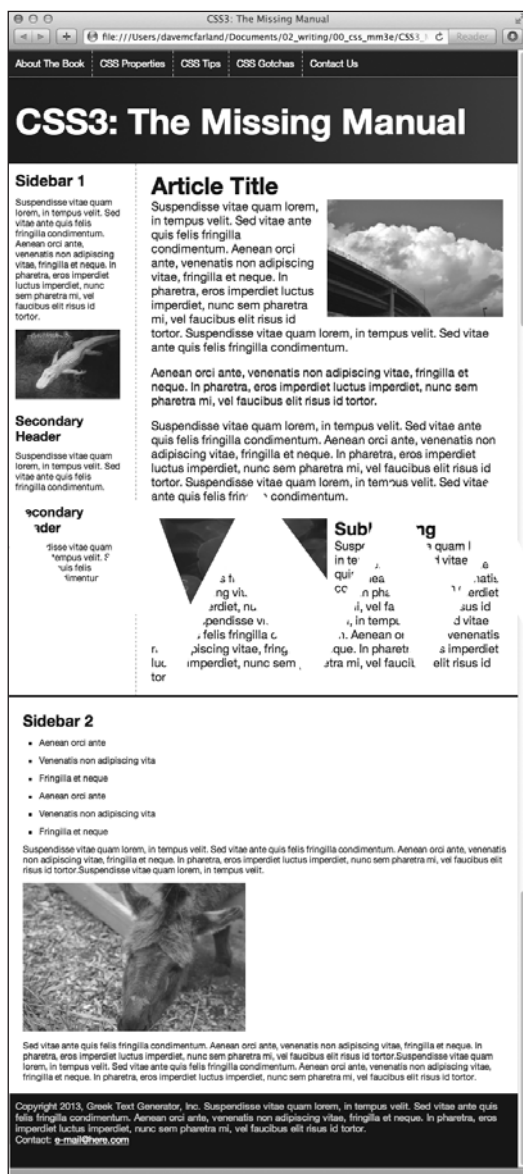


Рис. 14.7. Из трех колонок — в две с помощью одного медиазапроса и всего лишь трех простых стилей

## Добавление стилей для телефонов

И наконец, настал черед добавить стили для устройств, экран которых не шире 480 пикселей.

1. Откройте в текстовом редакторе файл `styles.css`. Перейдите в конец файла и добавьте еще один медиазапрос:

```
@media (max-width:480px) {  
}
```

Убедитесь в том, что он следует после другого медиазапроса и не находится у него внутри. Теперь мы нацелимся на устройства, экран которых не шире 480 пикселей. Сначала нужно будет превратить разметку в линейную, то есть удалить все плавающие элементы, чтобы содержимое выстроилось друг на друге в одну колонку для облегчения его чтения на небольшом экране.

2. Внутри этого медиазапроса добавьте следующий стиль:

```
.columnWrapper, .main, .sidebar1, .sidebar2 {  
  float: none;  
  width: auto;  
}
```

Он удалит плавающие элементы из элемента, заключающего в себе колонки, и из всех колонок. Установка для свойства `width` значения `auto` позволяет элементам заполнять свои контейнеры полностью. Таким образом, теперь три колонки становятся простыми элементами блочного уровня, устанавливаемыми друг на друга, с основным содержимым на вершине и двумя расположенными ниже боковыми панелями.

Область основного содержимого все еще имеет границу из стилей, следующих ранее в таблице стилей. В трехколоночной раскладке эти левые и правые границы служили визуальными разделителями основной колонки и левой и правой боковых панелей, но в этой раскладке, где все находится друг над другом, они не нужны, поэтому их можно удалить. С другой стороны, было бы желательно добавить визуальное разделение между блоками содержимого, которые теперь поставлены друг на друга.

3. После только что созданного стиля (и по-прежнему внутри 480-пиксельного медиазапроса) добавьте два стиля:

```
.main {  
  border: none;  
}  
  
.sidebar1, .sidebar2 {  
  border-top: 2px solid black;  
  margin-top: 25px;  
  padding-top: 10px;  
}
```

Второй представленный здесь групповой селектор добавляет границу вокруг каждой боковой панели, а также верхнее поле и отступ для четкого разделения различных областей содержимого страницы.

Сохраните файл CSS и просмотрите в веб-браузере файл `rwd.html`. Захватите мышью границу окна браузера и сделайте это окно шириной меньше 480 пикселей. Теперь вы увидите дизайн, состоящий из одной колонки. Если просмотреть страницу на телефоне, можно заметить, что заголовок выглядит слишком большим (рис. 14.8). Кроме того, кнопки навигации смотрятся какими-то неуправляемыми. Давайте далее займемся устранением этих двух проблем.



**Рис. 14.8.** Получена подходящая для мобильного телефона ширина страницы, однако заголовок выглядит огромным, занимая слишком много полезного пространства экрана

4. Добавьте внутри 480-пиксельного медиазапроса стиль для тега `<h1>`:

```
header h1 {
  font-size: 1.5em;
}
```

Этот стиль уменьшает размер текста в теге `<h1>`, чтобы теперь он поместился на одной строке. Уменьшение размеров заголовков при конвертировании дизайна для настольных систем в дизайн для просмотра на небольших телефонных экранах зачастую отнюдь не лишено смысла. Можно также прийти к выводу, что для облегчения чтения нужно увеличить размер шрифта основного текста. Для настройки телефонного дизайна вашей страницы существует множество способов, а каким должен быть наилучший внешний вид — решать вам.

Теперь займемся панелью навигации. Пунктирные границы, разделяющие кнопки, смотрятся плохо и сбивают с толку. Кроме того, кнопки выровнены по левому краю, создавая при этом несбалансированную асимметрию. Они будут лучше смотреться отцентрованными и без границ.

5. После стиля `header h1`, созданного в предыдущем шаге, добавьте еще один стиль:

```
nav {
  text-align: center;
}
```

Кнопки `nav` содержатся внутри HTML5-элемента `<nav>`. Установка для свойства `text-align` значения `center` позволяет отцентрировать весь текст, находящийся внутри этого `nav`-элемента. Но кнопки в данный момент указаны в качестве элементов, уплывающих влево, поэтому это только первый шаг. (Использование плавающих элементов для создания горизонтальной панели навигации более подробно рассмотрено в разделе «Создание панелей навигации» гл. 9.)

6. Добавьте после стиля `nav` еще два стиля:

```
nav li {  
  float: none;  
  display: inline-block;  
}
```

```
nav a {  
  float: none;  
  display: inline-block;  
  border: none;  
}
```

Вы удаляете плавающие элементы как из элементов списка, так и из ссылок. Кроме того, превращаете эти элементы в линейные блоки. Как уже упоминалось, использование линейных блоков в элементах навигации позволяет размещать ссылки рядом друг с другом, сохраняя при этом отступы и поля. Кроме того, применение линейных блоков является единственным способом центровки этих кнопок.

7. В окончательном варианте 480-пиксельный медиазапрос должен иметь следующий вид:

```
@media (max-width:480px) {  
  .columnwrapper, .main, .sidebar1, .sidebar2 {  
    float: none;  
    width: auto;  
  }  
  
  .main {  
    border: none;  
  }  
  
  .sidebar1, .sidebar2 {  
    border-top: 2px solid black;  
    margin-top: 25px;  
    padding-top: 10px;  
  }  
  
  header h1 {  
    font-size: 1.5em;  
  }  
  
  nav {  
    text-align: center;  
  }  
}
```



```

nav li {
  display: inline-block;
  float: none;
}

nav a {
  float: none;
  display: inline-block;
  border: none;
}
}

```

Сохраните все ваши файлы и откройте файл `rwd.html` в веб-браузере. Измените размер окна браузера, чтобы его ширина стала меньше 480 пикселей, и посмотрите на дизайн, предназначенный для телефона. Изменяйте ширину окна, пока в поле зрения не появится дизайн, состоящий из двух колонок, а затем продолжайте изменение, пока не увидите версию для настольных устройств, имеющую три колонки. Вы должны видеть дизайны, похожие на те, что показаны на рис. 14.9.



**Рис. 14.9.** Благодаря адаптивному дизайну отдельную веб-страницу можно преобразовать так, чтобы она наилучшим образом помещалась на устройстве с любой шириной экрана

Полная версия этого обучающего урока находится в папке `14_finished`.

# 15 **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 InDesign или Quark XPress. Благодаря лишь нескольким свойствам CSS-позиционирование позволяет поместить элемент в определенное место на странице, скажем, в 100 пикселях от вершины страницы и 200 пикселях от левого края. Казалось, что точное пиксельное размещение обещало, что вы наконец-то сможете проектировать страницу, просто помещая фотографию в одном месте, заголовок — в другом и т. д.

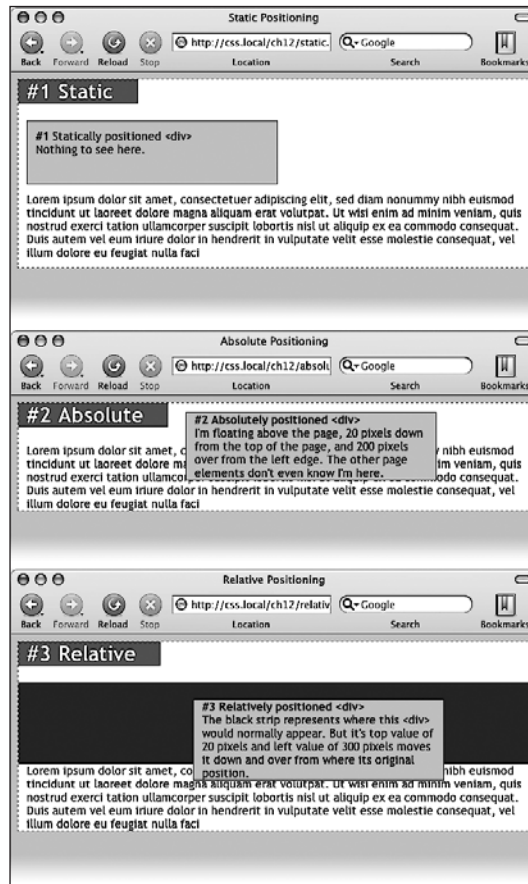
К сожалению, те возможности управления, которые разработчики ожидали от CSS-позиционирования, так и не были реализованы. Различные браузеры всегда отображали элементы, расположенные с помощью CSS, по-разному. Но, что еще более существенно, Сеть работает не так, как печатная брошюра, журнал или книга. 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, а это, 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 сверху над фотографией, 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 боковую панель на странице там, где вы этого желаете, или убедиться, что навигационная панель наверху страницы остается 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 пикселей от верхнего и 200 пикселей от левого края страницы, как показано на рис. 15.1, *посередине* (далее вы узнаете, как писать код с этими инструкциями).



**Рис. 15.1.** CSS предлагает несколько путей воздействия на размещение элементов на веб-странице

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 пикселей и значение `left` — 200 пикселей для относительно размещенного заголовка, вы переместите его на 20 пикселей вниз и 200 пикселей 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. 15.1, *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» раздела «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 неподвижную боковую панель или воспроизвести эффект HTML-фреймов, где только определенная часть страницы прокручивается. Вы скоро узнаете, как выполняется такой эффект.

- **Статическое** позиционирование просто означает, что содержимое соответствует нормальному нисходящему потоку HTML (см. рис. 15.1, *вверху*). Но зачем вам назначать элементу статическое позиционирование? Скорее всего, вы никогда не будете делать этого.

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.

Чтобы определить расстояние от края страницы до соответствующей стороны элемента, используйте любую из действующих в CSS единиц измерения: пиксели, em, проценты и т. д. При позиционировании можно также применять 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» раздела «Эффективные 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 тонкую боковую панель), 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. 15.2, *вверху*.

### ПРИМЕЧАНИЕ

---

Если не указать значение вертикальной позиции (`top` или `bottom`), браузер помещает элемент в то же самое место на странице по вертикали, где бы он располагался при отсутствии позиционирования. Это же справедливо для горизонтальных настроек (`left` или `right`). То есть, если просто установить для элемента позицию `absolute`, но не предоставить значений размещения `top`, `right`, `bottom` или `left`, браузер просто оставляет элемент на том же месте, но располагает поверх другого содержимого.

---

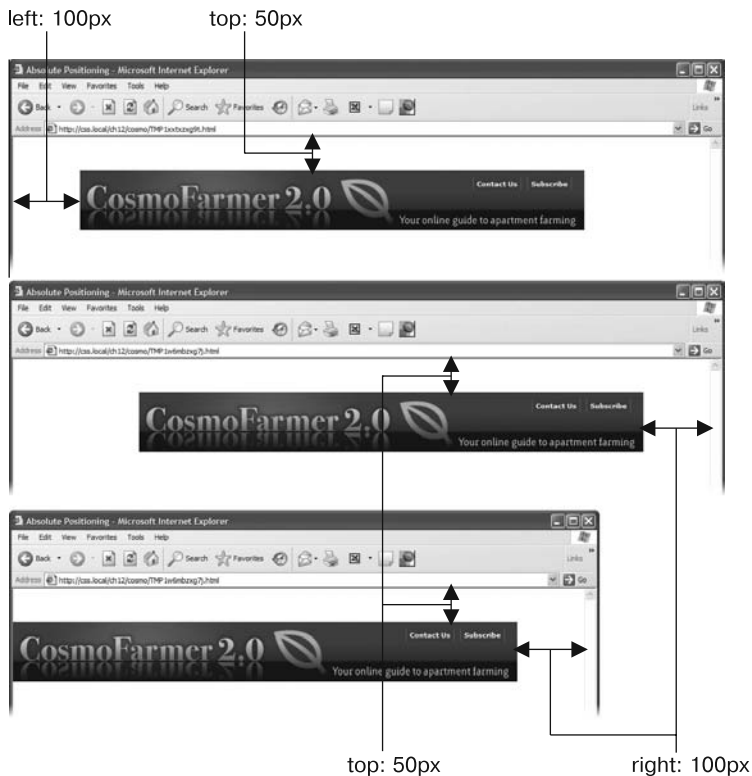
Рассмотрим другой пример: поместим элемент таким образом, чтобы он всегда оставался на фиксированном расстоянии от правой стороны браузера. Когда вы используете свойство `right`, браузер измеряет расстояние от правого края окна до правого края элемента (см. рис. 15.2, *посередине*). Чтобы поместить баннер на расстоянии 100 пикселей 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;
}

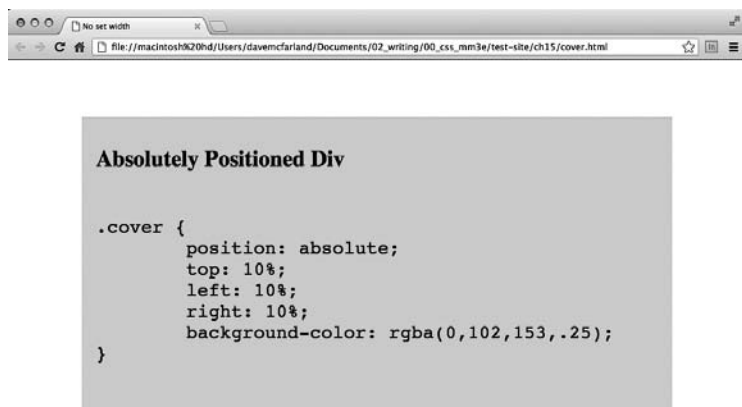
```



**Рис. 15.2.** 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 неизменной (*внизу*)

Поскольку рассчитываемая позиция основывается на правой стороне окна, регулирование его размера автоматически меняет расположение баннера, что вы можете видеть на рис. 15.2, *внизу*. Хотя баннер и перемещается, расстояние от правой стороны элемента до правого края окна браузера остается неизменным.

Можно даже 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 10 % от вершины окна и 10 пикселей от левой и правой его сторон. Чтобы определить блок, вы можете использовать стиль абсолютного позиционирования, который установит свойствам `top`, `left` и `right` значение 10 %. В окне браузера левый край блока начинается от левой стороны окна на расстоянии 10 % от ширины окна, а правый край находится на расстоянии 10 % от правой стороны (рис. 15.3). 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, по-прежнему составляют 10 % от ширины окна браузера.



**Рис. 15.3.** Можно позволить браузеру определить ширину элемента путем простого указания для элемента с абсолютным позиционированием значений как для левой, так и для правой позиции

Свойства `width` и `height`, о которых вы узнали из гл. 7, аналогично работают для позиционированных элементов. Чтобы поместить серый блок размерами  $50 \times 50$  пикселей 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 разделе «Определение параметров высоты и ширины» гл. 7: будьте внимательны с установкой высоты для элементов. Если вы не разрабатываете какую-нибудь графику с уже заданной высотой, то не можете знать, насколько высоким будет на странице конкретный элемент. Вы могли бы определить для боковой панели высоту 200 пикселей, но если вы заканчиваете его описание добавлением слов или картинок, которые сделают боковую панель больше чем 200 пикселей в высоту, то в конечном счете содержимое панели выйдет за ее 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.

## Когда абсолютное позиционирование является относительным

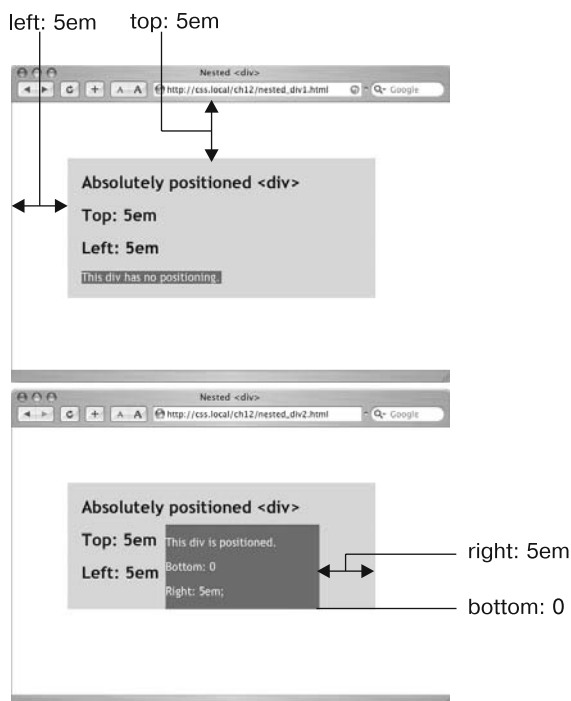
Ранее в этой главе мы говорили о расположении элемента в конкретном месте в окне браузера. Однако абсолютное позиционирование не всегда работает таким образом.

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 пикселей 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» раздела «Стилизация вложенных тегов» гл. 3.

В верхнем изображении на рис. 15.4 светло-серый блок абсолютно позиционирован на расстоянии 5 em от верхнего и левого краев окна браузера.



**Рис. 15.4.** Вы можете разместить элемент относительно окна браузера (*вверху*) или относительно позиционированного предка (*внизу*)

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. 15.4, *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.

## Когда и где использовать относительное позиционирование

Вы получаете существенную выгоду, размещая элемент относительно другого тега: если этот тег передвигается, позиционированный элемент перемещается 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 передвинется вместе с ним (две веб-страницы внизу рис. 15.5).

Рассмотрим представленный рисунок. На верхней веб-странице видно, что круглая графическая кнопка помещена внутри тега `<h1>` (см. рис. 15.5, *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. 15.5, *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. 15.5, *третья страница сверху*). Когда вы передвигаете заголовок ниже, графика также передвигается (см. рис. 15.5, *внизу*).

### ПРИМЕЧАНИЕ

Используйте свойство `background-image` (см. раздел «Добавление фоновых изображений» 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. 15.5),стройте изображение с помощью тега `<img>` и воспользуйтесь относительным позиционированием, рассматриваемым в данном разделе.

Чтобы установить изображение на странице, вы могли использовать значение `relative` свойства `position`, но здесь также есть недостатки. Когда вы задаете 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. 15.1, *внизу*).



Рис. 15.5. Пример использования относительного позиционирования применительно к изображению

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 него. 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 него, 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`.

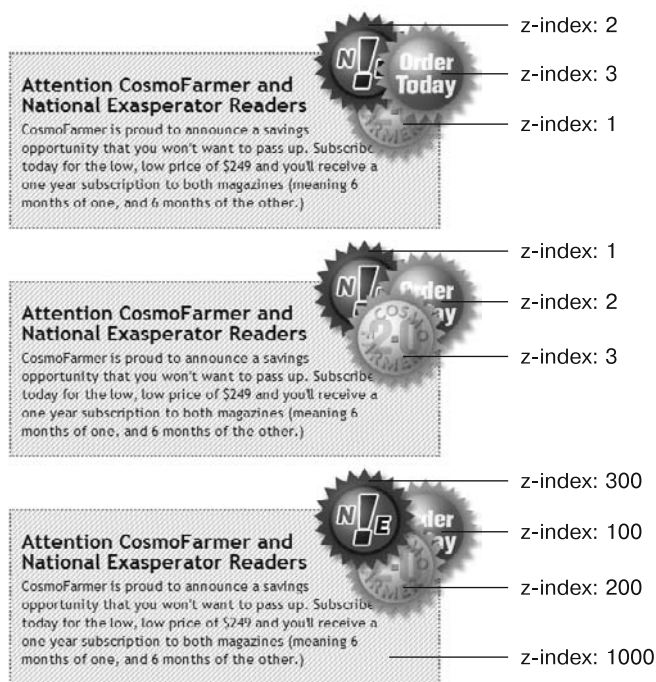
---

## Наложение элементов

Как вы можете видеть на рис. 15.6, 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. 15.6 текстовый блок не позиционирован: он разделяет индекс позиционного уровня самой страницы непосредственно, 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.

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.



**Рис. 15.6.** Веб-страница имеет многослойную структуру

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. 15.6, *вверху*). Когда вы меняете индекс позиционного уровня одного или нескольких изображений, вы изменяете порядок их наложения (см. рис. 15.6, *посередине*).

Для `z-index` можно даже использовать отрицательные числа, которые могут пригодиться, если нужно позиционировать элемент ниже его родительского элемента или любого из его предков. Например, на рис. 15.6, *вверху*, тег `<div>` имеет относительное позиционирование. Если нужно поместить одно из изображений за `<div>`-контейнером, можно воспользоваться отрицательным значением `z-index`:

`z-index: -1;`

Отрицательные значения `z-index` поддерживаются большинством браузеров, кроме Internet Explorer 7 и более ранних версий, поэтому, если нужна поддержка Internet Explorer 7, то отрицательные значения для `z-index` лучше вообще не использовать.

## СОВЕТ

Промежутки в значениях индекса позиционного уровня вполне допустимы. Другими словами, значения 10, 20, 30 определяют то же самое, что и 1, 2, 3. В действительности разрыв в числовых значениях дает возможность последующей вставки в набор дополнительных элементов. Если нужно обеспечить, чтобы над размещаемым элементом больше ничего не появлялось, задайте ему очень большой индекс позиционного уровня, например `z-index: 10 000`; но не переусердствуйте, поскольку максимальное значение, которое обрабатывают некоторые браузеры, равно 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.

Значение `hidden` свойства `visibility` подобно значению `none` свойства `display`, но между ними есть существенное различие. Когда вы устанавливаете свойству `display` элемента значение `none`, он буквально исчезает со страницы, не оставляя следов. Однако задание свойству `visibility` значения `hidden` предотвращает показ браузером содержимого элемента, но оставляет пустое пространство в том месте, где должен был быть элемент. При использовании с абсолютно позиционированными элементами, которые уже удаляются из потока страницы, свойства `visibility: hidden` и `display: none` ведут себя одинаково. Большинство разработчиков просто применяют метод `display: none`, и вообще не пользуются свойством `visibility`.

Есть и другой способ скрытия элемента — установка для его свойства непрозрачности `opacity` нулевого значения:

```
opacity: 0;
```

Чтобы элемент снова появился на экране, его свойству `opacity` можно вернуть значение 1:

```
opacity: 1;
```

Преимущество использования свойства `opacity` заключается в том, что это свойство является одним из CSS-свойств, которые браузер может анимировать. Это означает, что вы можете использовать CSS-переходы, рассмотренные в гл. 10, для анимации изменений непрозрачности. Например, элемент можно сделать постепенно появляющимся в поле зрения, изменяя его непрозрачность от 0 до 1 и добавляя переход.

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. раздел «Псевдоклассы и псевдоэлементы» гл. 3), чтобы сделать невидимый элемент видимым. Предположим, например, что вам нужно поместить надпись поверх изображения, но при этом она должна появляться только при прохождении указателя мыши посетителя над изображением (рис. 15.7). Для получения результата выполните следующие действия.

### 1. Создайте HTML-код для изображения и надписи.

Один из способов предполагает использование HTML5-тегов `<figure>` и `<figcaption>`:

```
<figure class="hat">

<figcaption>A picture of a hat</figcaption>
</figure>
```

Здесь к тегу `<figure>` применяется класс `hat`, поэтому стиль можно применить только к этому одному изображению.

#### ПРИМЕЧАНИЕ

---

Как уже упоминалось во врезке «Обходной прием. Как заставить Internet Explorer 8 понимать HTML5» в разделе «Дополнительные теги в HTML5» гл. 1, Internet Explorer 8 и более ранние версии не применяют CSS к элементам HTML5 без небольшой помощи со стороны кода JavaScript. Если вам не хочется возиться с этим кодом, можно просто воспользоваться вместо тегов `<figure>` и `<figcaption>` тегами `<div>`.

---

### 2. Укажите позицию для надписи.

Для помещения надписи поверх изображения используется абсолютное позиционирование. Чтобы поместить надпись относительно тега `<img>`, нужно установить для его родительского тега `<figure>` относительное позиционирование, а также задать параметры ширины и высоты, соответствующие размерам фотографии.

```
.hat {
  position: relative;
  width: 100px;
  height: 100px;
}

.hat figcaption {
  position: absolute;
  bottom: 0;
  left: 0;
  right: 0;
  background-color: white;
}
```

Надпись помещается в нижней части изображения (`bottom: 0`). За счет установки нулевых значений для его свойств `left` и `right` надпись займет всю ширину рисунка.

### 3. Скройте надпись.

При использовании данного кода браузер покажет надпись поверх изображения, но вам нужно, чтобы это происходило только при прохождении указателя чьей-либо мыши над изображением. Чтобы скрыть надпись, добавьте к стилю `.hat figcaption` либо `visibility: hidden`, либо `display: none`.

```
.hat figcaption {
  display: none;
  position: absolute;
  bottom: 0;
  left: 0;
  right: 0;
```

```
background-color: white;
}
```

4. Сделайте так, чтобы надпись появлялась, когда посетитель проводит указатель мыши над изображением.

Для этого воспользуйтесь псевдоклассом `:hover`. Нужно сделать надпись видимой при прохождении над изображением указателя мыши посетителя. К сожалению, нет способа создания стиля, влияющего на надпись при прохождении указателя мыши над изображением. Но так как надпись находится внутри тега `<figure>`, можно создать селектор потомка, влияющий на надпись при прохождении указателя мыши над изображением:

```
.hat:hover figcaption {
    display: block;
}
```

Этот селектор потомка предписывает следующее: «Нужно нацелиться на любой тег `<figcaption>`, который находится внутри элемента с классом `hat`, но только когда указатель мыши располагается над элементом». Данный селектор работает только потому, что тег `figcaption` является потомком элемента, над которым проходит указатель мыши.

Эту идею можно использовать для создания основанных на коде CSS появляющихся подсказок. Чтобы ознакомиться с основным CSS-методом добавления появляющихся подсказок — дополнительной информации, возникающей при прохождении указателя чьей-либо мыши над ссылкой, — обратитесь к сайту по адресу [www.menucool.com/tooltip/css-tooltip](http://www.menucool.com/tooltip/css-tooltip). Есть также множество JavaScript-вариантов на выбор: полноценным и простым в использовании JavaScript-средством для создания появляющихся подсказок, основанным на применении среды jQuery, является дополнительный jQuery-модуль `aTip2`: <http://craigsworks.com/projects/qtip2/>.

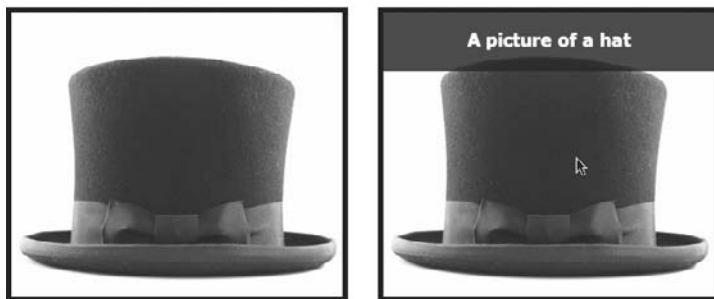
Как уже упоминалось, для скрытия и показа элемента можно также воспользоваться свойством `opacity`, а для анимации эффекта можно воспользоваться свойством `transition`. Для этого можно изменить предыдущие стили следующим образом:

```
.hat figcaption {
    opacity: 0;
    -webkit-transition: opacity .5s ease-in;
    -moz-transition: opacity .5s ease-in;
    -o-transition: opacity .5s ease-in;
    transition: opacity .5s ease-in;
    position: absolute;
    bottom: 0;
    left: 0;
    right: 0;
    background-color: white;
}

.hat:hover figcaption {
    opacity: 1;
}
```



Это действие можно увидеть в обучающем уроке в конце главы. Но необходимо помнить, что свойство `transition` не работает в Internet Explorer 9 и более ранних версиях. Однако в данном случае ничего страшного не произойдет. Ваши посетители все равно увидят надпись, но только без ее плавного появления и исчезновения.



**Рис. 15.7.** Элемент с абсолютным позиционированием можно поместить над другим элементом, но скрыть его (слева), пока посетитель не наведет указатель мыши на родительский элемент (справа)

## Эффективные стратегии позиционирования

Как говорилось в начале этой главы, если вы попытаетесь использовать CSS-позиционирование для размещения каждого элемента страницы, то можете столкнуться 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.

Из этого раздела вы узнаете, как применять 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.

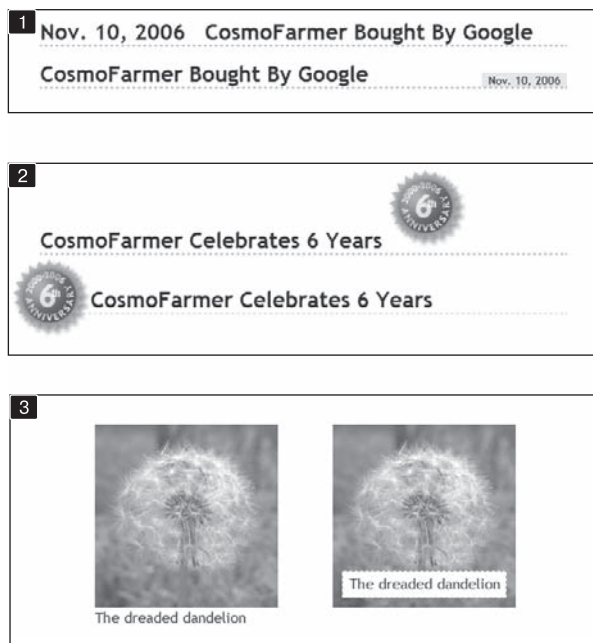
### Позиционирование внутри элемента

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. 15.8,1, 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 `<span>` (см. врезку «Информация для новичков. HTML-теги `<div>` и `<span>`» в разделе «Селекторы классов: точное управление» гл. 3) — распространенный выбор при использовании класса для расположенного в линии текста, чтобы определить его отдельно от остальной части абзаца.

```
<h1><span class="date">Nov. 10, 2006</span> CosmoFarmer Bought By Google</h1>
```





**Рис. 15.8.** Абсолютное позиционирование хорошо подходит для простых элементов дизайна, например для размещения даты в нижнем правом углу заголовка (1), выдавливания изображения из содержащего его блока (2) или расположения заголовка поверх фотографии (3)

Теперь стоит вопрос о создании стилей. Нужно задать содержащему элементу (здесь — тегу `<h1>`) значение `relative` (относительное позиционирование). Затем следует применить абсолютное позиционирование (значение `absolute`) для элемента, который желаете разместить, — даты. Здесь приведен CSS-код для нижнего изображения в первом примере на рис. 15.8,1:

```
h1 {
  position: relative;
  border-bottom: 1px dashed #999999;
}
h1 .date {
  position: absolute;
  bottom: 0;
  right: 0;
  font-size: .5em;
  background-color: #E9E9E9;
  color: black;
  padding: 2px 7px 0 7px;
}
```

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.

## Исключение элемента за пределы блока

Вы можете также использовать позиционирование для размещения элемента таким образом, чтобы он «выглядывал» из-за другого элемента. На рис. 15.8,2, 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 применяется 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;
}
```

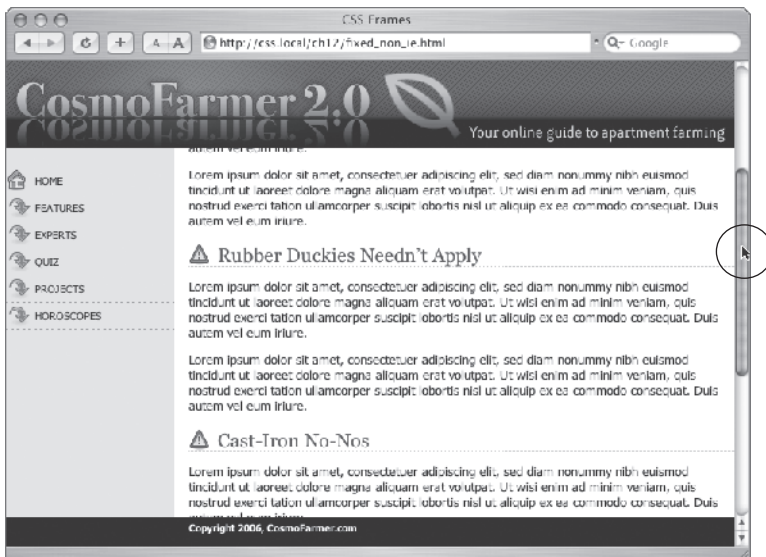
Основная концепция этого кода такая же, что и в предыдущем примере, но с некоторыми дополнениями. Значения `top` и `left` изображения отрицательные, так что графика фактически появляется на расстоянии 30 пикселей над вершиной заголовка и 30 пикселей слева от левого края заголовка. Будьте внимательны, 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 пикселей также «вытолкнет» текст заголовка из-под абсолютно позиционированного изображения.

## Создание фреймов с помощью стилей CSS с использованием фиксированного позиционирования

Поскольку многие веб-страницы длиннее одного экрана, может возникнуть необходимость сохранения на виду некоторых элементов страницы, например навигационной панели, поля для поиска или логотипа сайта. *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 посетителям, которые используют экранных дикторов из-за проблем со зрением или которые хотят 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 увидеть страницу, созданную с использованием фиксированного позиционирования, на рис. 15.9. С помощью значения `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.

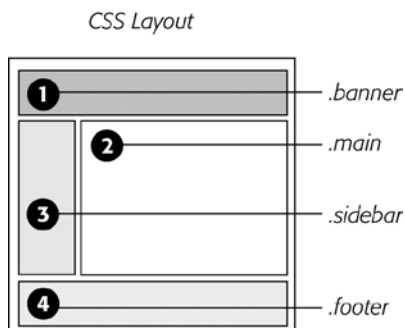


**Рис. 15.9.** Страница с фреймами, созданная посредством CSS. Полоса прокрутки (отмечена кружком) позволяет прокручивать только большую текстовую область; верхний и нижний баннеры и боковая панель остаются неподвижными

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. 15.9, у которой есть фиксированный баннер, боковая панель и нижний колонтитул, а также прокручиваемая область с главным содержимым.

1. Добавьте теги `<div>` с классами (или атрибутами ID) для каждого раздела страницы. У вас может быть четыре основных тега `<div>` с такими классами (или ID), как `banner`, `sidebar`, `main` и `footer` (рис. 15.10). 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.



**Рис. 15.10.** С фиксированным позиционированием можно закрепить все элементы страницы, чтобы они всегда были в поле зрения при прокрутке. В этом примере заголовок (1), боковая панель (3) и нижний колонтитул (4) зафиксированы, а область с главным содержанием (2) может прокручиваться

- Добавьте свой материал к каждому тегу `<div>`. 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, боковая панель 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 фиксированная боковая панель длиннее, чем окно браузера пользователя, посетитель не сможет увидеть боковую панель целиком. Поскольку фиксированные элементы не прокручиваются, у пользователя не будет никакой возможности (за исключением покупки большего монитора) увидеть содержимое боковой панели, которое не соответствует окну его браузера.

- 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, укажите ширину для элементов.

#### ПРИМЕЧАНИЕ

В отличие от абсолютного позиционирования, фиксированное всегда задается относительно окна браузера, даже когда элемент с таким позиционированием помещается внутри другого тега с относительным или абсолютным позиционированием.

Стили для размещения элементов 1, 3 и 4 на рис. 15.10 выглядят следующим образом:

```
.banner {
  position: fixed;
  left: 0;
  right: 0;
  top: 0;
}
```

```
.sidebar {
  position: fixed;
```

```
left: 0;
top: 110px;
width: 175px;
}

.footer {
position: fixed;
bottom: 0;
left: 0;
right: 0;
}
```

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 `<div>` 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;
}
```

Фиксированное позиционирование хорошо поддерживается в браузерах: в Internet Explorer 8 и выше, а также во всех других основных браузерах (включая самые последние браузеры мобильных устройств iOS и Android).

## 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 заключали фрагменты HTML-кода в теги `<div>` и добавляли к ним имена классов, в этих уроках большая часть работы с HTML уже была выполнена. Вы можете сосредоточиться на оттачивании ваших новых навыков в CSS.

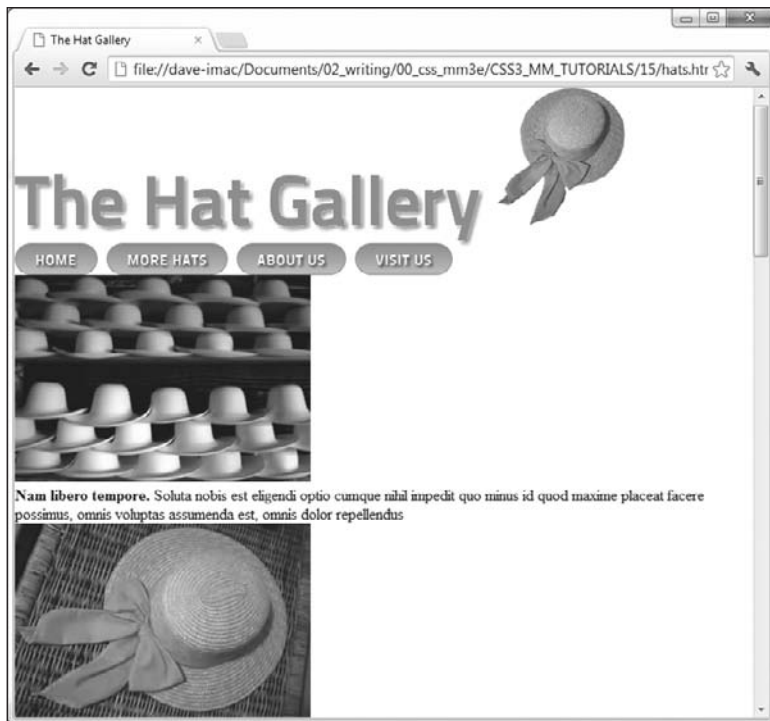
Чтобы приступить, загрузите учебные файлы, расположенные на сайте, [www.sawmac.com/css3/](http://www.sawmac.com/css3/). Как работать с ними, описывается в конце гл. 2.

### Улучшение баннера страницы

Сначала сделаем несколько маленьких, но визуально важных изменений в баннере страницы. Создадим стили, которые ссылаются на HTML-теги с примененными к ним классами (эта часть работы также уже сделана за вас).

1. Запустите браузер и откройте файл `hats.html` из папки 15.

На этой веб-странице (рис. 15.11) начните с перепозиционирования некоторых частей баннера.



**Рис. 15.11.** Обычная статическая HTML-страница, где все заполнено сверху донизу. Вы можете сделать ее более удобной для чтения, организовав все содержимое по столбцам

- Откройте файл `hats.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 небольшого изображения шляпы 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.

- Во внутренней таблице стилей добавьте новый стиль:

```
header .badge {
  position: absolute;
  top: -20px;
  left: -90px;
}
```

Графика находится внутри HTML5-тега `<header>` с классом `badge`. Этот стиль приводит к тому, что левый верхний угол графики помещается на 90 пикселей влево и 20 пикселей над верхним краем страницы.

Теперь просмотрите страницу. Вы увидите, что у нее есть несколько проблем. Она «нависает» над краем страницы, а вы хотите, чтобы она располагалась над краем области баннера. Займемся этой проблемой.

4. Добавьте перед только что созданным стилем следующий стиль:

```
header {  
  position: relative;  
}
```

CSS-код для стилей, которые управляют основным разделом страницы (как этот стиль `header`), принято помещать над кодом для стилей, формирующих 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 `header` идет первым во внутренней таблице стилей, потому что он применяется к большому фрагменту HTML-кода. Но вы должны держать стиль `header .badge` рядом с ним, так как добавляете больше стилей на страницу.

Стиль `header` 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 мы создали в шаге 3. Теперь он смещен на 20 пикселей вверх и на 90 пикселей влево от области баннера. Значок все еще немного «нависает» над страницей, поэтому нужно добавить небольшие поля для заголовка, чтобы немного его опустить.

5. Отредактируйте стиль `header`, добавив в нижнюю часть две строки, выделенные полужирным шрифтом:

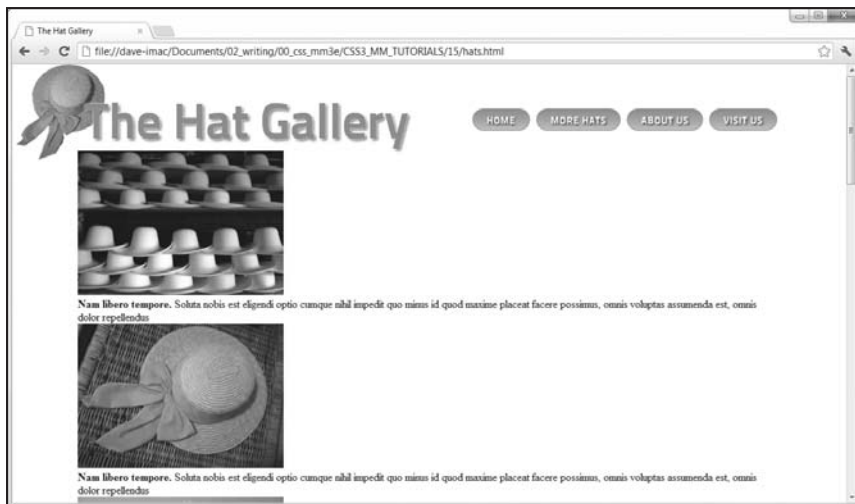
```
header {  
  position: relative;  
  margin-top: 20px;  
  padding: 20px 0 0 10px;  
}
```

Поле добавит достаточное пространство над тегом `header`, чтобы он и графика переместились вниз. Кроме того, отступ добавит пространство внутри тега заголовка, чтобы заголовок и рядом расположенная навигационная панель не смотрелись слишком скученно. Но теперь появилась другая проблема — заголовок `The Hat Gallery` частично скрыт под значком. Перекрывание элементов является одним из недостатков абсолютного позиционирования. В данном случае проблему можно решить путем изменения свойства знака `z-index` и помещения его за текстом.

6. Добавьте к стилю `header .badge` свойство `z-index: -1`:

```
header .badge {  
  position: absolute;  
  top: -20px;  
  left: -90px;  
  z-index: -1;  
}
```

Значение `-1` приводит к тому, что элемент с абсолютным позиционированием помещается за своим родительским элементом, в данном случае за текстом (рис. 15.12). Затем абсолютным позиционированием нужно воспользоваться для перемещения панели навигации в правую сторону элемента `header`.



**Рис. 15.12.** Абсолютное позиционирование является хорошим подспорьем в размещении небольших элементов вроде изображения шляпы и панели навигации

7. Добавьте после стиля `header .badge` еще один стиль.

```
header nav {
  position: absolute;
  right: 0;
  top: 45px;
}
```

Хотя задать позицию панели навигации можно за счет плавающего тега `<h1>`, в данном случае намного проще будет воспользоваться абсолютным позиционированием. Здесь создается стиль тега для HTML5-тега `<nav>`, когда он также находится в теге `<header>`. Следует помнить, что в шаге 4 мы задали `header` относительную позицию, следовательно, любые теги внутри этого тега, например тег `<nav>`, позиционируются относительно него, поэтому нулевое значение для свойства `right` в этом стиле приводит к тому, что правый край панели навигации помещается по правому краю баннера (см. рис. 15.12).

## Добавление надписи к фотографии

В гл. 8 мы рассмотрели один из способов добавления подписи к фотографии (см. раздел «Обучающий урок 1: совершенствуем изображения»). 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 `hats.html` в текстовом редакторе.

На этой странице находится галерея фотографий. HTML-код для одного изображения имеет следующий вид:

```
<figure>
  <img src="images/file00079963469.jpg" width="300" height="210"
```



```

    alt="Panama">
    <figcaption><strong>Nam libero tempore.</strong> Soluta nobis est
    eligendi
    optio cumque nihil impedit quo minus id quod maxime placeat facere
    possimus, omnis voluptas assumenda est, omnis dolor repellendus
    </figcaption>
</figure>

```

В этом примере используются HTML5-теги `<figure>` и `<figcaption>`. Тег `<figure>` обычно является тегом блочного уровня, но поскольку нужно, чтобы изображения располагались рядом друг с другом, начните с превращения их в линейные элементы.

---

#### ПРИМЕЧАНИЕ

Поскольку Internet Explorer 8 и более ранние версии не могут применять стиль к HTML5-тегам, чтобы эти браузеры справились с программой, нужно добавить немного кода JavaScript. Чтобы понять, как работает эта схема, прочитайте врезку «Обходной прием. Как заставить Internet Explorer 8 понимать HTML5» в разделе «Дополнительные теги в HTML5» гл. 1.

---

2. Ниже ранее созданного стиля `header nav` добавьте еще один стиль.

```

.gallery figure {
  display: inline-block;
  width: 300px;
  height: 210px;
  margin: 15px;
  position: relative;
}

```

Этот код создает селектор потомков, который применяется ко всем тегам `<figure>`, сгруппированным внутри тега `<div>` с классом `gallery`. Здесь используется селектор потомков, потому что к этой странице могут добавляться другие теги `figure`, не являющиеся частью галереи, которые должны быть отформатированы по-другому. Этот селектор потомков предназначен только для тех тегов `<figure>`, которые нас интересуют в данный момент.

После придания тегу `figure` свойств линейного блочного элемента (`inline-block`) все изображения смогут находиться рядом друг с другом. Значения `width` и `height` соответствуют ширине и высоте изображений. То есть нужно, чтобы теги `<figure>` были достаточно большими, чтобы вместить изображения. Значение `margin` добавляет небольшое пространство вокруг изображений, чтобы они не наталкивались друг на друга. И наконец, объявление `position: relative` устанавливает новый контекст позиционирования, чтобы каждую надпись можно было позиционировать относительно относящегося к ней изображения.

Теперь настал черед позиционирования надписей.

3. Ниже только что добавленного стиля добавьте еще один стиль:

```

.gallery figcaption {
  position: absolute;
  top: 15%;
  bottom: 15%;
  left: 0;
  right: 0;
}

```

```
background-color: rgb(153,153,153);
background-color: rgba(153,153,153,.9);
}
```

Теги `<figcaption>` получают абсолютное позиционирование с использованием всех четырех квадрантов позиционирования: `top`, `bottom`, `left` и `right`. По существу, надписи будут распространяться на все изображение, но помещаться немного ниже его верхнего края и немного выше нижнего края (фактически на 15 % ниже и выше). Использование всех четырех настроек означает, что вам не нужно переживать за настройку ширины или высоты надписей, вместо этого вы оставляете ее на усмотрение браузера.

И наконец, устанавливаются два объявления фонового цвета `background-color`. Первое предназначено для Internet Explorer 8 и более ранних версий (не понимающих настроек прозрачности в формате цвета RGBA). А второе — для современных браузеров, с установкой полупрозрачного фона поверх каждого изображения, что означает, что изображения можно видеть через фон надписей.

Теперь займемся простыми улучшениями текста.

4. Отредактируйте только что созданный стиль, добавив внизу код, выделенный полужирным шрифтом:

```
.gallery figcaption {
  position: absolute;
  top: 15%;
  bottom: 15%;
  left: 0;
  right: 0;
  background-color: rgb(153,153,153);
  background-color: rgba(153,153,153,.9);
  padding: 20px;
  font-family: Titillium, Arial, sans-serif;
  font-weight: 400;
  font-size: .9em;
  color: white;
}
```

Отступы (`padding`) создают небольшую разрядку для текста, а другие свойства задают шрифт, размер и цвет.

Если теперь просмотреть страницу, можно увидеть, что надписи появляются над всеми изображениями. Далее нужно будет изменить стиль, чтобы надписи появлялись только при прохождении указателя мыши посетителя над изображением. Начнем со скрытия надписей.

5. Добавьте к стилю `opacity: 0`:

```
.gallery figcaption {
  position: absolute;
  top: 15%;
  bottom: 15%;
  left: 0;
  right: 0;
  background-color: rgb(153,153,153);
```

```
background-color: rgba(153,153,153,.9);
padding: 20px;
font-family: Titillium, Arial, sans-serif;
font-size: .9em;
font-weight: 400;
color: white;
opacity: 0;
}
```

Установка для непрозрачности `opacity` значения 0 делает надпись невидимой. Для скрытия надписей можно также воспользоваться объявлением `display: none;` или `visibility: hidden;`, но выбранный способ позволяет анимировать значение непрозрачности путем использования CSS-свойства `transition`, и этот эффект будет вскоре добавлен. Но сначала нужно добавить состояние `:hover`, чтобы при прохождении над изображением указателя мыши посетителя появлялась надпись для этого изображения.

6. Добавьте к таблице стилей следующий код:

```
figure:hover figcaption {
  opacity: 1;
}
```

Этот хитрый фрагмент кода CSS можно перевести как «при прохождении указателя мыши посетителя над элементом `figure` (`figure:hover`), установить непрозрачность надписи в 1». То есть при перемещении указателя мыши над тегом `<figure>` его тег-потомок `<figcaption>` становится видимым. Сохраните страницу и посмотрите, что получилось. Когда указатель мыши проходит над изображением, должна появляться надпись. Мы можем анимировать этот эффект, добавив к стилю `.gallery figcaption` переход.

7. Отредактируйте стиль `.gallery figcaption`, чтобы он приобрел следующий вид (добавления выделены полужирным шрифтом):

```
figcaption {
  position: absolute;
  top: 15%;
  bottom: 15%;
  left: 0;
  right: 0;
  background-color: rgb(153,153,153);
  background-color: rgba(153,153,153,.9);
  padding: 20px;
  font-family: Titillium, Arial, sans-serif;
  font-size: .9em;
  font-weight: 400;
  color: white;
  opacity: 0;
  -webkit-transition: opacity .75s ease-out;
  -moz-transition: opacity .75s ease-out;
  -o-transition: opacity .75s ease-out;
  transition: opacity .75s ease-out;
}
```

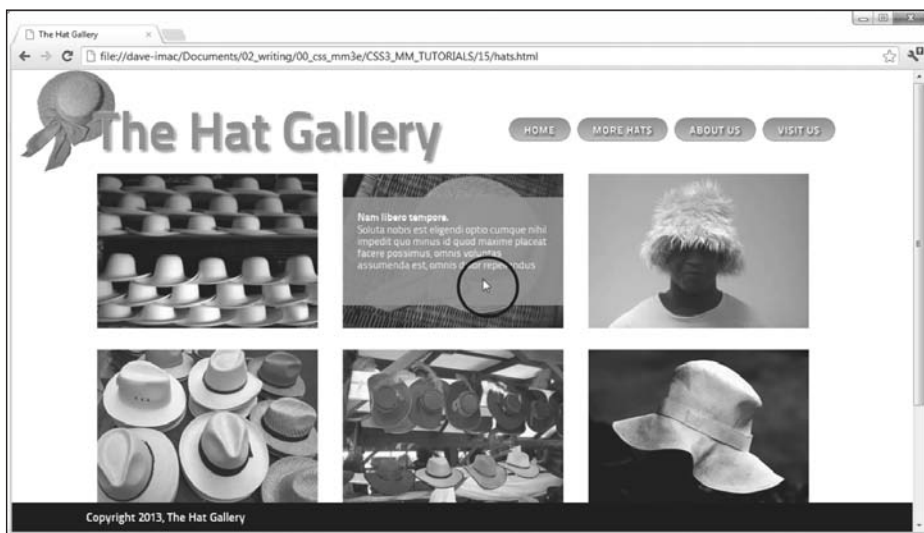
Вы добавили свойство `transition`. Разумеется, чтобы оно заработало во многих браузерах, нужно добавить соответствующие префиксы производителей (`-webkit-` и т. д.). Поскольку Internet Explorer 9 и более ранние версии не понимают переходы, но здесь это не создает проблем, надписи все равно будут появляться в этих браузерах. Они будут просто моментально появляться и исчезать, а не постепенно становиться видимыми и растворяться.

И наконец, нужно привязать сведения об авторских правах к нижней части окна браузера, используя фиксированное позиционирование.

8. Добавьте к таблице стилей еще один, последний стиль:

```
footer {  
  position: fixed;  
  bottom: 0;  
  left: 0;  
  right: 0;  
  padding: 10px;  
  background-color: black;  
  color: white;  
}
```

Как уже говорилось, фиксированное позиционирование позволяет «привязать» элемент к определенному месту в окне браузера. В данном случае привязка осуществляется к нижней части страницы (`bottom: 0`) и производится распространение на всю ширину страницы (благодаря объявлениям `left: 0` и `right: 0`). Последние три объявления просто добавляют немного пространства вокруг нижнего колонтитула, а также придают ему черный фон и белый текст. В окончательном виде страница должна выглядеть, как показано на рис. 15.13. Полную версию этого урока можно найти в папке `15_finished`.



**Рис. 15.13.** Используя искусный код CSS, можно без особого труда заставить надписи плавно появляться при наведении указателя на изображение (отмечено кружком)

## ЧАСТЬ 4

# CSS для продвинутых

**Глава 16.** CSS для распечатываемых веб-страниц

**Глава 17.** Совершенствуем навыки работы с CSS

# 16 CSS для распечатываемых веб-страниц

Возможно, в это сложно поверить, но есть люди, которым хочется увидеть вашу веб-страницу на бумаге. Люди, занимающиеся покупками на вашем сайте, могут, например, захотеть распечатать чек. Если вы ведете кулинарный сайт, то, конечно же, вам понадобится предоставить своим посетителям возможность распечатать их любимые рецепты или карточки для своей кухонной картотеки. Даже если ваш сайт состоит в основном из текстовых статей, предназначенных для чтения в режиме онлайн, найдутся посетители, пожелавшие получить распечатку для классификации проекта или архивирования. Так 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? Наверное, нет.

Веб-дизайнеры обычно выходили из этого затруднительного положения, создавая отдельные, предназначенные для печати, версии своих сайтов. То есть 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 способом, когда отображается на экране, и другим — при печати (рис. 16.1). 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. Обратите внимание, как на рис. 16.1 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 учли все возможные способы, которыми люди могут 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, таких как мобильные телефоны, карманные компьютеры и телевизоры, есть свои собственные уникальные требования, когда дело доходит до работы с веб-страницами.

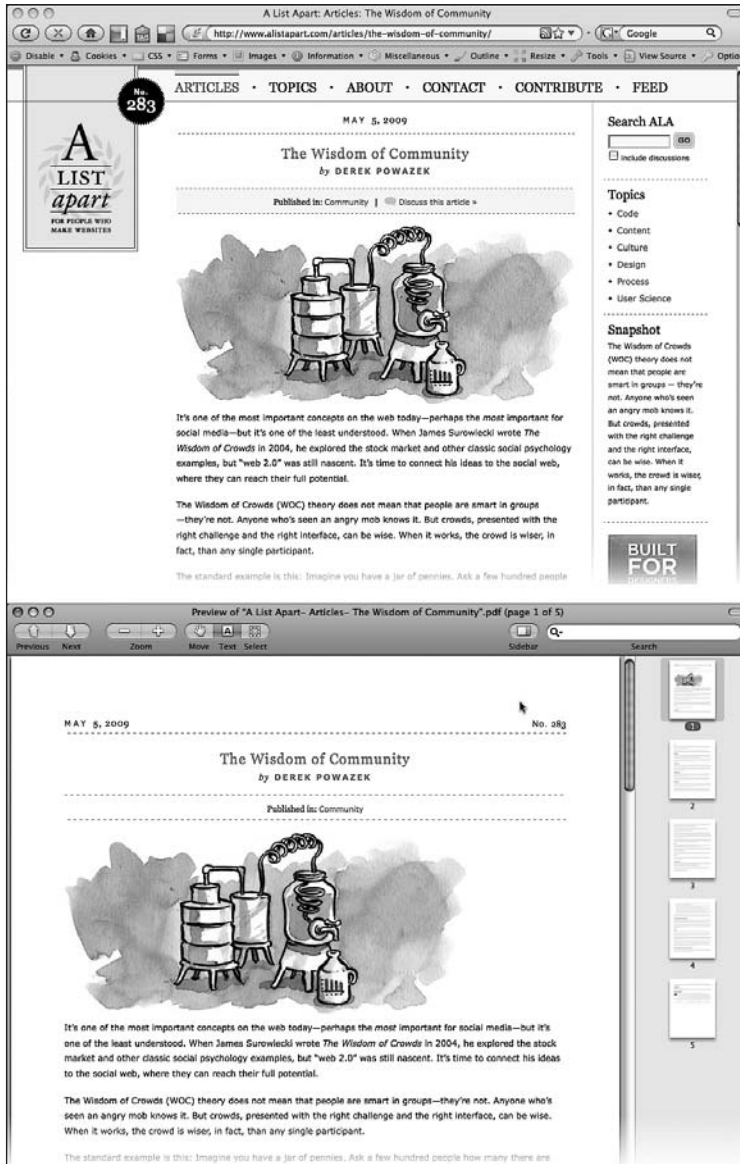


Рис. 16.1. Страница сайта, посвященного веб-разработке (A List Apart), выглядит привлекательной на экране (вверху) и простой и понятной на бумаге (внизу)

Чтобы обеспечить все мыслимые методы просмотра сайтов, CSS позволяет вам создавать стили и таблицы стилей, которые предназначены 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 для просмотра на экране, а другую — для печати. Многие из этих типов предназначены для специализированных приложений, таких как Braille Reader (программа оптического распознавания текста для слабовидящих пользователей), экранный диктор (для тех, кто хочет или кому нужно услышать, что содержится на странице) или телетайп. 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` — 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 подходящие для печати размеры шрифта, цвета, графику и т. д.

Один из подходов разработки стилей состоит в создании стилей сначала лишь для экрана и присоединении их каким-либо из методов, описанных далее (таким 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, негативно влияющих на вид страницы при ее печати. Этот подход рассматривается в разделе «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 веб-страницам, 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" media="print" href="print.css"/>
```

---

#### ПРИМЕЧАНИЕ

Формально CSS также позволяет определить тип устройства, когда вы используете правило `@import` для присоединения внешней таблицы стилей (см. раздел «Внешние таблицы стилей» гл. 2) таким образом: `@import url(print.css) print;`. Но, поскольку Internet Explorer 8 и более ранние версии не понимает этот код, его лучше не использовать.

---

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 задать множество типов устройств, разделяя их запятыми. Присоединенная внешняя таблица стилей, предназначенная для нескольких устройств, могла бы быть такой:

```
<link rel="stylesheet" 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 инструкцию `@media`. В гл. 14 вы уже сталкивались с инструкцией `@media` при создании медиазапросов для задания различных свойств и настроек веб-браузера (например, ширины экрана). Инструкцию `@media`

можно также использовать для задания различных экранов и при выводе на печать. 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 инструкцию `@media` следующего вида:

```
@media print {  
  /* описывайте стили для печати здесь */  
}
```

Не забудьте указать закрывающую фигурную скобку (на последней строке), иначе инструкция 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 инструкцию `@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. 16.2).

### СОВЕТ

Чтобы посмотреть 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.

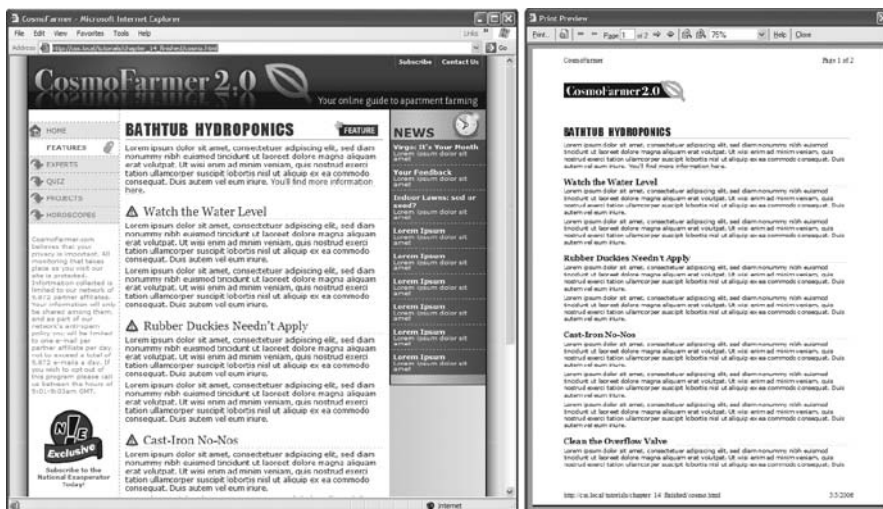


Рис. 16.2. С помощью таблицы стилей для принтера вы можете убрать боковую панель, навигационные панели и другое содержимое, разработанное для просмотра сайтов (слева). Результатом будет просто отформатированный документ — идеальный для печати (справа)

## Использование правила `!important` для отмены экранного стиля

Как говорилось ранее, часто бывает полезно 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 — применение 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

Необязательно, 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. Яркий зеленый текст будет хорошо выглядеть на экране, но в распечатке он может стать трудночитаемым бледно-серым текстом.

Задание размеров в пикселах и `em` (см. раздел «Установка размера шрифта» гл. 6) имеет смысл для экранного текста, но для печати лучше использовать *пункты*. Это единицы, в которых Word и другие программы обработки текста измеряют высоту шрифта. На практике большинство браузеров так или иначе переводят пиксели и `em` в нечто более подходящее для принтеров. Базовый экранный размер шрифта для большинства браузеров — 16 пикселей — 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;  
}
```

Аналогично экранные цвета часто плохо переводятся, когда выводятся 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 выше, 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» раздела «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 панелям и другим элементам страницы придает контрастность и визуальную привлекательность вашим веб-страницам. Но вы не можете быть уверенными, что фон правильно отобразится, когда эти страницы будут печататься. Поскольку цветные фоны съедают чернила и порошок принтера, большинство браузеров обычно не отправляют их на печать, а многие посетители Сети не включают фоны 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 после 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: #FFF;
}
```

Этот стиль устанавливает фону белый цвет — цвет бумаги. Кроме того, чтобы получить четкий печатный текст, этот стиль выбирает для шрифта черный цвет.

## ИНФОРМАЦИЯ ДЛЯ НОВИЧКОВ

### 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:

## ИНФОРМАЦИЯ ДЛЯ НОВИЧКОВ

`background: white`. Помните, что свойство `background` (см. раздел «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 в белый, 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 настроят 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 подходящий 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:

```

```

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.

## ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

## Показываем ссылки при печати

Предположим, сотрудник поделился с вами распечаткой интересной статьи, которую он обнаружил в Сети.

Вы читаете и сталкиваетесь с таким переходом: «И вот я нашел секрет вечной жизни здесь». Подчеркивание

## ИНФОРМАЦИЯ ДЛЯ НОВИЧКОВ

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/](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 (и более поздних версиях), Firefox, Safari и Opera, 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) ") ";
}
```

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](http://../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, 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/](http://www.alistapart.com/articles/goingtoprint/).

Кроме того, поскольку попадаются и слишком длинные ссылки, они могут не помещаться на распечатываемой странице и будут обрезаны по достижению края бумажного листа. Чтобы этого не произошло, можно воспользоваться свойством `word-wrap`, чтобы заставить браузер разбивать ссылки на несколько строк:

```
a {
  word-wrap: break-word;
}
```

Добавьте показанный выше стиль к своей таблице стилей, предназначенной для вывода на печать.

## Скрытие нежелательных областей страницы

Веб-страницы часто нагружаются вспомогательными информационными средствами, такими как навигационные панели, боковая панель, полная 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 в части 3 книги, один из способов разбить страницу состоит в том, чтобы определить теги `<div>` для различных элементов: баннеров, основной навигации, содержимого, уведомления об авторском праве и т. д. При разработке каждого тега `<div>`, используя относительное или абсолютное позиционирование,



вы можете разместить различные элементы страницы там, где 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 не выводить на печать боковую панель, 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, более подходящую для печати версию логотипа; добавить класс к изображению; создать стиль класса в главной таблице стилей со свойством `display`, которому установлено значение `none`; а затем задать свойству `display` того же класса значение `block` в таблице стилей для печати — и дело в шляпе! Логотип появится только при печати.

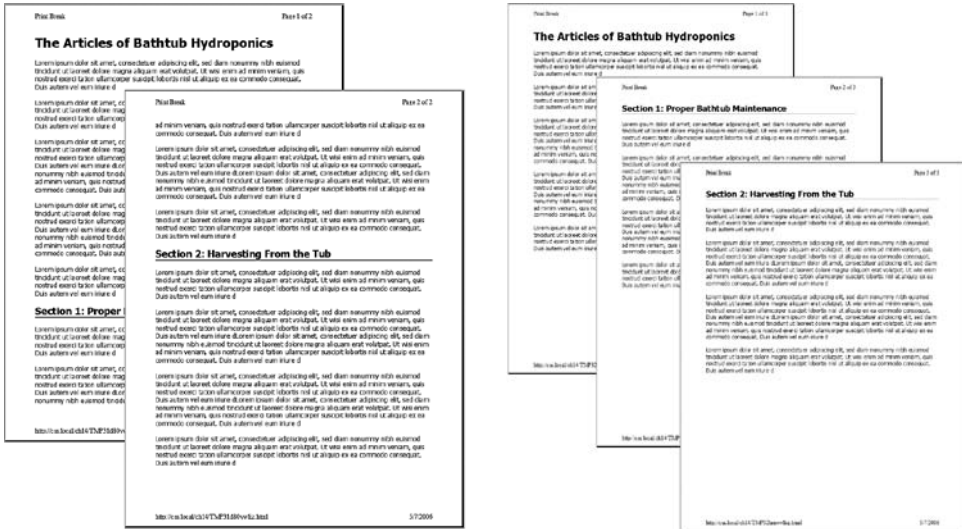
## Добавление разрывов страницы для печати

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](http://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 указывает браузеру вставить разрыв страницы перед данным



стилем. Скажем, вы хотите, чтобы определенные тексты всегда появлялись наверху страницы, как, например, названия различных разделов длинного документа (рис. 16.3). 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.



**Рис. 16.3.** При печати страницы браузер разделяет ее содержимое на множество страниц, чтобы вместить на каждой как можно больше текста (а). Если же вы хотите, чтобы разрывы страниц находились в более логичных местах, используйте свойство `page-break-before` (б)

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 странице. 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 использования этих свойств в своих интересах. Создайте два класса стилей, названных, например, `.break_after` и `.break_before`:

```
.break_before { page-break-before: always; }
.break_after { page-break-after: always; }
```

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">`.

## 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, рассказываете в конце гл. 2. Файлы для этого примера находятся в папке 16.

### 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 16.

У веб-страницы плавающая разметка, состоящая из нескольких тегов `<div>` (рис. 16.4). 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 боковой панели — пустая трата чернил, так что ваша таблица стилей для печати должна скрывать эти части страницы.



**Рис. 16.4.** Макет страницы позволяет управлять размещением элементов. При печати страницы лучше, чтобы некоторые элементы вовсе не появлялись. Баннер, навигационная панель и боковая панель не дадут никакой полезной информации в распечатанном документе

2. В текстовом редакторе создайте новый файл, названный `print.css`, и сохраните его в папке **16** 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 боковую панель:

```
.sidebar, .navWrapper, .footerNav {
  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 **16**.

У данной 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" 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. раздел «Особенности механизма каскадности: какие стили имеют преимущество» гл. 5 для получения большей 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, выполните команду **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 правая боковая панель и навигационные элементы исчезли. Но дизайн все равно еще не впечатляет. Главное содержимое не заполняет 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. 16.4 вы можете видеть, что страница разделена на несколько зон, каждая из которых создается собственным отдельным тегом `<div>`. Стили `.mainWrapper` и `.footer` центрируют области с основным содержимым и с нижним колонтитулом и устанавливают для них ширину 900 пикселей. 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 пикселям, для основного текста и нижнего колонтитула в файле `main.css` и оставляет точную ширину для браузера. Значение `auto` просто позволяет тегу `<div>` 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 `<div>` 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 правую боковую панель, для позиционирования которой 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 могут проявиться различные проблемы. Как мы обсуждали ранее, плавающие элементы могут внезапно выскочить за пределы их блока-контейнера, не позволяя фону 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 мы 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 ваши фоны были созданы просто для украшения и станут пустой тратой чернил, окажите услугу своим посетителям и отключите их.

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** (О нас), 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 `<div>`. Другими словами, она будет такой же, как и высота логотипа внутри него. 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 пикселях 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 подразделе 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 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» раздела «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, поскольку в стиле `p` из шага 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` (Нажмите здесь, чтобы узнать больше) напечатается как `Click here to found out more (http://www.outmore.com/)` (Нажмите здесь, чтобы узнать больше (<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 существенное улучшение для пользователей, которые распечатывают ваш сайт из браузеров Internet Explorer 8, 9 и 10, а также Chrome, Firefox и Safari.

1. Добавьте еще один, последний стиль к таблице стилей `print.css`:

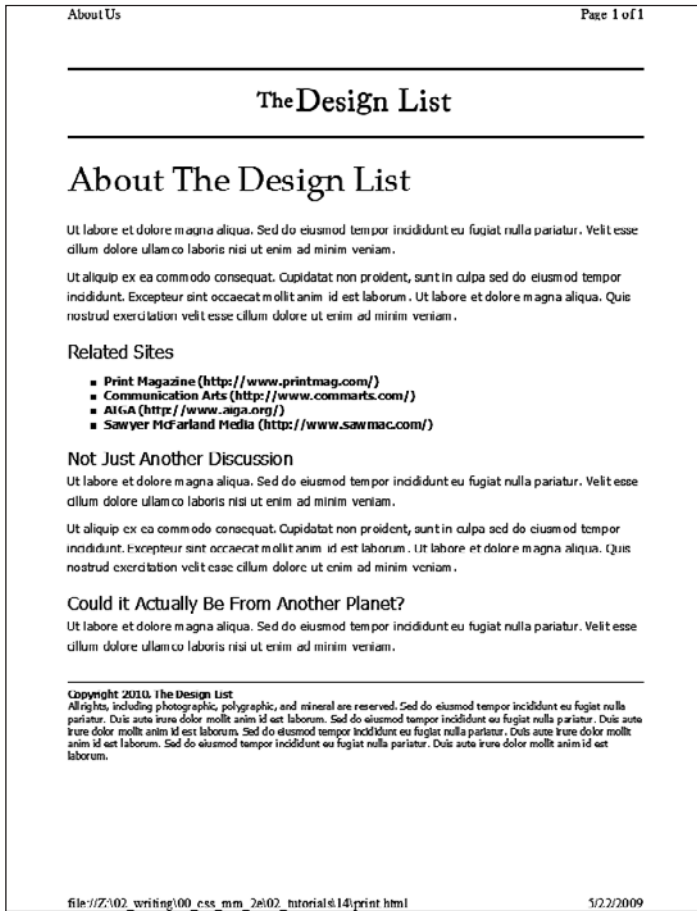
```
a:after {
  content: " (" attr(href) ) ";
}
```

На 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. 16.5, — простой, содержащей только факты.

Конечную версию страницы вы найдете в папке `16_finished`.



**Рис. 16.5.** 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, тратящие впустую чернила



# 17 Совершенствуем навыки работы с CSS

На данный момент мы рассмотрели большинство принципов каскадных таблиц стилей. С добавлением разметки, основанной на CSS, вы сможете 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.

## Добавление комментариев

Если приходится редактировать таблицу стилей через недели, месяцы или даже годы после того, как она была создана, может возникнуть вопрос: «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. Вам не нужно комментировать в своих таблицах стилей абсолютно все, в конечном счете большинство свойств, таких как `color`, `fontfamily`, `border-color` и т. д., говорят сами за себя. Но вполне резонно будет добавить комментарий для стиля, когда сразу не поймешь, что именно делает тот или иной стиль или свойство. Например, вы можете сбросить блочную модель CSS, чтобы ширина и высота элемента вычислялись с учетом границ и отступов:

```
/* {  
  -moz-box-sizing: border-box;  
  box-sizing: border-box;  
} */
```

В то время когда вы писали стиль, вы знали, что он делает, но будете ли вы так же хорошо помнить это три месяца спустя? А что, если кому-нибудь, кто не знаком

с этим приемом, понадобится отредактировать ваш код CSS? 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:

```
/* Сброс для всех элементов вычисления размеров
   в соответствии с моделью блоков */
* {
  -moz-box-sizing: border-box;
  box-sizing: border-box;
}
```

Вы можете оставить обширный комментарий, который может 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 предоставления полезной вводной информации, которая позволит отслеживать версию сайта или таблицы стилей, добавлять информацию об авторском праве и идентифицировать вас как владельца CSS-кода.

```

@charset "UTF-8";

/*
-----
CSS:      DevDays
Created:   May 2009
Author:    Mike Kus
URL:       http://www.carsonified.com
URL:       http://www.thethingswemake.co.uk
-----
*/

/*
Copyright (c) 2009, Yahoo! Inc. All rights reserved.
Code licensed under the BSD License:
http://developer.yahoo.net/yui/license.txt
version: 2.7.0
*/
html{color:#000;background:#FFF;}body,div,dl,dt,dd,ul,ol,li,h1,h2,h3,h4,h5
,h6,pre,code,form,fieldset,legend,input,button,textarea,p,blockquote,th,td
{margin:0;padding:0;}table{border-collapse:collapse;border-
spacing:0;}fieldset,img{border:0;}address,caption,cite,code,dfn,em,strong,
th,var,optgroup{font-style:inherit;font-weight:inherit;}del,ins{text-
decoration:none;}li{list-style:none;}caption,th{text-
align:left;h1,h2,h3,h4,h5,h6{font-size:100%;font-

```

Рис. 17.1. 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, вы 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 решат различные проблемы в будущем.

## Давайте стилям понятные имена

Вы уже изучили технические аспекты именования различных типов селекторов — селекторы класса начинаются с выражения `a.`, что позволяет идентифицировать стили как классы, 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.

Следует также избегать имен с указанием конкретных размеров, например `.font20px`. Сегодня этот шрифт может иметь размер 20 пикселей, а завтра, может быть, вы дадите ему размер 24 пикселя или перейдете от пикселей к `em` или процентам. Лучше, может быть, воспользоваться селектором тега: примените размер шрифта к тегу `h2` или к тегу `p`, или даже воспользуйтесь селектором потомков вроде `.sidebar1 p`.

- **Не используйте названия, основанные на месте расположения элемента.** По той же самой причине, по которой нужно избегать 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 левую боковую панель вправо, вверх или даже вниз страницы. И внезапно название `#leftSidebar` перестанет иметь какой-либо смысл вообще. Названия, более соответствующие назначению этой боковой панели, — `.news`, `.events`, `.secondaryContent`, `.mainNav` — идентифицируют ее 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, применение 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.

## Используйте несколько классов, чтобы сэкономить время

Часто два и 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 переместиться вправо и иметь левое поле.

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!

### ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

#### Препроцессоры CSS

Если вы хотите решать задачи быстро и эффективно, присмотритесь к препроцессорам CSS. Препроцессоры являются инструментами, получающими код и вырабатывающими из него конечный продукт. Что касается CSS, препроцессор позволяет создавать CSS в некой сокращенной версии, а затем вырабатывать из него конечный CSS-файл, помещаемый на ваш веб-сервер.

Применение препроцессоров CSS имеет ряд преимуществ.

- **CSS-переменные.** Препроцессоры позволяют определять переменные — своеобразные

хранилища информации — и использовать их в качестве значений в вашем коде CSS. Предположим, например, что у вас есть синий цвет, который вы собираетесь применять по всему сайту. Обычно цвет добавляется в CSS в виде шестнадцатеричного числа или RGB-значения. Когда нужно изменить этот цвет, приходится искать все экземпляры шестнадцатеричного значения и вносить в них изменения. С помощью препроцессора можно определить этот цвет всего один раз, в начале таблицы стилей. Если нужно изменить цвет, достаточно изменить значение переменной, и цвет обновится по всей таблице стилей.

## ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

- **Упрощение использования свойств CSS3.** Как уже ранее говорилось, при добавлении новых CSS-свойств зачастую приходится использовать префиксы производителей. Это сопряжено с написанием четырех и более строк кода для одного и того же свойства, например для `linear-gradient`. Большинство препроцессоров CSS могут создавать код за вас. Обычно вам остается только создать обычное CSS-свойство — `background image: linear-gradient(red, blue);`, а препроцессор добавит весь дополнительный код для работы с различными браузерами
- **Объединение таблиц стилей и уменьшение количества файлов.** Как уже говорилось, зачастую удобнее разбить ваш CSS-код на несколько таблиц стилей. В этом случае код CSS хранится в отдельных, легко управляемых файлах взаимосвязанных стилей. Но с точки зрения производительности веб-сервера лучше было бы отправить как можно меньше файлов. Препроцессор CSS позволяет хранить ваш код CSS в разных файлах, а затем объединить их в одну внешнюю таблицу стилей, которую смогут использовать ваши веб-страницы. Более того, вы можете выбросить все ненужное, CSS-

комментарии и другие символы, которые не нужны для работы CSS-кода и неоправданно увеличивают размер ваших CSS-файлов.

Двумя наиболее популярными препроцессорами CSS являются LESS (<http://lesscss.org>) и SASS (<http://sass-lang.com>). Но в их использовании есть два недостатка. Во-первых, предполагается, что вы уже достаточно знаете CSS. Если вы не слишком хорошо разбираетесь в селекторах и свойствах CSS, изучение порой не совсем обычного синтаксиса LESS и SASS может вас только запутать.

Во-вторых, оба препроцессора требуют довольно непростой установки на компьютер. Они представляют собой небольшие компьютерные программы, запускаемые при выполнении своих задач обработки в фоновом режиме. Однако эти ограничения можно обойти путем использования автономной программы вроде бесплатной Scout (<http://mhs.github.com/scout-app/>) или коммерческого приложения Compass.app (<http://compass.handlino.com>). Обе эти программы работают как на компьютерах с Windows, так и на Mac, и могут конвертировать файлы LESS и SASS в обычные прежние внешние таблицы стилей.

Рассмотрим страницы, изображенные на рис. 17.2. 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 на удивление немногие разработчики, — создание *нескольких классов*, применяемых к одному и тому же тегу. Это просто означает, что, когда вы применяете 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`.

Скажем, вы хотите использовать один и тот же стиль границы для группы изображений, но одну их часть желаете поместить слева, а другую — справа. Эту задачу можно решить 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.

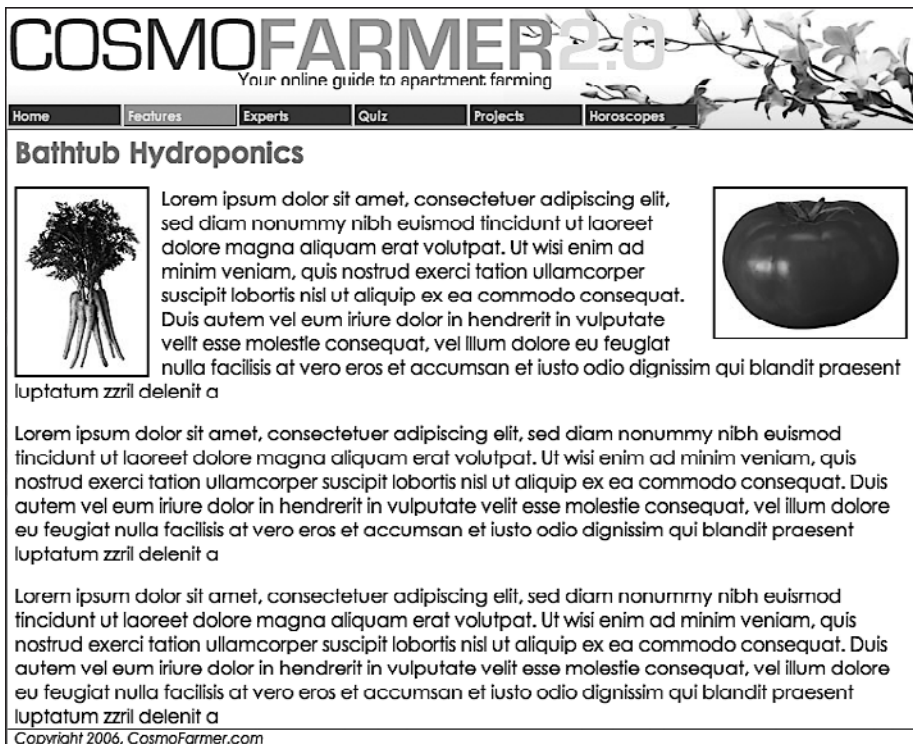


Рис. 17.2. Две фотографии с одним стилем класса

- Создайте два дополнительных класса стиля: один — для изображений, передвигаемых влево, а другой — для передвигаемых вправо, например `.floatLeft` и `.floatRight`.
- Примените оба класса к каждому тегу:

```

```

или

```

```

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 боковой панели, которое передвинет ее вправо, добавит интересные фоновые изображения и будет включать тщательно разработанное оформление. Вы можете использовать этот стиль повсюду на вашем сайте, но ширина боковой панели будет изменяться в различных случаях. Возможно, она равна 33 % на одних страницах и 25 % — на других. В таком случае создайте отдельный класс стиля (такой как `.sidebar`) с основным форматированием для боковой панели и отдельные классы только для определения ширины боковой панели, например `.w33per` и `.w25per`. Затем примените по два класса к каждой боковой панели: `<div class = "sidebar w33per">`.

## 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 с 500 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.** Независимо от того, какой подход вы предпочтете, убедитесь, что используете комментарии CSS, чтобы разделить каждую группу стилей. Скажем, вы собрали все стили, которые управляют 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. Если вы ищете интересную идею, посмотрите, как таблицы стилей комментируются на следующих сайтах: [www.wired.com](http://www.wired.com), [www.mezzoblue.com](http://www.mezzoblue.com) и <http://keikibulls.com>.

---

## 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](http://www.stopdesign.com/log/2005/05/03/css-tip-flags.html).

---

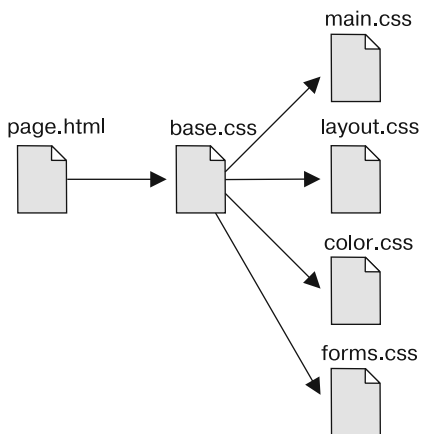


## Используйте несколько таблиц стилей

Из гл. 16 вы узнали, что 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.

Когда 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. Кроме того, чем больше внешних CSS-файлов, тем на большее количество запросов должен отвечать ваш веб-сервер, что становится одной из причин снижения скорости работы веб-сайта.

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 17.3 иллюстрирует этот подход.



**Рис. 17.3.** 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`

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. 17.3, *справа*).

#### ПРИМЕЧАНИЕ

Это всего лишь рекомендации. Организуйте ваши стили и таблицы стилей так, как вам кажется наиболее логичным и как будет работать наилучшим образом для вас. Для получения дополнительной информации можете прочитать статью о модульном CSS-дизайне на сайте [www.contentwithstyle.co.uk/content/modular-css](http://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.

3. Наконец, присоедините таблицу стилей из шага 2 к HTML-страницам вашего сайта, используя тег `<link>` или правило `@import` (о том, как использовать эти методы, читайте в разделе «Внешние таблицы стилей» гл. 2). Например:

```
<link rel="stylesheet" href="base.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. Не попадитесь с кэшированием» в разделе «Понимание таблиц стилей» 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 изме-

ните 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 оплату за хостинг. Тем не менее кэш может стать головной болью, когда вам необходимо обновить внешний вид сайта. Например, если все его страницы ссылаются на внешнюю таблицу стилей 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" 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" href="main.css?v=1">
```

Затем, когда вы обновите файл `main.css`, можно изменить тег `<link>` на следующий:

```
<link rel="stylesheet" href="main.css?v=2">
```

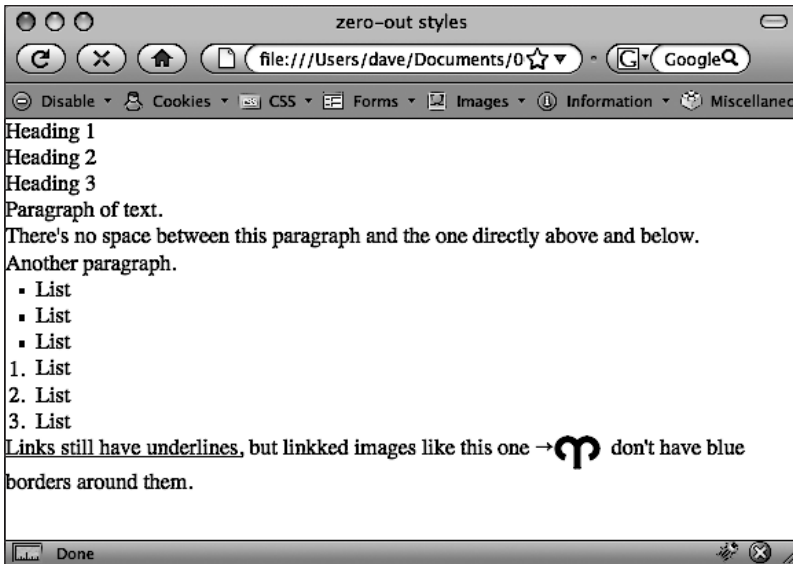
Браузер посчитает таблицу стилей отличной от сохраненной в кэше версии файла `main.css` и загрузит файл с веб-сервера. На самом деле `?v=1` ничего не делает и не влияет на то, как работает веб-сервер. Это просто способ заставить браузер перезагрузить файл.

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-caching-files-php-function>.

## 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.

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. 17.4). Все, что нужно сделать, — настроить в начале своей таблицы стилей некоторые основные стили, которые убирают ненужное форматирование.



**Рис. 17.4.** 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, например, 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.
- **Примите меры к тому, чтобы элементы HTML5 отображались в виде блоков.** Новые элементы HTML5, такие как `article`, `aside` и `section`, являются такими же блочными элементами, как заголовки и `div`-контейнеры. Но браузеры, не различающие эти элементы, могут и не показывать их в виде блоков. Поэтому лучше добавить для этих элементов стиль `display: block`. Кроме того, нужно воспользоваться средством HTML5 shiv (<http://code.google.com/p/html5shiv/>), как описано во врезке «Обходной прием. Как заставить Internet Explorer 8 понимать HTML5» в разделе «Дополнительные теги в HTML5» гл. 1, чтобы заставить

Internet Explorer 8 и более ранние версии разбираться в коде CSS, применяемом для HTML5-тегов.

- **Установите единую высоту строк.** У браузеров могут быть незначительные различия в значениях высоты строк, используемых по умолчанию. Задав коэффициент в теге `<body>` — `body { line-height: 1.2; }`, — можно обеспечить применение браузерами одинаковой высоты строк. Значение 1.2 эквивалентно 120 % от размеров текста тегов. Можно, разумеется, изменить это значение в соответствии с вашими дизайнерскими предпочтениями.
- **Улучшайте границы таблиц и создавайте согласующиеся ячейки.** Применяя границы к ячейкам таблицы, вы создаете промежутки 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.** Некоторые 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, Internet Explorer 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:

```
/* сброс стилей браузера */
```

```
html, body, div, span, object, iframe, h1, h2, h3, h4, h5, h6, p,
blockquote, pre, a, abbr, acronym, address, big, cite, code, del, dfn,
em, img, ins, kbd, q, s, samp, small, strike, strong, sub, sup, tt,
var, b, u, i, center, dl, dt, dd, ol, ul, li, fieldset, form, label,
legend,
```

```
table, caption, tbody, tfoot, thead, tr, th, td, article, aside, canvas,
details, embed,
```

```
figure, figcaption, footer, header, hgroup, menu, nav, output, ruby,
section, summary,
```

```
time, mark, audio, video {
  margin: 0;
  padding: 0;
  border: 0;
  font-size: 100%;
  vertical-align: baseline;
}

article, aside, details, figcaption, figure, footer, header, hgroup,
menu, nav, section {
  display: block;
}

body {
  line-height: 1.2;
}

table {
  border-collapse: collapse;
  border-spacing: 0;
}

ol {
  padding-left: 1.4em;
  list-style: decimal;
}

ul {
  padding-left: 1.4em;
  list-style: square;
}

blockquote, q {
  quotes: none;
}

blockquote:before, blockquote:after,
q:before, q:after {
  content: '';
  content: none;
}

/* завершение сброса стилей браузера */
```

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 17 обучающих материалов. Просто скопируйте код из этого файла в свои таблицы стилей.

---

#### ПРИМЕЧАНИЕ

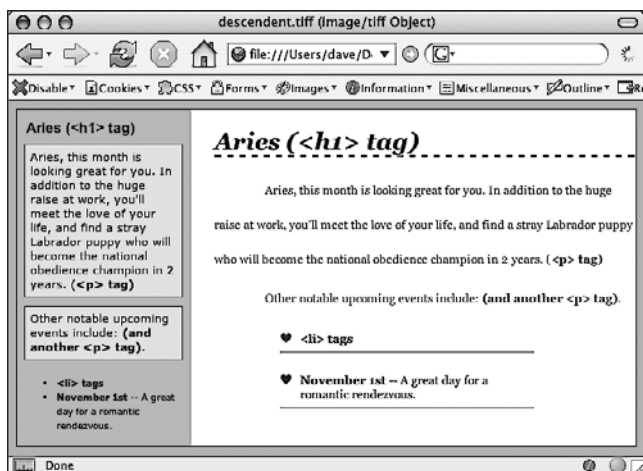
Некоторые разработчики используют при решении неоднообразия стилей браузеров другой подход. Проект `Normalize.css` (<http://nicolasgallagher.com/about-normalize-css/>) предназначен для предоставления однородных исходных стилей, сохраняя при этом основные различия в HTML-тегах. Например, вместо приведения к единому размеру заголовков и абзацев `Normalize.css` сохраняет различные размеры заголовков. В проект также включено множество других стилей, предназначенных для устранения недостатков, присущих некоторым браузерам.

---

## 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 задать, что только один абзац будет иметь внешний вид, определенный в стилевом классе `.intro`. Поскольку добавить класс или идентификатор к тегу не составит никакого труда, многие разработчики взяли моду добавлять их ко *всем элементам* (или почти ко всем), что не может не вызывать беспокойства. У профессионалов есть даже диагноз этому заболеванию — *тяга к классификации*. Добавление класса к каждому тегу — это не только лишняя трата времени, но и замедление загрузки HTML. К тому же есть лучший способ управления тегами, не прибегая к слишком большому количеству классов или идентификаторов, — использование селекторов потомков.

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 стили к каждой из этих ссылок. Другими словами, нам нужна возможность форматирования одинаковых HTML-элементов по-разному, в зависимости от того, где они располагаются, а это как раз то, что предлагают селекторы потомков (рис. 17.5).



**Рис. 17.5.** Один и тот же HTML-код был вставлен в левую боковую панель и в область справа. При использовании селекторов потомков идентичные HTML-теги (<h1>, <p>, <ul> и <li>) формируются по-разному, в зависимости от их расположения на странице

## Разбивайте страницы на разделы

Одним из самых главных элементов эффективного использования селекторов потомков является применение тега <div>. Поскольку этот тег позволяет создавать логические разделы на странице, вы можете применять его для идентификации различных элементов разметки, таких как баннер, боковая панель, колонка текста и т. д. Как мы обсуждали во врезке «Информация для новичков. HTML-теги <div> и <span>» в разделе «Селекторы классов: точное управление» гл. 3, можно организовать содержимое страницы в блоки, задавая для каждого HTML-тега <div>.

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 <div> идентифицируйте его для CSS с помощью атрибута class (<div class = "pullQuote">). Если вы хотите вставить один и тот же тип элемента разметки несколько раз на странице (например, несколько врезок в отдельной истории), используйте класс.

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 `<div>`, 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 `<div>` и просто добавить класс к неупорядоченному списку: `<ul class="storyNav">`. Можно также заключить тег `<ul>` в HTML5-тег `<nav>`, и применить класс к этому тегу.

---

Как только вы идентифицируете каждый тег `<div>` на странице, будет очень легко использовать селектор потомков, нацеленный на элементы внутри конкретного тега `<div>`. 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 `<div>` — 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>`, появляющийся внутри `<div>`, 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 конкретной 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. Простой способ разграничения тегов `<h1>` для домашней страницы — добавить атрибуты `class` к ее тегу `<body>`:

```
<body class="home">
```

Вы можете разработать стиль тега `<h1>` домашней страницы, используя селектор потомков: `.home h1`. Таким же образом можно создавать различное форматирование для любого тега конкретной страницы вашего сайта (рис. 17.6). 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">`. На рис. 17.6, когда для тега указан класс `home`, выделяется кнопка **Home** (Домой) (*вверху*). Измените класс на `about`, и выделится кнопка **About Us** (О нас) (*внизу*).

## 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. 17.6. Если страница находится в разделе новостей вашего сайта, вы можете выделить кнопку **News** (Новости), 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 class = "home">`. Сделайте то же самое для каждого раздела, чтобы у страниц в разделе новостей был следующий код: `<body class = "news">`.
2. Добавьте навигационную панель на страницу. Пошаговые инструкции по созданию навигационной панели вы найдете в разделе «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" class = "homeLink">Home</a>`. Атрибут `class` позволяет идентифицировать ссылку как указывающую на домашнюю страницу. Повторите это для других ссылок: `<href = "/news/" class = "newsLink">News</a>` и т. д.

На данный момент у вас достаточно информации в HTML-коде, чтобы уникальным образом отформатировать ссылку каждого раздела, используя CSS. В этом примере известно, что ссылка на домашнюю страницу вложена в тег `<body>` с атрибутом `class`, равным `home`, только на домашней странице.

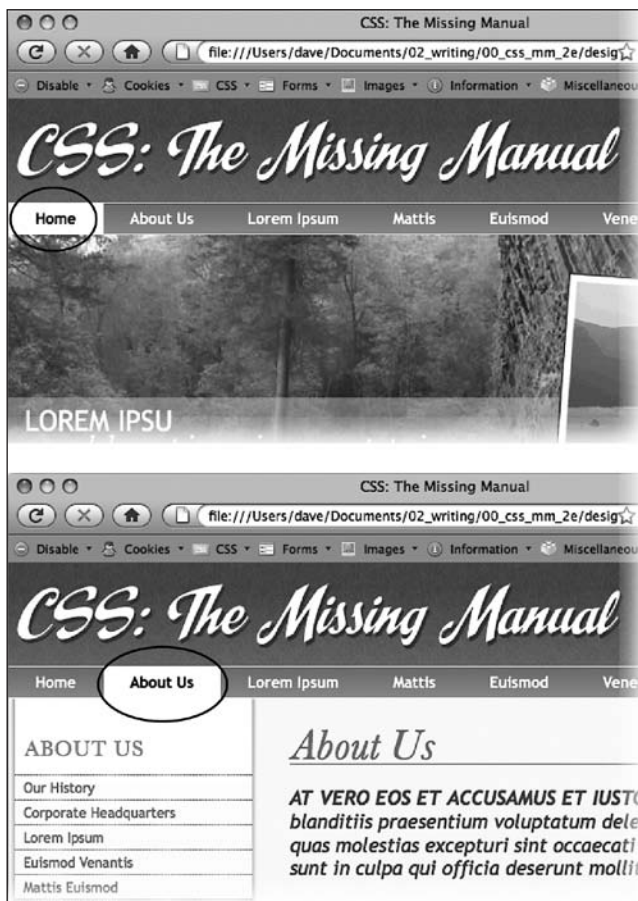
4. Создайте селектор потомков, который позволит отформатировать ссылку каждого раздела по-своему. Это форматирование применяется, когда ссылка находится внутри страницы для этого раздела. Для домашней страницы в этом примере селектор потомков должен быть примерно таким:

```
.home .homeLink
```

Селектор форматирует ссылку `.homeLink`, только когда она находится внутри другого тега с классом, равным `.home`. В большинстве случаев нужно, чтобы выделенная кнопка выглядела одинаково для каждого раздела сайта, так что лучше использовать групповой селектор (см. раздел «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 .homeLink,
.news .newLink,
.articles .articlesLink,
.links .linksLink {
  background-color: #FBFEF9;
}
```



**Рис. 17.6.** Используя селекторы потомков, можно выделить кнопку на навигационной панели, просто изменив атрибут class тега <body>

## 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. раздел «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 форматирован совершенно по-разному. Почти как по волшебству.

## ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

### Объектно-ориентированный код CSS

Объектно-ориентированный код CSS представляет еще один подход к организации и использованию CSS. Этот термин был придуман экспертом CSS Николь Салливан, а подход был разработан в ответ на сложность очень больших сайтов с широким разнообразием HTML-структур. На больших сайтах может быть очень разный HTML, содержащий сходные типы информации. Например, в одном месте может использоваться маркированный список для отображения имени, информации и фотографии для перечня контактов. А на другой странице может применяться тег `<article>` для каждого контакта. Предположим, что нужно отформатировать эти элементы одинаково, поскольку у них действительно одинаковый тип информации. При использовании CSS-стилей, зависящих от этого HTML, можно столкнуться либо с дублированием стилей, либо с такими групповыми селекторами:

```
article img, li img {
  /* сюда помещается форматирование */
}
```

Объектно-ориентированный код CSS рекомендует отказаться от привязки кода CSS к реальным HTML-элементам и использовать вместо них классы. То есть вы по всему коду HTML применяете классы, а затем задействуете для форматирования CSS-стили классов. При этом неважно, какие HTML-теги используются. Если они применяют одни и те же классы, у них будет одинаковый внешний вид.

Возможно, это похоже на ранее упоминавшуюся проблему «тяги к классификации», и по смыслу это так и есть. Объектно-ориентированный код CSS требует добавления большого количества атрибутов `class` к вашему HTML. Если используется такая система управления контентом, как WordPress, то объем работы может быть не столь существенным — имена классов можно добавлять к файлам шаблонов. Но если каждая страница создается вручную, такой подход может добавить большой объем работы.

Краткое введение в объектно-ориентированный код CSS находится по адресу <http://coding.smashingmagazine.com/2011/12/12/an-introduction-to-object-oriented-css-oocss/>. А проект OOCSS можно найти по адресу <https://github.com/stubbornella/oocss>.

Еще один подобный подход от Джонатана Снука (Jonathon Snoonk), который называется масштабируемой и модульной архитектурой для CSS — Scalable and Modular Architecture for CSS (или SMACSS), является простым руководством для создания повторно используемых CSS-компонентов. Дополнительные сведения об этом подходе можно получить по адресу <http://smacss.com>. А по адресу <http://tv.adobe.com/watch/adc-presents/smacss-introduction-to-a-scalable-and-modular-architecture-for-css/> можно также найти множество видеоподборок с дискуссиями, касающимися SMACSS.

## Различный код CSS для Internet Explorer

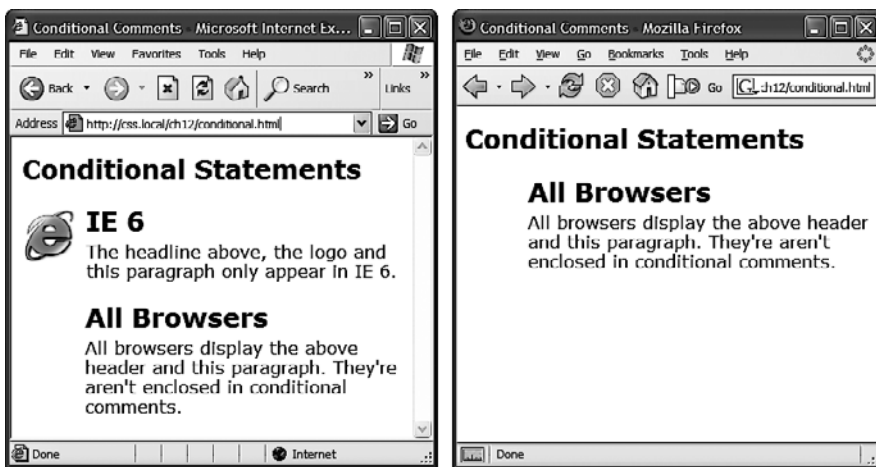
Браузеры не всегда ведут себя предсказуемо. Такие браузеры, как Safari, Firefox и Internet Explorer 10, достаточно хорошо обрабатывают CSS-код и последователь-

но и предсказуемо отображают основанные на CSS веб-страницы. Однако сделать так, чтобы ваш дизайн работал в Internet Explorer 7, 8 и даже 9, куда сложнее. Хотя эти браузеры по современным стандартам уже устарели, они все еще достаточно популярны.

Нередко создаются стили, предназначенные для устранения недостатков конкретных версий Internet Explorer. Ранее рассмотренное свойство `zoom`, например, решает проблемы, присущие Internet Explorer 7 и более ранним версиям. Internet Explorer 8 и более ранние версии не в состоянии правильно отформатировать HTML5-теги. Одно из решений заключается в отправке определенного содержания, такого как таблицы стилей, JavaScript и даже HTML, только для Internet Explorer. Фактически можно нацелиться на отдельные версии Internet Explorer для отправки разного кода для каждой из них.

Можно собирать стили только для Internet Explorer в отдельном месте — использовать возможности условных комментариев (рис. 17.7). Это изобретение Microsoft предоставляет способ вставки HTML-кода только для Internet Explorer. Другие браузеры принимают этот код за комментарий и игнорируют его.

Условные комментарии могут даже предназначаться для различных версий браузера. Вы можете поместить все стили только для Internet Explorer 7 в отдельной внешней таблице стилей (такой как `Internet Explorer7_styles.css`) и использовать условные комментарии, чтобы связывать ее только с этим браузером. Применяя данный подход, вы легко сможете удалить данные стили, когда Internet Explorer 7 наконец повторит судьбу динозавров. Нужно будет просто удалить внешнюю таблицу стилей.



**Рис. 17.7.** 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)

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]>
```

Сюда помещается HTML-код, применяемый только для Internet Explorer.

```
<![endif]-->
```

<!--[if Internet Explorer]> — непосредственно само условие. Оно переводится как «если этот браузер — Internet Explorer, то обрабатываем следующей 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.

**Нацеливание на различные версии Internet Explorer.** Условные комментарии позволяют определять, к браузеру какой версии относится таблица стилей. Скажем, вы хотите загружать специфическую таблицу стилей только для Internet Explorer 7. Добавьте следующие условные комментарии в начале вашей веб-страницы:

```
<!--[if IE 7]>
```

```
<link href="IE7_styles.css" rel="stylesheet" >
```

```
<![endif]-->
```

Или, используя правило @import:

```
<!--[if IE 7]>
```

```
<style>
```

```
@import url(IE7_styles.css)
```

```
</style>
```

```
<![endif]-->
```

Этот код нацелен конкретно на Internet Explorer 7 и не будет применяться к любой другой более ранней версии. Но иногда нужно включить и другие версии. Например, можно создать таблицу стилей, применяемую к Internet Explorer 6, 7 и 8. В таком случае можно воспользоваться ключевым словом `lte`, которое означает «less than or equal — меньше чем или равно», следовательно, `lte IE 8` означает «если это Internet Explorer 8 или более ранняя версия»:

```
<!--[if lte IE 8]>
```

```
<link href="IE_styles.css" rel="stylesheet">
```

```
<![endif]-->
```

Условные комментарии нужно добавлять после любых ссылок на таблицы 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 8 и таблицу стилей для Internet Explorer версий 7 и ниже.

```
<link href="global_styles.css" rel="stylesheet" type="text/css" />
<!--[if IE 8]>
<link href="IE8_styles.css" rel="stylesheet" type="text/css" />
<![endif]-->
<!--[if lte IE 7]>
<link href="IE_old_styles.css" rel="stylesheet" type="text/css" />
<![endif]-->
```

---

**ПРИМЕЧАНИЕ**

Для получения большей информации об условных комментариях Internet Explorer зайдите на сайт [http://msdn.microsoft.com/en-us/library/ms537512\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/ms537512(VS.85).aspx).

---

# Приложения

- Приложение 1.** Справочник свойств CSS
- Приложение 2.** Информационные ресурсы, касающиеся 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, я собрал в этом приложении 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, а также ряд наиболее полезных и поддерживаемых браузерами свойств CSS3.

---

## ПРИМЕЧАНИЕ

Самая последняя CSS-спецификация имеет довольно большой объем. Фактически, чтобы лучше справиться с разрастанием CSS, консорциум W3C разбил CSS на множество модулей — каждый модуль описывает конкретное свойство или набор родственных свойств. Чтобы получить полную информацию из первых рук, просмотрите самые последние спецификации CSS от World Wide Web Consortium по адресу [www.w3.org/Style/CSS/current-work](http://www.w3.org/Style/CSS/current-work).

---

## 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 CSS3 обещает предложить их еще больше в будущем (<http://www.w3.org/TR/css3-color/>). Например, рассмотренная в гл. 6 цветовая схема RGBA.



**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, применяя 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.
- **Целочисленные значения.** И наконец, для задания RGB-цвета можно использовать целочисленные значения. Формат такой же, как и для процентных значений, но для указания каждого цвета применяется число от 0 до 255: `rgb(255, 0, 33)`.
- **RGBA.** RGBA добавляет к смешанным цветам прозрачность. То есть можно сделать цвет, через который просматриваются расположенные ниже фоновые цвета, изображения или текст. Последним к цвету RGB добавляется значение от 0 (полная невидимость) до 1 (полная непрозрачность). Для задания цветовых составляющих можно использовать либо процентные, либо десятичные значения, а для прозрачности можно применять только десятичные значения. Например, оба следующих объявления создают светло-серый цвет, с 50-процентной прозрачностью:

```
rgba(50%,50%,50%,.5)
rgba(122,122,122,.5)
```

- **HSL означает hue — тон, saturation — насыщенность и lightness — освещенность (последняя буква иногда расшифровывается как luminance — светимость).** Это еще один способ задания цвета. Он не поддерживается в Internet Explorer 8 и более ранних версиях, но работает во всех остальных браузерах. Если вы привыкли использовать RGB- или Hex-цвета, синтаксис HSL может показаться немного непривычным. Вот как выглядит задание ярко-красного цвета:

```
hsl(0, 100%, 50%);
```

Первое число (hue) является углом от 0 до 360°, представляющим цветовой спектр на круге оттенков, где 0, и 360° представляют красный, а 180° — зеле-

ный цвет. Второе число задает насыщенность, или степень чистоты цвета, и выражается процентами от 0 (тусклый серый цвет) до 100 % (яркий и живой цвет). Последнее значение задает осветленность и также выражается процентами от 0 (полностью черный цвет) до 100 % (полностью белый цвет). Как и у RGB-цвета, существует версия HSL, поддерживающая прозрачность. Она называется HSLA: `hsla(0, 100%, 50%, .5)`.

Неважно, какой метод вы используете, — все они работают. Для последовательности вы должны выбрать один способ определения RGB-значения и придерживаться этого способа. Но проще и удобнее может быть применение ключевых слов, например `white` и `black`. В операционных системах Windows и Mac есть средства выбора цвета, которые позволяют подобрать идеально подходящий цвет из всей палитры цветов, а затем показывают вам его RGB-значение. В качестве альтернативы можно применять бесплатное средство выбора цвета с сайта [www.ficml.org/jemimap/style/color/wheel.html](http://www.ficml.org/jemimap/style/color/wheel.html) или более совершенное средство, позволяющее создавать и сохранять палитру цветов, которое находится на сайте <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, цитеро, точки, сантиметры, миллиметры, единицы измерения `em` и `ex`, пиксели и проценты.

Однако при всем многообразии единиц измерения многие из них не относятся к миру экранного отображения по причинам, обсуждаемым в разделе «Установка размера шрифта» гл. 6. На самом деле вам стоит задуматься только о трех: пикселях, `em` и процентах.

**Пиксели.** Пиксел — это одна точка на экране компьютера. Пиксели 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 пользователи устанавливают для своих мониторов различные разрешения —  $800 \times 600$ ,  $1024 \times 768$ ,  $1600 \times 1200$  и др., — 72 пиксела могут занять 1 дюйм на одном мониторе и всего 0,5 дюйма на другом. Однако пиксели дают вам наиболее последовательный контроль над отображением.

---

### ПРИМЕЧАНИЕ

Есть только один недостаток в использовании пикселей: пользователи Internet Explorer версий 6 или ниже не могут изменить размеры элемента, если они заданы в пикселях. Если ваш текст окажется слишком маленьким для кого-то, то посетитель будет не в состоянии увеличить его, чтобы сделать более приемлемым для чтения (см. раздел «Установка размера шрифта» гл. 6).

---

**Единица измерения `em`.** В типографике `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 пикселей. Однако любой может изменить эту базовую установку размера, так что 1 `em` будет равняться 16 пикселям в одном браузере и 24 пикселям — 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 пикселей в большинстве браузеров с основным размером шрифта 16 пикселей. 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 пикселей ( $16 \times 0,9 \times 0,9$ ). 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 на основании унаследованного значения текста. Скажем, общий размер шрифта для абзаца — 16 пикселей. Если вы создали стиль для отдельного абзаца и установили размер его шрифта равным 200 %, то этот текст отображается высотой 32 пикселя. Однако при применении к ширине проценты вычисляются на основе ширины строки или другого родительского элемента с заданной шириной.

## Ключевые слова

У многих свойств есть 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 изменяются 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 `<em>` и `<strong>` 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 данном случае теги `em` и `strong` отображают одно и то же значение `border`, как и их родительский тег `<p>`. Так, элементы `<em>` и `<strong>` абзаца получают 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 `<em>` и `<strong>` 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. раздел «Добавление фоновых изображений» 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. Это означает, что вы можете использовать абсолютный URL (например, `http://www.missingmanuals.com/images/tile.gif`), путь относительно корня (например, `/images/tile.gif`) или URL относительно документа (например, `.././images/tile.gif`). Прочитайте врезку «Информация для новичков. Типы URL» в разделе «Добавление фоновых изображений» 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. Задействование этой методики — быстрый способ создать всеохватывающий шрифт, цвет и т. д. для страницы или раздела.

### color (наследуется)

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 раздел «Выборка стилизуемых ссылок» гл. 9.

---

### font (наследуется)

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 (наследуется)

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.

**Значения:** названия шрифтов, разделенные запятыми. Когда название шрифта состоит из нескольких слов, между которыми стоит пробел, это название нужно указывать в кавычках. Последний перечисленный 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;.`

### ПРИМЕЧАНИЕ

---

Все браузеры позволяют указывать дополнительные шрифты, которые либо хранятся на вашем веб-сервере, либо загружаются браузерами ваших посетителей со службы веб-шрифтов. Подробное описание веб-шрифтов дано в разделе «Использование веб-шрифтов» гл. 6.

---

## font-size (наследуется)

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» раздела «Значения CSS» этого приложения), а также следующие ключевые слова: `xx-small, x-small, small, medium, large, x-large, xx-large, larger` и `smaller`; `medium` представляет 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 пикселям (берем в расчет то, что вы не корректировали базовый размер шрифта браузера); `x-small` — 10 пикселям, `small` — 13, `large` — 18, `x-large` — 24 и `xx-large` — 32 пикселям. Из-за неуверенности в том, как каждый браузер обрабатывает эти ключевые слова, многие дизайнеры используют вместо них пиксели, `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 (наследуется)

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 (наследуется)

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 (наследуется)

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 пиксели 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 (наследуется)

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. раздел «Форматирование абзацев текста» гл. 6).

**Значения:** большинство корректных размеров CSS (см. подраздел «Длины и размеры» раздела «Значения CSS» этого приложения), хотя `em`, пиксели и проценты наиболее распространены.

**Пример:** `line-height: 200%;`

## text-align (наследуется)

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 способов оформления, перечисляя значения (кроме `none`) через пробел.

**Пример:** `text-decoration: underline overline line-through;`

## text-indent (наследуется)

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. Пиксели 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. Чтобы первая строка выступала за левый край, 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-shadow (наследуется)

Позволяет добавлять любому тексту тень.

**Значения:** к тексту применяются два измеряемых значения (в `em` или пикселях) для горизонтального и вертикального смещения: значение, определяющее степень размытости тени, и значение цвета тени. Отрицательное значение для горизонтального смещения помещает тень слева от текста, а положительное значение — справа. Отрицательное значение для вертикального смещения помещает тень сверху от текста, а положительное значение — снизу. Каждое значение отделяется от предыдущего пробелом. Можно также добавить несколько теней, добавляя значения дополнительной тени через запятую.

**Примеры:**

```
text-shadow: -4px 4px 3px #999999;  
text-shadow: -4px 4px 3px #999999  
2px 3px 10px #000;
```

## 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` — бук-



вы, 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

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 ТМ, ® 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» раздела «Создание стилей для таблиц» гл. 11).

**Значения:** baseline, sub, super, top, text-top, middle, bottom, text-bottom, процентное или абсолютное значение (например, пиксели или em). Проценты вычисляются на основании значения line-height элемента.

**Примеры:**

```
vertical-align: top;  
vertical-align: -5px;  
vertical-align: 75%;.
```

## white-space

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, примените 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 (наследуется)

Работает, как свойство letter-spacing, но регулирует расстояние между словами, а не между буквами.

**Значения:** любая правильная единица измерения CSS, хотя em и пиксели 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 (наследуется)

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 покажет 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 (наследуется)

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 предоставит больше возможностей управления (см. раздел «Добавление фоновых изображений» гл. 8).

---

### list-style-position (наследуется)

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 (наследуется)

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`, `lower-alpha`, `upper-roman`, `lower-roman`, `lower-greek`, `none`.

**Пример:** `list-style-type: square;`

## Отступы, границы и поля

Следующие свойства управляют пространством вокруг элемента.

### box-shadow

Добавляет тень вокруг блочного элемента.

**Значения:** к блочному значению применяется необязательное значение `inset`, добавляющее тень внутри элемента. За ним следуют четыре измеряемых значения и цвет. Первые два измеряемых значения (в `em` или пикселах) задают горизонтальное и вертикальное смещения. Третье значение определяет степень размытости тени, Четвертое, необязательное значение указывает «расширение» тени, делая ее шире. Отрицательное значение для горизонтального смещения помещает тень слева от блока, а положительное значение — справа. Отрицательное значение для вертикального смещения помещает тень сверху от блока, а положительное значение — снизу. Каждое значение отделяется от предыдущего пробелом. Можно также добавить несколько теней, добавляя значения дополнительной тени через запятую.

#### Примеры:

```
box-shadow: -4px 4px 3px #999999;  
box-shadow: inset 4px 4px 3px 5px #999999;  
box-shadow: inset 4px 4px 3px 5px #999999  
            2px 2px 5px black;
```

### 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-radius

Скругляет углы элемента. Визуальный эффект проявляется, только если у элемента есть граница, цвет фона или изображение.

**Значения:** одно, два, три или четыре значения (в `px`, `em` или `%`), задающие размер радиуса окружности, рисуемой в углах блока элемента. Если предоставлено только одно значение, один и тот же размер скругленного угла применяется ко всем четырем углам. Если предоставлено два значения, первое определяет радиус скругления верхнего левого и нижнего правого углов, а второе — верхнего правого и нижнего левого углов. Если используются три значения, первое определяет радиус скругления верхнего левого угла, второе — верхнего правого и нижнего левого угла, а третье — нижнего правого угла. Если задействуются четыре значения, то каждое из них применяется по порядку к верхнему левому, верхнему правому, нижнему правому и нижнему левому углам. Для создания вместо углов в форме окружности углов в форме эллипса можно добавить `slash /`, после которого указать второе значение радиуса.

**Примеры:**

```
border-radius: .5em;  
border-radius: 15px 10px 25px 5px;  
border-radius: 15px / 5px;
```

**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, заданный 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`.

**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 также 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`.

**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 пиксели.

**Пример:** `border-width: 1px;`

У этого свойства есть также сокращенная запись, позволяющая присваивать различную толщину линии 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`.

**Пример:** `border-bottom-width: 3em;`

## **box-sizing**

Предписывает браузеру порядок измерения высоты и ширины элемента. Обычно браузеры для определения количества экранного пространства, занимаемого элементом, объединяют свойства `border`, `padding` и `width`. Такая система может привести к запутанной ситуации, когда для ширины элемента указывается значение 300 пикселей, но если у элемента также есть отступы и границы, браузер отображает элемент на экране в пространстве, превышающем по ширине 300 пикселей.

**Значения:** `content-box`, `padding-box` или `border-box`. Вариант `content-box` задает обычный порядок работы браузера. `padding-box` предписывает браузеру включать в расчет наряду со значениями `width` и `height` еще и значение `padding`. Вариант `border-box` заставляет браузер включать в расчет еще и значение `border`. Например, если у вас есть тег `<div>`, для которого установлена ширина 300 пикселей, отступы по 20 пикселей и граница толщиной 2 пиксела, браузер обычно покажет этот `div`-элемент в 344 пикселях экранного пространства ( $300 + 20 + 20 + 2 + 2$ ), но, если для свойства `box-sizing` установлено значение `padding-box`, браузер покажет этот `div`-элемент в 304 пикселях экранного пространства ( $300 + 2 + 2$ ), потому что отступы считаются частью этих 300 пикселей, а если установлено значение `border-box`, то браузеру будет просто предписано сделать `div`-элемент шириной 300 пикселей. Вариант `border-box` широко используется для упрощения отслеживания ширины элемента. К сожалению, Internet Explorer 7 и более ранние версии не понимают этого свойства.

**Пример:** `box-sizing: border-box;`

## 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 8-й 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`).

**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 пиксели.

**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).

**Значения:** любая корректная единица измерения CSS, такая как пиксели 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 боковой панелью 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. обучающий урок в гл. 7).

---

**Значения:** любая корректная единица измерения CSS, такая как пиксели 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;.`

## Фоны

CSS предоставляет несколько свойств для управления фоном элемента, включая изменение цвета фона, размещение изображения позади элемента и управление 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. раздел «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 наряду с остальным содержимым, 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 наряду с текстом. `Fixed` закрепляет изображение на месте.

**Пример:** `background-attachment: fixed;.`

## background-clip

Это свойство ограничивает область, в которой появляется фоновое изображение. Обычно фоновое изображение заполняет всю область элемента, включая его границы, отступы и содержимое. Но может быть необходимо, чтобы изображение появлялось только за областью отступов и не распространялось на границы. Это может пригодиться в случае использования точечных или пунктирных границ, чтобы изображение не появлялось в разрывах границы. Может также потребоваться исключить область отступов, чтобы размноженное изображение фона появлялось только в области содержимого, а в области отступов был сплошной цвет. В Internet Explorer 8 и более ранних версиях это свойство не поддерживается.

**Значения:** border-box, padding-box или content-box. Вариант border-box задает обычный метод, при котором изображение помещается под границей, отступами и содержимым. Вариант padding-box сохраняет фоновое изображение только в области отступов и содержимого, и оно не появляется под границей. Вариант content-box помещает фоновое изображение только в области содержимого, и оно не появляется ни в области отступов, ни под границей.

### Примеры:

```
background-clip: content-box;
background-clip: padding-box;
```

## 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» раздела «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. В CSS3 допускается использование нескольких фоновых изображений.

**Значения:** URL-адрес изображения. Может также включать сгенерированный браузером линейный или радиальный градиент.

### Примеры:

```
background-image: url(images/photo.jpg);
background-image: url(http://www.example.org/photo.jpg);.
```

## background-origin

Сообщает браузеру, куда помещать фоновое изображение относительно границы, отступов и содержимого элемента. Больше подходит для неповторяющегося



изображения, поскольку позволяет уточнить его позицию. Internet Explorer 8 и более ранние версии это свойство не поддерживают.

**Значения:** border-box, padding-box или content-box. Вариант border-box задает обычный метод — помещение изображения в верхний левый угол границы. Вариант padding-box определяет начало фонового изображения только в области отступов, и под границей оно не появляется. Вариант content-box указывает размещение фонового изображения только в области содержимого, и оно не появляется в области отступов и под границей. Но если изображение размножено, вы все равно его увидите под границей и отступами, поскольку это свойство управляет только тем, где начинается изображение. Чтобы размноженное изображение не появлялось под границей и отступами, следует воспользоваться свойством background-clip.

**Пример:** background-origin: content-box;.

## 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 налево, а также вниз и направо. 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 пиксели 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 пикселях и em рассчитываются от верхнего левого угла элемента, поэтому, чтобы поместить изображение на расстоянии 5 пикселей от левого края и 10 пикселей 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» раздела «Позиционирование фоновых изображений» гл. 8). Вы можете смешать и сопоставить эти значения: если хотите, используйте значение в пикселях 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 изображе-

ние в фоне один раз без какого-либо повторения. Repeat-x повторяет изображение только сверху вниз — это удобно при добавлении графической боковой панели. Repeat-y повторяет изображение только слева направо, так что вы можете добавить графическую полосу вверху, посередине или внизу элемента.

**Пример:** background-repeat: no-repeat;.

## background-size

Позволяет изменять размер фонового изображения, масштабируя его в разных направлениях или даже искажая его пропорции.

**Значения:** можно использовать конкретные значения в пикселах, em или процентах, или же воспользоваться одним из двух ключевых слов: contain или cover. Ключевое слово contain заставляет изображение изменить свои размеры, чтобы полностью поместиться в размеры элемента, сохраняя при этом свои пропорции. Ключевое слово cover заставляет ширину и высоту изображения приобрести такие же значения, как у ширины и у высоты элемента, что обычно искажает изображение, растягивая его или сжимая, чтобы вписать его в размеры элемента.

**Примеры:**

```
background-size: 200px 400px;  
background-size: contain;
```

## 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 применять 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 относительное позиционирование» раздела «Как работают свойства позиционирования» гл. 15).

**Значения:** любая корректная единица измерения CSS, такая как пиксели, em или проценты. Проценты вычисляются на основании ширины содержащего элемента.

**Пример:** bottom: 5em;.

### 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 элемент от обертывания вокруг перемещенных либо влево, либо вправо элементов. Значение `none` отключает свойство, так что используйте это значение для отмены ранее установленного свойства `clear`. Этот прием применяется при наличии у конкретного тега стиля, вызывающего выпадение плавающего элемента, когда нужно, чтобы тег охватывал его только в данном случае. Чтобы отменить свойство `float` только для одного этого тега, создайте более конкретный стиль.

**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 пикселей. Последнее число — левое смещение — левый край окна отсечения. В этом примере смещение равно `10px`, так что будет скрыто все слева (первые 9 пикселей элемента). Второе число определяет ширину окна отсечения плюс последнее число; если левый край отсечения составляет 10 пикселей и вы хотите, чтобы видимая область была шириной 100 пикселей, второе число должно быть `110px`. Третье число — высота области отсечения плюс верхнее смещение (первое число). Так, в этом примере высота области отсечения равна 30 пикселей (`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 линейного (см. подраздел «Отображение линейных и блочных элементов» раздела «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 (линейный 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. раздел «Псевдоклассы и псевдоэлементы» гл. 3).

Еще некоторые 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 пиксели, `em` или проценты. Проценты вычисляются на основании высоты содержащего элемента.

**Пример:** `height: 50%;`

### ПРИМЕЧАНИЕ

---

Иногда содержимое получается больше установленной высоты: если вы вводите, например, много текста 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.

---

## left

При использовании с абсолютным или фиксированным позиционированием (см. раздел «Как работают свойства позиционирования» гл. 15) 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 задействовать это свойство, чтобы поместить изображение на расстоянии 20 пикселей 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 пиксели, 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 пиксели, 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 уже, 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. раздел «Типы разметок веб-страницы» гл. 12). 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 пиксели, 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 пиксели, 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 уже. 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 пиксели, `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. 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`.

**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 мини-фрейма. 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 похоже на фреймы 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. 15.

---

## 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 применять это свойство, чтобы поместить боковую панель 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 пиксели, `em` или проценты.

**Пример:** `right: 5em;.`

---

#### ПРИМЕЧАНИЕ

В Internet Explorer версий 6 и ниже могут быть проблемы при позиционировании элементов с использованием свойства `right`.

---

## 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 стилизуемый 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 пиксели, `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 переключать свойство `visibility` для показа и скрытия элементов на странице. Вы можете также использовать псевдокласс `:hover` (см. раздел «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 шире, 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 пиксели, `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 свойству `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. 15.6).

**Пример:** `z-index: 12;`

### СОВЕТ

---

Значения не должны задаваться в строгом порядке. Если для элемента А установлено значение 1 свойства `z-index`, вам не обязательно указывать значение 2 для элемента В, 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 анимации, преобразований и переходов

В CSS3 добавлены весьма интересные свойства для преобразования элементов путем масштабирования, вращения, наклона и перемещения, а также возможность анимировать изменения от одного CSS-свойства к другому.



## @keyframes

Основой CSS-анимации является правило @keyframes. Оно позволяет дать анимации имя, которое затем можно будет применить к любому элементу страницы и создать набор ключевых кадров (keyframes). (Согласно описанию в гл. 10, ключевые кадры являются теми местами анимации, где происходят изменения CSS-свойств.) Например, чтобы постепенно превратить фон элемента из черного в белый, нужны два ключевых кадра: первый для CSS-установки цвета фона в черный, а второй — для его установки в белый. Ключевых кадров с различными CSS-свойствами может быть сколько угодно.

На момент написания данной книги к правилу @keyframe необходимо было добавлять префикс производителя, а анимация не работала в Internet Explorer 9 и более ранних версиях.

**Значения:** правило @keyframes не похоже на любые другие свойства CSS, фактически это вообще не свойство. Оно названо правилом и намного сложнее обычного свойства. Ему нужно присвоить имя (которое позже будет использоваться для применения анимации к элементу на странице), а затем добавить набор фигурных скобок: { }. Внутри скобок находятся ключевые кадры, в качестве которых могут быть просто два ключевых слова from и to для обозначения первого и последнего ключевого кадра. Каждый ключевой кадр имеет свой набор фигурных скобок, внутри которого помещаются анимируемые CSS-свойства: background-color, font-size, позиция на странице и т. д. Более подробную информацию о работе анимации можно найти в гл. 10.

### Пример:

```
@keyframes myAnimation {
  from {
    background-color: black;
  }

  to {
    background-color: white;
  }
}
```

## animation

Это сокращенный метод применения анимации к элементу. Он является кратким способом включения следующих свойств в одно свойство: animation-name, animation-duration, animation-timing-function, animation-iteration-count, animation-direction, animation-delay и animation-fill-mode. Все эти свойства рассматриваются на следующих страницах. Так как это свойство относится к CSS3, оно для многих браузеров требует применения префиксов производителей и не работает в Internet Explorer 9 и более ранних версиях.

**Значения:** список значений, разделенных пробелами, в том числе свойства анимации, которые перечислены в предыдущем абзаце. Конкретные типы значений, предоставляемые каждому из свойств, будут рассмотрены в следующих статьях (animation-name, animation-duration) и т. д. Обязательными являются только два значения — animation-name и animation-duration. Свойство animation требует использования префиксов производителей. К одному и тому же элементу можно приме-

нить несколько именованных анимаций, предоставив список значений анимаций, с запятой в качестве разделителя.

**Примеры:**

```
animation: myAnimation 2s;  
animation: myAnimation 2s ease-in 2 alternate 5s forwards;  
animation: fadeOut 2s ease-in-out 2 alternate 5s forwards,  
glow 5s;
```

## animation-name

Это свойство используется для назначения анимации, созданной с помощью правила @keyframes. Свойство добавляется в качестве части CSS-селектора, применяемого к одному или нескольким элементам страницы. Например, добавление этого свойства к стилю тега h1 сообщит браузеру, что при загрузке страницы нужно запустить указанную анимацию в отношении всех тегов <h1>. Чтобы это сработало, необходимо также назначить свойство animation-duration. Это свойство требует применения префиксов производителей и не работает в Internet Explorer 9 и более ранних версиях.

**Значения:** имя из правила @keyframes.

**Пример:** animation-name: myAnimation;.

## animation-duration

Определяет продолжительность анимации, указанной свойством animation-name.

**Значения:** значение в секундах — 1s или в миллисекундах — 1000ms.

**Пример:** animation-duration: 2s;.

## animation-timing-function

Задаёт скорость анимации в течение выделенного ей периода. Так, установив продолжительность перехода 5 секунд, можно также управлять тем, как проигрывается переход в течение этих 5 секунд, например медленный старт и быстрый финиш. Это свойство требует применения префиксов производителей и не работает в Internet Explorer 9 и более ранних версиях.

**Значения:** одно из пяти ключевых слов: linear, ease, ease-in, ease-out и ease-in-out. Для пользовательской функции распределения по времени можно также принимать значение кубической кривой Безье.

**Примеры:**

```
animation-timing-function: ease-out;  
animation-timing-function: cubic-bezier(.20, .96, .74, .07);
```

## animation-delay

Определяет время задержки начала воспроизведения анимации в секундах или миллисекундах. Это свойство требует применения префиксов производителей и не работает в Internet Explorer 9 и более ранних версиях.

**Значения:** значение в секундах — 1s или в миллисекундах — 1000ms.

**Пример:** animation-delay: 1.5s;.

## animation-iteration-count

Определяет количество запусков анимации. Обычно анимация запускается один раз, а затем останавливается, но вы можете заставить анимацию запускаться 4, 5, 100 или бесконечное количество раз. Это свойство требует применения префиксов производителей и не работает в Internet Explorer 9 и более ранних версиях.

**Значения:** положительное целое число или ключевое слово `infinite`.

**Примеры:**

```
animation-iteration-count: 5;  
animation-iteration-count: infinite;
```

## animation-direction

Когда анимация запускается более одного раза, это свойство указывает стартовую точку для каждой последующей анимации. Обычно браузер просто заново проигрывает анимацию от начальной стартовой точки снова и снова. Но можно также запустить анимацию вперед, а затем назад, затем опять вперед. Например, при наличии анимации, постепенно превращающей цвет фона из белого в черный, может потребоваться придать ей пульсирующий характер, заставив запускаться много раз и превращать белый цвет в черный, затем черный в белый, затем опять белый в черный и черный в белый и т. д. Это свойство требует применения префиксов производителей и не работает в Internet Explorer 9 и более ранних версиях.

**Значения:** ключевое слово `normal` или `alternate`. Обычно браузер проигрывает анимацию в режиме `normal`, поэтому данное свойство нужно использовать только при потребности использования ключевого слова `alternate`.

**Пример:** `animation-direction: alternate;`

## animation-fill-mode

Задает стилизацию анимируемого элемента в начале и (или) в конце анимации. Это свойство требует применения префиксов производителей и не работает в Internet Explorer 9 и более ранних версиях.

**Значения:** одно из трех ключевых слов: `backwards`, `forwards` или `both`. Ключевое слово `forwards` применяется наиболее часто, поскольку оно оставляет элемент стилизованным так же, как в конце анимации, вместо возвращения к тому стилю, который был до начала анимации.

**Пример:** `animation-fill-mode: backwards;`

## animation-play-state

Управляет проигрыванием анимации. Это свойство можно использовать с псевдоклассом `:hover` для приостановки анимации при прохождении указателя мыши посетителя над элементом. Свойство требует применения префиксов производителей и не работает в Internet Explorer 9 и более ранних версиях.

**Значения:** одно из двух ключевых слов: `pause` или `running`. Ключевое слово `pause` приостанавливает анимацию, а `running` заставляет продолжить ее выполнение. По умолчанию используется ключевое слово `running`.

**Пример:** `animation-play-state: paused;`

## transform

Приводит к изменению элемента одним или несколькими способами, включая масштабирование, вращение, наклон или перемещение. Это свойство требует применения префиксов производителей и не работает в Internet Explorer 8 и более ранних версиях.

**Значения:** ключевые слова `rotate()`, `translate()`, `skew()` или `scale()`. Каждое ключевое слово получает собственный тип значения. Например, `rotate` требует значения угла вращения — `180deg`, `translate` требует такого показателя, как процент, `em` или пиксели, `skew` получает два угловых значения, а `scale` получает положительное или отрицательное число. К одному и тому же элементу можно применить более одного вида преобразования.

**Примеры:**

```
transform: rotate(45deg);
transform: scale(1.5);
transform: skew(45deg 0) rotate(200deg) translate(100px, 0)
scale(.5);
```

## transform-origin

Управляет точкой, в которой применяется преобразование. Например, обычно при вращении элемента за ось вращения берется центр элемента. Но его можно также вращать вокруг одного из его четырех углов. Это свойство требует применения префиксов производителей и не работает в Internet Explorer 8 и более ранних версиях.

**Значения:** два значения, одно для горизонтальной координаты исходной точки, а другое — для вертикальной. Можно использовать те же ключевые слова и значения, что и для свойства `background-position`.

**Примеры:**

```
transform-origin: left top;
transform-origin: 0% 100%;
transform-origin: 10px -100px;
```

## transition

Краткий метод определения свойств `transition-property`, `transition-duration`, `transition-timing-function` и `transition-delay` (рассматриваемых ниже).

Свойство `transition` заставляет браузер анимировать изменения CSS-свойств элемента. Например, можно анимировать изменение фонового цвета кнопки навигации с красного на зеленый при проходе над ней указателя мыши посетителя. Это свойство требует применения префиксов производителей и не работает в Internet Explorer 9 и более ранних версиях.

**Значения:** список свойств с пробелом в качестве разделителя, включающий свойства `transition` (необязательное, по умолчанию имеет значение `all`), `transition-duration` (обязательное), `transition-timing-function` (необязательное, по умолчанию имеет значение `ease`), `transition-delay` (необязательное, по умолчанию имеет значение `0`).

**Пример:** `transition: background-color 1.5s ease-in-out 500ms;`

## transition-property

Определяет конкретные CSS-свойства, подвергаемые анимированию при изменении CSS-свойств элемента. Это свойство требует применения префиксов производителей и не работает в Internet Explorer 9 и более ранних версиях.

**Значения:** анимируемое CSS-свойство или ключевое слово `all`.

**Примеры:**

```
transition-property: width, left, background-color;
transition-property: all;
```

## transition-duration

Определяет продолжительность анимации перехода.

**Значения:** значение в секундах — `1s` или в миллисекундах — `1000ms`.

**Пример:** `animation-duration: 2s;`

## transition-timing-function

Задаёт скорость анимации перехода в течение определенного периода. Например, установив продолжительность перехода 5 секунд, можно также управлять тем, как проигрывается переход в течение этих 5 секунд, включив, допустим, медленный старт и быстрый финиш. Это свойство требует применения префиксов производителей и не работает в Internet Explorer 9 и более ранних версиях.

**Значения:** одно из пяти ключевых слов: `linear`, `ease`, `ease-in`, `ease-out` или `ease-in-out`. Для пользовательской функции распределения по времени можно применить значение кубической кривой Безье.

**Примеры:**

```
transition-timing-function: ease-out;
transition-timing-function: cubic-bezier(.20, .96, .74, .07);
```

## transition-delay

Определяет время задержки перед началом анимации перехода в секундах или миллисекундах. Это свойство требует применения префиксов производителей и не работает в Internet Explorer 9 и более ранних версиях.

**Значения:** значение в секундах — `1s` или в миллисекундах — `1000ms`.

**Пример:** `transition-delay: 1.5s;`

## Свойства таблицы

Существует несколько свойств CSS, которые относятся исключительно к таблицам HTML. В гл. 11 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 пикселей между каждой ячейкой. Даже если вы уберете это пространство, установив значение 0 атрибуту `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. раздел «Создание стилей для таблиц» гл. 11). 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 возникает 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 версии 8 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, форматующему 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 заставляет 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, формирующему `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, расширяют управление 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. Показываем ссылки при печати» в подразделе «Стилизация фонов для печати» раздела «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 ограничено.

**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](http://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`, `nesize`, `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](http://www.quirksmode.org/css/cursor.html).

## opacity

Позволяет управлять прозрачностью любого элемента и всех его потомков. При этом через элемент могут просвечиваться находящиеся под ним цвета, изображения и содержимое. Учтите, что при применении свойства `opacity` к `div`-контейнеру тот же уровень прозрачности получают все содержащиеся в нем заголовки, изображения, абзацы и другие `div`-контейнеры. То есть, если для тега `<div>` установить значение свойства `opacity`, равное `.5` (50 % прозрачность), изображение внутри этого тега также получит 50 %-ную прозрачность, даже если конкретно для него будет установлено значение `1`, она не отменит 50 %-ную прозрачность.

**Значения:** десятичное значение от 0 до 1. Значение 0 означает невидимость, а значение 1 — полную непрозрачность.

**Пример:** `opacity: .5;`

## 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 Орега 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 заставить браузер напечатать дополнительную пустую страницу. Но, поскольку никакие браузеры не понимают эти значения, 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 наверх 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 `<div>`, 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 наверх следующей страницы. Чтобы предотвратить появление таких «висячих» строк, используйте свойство `widows`, заставляющее браузер переместить по крайней мере две или три строки вместе наверх печатной страницы (только Opera понимает это свойство, поэтому оно ограничено в использовании).

**Значения:** числа, например 1, 2, 3 или 5.

**Пример:** `widows: 3;`

# 2 Информационные ресурсы, касающиеся CSS

К сожалению, одна книга не может ответить на все ваши вопросы по 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 в 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)

Текущие разработки CSS ([www.w3.org/Style/CSS/current-work](http://www.w3.org/Style/CSS/current-work)). Здесь находятся все CSS-спецификации, включая самые новые дополнения. Различные спецификации разбиты по состоянию, например Completed — завершённые, Revising — пересматриваемые или Abandoned — отменённые. Можно щелкнуть на любой спецификации, чтобы получить ее подробное описание, но принадлежность спецификации к разряду Completed (завершённая) еще не означает, что она полностью реализована во всех веб-браузерах. Но все же этот сайт предоставляет последнее слово, касающееся CSS (иногда чересчур сложное и малопонятное).

### Онлайн-справочники

- CSS-справочник в сети Mozilla Developer Network ([https://developer.mozilla.org/en-US/docs/CSS/CSS\\_Reference](https://developer.mozilla.org/en-US/docs/CSS/CSS_Reference)). Сеть разработчиков Mozilla Developer Network (MDN) предоставляет один из самых полных справочников по CSS (а также по HTML5, JavaScript и другим веб-технологиям).
- CSS-справочник Sitepoint (<http://reference.sitepoint.com/css>) предоставляет четкий, ориентированный на разработчиков справочник CSS-свойств и понятий. На него вполне можно поменять документацию W3C или MDN, если от них ваши глаза уже устали.
- Can I use... (Могу ли я использовать...)(<http://caniuse.com>). Этот часто обновляемый сайт предоставляет подробную информацию по CSS и браузерной

совместимости. Здесь можно определить, будет ли то или иное CSS-свойство работать, например, в Internet Explorer 9.

- CSS3-файлы (<http://www.css3files.com/>) предоставляет неплохие инструкции и еще более хорошие демонстрации наиболее популярных свойств CSS3, таких как анимация, тени, градиенты и многое другое.

## Справочная информация по CSS

Даже при наличии лучших справочников (таких как эта книга) иногда приходится обращаться за помощью к знатокам. 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.

**Расылки.** 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

- CSSCreator ([www.csscreator.com/css-forum/](http://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](http://www.sitepoint.com/forums/forumdisplay.php?f=53)). Другая полезная группа для любителей CSS.
- CSS-Tricks.com (<http://css-tricks.com/forums>). На этом относительно новом форуме можно найти полезную информацию. Если вы программируете на PHP или JavaScript, то найдете здесь много полезного для себя.

## Подсказки, приемы и советы по CSS

Сеть упрощает для всех возможность стать издателем. Однако когда каждый является издателем, тяжело перебирать весь материал и найти в нем понятную, краткую 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](http://www.sitepoint.com/subcat/css)). Содержит множество статей и руководств по технологии CSS. Здесь также часто появляются самые последние новости и информация по CSS.

- Smashing Magazine ([www.smashingmagazine.com/category/css/](http://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.
- NetMagazine (<http://www.netmagazine.com/tag/css?ct=tutorial>). На этом сайте часто можно найти интересные и полезные руководства по CSS.

## CSS-навигация

В гл. 9 было показано, 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, которые могут вдохновить вас на новые свершения.

### Учебные пособия.

- Listutorial (<http://css.maxdesign.com.au/listutorial/>). Пошаговые пособия по созданию навигационных систем из неупорядоченных списков.
- 40 Premium CSS Menu and Navigation Solutions ([www.tripwiremagazine.com/2012/06/css-menu-and-navigation.html](http://www.tripwiremagazine.com/2012/06/css-menu-and-navigation.html)). Дополнительные учебники.
- Меню навигации Apple.com, созданной исключительно с применением средств CSS3 ([www.marcofolio.net/css/the\\_apple.com\\_navigation\\_menu\\_created\\_using\\_only\\_css3.html](http://www.marcofolio.net/css/the_apple.com_navigation_menu_created_using_only_css3.html)). Если вам нравится простой и понятный внешний вид Apple.com, можно поинтересоваться, как создать их меню с помощью CSS.

### Примеры в режиме онлайн

- CSS Navigation Bar Code Generator (<http://lab.mattvarone.com/navbar>). Ленились или желаете сэкономить время? Позвольте этому онлайн-инструменту создать весь необходимый код для использования волшебного метода, рассмотренного ранее в этой книге.
- CSS Menu Maker (<http://cssmenumaker.com>). Большой выбор интересных панелей навигации с широким использованием новых свойств CSS3. Их можно настроить под нужды вашего сайта, а затем загрузить код.
- CSS Showcase ([www.alvit.de/css-showcase/](http://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/>). CSS-меню, включающее вложенные списки с подменю.
- CSS Menu Builder ([www.cssmenubuilder.com/](http://www.cssmenubuilder.com/)). Множество интересных меню и полезных технологий. Посмотрите обязательно.
- Адаптируемое навигационное меню CSS (<http://pixelsdaily.com/resources/html-css/responsive-css-navigation-menu/>). Адаптируемое меню, изменяющееся от горизонтальной панели до двухстрочной панели и до стопки кнопок в два столбца.

## 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/](http://www.redmelon.net/tstme/box_model/)) Забавный интерактивный инструмент для визуализации блочной модели.
- Блог-пост Пола Айриша (Paul Irish) \* { box-sizing: border-box } FTW (<http://paulirish.com/2012/box-sizing-border-box-ftw/>). Довольно влиятельный блог-пост, выступающий за отмену старой блочной модели CSS и за использование свойства `box-sizing` в качестве более подходящего способа управления высотой и шириной элементов.

### Примеры разметок

- PageBlox ([www.pageblox.com](http://www.pageblox.com)). Предоставляет онлайн-средство для создания адаптируемых конструкций средствами CSS. Учитывает недостатки браузеров и генерирует медиазапросы для больших и маленьких экранов.
- CSS Layout Generator ([www.pagecolumn.com](http://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 теперь нужна эта книга? (Шучу.)
- Even More Layout Generators ([www.webdesignbooth.com/15-extremely-usefulcss-grid-layout-generator-for-web-designers](http://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/>). Есть даже онлайн-средство, помогающее создавать свою собственную grid-структуру на основе grid-системы 960: <http://grids.herokuapp.com>.
- 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.
- Twitter Bootstrap (<http://twitter.github.com/bootstrap/>). Полный набор инструментов для создания веб-сайта, включающий компоненты HTML, CSS и JavaScript, упрощающие создание полноценной, адаптируемой страницы на основе grid-технологии с весьма интересными вставками на JavaScript.
- Foundation (<http://foundation.zurb.com>). Еще один комплект инструментов для разработки веб-сайтов. Он очень похож на Twitter Bootstrap и включает HTML, CSS и JavaScript. Содержит прекрасную документацию и относительно легко в освоении.

### Разнообразные ресурсы по разметкам

- Adaptive CSS Layouts ([www.smashingmagazine.com/2009/06/09/smart-fixesfor-fluid-layouts](http://www.smashingmagazine.com/2009/06/09/smart-fixesfor-fluid-layouts)). Предоставляет множество ресурсов для создания гибких разметок, адаптирующихся под полную ширину окна браузера.
- TJK Design ([http://tjkdesign.com/articles/one\\_html\\_markup\\_many\\_css\\_layouts.asp](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/](http://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.

## 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](http://www.csszengarden.com)). Источник всех сайтов-выставок CSS: много различных дизайнов для одного и того же HTML-кода.
- CSS Beauty ([www.cssbeauty.com/](http://www.cssbeauty.com/)). Замечательная галерея вдохновляющих CSS-дизайнов.
- CSS Elite ([www.csselite.com](http://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.
- CSS Winner ([www.csswinner.com](http://www.csswinner.com)). На этом сайте каждый день объявляется новый «победитель». Не могу понять, как при таких суждьях, являющихся на самом деле простыми подростками, бродящими по Интернету, не выходя из своих комнат, выявляются весьма интересно сконструированные сайты.

## 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.

### Windows и Mac

- Style Master ([www.westciv.com/style\\_master/product\\_info/](http://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/](http://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.
- Sublime Text (<http://www.sublimetext.com>). Эффективный редактор кода, подходящий для создания широкого спектра текстовых документов. Он не является инструментальным средством для разработки CSS, но пользуется широкой популярностью среди веб-разработчиков.

#### **Только для Windows**

- Top Style ([www.topstyle4.com](http://www.topstyle4.com)). 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](http://www.microsoft.com/expression/products/Web_Overview.aspx)). Полноценный инструмент для проектирования сайтов, который очень хорошо работает с CSS.

#### **Только для Mac**

- Espresso (<http://www.macrabbit.com/espresso/>). Универсальное средство для веб-дизайна, включающее редактор кода, FTP-программу и CSSEdit 3, эффективный и простой редактор CSS.
- Code (<http://panic.com/coda/>). Еще одно универсальное средство для создания веб-приложений со встроенным средством редактирования CSS.