

НИГТИЕН

РЕЙЧЕЛ ЭНДРЮ

3  
издание

# ССС

100 и 1 совет



СИМВОЛ®

По договору между издательством «Символ-Плюс» и Интернет-магазином «Books.Ru – Книги России» единственный легальный способ получения данного файла с книгой ISBN 978-5-93286-182-0, название «CSS: 100 и 1 совет, 3-е издание» – покупка в Интернет-магазине «Books.Ru – Книги России». Если Вы получили данный файл каким-либо другим образом, Вы нарушили международное законодательство и законодательство Российской Федерации об охране авторского права. Вам необходимо удалить данный файл, а также сообщить издательству «Символ-Плюс» ([piracy@symbol.ru](mailto:piracy@symbol.ru)), где именно Вы получили данный файл.

# The CSS Anthology

101 Essential Tips, Tricks & Hacks

Third Edition

*Rachel Andrew*



H I G H T E C H

# CSS

## 100 и 1 совет

Третье издание

*Рейчел Эндрю*



---

*Санкт-Петербург — Москва*  
*2010*



Серия «High tech»

Рейчел Эндрю

## CSS: 100 и 1 совет, 3-е издание

Перевод А. Минаевой

Главный редактор	<i>А. Галунов</i>
Зав. редакцией	<i>Н. Макарова</i>
Выпускающий редактор	<i>П. Щеголев</i>
Научный редактор	<i>В. Коршунов</i>
Редактор	<i>Ю. Бочина</i>
Корректор	<i>Е. Кирюхина</i>
Верстка	<i>К. Чубаров</i>

*Эндрю Р.*

CSS: 100 и 1 совет, 3-е издание. – Пер. с англ. – СПб: Символ-Плюс, 2010. – 336 с., ил.

ISBN 978-5-93286-182-0

«CSS: 100 и 1 совет» представляет интерес для веб-дизайнеров и разработчиков, у которых нет времени на штудирование огромного количества теоретических материалов при создании собственного сайта. Это сборник готовых решений наиболее распространенных проблем, которые можно сразу применить на практике; более того, они могут послужить основой для разработки собственных методов. Здесь представлены ответы на сложные вопросы и практические методы использования CSS, примеры создания сложных макетов страниц, элементов навигации и форм, решение проблем, связанных с особенностями различных браузеров.

Чтение книги не требует специальной подготовки: первая глава содержит обзор основных особенностей CSS, но в дальнейшем сложность приводимых методов постепенно возрастает, так что наличие у читателя базовых знаний CSS существенно облегчит восприятие материала.

Третье издание полностью пересмотрено и обновлено с целью охвата новейших технологий и особенностей самых современных браузеров, включая Firefox 3 и Internet Explorer 8. Данную книгу можно использовать как справочник для поиска подходящего решения при создании сайта и тем самым выиграть время и сдать проект в срок.

**ISBN 978-5-93286-182-0**

**ISBN 978-0-9805768-0-1 (англ)**

© Издательство Символ-Плюс, 2010

Authorized translation of the English edition © 2009 SitePoint Pty Ltd. This translation is published and sold by permission of SitePoint Pty Ltd, the owner of all rights to publish and sell the same.

Все права на данное издание защищены Законодательством РФ, включая право на полное или частичное воспроизведение в любой форме. Все товарные знаки или зарегистрированные товарные знаки, упоминаемые в настоящем издании, являются собственностью соответствующих фирм.

Издательство «Символ-Плюс». 199034, Санкт-Петербург, 16 линия, 7, тел. (812) 324-5353, www.symbol.ru. Лицензия ЛП N 000054 от 25.12.98.

Подписано в печать 22.04.2010. Формат 70×100<sup>1/16</sup>. Печать офсетная.

Объем 21 печ. л. Тираж 1500 экз. Заказ №

Отпечатано с готовых диапозитивов в ГУП «Типография «Наука»  
199034, Санкт-Петербург, 9 линия, 12.

*Посвящается Бетани*



# Оглавление

Предисловие .....	13
<b>1. CSS: основы основ .....</b>	<b>20</b>
Определение стиля с помощью CSS .....	20
Что такое селекторы и как их правильно использовать.....	25
Каким образом браузер определяет, какие стили нужно использовать .....	32
Заключение .....	34
<b>2. Оформление текста и другие базовые возможности .....</b>	<b>35</b>
Задание определенного шрифта для текста .....	35
Выбор единиц измерения размера шрифтов: пиксели, пункты, пики или что-то другое .....	37
Удаление подчеркивания ссылок .....	44
Создание ссылки, меняющей цвет при наведении на нее указателя мыши .....	46
Использование на одной странице различных стилей ссылок .....	48
Присваивание первому элементу в списке отличного от последующих элементов стиля.....	50
Создание цветного фона для заголовка .....	51
Подчеркивание заголовков.....	52
Устранение отступа между элементом h1 и следующим за ним абзацем .....	53
Выделение текста на странице .....	55
Изменение высоты строки (межстрочного интервала) в тексте.....	56
Выравнивание текста по ширине.....	57
Изменение стиля горизонтальной линии .....	58
Вывод текста с отступом.....	59
Центрирование текста .....	60
Вывод текста заглавными буквами с помощью CSS .....	61
Изменение стиля маркеров списка или удаление маркеров .....	62
Использование изображения вместо маркера списка.....	64
Удаление у пунктов списка отступа слева .....	64

Размещение пунктов списка по горизонтали .....	66
Удаление отступов от края страницы .....	66
Удаление отступов по умолчанию для всех элементов страницы .....	67
Добавление комментария в файл с каскадной таблицей стилей .....	69
Заключение .....	69
<b>3. CSS и графика .....</b>	<b>70</b>
Добавление рамки к изображению .....	70
Удаление средствами CSS синей рамки вокруг изображения, выполняющего функцию ссылки .....	72
Задание фонового изображения для страницы с помощью CSS .....	72
Как изменить способ размножения фонового изображения .....	74
Как позиционировать фоновое изображение .....	76
Как сделать фоновое изображение неподвижным при прокрутке контента .....	79
Для каких элементов можно задавать фоновое изображение .....	81
Размещение текста поверх изображения .....	84
Как задать для документа более одного фонового изображения .....	85
Применение эффекта прозрачности .....	87
Создание сложных рамок вокруг изображений, например двойных .....	89
Заключение .....	91
<b>4. Навигация .....</b>	<b>92</b>
Оформление списка в виде навигационного меню .....	93
Изменение вида ссылки при наведении на нее указателя мыши с помощью CSS без использования изображений или сценариев на JavaScript .....	96
Создание навигационного меню с подпунктами с помощью списков и таблиц стилей .....	98
Создание горизонтального меню с помощью списков и CSS .....	102
Создание средствами CSS навигационной панели с кнопками .....	105
Создание с помощью CSS панели навигации на основе вкладок .....	107
Выделение ссылок, ведущих на внешний сайт .....	113
Изменение вида курсора .....	116
Реализация смены изображений на панели навигации без использования JavaScript .....	118
Оформление карты сайта .....	122
Создание выпадающего меню исключительно средствами CSS .....	125
Создание доступного навигационного меню на основе изображений с помощью CSS .....	126
Заключение .....	131

<b>5. Табличные данные</b> .....	132
Представление табличных данных с помощью CSS .....	132
Организация табличных данных: удобство доступа и наглядность.....	134
Создание рамки вокруг таблицы без использования HTML-атрибута border .....	137
Удаление пустого пространства между ячейками, появляющегося после добавления рамок .....	139
Представление табличных данных в привлекательной и удобной форме .....	139
Чередование фонового цвета строк таблицы .....	143
Изменение фонового цвета строки при наведении на нее указателя мыши .....	147
Чередование фонового цвета столбцов таблицы .....	150
Создание календаря с помощью CSS .....	153
Заключение .....	163
<b>6. Формы и пользовательские интерфейсы</b> .....	164
Изменение вида элементов формы с помощью CSS .....	165
Использование разных стилей для разных полей одной и той же формы.....	168
Избавление от переносов строки и потери места на странице .....	171
Придание кнопке подтверждения вида текста .....	172
Возможность заполнения формы для пользователей текстовых устройств .....	173
Создание двухколоночной формы с помощью CSS вместо таблиц ....	176
Группировка связанных полей формы .....	180
Задание стиля для клавиш быстрого доступа .....	184
Использование цветного фона для меню, созданного с помощью элементов select .....	186
Создание таблицы стилей для формы с возможностью ввода данных, как в электронную таблицу.....	188
Выделение поля формы, по которому пользователь щелкает мышью .....	194
<b>7. Кросс-браузерные решения</b> .....	197
В каких браузерах следует протестировать свой сайт? .....	198
Тестирование сайта в различных браузерах при наличии только одной ОС .....	199
Сервисы, показывающие вид сайта в различных браузерах .....	203
Возможность поддержки нескольких версий Internet Explorer в Windows .....	205

Определение круга браузеров, для которых необходимо поддерживать все аспекты дизайна страницы .....	206
Указание базовой таблицы стилей для самых старых браузеров.....	207
Что такое режим совместимости и как его избежать.....	211
Задание разных таблиц стилей для Internet Explorer 6 и 7 .....	213
Уход от наиболее распространенных ошибок в Internet Explorer 6 и 7 .....	215
Достижение прозрачности изображения в формате PNG в Internet Explorer 6 .....	219
Корректное отображение в IE 8 сайта, соответствующего стандартам W3C .....	223
Что делать, если CSS не работает .....	224
Интерпретация сообщений, выводимых инструментом W3C Validator .....	227
Заключение .....	229
<b>8. Доступность и альтернативные устройства .....</b>	<b>230</b>
Аспекты доступности, о которых следует помнить при использовании CSS.....	230
Тестирование сайта в текстовом браузере .....	232
Тестирование сайта с помощью экранного диктора .....	234
Создание отдельных таблиц стилей для различных устройств .....	235
Создание таблицы стилей для печатной версии документа .....	238
Добавление на сайт альтернативных таблиц стилей .....	246
Нужно ли отображать на сайте инструменты для изменения размера шрифта или переключения между различными таблицами стилей?.....	251
Создание альтернативных таблиц стилей без копирования кода из основной таблицы .....	252
Заключение .....	256
<b>9. Позиционирование элементов с помощью CSS.....</b>	<b>257</b>
В каких случаях следует использовать классы, а в каких – идентификатор .....	257
Отображение строкового элемента как блочного, и наоборот .....	258
Задание внешних и внутренних отступов с помощью CSS.....	261
Обтекание текстом изображения .....	264
Как избежать смещения следующего элемента вверх при использовании свойства float .....	266
Как расположить логотип сайта слева, а слоган – справа .....	270
Позиционирование элемента на странице с помощью CSS .....	273
Центрирование блока на странице .....	277
Создание блока с закругленными краями .....	279

---

Создание «резинового» макета: слева – меню, а справа – область с контентом .....	286
Изменение расположения элементов макета на противоположное, чтобы меню было справа .....	292
Макет фиксированной ширины с двумя колонками по центру страницы.....	294
Создание колонки, занимающей все доступное пространство по высоте.....	305
Добавление тени к блоку .....	307
Создание макета с тремя колонками средствами CSS.....	310
Добавление к «резиновому» макету нижнего блока.....	315
Создание галереи миниатюр .....	318
Создание макета страницы с помощью CSS-таблиц .....	323
Заключение .....	328
<b>Алфавитный указатель.....</b>	<b>330</b>





## Предисловие

Кроме написания книг, подобных той, что вы держите в руках, я пишу код. Я зарабатываю на жизнь разработкой веб-сайтов и приложений, надо полагать, как и многие мои читатели. Я использую CSS в повседневной работе, и мне прекрасно известно, чего стоят отчаянные попытки найти ошибки в коде, когда проект должен быть сдан на следующее утро.

Многие дизайнеры и разработчики, предпочитающие использовать каскадные таблицы стилей лишь для простого форматирования текста или вовсе обходиться без их использования, объясняют это нехваткой времени на изучение новой для них технологии с нуля. В конечном счете применение таблиц и изображений в формате GIF позволяет достичь поставленной цели, а значит, работа выполнена и оплачивается.

Мне повезло: я начала изучение технологии CSS с момента ее появления и настолько увлеклась, что мне захотелось постоянно совершенствовать свои навыки. Именно благодаря рано появившемуся интересу к CSS мои знания развивались вместе с самой технологией, и теперь за моими плечами девятилетний опыт проектирования веб-сайтов с помощью этой технологии.

В данной книге я намерена представить полезные приемы и методы использования CSS, упрощающие процесс и сокращающие время разработки веб-сайтов и приложений.

Эта книга не предлагает вам многостраничных теоретических рассуждений. В ней вы найдете решения, которые можно сразу же применить на практике; более того, они могут послужить основой для разработки собственных методов. По опыту могу сказать, что легче чему-то научиться делая, чем просто читая, поэтому вполне можете полагаться на данную книгу как на справочник для поиска подходящего решения при создании клиентского веб-сайта и тем самым выиграть время и сдать проект в срок. При этом хочется надеяться, что экспериментирование с предлагаемыми примерами поможет освоению новых технологий.

Книга составлена таким образом, чтобы вы смогли использовать представленные материалы для решения текущих задач. Ее необязательно читать от корки до корки – достаточно изучить интересующий вас в данный момент раздел. Каждый пример сопровождается комментарием, объясняющим механизм работы данного метода. Это позволит вам использовать эти примеры в дальнейшем, модифицируя их в соответствии с решаемыми в данный момент проблемами.

Надеюсь, что вам понравится эта книга. Я получила большое удовольствие от самого процесса ее написания и надеюсь, что она станет для вас полезным ежедневным справочником, а также помощником в освоении новых методов использования CSS.

## Для кого предназначена эта книга

Данная книга будет интересна всем, кому приходится работать с CSS, а в особенности веб-дизайнерам и разработчикам, которые, конечно, видели много красивых веб-сайтов, основанных на применении CSS, но у которых нет времени на то, чтобы проштудировать огромное количество теоретических и накопленных практических материалов при создании собственного сайта. В этой книге вы найдете готовые решения различных проблем, которые могут быть использованы непосредственно в представленном здесь виде или взяты за основу при создании собственных решений.

В целом данную книгу нельзя назвать учебником; так как первая глава содержит обзор только основных особенностей CSS и сложность приводимых методов возрастает при движении от начальных глав к заключительным, то наличие у вас базовых знаний CSS существенно облегчило бы восприятие материала книги.

## О чем написано в этой книге

### Глава 1. CSS: основы основ

Данная глава по форме сильно отличается от прочих частей книги. По сути, она является кратким учебником по базовым возможностям CSS, который позволит также освежить имеющиеся знания. Если вы активно используете CSS в своих проектах, можете пропустить эту главу и возвращаться к ней по мере необходимости – если нужно более глубоко изучить основополагающие принципы технологии.

### Глава 2. Форматирование текста и другие базовые возможности

Данная глава содержит информацию о различных методах форматирования и задания стиля текста в ваших документах: настройки размера шрифта, цвета и избавления от раздражающего пустого пространства, обрамляющего элементы страницы. Вы наверняка найдете для себя интересные приемы, даже если давно пользуетесь CSS для задания стилизового оформления текста.

### Глава 3. CSS и графика

В данной главе я предлагаю различные приемы совместного использования CSS и графических элементов, позволяющие добиться впечатляющих результатов. Среди рассматриваемых тем – управление фоновым изображением различных элементов, позиционирование текста и изображений, а также многие другие.

#### **Глава 4. Навигация**

Без навигации не обойдется ни один проект, и в данной главе мы рассмотрим методы ее создания с помощью CSS. Представленные приемы позволят использовать CSS вместо графического меню, реализовать навигацию с помощью *вкладок*, создавать удобные и привлекательные панели навигации на основе CSS и фоновых изображений, а также использовать списки для структурирования системы навигации.

#### **Глава 5. Табличные данные**

Следует избегать использования таблиц для позиционирования элементов на странице: их необходимо применять по прямому назначению – для представления табличных данных, как в электронных таблицах. В данной главе мы познакомимся с приемами оформления табличных данных в наиболее привлекательной и удобной для использования форме.

#### **Глава 6. Формы и пользовательские интерфейсы**

Если вы работаете в качестве дизайнера или разработчика, то, несомненно, тратите уйму времени на создание форм для ввода данных. Из данной главы вы узнаете, как с помощью CSS сделать ваши формы удобными, привлекательными и доступными для различных категорий пользователей.

#### **Глава 7. Кросс-браузерные решения**

Как быть со старыми версиями браузеров, браузерами, неправильно интерпретирующими определенные свойства CSS, и альтернативными устройствами? Решением этих проблем мы займемся в главе 7. Кроме того, мы рассмотрим способы поиска «багов» и тестирования веб-страниц в максимально возможном количестве браузеров.

#### **Глава 8. Доступность для людей с ограниченными возможностями и альтернативные устройства**

Ваши страницы прекрасно выглядят для большинства посетителей сайта? Замечательно, однако как же быть людям, вынужденным использовать специальные возможности, такие как экранная лупа и экранный диктор? А пользователям, по тем или иным причинам предпочитающим при просмотре веб-сайтов пользоваться клавиатурой вместо мыши? В данной главе вы узнаете о том, как сделать ваши страницы одинаково доступными для *всех* пользователей, а не только для тех, кто не имеет никаких ограничений при доступе к сайту.

#### **Глава 9. Позиционирование элементов с помощью CSS**

В данной главе мы рассмотрим различные способы позиционирования элементов страницы и множество приемов, которые можно сочетать произвольным образом или взять за основу для своих собственных методов при создании оригинальной разметки страницы.

## Веб-сайт книги

Сайт данной книги, доступный по адресу <http://www.sitepoint.com/books/cssant3/>, предлагает следующие возможности.

## Архив кода

По ходу чтения книги вы заметите расположенные над листингами названия файлов. Они соответствуют именам файлов в архиве кода, который можно загрузить в виде файла в формате ZIP, щелкнув по ссылке Code Archive на сайте книги. В архиве содержатся все завершённые примеры, представленные в данной книге<sup>1</sup>.

## Опечатки и обновления

Ни одна книга не безупречна, и внимательный читатель наверняка заметит пару-тройку ошибок. Страница Corrections and Typos на сайте книги содержит актуальную информацию о найденных типографских ошибках и опечатках в коде, а также обновления, связанные с выходом новых версий браузеров и появлением новых стандартов.<sup>2</sup>

## Форумы SitePoint

В сообществе дизайнеров SitePoint вы можете обсудить данную книгу.<sup>3</sup> В частности, на форуме, посвященном CSS, вы найдете огромное количество информации, выходящей далеко за пределы охвата данной книги. Здесь общаются многие опытные веб-разработчики и дизайнеры, делясь новыми приемами и трюками, отвечая на вопросы, – вы не пожалеете о затраченном времени.<sup>4</sup>

## Электронные рассылки SitePoint

Помимо книг, подобных этой, SitePoint составляет бесплатные электронные рассылки, среди которых *The SitePoint Tribune*, *The SitePoint Tech Times* и *The SitePoint Design View*. В них содержится самая свежая информация, охватывающая различные аспекты разработки веб-сайтов: последние новости, выпуск новых продуктов, актуальные тенденции и полезные приемы. Возможно, вы захотите читать только новости, касающиеся CSS, но если вас интересуют и другие технологии, то вы сможете найти для себя множество полезных тем. Оформить бесплатную подписку можно по адресу <http://www.sitepoint.com/newsletter>.

---

<sup>1</sup> Архив примеров кода можно скачать с сайта издательства по адресу [www.symbol.ru/library/css-101-tips](http://www.symbol.ru/library/css-101-tips)

<sup>2</sup> <http://www.sitepoint.com/books/cssant3/errata.php>

<sup>3</sup> <http://www.sitepoint.com/forums/>

<sup>4</sup> <http://www.sitepoint.com/launch/cssforum/>

## Подкаст SitePoint

Подкасты SitePoint содержат актуальные новости и интервью; веб-разработчики и дизайнеры делятся своим мнением по различным вопросам. Они обсуждают последние тенденции в области разработки веб-сайтов, приглашают интересных гостей и проводят интервью с самыми влиятельными людьми в отрасли. Предыдущие и текущие подкасты можно послушать по адресу <http://www.sitepoint.com/blogs/category/podcast/>; кроме того, вы можете оформить подписку через iTunes.

## Обратная связь

Если вы не смогли найти ответ на свой вопрос на форумах или хотите связаться с нами по какой-то иной причине, это можно сделать по адресу [books@sitepoint.com](mailto:books@sitepoint.com). Ваши письма будут рассмотрены службой поддержки, сотрудники которой готовы ответить на ваши вопросы. Кроме того, мы с благодарностью примем ваши предложения с уточнениями и сообщения о найденных ошибках.

## Благодарности

Прежде всего, мне хотелось бы поблагодарить всех сотрудников SitePoint – благодаря им эта книга увидела свет. С вами было приятно работать, несмотря на различия часовых поясов – я уже засыпала, когда ваш рабочий день только начинался.

Спасибо тем, кто развивает новые идеи в мире CSS, многие из которых будут рассмотрены на страницах книги, и тем, кто делится ими с сообществом разработчиков.

Спасибо Дрю за одобрение и поддержку, за то, что обсуждал со мной методы использования CSS, когда я работала над примерами для книги, за то, что заставлял меня улыбнуться, когда я начинала раздражаться от усталости, за готовность примириться с тем, что мы практически совершенно выпали из светской жизни. И наконец, я хочу поблагодарить свою дочь Бетани за понимание, когда я проводила все время за компьютером, и за то, что каждый день напоминала мне о том, что в жизни действительно важно. Благодаря вам двоим многое становится возможным, спасибо.

## Обозначения, принятые в книге

Любая разметка – HTML или CSS – выводится моноширинным шрифтом, как в следующем примере:

```
<h1>A perfect summer's day</h1>
<p>It was a lovely day for a walk in the park. The birds
were singing and the kids were all back at school.</p>
```

Если данный код представлен в размещенном на сайте архиве, в верхней части листинга будет отображено имя соответствующего файла, например:

*example.css*

```
.footer {
  background-color: #CCC;
  border-top: 1px solid #333;
}
```

Если выводится только часть кода, к названию будет добавлено слово **фрагмент**:

*example.css (фрагмент)*

```
border-top: 1px solid #333;
```

Если к рабочему примеру добавляется новый код, он выделяется жирным шрифтом:

```
function animate() {
  new_variable = "Hello";
}
```

Если для контекста необходим существующий код, вместо повторения будет использовано вертикальное многоточие:

```
function animate() {
  ⋮
  return new_variable;
}
```

Некоторые части кода не умецаются в одну строчку из-за ограниченной ширины печатной страницы. Для обозначения разрыва строки используется знак ↵. Он выполняет исключительно функцию форматирования.

```
URL.open("http://www.sitepoint.com/blogs/2007/05/28/user-style-she
↵ets-come-of-age/");
```

Примечания, заметки, советы и предупреждения оформлены следующим образом:

#### Совет

---

Так будут представлены полезные советы и подсказки.

---

#### Примечание

---

Заметки содержат дополнительную информацию, связанную с рассматриваемой в данный момент темой.

---

**Внимание** 

---

На эти важные аспекты следует обратить внимание.

---

**Предупреждение** 

---

Предупреждения указывают на подводные камни, с которыми вы можете столкнуться в ходе работы.

---

С сайта издательства ([www.symbol.ru/library/css-101-tips](http://www.symbol.ru/library/css-101-tips)) можно скачать цветные версии тех иллюстраций, где наличие цвета может помочь лучше разобраться в предлагаемом материале.



# 1

## CSS: ОСНОВЫ ОСНОВ

Каскадные таблицы стилей... Звучит несколько устрашающе. Такое название способно вызвать в воображении нагромождение таинственных символов кода, едва ли доступных для понимания новичку. Однако на самом деле CSS – один из самых простых и удобных в использовании инструментов, имеющихся в распоряжении веб-разработчиков. В первой главе, отличающейся по формату от последующих частей, мы познакомимся с основами CSS и использованием этой технологии для облегчения задачи управления единообразно оформленным сайтом. Если у вас уже есть некоторый опыт работы с CSS для форматирования текста на веб-страницах, можете пропустить данную главу и приступить к чтению второй главы, где мы начнем рассмотрение практических решений.

### Определение стиля с помощью CSS

CSS используется главным образом для создания **описаний** (или **деклараций**) **стилей** (к примеру, шрифта, размера или цвета) и их применения к выбранным частям HTML-кода посредством **селекторов** – ссылок на элемент или группу элементов, по отношению к которым нужно применить указанный стиль.

### Решение

Рассмотрим данный основополагающий механизм на примере следующего HTML-документа:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
  <head>
    <title>A Simple Page</title>
```

```
<meta http-equiv="content-type"
      content="text/html; charset=utf-8" />
</head>
<body>
  <h1>First Title</h1>
  <p>A paragraph of interesting content.</p>

  <h2>Second Title</h2>
  <p>A paragraph of interesting content.</p>

  <h2>Third title</h2>
  <p>A paragraph of interesting content.</p>
</body>
</html>
```

Он содержит три заголовка (выделенных жирным шрифтом), созданных с помощью тегов `<h1>` и `<h2>`. Если разработчик не определил собственные стили, заголовки будут оформлены в соответствии со стандартной таблицей стилей браузера, т. е. заголовок `h1` будет отображаться крупным шрифтом, `h2` – мельче, чем `h1`, но крупнее обычного текста абзаца. Документ, оформленный стилями по умолчанию, будет вполне *читаемым*, если он достаточно прост. Для изменения внешнего вида этих элементов будет достаточно добавить несколько строчек CSS-кода:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
  <head>
    <title>A Simple Page</title>
    <meta http-equiv="content-type"
          content="text/html; charset=utf-8" />
    <style type="text/css">
      h1, h2 {
        font-family: sans-serif;
        color: #3366CC;
      }
    </style>
  </head>
  <body>
    <h1>First Title</h1>
    <p>A paragraph of interesting content.</p>

    <h2>Second Title</h2>
    <p>A paragraph of interesting content.</p>

    <h2>Third title</h2>
    <p>A paragraph of interesting content.</p>
  </body>
</html>
```

Все преобразования разместились между тегами `<style>`, расположенными в заголовке страницы (элемент `head`). Мы определили, что заго-

ловки `h1` и `h2` должны отображаться шрифтом `sans-serif` светло-голубого цвета. Особенности синтаксиса мы рассмотрим чуть позднее. Нам ничего не потребуется менять в разметке страницы – изменения в описании стиля, размещенного в верхней части страницы, распространяются на все три имеющихся на странице заголовка и будут также применены ко всем заголовкам, добавленным позднее с помощью тегов `<h1>` или `<h2>`.

### Примечание

**HTML или XHTML?** На протяжении всей книги при упоминании веб-страниц, разметки и примеров кода я буду использовать термин HTML. Это можно интерпретировать как HTML и/или XHTML, если специально не оговорено иначе.

Теперь, когда вы имеете представление о назначении стилей CSS, настало время рассмотреть способы их использования в HTML-документах.

## Внутритекстовые стили

Проще всего изменять оформление элементов страницы с помощью **внутритекстовых стилей**. Этот способ заключается в присваивании HTML-элементу определенного стиля посредством атрибута `style`, как в следующем примере:

```
<p style="font-family: sans-serif; color: #3366CC;">
  Amazingly few discotheques provide jukeboxes.
</p>
```

Селекторы при этом не используются – описание стиля применяется непосредственно к элементу, в тег которого оно добавлено (в вышеприведенном примере – в тег `<p>`).

Основным недостатком данного способа является отсутствие возможности повторного использования внутритекстовых стилей. Например, чтобы применить тот же самый стиль к другому элементу `p`, нужно снова указать его в качестве значения атрибута `style`. Если в дальнейшем появится необходимость в изменении определения стиля, придется искать и редактировать все теги, в которых оно было использовано. Кроме того, размещение внутритекстовых стилей в разметке страницы затрудняет чтение и дальнейшее управление кодом.

## Встраиваемые стили

CSS-стили можно применять к веб-страницам с помощью элемента `style`, как в приведенном выше примере. Таким образом, у вас появляется возможность создать произвольное количество деклараций стилей, размещаемых между открывающим и закрывающим тегами `<style>`:

```
<style type="text/css">
  : CSS Styles...
</style>
```

Теги `<style>` расположены внутри элемента `head` веб-страницы. Атрибут `type` используется для указания языка, на котором написана таблица стилей. Единственным широко распространенным языком для создания стилей является CSS, чему соответствует значение `text/css`.

Этот простой и понятный метод размещения стилей между тегами `<style>` обладает одним недостатком: чтобы использовать определенный набор стилей на всем сайте, придется скопировать его и разместить в заголовке каждой страницы сайта.

Гораздо разумнее вынести все определения стилей в отдельный текстовый файл, а в самом документе создать ссылку на него. Такой отдельный файл называют внешней таблицей стилей.

## Внешние таблицы стилей

**Внешняя таблица стилей** – это отдельный файл (как правило, с расширением `.css`), содержащий все описания CSS-стилей для данного сайта. На один и тот же файл могут ссылаться множество страниц, и любые изменения описаний стиля будут распространяться на каждую из них. Такой метод позволяет создавать набор деклараций для целого сайта, о чем уже говорилось выше.

Для создания в документе ссылки на внешнюю таблицу стилей (к примеру, `styles.css`) достаточно разместить в его заголовке элемент `link`:

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

Помните наш первый пример, в котором для трех заголовков был определен одинаковый стиль? Реализуем то же самое с помощью ссылки на внешнюю таблицу стилей с именем `styles.css`:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
<head>
<title>A Simple Page</title>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<link rel="stylesheet" type="text/css" href="styles.css" />
</head>
<body>

<h1>First Title</h1>
<p>...</p>

<h2>Second Title</h2>
<p>...</p>

<h2>Third Title</h2>
<p>...</p>
</body>
</html>
```

Атрибуту `rel` необходимо присвоить значение `stylesheet`, а `type` – `text/css`. Атрибут `href` указывает на размещение и имя файла с таблицами стилей.

Файл `styles.css` содержит следующие описания стилей:

```
h1, h2 {
  font-family: sans-serif;
  color: #3366CC;
}
```

Подобно изображениям, файл `styles.css` можно использовать многократно. Поэтому вы избавляетесь от необходимости повторно вводить описания стилей, а также можете быть уверены, что все заголовки на сайте будут оформлены одинаково.

## Синтаксис CSS

Таблица стилей состоит из набора определений стилей. Двумя основными компонентами определения (или правила) стиля являются:

- Список из одного или более **селекторов**, разделенных запятыми. Они указывают на элемент, к которому будет применен данный стиль.
- **Блок описаний**, заключенный в фигурные скобки, в котором содержится информация о том, как данный стиль повлияет на отображение элемента.

Блок описаний состоит из одного или нескольких **описаний стиля**, каждый из которых устанавливает значение определенного **свойства**. Если описаний несколько, они отделяются друг от друга точкой с запятой (;). Описание свойства состоит из его имени и значения, разделенных двоеточием (:). Схематично все описанные элементы изображены на рис. 1.1.



*Рис. 1.1. Составные части правила CSS: список селекторов и блок описаний*

Декларация свойства определяет значения параметров шрифта, цветов и других настроек, которые будут присвоены элементу после применения стиля. Предлагаемые в данной книге решения в большинстве своем основаны на сочетании различных свойств и их значений.

На рис. 1.1 также видно, что правило стиля можно записать в одну строчку. Одни разработчики предпочитают переносить каждое новое

свойство на следующую строку – такой код легче читать; другие пишут правила в одну строчку, чтобы сэкономить место. Ниже приведены два варианта записи одного и того же стиля:

```
h1, h2 {  
  font-family: sans-serif;  
  color: #3366CC;  
}
```

```
h1, h2 { font-family: sans-serif; color: #3366CC; }
```

Форматирование не влияет на интерпретацию стилей, поэтому вы можете выбрать более близкий вам способ.

## Что такое селекторы и как их правильно использовать

В следующем примере используются два селектора – `h1` и `h2`. Это означает, что указанные стили будут применяться ко всем элементам `h1` и `h2`:

```
h1, h2 {  
  font-family: sans-serif;  
  color: #3366CC;  
}
```

### Решение

В этом разделе представлено описание широко используемых в настоящее время селекторов CSS2.1 с примерами их практического использования.

#### Селекторы типа

Самыми простыми селекторами являются **селекторы типа**, с которыми мы уже встречались в приведенных выше примерах. Адресация по имени HTML-элемента обеспечивает применение правила стиля ко всем элементам с данным именем, присутствующим в документе. Селекторы типа часто используются для определения основных стилей для всех страниц сайта. К примеру, следующее правило стиля задает шрифт абзаца, используемый на сайте по умолчанию:

```
p {  
  font-family: Tahoma, Verdana, Arial, Helvetica, sans-serif;  
  font-size: 1em;  
  color: #000000;  
}
```

Таким образом устанавливается шрифт, размер и цвет для всех абзацев (элементов `p`) в документе.

## Селекторы класса

Задавать стили для элемента, конечно, удобно, однако что делать, если возникает необходимость определения различных стилей для нескольких элементов одного типа, размещенных в различных частях страницы? На помощь придут **классы**.

Рассмотрим следующий пример, в котором задается синий цвет шрифта для всех абзацев:

```
p {
  color: #0000FF;
}
```

Отлично! А теперь представьте, что на вашей странице будет расположена боковая панель с голубым фоном. При этом голубой текст на панели сольется с ним, и разобрать что-либо станет просто невозможно. Эту проблему можно решить, определив класс для текста, выводимого на боковой панели, а затем задав для этого класса необходимый стиль с помощью CSS.

Для создания абзаца класса `sidebar` необходимо вначале добавить атрибут `class` с соответствующим значением в открывающий тег:

```
<p class="sidebar">This text will be white, as specified by the CSS style
definition below.</p>
```

Затем можно приступить к описанию стиля для данного класса:

```
p {
  color: #0000FF;
}
.sidebar {
  color: #FFFFFF;
}
```

Во втором правиле используется селектор класса, определяющий стиль для любого элемента со значением `sidebar` атрибута `class`. Перед именем ставится точка, указывающая, что это класс, а не HTML-элемент.

А если на панели также присутствуют ссылки? По умолчанию все ссылки в вашем документе будут оформлены одинаковым образом. Однако, если добавить в тег ссылки атрибут `class="sidebar"`, они также станут белыми:

```
<p class="sidebar">This text will be white, <a class="sidebar"
href="link.html">and so will this link</a>.</p>
```

Прекрасно. Но, возможно, вам хотелось бы, чтобы ссылки отличались от основного текста? К примеру, им можно было бы придать жирное начертание. Добавление описания стиля для жирного начертания текста к набору стилей для класса `sidebar` приведет к тому, что весь текст

на панели будет выведен жирным шрифтом, что противоречит поставленной нами цели. Нужен CSS-селектор для выбора ссылок, размещенных внутри элемента класса `sidebar`. Далее путем комбинации селектора типа и селектора класса можно добиться желаемого эффекта:

```
p {
  color: #0000FF;
}
.sidebar {
  color: #FFFFFF;
}
a.sidebar:link, a.sidebar:visited {
  font-weight: bold;
}
```

Обратите внимание, что в данном примере используются псевдоклассы `:link` и `:visited`. Мы поговорим о псевдоклассах чуть позже в этой главе.

Если вы добавите приведенные выше стили в таблицу стилей вашего сайта и обновите страницу в браузере, то увидите, что ссылки будут отображены жирным шрифтом белого цвета, поскольку к ним были применены оба определенных для класса `sidebar` правила стилей. Однако если бы мы добавили в третье правило стиль изменения цвета шрифта, цвет ссылок также был бы изменен, поскольку третий селектор более конкретен, а правила CSS применяются в порядке увеличения конкретности селекторов.

Кстати, процесс применения различных правил стилей к одному и тому же элементу называется **каскадированием**; отсюда и термин «каскадные таблицы стилей». Ближе к концу главы мы узнаем о степени конкретности различных селекторов и правилах каскадирования.

## Селекторы ID

В отличие от селекторов класса, определяющих стиль для группы элементов, **селекторы ID** используются для присваивания стиля одному конкретному элементу. Чтобы воспользоваться таким селектором, необходимо прежде всего добавить атрибут `id` элементу, стиль которого должен быть изменен. Важно помнить, что идентификатор (ID) в HTML-документах уникален:

```
<p id="tagline">This paragraph is uniquely identified by the ID "tagline".</p>
```

В качестве ссылки на данный элемент (посредством селектора ID) необходимо указать идентификатор с предшествующим ему знаком «решетка» (`#`). К примеру, следующее правило стиля задает белый цвет шрифта для приведенного выше абзаца:

```
#tagline {
  color: #FFFFFF;
}
```



Селекторы ID можно использовать совместно с селекторами других типов. В следующем примере правило стиля применяется ко всем элементам класса `new`, расположенным внутри абзаца со значением `tagline` атрибута `id`:

```
#tagline .new {
  font-weight: bold;
  color: #FFFFFF;
}
```

Использованный в данном примере селектор называется контекстным; селекторам этого типа посвящен следующий раздел.

## Контекстные селекторы

Если на панели расположено несколько абзацев, *можно* добавить атрибут класса каждому открывающему тегу `<p>`. Однако гораздо лучше указать атрибут `class` со значением `sidebar` для элемента, служащего контейнером, а затем изменить цвет каждого расположенного внутри контейнера элемента `p` на белый. Для этого достаточно написать всего одно правило стиля с использованием **контекстного селектора (селектора-потомка)**.

Ниже приведено правило с новым селектором:

```
p {
  color: #0000FF;
}
.sidebar p {
  color: #FFFFFF;
}
```

А так выглядит обновленный HTML-код:

```
<div class="sidebar">
  <p>This paragraph will be displayed in white.</p>
  <p>So will this one.</p>
</div>
```

Как видите, контекстный селектор состоит из списка селекторов, разделенных пробелами, соответствующих элементам страницы от *внешних к внутренним*. В данном случае на странице имеется элемент `div` класса `sidebar`, и контекстный селектор `.sidebar p` ссылается на все элементы `p`, расположенные внутри этого `div`.

## Дочерние селекторы

Контекстный селектор ссылается на всех потомков родительского элемента, в том числе и *непрямых* потомков.

Рассмотрим следующий код:

```
<div class="sidebar">
  <p>This paragraph will be displayed in white.</p>
```

```
<p>So will this one.</p>
<div class="tagline">
  <p>If we use a descendant selector, this will be white too. But if we
    use a child selector, it will be blue.</p>
</div>
</div>
```

В данном примере используется контекстный селектор из предыдущего раздела – `.sidebar p`. Он ссылается на все элементы `p`, вложенные в `div` класса `.sidebar p`, *включая* и абзацы, расположенные внутри `div` класса `tagline`. Чтобы указанное оформление распространялось исключительно на *прямых* потомков элемента `div` класса `sidebar`, следует использовать **дочерний селектор**. Для указания прямого потомка в селекторе используется знак `>`.

В приведенном ниже коде используется новый селектор для задания белого цвета текста абзацев, расположенных непосредственно внутри элемента `div` класса `sidebar` (но не внутри элемента `div` класса `tagline`):

```
p {
  color: #0000FF;
}
.sidebar>p {
  color: #FFFFFF;
}
```

---

### Предупреждение

**Internet Explorer 6 не поддерживает дочерние селекторы.** Поэтому в том случае, если задаваемый с их помощью стиль важен для форматирования страницы, необходимо продумать альтернативные способы обращения к требуемому элементу.

---

### Смежные селекторы

Смежные селекторы ссылаются на элемент, расположенный непосредственно после другого обозначенного элемента. Таким образом, если на странице имеется следующий HTML-код:

```
<h2>This is a title</h2>
<p>This paragraph will be displayed in white.</p>
<p>This paragraph will be displayed in black.</p>
```

А затем мы используем следующий селектор:

```
p {
  color: #000000;
}
h2+p {
  color: #FFFFFF;
}
```

При этом только первый абзац отображается в белом цвете. Второй элемент `p` не является смежным с `h2` и потому его текст выводится черным шрифтом, что указано в первом правиле для элементов `p`.

### Предупреждение

**Internet Explorer 6 не поддерживает смежные селекторы.** Потому, если задаваемый с их помощью стиль важен для форматирования страницы, следует найти альтернативный способ обращения к требуемому элементу.

## Селекторы псевдоклассов для ссылок

HTML предлагает гораздо более широкие возможности для форматирования ссылок (создаваемых с помощью элемента `a` или **якоря**) по сравнению с остальными элементами. С помощью CSS можно задавать стиль отображения ссылок в зависимости от их состояния – посещенных или непосещенных – при наведении на них указателя мыши или при щелчке по ней. Рассмотрим следующий пример:

```
a:link { color: #0000FF; }
a:visited { color: #FF00FF; }
a:hover { color: #00CCFF; }
a:active { color: #FF0000; }
```

Приведенный выше код содержит четыре определения стилей CSS. При этом с каждым из селекторов используется так называемый **псевдокласс** элемента `a`. Псевдокласс по сути является ярлычком из набора, который может быть добавлен к селектору для указания на состояние соответствующего элемента. Между селектором и псевдоклассом ставится двоеточие (:). Ниже представлены широко используемые псевдоклассы для ссылок:

- `:link` распространяет свое действие исключительно на *непосещенные* ссылки и задает для них синий цвет шрифта.
- `:visited` действует на *посещенные* ссылки и выводит их в фиолетовом цвете.
- `:hover` окрашивает ссылки в светло-голубой цвет при наведении на них указателя мыши вне зависимости от того, были они посещены или нет.
- `:active` изменяет цвет ссылки на красный после щелчка по ней.

Важное значение имеет порядок следования селекторов псевдоклассов в таблице стилей. В данном случае `:active` стоит на последнем месте; это означает, что его действие имеет более высокий приоритет над перечисленными ранее тремя описаниями, и потому его применение не зависит от того, были ли ссылки посещены или нет, навел ли пользователь на них указатель мыши или нет.

Состояния `:hover` и `:active` формально называют динамическими селекторами псевдоклассов, поскольку они появляются только при взаимодействии пользователя с соответствующими элементами путем наведения указателя мыши и щелчка по ссылке соответственно.

### Примечание

**Применение `:hover` с другими элементами.** Динамический селектор псевдокласса `:hover`, помимо ссылок, можно использовать с другими элементами, что позволяет создавать эффекты вроде подсветки ряда таблицы при наведении на него указателя мыши. Однако Internet Explorer 6 и более ранние версии поддерживают использование данного псевдокласса исключительно с элементами ссылок.

## Псевдокласс `:first-child`

Еще одним примером псевдокласса является `:first-child`. В то время как смежный селектор ссылается на элемент, *следующий за* указанным элементом в исходном коде документа, селектор псевдокласса `:first-child` ссылается на первый по порядку дочерний элемент обозначенного родителя. Таким образом, единственным отличием в использовании данных элементов является то, что в последнем случае поставленному условию удовлетворяет только первый дочерний элемент:

```
<div class="article">
  <p>
    This is an intro paragraph to be
    displayed with a larger font size.
  </p>
  <p>
    Here is a second paragraph of text
    displayed at normal text size.
  </p>
</div>
```

CSS-код будет выглядеть следующим образом:

```
p {
  font-size: 100%
}
.article p:first-child {
  font-size: 160%;
}
```

Поскольку первый абзац является первым дочерним элементом родительского блока `div` класса `article`, его текст будет отображаться более крупным шрифтом. Размер шрифта второго абзаца соответствует определенному для всех элементов `p` в документе.

### Предупреждение

**Internet Explorer 6 не поддерживает псевдокласс `:first-child`.** Как оказалось, браузер IE6 не поддерживает и селектор псевдокласса `:first-child`, поэтому, если задаваемый с их помощью стиль важен для форматирования страницы, следует найти альтернативный способ адресации.

## Каким образом браузер определяет, какие стили нужно использовать

Как браузер может «понять» намерения разработчика? Если к одному и тому же элементу могут быть применены несколько стилей, то определение, какие из них будут использованы, происходит по принципу **каскадирования**.

Принцип каскадирования чрезвычайно важен для понимания CSS, поскольку большинство ошибок разработки, связанных с таблицами стилей, происходят из-за того, что применение стиля не соответствует намерениям разработчика. В данной главе мы уже рассмотрели пример, в котором было определено общее правило стиля для элементов типа абзац, а также более конкретное правило для одного или нескольких определенных абзацев. Оба правила предназначены для изменения оформления абзацев, *однако более конкретные правила имеют приоритет над более общими.*

### Решение

На выбор применяемого браузером стиля влияют четыре фактора: вес, источник, конкретность и порядок следования.

**Вес** определенного описания стиля можно определить с помощью ключевого слова `!important`, добавляемого после значения свойства. При этом свойство приобретает приоритетное значение над аналогичными свойствами, указанными в других правилах стиля, за исключением редких случаев. Применение ключевого свойства `!important` сильно усложняет процесс поддержки кода; кроме того, необходимость в его использовании возникает не слишком часто. Ввиду приведенных причин рекомендуется по возможности избегать его, как в примерах данной книги. Дополнительную информацию о данном ключевом слове можно найти в документации SitePoint CSS Reference.<sup>1</sup>

Существуют три возможных **источника** стилей – они могут быть определены браузером, разработчиком или пользователем. В данной книге нас интересуют **таблицы стилей, написанные разработчиком** веб-страницы, т. е. вами. Мы уже говорили, что у браузера есть своя внутренняя таблица стилей, определяющая оформление всех элементов по умолча-

<sup>1</sup> <http://reference.sitepoint.com/css/importantdeclarations/>

нию, однако стили, написанные разработчиком, всегда имеют более высокий приоритет и переопределяют настройки по умолчанию. Кроме того, возможно применение **пользовательской таблицы стилей** – набора стилей, созданного пользователями, однако и они, за исключением редких случаев, переопределяются таблицей стилей разработчика. Источникам стилей посвящен целый раздел документации SitePoint CSS Reference.<sup>1</sup>

Помимо этого, влияние каскадирования на поведение стилей CSS зависит от степени конкретности свойств и порядка их следования.

Правило стиля с более конкретным селектором переопределяет стили, задаваемые правилом с более общим селектором. Рассмотрим этот механизм на практическом примере со следующим несложным HTML-кодом:

```
<div id="content">
  <p class="message">
    This is an important message.
  </p>
</div>
```

А теперь обратите внимание на правила стилей, применяемых к приведенной выше разметке:

```
p { color: #000000; }

.message { color: #CCCCCC; }

p.message { color: #0000FF; }

#content p.message { color: #FF0000; }
```

В данном примере действие всех четырех селекторов, задающих цвет шрифта, направлено на элемент `p`. Какой же цвет будет использован? Совершенно верно, красный (`#FF0000`). Селекторы `p` (любой элемент `p`) и `.message` (любой элемент класса `message`) имеют одинаковую степень значимости; приоритет селектора `p.message` (любой элемент `p` класса `message`) выше, но самым высоким приоритетом обладает селектор `#content p.message` (любой элемент `p` класса `message`, являющийся дочерним для элемента со значением `content` атрибута `id`).

Однако длину селекторов не стоит рассматривать как показатель степени конкретности. К примеру, селектор ID всегда будет иметь более высокий приоритет, чем селектор типа или класса. Чем сложнее используемые вами селекторы, тем труднее разобраться в степени их важности, при этом приведенные в данной книге примеры достаточно про-

---

<sup>1</sup> <http://reference.sitepoint.com/css/cascade/>

сты. Точную формулу измерения степени конкретности вы можете найти в документации SitePoint CSS Reference.<sup>1</sup>

Если, несмотря на вышеперечисленные факторы, остается несколько стилей, которые могут быть применены по отношению к одному и тому же элементу, в расчет принимается **порядок следования** правил – в этом случае будет использовано последнее из них. В приведенном выше примере степень значимости селекторов `p` и `.message` одинакова, поэтому при отсутствии иных регулирующих принципов используется правило стиля, записанное последним, – в данном случае с селектором `.message`. Так же дело обстоит и при объявлении в таблице стилей нескольких правил с одним и тем же селектором – например, `.message`. В этом случае будет применено правило, определенное вторым. В последующих главах мы увидим, что такой механизм работы можно эффективно использовать для достижения своих целей.

## Заключение

В данной главе вы познакомились с основными принципами и способами использования CSS. Мы даже затронули такую сложную тему, как правила каскадирования. Даже если вы никогда раньше не работали с CSS, но разобрались с понятиями, представленными в данной главе, этого будет достаточно для понимания приводимых далее примеров.

Так как примеры, представленные в начале книги, проще, чем примеры в конце книги, то, если вы новичок в области применения CSS, рекомендую вам начать чтение с первых глав. Таким образом вы сможете пополнить знания, приобретенные в первой главе, и приступить к практике, погрузившись в увлекательный (надеюсь) мир каскадных таблиц стилей.

---

<sup>1</sup> <http://reference.sitepoint.com/css/specificity>

# 2

## Оформление текста и другие базовые возможности

В данной главе мы рассмотрим основные способы использования CSS для задания стилового оформления текста, а также прочие основные возможности; вы найдете ответы на наиболее часто задаваемые вопросы об их применении. Если вы новичок в области CSS, то с помощью приведенных примеров вы узнаете о существовании многих свойств и о методах их использования, что создаст необходимую базу для написания собственного кода. Если вы уже хорошо знакомы с технологией CSS, то эта глава сможет подсказать вам необходимые приемы, если в процессе разработки вы вдруг обнаружите, что забыли, как достичь того или иного эффекта.

Приведенные примеры поддерживаются подавляющим большинством браузеров, однако следует помнить о важности тестирования написанного кода в различных версиях браузеров. Хотя некоторые несоответствия или отсутствие поддержки предлагаемых методов могут встречаться в более ранних версиях браузеров, в целом представленные решения не должны вызывать каких-то серьезных проблем.

### Задание определенного шрифта для текста

#### Решение

Задать определенный шрифт для оформления текста можно с помощью свойства `font-family`, как показано в следующем примере:

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



## Обсуждение

С помощью CSS можно задавать как определенный шрифт, так и семейства шрифтов:

- serif
- sans-serif
- monospace
- cursive
- fantasy

При этом необходимо помнить, что на компьютере пользователя может не оказаться тех шрифтов, что установлены у вас. Если указанный шрифт отсутствует, текст будет оформлен с помощью используемых браузером по умолчанию шрифтов, вне зависимости от определенных вами настроек.

Во избежание подобных недоразумений можно просто-напросто задать название семейства шрифтов, а система пользователя сама определит, какой шрифт следует применить. К примеру, если вы хотите оформить текст шрифтом из семейства sans-serif, к которому относится, например, Arial, можно использовать следующее правило стиля:

```
p {
  font-family: sans-serif;
}
```

Вы хотели бы иметь более гибкую возможность управлять внешним видом вашего сайта? Это возможно. В одном блоке описаний можно определить как названия конкретных шрифтов, так и семейств. В качестве примера рассмотрим следующий код, содержащий правило стиля для элемента p:

```
p {
  font-family: Verdana, Geneva, Arial, Helvetica, sans-serif;
}
```

В данном случае указано, что при наличии в системе шрифта Verdana следует использовать именно его; в противном случае браузер должен проверить, установлен ли в системе шрифт Geneva; если и его не окажется, будет осуществлен поиск Arial, а затем Helvetica. При отсутствии в системе всех перечисленных шрифтов браузер должен применить используемый в системе по умолчанию шрифт из семейства шрифтов sans-serif.

Если в названии семейства шрифтов присутствуют пробелы, его следует заключить в кавычки, как показано ниже:

```
p {
  font-family: "Courier New", "Andale Mono", monospace;
}
```

Общее название семейства шрифтов никогда не заключается в кавычки и всегда приводится в самом конце списка.

Вы можете абсолютно спокойно использовать на своих страницах следующие шрифты:

**Windows** Arial, Lucida, Impact, Times New Roman, Courier New, Tahoma, Comic Sans, Verdana, Georgia, Garamond

**Mac** Helvetica, Futura, Bodoni, Times, Palatino, Courier, Gill Sans, Geneva, Baskerville, Andale Mono

Одного взгляда на данный список достаточно, чтобы понять, почему в нашем правиле стиля мы определили именно такие шрифты: в начале указан наиболее предпочтительный для нас шрифт **Verdana**, широко распространенный в **Windows**, затем **сходный шрифт для Mac – Geneva**. Далее приводятся другие шрифты, которые будут использованы, если ни один из первых двух шрифтов не будет найден в системе пользователя.

## Выбор единиц измерения размера шрифтов: пиксели, пункты, пики или что-то другое

Размер шрифта в CSS задается с помощью свойства `font-size`, например:

```
font-size: 12px;
```

В данном примере приводится размер в пикселях, но для определения значения данного свойства можно использовать и другие единицы измерения. Прежде чем сделать выбор в пользу одного или другого из них, следует взвесить все аргументы за и против.

### Решение

#### Единицы измерения для задания размера шрифтов

В табл. 2.1 приведены единицы, которые можно использовать для задания размера шрифта.

Таблица 2.1. Единицы измерения для задания размера шрифта

Идентификатор	Единица измерения
pt	пункты
pc	пики
px	пиксели
em	относительная единица измерения, равная значению свойства <code>font-size</code> заданного шрифта
ex	относительная единица измерения, равная высоте строчной буквы «x» заданного шрифта
%	проценты

Рассмотрим применение каждого из них более подробно.

### Пункты и пики

```
p {  
  font-size: 10pt;  
}
```

Следует избегать использования **пунктов** и **пик** для оформления текста, предназначенного для отображения на экране. Эти единицы идеально подходят для задания размера шрифта для печати; измерение в пунктах пришло из полиграфии. Пункт составляет  $1/72$  часть дюйма, а пика – шестую часть дюйма. После печати текстовые документы, шрифт в которых задан в данных единицах, будут в точности соответствовать намерениям автора – в конечном счете шестая часть дюйма остается шестой частью дюйма как на листе формата A4, так и на ватмане. Однако компьютеры не в состоянии отобразить физические размеры с такой точностью, и они пытаются угадать – причем не слишком успешно – размер пункта или пика, из-за чего под различными платформами один и тот же документ может отображаться по-разному.

Если вы пишете таблицу стилей для печати (чем мы займемся в главе 8, в разделе «Создание таблицы стилей для печати») или же сам документ предназначен для печати, а не для просмотра с экрана, следует использовать пункты и пики. Однако базовое правило при дизайне страницы для отображения в веб-среде: избегайте их применения.

### Пиксели

```
p {  
  font-size: 12px;  
}
```

Многие дизайнеры любят использовать **пиксели** в качестве единиц измерения для задания размера шрифта, поскольку это позволяет добиться одинакового отображения в различных браузерах и под разными платформами. Однако при этом игнорируются настройки браузера пользователя; более того, при просмотре такой страницы в Internet Explorer пользователь не сможет изменить размер шрифта. Это представляет серьезные ограничения для пользователей со слабым зрением, для которых возможность прочтения зависит от наличия функции увеличения шрифта.

Может показаться, что использование пикселей для задания размера шрифта – самое простое и удобное решение, однако при наличии альтернативных методов следует отказаться от этого способа, в особенности если речь идет о больших блоках с контентом. При создании документа, предназначенного для печати, или написании таблицы стилей для печати совершенно нецелесообразно использовать пиксели – в мире бумажных документов они полностью теряют смысл, подобно

пунктам в экранной среде. При указании размера шрифта текста для печати в пикселах будет предпринята попытка угадать, каким образом он должен отображаться на бумаге, и результаты далеко не всегда совпадут с ожидаемыми.

## Em

**Em** – относительная единица измерения размеров шрифта. Ее название пришло из области типографии, где оно соответствует размеру заглавной буквы **M**, которая, как правило, является самым широким символом шрифта. В CSS `1em` соответствует размеру шрифта, используемому в системе пользователя по умолчанию, или размеру шрифта родительского элемента, если он отличается от используемого по умолчанию.

Если вы используете **em** (или другие относительные единицы) для задания размера всех шрифтов, пользователи смогут изменять размер символов текста в соответствии с настройками размера символов, установленными в их браузере. Для примера создадим описание стиля, задающее размер шрифта внутри элемента `p` равным `1em`:

```
p {
  font-size: 1em;
}
```

При просмотре страницы в браузере **Internet Explorer 8**, в котором размер шрифта установлен на **Medium**, данный абзац будет выглядеть, как на рис. 2.1.

Если в браузере настройка размера шрифта – **Largest**, то абзац с размером шрифта в `1em` будет выглядеть, как на рис. 2.2.

Использование **em** для задания размера шрифта ограничивает ваши возможности по управлению отображением документа. Однако этот подход означает, что посетитель, которому необходим крупный шрифт, сможет прочесть ваш контент, ведь в конечном счете именно для этого вы и размещаете текст на страницу.

Значения в **em** можно задавать десятичными числами. К примеру, чтобы задать размер шрифта на 10 процентов меньше используемого по умолчанию (или размера шрифта родительского элемента), можно использовать следующее правило:

```
p {
  font-size: 0.9em;
}
```

Чтобы текст стал на 10 процентов крупнее, чем при использовании значения по умолчанию или унаследованного значения, можно использовать следующее правило:

```
p {
  font-size: 1.1em;
}
```

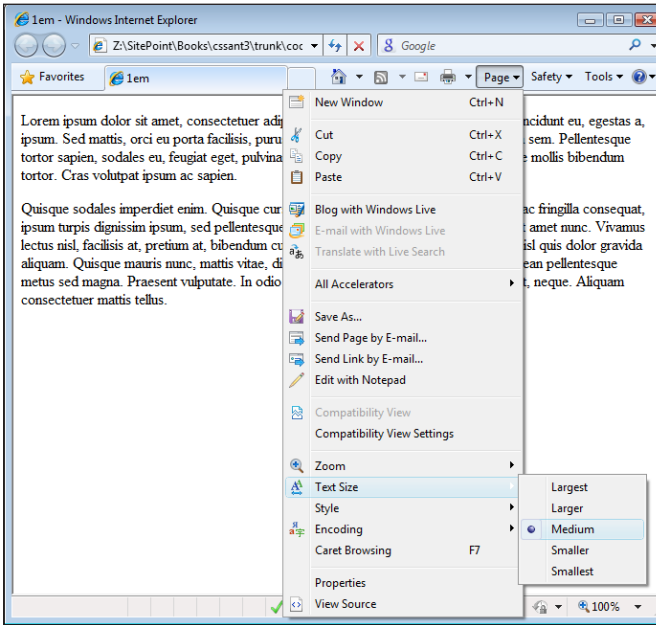


Рис. 2.1. Отображение абзаца при установке свойства font-size равным 1em и text size – Medium

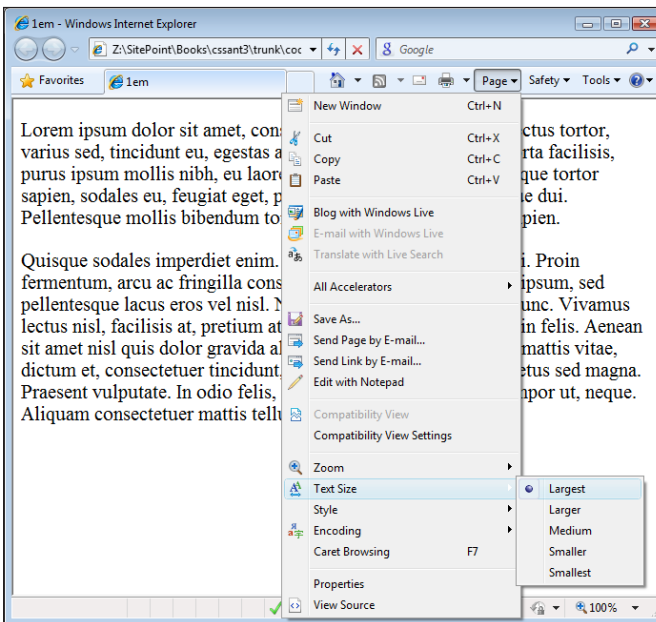


Рис. 2.2. Отображение абзаца при установке свойства font-size равным 1em и text size – Largest

## Ех

**Ех** – относительная единица измерения размеров, соответствующая высоте строчной буквы «х» шрифта по умолчанию. Теоретически, если присвоить свойству `font-size` абзаца значение `1ex`, его заглавные буквы будут той же высоты, какую имела бы строчная буква `x`, если бы размер шрифта не был задан (при этом размер строчных букв рассчитывался бы относительно заглавных).

К сожалению, современные браузеры еще не поддерживают необходимые для корректного отображения `ex` типографские функции – как правило, в процессе измерений используется достаточно грубый расчет.

### Значение в процентах

```
p {
  font-size: 100%;
}
```

Как и в случае с `em` и `ex`, при задании размеров **в процентах** размеры будут соотноситься с настройками пользовательского браузера, и пользователь сможет самостоятельно изменять размер шрифта. Значение **100%** размера шрифта для элемента `p` соответствует размеру шрифта, заданному настройками по умолчанию (так же, как и при задании значения `1em`). При уменьшении процентного значения размер текста уменьшается:

```
p {
  font-size: 90%;
}
```

При увеличении процентного значения размер текста увеличивается:

```
p {
  font-size: 150%;
}
```

## Изменение размера шрифта с помощью ключевых слов

Размер шрифта можно задать, помимо присваивания числовых значений, посредством ключевых слов с абсолютными и относительными значениями.

### Ключевые слова с абсолютными значениями

В распоряжении веб-разработчика есть семь абсолютных ключевых слов CSS:

- `xx-small`
- `x-small`
- `small`

- medium
- large
- x-large
- xx-large

Значения этих ключевых слов определены одни относительно других, и разные браузеры интерпретируют их по-своему. Большинство браузеров отображают текст с размером `medium` аналогично тексту с настройками по умолчанию, а остальные ключевые слова изменяет размер текста относительно `medium` в соответствии с их названиями.

Измерения с помощью ключевых слов называются абсолютными, поскольку при этом отсутствует наследование от родительского элемента. Тем не менее в отличие от применения абсолютных значений, например в пикселах или пунктах, использование абсолютных ключевых слов позволяет посетителю сайта изменять размер шрифта вручную, а также не переопределяет пользовательские настройки браузера. Основной проблемой при применении абсолютных ключевых слов становится несоответствие отображения текста, оформленного с их помощью, в разных браузерах – текст с размером `xx-small` может быть отчетливо виден в одном браузере и совершенно нечитаем в другом. Internet Explorer 6 в режиме совместимости, к примеру, интерпретирует значение `small` в соответствии с настройками по умолчанию. Мы рассмотрим режим совместимости более подробно в разделе «Что такое режим совместимости и как его избежать» в главе 7.

### Ключевые слова с относительными значениями

При задании размера шрифта с помощью относительных ключевых слов – `larger` и `smaller` – он определяется по отношению к размеру, установленному для родительского элемента, подобно тому, как это происходит при использовании `em` и `%`. Таким образом, если размер элемента `p` задан с помощью абсолютного ключевого слова `small`, и вы хотите, чтобы выделенный фрагмент текста отображался более крупным шрифтом, можно использовать следующую таблицу стилей:

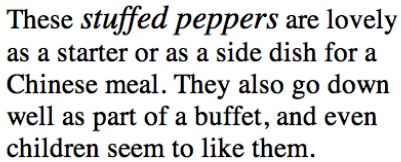
*chapter02/relative.css*

```
p {
    font-size: small;
}
em {
    font-size: larger;
}
```

Данная разметка будет отображена, как показано на рис. 2.3, поскольку текст между тегами `<em>` и `</em>` отображается более крупным шрифтом, чем текст родительского элемента `p`:

*chapter02.relative.html (фрагмент)*

```
<p>These <em>stuffed peppers</em> are lovely as a starter or as a side dish  
for a Chinese meal. They also go down well as part of a buffet, and even  
children seem to like them.</p>
```



These *stuffed peppers* are lovely as a starter or as a side dish for a Chinese meal. They also go down well as part of a buffet, and even children seem to like them.

*Рис. 2.3. Выделенный текст отображается более крупным шрифтом по сравнению с содержащим его абзацем*

## Обсуждение

При выборе метода задания размера шрифта рекомендуется выбирать такой, который даст пользователям возможность масштабирования текста и обеспечит соответствие текста установленным настройкам браузера. Относительное задание шрифтов является оптимальным вариантом, однако этот метод требует внимательного контроля за наследованием размеров элементами. В любом случае, чтобы сделать ваши страницы доступными для различных категорий пользователей, в том числе и для людей с ограниченными возможностями, необходимо обеспечить наличие возможности установки предпочтительного для них размера шрифта.

При разработке дизайна страницы следует помнить об этой необходимости; это также позволит вам избежать проблемы, возникающей в браузерах, поддерживающих возможность изменения размера шрифта, заданного в пикселях, когда дизайнер исходит из предположения, что это позволит ему зафиксировать высоту контейнеров или разместить текст над изображением с фиксированной высотой. Этот метод прекрасно работает при условии просмотра страницы в браузере **Internet Explorer**, который не изменяет размер текста, указанный в пикселях; однако в Firefox (3-я версия со значением **zoom text only** настройки **Zoom** и более ранние версии) все это может превратиться в мешанину из перекрывающихся друг друга кусков текста, поскольку высота контейнеров для текста никогда не известна.

## Относительное изменение размеров и наследование

При применении любых методов относительного определения размеров элемента следует помнить о том, что элемент наследует это значение от родителя, и уже это значение следует подгонять для достижения требуемого результата. При этом следует с осторожностью использовать относительное задание размеров родительского элемента, поскольку при



сложной организации контейнеров может быть непросто определить путь наследования. Рассмотрим следующий код:

*chapter02/nesting.html (фрагмент)*

```
<div>
  <p>
    You'll <em>probably</em> be surprised when using
    <a href="#">a relative <code>font-size</code></a>
    and nested elements.
  </p>
</div>
```

Допустим, для свойства `font-size` вышеприведенного текста требовалось задать значение **130%** по отношению к размеру по умолчанию, и мы ошибочно сделали это следующим образом:

*chapter02/nesting.css (фрагмент)*

```
div, p, em, a, code {
  font-size: 130%;
}
```

В результате размер шрифта вложенных элементов станет крупнее, составив **130%** от размера родительского элемента, который, в свою очередь, составляет **130%** от размера *своего* родительского элемента и т. д., как показано на рис. 2.4.<sup>1</sup>

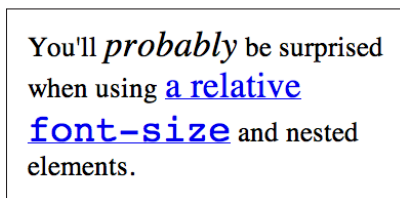


Рис. 2.4. Относительное задание размера шрифта для вложенных элементов

## Удаление подчеркивания ссылок

Основным указанием на то, что перед нами ссылка, является то, что она подчеркнута и ее цвет изменен по сравнению с основным текстом. Такое визуальное оформление используется по умолчанию. Однако в определенных случаях может возникнуть необходимость в отображении ссылки без подчеркивания.

### Решение

Для того чтобы убрать подчеркивание ссылок, воспользуемся свойством `text-decoration`. По умолчанию браузер устанавливает значение

---

<sup>1</sup> Цветные иллюстрации доступны по адресу [www.symbol.ru/library/css-101-tips](http://www.symbol.ru/library/css-101-tips)

`underline` данного свойства для всех элементов `a`. Поэтому нам достаточно всего лишь установить его значение `none` для ссылок:

```
text-decoration: none;
```

Такой результат был достигнут с помощью следующего CSS-кода:

*chapter02/textdecoration.css*

```
a:link, a:visited {
    text-decoration: none;
}
```



*Рис. 2.5. Использование свойства `text-decoration` позволяет убрать подчеркивание для ссылок*

## Обсуждение

Помимо `underline` и `none` свойство `text-decoration` может принимать следующие значения:

- `overline`
- `line-through`
- `blink`

Эти значения можно комбинировать. К примеру, если бы вам захотелось оформить ссылку с помощью как нижнего, так и верхнего подчеркивания, как показано на рис. 2.6, то можно было бы использовать следующее правило стиля:

*chapter02/textdecoration2.css*

```
a:link, a:visited {
    text-decoration: underline overline;
}
```



*Рис. 2.6. Комбинирование различных значений свойства `text-decoration` позволяет создавать ссылки с нижним и верхним подчеркиванием*

---

### Предупреждение

**Не вводите пользователей в заблуждение.** Свойство `text-decoration` можно использовать для создания эффекта подчеркивания по отношению к любому тексту, не только к ссылкам, но это следует делать с большой осторожностью. Поскольку подчеркивание ссылок является очень распространенным и общепринятым приемом, пользователи склонны воспринимать *любой* подчеркнутый текст как ссылку на другой документ.

---

### Внимание

**В каких случаях подчеркивание следует оставить?** Подчеркивание ссылок применяется всеми браузерами, и, следовательно, пользователи к этому привыкли. Убирая подчеркивание ссылок в тексте, вы усложняете пользователю задачу обнаружения ссылок среди прочего выделенного текста. Я рекомендую воздержаться от использования не подчеркнутых ссылок в тексте. Существует множество способов привлекательного оформления ссылок, и действительная необходимость в удалении подчеркивания может возникнуть лишь в самых редких случаях.

Совсем другое дело, когда мы имеем дело с ссылками в меню или ссылками, расположенными иным образом так, что не остается никаких сомнений в их предназначении – например, когда их текст оформлен в виде кнопок. В этом случае при желании можно убрать подчеркивание, поскольку множество других факторов указывает на то, что данный текст является ссылкой.

---

## Создание ссылки, меняющей цвет при наведении на нее указателя мыши

Можно создать привлекательный эффект изменения цвета или иных параметров ссылки при наведении на нее указателя мыши. Его можно с успехом использовать в созданных на основе CSS панелях навигации; он применим и по отношению к ссылкам в тексте.

### Решение

Для создания описанного эффекта необходимо воспользоваться динамическими псевдоклассами якоря `:hover` и `:active`.

Рассмотрим следующий пример. Ниже представлено типичное правило стиля, в котором описания всех псевдоклассов якоря одинаковы:

*chapter02/textdecoration3.css*

```
a:link, a:visited, a:hover, a:active {
  text-decoration: underline;
  color: #6A5ACD;
  background-color: transparent;
}
```

Если мы применим данное правило, ссылки будут отображаться синим цветом (#6A5ACD) с подчеркиванием, как на рис. 2.7.



*Рис.2.7. Использование одинакового описания для всех псевдоклассов ссылок*

Для выполнения поставленной задачи нам нужно удалить селекторы псевдоклассов `:hover` и `:active` из общего описания и задать каждому из них свое собственное описание. В приведенном ниже CSS-коде к нижнему подчеркиванию добавлено еще и верхнее. Кроме того, в нем задан фоновый цвет и более темный цвет для текста ссылки. На рис. 2.8. показан результат обработки данного кода браузером:

*chapter02/textdecoration4.css*

```
a:link, a:visited {
    text-decoration: underline;
    color: #6A5ACD;
    background-color: transparent;
}
a:hover, a:active {
    text-decoration: underline overline;
    color: #191970;
    background-color: #C9C3ED;
}
```

Как вы уже наверняка догадались, остальным псевдоклассам также можно задавать отдельные описания. В частности, вы можете захотеть использовать специальное оформление для посещенных ссылок. Для этого нужно составить отдельное описание для псевдокласса `:visited`.



*Рис. 2.8. Наведение указателя мыши на ссылку, для которой указан специальный стиль при данном событии*

При определении стиля для псевдоклассов избегайте изменения размера или начертания шрифта. В противном случае пользователь будет на-

блюдать постоянное дрожание страницы, ведь окружающий контент будет отодвигаться, чтобы освободить место для отображения более крупного текста при наведении на него указателя мыши.

### Совет

**Порядок следования описаний псевдоклассов.** Псевдоклассы якоря должны следовать в порядке `:link`, `:visited`, `:hover`, `:active`. В противном случае результат обработки кода может отличаться от ожидаемого. Для запоминания часто используется слово `LoVeHATe`.<sup>1</sup>

## Использование на одной странице различных стилей ссылок

В предыдущей секции мы рассмотрели способ стилового оформления различных селекторов элемента `a`, но что делать в том случае, если разные ссылки в одном и том же документе должны выглядеть по-разному? Например, ссылки на панели навигации должны отображаться без подчеркивания, но ссылки в тексте должны быть легко распознаваемы. Или же фон одной части документа темнее, чем фон другой, и здесь ссылки должны быть светлее.

### Решение

Попробуем создать различные стили для ссылок на примере, в котором мы уже определили один обычный стиль для них:

*chapter02/linktypes.css (фрагмент)*

```
a:link, a:visited {
    text-decoration: underline;
    color: #6A5ACD;
    background-color: transparent;
}

a:hover, a:active {
    text-decoration: underline overline;
    color: #191970;
    background-color: #C9C3ED;
}
```

Данные правила будут использоваться по умолчанию – они описывают стиль отображения обычных ссылок в вашем документе. Первое правило задает синий цвет ссылок, поэтому, если часть страницы будет залита синим фоном, их будет не видно. Следовательно, необходимо создать

---

<sup>1</sup> ЛюбoВь-НенАвисть – Прим. перев.

второй набор стилей, который будет применяться к ссылкам, расположенным в данной области.

Прежде всего, создадим class или id для элемента, в котором будут размещаться ссылки другого цвета. Если стиль контейнера уже задан средствами CSS, у него наверняка есть атрибут class или id, который мы можем использовать для выполнения поставленной задачи. Допустим, документ содержит следующую разметку:

*chapter02/linktypes.html (фрагмент)*

```
<div class="boxout">
  <p>Visit our <a href="store.html">online store</a>, for all your widget
  needs.</p>
</div>
```

Создадим правило стиля, применимое по отношению ко всем ссылкам внутри элемента класса boxout:

*chapter02/linktypes.css (фрагмент)*

```
.boxout {
  color: #FFFFFF;
  background-color: #6A5ACD;
}
.boxout a:link, .boxout a:visited {
  text-decoration: underline;
  color: #E4E2F6;
  background-color: transparent;
}
.boxout a:hover, .boxout a:active {
  background-color: #C9C3ED;
  color: #191970;
}
```

Как видно на рис. 2.9, теперь все ссылки в документе будут отображены в соответствии с первым правилом, за исключением ссылок, размещенных внутри элемента div класса boxout – цвет текста последних будет более светлым.

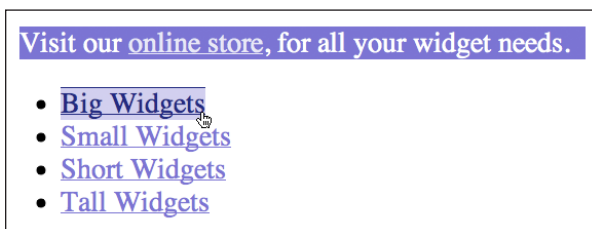


Рис. 2.9. Использование двух различных стилей ссылок в одном документе

## Присваивание первому элементу в списке отличного от последующих элементов стиля

Дизайнеры часто сталкиваются с необходимостью специального оформления первого элемента из набора – будь то список или несколько абзацев, расположенных в контейнере, – отличающего его от остальных элементов. Этого можно добиться путем присвоения определенного класса первому элементу, однако существует более подходящий способ создания такого эффекта, который поддерживается современными браузерами. Он состоит в использовании селектора псевдокласса `first-child`.

### Решение

Ниже представлена разметка простого неупорядоченного списка:

*chapter02/firstchild.html (фрагмент)*

```
<ul>
  <li>Brie</li>
  <li>Cheddar</li>
  <li>Red Leicester</li>
  <li>Shropshire Blue</li>
</ul>
```

### Использование селектора псевдокласса `first-child`

С помощью селектора псевдокласса `first-child` можно изменить стиль первого пункта списка, не повлияв при этом на оформление соседних элементов. Он позволяет применить стиль к первому элементу внутри контейнера `ul`, как показано на рис. 2.10:

*chapter02/firstchild.css (фрагмент)*

```
li:first-child {
  color: red;
}
```



Рис. 2.10. Первый пункт списка отображается красным цветом

К сожалению, `:first-child` не поддерживается браузером **Internet Explorer 6**. Поэтому, если данный эффект для вас важен и количество посетителей, просматривающих сайт с его помощью, велико, следует воспользоваться каким-либо другим методом, например применить селектор класса.

## Использование селектора класса

Для создания аналогичного эффекта, обрабатываемого IE6, нужно задать атрибут `class` или `id` элементу, стиль которого должен отличаться от остальных. В данном примере мы будем использовать `class`:

*chapter02/firstchildwithclass.html (фрагмент)*

```
<ul>
  <li class="unusual">Brie</li>
  <li>Cheddar</li>
  <li>Red Leicester</li>
  <li>Shropshire Blue</li>
</ul>
```

А теперь создадим правило стиля для достижения желаемого эффекта:

*chapter02/firstchildwithclass.css (фрагмент)*

```
li.unusual {
  color: red;
}
```

## Создание цветного фона для заголовка

Средствами CSS можно задать цвет фона для любого элемента, в том числе и для заголовка.

### Решение

Ниже приведено правило CSS, применяемое по отношению ко всем заголовкам первого уровня в документе:

*chapter02/headingcolor.css (фрагмент)*

```
h1 {
  background-color: #ADD8E6;
  color: #256579;
  font: 1.6em Verdana, Geneva, Arial, Helvetica, sans-serif;
  padding: 0.2em;
}
```

Результат выполнения данного кода представлен на рис. 2.11.

### Совет

**Цвет идет, дорогу цвету!** При добавлении цветного фона для заголовка вы, возможно, захотите также сделать необходимые отступы, чтобы между текстом заголовка и границей закрашенной области оставалось свободное пространство, как в данном примере.





Рис. 2.11. Отображение заголовка с цветным фоном

## Подчеркивание заголовков

### Решение

Существует два способа добавления эффекта подчеркивания для текста. Самым простым является использование свойства `text-decoration`, о котором мы уже говорили в данной главе в разделе «Удаление подчеркивания ссылок». Такой метод позволяет добавить подчеркивание того же цвета, что и сам текст, как видно из следующего кода и рис. 2.12:

*chapter02/headingunderline.css (фрагмент)*

```
h1 {  
    font: 1.6em Verdana, Geneva, Arial, Helvetica, sans-serif;  
    text-decoration: underline;  
}
```



Рис. 2.12. Добавление эффекта подчеркивания для заголовка с помощью свойства `text-decoration`

Аналогичного эффекта можно добиться и путем добавления нижней части рамки к заголовку. Данный метод, результат применения которого показан на рис. 2.13, отличается большей гибкостью, поскольку его применение позволяет отделить подчеркивающую линию от текста заголовка пустым пространством и задать для нее произвольный цвет, отличный от цвета самого заголовка.

Кроме того, вероятность спутать заголовок с эффектом подчеркивания, созданным описанным выше методом, со ссылкой гораздо ниже, чем при использовании свойства `text-decoration`. Однако внешний вид

данного эффекта может варьироваться в зависимости от используемого браузера, поэтому необходимо тщательно протестировать его во всех браузерах, которыми могут пользоваться потенциальные посетители вашего сайта. Ниже представлено необходимое правило стиля:

*chapter02/headingunderline2.css*

```
h1 {  
  font: 1.6em Verdana, Geneva, Arial, Helvetica, sans-serif;  
  padding: 0.2em;  
  border-bottom: 1px solid #AAAAAA;  
}
```

## Chinese-style stuffed peppers

These stuffed peppers are lovely as a starter or as a side dish for a Chinese meal. They also go down well as part of a buffet and even children seem to like them.

*Рис. 2.13. Создание эффекта подчеркивания с помощью нижней части рамки*

## Устранение отступа между элементом h1 и следующим за ним абзацем

По умолчанию при просмотре страницы между любыми заголовками и абзацами образуется пустое пространство, возникающее из-за того, что браузеры добавляют отступы для каждого из этих элементов. На рис. 2.14 отображен заголовок с отступом по умолчанию. Его можно убрать средствами CSS.

## Chinese-style stuffed peppers

These stuffed peppers are lovely as a starter or as a side dish for a Chinese meal. They also go down well as part of a buffet and even children seem to like them.

*Рис. 2.14. Пустое пространство между заголовком и абзацем, возникающее по умолчанию*

### Решение

Избежать возникновения пустого пространства возможно, убрав отступ снизу для заголовка и отступ сверху для абзаца. В современных броу-

зерах, включая Internet Explorer 7 и выше, это достигается с помощью смежного селектора в CSS. Однако для отображения аналогичного эффекта при просмотре страницы в более старых браузерах придется прибегнуть к другим приемам с лучшей реализованной поддержкой.

## Использование смежного селектора

Смежный селектор позволяет применять стили к элементу, следующему за другим элементом, если оба они являются дочерними по отношению к одному и тому же родительскому элементу. На самом деле с его помощью можно указать также элемент, следующий не за одним, а за несколькими другими элементами; элемент, к которому применяется стиль, называется *последним элементом в цепочке*. Если вышесказанное привело вас в замешательство, поверьте, что все сразу станет ясно, как только мы рассмотрим практические примеры.

Следующее правило стиля убирает верхний отступ для любого абзаца, следующего непосредственно за заголовком первого уровня. Обратите внимание, что отступ удаляется для абзаца, следующего за h1, а не для самого заголовка:

*chapter02/headingnospace.css (фрагмент)*

```
h1 {
    font: 1.6em Verdana, Geneva, Arial, Helvetica, sans-serif;
    margin-bottom: 0;
}
h1+p {
    margin-top: 0;
}
```

На рис. 2.15 показано, как будет отображаться страница после применения данного правила.



*Рис. 2.15. Использование смежного селектора для изменения стиля отображения заголовка*

Как видите, первый следующий за элементом h1 абзац отображается без отступа сверху; при этом все последующие абзацы отображаются с отступами.

Как уже было сказано выше, только самые современные версии браузеров поддерживают смежные селекторы – к примеру, в Internet Explorer

их поддержка реализована только начиная с 7-й версии. В определенных случаях вполне можно допустить, что пользователи более старых версий будут видеть зазор между заголовком и текстом. Если же вы твердо намерены убрать отступы, отображаемые в старых браузерах, в вашем распоряжении несколько возможностей.

Можно воспользоваться селекторами классов, как в разделе «Использование на одной странице различных стилей ссылок», присвоив свойству `margin` для каждого элемента класса нулевое значение. Если вы прочли указанный раздел, то использование такого метода не представит никаких сложностей. Кроме того, заголовку можно задать отрицательный отступ. Этот метод далее мы рассмотрим более подробно.

### Совет

**Применение отрицательных отступов.** В CSS свойство `margin`, задающее величину внешних отступов, может принимать как положительные, так и отрицательные значения. Однако значение свойства `padding`, обеспечивающее отступ от границы родительского элемента до границы дочернего, может быть только положительным.

Применение отрицательного значения свойства `margin` также позволяет убрать ненужный отступ между заголовком и первым абзацем. Приведенный ниже код позволяет достичь такого же результата, как мы видели на рис. 2.15:

```
h1 {
  font: 1.6em Verdana, Geneva, Arial, Helvetica, sans-serif;
  margin-bottom: -0.6em;
}
```

## Выделение текста на странице

На многих веб-страницах используется выделение важных терминов, например, слов, по которым посетитель пришел на сайт с помощью поисковой системы. Такое выделение текста можно без труда организовать средствами CSS.

### Решение

Если отрывок текста обрамлен тегами `<span>` с атрибутом `class`, для данного класса можно добавить правило стиля с помощью CSS. Например, в следующем абзаце фраза «**stuffed peppers**» расположена между тегами `<span>` с атрибутом класса `hilite`.

*chapter02/hilite.html (фрагмент)*

```
<p>These 
```

Ниже приведено правило стиля для класса `hilite`; отображение выделенного текста показано на рис. 2.16. В данном примере слова «stuffed peppers» отображаются красными буквами на желтом фоне.

*chapter02/hilite.css (фрагмент)*

```
.hilite {  
  background-color: #FFFFCC;  
  color: #B22222;  
}
```

## Chinese-style stuffed peppers

These **stuffed peppers** are lovely as a starter or as a side dish for a Chinese meal. They also go down well as part of a buffet and even children seem to like them.

*Рис. 2.16. Выделение текста с помощью CSS*

## Изменение высоты строки (межстрочного интервала) в тексте

Одним из основных преимуществ использования CSS перед более старыми методами, основанными на применении тегов `<font>` и т. д., состоит в том, что оно дает разработчику больше возможностей управления внешним видом текста на странице. В следующем примере мы попробуем изменить межстрочный интервал в тексте документа.

### Решение

Если межстрочный интервал по умолчанию оказывается слишком маленьким, его можно изменить с помощью свойства `line-height`:

*chapter02/leading.css*

```
p {  
  font: 1em Verdana, Geneva, Arial, Helvetica, sans-serif;  
  line-height: 2.0;  
}
```

Результат показан на рис. 2.17.

Проще простого! Единственное, на что стоит обратить внимание, – не стоит задавать слишком большие интервалы, иначе текст будет трудно читать.

## Chinese-style stuffed peppers

These stuffed peppers are lovely as a starter or as a side dish for a Chinese meal. They also go down well as part of a buffet and even children seem to like them.

*Рис. 2.17. Изменение межстрочного интервала с помощью свойства `line-height`*

### Совет

**А как же единицы измерения?** Как видите, в данном примере вместо единиц измерения мы использовали коэффициент 2.0. Вы *можете* задать значение `line-height` в стандартных единицах измерения CSS, например пикселах или `em`, однако при этом теряется связь между высотой строки и размером шрифта для дочерних элементов.

К примеру, если бы в предыдущем примере присутствовал элемент `span` с большим значением свойства `font-size`, величина межстрочного интервала изменилась бы пропорционально размеру шрифта, поскольку свойству `line-height` для абзаца задано числовое значение 2.0. Однако если бы данному свойству было присвоено значение `2em` или `200%`, элемент `span` унаследовал бы действительную, а не пропорциональную величину межстрочного интервала, на который не повлияет увеличенный размер шрифта. В определенных случаях это, конечно, может оказаться результатом, которого вы добиваетесь.

## Выравнивание текста по ширине

Выравнивание текста по ширине таким образом, чтобы его края были ровными с обеих сторон, обеспечивается благодаря изменению расстояний между словами. Этого эффекта легко достичь средствами CSS.

### Решение

Абзац можно выровнять по ширине с помощью свойства `text-align`, например:

*chapter02/justify.css*

```
p {
  text-align: justify;
  font: 1em Verdana, Geneva, Arial, Helvetica, sans-serif;
  line-height: 2.0;
}
```

На рис. 2.18 изображен результат присваивания значения `justify` свойству `text-align`.



Рис. 2.18. Выравнивание текста с помощью `text-align`

## Обсуждение

Свойство `text-align` также может принимать следующие значения:

- right**     выравнивает текст по правой стороне контейнера
- left**      выравнивает текст по левой стороне контейнера
- center**    центрирует текст в контейнере

## Изменение стиля горизонтальной линии

Как правило, при разметке документа следует избегать использования элементов, выполняющих чисто визуальные функции, такие как горизонтальная линия (`hr`). Документ, структура которого описывает назначение его элементов, легче поддерживать; он быстрее загружается и более понятен для поисковых систем. Эффекта, аналогичного использованию горизонтальной линии, можно добиться с помощью рамок существующих элементов.

Однако в определенных случаях использование `hr` может оказаться оптимальным способом достижения желаемого результата или необходимым средством оформления неразмеченного стилями контента для просмотра в старых браузерах, плохо поддерживающих CSS.

## Решение

С помощью CSS можно изменить цвет, высоту и ширину горизонтальной линии. Однако это следует делать с осторожностью, поскольку один и тот же эффект может по-разному выглядеть в различных браузерах. К примеру, в следующем примере используются два свойства, `color` и `background-color`, с одним и тем же значением, поскольку браузеры, основанные на движке Gecko, например Firefox, изменяют цвет линии с помощью свойства `background-color`, а Internet Explorer – с помощью `color`:

*chapter02/hrstyle.css (фрагмент)*

```
hr {
  border: none;
  background-color: #256579;
  color: #256579;
  height: 2px;
  width: 80%;
}
```

Результат можно увидеть на рис. 2.19.



*Рис. 2.19. Изменение цвета, высоты и ширины горизонтальной линии*

## Вывод текста с отступом

### Решение

Для добавления отступа от элемента до края контейнера последнему нужно задать свойство `padding-left`, как в следующем примере:

*chapter02/indent.html (фрагмент)*

```
<h1>Chinese-style stuffed peppers</h1>
<p class="indent">These stuffed peppers ...</p>
```

*chapter02/indent.css (фрагмент)*

```
.indent {
  padding-left: 1.5em;
}
```

Получившийся абзац с отступом можно увидеть на рис. 2.20.

### Обсуждение

Для создания отступов не стоит использовать HTML-тег `<blockquote>`, если текст не является в самом деле цитатой. Такая вредная привычка раньше прививалась программами визуального редактирования вроде



## Chinese-style stuffed peppers

These stuffed peppers are lovely as a starter or as a side dish for a Chinese meal. They also go down well as part of a buffet and even children seem to like them.

*Рис. 2.20. Добавление отступа с помощью CSS*

Dreamweaver. Если вы применяете редакторы, использующие данный тег в описанных целях, то вместо этого следует задать **CSS-правило**, подобное тому, что было описано выше. Или же смените редактор.

Тег `<blockquote>` был создан для выделения цитат, что особенно важно для пользователей со слабым зрением, пользующихся экранными дикторами: программа воспроизводит текст, заключенный между этими тегами, таким образом, что сразу становится ясно – это цитата. При использовании `<blockquote>` для создания отступов пользователям, воспринимающим страницу на слух, наверняка покажется странным, что текст читается как цитата.

### Совет

**С красной строки.** Можно создать отступ и только для первой строки каждого абзаца. Для этого к нему – или классу абзацев – нужно применить свойство `text-indent`:

*chapter02/indent2.css*

```
p {
    text-indent: 1.5em;
}
```

## Центрирование текста

Центрировать текст или любой другой элемент можно с помощью свойства `text-align` со значением `center`:

*chapter02/center.html (фрагмент)*

```
<h1>Chinese-style stuffed peppers</h1>
<p class="centered">These stuffed peppers ...</p>
```

*chapter02/center.css (фрагмент)*

```
.centered {
    text-align: center;
}
```

Результат использования данного правила отражен на рис. 2.21.



Рис. 2.21. Центрирование текста с помощью свойства `text-align`

## Вывод текста заглавными буквами с помощью CSS

### Решение

Для изменения всех букв на заглавные и других трансформаций текста используется свойство `text-transform`:

*chapter02/uppercase.html (фрагмент)*

```
<h1>Chinese-style stuffed peppers</h1>
<p class="transform">These stuffed peppers are lovely ...</p>
```

*chapter02/uppercase.css (фрагмент)*

```
.transform {
  text-transform: uppercase;
}
```

Обратите внимание на текст на рис. 2.22.

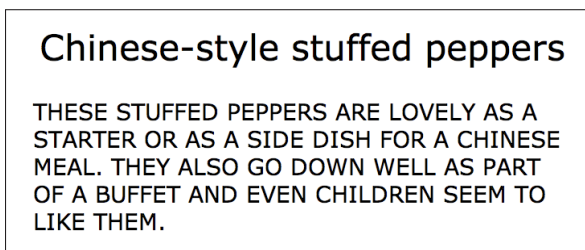


Рис. 2.22. Использование свойства `text-transform` для вывода текста заглавными буквами

### Обсуждение

Свойство `text-transform` может принимать и другие полезные значения. При использовании значения `capitalize` каждое слово будет начинаться с заглавной буквы, как видно на рис. 2.23.

*chapter02/capitalize.css (фрагмент)*

```
.transform {
  text-transform: capitalize;
}
```

Данному свойству также можно присваивать следующие значения:

- lowercase (для отображения текста строчными буквами)
- none (отсутствие трансформаций, значение по умолчанию)



*Рис. 2.23. Использование свойства text-transform для вывода каждого слова с большой буквы*

## Изменение стиля маркеров списка или удаление маркеров

### Решение

Стиль маркеров нумерованного списка можно изменить с помощью свойства list-style-type. Прежде всего рассмотрим следующую разметку списка:

*chapter02/listtype.html (фрагмент)*

```
<ul>
  <li>list item one</li>
  <li>list item two</li>
  <li>list item three</li>
</ul>
```

Для отображения маркеров в виде квадратиков, как на рис. 2.24, следует присвоить свойству list-style-type значение square:

*chapter02/listtype.css*

```
ul {
  list-style-type: square;
}
```

- 1 tablespoon of oil
- 1 crushed garlic clove
- Peeled and finely chopped fresh ginger root
- 250g minced pork, beef or Quorn
- 1 chopped spring onion
- 1 chopped celery stick
- Grated rind of 1 lemon
- Finely chopped red chilli (optional)
- 4 large green peppers

*Рис. 2.24. Маркеры списка в виде квадратиков*

## Обсуждение

Свойство `list-style-type` может иметь следующие значения: `disc`, `circle`, `decimal-leading-zero`, `decimal`, `lower-roman`, `upper-roman`, `lower-alpha`, `upper-alpha` и `none`.

Не все браузеры обеспечивают поддержку всех перечисленных свойств. Если браузер не поддерживает указанное значение, будет отображаться маркер, используемый по умолчанию. Информацию о различных типах маркеров и их поддержке браузерами можно найти в наборе тестов CSS2 для `list-style-type`.<sup>1</sup> Значение `none` удаляет маркеры из списка, причем каждый пункт по-прежнему будет отображаться с отступом, как будто на месте маркеров осталось пустое пространство, как можно видеть на рис. 2.25.

```
ul {  
  list-style-type: none;  
}
```

- 1 tablespoon of oil
- 1 crushed garlic clove
- Peeled and finely chopped fresh ginger root
- 250g minced pork, beef or Quorn
- 1 chopped spring onion
- 1 chopped celery stick
- Grated rind of 1 lemon
- Finely chopped red chilli (optional)
- 4 large green peppers

*Рис. 2.25. Список без маркеров*

---

<sup>1</sup> <http://www.meyerweb.com/eric/css/tests/css2/sec12-06-02a.htm>

## Использование изображения вместо маркера списка

### Решение

Чтобы вместо маркеров списка отобразить заказное изображение, нужно использовать свойство `list-style-image` вместо `list-style-type`. Его значением может быть URL-адрес пути к файлу изображения:

*chapter02/listimage.css*

```
ul {
  list-style-image: url(bullet.gif);
}
```

На рис. 2.26 видно, как можно оформить список с помощью описанного приема:

- ✓ 1 tablespoon of oil
- ✓ 1 crushed garlic clove
- ✓ Peeled and finely chopped fresh ginger root
- ✓ 250g minced pork, beef or Quorn
- ✓ 1 chopped spring onion
- ✓ 1 chopped celery stick
- ✓ Grated rind of 1 lemon
- ✓ Finely chopped red chilli (optional)
- ✓ 4 large green peppers

*Рис. 2.26. Использование изображения в качестве маркера списка*

### Совет

**Добавление маркеров к отдельным пунктам списка.** Свойство `list-style-image` применяется по отношению к пунктам списка (элементам `li`). Если применить его по отношению к списку в целом (элементу `ul` или `ol`), оно будет унаследовано каждым отдельным пунктом. Тем не менее вы можете задать значение данного свойства отдельно для каждого пункта списка, добавив ему атрибут `class` или `id`. Это позволяет использовать разные маркеры для разных пунктов списка.

## Удаление у пунктов списка отступа слева

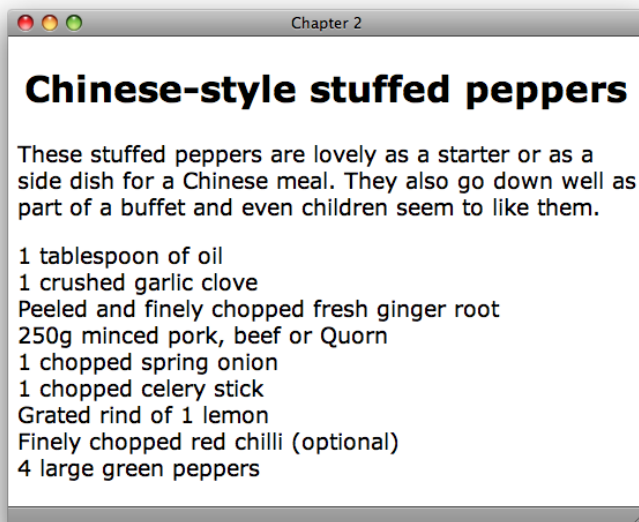
Если вы задали свойству `list-style-type` значение `none`, то, возможно, вам также хотелось бы уменьшить или вовсе убрать возникающие при этом отступы слева.

## Решение

Можно убрать отступы и выровнять пункты списка по левому краю, так что по вертикали они будут отображаться на одном уровне с предшествующим абзацем, как показано на рис. 2.27. Для этого нужно создать такое правило:

*chapter02/listnomargin.css*

```
ul {
  list-style-type: none;
  padding-left: 0;
  margin-left: 0;
}
```



*Рис. 2.27. Список без маркеров и без отступов*

## Обсуждение

При желании можно изменить величину отступов для пунктов списка. Например, чтобы отодвинуть список на несколько пикселей, можно использовать следующий код:

*chapter02/listsmallmargin.css*

```
ul {
  list-style-type: none;
  padding-left: 5px;
  margin-left: 0;
}
```

## Размещение пунктов списка по горизонтали

По умолчанию пункты списка располагаются как блочные элементы, т. е. каждый новый пункт размещается на новой строке. Однако на странице вполне может оказаться контент, по сути являющийся списком, который вам хотелось бы отобразить иным образом. Хорошим примером может послужить набор навигационных ссылок. Можно ли разместить их горизонтально?

### Решение

Для горизонтального размещения пунктов списка нужно присвоить свойству `display` для элемента `li` значение `inline`, например:

*chapter02/listinline.html (фрагмент)*

```
<ul class="horiz">
  <li><a href="#">Big Widgets</a></li>
  <li><a href="#">Small Widgets</a></li>
  <li><a href="#">Short Widgets</a></li>
  <li><a href="#">Tall Widgets</a></li>
</ul>
```

*chapter02/listinline.css*

```
ul.horiz li {
  display: inline;
}
```

Результат применения данного правила стиля изображен на рис. 2.28.



*Рис. 2.28. Горизонтальное расположение пунктов списка*

## Удаление отступов от края страницы

По умолчанию в большинстве браузеров присутствует пустое пространство между их рамкой и контентом страницы. Это означает, что при отсутствии таблицы стилей разработчика текст не будет соприкасаться с границей окна браузера. Возможно, вам потребуется убрать отступы или задать их размер, не оставляя определение этого размера на усмотрение браузера.

### Решение

Для удаления всех отступов вокруг контента страницы используется следующее правило стиля, определенное для элемента `body`:

```
body {  
  margin: 0;  
  padding: 0;  
}
```

Результат показан на рис. 2.29.

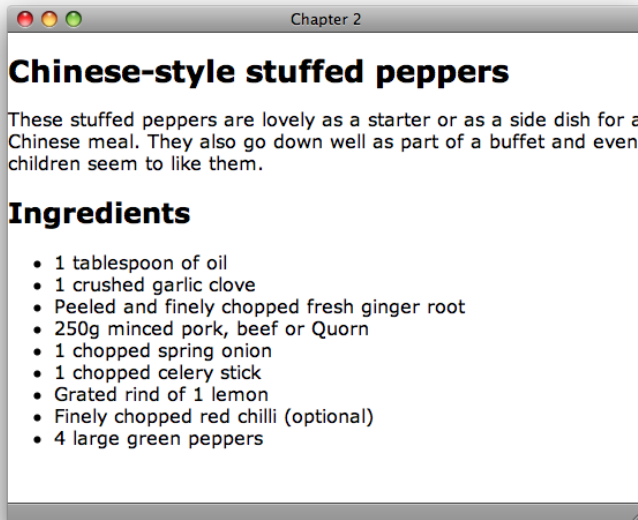


Рис. 2.29. Удаление отступов по умолчанию для тела документа

## Удаление отступов по умолчанию для всех элементов страницы

Если для документа не написано специальной таблицы стилей, то он будет оформлен в соответствии с внутренней таблицей стилей браузера. Поскольку внутренние таблицы разных браузеров немного отличаются, документы без стилового оформления будут отображаться слегка по-разному.

### Решение

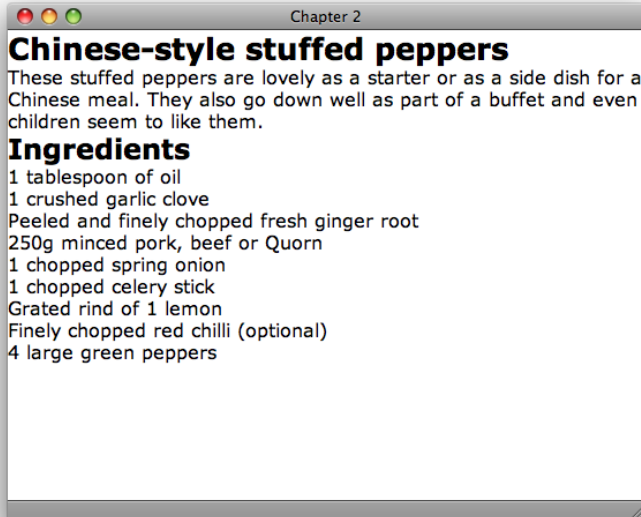
Эту проблему можно решить, убрав все отступы для всех элементов страницы перед созданием своих стилей.

Следующее правило задает нулевое значение отступов для *всех* элементов. В результате все абзацы, списки, заголовки и т. д. будут расположены вплотную к соседним элементам, как на рис. 2.30.



*chapter02/zeropagemargin.css (фрагмент)*

```
* {  
  margin: 0;  
  padding: 0;  
}
```



*Рис. 2.30. Удаление отступов для всех элементов страницы*

## Обсуждение

В приведенном правиле используется универсальный селектор «\*» для удаления отступов *везде*. Этот прием называют **глобальным удалением промежутков**<sup>1</sup>. Если дизайн вашей страницы отличается сложностью, то начать разработку стоит именно с этого.

Однако впоследствии при необходимости вам придется снова добавлять отступы для отдельных элементов. При этом особенно важно учитывать, что после применения данного правила может оказаться невозможным просмотр или использование некоторых элементов.

Если дизайн страницы относительно прост, такое удаление отступов, как правило, излишне, и в дальнейшем приводит к необходимости выполнения дополнительной работы по их восстановлению для таких элементов, как абзацы, цитаты и списки. Вместо этого можно удалить отступы только для определенной группы элементов. В приведенном ниже примере аннулируются отступы для всех заголовков и списков:

<sup>1</sup> <http://leftjustified.net/journal/2004/10/19/global-ws-reset/>

```
h1, h2, h3, h4, h5, h6, ul, ol {
    margin: 0;
    padding: 0;
}
```

## Добавление комментария в файл с каскадной таблицей стилей

В CSS-файлы можно и нужно добавлять комментарии (разве что, возможно, за исключением тех случаев, когда размещенная в них таблица стилей проста и содержит всего несколько правил). Однако при наличии большого количества правил стилей и при использовании нескольких таблиц на одном сайте комментарии приходится как нельзя кстати. Без них вы потратите уйму времени на поиск необходимых классов, на раздумья о назначении тех или иных классов и на попытки определить, в какой таблице они используются.

### Решение

Как и в коде на JavaScript, в CSS-коде можно располагать многострочные комментарии. Для этого используется следующая последовательность символов:

```
/*
  : Здесь располагаются комментарии...
*/
```

По меньшей мере в начале каждой таблицы стилей разработчик должен разместить комментарий с описанием ее назначения:

```
/* Эта таблица стилей используется по умолчанию для оформления текста на
всем сайте */
```

### Заключение

В настоящей главе мы рассмотрели ответы на наиболее часто встречающиеся вопросы, с которыми сталкиваются начинающие разработчики. Они касаются стилового оформления текста на странице. Путем комбинирования указанных приемов можно создавать привлекательные эффекты, корректное отображение которых, однако, зависит от степени поддержки CSS используемым браузером.

# 3

## CSS и графика

Во Всемирной паутине можно найти множество сайтов с привлекательным графическим дизайном, опирающимся на использование CSS. Чтобы научиться работать с графикой с помощью CSS, достаточно выучить несколько простых приемов, комбинирование которых позволит создать огромное число интересных эффектов. Представленные в настоящей главе решения иллюстрируют основные принципы работы с изображениями и дают ответы на наиболее часто встречающиеся вопросы. Работать с графикой мы будем и в последующих главах, однако даже с предложенными в этой главе решениями можно свободно экспериментировать для создания собственных уникальных эффектов.

### Добавление рамки к изображению

Фотографии, используемые в качестве иллюстраций к статье или размещаемые в фотоальбоме, выглядят более аккуратно с обрамлением в виде тонкой рамки. Однако добавление рамки в графическом редакторе для каждого изображения займет немало времени, а каждый раз, когда потребуется изменить толщину или цвет рамки, вы окажетесь перед необходимостью повторять этот трудоемкий процесс заново. К счастью, CSS предлагает гораздо более легкий способ выполнения аналогичных задач.

### Решение

Добавить рамку к изображению с помощью CSS предельно просто. Допустим, в документе на рис. 3.1 есть два изображения.

Благодаря следующему правилу к обоим изображениям добавляется сплошная черная рамка толщиной в 1 пиксел:



*Рис. 3.1. Просмотр двух изображений в веб-браузере*

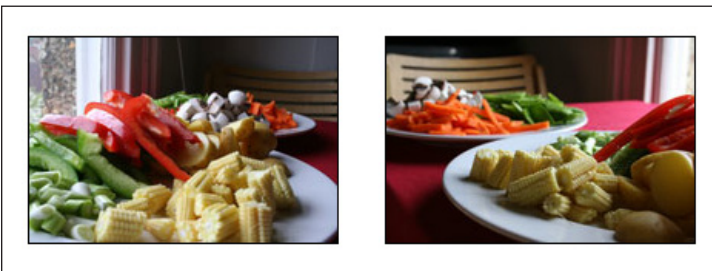
```
img {  
  border-width: 1px;  
  border-style: solid;  
  border-color: #000000;  
}
```

Это правило можно записать в сокращенной форме, например:

*chapter03/borderbasic.css (фрагмент)*

```
img {  
  border: 1px solid #000000;  
}
```

На рис. 3.2. показан результат выполнения данного кода.



*Рис. 3.2. С добавленной с помощью CSS рамкой изображение выглядит привлекательнее*

Все это замечательно, но ваш документ ведь наверняка содержит изображения, которым черная рамка *не* нужна. В таком случае можно создать класс для изображений с рамками и применить его по отношению к соответствующим элементам:

*chapter03/borderclass.css (фрагмент)*

```
.imgborder {  
  border: 1px solid #000000;  
}
```

*chapter03/borderclass.html (фрагмент)*

```

```

Если на странице отображается подборка изображений, например фотоальбом, можно задать параметры рамки для элементов, расположенных внутри определенного контейнера, допустим, списка с уникальным идентификатором.

*chapter03/borderalbum.css (фрагмент)*

```
#album img {  
border: 1px solid #000000;  
}
```

*chapter03/borderalbum.html (фрагмент)*

```
<ul id="album">  
<li></li>  
<li></li>  
</ul>
```

Такой прием избавляет от необходимости добавления атрибута с именем класса каждому отдельному изображению в контейнере.

## Удаление средствами CSS синей рамки вокруг изображения, выполняющего функцию ссылки

Если вы используете изображения в качестве ссылок, то наверняка заметили, что вокруг них появляется некрасивая рамка синего цвета, напоминающая подчеркивание текстовых ссылок. CSS поможет от нее избавиться.

### Решение

Процесс удаления рамки схож с процедурой ее добавления. Для этого используется правило, присваивающее свойству `border` значение `none`:

*chapter03/bordernone.css (фрагмент)*

```
img {  
border: none;  
}
```

## Задание фонового изображения для страницы с помощью CSS

Для указания фонового изображения веб-страницы используется свойство `background-image` по отношению к элементу `body`.

## Решение

Следующее правило использует в качестве фона страницы изображение **background-norepeat.jpg**. Оно распространяется на любую страницу, к которой привязана содержащая его таблица стилей.

*chapter03/backgrounds.css (фрагмент)*

```
body {
  font: 0.9em Verdana, Geneva, Arial, Helvetica, sans-serif;
  background-color: #D2D7E4;
  color: #000000;
  background-image: url(background-norepeat.jpg);
  background-repeat: no-repeat;
}
```

Результат выполнения правила показан на рис. 3.3.<sup>1</sup>



*Рис. 3.3. Использование фонового изображения*

## Обсуждение

С помощью CSS-свойства `background-image` можно указать в таблице стилей путь к файлу с фоновым изображением. Чтобы использовать фоно-

<sup>1</sup> Цветные иллюстрации доступны по адресу [www.symbol.ru/library/css-101-tips](http://www.symbol.ru/library/css-101-tips)

вое изображение для самой страницы целиком, задайте данное свойство для элемента `body`. Впрочем, как вы увидите в дальнейшем, фоновое изображение можно задать для любого элемента страницы.

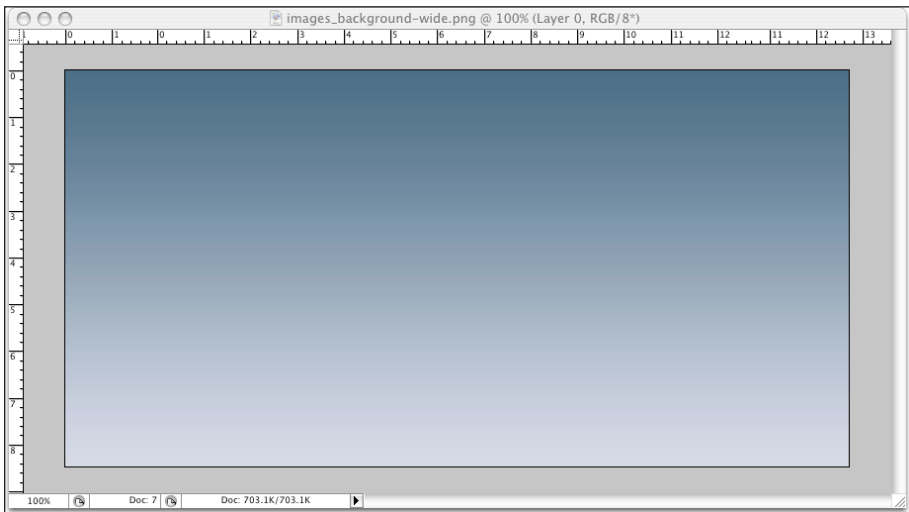
По умолчанию изображение будет повторяться, располагаясь мозаично по горизонтали и по вертикали, пока не заполнит все доступное пространство. Показанный на рис. 3.3. результат был достигнут путем использования изображения с рис. 3.4. со значением `no-repeat` свойства `background`.

## Как изменить способ размножения фонового изображения

По умолчанию изображение будет повторяться, пока не заполнит все доступное пространство по горизонтали и по вертикали. Однако его поведением можно управлять с помощью свойства `background-repeat`.

### Решение

Результат использования изображения с рис. 3.4 со значением `no-repeat` свойства `background-repeat`.



*Рис. 3.4. Заполнение фона страницы с помощью достаточно широкого изображения; свойству `background-repeat` присвоено свойство `no-repeat`*

Высота изображения составляет лишь 400 пикселей, что гораздо меньше высоты типичной веб-страницы, поэтому ее фону был задан тот же цвет, что и у последнего ряда пикселей градиентного изображения. Таким образом происходит плавный переход от изображения к фону страницы.

Такого эффекта можно добиться и более эффективным способом, состоящим в использовании меньшего по размеру изображения, которое быстрее загружается. Для этого нужно отрезать от нашего градиентного изображения тонкую полосу, как показано на рис. 3.5.

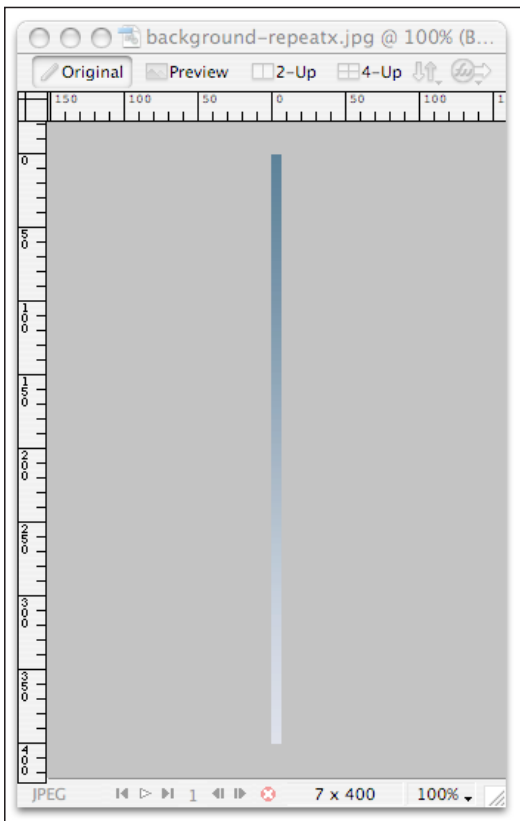


Рис. 3.5. Отрезанная полоса градиентного изображения

Задав свойству `background-repeat` для созданного изображения значение `repeat-x`, получим такой же результат, что и в первом примере, используя при этом файл гораздо меньшего размера. Чтобы обеспечить сплошное покрытие отображаемой в браузере страницы градиентом, укажем в качестве ее фона цвет, совпадающий с нижней частью градиента.

Если бы градиентный переход осуществлялся не сверху вниз, а справа налево, для создания фоновой заливки можно было бы воспользоваться тем же методом, повернув эффект на 90 градусов. Взяв в качестве фонового изображения горизонтальную полосу и указав свойству `background-repeat` значение `repeat-y`, мы обеспечиваем повторение градиента сверху вниз, как показано на рис. 3.6.





*Рис. 3.6. Градиентное изображение со значением `repeat-y` свойства `background-repeat`*

## Как позиционировать фоновое изображение

По умолчанию добавленное на страницу одиночное, без повторения, фоновое изображение появляется в левом верхнем углу страницы. Если в правилах стилей будет указано, что изображение должно размножаться, то отправной точкой послужит именно эта позиция. Однако фоновое изображение можно разместить и в другом месте страницы.

### Решение

Для позиционирования фонового изображения используется CSS-свойство `background-position`:

*chapter03/backgroundposition.css (фрагмент)*

```
#content {
  margin: 2em 4em 2em 4em;
  background-color: #FFFFFF;
  padding: 1em 1em 40px 1em;
  background-image: url(tick.gif);
  background-repeat: no-repeat;
```

```
background-position: bottom right;  
}
```

Приведенное выше правило размещает изображение в правом нижнем углу области с контентом, как показано на рис. 3.7. Чтобы текст не перекрывал изображение, для содержащего его контейнера были добавлены внутренние отступы.

## Обсуждение

Значениями свойства `background-position` могут быть ключевые слова, процентные соотношения и числовые значения в различных единицах измерения.



*Рис. 3.7. Позиционирование фонового изображения с помощью свойства `background-position`*

## Ключевые слова

В предыдущем примере, чтобы указать, что фоновое изображение должно появиться в правом нижнем углу элемента `div`, являющегося контейнером для текстового контента, мы использовали ключевое слово.

*chapter03/backgroundposition.css (фрагмент)*

```
background-position: bottom right;
```

**Вы можете использовать любую из следующих комбинаций ключевых слов:**

- top left
- top center
- top right
- center left
- center center
- center right
- bottom left
- bottom center
- bottom right

**Если вы укажете только первое из ключевых слов, то второе по умолчанию будет соответствовать значению center:**

```
background-position: top;
```

**Это описание тождественно следующей записи:**

```
background-position: top center;
```

### **Значения в процентах**

Добиться более точного расположения можно при указании значения в процентах. Это особенно эффективно, если величина и расположение других элементов страницы также указаны в процентах: в этом случае они будут менять свой масштаб в зависимости от разрешения и размеров экрана пользователя (такой прием также называют «резиновой» версткой, о которой мы поговорим в главе 9):

```
background-position: 30% 80%;
```

Первое значение задает расположение элемента по горизонтали, а второе – по вертикали. Отсчет начинается из верхнего левого угла, который соответствует значению 0% 0%; соответственно, при значении 100% 100% правый нижний угол изображения будет расположен в правом нижнем углу страницы.

Как и в случае с ключевыми словами, при указании только одного процентного значения второе сохранит значение по умолчанию, составляющее 50%. Рассмотрим следующую декларацию стиля:

```
background-position: 30%;
```

**Она тождественна следующей записи:**

```
background-position: 30% 50%;
```

## Использование единиц измерения

При позиционировании изображения с помощью CSS можно указать числовое значение в различных единицах измерения, например, пикселах:

```
background-position: 20px 20px;
```

Как и при указании значения в процентах, первое значение задает расположение элемента по горизонтали, а второе – по вертикали. Однако данный способ используется для четкого определения позиции верхнего левого угла фонового изображения.

Единицы измерения можно использовать совместно с процентными значениями, и при указании одного значения второе по умолчанию составит 50%.

## Как сделать фоновое изображение неподвижным при прокрутке контента

Вам наверняка встречались сайты, на которых фоновое изображение не меняет своего положения, в то время как контент прокручивается над ним. Такого эффекта можно добиться с помощью свойства `background-attachment`.

### Решение

Путем присвоения значения `fixed` свойству `background-attachment` можно зафиксировать положение фонового изображения, и оно будет оставаться на месте при прокрутке контента.

*chapter03/backgroundfixed.html (фрагмент)*

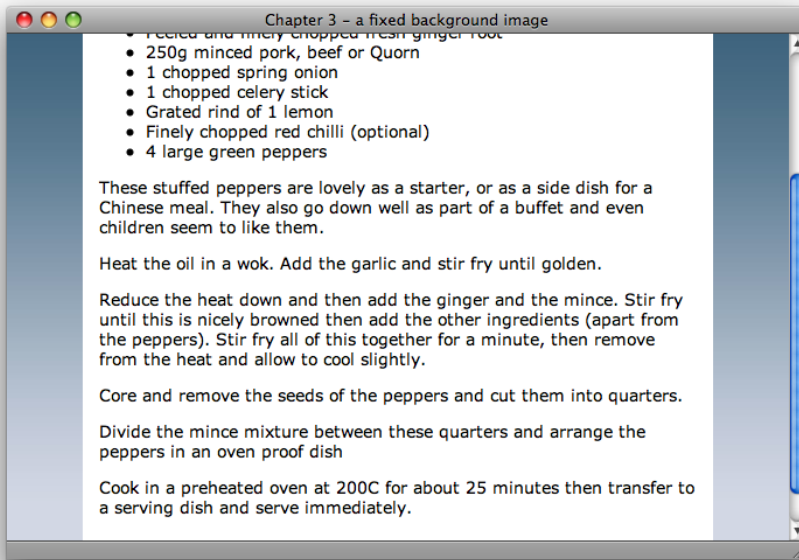
```
body {
  font: 0.9em Verdana, Geneva, Arial, Helvetica, sans-serif;
  background-color: #D2D7E4;
  color: #000000;
  background-image: url(background-repeatx.jpg);
  background-repeat: repeat-x;
  background-attachment: fixed;
}
```

Результат показан на рис. 3.8.

### Обсуждение

В данном случае для добавления в документ фонового изображения, его позиционирования и описания поведения при прокрутке контента было использовано несколько CSS-свойств.

В качестве варианта можно было бы воспользоваться сокращенным методом записи той же самой информации с помощью свойства `background`.



*Рис. 3.8. Зафиксированное фоновое изображение остается неподвижным при прокрутке контента*

Оно позволяет одновременно указать значения свойств `background-color`, `background-image`, `background-repeat`, `background-attachment` и `background-position` в одном описании. Для примера рассмотрим следующее правило CSS:

*chapter03/backgroundfixed.css (фрагмент)*

```
body {
    background-color: #D2D7E4;
    background-image: url(background-repeatx.jpg);
    background-repeat: repeat-x;
    background-attachment: fixed;
    background-position: 0 0;
}
```

Эти описания можно представить в более сжатой форме следующим образом:

```
body {
    background: #D2D7E4 url(background-repeatx.jpg) repeat-x fixed 0 0;
}
```

В заключение более подробно остановимся на описании `background-attachment: fixed`. Поддержка данного свойства, как и многих других CSS-свойств, у семейства браузеров Internet Explorer ограничена. Ранние версии IE обрабатывают данное свойство некорректно, что было ис-

правлено только в IE7. Существуют обходные пути с использованием JavaScript, но, как правило, в этом случае игра едва ли стоит свеч.<sup>1</sup> По умолчанию пользователи ранних версий Internet Explorer, не поддерживающих декларацию `background-attachment: fixed`, увидят прокрутку фонового изображения, что обычно можно считать вполне приемлемым компромиссом (это даже может подвигнуть пользователей на обновление своих браузеров).

## Для каких элементов можно задавать фоновое изображение

В данной главе мы уже рассмотрели способ задания фонового изображения для основной области страницы. Однако фон других элементов также можно задавать с помощью изображений.

### Решение

Данное правило стиля задает фоновое изображение для блока со списком ингредиентов в рецепте, как видно на рис. 3.9.

*chapter03/backgrounds2.css (фрагмент)*

```
#smallbox {
    background-image: url(boxbg.gif);
    background-repeat: repeat-x;
    float: left;
    margin-right: 20px;
    width: 220px;
    border: 1px solid #D2D7E4;
}
```

В качестве фонового изображения блока со списком ингредиентов также используется градиент, подобный тому, что мы использовали для основной страницы, за тем лишь исключением, что в данном случае осуществляется переход от голубого к белому цвету. Я также задала для блока рамку того же цвета, что и самая темная часть градиента.

### Обсуждение

Фоновые изображения можно использовать и для других элементов страницы, в том числе и для заголовков, как показано на рис. 3.10. Как видите, отображаемая под заголовком прерывистая линия создана путем повторения одного и того же изображения. Оно изначально расположено в левом нижнем углу заголовка, а чтобы слегка отделить его от текста заголовка, я задала для последнего внутренний отступ снизу в шесть пикселей.

---

<sup>1</sup> <http://www.howtcreate.co.uk/fixedBackground.html>



**Рис. 3.9.** Использование фонового изображения для отображения градиентного перехода в блоке со списком ингредиентов

*chapter03/backgrounds2.html (фрагмент)*

```
<h1>Chinese-style stuffed peppers</h1>
```

*chapter03/backgrounds2.css (фрагмент)*

```
h1 {
  background-image: url(dotty.gif);
  background-repeat: repeat-x;
  background-position: bottom left;
  padding: 0 0 6px 0;
  color: #41667F;
  font-size: 160%;
  font-weight: normal;
  background-color: transparent;
}
```

Даже ссылкам можно задавать фон, что порой может привести к весьма интересным эффектам, как на рис. 3.11.

*chapter03/backgrounds2.css (фрагмент)*

```
a:link, a:visited {
  color: #41667F;
```

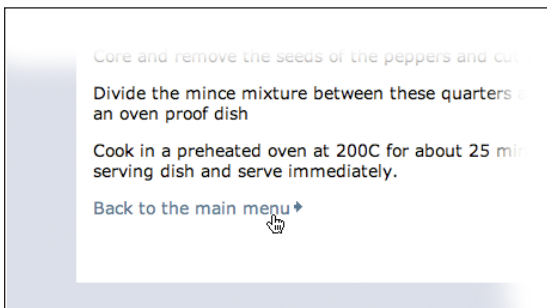
```

background-color: transparent;
padding-right: 10px;
}
a:hover {
background-image: url(arrow.gif);
text-decoration: none;
background-position: center right;
background-repeat: no-repeat;
}

```



**Рис. 3.10.** Использование фонового изображения для отображения линии под заголовком



**Рис. 3.11.** Применение фонового изображения к ссылке при наведении на нее указателя мыши



## Размещение текста поверх изображения

В те дни, когда CSS еще не существовало, добавить текст поверх изображения можно было только с помощью графической программы! CSS предлагает гораздо более эффективный метод создания такого эффекта.

### Решение

Для размещения текста над изображением проще всего использовать изображение в качестве фонового. Изображение под заголовком **Ingredients** в блоке со списком ингредиентов появилось благодаря использованию следующего правила стилей:



*Рис. 3.12. Задание фонового изображения для заголовка блока со списком ингредиентов*

*chapter03/backgrounds3.css (фрагмент)*

```
#smallbox h2 {
  margin: 0;
  padding: 0.2em;
  background-image: url(boxheaderbg.jpg);
  background-repeat: no-repeat;
  color: #FFFFFF;
  background-color: red;
```

```
font-size: 140%;  
font-weight: normal;  
}
```

## Обсуждение

Использование CSS вместо графического редактора для добавления текста поверх изображения эффективнее по нескольким причинам.

Во-первых, в дальнейшем будет сложнее изменить сам текст: для этого каждый раз придется искать исходное изображение, редактировать его в графической программе и снова загружать на сервер.

Во-вторых, представление информации на странице в виде текста, а не изображения, обладает большей гибкостью. Броузеры, не поддерживающие графику, смогут отобразить текстовое содержимое, добавленное с помощью CSS; размер шрифта такого текста может быть изменен пользователем. Это также благотворно сказывается на рейтинге сайта в поисковых системах: они не умеют распознавать текст на графических изображениях, но прекрасно индексируют текстовые данные, расположенные поверх изображения средствами CSS.

### Совет

**Проверьте контрастность!** При добавлении светлого текста поверх изображения (как на рис. 3.12) область его размещения следует залить темным цветом. Это обеспечит видимость текста при отключении изображений или при медленной скорости соединения, когда изображения загружаются не сразу.

## Как задать для документа более одного фонового изображения

Несмотря на наличие в спецификации CSS2 возможности применения к документу более одного фонового изображения, в действительности ее поддержка реализована в единственном браузере – Safari от Apple. Что же делать при возникновении необходимости в одновременном использовании двух изображений – допустим, одно используется для заполнения всей области, а другое отображается статично?

## Решение

Имитировать эффект использования нескольких фоновых изображений можно путем их задания для разных вложенных элементов, какими являются, например, `html` и `body`:

*chapter03/backgrounds4.css (фрагмент)*

```
html {
```

```

background-image: url(background-repeatx.jpg);
background-repeat: repeat-x;
background-color: #D2D7E4;
}
body {
font: 0.9em Verdana, Geneva, Arial, Helvetica, sans-serif;
color: #000000;
background-image: url(recipes.gif);
background-repeat: no-repeat;
background-position: 98% 2%;
margin: 0;
padding: 46px 0 0 0;
}

```

Результат применения данных правил показан на рис. 3.13.

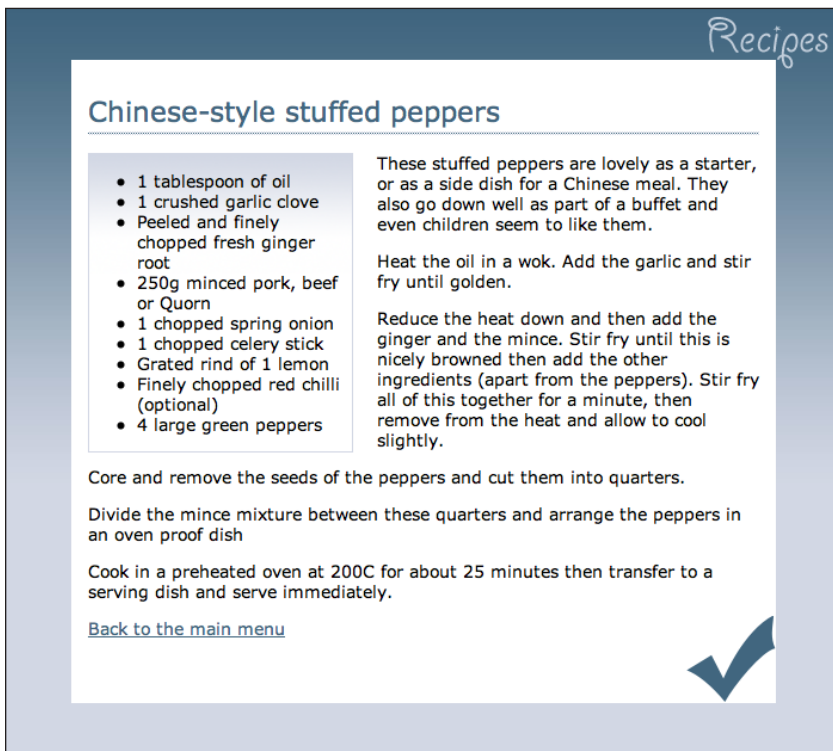


Рис. 3.13. Добавление фоновых изображений для элементов html и body

## Обсуждение

Такой простой пример может послужить основой для создания более сложных эффектов с использованием нескольких фоновых изображений. Как вы уже могли убедиться на представленных в данной книге

примерах, фоновое изображение можно задать для любого элемента страницы. Внимательный и творческий подход к применению изображений позволит создать множество интересных визуальных решений, не влияющих на доступность документа (поскольку фоновые изображения не связаны со структурой документа).

Многие из предлагаемых на сайте CSS Zen Garden решений основаны именно на такого рода игре с фоновыми изображениями.<sup>1</sup>

## Применение эффекта прозрачности

Добиться прозрачности можно с помощью графики в формате PNG – сохранив ваши изображения в формате 24-битного PNG, можно реализовать эффект прозрачности и полупрозрачности. Изображения в формате GIF также поддерживают прозрачность, однако при сохранении необходимо добавить к нему **кромку (matte)** – цвет, сходный с цветом фона, на котором оно будет расположено.

Создать изображение GIF с поддержкой прозрачности, одинаково отображаемое на различном фоне, очень сложно с технической точки зрения. Для этого, как правило, нужно разделить изображение на два и сохранить каждое из них отдельно, а затем вновь собрать их воедино на странице. Этот подход отдает чем-то совершенно дремучим, чего следует избегать при верстке документа с помощью CSS. Прокрутка контента с изображением в формате GIF над зафиксированным фоном приводит к возникновению «эффекта гало» (ореола), что выглядит не очень красиво. На рис. 3.14 мы видим и проявление прозрачности, но также видим и проявление нежелательного «эффекта гало».



Рис. 3.14. Нежелательный «эффект гало»

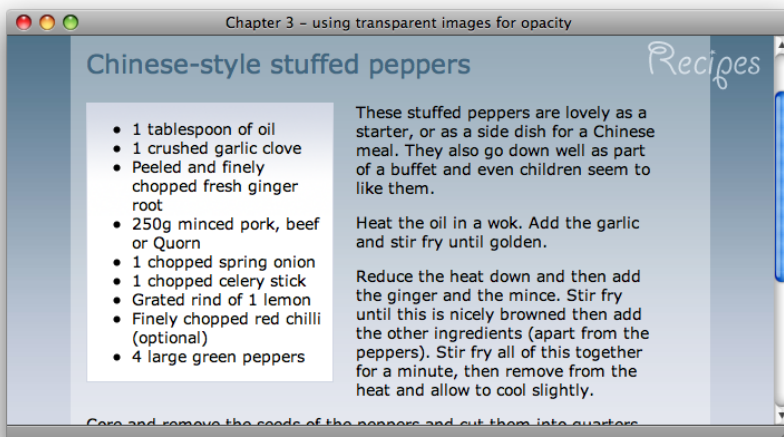
<sup>1</sup> <http://csszengarden.com/>

## Решение

В примере на рис. 3.15 использовано два изображения в формате PNG. Первое из них используется вместо белого фона элемента #content. Это десятипиксельное изображение было создано в Photoshop. К прозрачному изображению был добавлен новый слой белого цвета, степень прозрачности которого была уменьшена до 40%. Полученное изображение было сохранено в формате 24-битного PNG с именем `opaque.png`.

Второе изображение в формате 24-битного PNG с прозрачным фоном служит для замещения фонового изображения `recipes.gif`. Мне хотелось бы закрепить его в правом верхнем углу окна (с помощью декларации `background-attachment: fixed`), чтобы оно оставалось неподвижным при прокрутке страницы. Если бы вместо него использовалось изображение в формате GIF (с кромкой темно-синего цвета), то при движении оно оказалось бы над светлым фоном и возник описанный выше «эффект гало».

Достигнутый с помощью CSS эффект представлен на рис. 3.15:



*Рис. 3.15. Использование фонового изображения без «эффекта гало» для картинки с надписью Recipes*

*chapter03/background5.css (фрагмент)*

```
body {
  font: 0.9em Verdana, Geneva, Arial, Helvetica, sans-serif;
  color: #000000;
  background-image: url(recipes.png);
  background-repeat: no-repeat;
  background-position: 98% 2%;
  background-attachment: fixed;
```

```
margin: 0;
padding: 46px 0 0 0;
}

#content {
margin: 0 4em 2em 4em;
background-image: url(opaque.png);
padding: 1em 50px 40px 1em;
}
```

## Обсуждение

С помощью полупрозрачных изображений в формате PNG можно добиться совершенно невероятных привлекательных эффектов. Но, к сожалению, браузер **Internet Explorer 6** не обеспечивает должной поддержки их использования.

Однако, тщательно продумав все аспекты верстки вашей страницы, чаще всего можно найти решение, позволяющее создать такого рода эффект для пользователей более современных браузеров, таких как Firefox, Safari, Opera и Internet Explorer версий 7 и выше. Кроме того, ограничения Internet Explorer 6 можно обойти с помощью JavaScript. Я расскажу об этом методе в главе 7.

## Создание сложных рамок вокруг изображений, например двойных

Чтобы представить фотографии в лучшем виде, у вас вполне может появиться желание украсить их более сложными рамками, чем те, что были рассмотрены в настоящей главе. Совместное использование рамок и фоновых изображений позволяет добиться потрясающих визуальных эффектов – и все это с помощью CSS. Опять же это избавит вас от необходимости обработки изображений в Photoshop для создания рамок.

## Решение

На рис. 3.16 представлена фотография, которая может быть использована в качестве элемента страницы или быть частью фотоальбома. Самый простой метод создания двойной рамки состоит в задании фонового цвета и внутреннего отступа для изображения. В этом случае цвет фона послужит первой рамкой, а собственно рамка – второй:

*chapter03/doubleborder.css (фрагмент)*

```
img.doubleborder {
border: 1px solid #333;
padding: 5px;
background-color: #EEEEEE;
}
```



*Рис. 3.16. Простая двойная рамка, созданная исключительно средствами CSS*

С помощью фонового изображения можно добиться более сложных эффектов. В следующем примере CSS-кода используется заполнение области под фотографией небольшим мозаичным изображением для создания эффекта, показанного на рис. 3.17:

*chapter03/doubleborder-bg.css (фрагмент)*

```
img.doubleborder {  
    border: 5px solid #8E787B;  
    padding: 20px;  
    background-image: url(doubleborder-bg.gif);  
}
```



*Рис. 3.17. Создание двойной рамки с помощью фонового изображения*

## Обсуждение

Для достижения описанных выше эффектов использовались внутренние отступы для изображения, создающие пустое пространство между ним и рамкой. Пустое пространство будет заполнено цветом фона или

фоновым изображением, и таким образом можно создавать привлекательные рамки без использования вложенных элементов.

## Заключение

В настоящей главе были представлены ответы на наиболее часто возникающие вопросы о возможностях использования изображений. Основное внимание было уделено фоновым изображениям, которые, по сути, являются важным строительным материалом для верстки с помощью CSS страниц со сложным графическим дизайном. Использование фоновых изображений дает возможность более гибкого изменения внешнего вида страницы с помощью альтернативных таблиц стилей, а также создания интересных визуальных эффектов.

По ходу прочтения данной книги вы, безусловно, еще столкнетесь с вопросами, связанными с использованием изображений. В частности, в главе 9 рассматриваются способы их позиционирования по отношению к другим элементам страницы и их применение на страницах с более сложной структурой, чем те, что были представлены в этой главе.



# 4

## Навигация

Если на вашем сайте более одной страницы, вам придется разработать систему навигации по сайту. На самом деле навигация является одной из самых важных частей архитектуры любого сайта, и потому ее следует тщательно продумать, чтобы предоставить пользователям возможность удобного перемещения от одного раздела к другому.

Когда речь идет о создании удобной системы навигации, возможности CSS предстают во всей своей красе. Применяемые ранее методы были основаны, как правило, на использовании большого количества изображений, вложенных таблиц и сценариев на JavaScript. Безусловно, это оказывало негативное влияние на юзабилити и доступность сайтов. Если по сайту нельзя перемещаться при использовании устройств, не поддерживающих JavaScript, **то вы рискуете потерять часть пользовательской аудитории, отключившей JavaScript, и тех, кто пользуется устройствами, воспринимающими исключительно текст** (например, экранными дикторами). Не стоит забывать и о роботах поисковых систем, которые в такой ситуации не смогут пройти дальше главной страницы сайта. Если ваших клиентов не беспокоят вопросы доступности, объясните им, что с такими неуклюжими меню они рискуют опуститься в самый низ списка в рейтинге поисковых систем!

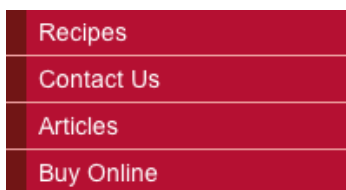
С помощью CSS можно создать привлекательную панель навигации с текстовыми ссылками, которые можно разметить таким образом, что они будут доступны и пользователям, физически не имеющим возможности увидеть ваш сайт, но желающим получить размещенную на нем информацию. В настоящей главе мы рассмотрим множество различных методов создания панелей навигации с помощью CSS. Некоторые из них вполне подойдут для замены старых графических меню на уже существующих сайтах, что сократит время их загрузки и сделает более доступными для различных категорий пользователей и поисковых систем. Другие предназначены для использования на сайтах, сверстаных исключительно на CSS.

## Оформление списка в виде навигационного меню

По сути, меню навигации представляет собой список разделов сайта, поэтому имеет смысл разметить его именно в виде списка, а затем применить стили CSS по отношению к его элементам. Однако визуально он должен отличаться от стандартного маркированного списка, который выводится на экран при использовании внутренней таблицы стилей браузера.

### Решение

Меню навигации, представленное на рис. 4.1, является списком, оформленным с помощью CSS.



*Рис. 4.1. Создание навигации путем написания правил стилей для списка*

Ниже представлена разметка, необходимая для создания списка:

*chapter04/listnav1.html*

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
  <head>
    <title>Lists as navigation</title>
    <meta http-equiv="content-type" content="text/html;
      charset=utf-8" />
    <link rel="stylesheet" type="text/css" href="listnav1.css" />
  </head>
  <body>
    <div id="navigation">
      <ul>
        <li><a href="#">Recipes</a></li>
        <li><a href="#">Contact Us</a></li>
        <li><a href="#">Articles</a></li>
        <li><a href="#">Buy Online</a></li>
      </ul>
    </div>
  </body>
</html>
```

Далее представлен полный CSS-код, преобразующий незамысловатый маркированный список в привлекательное меню навигации:

*chapter04/listnav1.css*

```
#navigation {
    width: 200px;
}
#navigation ul {
    list-style: none;
    margin: 0;
    padding: 0;
}
#navigation li {
    border-bottom: 1px solid #ED9F9F;
}
#navigation li a:link, #navigation li a:visited {
    font-size: 90%;
    display: block;
    padding: 0.4em 0 0.4em 0.5em;
    border-left: 12px solid #711515;
    border-right: 1px solid #711515;
    background-color: #B51032;
    color: #FFFFFF;
    text-decoration: none;
}
```

## Обсуждение

Для создания панели навигации на основе маркированного списка необходимо прежде всего создать сам список, разместив ссылки внутри элементов `li`:

*chapter04/listnav1.html (фрагмент)*

```
<ul>
  <li><a href="#">Recipes</a></li>
  <li><a href="#">Contact Us</a></li>
  <li><a href="#">Articles</a></li>
  <li><a href="#">Buy Online</a></li>
</ul>
```

Затем разместим список внутри элемента `div` с соответствующим ID:

*chapter04/listnav1.html (фрагмент)*

```
<div id="navigation">
  <ul>
    <li><a href="#">Recipes</a></li>
    <li><a href="#">Contact Us</a></li>
    <li><a href="#">Articles</a></li>
    <li><a href="#">Buy Online</a></li>
  </ul>
</div>
```

Как видно на рис. 4.2, при применении внутренней таблицы стилей, используемой браузером по умолчанию, такая разметка выглядит довольно обычно.



Рис. 4.2. Первоначальный вид списка (стилевое оформление отсутствует)

Оформление списка лучше начать с написания правил стилей для контейнера, в котором размещается сам список, – в данном случае с идентификатором `navigation`:

*chapter04/listnav1.css (фрагмент)*

```
#navigation {  
  width: 200px;  
}
```

Данное правило задает ширину контейнера. Если бы он являлся частью сверстанного на CSS макета, то, скорее всего, нужно было бы добавить в него информацию о позиционировании.

Затем займемся написанием правил стилей для списка:

*chapter04/listnav1.css (фрагмент)*

```
#navigation ul {  
  list-style: none;  
  margin: 0;  
  padding: 0;  
}
```

Как видно на рис. 4.3, вышеприведенное правило удаляет маркеры списка и убирает внешние отступы, создаваемые браузером вокруг списка по умолчанию.



Рис. 4.3. Внешний вид списка после удаления маркеров и внешних отступов

Теперь настало время создать стиль для дочерних элементов `li` блока `navigation`, добавив для них нижнюю рамку:

*chapter04/listnav1.css (фрагмент)*

```
#navigation li {  
    border-bottom: 1px solid #ED9F9F;  
}
```

Наконец, зададим стиль для самой ссылки:

*chapter04/listnav1.css (фрагмент)*

```
#navigation li a:link, #navigation li a:visited {  
    font-size: 90%;  
    display: block;  
    padding: 0.4em 0 0.4em 0.5em;  
    border-left: 12px solid #711515;  
    border-right: 1px solid #711515;  
    background-color: #B51032;  
    color: #FFFFFF;  
    text-decoration: none;  
}
```

Большинство изменений вносятся именно с помощью указанных выше правил CSS, задающих стиль рамки справа и слева, удаляющих подчеркивание и т. д. Первое описание присваивает значение `block` свойству `display`, благодаря чему ссылка отображается как блочный элемент. Это означает, что при наведении указателя мыши на любое место навигационной «кнопки» она будет активирована. Аналогичного эффекта можно было бы достичь путем использования изображения в качестве ссылки.

## Изменение вида ссылки при наведении на нее указателя мыши с помощью CSS без использования изображений или сценариев на JavaScript

При наведении указателя мыши на кнопку навигационного меню ее вид может изменяться, привлекая к себе внимание. Для реализации такого эффекта в навигации, основанной на применении графики, необходимо использовать два изображения и JavaScript.

### Решение

Создать такой привлекательный эффект гораздо проще с помощью CSS, а не графики. Для этого используется селектор псевдокласса `:hover`, как и при задании стиля для ссылок при наведении на них указателя мыши.

Вернемся к рассмотренному ранее примеру навигационного меню и добавим в таблицу стилей следующее правило:

*chapter04/listnav2.css (фрагмент)*

```
#navigation li a:hover {  
    background-color: #711515;
```

```
color: #FFFFFF;  
}
```

На рис. 4.4 показан вид меню при наведении курсора на первый пункт:

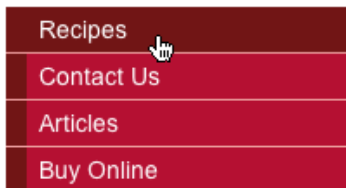


Рис. 4.4. Эффект смены визуального оформления пунктов меню, реализованный средствами CSS

## Обсуждение

Используемый для создания такого эффекта CSS-код предельно прост. Механизм создания разного визуального оформления ссылок одинаков при работе со ссылками различной сложности. В данном примере реализована смена цвета фона ссылки на цвет левой части рамки, однако в вашем распоряжении имеется огромное количество возможностей для создания интересных навигационных эффектов: например, можно изменить цвет фона, текста и рамки.

### Примечание

**Мышка тут, мышка там...** При условии просмотра сайта в современных браузерах, включая Internet Explorer 7, селектор псевдокласса `:hover` можно использовать по отношению к *любому* элементу, однако версии Internet Explorer 6 и ниже корректно обрабатывают его только для ссылок.

В ранних версиях Internet Explorer для перехода по ссылке необходимо было щелкнуть по ее тексту, поскольку ссылка не заполняла весь контейнер (в данном случае – элемент `li`) целиком. Это означает, что для перехода к определенному разделу пользователь должен щелкнуть по *тексту*: щелчок по красному фоновому цвету не сработает.

Данную проблему можно решить путем увеличения ширины ссылки средствами CSS (однако такой способ приемлем только для Internet Explorer 6 и ниже). Здесь представлено необходимое для этого правило стиля:

```
* html #navigation li a {  
width: 100%;  
}
```

Безусловно, в некоторых случаях вполне можно обойтись без таких трюков и оставить все как есть. Мы рассмотрим проблему создания кросс-браузерного кода более подробно в главе 7.

## Создание навигационного меню с подпунктами с помощью списков и таблиц стилей

В приводимых ранее примерах были рассмотрены навигационные панели в виде одноуровневых списков. Иногда этого недостаточно. Но можно ли создать многоуровневое меню навигации с помощью списков и CSS?

### Решение

Для создания подуровня системы навигации оптимально создать список второго уровня внутри основного списка. Такая разметка четко обозначает различные уровни навигации и вполне понятна даже браузерам, не поддерживающим CSS в полной мере.

Вернемся к изображенному на рис. 4.4 примеру и добавим в навигационное меню подуровень путем создания вложенного списка и задания стиля его пунктов:

*chapter04/listnav\_sub.html*

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
<head>
<title>Lists as navigation</title>
<meta http-equiv="content-type"
  content="text/html; charset=utf-8" />
<link rel="stylesheet" type="text/css" href="listnav_sub.css" />
</head>
<body>
<div id="navigation">
  <ul>
    <li><a href="#">Recipes</a>
      <ul>
        <li><a href="#">Starters</a></li>
        <li><a href="#">Main Courses</a></li>
        <li><a href="#">Desserts</a></li>
      </ul>
    </li>
    <li><a href="#">Contact Us</a></li>
    <li><a href="#">Articles</a></li>
    <li><a href="#">Buy Online</a></li>
  </ul>
</div>
</body>
</html>

```

*chapter04/listnav\_sub.css*

```

#navigation {
  width: 200px;

```

```
}
#navigation ul {
  list-style: none;
  margin: 0;
  padding: 0;
}
#navigation li {
  border-bottom: 1px solid #ED9F9F;
}
#navigation li a:link, #navigation li a:visited {
  font-size: 90%;
  display: block;
  padding: 0.4em 0 0.4em 0.5em;
  border-left: 12px solid #711515;
  border-right: 1px solid #711515;
  background-color: #B51032;
  color: #FFFFFF;
  text-decoration: none;
}
#navigation li a:hover {
  background-color: #711515;
  color: #FFFFFF;
}
#navigation ul ul {
  margin-left: 12px;
}
#navigation ul ul li {
  border-bottom: 1px solid #711515;
  margin: 0;
}
#navigation ul ul a:link, #navigation ul ul a:visited {
  background-color: #ED9F9F;
  color: #711515;
}
#navigation ul ul a:hover {
  background-color: #711515;
  color: #FFFFFF;
}
```

Результат добавления данного кода представлен на рис. 4.5.

## Обсуждение

Использование вложенных списков – оптимальный способ создания подобных панелей навигации. Основной список содержит ключевые разделы сайта, в то время как в списке второго уровня под пунктом **Recipes** отображаются подразделы данной категории. Структура списка отличается четкостью и ясностью и без использования CSS, как видно на рис. 4.6.





Рис. 4.5. Созданное средствами CSS многоуровневое навигационное меню



Рис. 4.6. Логичность структуры системы навигации очевидна и без использования CSS

Для создания такого списка используется простая HTML-разметка: вложенный список добавляется внутри соответствующего элемента li.

*chapter04/listnav\_sub.html*

```
<div id="navigation">
  <ul>
    <li><a href="#">Recipes</a>
      <ul>
        <li><a href="#">Starters</a></li>
        <li><a href="#">Main Courses</a></li>
        <li><a href="#">Desserts</a></li>
      </ul>
    </li>
    <li><a href="#">Contact Us</a></li>
    <li><a href="#">Articles</a></li>
    <li><a href="#">Buy Online</a></li>
  </ul>
</div>
```

При обработке данного HTML-кода в браузере без изменения таблицы стилей мы увидим результат, изображенный на рис. 4.7 слева; в этом

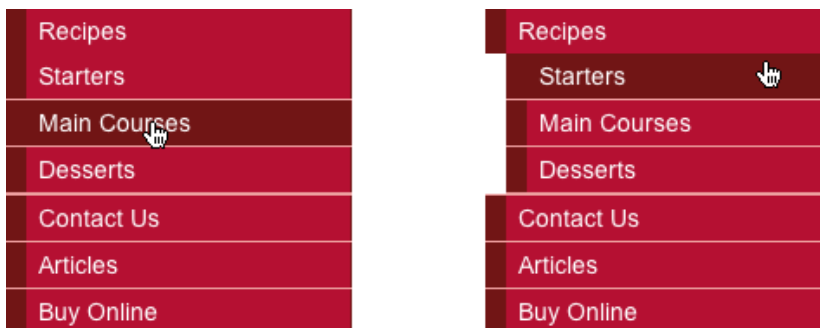
случае элементы вложенного списка унаследуют стили, определенные для пунктов основного меню.

Оформим вложенный список таким образом, чтобы на вид можно было определить, что это список подразделов, отличный от основного меню:

*chapter04/listnav\_sub.css (фрагмент)*

```
#navigation ul ul {
    margin-left: 12px;
}
```

Данное правило добавляет для вложенного списка внешний отступ от левого края меню и приводит к выравниванию всех пунктов меню по правому краю, как показано на рис. 4.7, справа.



*Рис. 4.7. Список второго уровня в стиле основного меню и тот же список с добавлением отступов*

Наконец, добавим несколько простых правил стилей для элементов `li` и `a`, расположенных внутри вложенного списка:

*chapter04/listnav\_sub.css (фрагмент)*

```
#navigation ul ul li {
    border-bottom: 1px solid #711515;
    margin: 0;
}
#navigation ul ul a:link, #navigation ul ul a:visited {
    background-color: #ED9F9F;
    color: #711515;
}
#navigation ul ul a:hover {
    background-color: #711515;
    color: #FFFFFF;
}
```

## Создание горизонтального меню с помощью списков и CSS

Во всех предыдущих примерах мы рассматривали вертикальную панель навигации, которую чаще всего можно встретить на сайте слева или справа по отношению к области с основным контентом. Однако навигационные ссылки также часто располагаются по горизонтали в верхней части документа.

### Решение

Как видно на рис. 4.8, такое меню можно создать путем написания CSS-стилей для списка. Свойству `display` для элементов `li` нужно присвоить значение `inline` во избежание переноса каждого следующего пункта на новую строку.



*Рис. 4.8. Создание горизонтально расположенной панели навигации с помощью CSS*

Ниже представлен использованный HTML- и CSS-код.

#### *chapter04/listnav\_horiz.html*

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
<head>
<title>Lists as navigation</title>
<meta http-equiv="content-type"
  content="text/html; charset=utf-8" />
<link rel="stylesheet" type="text/css" href="listnav_horiz.css" />
</head>
<body>
<div id="navigation">
  <ul>
    <li><a href="#">Recipes</a></li>
    <li><a href="#">Contact Us</a></li>
    <li><a href="#">Articles</a></li>
    <li><a href="#">Buy Online</a></li>
  </ul>
</div>
</body>
</html>
```

#### *chapter04/listnav\_horiz.css*

```
body {
  padding: 1em;
```

```
}
#navigation {
  font-size: 90%;
}
#navigation ul {
  list-style: none;
  margin: 0;
  padding: 0;
  padding-top: 1em;
}
#navigation li {
  display: inline;
}
#navigation a:link, #navigation a:visited {
  padding: 0.4em 1em 0.4em 1em;
  color: #FFFFFF;
  background-color: #B51032;
  text-decoration: none;
  border: 1px solid #711515;
}
#navigation a:hover {
  color: #FFFFFF;
  background-color: #711515;
}
```

## Обсуждение

Чтобы разместить пункты меню по горизонтали, вначале создадим список, аналогичный тому, что был использован для реализации вертикальной панели навигации:

*chapter04/listnav\_horiz.html (фрагмент)*

```
<div id="navigation">
  <ul>
    <li><a href="#">Recipes</a></li>
    <li><a href="#">Contact Us</a></li>
    <li><a href="#">Articles</a></li>
    <li><a href="#">Buy Online</a></li>
  </ul>
</div>
```

Напишем для контейнера `#navigation` несколько правил стилей для задания базовых параметров шрифта, как и в примере с вертикальным меню. Для данного элемента также скорее всего потребуются задать свойства для его позиционирования на странице:

*chapter04/listnav\_horiz.css (фрагмент)*

```
#navigation {
  font-size: 90%;
}
```

Затем уберем маркеры и отступы, создаваемые браузером по умолчанию для элемента `ul`:

*chapter04/listnav\_horiz.css (фрагмент)*

```
#navigation ul {
  list-style: none;
  margin: 0;
  padding: 0;
  padding-top: 1em;
}
```

Свойство, располагающее список по горизонтали, применяется по отношению к элементу `li`. Таким образом, после присваивания свойству `display` значения `inline` список будет выглядеть, как показано на рис. 4.9.

*chapter04/listnav\_horiz.css (фрагмент)*

```
#navigation li {
  display: inline;
}
```



Recipes Contact Us Articles Buy Online

*Рис. 4.9. Отображение пунктов меню по горизонтали*

Теперь остается только определить стиливое оформление ссылок навигационной панели:

*chapter04/listnav\_horiz.css (фрагмент)*

```
#navigation a:link, #navigation a:visited {
  padding: 0.4em 1em 0.4em 1em;
  color: #FFFFFF;
  background-color: #B51032;
  text-decoration: none;
  border: 1px solid #711515;
}
#navigation a:hover {
  color: #FFFFFF;
  background-color: #711515;
}
```

Если вы хотите, чтобы каждая ссылка располагалась в «окошке», как в этом примере, запомните, что для создания пустого пространства между его краями и ссылкой необходимо задать больше внутренних отступов (`padding`) справа и слева, а для отделения элементов навигационного меню друг от друга – задать внешние отступы (`margin`) справа и слева.

## Создание средствами CSS навигационной панели с кнопками

На многих сайтах используется панель навигации, по виду состоящая из кнопок. Такого рода меню часто создают с помощью изображений, к которым применяют различные эффекты для имитации кнопки. Для смены изображения нередко пишется сценарий на JavaScript, что позволяет создать впечатление, будто кнопки продавливаются при наведении на них указателя мыши или при щелчке.

У вас может возникнуть вполне закономерный вопрос: «Возможно ли создать такую панель навигации с кнопками исключительно средствами CSS?» Ответ однозначный – да!

### Решение

С помощью CSS можно без особого труда сделать элементы панели навигации похожими на кнопки. Такой эффект основан на использовании свойства border.



Рис. 4.10. Создание навигационного меню с кнопками средствами CSS

Ниже представлен необходимый код:

*chapter04/listnav\_button.html*

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
<head>
<title>Lists as navigation</title>
<meta http-equiv="content-type"
  content="text/html; charset=utf-8" />
<link rel="stylesheet" type="text/css" href="listnav_button.css"
  />
</head>
<body>
<div id="navigation">
  <ul>
    <li><a href="#">Recipes</a></li>
    <li><a href="#">Contact Us</a></li>
    <li><a href="#">Articles</a></li>
    <li><a href="#">Buy Online</a></li>
  </ul>
</div>
</body>
</html>
```

*chapter04/listnav\_button.css*

```
#navigation {
    font-size:90%
}
#navigation ul {
    list-style: none;
    margin: 0;
    padding: 0;
    padding-top: 1em;
}
#navigation li {
    display: inline;
}
#navigation a:link, #navigation a:visited {
    margin-right: 0.2em;
    padding: 0.2em 0.6em 0.2em 0.6em;
    color: #A62020;
    background-color: #FCE6EA;
    text-decoration: none;
    border-top: 1px solid #FFFFFF;
    border-left: 1px solid #FFFFFF;
    border-bottom: 1px solid #717171;
    border-right: 1px solid #717171;
}
#navigation a:hover {
    border-top: 1px solid #717171;
    border-left: 1px solid #717171;
    border-bottom: 1px solid #FFFFFF;
    border-right: 1px solid #FFFFFF;
}
```

## Обсуждение

Для создания такого эффекта мы будем использовать горизонтальное меню навигации, созданное в разделе «Создание горизонтального меню с помощью списков и CSS». Чтобы пункты меню стали похожими на кнопки, окрасим левую и нижнюю части рамки одним цветом, а верхнюю и правую – другим, более светлым. Таким образом, кнопка будет казаться объемной:

*chapter04/listnav\_button.css (фрагмент)*

```
#navigation a:link, #navigation a:visited {
    margin-right: 0.2em;
    padding: 0.2em 0.6em 0.2em 0.6em;
    color: #A62020;
    background-color: #FCE6EA;
    text-decoration: none;
    border-top: 1px solid #FFFFFF;
    border-left: 1px solid #FFFFFF;
```

```
border-bottom: 1px solid #717171;
border-right: 1px solid #717171;
}
```

При наведении указателя мыши на кнопку цвет разных частей рамки поменяется местами, создавая эффект нажатия:

*chapter04/listnav\_button.css (фрагмент)*

```
#navigation a:hover {
border-top: 1px solid #717171;
border-left: 1px solid #717171;
border-bottom: 1px solid #FFFFFF;
border-right: 1px solid #FFFFFF;
}
```

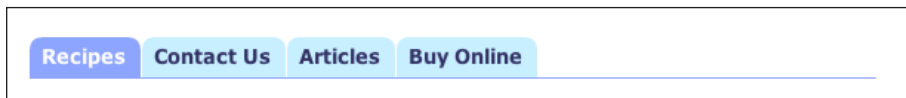
Поэкспериментируйте с толщиной рамки и фоновыми изображениями для создания оптимального сочетания, гармонирующего с дизайном вашей страницы.

## Создание с помощью CSS панели навигации на основе вкладок

На многих веб-сайтах в верхней части страницы можно встретить навигационное меню, состоящее из вкладок. Его часто создают с использованием изображений. Но такой подход снижает доступность сайта и может порождать сложности, если навигация создается с помощью системы управления контентом (CMS), предоставляющей пользователю возможность удаления и добавления вкладок, а также изменения текста ссылок. Однако вкладки можно создать с помощью фоновых изображений и текста, оформленных с помощью CSS.

### Решение

Изображенная на рис. 4.11 панель навигации с вкладками была создана путем написания правил стилей для горизонтального списка.



*Рис. 4.11. Создание навигационной панели с вкладками с помощью CSS*

Ниже представлен использованный HTML- и CSS-код.

*chapter04/tabs.html*

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
```



```

<head>
<title>Lists as navigation</title>
<meta http-equiv="content-type"
  content="text/html; charset=utf-8" />
<link rel="stylesheet" type="text/css" href="tabs.css" />
</head>
<body id="recipes">
<div id="header">
<ul>
  <li class="recipes"><a href="#">Recipes</a></li>
  <li class="contact"><a href="#">Contact Us</a></li>
  <li class="articles"><a href="#">Articles</a></li>
  <li class="buy"><a href="#">Buy Online</a></li>
</ul>
</div>
<div id="content">
<h1>Recipes</h1>
<p>Lorem ipsum dolor sit amet, ... </p>
</div>
</body>
</html>

```

#### *chapter04/tabs.css*

```

body {
  font: .8em/1.8em verdana, arial, sans-serif;
  background-color: #FFFFFF;
  color: #000000;
  margin: 0 10% 0 10%;
}

#header {
  float: left;
  width: 100%;
  border-bottom: 1px solid #8DA5FF;
  margin-bottom: 2em;
}

#header ul {
  margin: 0;
  padding: 2em 0 0 0;
  list-style: none;
}

#header li {
  float: left;
  background-image: url("images/tab_left.gif");
  background-repeat: no-repeat;
  margin: 0 1px 0 0;
  padding: 0 0 0 8px;
}

```

```
#header a {
  float: left;
  display: block;
  background-image: url("images/tab_right.gif");
  background-repeat: no-repeat;
  background-position: right top;
  padding: 0.2em 10px 0.2em 0;
  text-decoration: none;
  font-weight: bold;
  color: #333366;
}
#recipes #header li.recipes,
#contact #header li.contact,
#articles #header li.articles,
#buy #header li.buy {
  background-image: url("images/tab_active_left.gif");
}

#recipes #header li.recipes a,
#contact #header li.contact a,
#articles #header li.articles a,
#buy #header li.buy a {
  background-image: url("images/tab_active_right.gif");
  background-color: transparent;
  color:#FFFFFF;
}
}
```

## Обсуждение

Для создания навигационного меню со вкладками в данном случае был использован многократно проверенный на практике метод «раздвижных дверей» Дугласа Боумана (Douglas Bowman).<sup>1</sup> По структуре меню навигации является маркированным списком, подобным тому, с которым мы работали на протяжении настоящей главы, за тем лишь исключением, что теперь каждому элементу `li` был присвоен атрибут `class` для описания содержащейся внутри ссылки. Кроме того, список является дочерним элементом контейнера `div` с идентификатором (`id`) `header`. Название приема возникло по аналогии – используемые два изображения перекрывают друг друга и при увеличении размера шрифта «разъезжаются».

Для создания такого эффекта нам потребуется четыре изображения: два для представления обычного вида вкладки и два для представления ее вида при наведении указателя мыши. Используемые в данном примере изображения показаны на рис. 4.12. Как видите, их размер значительно превышает обычный размер вкладки, что обеспечивает возможность ее увеличения, например, если пользователь сделает шрифт крупнее.

---

<sup>1</sup> <http://www.alistapart.com/articles/slidingdoors/>

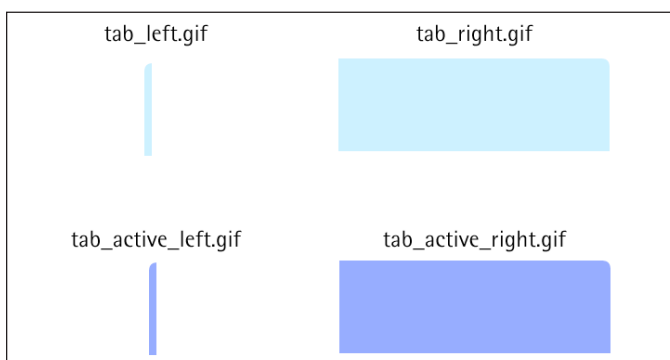


Рис. 4.12. Файлы изображений для создания вкладок

Ниже представлен список основных пунктов меню:

*chapter04/tabs.html (фрагмент)*

```
<div id="header">
  <ul>
    <li class="recipes"><a href="#">Recipes</a></li>
    <li class="contact"><a href="#">Contact Us</a></li>
    <li class="articles"><a href="#">Articles</a></li>
    <li class="buy"><a href="#">Buy Online</a></li>
  </ul>
</div>
```

Прежде всего нужно задать стиль контейнера, окружающего меню навигации. В данном упражнении достаточно будет указать для него параметры нижней рамки, но на реальных сайтах он, скорее всего, будет содержать другие элементы помимо вкладок (например, логотип или поле для ввода поискового запроса).

```
#header {
  float: left;
  width: 100%;
  border-bottom: 1px solid #8DA5FF;
  margin-bottom: 2em;
}
```

Как видите, контейнер объявлен плавающим с помощью свойства `float` с выравниванием влево. Это свойство нужно указать и для отдельных пунктов списка; объявление родительского контейнера плавающим гарантирует, что плавающие элементы списка останутся внутри контейнера; соответственно, рамка всегда будет располагаться под ними.

Далее создадим стиль для дочернего элемента `ul`:

*chapter04/tabs.css (фрагмент)*

```
#header ul {
  margin: 0;
```

```
padding: 2em 0 0 0;  
list-style: none;  
}
```

Данное правило стиля удаляет маркеры списка и изменяет размер внешних и внутренних отступов. Теперь внутренний отступ элемента `ul` сверху составит `2em`. На рис. 4.13 отображается текущий результат преобразований:

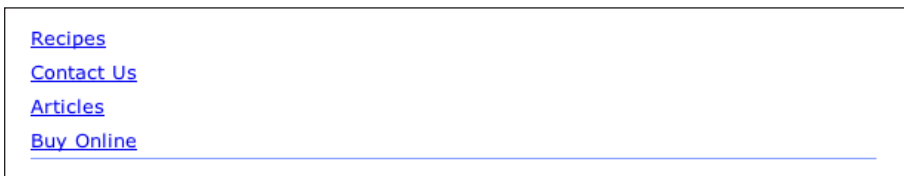


Рис. 4.13. Вид меню навигации после задания стиля элемента `ul`

Настало время создать стили для пунктов списка:

*chapter04/tabs.css (фрагмент)*

```
#header li {  
  float: left;  
  background-image: url("images/tab_left.gif");  
  background-repeat: no-repeat;  
  margin: 0 1px 0 0;  
  padding: 0 0 0 8px;  
}
```

С помощью свойства `float` пункты списка располагаются по горизонтали, оставаясь при этом блочными элементами. Затем на сцену выходит одно из изображений «раздвижных дверей» – тонкая полоса, отображаемая слева от вкладки. Оно будет выступать в качестве фонового. Отступ справа в один пиксел создает зазор между вкладками. На рис. 4.14 видно, что теперь слева от каждой вкладки отображается соответствующее изображение.



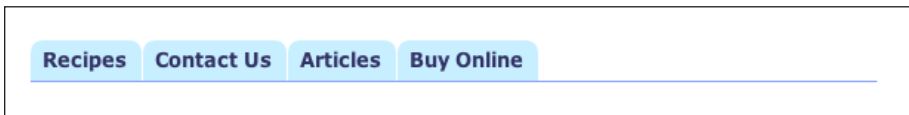
Рис. 4.14. Панель навигации после добавления новых стилей

Последним штрихом к оформлению панели навигации в обычном состоянии станет написание стилей для ссылок. Каждая из них должна отображаться на фоне второго изображения, расположенного справа от уже имеющегося, формируя вкладку:

*chapter04/tabs.css (фрагмент)*

```
#header a {
  float: left;
  display: block;
  background-image: url("images/tab_right.gif");
  background-repeat: no-repeat;
  background-position: right top;
  padding: 0.2em 10px 0.2em 0;
  text-decoration: none;
  font-weight: bold;
  color: #333366;
}
```

Результат показан на рис. 4.15.



*Рис. 4.15. Внешний вид вкладок*

Попробуйте увеличить размер шрифта в браузере, и вы увидите, что размер вкладок изменится соответствующим образом. При этом они не перекрывают друг друга, а текст не вылезает за пределы вкладки – все это благодаря тому, что мы взяли размер изображения «с запасом».

Чтобы придать панели навигации более завершенный вид, нужно выделить вкладку, соответствующую текущей странице. Если помните, каждой ссылке было присвоено уникальное имя класса. Присвоим элементу `body` атрибут `ID` с таким же значением, а остальное за нас сделает CSS.

*chapter04/tabs.html (фрагмент)*

```
<body id="recipes">
```

Хотя может показаться, что CSS-код довольно длинен, но применяемый нами подход для выделения вкладки, имя класса которой совпадает с идентификатором элемента `body`, предельно прост. Используемые изображения в точности копируют две составляющие вкладку картинки, но окрашены другим цветом, за счет чего создается эффект выделения одной вкладки.

Ниже представлен CSS-код:

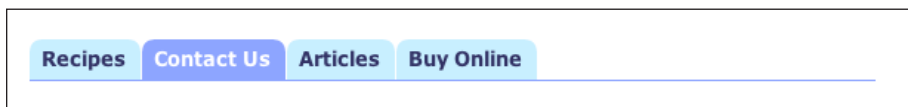
*chapter04/tabs.css (фрагмент)*

```
#recipes #header li.recipes,
#contact #header li.contact,
#articles #header li.articles,
#buy #header li.buy {
```

```
background-image: url("images/tab_active_left.gif");
}

#recipes #header li.recipes a,
#contact #header li.contact a,
#articles #header li.articles a,
#buy #header li.buy a {
background-image: url("images/tab_active_right.gif");
background-color: transparent;
color: #FFFFFF;
}
```

При использовании таких правил стилей достаточно задать элементу `body` атрибут `ID` со значением `recipes`, и соответствующая вкладка `Recipes` будет выделена; если значением будет `contact`, выделена будет вкладка `Contact Us` и т. д. Результат проделанной работы можно увидеть на рис. 4.16.



*Рис. 4.16. Выделение вкладки `Contact Us` путем присвоения значения `contact` атрибуту `ID` элемента `body`*

### Совет

**Обратите внимание на полезный прием.** Указание атрибута `ID` для элемента `body` может сослужить хорошую службу. К примеру, различные разделы вашего сайта могут иметь разное цветовое оформление, помогая пользователям сориентироваться. Для этого нужно задать элементу `body` идентификатор со значением, соответствующим названию раздела, а затем использовать его в таблице стилей, как в данном примере.

## Выделение ссылок, ведущих на внешний сайт

При создании ссылки на внешние ресурсы целесообразно использовать визуальные подсказки, дающие пользователю понять, что ссылка ведет на другой сайт. Это можно сделать с помощью **CSS**, не изменяя разметку самого документа.

### Решение

Для оформления внешних ссылок можно использовать селектор **CSS3**, поддерживаемый многими современными браузерами. В следующем абзаце первая ссылка ведет на другую страницу того же сайта, а вторая – на другой веб-сайт (Google).

*chapter04/external\_links.html*

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
  <head>
    <title>Chapter 4 - Show external links</title>
    <meta http-equiv="content-type" content="text/html;
      charset=utf-8" />
    <link rel="stylesheet" type="text/css"
      href="external_links.css" />
  </head>
  <body>
    <p>Lorem ipsum dolor sit amet, <a href="page2.html">consectetur
adipiscing elit</a>. Aenean porta. Donec eget quam. Morbi libero. Curabitur
ut justo vehicula elit feugiat lacinia. Morbi
ac quam. <a href="http://www.google.com">Sed venenatis</a>,
lectus quis porta viverra, lectus sapien tempus odio, ac
volutpat mi dolor ac elit.</p>
  </body>
</html>

```

Для обращения к ссылке, начинающейся с `http:`, и добавления к ней пиктограммы можно использовать селектор, описанный в спецификации CSS3:

*chapter04/external\_links.css*

```

a[href ^="http:"] {
  padding-left: 20px;
  background-image: url(link_icon_external.gif);
  background-repeat: no-repeat;
}

```

Lorem ipsum dolor sit amet, [consectetur adipiscing elit](#). Aenean porta. Donec eget quam. Morbi libero. Curabitur ut justo vehicula elit feugiat lacinia. Morbi ac quam. 🌐 [Sed venenatis](#), lectus quis porta viverra, lectus sapien tempus odio, ac volutpat mi dolor ac elit.

**Рис. 4.17.** Рядом со ссылкой, ведущей на другой сайт, отображается пиктограмма

Рядом со всеми ссылками, начинающимися с `http:` (они должны вести на внешние ресурсы, поскольку для ссылок, ведущих на другие страницы того же сайта не нужно вводить `http:`), будет показана пиктограмма в виде земного шара.

## Обсуждение

Селекторы атрибутов CSS3 поддерживаются большинством современных браузеров (исключение составляет **Internet Explorer 6**). В остальных браузерах вид ссылки не изменится, поэтому можно считать этот эффект приятным дополнением для пользователей более новых браузеров.

Рассмотрим селектор атрибута `a[href ^= "http:"]` более подробно.

Атрибутом для данного селектора выбран `href`, и мы хотим, чтобы он ссылался на элементы, у которых значение данного атрибута начинается с `http:`. Оператор `^=` означает «начинается с». С помощью похожего селектора атрибута можно, к примеру, выбрать все ссылки на электронные адреса: `a[href ^= "mailto:"]`.

Селекторы атрибутов удобно также использовать для анализа расширения файла, на который указывает ссылка. Например, анализируя расширение файла в тексте ссылки, можно добавить значок, указывающий на то, что это файл в формате PDF и т. п. Селектор `a[href $=".pdf"]` указывает на все ссылки с расширением файла `.pdf`. Оператор `$=` означает «заканчивается на», поэтому данный селектор выбирает все элементы, значение атрибута `href` которых заканчивается на `.pdf`. Ниже представлен пример использования всех рассмотренных трех селекторов:

### *chapter04/external\_links.html*

```
<ul class="links">
  <li><a href="http://www.google.com">Go somewhere else</a></li>
  <li><a href="/files/example.pdf">Download a PDF</a></li>
  <li><a href="mailto:info@example.com">Email someone</a></li>
</ul>
```

### *chapter04/external\_links.css*

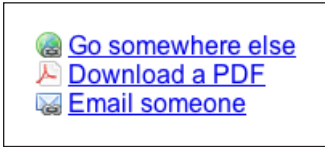
```
a[href ^= "http:"] {
  padding-left: 20px;
  background-image: url(link_icon_external.gif);
  background-repeat: no-repeat;
}

a[href ^= "mailto:"] {
  padding-left: 20px;
  background-image: url(link_icon_email.gif);
  background-repeat: no-repeat;
}

a[href $=".pdf"] {
  padding-left: 20px;
  background-image: url(link_icon_pdf.gif);
}
```



```
background-repeat: no-repeat;
}
```



*Рис. 4.18. Ссылки с пиктограммами, указывающими, что они ведут на внешний ресурс, на адрес электронной почты и файл pdf<sup>1</sup>*

Селекторы атрибутов полезно использовать в подобных ситуациях – для внесения привлекательных дополнений в дизайн вашей страницы.

## Изменение вида курсора

При наведении курсора на ссылку или какой-либо иной элемент страницы он нередко приобретает вид руки. Иногда у разработчика может возникнуть необходимость в изменении вида курсора для акцентирования внимания на каком-либо действии (или для соответствия определенному дизайну интерфейса).

### Решение

Вид курсора можно изменить с помощью CSS-свойства `cursor`. К примеру, чтобы курсор изменял свой вид при наведении на ссылки на справочную документацию, можно задать следующее правило стиля:

```
a.help {
  cursor: help;
}
```

В табл. 4.1 представлены доступные в CSS 2.1 свойства и результат их применения при просмотре документа в браузере Internet Explorer 8.

### Обсуждение

Свойство `cursor` может принимать много различных значений. Изменение вида курсора в веб-приложении – полезный прием, позволяющий сделать интерфейс более дружелюбным. Это дает пользователю дополнительную информацию о назначении тех или иных элементов (например, при наведении курсора на справочный текст он приобретает вид вопросительного знака).

<sup>1</sup> Пиктограммы, представленные на рис. 4.18, взяты с <http://www.famfamfam.com/lab/icons/silk/>.

**Предупреждение**

**Изменение вида курсора может привести к путанице!** Описанный прием следует использовать с осторожностью, принимая во внимание тот факт, что пользователи обычно привыкают к стандартному поведению браузера – например, к тому, что при наведении на ссылку курсор превращается в руку с указующим пальцем.







Таблица 4.1. Стандартные курсоры CSS2.1 при просмотре в IE8

 pointer	 default	 crosshair
 text	 help	 move
 n-resize	 ne-resize	 nw-resize
 s-resize	 se-resize	 sw-resize
 e-resize	 w-resize	 wait
 progress	определяется браузером auto	пользовательское изображение url("url")

В табл. 4.1 представлены различные значения CSS-свойства `cursor`. Они поддерживаются большинством современных браузеров, включая **Internet Explorer 6 и выше, Safari, Opera, Firefox и Chrome**. Однако отдельные значения могут не обрабатываться, поэтому не забывайте о необходимости тестирования.

В спецификации CSS3 описаны дополнительные значения данного свойства, представленные в табл. 4.2, однако их поддержка реализована в полной мере браузерами Safari, Firefox и Chrome; Internet Explorer 8 также поддерживает большинство из них, однако их поддержка браузером Opera на момент написания данной книги отсутствовала (он поддерживает только значения CSS 2.1).

Таблица 4.2. Новые значения свойства *cursor* в CSS3

Значение	Отображение (как в IE8)
copy	поддержка отсутствует
alias	поддержка отсутствует
cell	поддержка отсутствует
all-scroll	
no-drop	
not-allowed	
col-resize	
row-resize	
vertical-text	

## Реализация смены изображений на панели навигации без использования JavaScript

При создании навигации исключительно средствами CSS можно добиться многих интересных эффектов, но для реализации некоторых из них все же не обойтись без изображений. Можно ли сохранить все преимущества созданной на основе текста навигации и добавить при этом изображения?

### Решение

С помощью CSS можно создать эффект смены изображений без сценариев на JavaScript. Данное решение основано на приемах, описанных на WellStyled.com.<sup>1</sup> Ниже представлен необходимый код:

*chapter04/images.html*

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">

```

<sup>1</sup> <http://wellstyled.com/css-nopreload-rollovers.html>.

```
<head>
<title>Lists as navigation</title>
<meta http-equiv="content-type"
  content="text/html; charset=utf-8" />
<link rel="stylesheet" type="text/css" href="images.css" />
</head>
<body>
<ul id="nav">
  <li><a href="#">Recipes</a></li>
  <li><a href="#">Contact Us</a></li>
  <li><a href="#">Articles</a></li>
  <li><a href="#">Buy Online</a></li>
</ul>
</body>
</html>
```

#### *chapter04/images.css*

```
ul#nav {
  list-style-type: none;
  padding: 0;
  margin: 0;
}
#nav a:link, #nav a:visited {
  display: block;
  width: 150px;
  padding: 10px 0 16px 32px;
  font: bold 80% Arial, Helvetica, sans-serif;
  color: #FF9900;
  background: url("peppers.gif") top left no-repeat;
  text-decoration: none;
}
#nav a:hover {
  background-position: 0 -69px;
  color: #B51032;
}
#nav a:active {
  background-position: 0 -138px;
  color: #006E01;
}
```

Результат обработки данного кода показан на рис. 4.19, однако чтобы увидеть полученный эффект во всей красе, попробуйте самостоятельно создать подобный документ и щелкнуть по ссылке!

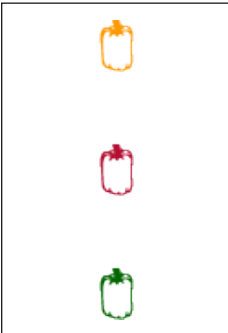
## Обсуждение

Данное решение основано на использовании на панели навигации изображений без необходимости предварительной загрузки множества отдельных файлов.



*Рис. 4.19. Добавление нового эффекта к готовому меню с помощью изображений*

Каждый элемент панели навигации может находиться в одном из трех состояний, однако для их отображения вовсе не обязательно использовать отдельные изображения. Вместо этого мы будем работать с одним большим изображением, включающим в себя графические элементы, соответствующие трем разным состояниям, как показано на рис. 4.20.



*Рис. 4.20. Три состояния элемента навигации*

Разметка панели навигации представляет собой простой список:

*chapter04/images.html (фрагмент)*

```
<ul id="nav">
  <li><a href="#">Recipes</a></li>
  <li><a href="#">Contact Us</a></li>
  <li><a href="#">Articles</a></li>
  <li><a href="#">Buy Online</a></li>
</ul>
```

Стили для управления отображением фонового изображения расположены в блоке описаний для навигационных ссылок. Поскольку размер изображения превышает размер отведенной для него области, поначалу мы увидим только желтый перец:

*chapter04/images.css (фрагмент)*

```
#nav a:link, #nav a:visited {
    display: block;
    width: 150px;
    padding: 10px 0 16px 32px;
    font: bold 80% Arial, Helvetica, sans-serif;
    color: #FF9900;
    background: url("peppers.gif") top left no-repeat;
    text-decoration: none;
}
```

При активации состояния `:hover` фоновое изображение перемещается вверх ровно на столько пикселей, чтобы пользователь увидел рисунок красного перца. В данном примере используется значение 69 пикселей, которое может меняться в зависимости от используемого вами изображения. Значение можно вычислить или подобрать на практике, каждый раз передвигая фоновое изображение на несколько пикселей до тех пор, пока не будет найдена оптимальная позиция:

*chapter04/images.css (фрагмент)*

```
#nav a:hover {
    background-position: 0 -69px;
    color: #B51032;
}
```

При активации состояния `:active` фоновое изображение передвигается вновь, и на этот раз при щелчке по ссылке мы видим изображение зеленого перца:

*chapter04/images.css (фрагмент)*

```
#nav a:active {
    background-position: 0 -138px;
    color: #006E01;
}
```

Вот и все! Данный эффект может быть испорчен, если пользователь увеличит размер шрифта, поскольку при этом могут быть видны края скрытых изображений. Не забывайте о возможности возникновения такой ситуации – ее можно предотвратить, оставив больше пустого пространства между изображениями.

**Предупреждение**

**Мелькание в Internet Explorer.** При использовании описанного выше приема возможно мелькание элементов навигации в Internet Explorer. По результатам тестирования я пришла к выводу, что такой негативный эффект может возникнуть при использовании изображений большего размера, чем в данном примере. Если вы столкнулись с такой проблемой, то вам поможет подробно описанный способ ее решения.<sup>1</sup>

---

<sup>1</sup> <http://wellstyle.com/css-nopreload-rollovers.html>

## Оформление карты сайта

Карта сайта располагается на отдельной странице и содержит список всех его разделов. Ею можно воспользоваться в том случае, когда не получается найти необходимую информацию с помощью навигационного меню. Кроме того, она представляет собой краткий обзор содержания сайта и дает возможность перехода к заинтересовавшему разделу одним щелчком мыши.

### Решение

Поскольку карта сайта является списком доступных разделов вашего сайта, ее лучше всего разметить как набор вложенных списков. Пунктами основного списка станут основные навигационные разделы; каждый из пунктов будет содержать подпункты – навигацию второго уровня. Такой метод подойдет и для сайтов с многоуровневой системой навигации; благодаря такой разметке новые пункты будет легко создавать с помощью системы управления контентом. На рис. 4.21 отображается результат обработки следующего кода:

*chapter04/sitemap.html (фрагмент)*

```
<ul id="sitemap">
  <li><a href="/about">About us</a>
  <ul>
    <li><a href="/about/team">The team</a></li>
    <li><a href="/about/history">Our history</a></li>
  </ul>
</li>
<li><a href="/products">Our products</a></li>
<li><a href="/order">Ordering information</a>
<ul>
  <li><a href="/order/shops">Our shops</a></li>
  <li><a href="/order/stockists">Other stockists</a></li>
  <li><a href="/order/onlinestockists">Online stockists</a></li>
</ul>
</li>
<li><a href="/contact">Contact us</a></li>
</ul>
```

*chapter04/sitemap.css (фрагмент)*

```
ul#sitemap {
  margin: 0;
  padding: 0;
  list-style: none;
}

ul#sitemap ul {
  padding-left: 1em;
  list-style: none;
```

```
}

ul#sitemap li {
  border-bottom: 2px solid #FFFFFF;
}

ul#sitemap li a:link, ul#sitemap li a:visited{
  background-color: #CCCCCC;
  display: block;
  padding: 0.4em;
  text-decoration: none;
  color: #057FAC;
}

ul#sitemap li a:hover {
  background-color: #999999;
  color: #FFFFFF;
}

ul#sitemap li li a:link, ul#sitemap li li a:visited{
  background-color: #FFFFFF;
  display: block;
  padding: 0.4em;
}

ul#sitemap li li a:hover {
  background-color: #FFFFFF;
  color: #057FAC;
}
```

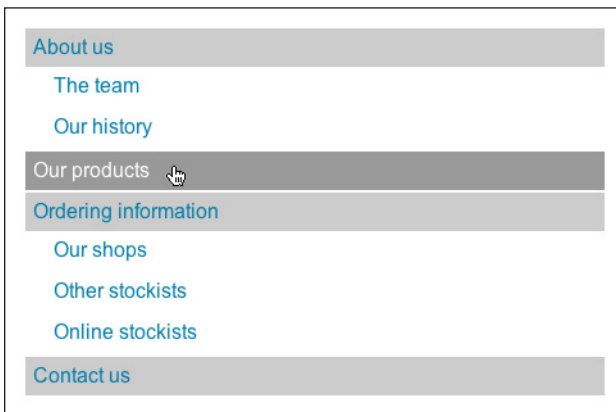


Рис. 4.21. Оформленная с помощью CSS карта сайта

## Обсуждение

Вначале карта сайта представлена в виде списка основных элементов навигации и вложенных подменю, как и в рассмотренном ранее примере



создания навигации на основе списка. Отличие лишь в том, что на карте сайта все пункты списка, в том числе и подпункты, отображаются одновременно. Если навигация включает большее количество уровней, их можно представить по тому же принципу – в виде вложенных списков.

Будьте внимательны при работе со вложенными списками. Вложенный список необходимо разместить перед закрывающим тегом `</li>` родительского списка. Без применения таблицы стилей карта сайта будет выглядеть, как на рис. 4.22. Теперь создаем стили для родительского списка и вложенных списков. Для первого списка я убрала внутренние и внешние отступы, а для вложенных списков добавила отступ слева размером в `1em`, чтобы сразу было видно, что они представляют собой следующий уровень навигации:

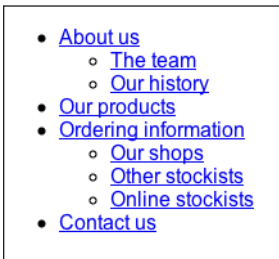


Рис. 4.22. Вид карты сайта без использования CSS

*chapter04/sitemap.css (фрагмент)*

```

ul#sitemap {
    margin: 0;
    padding: 0;
    list-style: none;
}

ul#sitemap ul {
    padding-left: 1em;
    list-style: none;
}

```

Теперь следует создать отличительный стиль пунктов основного списка, чтобы подчеркнуть, что они представляют основные разделы сайта. Как и при оформлении навигации, я задаю стиль элементу, присвоив значение `block` свойству `display`, чтобы пользователь для перехода по ссылке мог щелкнуть по всей занимаемой элементом области, а не только по самому тексту:

*chapter04/sitemap.css (фрагмент)*

```

ul#sitemap li {
    border-bottom: 2px solid #FFFFFF;
}

```

```
ul#sitemap li a:link, ul#sitemap li a:visited{
    background-color: #CCCCCC;
    display: block;
    padding: 0.4em;
    text-decoration: none;
    color: #057FAC;
}

ul#sitemap li a:hover {
    background-color: #999999;
    color: #FFFFFF;
}
```

Данные стили будут применены по отношению к элементам списка, поэтому необходимо также задать специальные стили для вложенных ссылок, поскольку у них не должно быть серого фона. Поэтому воспользуемся более конкретным селектором для выбора этих элементов:

*chapter04/sitemap.css (фрагмент)*

```
ul#sitemap li li a:link, ul#sitemap li li a:visited{
    background-color: #FFFFFF;
    display: block;
    padding: 0.4em;
}

ul#sitemap li li a:hover {
    background-color: #FFFFFF;
    color: #057FAC;
}
```

Вот мы и написали таблицу стилей для оформления карты сайта. На основе представленных приемов вы вполне сможете создать свои собственные стили.

## Создание выпадающего меню исключительно средствами CSS

В предыдущем издании данной книги был представлен пример создания выпадающего меню только с помощью каскадных таблиц стилей. Я приняла решение не включать его в настоящее издание.

Когда создавался первый вариант данной книги, я, как и многие веб-разработчики, смотрела вперед с надеждой, что в будущем с помощью CSS можно будет создавать все, что пожелаешь, в том числе выпадающие меню и подобные эффекты без использования JavaScript. Более глубокое изучение особенностей перечисленных технологий и поведения пользователей позволило мне прийти к выводу, что в данном случае куда эффективнее воспользоваться средствами JavaScript.

В принципе, создать выпадающее меню с помощью CSS вполне возможно. Однако на практике такое меню гораздо менее доступно по сравнению с хорошо продуманным выпадающим меню, реализованном с помощью JavaScript. По нему можно, например, перемещаться с помощью клавиатуры, а также управлять его видимостью с помощью описания `display: none` так, что оно будет скрыто от экранных дикторов и пользователей, просматривающих сайт с помощью обычного браузера. Если вы хотите создать выпадающее меню, рекомендую вам ознакомиться с отличным меню на сайте UDM4, которое можно оформить с помощью CSS и которое обеспечивает равные возможности для всех посетителей вашего сайта.<sup>1</sup>

## Создание доступного навигационного меню на основе изображений с помощью CSS

Метод создания текстовой навигации с помощью CSS обладает массой преимуществ, однако в определенных случаях у вас может возникнуть необходимость в использовании изображений для достижения определенного эффекта или отображения специфического шрифта. При этом необходимо использовать все имеющиеся знания о технологии CSS, чтобы использование изображений не породило дополнительных проблем. Предлагаемое ниже решение сочетает несколько различных приемов и трюков для создания навигационного меню с помощью изображений.

### Решение

Вначале создадим текстовое навигационное меню и затем заменим его на графическое, опираясь на одно изображение и CSS. На рис. 4.23 показан результат выполнения следующего кода.

*chapter04/image\_nav.html (фрагмент)*

```
<ul id="nav">
  <li class="recipes"><a href="#"><span>Recipes</span></a></li>
  <li class="contact"><a href="#"><span>Contact Us</span></a></li>
  <li class="articles"><a href="#"><span>Articles</span></a></li>
  <li class="buy"><a href="#"><span>Buy Online</span></a></li>
</ul>
```

*chapter04/image\_nav.css (фрагмент)*

```
ul#nav {
  width: 360px;
  height: 30px;
  overflow: hidden;
  margin: 0;
```

---

<sup>1</sup> <http://www.udm4.com/>

```
padding: 0;
list-style: none;
}

ul#nav li {
float: left;
}

ul#nav li a span {
margin-left: -5000px;
}

ul#nav li a {
background-image: url(reflectonav.gif);
background-repeat: no-repeat;
display: block;
width: 75px;
overflow: hidden;
}

ul#nav li.recipes a {
background-position: 0 0;
}

ul#nav li.recipes a:hover {
background-position: 0 -42px;
}

ul#nav li.contact a {
background-position: -75px 0;
width: 105px;
}

ul#nav li.contact a:hover {
background-position: -75px -42px;
}

ul#nav li.articles a {
background-position: -180px 0;
width: 85px;
}

ul#nav li.articles a:hover {
background-position: -180px -42px;
}

ul#nav li.buy a {
background-position: -265px 0;
width: 85px;
}
```

```
ul#nav li.buy a:hover {
    background-position: -265px -42px;
}
```



Рис. 4.23. Конечный вид панели навигации

## Обсуждение

Наша цель состоит в создании с помощью изображений навигации, доступной для текстовых устройств, таких как экранные дикторы и поисковые системы. Поэтому вначале необходимо создать меню в виде маркированного списка. Единственным отличием такого списка от представленных в предыдущих примерах является элемент `span`, обрамляющий текст внутри каждого пункта списка:

*chapter04/image\_nav.html (фрагмент)*

```
<ul id="nav">
  <li class="recipes"><a href="#"><span>Recipes</span></a></li>
  <li class="contact"><a href="#"><span>Contact Us</span></a></li>
  <li class="articles"><a href="#"><span>Articles</span></a></li>
  <li class="buy"><a href="#"><span>Buy Online</span></a></li>
</ul>
```

Пока навигация представляет собой простой структурированный список. Теперь настало время для создания изображений. Как и в примере с созданием эффекта смены изображений, мы будем использовать одно изображение, содержащее несколько состояний, в данном случае – все пункты навигации и их вид при наведении курсора, как показано на рис. 4.24.

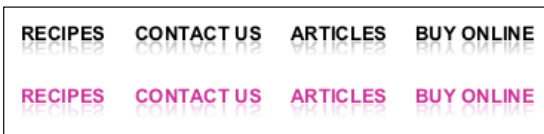


Рис. 4.24. Фоновое изображение, используемое в данном примере

Использование одного изображения избавляет от отправки многократных запросов на сервер, а размер файла, содержащего одно изображение, меньше, чем суммарный размер файла при создании меню с помощью восьми отдельных картинок. Итак, когда у нас есть и список, и изображение, приступим к написанию таблицы стилей. Прежде всего следует изменить используемое по умолчанию оформление элемента `ul` и создать обтекание по левому краю для пунктов списка, чтобы они располагались по горизонтали.

Кроме того, нужно задать ширину и высоту панели навигации. Эти параметры нам хорошо известны, поскольку панель состоит из изображений. Указание высоты позволит избежать отображения лишних частей фонового изображения:

*chapter04/image\_nav.css (фрагмент)*

```
ul#nav {
    width: 360px;
    height: 30px;
    overflow: hidden;
    margin: 0;
    padding: 0;
    list-style: none;
}

ul#nav li {
    float: left;
}
```

Теперь нужно скрыть текст от браузеров, поддерживающих изображения и CSS. Этого можно достичь путем указания отрицательного внешнего отступа для элементов `span` внутри пунктов списка, чтобы они «уехали» за край экрана:

*chapter04/image\_nav.css (фрагмент)*

```
ul#nav li a span {
    margin-left: -5000px;
}
```

Далее заменим текст фоновым изображением. Создадим для элемента `a` правило, в котором свойству `display` задано значение `block`, чтобы он занял всю отведенную для `li` область, а затем добавим фоновое изображение:

*chapter04/image\_nav.css (фрагмент)*

```
ul#nav li a {
    display: block;
    background-image: url(reflectonav.gif);
    background-repeat: no-repeat;
    width: 75px;
    overflow: hidden;
}
```

При просмотре полученного на данный момент результата в браузере вы увидите четырехкратное повторение пункта `Recipes`. Дело в том, что мы задали для каждого пункта списка фоновое изображение, располагающееся в его верхней части, как показано на рис. 4.25.

Для исправления положения необходимо указать различное расположение фонового изображения для каждого пункта навигации. Чаще всего



*Рис. 4.25. Вид навигации после добавления фонового изображения*

его удобнее определить экспериментальным путем, передвигая изображение пиксел за пикселом, пока не будет достигнута оптимальная позиция. В следующем CSS-коде задается позиция фонового изображения для каждой ссылки, а на рис. 4.26 показан результат его выполнения.

*chapter04/image\_nav.css (фрагмент)*

```
ul#nav li.recipes a {
    background-position: 0 0;
}

ul#nav li.contact a {
    background-position: -75px 0;
    width: 105px;
}

ul#nav li.articles a {
    background-position: -180px 0;
    width: 85px;
}

ul#nav li.buy a {
    background-position: -265px 0;
    width: 85px;
}
```



*Рис. 4.26. Вид навигации после определения позиции фонового изображения*

Осталось только добавить стиль для задания вида пункта списка при наведении на него указателя мыши. Это можно сделать таким же способом, как и в предыдущем примере со сменой изображений. С помощью CSS можно передвигать фоновое изображение, чтобы нужная область появилась в поле зрения:

*chapter04/image\_nav.css (фрагмент)*

```
ul#nav li.recipes a:hover {
    background-position: 0 -42px;
}
```

```
ul#nav li.contact a:hover {  
    background-position: -75px -42px;  
}  
  
ul#nav li.articles a:hover {  
    background-position: -180px -42px;  
}  
  
ul#nav li.buy a:hover {  
    background-position: -265px -42px;  
}
```

## Заклучение

В настоящей главе были рассмотрены различные способы создания навигации на основе списков с логической структурой, а также примеры, которые могут послужить отправной точкой для начала собственных экспериментов.

Для уже существующих сайтов, полное изменение дизайна которых представляется нецелесообразным, создание навигации средствами CSS может существенно улучшить параметры доступности и производительности – «малой кровью» и без кардинальных изменений во внешнем виде сайта.



# 5

## Табличные данные

Вам наверняка знакома заповедь: «Таблицы предназначены для табличных данных, а не для описания структуры страницы». Изначально таблицы были созданы для корректного представления табличных данных в HTML-документах, но вскоре их стали широко использовать в иных целях – для разметки всей страницы. В то время любой уважающий себя веб-дизайнер должен был уметь создавать сложные страницы с помощью вложенных таблиц. Однако объем кода при использовании такого метода существенно возрастал, а пользователи текстовых устройств и экранных дикторов испытывали массу неудобств. В настоящее время для верстки страниц веб-стандарты предписывают вместо таблиц использовать каскадные таблицы стилей, специально предназначенные для этих целей. Этот подход обладает гораздо большей гибкостью, в чем мы еще раз убедимся в главе 9.

Однако не следует считать таблицы чем-то абсолютно вредным: их можно (и нужно) использовать по их прямому назначению – для представления табличных данных. В данной главе будут рассмотрены распространенные методы правильного применения таблиц, в том числе примеры использования различных элементов и атрибутов, позволяющие повысить доступность таблиц. Кроме того, будет показано, как сделать таблицы более привлекательными и удобными в использовании для пользователей, просматривающих страницу с помощью обычного веб-браузера.

### Представление табличных данных с помощью CSS

#### Решение

Говоря вкратце, этого вообще не следует делать! Электронные таблицы по определению содержат табличные данные и должны быть представлены средствами HTML. Однако, как увидите, с помощью CSS их мож-

но сделать привлекательнее. Кроме того, не следует пренебрегать вопросами доступности, даже если вы используете таблицы именно для отображения табличных данных.

## Обсуждение

**Табличные данные** – данные, которые можно отобразить в виде таблицы и которые можно логически упорядочить по строкам и столбцам.

Хорошим примером таких данных может послужить электронная таблица, содержащая информацию о вашем бюджете. Если перед вами стоит задача представления ежегодного финансового отчета организации, для которой вы создаете сайт, то в качестве исходных данных заказчик, скорее всего, предоставил бы вам таблицу, аналогичную представленной на рис. 5.1.

	A	B	C	D	E	F
1		1999	2000	2001	2002	
2	Grants	11,980	12,650	9,700	10,600	
3	Donations	4,780	4,989	6,700	6,590	
4	Investments	8,000	8,100	8,760	8,490	
5	Fundraising	3,200	3,120	3,700	4,210	
6	Sales	28,400	27,100	27,950	29,050	
7	Miscellaneous	2,100	1,900	1,300	1,760	
8	Total	58,460	57,859	58,110	60,700	

*Рис. 5.1. Отображение данных о бюджете компании в виде табличных данных в формате Excel*

Вполне очевидно, что перед нами табличные данные. Мы видим заголовки столбцов и строк, которые определяют данные, находящиеся в каждой ячейке. Наша задача – представить эти данные в виде таблицы с заголовками, подтверждающими сохранение логической структуры исходной таблицы, как показано на рис. 5.2.

Yearly Income 1999 - 2002

	1999	2000	2001	2002
<b>Grants</b>	11,980	12,650	9,700	10,600
<b>Donations</b>	4,780	4,989	6,700	6,590
<b>Investments</b>	8,000	8,100	8,760	8,490
<b>Fundraising</b>	3,200	3,120	3,700	4,210
<b>Sales</b>	28,400	27,100	27,950	29,050
<b>Miscellaneous</b>	2,100	1,900	1,300	1,760
<b>Total</b>	58,460	57,859	58,110	60,700

*Рис. 5.2. Информация о бюджете компании в виде HTML-таблицы*

## Организация табличных данных: удобство доступа и наглядность

### Решение

В соответствии со спецификацией HTML таблицы могут содержать дополнительные элементы и атрибуты, предназначенные не для визуального оформления, а для обеспечения их корректной интерпретации программами вроде экранного диктора, которыми пользуются посетители со слабым зрением. Их применение не представляет никаких трудностей, однако многие веб-разработчики упускают их из виду. Рассмотрим следующий пример:

*chapter05/table.html (фрагмент)*

```
<table summary="This table shows the yearly income for years 1999
through 2002">
  <caption>Yearly Income 1999 - 2002</caption>
  <tr>
    <th></th>
    <th scope="col">1999</th>
    <th scope="col">2000</th>
    <th scope="col">2001</th>
    <th scope="col">2002</th>
  </tr>
  <tr>
    <th scope="row">Grants</th>
    <td>11,980</td>
    <td>12,650</td>
    <td>9,700</td>
    <td>10,600</td>
  </tr>
  <tr>
    <th scope="row">Donations</th>
    <td>4,780</td>
    <td>4,989</td>
    <td>6,700</td>
    <td>6,590</td>
  </tr>
  <tr>
    <th scope="row">Investments</th>
    <td>8,000</td>
    <td>8,100</td>
    <td>8,760</td>
    <td>8,490</td>
  </tr>
  <tr>
    <th scope="row">Fundraising</th>
    <td>3,200</td>
    <td>3,120</td>
    <td>3,700</td>
```

```
        <td>4,210</td>
    </tr>
    <tr>
        <th scope="row">Sales</th>
        <td>28,400</td>
        <td>27,100</td>
        <td>27,950</td>
        <td>29,050</td>
    </tr>
    <tr>
        <th scope="row">Miscellaneous</th>
        <td>2,100</td>
        <td>1,900</td>
        <td>1,300</td>
        <td>1,760</td>
    </tr>
    <tr>
        <th scope="row">Total</th>
        <td>58,460</td>
        <td>57,859</td>
        <td>58,110</td>
        <td>60,700</td>
    </tr>
</table>
```

## Обсуждение

В приведенном выше примере была использована таблица, содержащая элементы и атрибуты для четкого описания содержания каждой ячейки. Рассмотрим значение всех этих дополнительных элементов и атрибутов более подробно.

### Атрибут `summary` элемента `table`

*chapter05/table.html (фрагмент)*

```
<table summary="This table shows the yearly income for years 1999
through 2002">
```

Значение атрибута `summary` невидимо для пользователей браузеров, но оно будет прочтено экранным диктором. Его следует использовать, чтобы дать пользователю представление о содержании и назначении таблицы. Эта информация не является необходимой для пользователей обычных браузеров, поскольку они могут получить ее при просмотре страницы, однако она важна для пользователей с ограниченными возможностями.

### Элемент `caption`

*chapter05/table.html (фрагмент)*

```
<caption>Yearly Income 1999 - 2002</caption>
```

Элемент `caption` добавляет к таблице заголовок. По умолчанию он отображается сверху, но его расположение по отношению к таблице можно изменить с помощью CSS-свойства `caption-side`.

```
table {
  caption-side: bottom;
}
```

Зачем вообще использовать этот элемент, если можно просто разместить рядом с таблицей заголовок или текст абзаца? С помощью `caption` текст связывается с таблицей и далее интерпретируется в качестве ее заголовка – экранный диктор не сможет интерпретировать этот текст в качестве отдельного элемента. Вы хотите, чтобы заголовок отображался как обычный текст абзаца или заголовок третьего уровня в графическом браузере? Нет ничего проще – для этого достаточно всего лишь создать соответствующие правила стилей на CSS, как вы сделали бы это для любого другого элемента.

## Элемент `th`

```
<th scope="col">2000</th>
```

Элемент `th` предназначен для обозначения данных, используемых в качестве заголовка строки или столбца. В приведенном примере он применяется по отношению и к тем и другим; для более конкретного указания используется атрибут `scope` тега `<th>`. Он может принимать значение `col` (колонка, столбец) или `row` (ряд, строка) в зависимости от того, к чему относится.

Перед тем как начать оформление вида таблиц в соответствии с общим дизайном сайта, рекомендуется вначале обеспечить их доступность для пользователей таких устройств, как экранные дикторы. Доступность относится к тем вопросам, решение которых большинство разработчиков откладывают на потом, говоря: «Сначала разберусь с дизайном, а потом возьмусь за это». Однако если контроль за доступностью оставляют на конец всей разработки, то может случиться, что к нему и вовсе не возвращаются, а если и возвращаются, то для исправления возникающих проблем чаще всего требуется значительное количество времени, особенно в сложных приложениях. Если вы возьмете за правило продумывать эти аспекты непосредственно в процессе разработки сайта, то вскоре обнаружите, что это стало вашей второй натурой, лишь незначительно увеличив время работы над проектом.

CSS-атрибуты позволяют оформить таблицу быстро и просто. К примеру, в самом начале разработки сайта, в котором будет много таблиц, я создаю правило стиля с селектором класса `.datatable`, которое будет содержать основные описания стилей для всех таблиц и которое можно с легкостью добавить к тегу `<table>` каждого из них. Затем я создаю правило стиля для `.datatable th` (ячеек с заголовком), `.datatable td` (обычных ячеек) и `.datatable caption` (заголовков таблицы).

Такой подход значительно облегчает добавление новых таблиц. Поскольку все стили уже имеются, достаточно всего лишь применить по отношению к таблице класс `.datatable`. Если в дальнейшем потребуются изменить вид таблиц на сайте, достаточно отредактировать каскадную таблицу стилей.

## Создание рамки вокруг таблицы без использования HTML-атрибута border

С помощью HTML-атрибута можно создать лишь стандартную невыразительную рамку, поэтому использовать его не рекомендуется. Вместо этого можно воспользоваться средствами CSS – это дает гораздо большую свободу выбора визуального оформления. Следующий код задает рамку для таблицы:

*chapter05/table.css (фрагмент)*

```
.datatable {  
    border: 1px solid #338BA6;  
}
```

Применение данного правила стиля создает вокруг таблицы светло-голубую рамку толщиной в один пиксел, как показано на рис. 5.3.

Yearly Income 1999 - 2002				
	1999	2000	2001	2002
<b>Grants</b>	11,980	12,650	9,700	10,600
<b>Donations</b>	4,780	4,989	6,700	6,590
<b>Investments</b>	8,000	8,100	8,760	8,490
<b>Fundraising</b>	3,200	3,120	3,700	4,210
<b>Sales</b>	28,400	27,100	27,950	29,050
<b>Miscellaneous</b>	2,100	1,900	1,300	1,760
<b>Total</b>	58,460	57,859	58,110	60,700

*Рис. 5.3. Добавление рамки для таблицы в целом с помощью CSS*

Можно также определить рамки для индивидуальных ячеек:

*chapter05/table.css (фрагмент)*

```
.datatable td, .datatable th {  
    border: 1px solid #73C0D4;  
}
```

Применение данного правила стиля приводит к отображению чуть более светлой рамки вокруг ячеек `td` и `th` класса `datatable`, как показано на рис. 5.4.

Yearly Income 1999 - 2002				
	1999	2000	2001	2002
<b>Grants</b>	11,980	12,650	9,700	10,600
<b>Donations</b>	4,780	4,989	6,700	6,590
<b>Investments</b>	8,000	8,100	8,760	8,490
<b>Fundraising</b>	3,200	3,120	3,700	4,210
<b>Sales</b>	28,400	27,100	27,950	29,050
<b>Miscellaneous</b>	2,100	1,900	1,300	1,760
<b>Total</b>	58,460	57,859	58,110	60,700

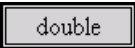

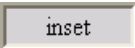
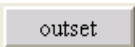

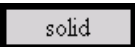

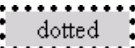
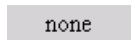
Рис. 5.4. Добавление рамки для отдельных ячеек с помощью CSS

## Обсуждение

Экспериментируя с различными значениями CSS-свойств, задающих параметры табличных рамок, можно добиться множества притягивающих внимание визуальных эффектов, пусть даже таблица содержит совершенно прозаические данные! Можно использовать рамки разных цветов для ячеек с заголовками и обычных ячеек, а также изменять толщину и вид рамки ячеек. При желании можно использовать для верхней и левой частей рамки один оттенок цвета, а для нижней и правой – другой.

Свойство `border-style` может принимать множество различных значений. Мы уже знакомы со значением `solid`, используемым для отображения сплошной рамки; это и другие значения и результат их обработки показан в табл. 5.1.

Таблица 5.1. Значения CSS-свойства `border-style`

 double	 groove	 inset
 outset	 ridge	 solid
 dashed	 dotted <sup>a</sup>	 none / hidden

<sup>a</sup> Internet Explorer 6 ведет себя немного странно: при задании значения `dotted` для рамки толщиной в один пиксел она будет отображаться как `dashed`.

## Удаление пустого пространства между ячейками, появляющегося после добавления рамок

Вы наверняка когда-нибудь пытались удалить пустое пространство между ячейками с помощью атрибута `cellspacing="0"` элемента `table`. Однако в результате получалась рамка толщиной в два пиксела, поскольку рамки соприкасаются, но не перекрывают друг друга. Ниже приводится метод создания аккуратной однопиксельной рамки вокруг ячеек.

### Решение

Пустое пространство между ячейками можно убрать путем присвоения значения `collapse` свойству `border-collapse`:

*chapter05/table.css*

```
.datatable {
    border: 1px solid #338BA6;
    border-collapse: collapse;
}

.datatable td, .datatable th {
    border: 1px solid #73C0D4;
}
```

На рис. 5.4. показана таблица до применения свойства `border-collapse`; на рис. 5.5 отображается эффект применения данного свойства.

Yearly Income 1999 - 2002				
	1999	2000	2001	2002
<b>Grants</b>	11,980	12,650	9,700	10,600
<b>Donations</b>	4,780	4,989	6,700	6,590
<b>Investments</b>	8,000	8,100	8,760	8,490
<b>Fundraising</b>	3,200	3,120	3,700	4,210
<b>Sales</b>	28,400	27,100	27,950	29,050
<b>Miscellaneous</b>	2,100	1,900	1,300	1,760
<b>Total</b>	58,460	57,859	58,110	60,700

*Рис. 5.5. Перекрытие рамок таблицы*

## Представление табличных данных в привлекательной и удобной форме

Для структурирования табличных данных лучше всего использовать HTML, несмотря на то что по умолчанию такие таблицы выглядят не



слишком привлекательно. К счастью, вид таблицы можно изменить с помощью CSS, что почти не увеличивает объем разметки и позволяет в дальнейшем управлять отображением табличных данных из таблицы стилей.

Yearly Income 1999 - 2002				
	1999	2000	2001	2002
<b>Grants</b>	11,980	12,650	9,700	10,600
<b>Donations</b>	4,780	4,989	6,700	6,590
<b>Investments</b>	8,000	8,100	8,760	8,490
<b>Fundraising</b>	3,200	3,120	3,700	4,210
<b>Sales</b>	28,400	27,100	27,950	29,050
<b>Miscellaneous</b>	2,100	1,900	1,300	1,760
<b>Total</b>	58,460	57,859	58,110	60,700

*Рис. 5.6. Неотформатированные табличные данные выглядят не слишком привлекательно*

Представленные ранее в настоящей главе с помощью HTML-таблицы данные могут послужить примером табличных данных. В основу следующего примера легла разметка, обработка которой в браузере показана на рис. 5.6.

Применим к этой таблице такую таблицу стилей:

*chapter05/spreadsheet.css*

```
body {
    font: 0.8em Verdana, Geneva, Arial, Helvetica, sans-serif;
}

.datatable {
    border: 1px solid #D6DDE6;
    border-collapse: collapse;
}

.datatable td, .datatable th {
    border: 1px solid #D6DDE6;
    text-align: right;
    padding: 0.2em;
}

.datatable th {
    border: 1px solid #828282;
    background-color: #BCBCBC;
    font-weight: bold;
    text-align: left;
    padding: 0.2em;
}
```

```
.datatable caption {
  font: bold 120% "Times New Roman", Times, serif;
  background-color: #B0C4DE;
  color: #33517A;
  padding: 0.4em 0 0.3em 0;
  border: 1px solid #789AC6;
}
```

На рис. 5.7 показан результат. Надо сказать, получилось вполне симпатично!

Yearly Income 1999 - 2002				
	1999	2000	2001	2002
<b>Grants</b>	11,980	12,650	9,700	10,600
<b>Donations</b>	4,780	4,989	6,700	6,590
<b>Investments</b>	8,000	8,100	8,760	8,490
<b>Fundraising</b>	3,200	3,120	3,700	4,210
<b>Sales</b>	28,400	27,100	27,950	29,050
<b>Miscellaneous</b>	2,100	1,900	1,300	1,760
<b>Total</b>	58,460	57,859	58,110	60,700

Рис. 5.7. Благодаря CSS таблица выглядит более привлекательно

## Обсуждение

В данном случае я поставила перед собой цель максимально приблизить вид таблицы к виду стандартной электронной таблицы Excel. Прежде всего было создано базовое правило для элемента `body`. Подобные правила, скорее всего, станут частью большинства каскадных таблиц стилей для сайтов, создаваемых на основе CSS:

*chapter05/spreadsheet.css (фрагмент)*

```
body {
  font: 0.8em Verdana, Geneva, Arial, Helvetica, sans-serif;
}
```

Затем зададим стиль для таблицы в целом:

*chapter05/spreadsheet.css (фрагмент)*

```
.datatable {
  border: 1px solid #D6DDE6;
  border-collapse: collapse;
}
```

Как мы уже видели, свойство `border` позволяет задать параметры рамки вокруг таблицы, а свойство `border-collapse` удаляет пустое пространство между ее ячейками.

Затем обратимся к ячейкам таблицы:

*chapter05/spreadsheet.css (фрагмент)*

```
.datatable td {
  border: 1px solid #D6DDE6;
  text-align: right;
  padding: 0.2em
}
```

Здесь я добавила рамку для ячеек таблицы и выровняла их содержимое по правой стороне с помощью свойства `text-align`, чтобы она больше походила на *электронную таблицу*. На данный момент при просмотре документа вы увидите рамку вокруг всех ячеек, кроме тех, что содержат заголовки, как показано на рис. 5.8.

Yearly Income 1999 - 2002				
	1999	2000	2001	2002
<b>Grants</b>	11,980	12,650	9,700	10,600
<b>Donations</b>	4,780	4,989	6,700	6,590
<b>Investments</b>	8,000	8,100	8,760	8,490
<b>Fundraising</b>	3,200	3,120	3,700	4,210
<b>Sales</b>	28,400	27,100	27,950	29,050
<b>Miscellaneous</b>	2,100	1,900	1,300	1,760
<b>Total</b>	58,460	57,859	58,110	60,700

Рис. 5.8. Применение свойства `border` по отношению к элементам `table` и `td`

Затем была добавлена рамка и цвет фона для ячеек с заголовками. Я использовала более темный цвет для рамки, поскольку их фон также темнее фона обычных ячеек, что позволяет выделить заголовки:

*chapter05/spreadsheet.css (фрагмент)*

```
.datatable th {
  border: 1px solid #828282;
  background-color: #B0B0B0;
  font-weight: bold;
  text-align: left;
  padding: 0.2em;
}
```

В завершение зададим стиль для элемента `caption`, чтобы он выглядел частью таблицы.

*chapter05/spreadsheet.css (фрагмент)*

```
.datatable caption {
  font: bold 0.9em "Times New Roman", Times, serif;
  background-color: #B0C4DE;
```

```
color: #33517A;
padding: 0.4em 0 0.3em 0;
border: 1px solid #789AC6;
}
```

## Чередование фонового цвета строк таблицы

### Решение

При просмотре таблицы с большим количеством данных бывает трудно удержать взгляд на конкретной строке. Чтобы помочь пользователям сориентироваться, можно использовать два различных фоновых цвета поочередно. Неважно, добавляете ли вы ряды таблицы вручную или же отображаете данные из базы данных, — для создания описанного эффекта можно использовать классы CSS.

Ниже представлена необходимая разметка:

*chapter05/alternate.html (фрагмент)*

```
<table summary="List of new students 2003" class="datatable">
  <caption>Student List</caption>
  <tr>
    <th scope="col">Student Name</th>
    <th scope="col">Date of Birth</th>
    <th scope="col">Class</th>
    <th scope="col">ID</th>
  </tr>
  <tr>
    <td>Joe Bloggs</td>
    <td>27/08/1997</td>
    <td>Mrs Jones</td>
    <td>12009</td>
  </tr>
  <tr class="altrow">
    <td>William Smith</td>
    <td>20/07/1997</td>
    <td>Mrs Jones</td>
    <td>12010</td>
  </tr>
  <tr>
    <td>Jane Toad</td>
    <td>21/07/1997</td>
    <td>Mrs Jones</td>
    <td>12030</td>
  </tr>
  <tr class="altrow">
    <td>Amanda Williams</td>
    <td>19/03/1997</td>
    <td>Mrs Edwards</td>
    <td>12021</td>
  </tr>
```

```

</tr>
<tr>
  <td>Kylie Jameson</td>
  <td>18/05/1997</td>
  <td>Mrs Jones</td>
  <td>12022</td>
</tr>
<tr class="altrow">
  <td>Louise Smith</td>
  <td>17/07/1997</td>
  <td>Mrs Edwards</td>
  <td>12019</td>
</tr>
<tr>
  <td>James Jones</td>
  <td>04/04/1997</td>
  <td>Mrs Edwards</td>
  <td>12007</td>
</tr>
</table>

```

**А ниже представлена связанная с ним таблица стилей:**

*chapter05/alternate.css (фрагмент)*

```

body {
  font: 0.8em Arial, Helvetica, sans-serif;
}
.datatable {
  border: 1px solid #D6DDE6;
  border-collapse: collapse;
  width: 80%;
}
.datatable td {
  border: 1px solid #D6DDE6;
  padding: 0.3em;
}
.datatable th {
  border: 1px solid #828282;
  background-color: #BCBCBC;
  font-weight: bold;
  text-align: left;
  padding-left: 0.3em;
}
.datatable caption {
  font: bold 110% Arial, Helvetica, sans-serif;
  color: #33517A;
  text-align: left;
  padding: 0.4em 0 0.8em 0;
}

```

```
.datatable tr.altrow {
    background-color: #DFE7F2;
    color: #000000;
}
```

Результат обработки данного кода показан на рис. 5.9.

### Student List

Student Name	Date of Birth	Class	ID
Joe Bloggs	27/08/1997	Mrs Jones	12009
William Smith	20/07/1997	Mrs Jones	12010
Jane Toad	21/07/1997	Mrs Jones	12030
Amanda Williams	19/03/1997	Mrs Edwards	12021
Kylie Jameson	18/05/1997	Mrs Jones	12022
Louise Smith	17/07/1997	Mrs Edwards	12019
James Jones	04/04/1997	Mrs Edwards	12007

Рис. 5.9. Чередование различного цвета фона для облегчения ориентирования в больших таблицах

## Обсуждение

Каждая вторая строка в приведенной выше таблице относится к классу `altrow`:

*chapter05/alternate.html (фрагмент)*

```
<tr class="altrow">
```

Если вы прочли предыдущие решения в этой главе, то использованные в CSS-коде свойства должны быть вам хорошо знакомы. Кроме того, я добавила следующий класс:

*chapter05/alternate.css (фрагмент)*

```
.datatable tr.altrow {
    background-color: #DFE7F2;
    color: #000000;
}
```

Эти стили будут применены ко всем элементам `tr` класса `altrow`, находящимся внутри таблицы класса `datatable`.

Если вы формируете таблицу динамически с помощью ASP, PHP или иной подобной технологии, извлекающей данные из базы данных, то для чередования фонового цвета придется приписывать данный класс каждому второму отображаемому ряду.

## Взгляд в будущее

Добавлять имя класса для каждого второго ряда довольно утомительно, и вы не ошиблись, если задумались о наличии альтернативного решения той же самой задачи. Эффекта чередования можно достичь исключительно средствами CSS. Надо отметить, что этот метод гораздо эффективнее. Воспользовавшись селектором псевдокласса CSS3 `:nth-child`, можно выбрать все четные или нечетные ряды таблицы, ничего не меняя в HTML-разметке. Наступят времена, когда все браузеры будут поддерживать этот псевдокласс, и жизнь разработчиков на CSS станет гораздо проще. Понять механизм функционирования этого псевдокласса достаточно сложно, поэтому рекомендую вам обратиться к справочной документации CSS Reference, в которой он прекрасно представлен.<sup>1</sup> С помощью данного селектора псевдокласса можно выбирать элементы в зависимости от количества предшествующих ему элементов того же уровня.

Вернемся к изначальному коду таблицы до добавления имени класса для рядов. Можно обратиться ко всем четным рядам таблицы с помощью следующего CSS-кода:

```
.datatable tr:nth-child(2n) {
  background-color: #DFE7F2;
  color: #000000;
}
```

Для обращения ко всем нечетным рядам можно использовать следующий CSS-код:

```
.datatable tr:nth-child(2n+1) {
  background-color: #DFE7F2;
  color: #000000;
}
```

Содержащиеся в данном селекторе числовые выражения могут показаться замысловатыми, но, к счастью, разработчикам настолько часто приходится выполнять подобную задачу, что в спецификации CSS3 был описан более простой синтаксис. Вместо числовых выражений в скобках можно использовать ключевые слова:

```
.datatable tr:nth-child(even) {
  :
}
.datatable tr:nth-child(odd) {
  :
}
```

На момент написания данной книги эти примеры обрабатывались последними версиями Opera и Safari, но, пока данный псевдокласс не по-

---

<sup>1</sup> <http://reference.sitepoint.com/css/understandingnthchildexpressions/>

лучит широкую поддержку во многих браузерах, использовать такой прием рано.

### Совет

**Обеспечение поддержки с помощью JavaScript.** JavaScript-библиотека jQuery поддерживает селекторы CSS3,<sup>1</sup> поэтому ее можно использовать для чередования фонового цвета ячеек таблицы. Для создания описанного эффекта, выполняющего эстетическую функцию, можно использовать JavaScript; при этом не нужно добавлять имена классов элементам разметки.

## Изменение фонового цвета строки при наведении на нее указателя мыши

### Решение

Чтобы таблицу было легче воспринимать, можно изменять цвет строки при наведении на нее указателя мыши – это выделит содержимое. Такой эффект показан на рис. 5.10.

#### Student List

Student Name	Date of Birth	Class	ID
Joe Bloggs	27/08/1997	Mrs Jones	12009
William Smith	20/07/1997	Mrs Jones	12010
Jane Toad	21/07/1997	Mrs Jones	12030
Amanda Williams	19/03/1997	Mrs Edwards	12021
Kylie Jameson	18/05/1997	Mrs Jones	12022
Louise Smith	17/07/1997	Mrs Edwards	12019
James Jones	04/04/1997	Mrs Edwards	12007

Рис. 5.10. Выделение строки при наведении на нее указателя мыши

Реализовать его достаточно просто, для этого достаточно добавить в таблицу стилей следующее правило:

*chapter05/alternate.css (фрагмент)*

```
.datatable tr:hover {
    background-color: #DFE7F2;
    color: #000000;
}
```

Готово!

<sup>1</sup> <http://docs.jquery.com/Selectors>



## Обсуждение

Такой документ будет корректно отображаться во всех современных браузерах, в том числе и Internet Explorer 7, но за исключением Internet Explorer 6. Однако если данные в таблице представлены логично и все понятно и без фонового выделения, то такой эффект можно считать приятным дополнением, но не обязательным – без которого нельзя было бы пользоваться сайтом.

Если по каким-либо причинам вам *непрерывно нужно* реализовать данную функцию для пользователей Internet Explorer 6, можно добавить простой сценарий на JavaScript. Для изменения фонового цвета ряда таблицы при наведении на него указателя мыши в Internet Explorer 6 нужно задать необходимые свойства для CSS-класса (в этом примере я назову его hilite).

*chapter05/hiliterow.css (фрагмент)*

```
.datatable tr:hover, .datatable tr.hilite {
    background-color: #DFE7F2;
    color: #000000;
}
```

После таблицы добавьте на страницу следующий код на JavaScript:

*chapter05/hiliterow.html (фрагмент)*

```
<script type="text/javascript">
var rows = document.getElementsByTagName('tr');
for (var i = 0; i < rows.length; i++) {
    rows[i].onmouseover = function() {
        this.className += ' hilite';
    }
    rows[i].onmouseout = function() {
        this.className = this.className.replace('hilite', '');
    }
}
</script>
```

В данном сценарии осуществляется поиск всех тегов <tr> в документе и присвоение каждому из них обработчика событий `mouseover` и `mouseout`. Они присваивают рядом CSS-класс `hilite` при наведении на него указателя мыши и удаляют его при его смещении в другую область. Как видно на рис. 5.11, такое сочетание CSS и JavaScript приводит к желаемому результату.

Рассматриваемый сценарий на JavaScript динамически задает CSS-класс для тега. В данном случае мы добавляем класс `hilite` в тег <tr> по событию `mouseover`, для чего используется свойство `onmouseover`:

*chapter05/hiliterow.html (фрагмент)*

```
rows[i].onmouseover = function() {
    this.className += ' hilite';
}
```

**Student List**

Student Name	Date of Birth	Class	ID
Joe Bloggs	27/08/1997	Mrs Jones	12009
William Smith	20/07/1997	Mrs Jones	12010
Jane Toad	21/07/1997	Mrs Jones	12030
Amanda Williams	19/03/1997	Mrs Edwards	12021
Kylie Jameson	18/05/1997	Mrs Jones	12022
Louise Smith	17/07/1997	Mrs Edwards	12019
James Jones	04/04/1997	Mrs Edwards	12007

*Рис. 5.11. Выделение ряда в Internet Explorer 6 с помощью JavaScript*

После наступления события `mouseout` мы удаляем класс:

*chapter05/hiliterow.html (фрагмент)*

```
rows[i].onmouseout = function() {
    this.className = this.className.replace('hilite', '');
}
```

Изменяя имя класса элементов в ответ на действия пользователя с помощью JavaScript, можно добиться разнообразных привлекательных и ненавязчивых эффектов. Этот прием можно применить, например, для выделения определенной области путем изменения имени класса соответствующего элемента `div` по событию `mouseover`.

**Примечание**

**Ненавязчивый JavaScript.** Вы наверняка заметили, что код на JavaScript был добавлен не к самой таблице; весь сценарий был размещен только внутри элемента `script`. Такой прием называется **ненавязчивым JavaScript**; его смысл состоит в разделении документа и сценариев, подобно тому, как мы размещаем отдельно разметку и каскадную таблицу стилей.

Сценарий на JavaScript должен быть запущен после загрузки таблицы, поскольку до этого отсутствует сам объект, по отношению к которому сценарий должен выполняться. В качестве варианта можно написать функцию, которая будет выполняться по окончании загрузки страницы; в этом случае код на JavaScript можно хранить в отдельном файле, указав ссылку на него в коде страницы. Кроме того, чтобы сценарий загружался только при открытии страницы в браузере Internet Explorer 6, можно использовать условные комментарии. Подробнее об этом мы поговорим в разделе «Определение различных стилей для Internet Explorer 6 и 7» главы 7.

Как и в предыдущем примере, в данном случае на помощь может прийти использование различных библиотек, таких как `jQuery`, которые как нельзя кстати, когда поддержка тех или иных свойств в старых браузерах отсутствует.

## Чередование фонового цвета столбцов таблицы

Чередование фонового цвета строк встречается гораздо чаще, чем чередование цвета столбцов. Однако этот прием достаточно эффективен для выделения группы данных.

### Решение

Для описания колонок таблицы мы воспользуемся элементом `col`. При этом можно задать для них фоновый цвет с помощью CSS. В следующей разметке видны добавленные для каждого столбца элементы `col`, каждому из которых присвоено имя класса (как и в примере с изменением фонового цвета строк в разделе «Чередование фонового цвета строк таблицы»).

*chapter05/columns.html (фрагмент)*

```
<table class="datatable">
  <col class="odd" />
  <col class="even" />
  <col class="odd" />
  <col class="even" />

  <tr>
    <th>Pool A</th>
    <th>Pool B</th>
    <th>Pool C</th>
    <th>Pool D</th>
  </tr>
  <tr>
    <td>England</td>
    <td>Australia</td>
    <td>New Zealand</td>
    <td>France</td>
  </tr>
  <tr class="even">
    <td>South Africa</td>
    <td>Wales</td>
    <td>Scotland</td>
    <td>Ireland</td>
  </tr>
  <tr>
    <td>Samoa</td>
    <td>Fiji</td>
    <td>Italy</td>
    <td>Argentina</td>
  </tr>
  <tr class="even">
    <td>USA</td>
```

```
<td>Canada</td>
<td>Romania</td>
<td>Europe 3</td>
</tr>
<tr>
<td>Repechage 2</td>
<td>Asia</td>
<td>Repechage 1</td>
<td>Namibia</td>
</tr>
</table>
```

**Классам, присвоенным элементам col, можно задать определенные правила стилей, как показано ниже; результат обработки данного кода показан на рис. 5.12.**

*chapter05/columns.css (фрагмент)*

```
body {
  font: 0.8em Arial, Helvetica, sans-serif;
}
.datatable {
  border: 1px solid #D6DDE6;
  border-collapse: collapse;
  width: 80%;
}
.datatable col.odd {
  background-color: #80C9FF;
  color: #000000;
}
.datatable col.even {
  background-color: #BFE4FF;
  color: #000000;
}
.datatable td {
  border: 2px solid #ffffff;
  padding: 0.3em;
}
.datatable th {
  border: 2px solid #ffffff;
  background-color: #00487D;
  color: #FFFFFF;
  font-weight: bold;
  text-align: left;
  padding: 0.3em;
}
```

Pool A	Pool B	Pool C	Pool D
England	Australia	New Zealand	France
South Africa	Wales	Scotland	Ireland
Samoa	Fiji	Italy	Argentina
USA	Canada	Romania	Europe 3
Repechage 2	Asia	Repechage 1	Namibia

*Рис. 5.12. Чередование фонового цвета столбцов путем задания стилей для элемента col*

## Обсуждение

С помощью элемента `col` можно более гибко задавать стили для столбцов таблицы, чтобы они отличались визуально, и пользователю было удобнее воспринимать содержащиеся в них данные. Если элементы `col` являются дочерними по отношению к элементу `colgroup`, то для управления внешним видом столбцов можно применить правила стилей по отношению к нему. При отсутствии элемента `colgroup` браузер считает, что в таблице только одна группа столбцов, содержащая все элементы `col`.

Ниже представлен пример вложенных элементов `col`:

*chapter05/colgroups.html (фрагмент)*

```
<table class="datatable">
  <colgroup class="odd">
    <col />
    <col />
  </colgroup>
  <colgroup class="even">
    <col />
    <col />
  </colgroup>
  ...
```

Ниже представлены правила стилей, применяемые по отношению к элементу `colgroup`, а не `col`:

*chapter05/colgroups.css (фрагмент)*

```
.datatable colgroup.odd {
  background-color: #80C9FF;
  color: #000000;
}
.datatable colgroup.even {
  background-color: #BFE4FF;
  color: #000000;
}
```

В результате получим таблицу с двумя колонками одного цвета и двумя колонками другого цвета, что показано на рис. 5.13.

Pool A	Pool B	Pool C	Pool D
England	Australia	New Zealand	France
South Africa	Wales	Scotland	Ireland
Samoa	Fiji	Italy	Argentina
USA	Canada	Romania	Europe 3
Repechage 2	Asia	Repechage 1	Namibia

Рис. 5.13. Задание вида таблицы при использовании `colgroup`

## Создание календаря с помощью CSS

Календари также содержат табличные данные. Пример календаря, взятый из настольного приложения, показан на рис. 5.14. В качестве заголовков столбцов выступают расположенные сверху названия дней недели. Поэтому для создания календарей целесообразно использовать таблицы; при этом объем HTML-кода можно свести к минимуму, применив CSS для задания внешнего вида календаря.

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	1	2	3	4
5	6	7	8	9	10	11

Рис. 5.14. Календарь из настольного приложения

## Решение

Решение основано на использовании простой и доступной таблицы, которая средствами CSS преобразуется в привлекательный календарь, изображенный на рис. 5.15. Учитывая его несложную структуру, он идеально подходит для приложения, связанного с базой данных, в котором таблица создается кодом на стороне сервера:

### *chapter05/cal.html (фрагмент)*

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
<head>
<title>Calendar</title>
<meta http-equiv="content-type"
  content="text/html; charset=utf-8" />
<link rel="stylesheet" type="text/css" href="cal.css" />
</head>
<body>
<table class="clmonth" summary="Calendar for June 2009">
<caption>June 2009</caption>
<tr>
  <th scope="col">Monday</th>
  <th scope="col">Tuesday</th>
  <th scope="col">Wednesday</th>
  <th scope="col">Thursday</th>
  <th scope="col">Friday</th>
  <th scope="col">Saturday</th>
  <th scope="col">Sunday</th>
</tr>
<tr>
  <td class="previous">31</td>
  <td class="active">1
    <ul>
      <li>New pupils' open day</li>
      <li>Year 8 theater trip</li>
    </ul>
</td>
  <td>2</td>
  <td>3</td>
  <td>4</td>
  <td>5</td>
  <td>6</td>
</tr>
<tr>
  <td class="active">7
    <ul>
      <li>Year 7 English exam</li>
    </ul>
</td>
  <td>8</td>

```

```
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>14</td>
<td>15</td>
<td>16</td>
<td class="active">17
  <ul>
    <li>Sports Day</li>
  </ul></td>
<td class="active">18
  <ul>
    <li>Year 7 parents' evening</li>
    <li>Prizegiving</li>
  </ul></td>
<td>19</td>
<td>20</td>
</tr>
<tr>
<td>21</td>
<td>22</td>
<td>23</td>
<td class="active">24
  <ul>
    <li>Year 8 parents' evening</li>
  </ul></td>
<td>25</td>
<td>26</td>
<td>27</td>
</tr>
<tr>
<td>28</td>
<td>29</td>
<td class="active">30
  <ul>
    <li>First night of school play</li>
  </ul></td>
<td class="next">1</td>
<td class="next">2</td>
<td class="next">3</td>
<td class="next">4</td>
</tr>
</table>
</body>
</html>
```



*chapter05/cal.css*

```
body {
    background-color: #FFFFFF;
    color: #000000;
    font-size: 90%;
}
.clmonth {
    border-collapse: collapse;
    width: 780px;
}
.clmonth caption {
    text-align: left;
    font: bold 110% Georgia, "Times New Roman", Times, serif;
    padding-bottom: 0.4em;
}
.clmonth th {
    border: 1px solid #AAAAAA;
    border-bottom: none;
    padding: 0.2em 0.6em 0.2em 0.6em;
    background-color: #CCCCCC;
    color: #3F3F3F;
    font: 80% Verdana, Geneva, Arial, Helvetica, sans-serif;
    width: 110px;
}
.clmonth td {
    border: 1px solid #EAEAEA;
    font: 80% Verdana, Geneva, Arial, Helvetica, sans-serif;
    padding: 0.2em 0.6em 0.2em 0.6em;
    vertical-align: top;
}
.clmonth td.previous, .clmonth td.next {
    background-color: #F6F6F6;
    color: #C6C6C6;
}
.clmonth td.active {
    background-color: #B1CBE1;
    color: #2B5070;
    border: 2px solid #4682B4;
}
.clmonth ul {
    list-style-type: none;
    margin: 0;
    padding-left: 1em;
    padding-right: 0.6em;
}
.clmonth li {
    margin-bottom: 1em;
}
```

June 2009						
Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
31	1 New pupils' open day Year 8 theater trip	2	3	4	5	6
7 Year 7 English exam	8	9	10	11	12	13
14	15	16	17 Sports Day	18 Year 7 parents' evening Prizegiving	19	20
21	22	23	24 Year 8 parents' evening	25	26	27
28	29	30 First night of school play	1	2	3	4

Рис. 5.15. Окончательный вид календаря, оформленного с помощью CSS

## Обсуждение

Решение поставленной задачи начнем с создания самой простой таблицы. У нее есть заголовок (текущий месяц), а названия дней недели выступают в качестве заголовков колонок, определенных с помощью тега `<th>`:

*chapter05/cal.html (фрагмент)*

```
<table class="clmonth" summary="Calendar for June 2009">
  <caption>June 2009</caption>
  <tr>
    <th scope="col">Monday</th>
    <th scope="col">Tuesday</th>
    <th scope="col">Wednesday</th>
    <th scope="col">Thursday</th>
    <th scope="col">Friday</th>
    <th scope="col">Saturday</th>
    <th scope="col">Sunday</th>
  </tr>
```

Таблице присвоено имя класса `clmonth`. Использование класса вместо ID объясняется тем, что в некоторых случаях может возникнуть необходимость отображения сразу нескольких месяцев на одной странице. Если потом выяснится, что таблице необходимо присвоить ID – например, для создания функции скрытия и отображения страницы с помощью JavaScript, – то можно кроме класса добавить и идентификатор.

Числа содержатся в отдельных ячейках, а мероприятия представлены в виде списка в соответствующей ячейке.

В приведенной ниже разметке видно, что двум ячейкам таблицы добавлены классы. Класс `previous` отмечает ячейки, содержащие дни, которые относятся к предыдущему месяцу (для выделения дней, относящихся к следующему месяцу, будет использоваться класс `next`); ячейкам, содержащим информацию о событиях, присваивается класс `active`:

*chapter05/cal.html (фрагмент)*

```
<tr>
  <td class="previous">31</td>
  <td class="active">1
    <ul>
      <li>New pupils' open day</li>
      <li>Year 8 theater trip</li>
    </ul></td>
  <td>2</td>
  <td>3</td>
  <td>4</td>
  <td>5</td>
  <td>6</td>
</tr>
```

Вид таблицы без использования CSS представлен на рис. 5.16.

June 2009						
Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
	1					
31	<ul style="list-style-type: none"> <li>New pupils' open day</li> <li>Year 8 theater trip</li> </ul>	2	3	4	5	6
7						
<ul style="list-style-type: none"> <li>Year 7 English exam</li> </ul>	8	9	10	11	12	13
				17	18	
14	15	16	<ul style="list-style-type: none"> <li>Sports Day</li> </ul>	<ul style="list-style-type: none"> <li>Year 7 parents' evening</li> <li>Prizegiving</li> </ul>	19	20
			24			
21	22	23	<ul style="list-style-type: none"> <li>Year 8 parents' evening</li> </ul>	25	26	27
		30				
28	29	<ul style="list-style-type: none"> <li>First night of school play</li> </ul>	1	2	3	4

*Рис. 5.16. Вид календаря без использования CSS*

Итак, разметка готова, и можно приступить к визуальному оформлению. Зададим основные стили для элемента `body`, в том числе и параметры шрифта. Затем зададим стиль для класса `clmonth`, чтобы рамки перекрывали друг друга и между ячейками не оставалось пустого пространства, а также укажем ширину таблицы:

*chapter05/cal.css (фрагмент)*

```
body {
  background-color: #FFFFFF;
  color: #000000;
  font-size: 90%;
}
.clmonth {
  border-collapse: collapse;
  width: 780px;
}
```

Мы указали стили для элемента `caption`, являющегося дочерним по отношению к элементу класса `clmonth`, а затем для заголовков (`th`) и ячеек (`td`) таблицы:

*chapter05/cal.css (фрагмент)*

```
.clmonth caption {
  text-align: left;
  font: bold 110% Georgia, "Times New Roman", Times, serif;
  padding-bottom: 0.4em;
}
.clmonth th {
  border: 1px solid #AAAAAA;
  border-bottom: none;
  padding: 0.2em 0.6em 0.2em 0.6em;
  background-color: #CCCCCC;
  color: #3F3F3F;
  font: 80% Verdana, Geneva, Arial, Helvetica, sans-serif;
  width: 110px;
}
.clmonth td {
  border: 1px solid #EAEAEA;
  font: 80% Verdana, Geneva, Arial, Helvetica, sans-serif;
  padding: 0.2em 0.6em 0.2em 0.6em;
  vertical-align: top;
}
```

Как видно на рис. 5.17, наш календарь начинает приобретать определенные визуальные черты.

Теперь можно задать оформление для списков событий, расположенных внутри ячеек, удалив маркеры и добавив больше пустого пространства между пунктами списка:

*chapter05/cal.css (фрагмент)*

```
.clmonth ul {
  list-style-type: none;
  margin: 0;
  padding-left: 1em;
  padding-right: 0.6em;
}
```

June 2009						
Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
31	1 <ul style="list-style-type: none"> <li>• New pupils' open day</li> <li>• Year 8 theater trip</li> </ul>	2	3	4	5	6
7 <ul style="list-style-type: none"> <li>• Year 7 English exam</li> </ul>	8	9	10	11	12	13
14	15	16	17 <ul style="list-style-type: none"> <li>• Sports Day</li> </ul>	18 <ul style="list-style-type: none"> <li>• Year 7 parents' evening</li> <li>• Prizegiving</li> </ul>	19	20
21	22	23	24 <ul style="list-style-type: none"> <li>• Year 8 parents' evening</li> </ul>	25	26	27
28	29	30 <ul style="list-style-type: none"> <li>• First night of school play</li> </ul>	1	2	3	4

**Рис. 5.17.** Благодаря новому оформлению элементов *caption*, *th* и *td* календарь выглядит привлекательнее

```
.clmonth li {
    margin-bottom: 1em;
}
```

Наконец, добавим стили классов `previous` и `next`, чтобы не относящиеся к текущему месяцу дни отображались в оттенках серого. Кроме того, зададим стиль для класса `active`, используемого для выделения дней с событиями:

*chapter05/cal.css (фрагмент)*

```
.clmonth td.previous, .clmonth td.next {
    background-color: #F6F6F6;
    color: #C6C6C6;
}
.clmonth td.active {
    background-color: #B1CBE1;
    color: #2B5070;
    border: 2px solid #4682B4;
}
```

Мы рассмотрели лишь один из немногих способов создания календарей. Календари часто используют в блогах: при щелчке по числу можно перейти к сделанной в этот день записи. Если убрать из нашего HTML-кода все события, оставить только первые буквы названий дней недели

и слегка изменить **CSS-код**, мы получим простой миниатюрный календарь, который вполне можно использовать в этих целях (рис. 5.18).

M	T	W	T	F	S	S
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	1	2	3	4

*Рис. 5.18. Создание миниатюрного календаря*

Ниже представлены HTML- и CSS-коды этой версии календаря:  
*chapter05/cal\_mini.html (фрагмент)*

```
<table class="clmonth" summary="Calendar for June 2009">
  <caption>June 2009</caption>
  <tr>
    <th scope="col">M</th>
    <th scope="col">T</th>
    <th scope="col">W</th>
    <th scope="col">T</th>
    <th scope="col">F</th>
    <th scope="col">S</th>
    <th scope="col">S</th>
  </tr>
  <tr>
    <td class="previous">31</td>
    <td class="active">1</td>
    <td>2</td>
    <td>3</td>
    <td>4</td>
    <td>5</td>
    <td>6</td>
  </tr>
  <tr>
    <td class="active">7</td>
    <td>8</td>
    <td>9</td>
    <td>10</td>
    <td>11</td>
    <td>12</td>
    <td>13</td>
  </tr>
  <tr>
    <td>14</td>
    <td>15</td>
    <td>16</td>
    <td class="active">17</td>
    <td class="active">18</td>
    <td>19</td>
    <td>20</td>
  </tr>
  <tr>
    <td>21</td>
    <td>22</td>
    <td>23</td>
    <td class="active">24</td>
    <td>25</td>
    <td>26</td>
    <td>27</td>
  </tr>
  <tr>
    <td>28</td>
    <td>29</td>
    <td class="active">30</td>
    <td>1</td>
    <td>2</td>
    <td>3</td>
    <td>4</td>
  </tr>
</table>
```

```

        <td>15</td>
        <td>16</td>
        <td class="active">17</td>
        <td class="active">18</td>
        <td>19</td>
        <td>20</td>
    </tr>
    <tr>
        <td>21</td>
        <td>22</td>
        <td>23</td>
        <td class="active">24</td>
        <td>25</td>
        <td>26</td>
        <td>27</td>
    </tr>
    <tr>
        <td>28</td>
        <td>29</td>
        <td class="active">30</td>
        <td class="next">1</td>
        <td class="next">2</td>
        <td class="next">3</td>
        <td class="next">4</td>
    </tr>
</table>

```

### *chapter05/cal\_mini.css*

```

body {
    background-color: #FFFFFF;
    color: #000000;
    font-size: 90%;
}
.clmonth {
    border-collapse: collapse;
}
.clmonth caption {
    text-align: left;
    font: bold 110% Georgia, "Times New Roman", Times, serif;
    padding-bottom: 0.4em;
}
.clmonth th {
    border: 1px solid #AAAAAA;
    border-bottom: none;
    padding: 0.2em 0.4em 0.2em 0.4em;
    background-color: #CCCCCC;
    color: #3F3F3F;
    font: 80% Verdana, Geneva, Arial, Helvetica, sans-serif;
}
.clmonth td {
    border: 1px solid #EAEAEA;
}

```

```
font: 80% Verdana, Geneva, Arial, Helvetica, sans-serif;
padding: 0.2em 0.4em 0.2em 0.4em;
vertical-align: top;
}
.clmonth td.previous, .clmonth td.next {
background-color: #F6F6F6;
color: #C6C6C6;
}
.clmonth td.active {
background-color: #B1CBE1;
color: #2B5070;
border: 2px solid #4682B4;
}
```

## Заклучение

В данной главе мы увидели, что таблицы живы-здоровы и еще не раз сослужат нам хорошую службу, если их правильно использовать – для представления табличных данных, конечно! С помощью CSS можно превратить обычные таблицы в весьма привлекательные элементы интерфейса, доступные для альтернативных устройств. Поэтому примите таблицы такими, какие они есть, и пусть они выполняют свою привычную работу по представлению табличных данных!



# 6

## Формы и пользовательские интерфейсы

Формы являются неотъемлемой частью практически любого сайта. Они используются для передачи персональных данных от пользователей, для написания сообщений на форумах, добавления товаров в корзину, обновления блогов – и это только начало перечня!

Формы получили широкое распространение в веб-среде, однако, несмотря на это, **HTML**, по сути, не дает разработчику никаких возможностей выбора визуального оформления, поэтому формы, как правило, отображаются в соответствии с внутренней таблицей стилей браузера. Появление **CSS** принесло с собой множество способов адресации к элементам форм, и в данной главе мы увидим, для каких из них можно задать собственные стили и зачем это может понадобиться. Кроме того, мы рассмотрим несколько нечасто используемых **HTML-тегов** и **атрибутов** форм, которые позволят повысить доступность и юзабилити наших форм; они также могут послужить дополнительными элементами, по отношению к которым можно применять **CSS**-стили.

На протяжении следующих нескольких страниц мы рассмотрим различные методы создания форм – как с помощью таблиц, так и путем позиционирования элементов средствами **CSS**. По поводу целесообразности разметки форм с помощью таблицы ведутся оживленные дискуссии; моя точка зрения по этому вопросу состоит в следующем: если форма по сути представляет собой таблицу (как в примере с электронной таблицей, который вы найдете в этой главе), то ее структуру вполне естественно было бы отобразить с помощью таблицы. В противном случае ваша форма, скорее всего, будет более доступна для всех категорий пользователей и для альтернативных устройств, если она создана средствами **CSS**.

При работе с формами крайне важно помнить о проблемах юзабилити. Формы предназначены для того, чтобы пользователь мог ввести с их

помощью какие-либо данные, но если ему непонятно, как ими пользоваться, они будут совершенно бесполезными, пусть и очень красивыми. В большинстве случаев я советую не увлекаться чрезмерным украшательством форм, поскольку это может привести посетителей сайта в замешательство. Кроме того, учтите, что различные браузеры обладают разными возможностями управления внешним видом элементов форм, поэтому обязательно тестируйте написанный CSS-код в как можно большем количестве браузеров под различными платформами.

## Изменение вида элементов формы с помощью CSS

Без применения каскадных таблиц стилей элементы формы будут отображены в соответствии с используемым браузером и операционной системой настройками по умолчанию. Однако использование CSS позволяет создать формы, соответствующие дизайну вашего сайта.

### Решение

Стиль элемента формы, как и любого другого HTML-элемента, можно задать с помощью CSS.

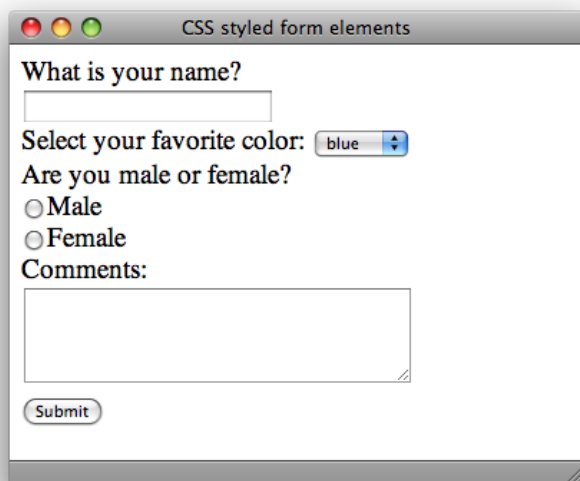


Рис. 6.1. Вид формы по умолчанию в браузере Safari

Изображенная на рис. 6.1 форма оформлена в соответствии с настройками по умолчанию, используемыми браузером Safari под Mac OS X. Ее вид будет изменяться при просмотре в различных браузерах под разными платформами. Ниже представлена типичная форма:

*chapter06/elements.html (фрагмент)*

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
<head>
  <title>CSS styled form elements</title>
  <meta http-equiv="content-type"
    content="text/html; charset=utf-8" />
  <link rel="stylesheet" type="text/css" href="elements.css" />
</head>
<body>
  <form method="post" action="example1.html" id="form1">
    <div><label for="name">What is your name?</label><br/>
    <input type="text" name="name" id="name" /></div>
    <div><label for="color">Select your favorite color:</label>
    <select name="color" id="color">
      <option value="blue">blue</option>
      <option value="red">red</option>
      <option value="green">green</option>
      <option value="yellow">yellow</option>
    </select>
    </div>
    <div><label for="sex">Are you male or female?</label><br/>
    <input type="radio" name="sex" id="male"
value="male" />Male<br/>
    <input type="radio" name="sex" id="female"
value="female" />Female
    </div>
    <div>
    <label for="comments">Comments:</label><br/>
    <textarea name="comments" id="comments" cols="30"
rows="4"></textarea>
    </div>
    <div>
    <input type="submit" name="btnSubmit" id="btnSubmit"
value="Submit" />
    </div>
  </form>
</body>
</html>

```

**Внешний вид этой формы можно изменить, написав правила стилей для элементов form, input, textarea и select:**

*chapter06/elements.css*

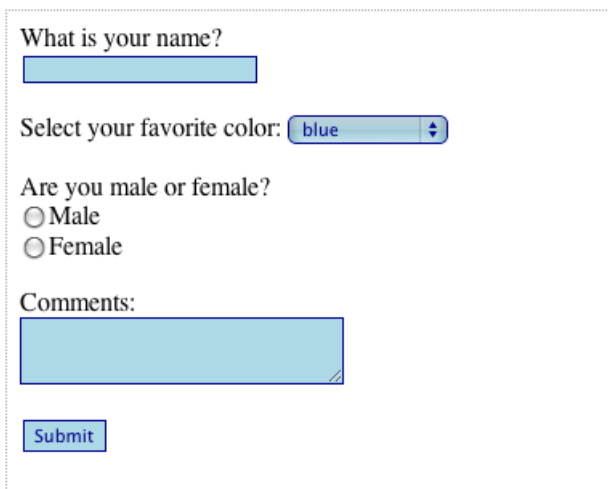
```

form {
  border: 1px dotted #AAAAAA;
  padding: 0.5em;
}
input {

```

```
    color: #00008B;
    background-color: #ADD8E6;
    border: 1px solid #00008B;
}
select {
    width: 100px;
    color: #00008B;
    background-color: #ADD8E6;
    border: 1px solid #00008B;
}
textarea {
    width: 200px;
    height: 40px;
    color: #00008B;
    background-color: #ADD8E6;
    border: 1px solid #00008B;
}
```

Теперь форма выглядит, как на рис. 6.2.



The image shows a web form with the following elements and styling:

- Label: "What is your name?"
- Text input field with a light blue background and a dark blue border.
- Label: "Select your favorite color:"
- Select dropdown menu with "blue" selected and a dark blue border.
- Label: "Are you male or female?"
- Radio buttons for "Male" and "Female".
- Label: "Comments:"
- Text area with a light blue background and a dark blue border.
- Submit button with a light blue background and a dark blue border.

Рис. 6.2. Та же форма после применения CSS-стилей

## Обсуждение

Как вы наверняка догадались, заданные правила стилей для HTML-элементов `form`, `input`, `textarea` и `select` будут применены по отношению к каждому их экземпляру, находящемуся на странице, с которой связан файл таблицы стилей. Для изменения оформления полей формы можно использовать множество различных свойств. С помощью CSS можно управлять практически всеми аспектами поля `<input type="text">`:

```

<input type="text" name="name" id="name" />

input {
  color: #00008B;
  background-color: #ADD8E6;
  border: 1px solid #00008B;
  font: 0.9em Arial, Helvetica, sans-serif;
  padding: 0.2em;
  width: 200px;
}

```

### Совет

**Формы и фоновый цвет.** Часть посетителей вашего сайта могут плохо различать цвета, а часть, возможно, пользуется голосовым браузером. Поэтому цвета никогда не должны выполнять важных функций – к примеру, указания в роде «Желтые поля обязательны для заполнения» должны быть под полным запретом.

Рассмотрим значения свойств более подробно:

<b>color</b>	изменяет цвет текста, расположенного в поле
<b>background-color</b>	определяет фоновый цвет поля
<b>border</b>	применяется по отношению к рамке вокруг поля; может изменять различные параметры
<b>font</b>	изменяет размер и тип шрифта текста поля
<b>padding</b>	отодвигает вводимый в поле текст от краев окошка
<b>width</b>	позволяет создать поля формы, соответствующей ожидаемым данным ширины (к примеру, для ввода инициалов достаточно короткого поля)

## Использование разных стилей для разных полей одной и той же формы

Элемент `input` может быть различных типов, и вам наверняка захочется определить разные стили для кнопок и флажков. Как создать разные правила стилей для полей формы?

### Решение

Чтобы задать стиль для различных полей, можно воспользоваться CSS-классами. Форма в следующем примере содержит два элемента `input`, один из которых является текстовым полем, а другой – кнопкой `Submit`. Каждому из них присвоен соответствующий класс:

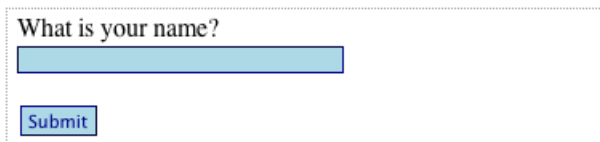
*chapter06/fields.html (фрагмент)*

```
<form method="post" action="fields.html">
<div>
  <label for="name">What is your name?</label><br />
  <input type="text" name="name" id="name" class="txt" />
</div>
<input type="submit" name="btnSubmit" id="btnSubmit"
  value="Submit" class="btn" />
</form>
```

*chapter06/fields.css*

```
form {
  border: 1px dotted #AAAAAA;
  padding: 3px 6px 3px 6px;
}
input.txt {
  color: #00008B;
  background-color: #ADD8E6;
  border: 1px inset #00008B;
  width: 200px;
}
input.btn {
  color: #00008B;
  background-color: #ADD8E6;
  border: 1px outset #00008B;
  padding: 2px 4px 2px 4px;
}
```

На рис. 6.3 показан результат выполнения данного кода.



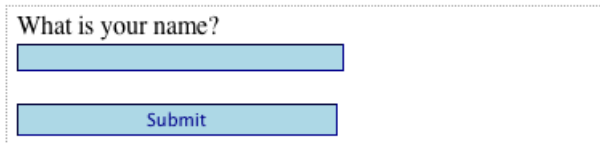
*Рис. 6.3. Применение различных классов для каждого из элементов input*

## Обсуждение

Итак, элементы `input` могут быть различных типов, и для каждого из них нужно определить свой собственный стиль, чтобы они отображались соответствующим образом. В приведенном выше примере для применения различных стилей для элементов `input` типа `text` и типа `submit` использовались классы. Если бы мы определили один набор стилей для элемента `input`, то в результате получили бы следующее (задав ширину и параметры рамки для текстового поля):

```
input {
  color: #00008B;
  background-color: #ADD8E6;
  border: 1px inset #00008B;
  width: 200px;
}
```

После применения данного правила стиля форма будет выглядеть следующим образом:

The image shows a web form with a text input field and a submit button. The text input field contains the text "What is your name?". Both the input field and the submit button are styled with a light blue background, a dark blue border, and dark blue text. The submit button is labeled "Submit".

*Рис. 6.4. Применение одинаковых стилей к обоим полям формы*

Теперь кнопка Submit гораздо больше похожа на текстовое поле, чем на кнопку!

С помощью разных классов можно оформить каждый элемент по отдельности, чтобы он отображался именно так, как задумано. В любом приложении формы, скорее всего, предназначаются для ввода различных типов данных. Некоторые текстовые поля нужны для ввода всего лишь пары символов; некоторые – для ввода имени или короткого слова, а другие – для ввода целых приложений. Их можно оформить путем создания CSS-классов для коротких, средних и длинных полей. Кроме того, это позволяет пользователям быть уверенными, что они вводят правильную информацию.

---

#### Совет

**Стили: задайте сразу, применяйте часто.** Начиная работу над сайтом, содержащим большое количество форм, я практически сразу создаю в таблице стилей несколько классов для стандартных форм. Неважно, что в дальнейшем может потребоваться изменить их стиль – для этого достаточно будет всего лишь поменять значения свойств. Главное, что стили применяются с самого начала, и потому любые изменения будут относиться ко всем имеющимся на сайте формам.

---

#### Совет

**Использование селекторов атрибутов для обращения к различным элементам форм.** Вместо классов для обращения к различным элементам форм можно использовать селекторы атрибутов. Мы уже познакомились с ними в разделе «Выделение ссылок, ведущих на внешний сайт» главы 4. Для адресации текстового поля в приведенном выше примере можно было бы воспользоваться следующим селектором:

```
input[type="text"] {  
  ⋮  
}
```

Для обращения к кнопке для подтверждения ввода данных можно было бы использовать следующий селектор:

```
input[type="submit"] {  
  ⋮  
}
```

При этом пропадает необходимость в добавлении каких-либо дополнительных классов в разметку. Следует учитывать, что **Internet Explorer 6 не поддерживает** данный селектор, поэтому при его использовании формы могут выглядеть странно или стать непригодными для использования. Если ваши посетители могут использовать данный браузер, придется применить метод классов.

## Избавление от переносов строки и потери места на странице

Форма является блочным элементом и, подобно абзацу, располагается на новой строке. Как правило, такое поведение вполне соответствует поставленным целям, однако в определенных случаях может возникнуть необходимость добавления небольшой формы непосредственно в поток элементов документа. В качестве примера можно привести окошко поиска, расположенное на одном уровне с другими элементами «шапки».

### Решение

Чтобы форма отображалась как внутритекстовый, а не блочный, элемент, можно задать значение `inline` свойству `display`:

*chapter06/inline.html (фрагмент)*

```
Your email address:  
<form method="post" action="inline.html">  
  <div><input type="text" name="name" id="name" class="txt" />  
  <input type="submit" name="btnSubmit" id="btnSubmit"  
    value="Submit" class="btn" /></div>  
</form>
```

*chapter06/inline.css*

```
form {  
  display: inline;  
}  
input.txt {  
  color: #00008B;  
  background-color: #E3F2F7;  
  border: 1px inset #00008B;
```



```

width: 200px;
}
input.btn {
color: #00008B;
background-color: #ADD8E6;
border: 1px outset #00008B;
}

```

Как видно на рис. 6.5, благодаря использованию CSS форма встраивается в общий поток документа и отображается как внутритекстовый элемент рядом с окружающим текстом:

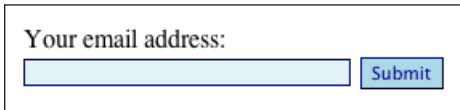


Рис. 6.5. Форма как внутритекстовый элемент

## Придание кнопке подтверждения вида текста

Кнопки должны выглядеть как кнопки, если вы хотите, чтобы пользователю было понятно, куда нужно нажимать. Однако в определенных случаях вы можете захотеть придать кнопке Submit формы вид обычного текста.

### Решение

Рассмотрим следующее правило стиля:

*chapter06/textbutton.css (фрагмент)*

```

.btn {
background-color: transparent;
border: 0;
padding: 0;
}

```

Расположенный на второй строчке текст `Next >>` на самом деле является кнопкой (рис. 6.6)!

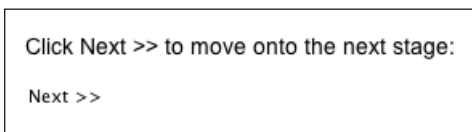


Рис. 6.6. Кнопка, похожая на обычный текст

## Возможность заполнения формы для пользователей текстовых устройств

Создание красивых и удобных форм для пользователей обычных браузеров – дело хорошее, но не забывайте о том, что многие пользователи увидят текстовую версию вашего сайта. Перед тем как начать написание таблицы стилей для вашей формы, убедитесь, что ее структура не затрудняет процесс заполнения для пользователей текстовых устройств.

### Решение

Чтобы ваша форма стала более доступной для различных пользователей, необходимо связать все ярлыки с соответствующими полями формы. При просмотре сайта с помощью текстового браузера или прослушивании содержащейся на нем информации посредством экранного диктора пользователю, возможно, будет нелегко определить, какого рода данные нужно ввести в каждое из полей, если структура формы недостаточно хорошо продумана.

Элемент `label` связывает ярлык с соответствующим ему полем формы. Его использование решает описанную проблему. Подобно другим элементам, стиль `label` можно с легкостью задать с помощью CSS:

*chapter06/textonly.html (фрагмент)*

```
<form method="post" action="textonly.html">
  <table>
    <tr>
      <td><label for="fullname">Name:</label></td>
      <td><input type="text" name="fullname" id="fullname"
class="txt" /></td>
    </tr>
    <tr>
      <td><label for="email">Email Address:</label></td>
      <td><input type="text" name="email" id="email" class="txt"/></td>
    </tr>
    <tr>
      <td><label for="password1">Password:</label></td>
      <td><input type="password" name="password1" id="password1" class="txt"
/></td>
    </tr>
    <tr>
      <td><label for="password2">Confirm Password:</label></td>
      <td><input type="password" name="password2" id="password2" class="txt"
/></td>
    </tr>
    <tr>
      <td><label for="level">Membership Level:</label></td>
```

```

        <td><select name="level">
            <option value="silver">silver</option>
            <option value="gold">gold</option>
        </select></td>
    </tr>
</table>
<p>
    <input type="submit" name="btnSubmit" id="btnSubmit"
        value="Sign Up!" class="btn" />
</p>
</form>

```

### *chapter06/textonly.css*

```

h1 {
    font: 1.2em Arial, Helvetica, sans-serif;
}
input.txt {
    color: #00008B;
    background-color: #E3F2F7;
    border: 1px inset #00008B;
    width: 200px;
}
input.btn {
    color: #00008B;
    background-color: #ADD8E6;
    border: 1px outset #00008B;
}
label {
    font : bold 0.9em Arial, Helvetica, sans-serif;
}

```

Результат применения этих правил стилей показан на рис. 6.7, однако примите во внимание, что в печатной книге практически невозможно отобразить преимущества их использования для пользователей со слабым зрением. Кроме того, помимо улучшения параметров юзабилити для текстовых браузеров и экранных дикторов, эти правила стилей

**User Registration Form**

**Name:**

**Email Address:**

**Password:**

**Confirm Password:**

**Membership Level:**

*Рис. 6.7. Отображение формы в браузере*

обеспечивают размещение курсора в соответствующем поле, когда пользователь щелкает по ярлыку при просмотре сайта в обычном визуальном браузере. Таким образом, от использования `label` выигрывают все!

## Обсуждение

С помощью элемента `label` можно точно указать, какого рода информацию пользователь должен ввести в поле. Как мы уже говорили, четкое указание на назначение каждого поля особенно важно для пользователей экранных дикторов. Использование `label` особенно актуально для форм, созданных с помощью таблицы, где ярлык располагается в одной ячейке, а соответствующее ему поле – в другой. (Чуть ниже я расскажу, как создать форму такой структуры без использования таблиц.)

Связь между элементом `label` и соответствующим полем формы обеспечивается с помощью атрибута `for` тега `<label>`. Значением атрибута служит ID поля:

*chapter06/textonly.html (фрагмент)*

```
<tr>
  <td><label for="fullname">Name:</label></td>
  <td><input type="text" name="fullname" id="fullname"
class="txt" /></td>
</tr>
```

Использование элементов `label` – важный шаг на пути к обеспечению правильного восприятия форм пользователями экранных дикторов. Не забывайте о том, что элементу `label` также можно задавать CSS-стили:

*chapter06/textonly.css (фрагмент)*

```
label {
  font: bold 0.9em Arial, Helvetica, sans-serif;
}
```

## Предупреждение

**Использование явных ярлыков.** Вместо использования элемента `label` можно указать ярлык явно. Для этого элемент формы должен стать дочерним для элемента `label` (что указывает на их связь); при этом не нужно использовать атрибут `for`, например:

```
<label> Name: <input type="text" name="fullname" id="fullname"
class="txt" /></label>
```

Такая верстка считается валидной, однако использовать такой прием не рекомендуется, поскольку некоторые программы специальных возможностей обрабатывают такой код некорректно.<sup>1</sup> Во избежание недоразумений всегда пользуйтесь атрибутом `for`.

---

<sup>1</sup> <http://www.w3.org/TR/WCAG20-GENERAL/H44.html>

## Создание двухколоночной формы с помощью CSS вместо таблиц

Сверстать форму без таблиц сложно, но вовсе не невозможно. На рис. 6.8 изображена сильно напоминающая таблицу форма, однако в исходном коде вы не найдете ни намека на HTML-таблицу.

The image shows a user registration form titled "User Registration Form". It is styled to look like a table with two columns. The first column contains labels for "Name:", "Email Address:", "Password:", and "Confirm Password:". The second column contains corresponding input fields: a text box for the name, a text box for the email address, a text box for the password, and another text box for the confirm password. Below these is a "Membership Level:" label followed by a dropdown menu currently showing "silver". At the bottom left of the form is a blue "Sign Up!" button.

Рис. 6.8. Форма в две колонки, созданная средствами CSS

### chapter06/tablefree.html

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
<head>
<title>Table-free form layout</title>
<meta http-equiv="content-type"
  content="text/html; charset=utf-8" />
<link rel="stylesheet" type="text/css" href="tablefree.css" />
</head>
<body>
<h1>User Registration Form</h1>
<form method="post" action="tablefree.html">
  <div>
    <label for="fullname">Name:</label>
    <input type="text" name="fullname" id="fullname"
      class="txt" />
  </div>
  <div>
    <label for="email">Email Address:</label>
    <input type="text" name="email" id="email" class="txt" />
  </div>
  <div>
    <label for="password1">Password:</label>

```

```
        <input type="password" name="password1" id="password1"
            class="txt" />
    </div>
</div>
<div>
    <label for="password2">Confirm Password:</label>
    <input type="password" name="password2" id="password2"
        class="txt" />
</div>
</div>
<div>
    <label for="level">Membership Level:</label>
    <select name="level">
        <option value="silver">silver</option>
        <option value="gold">gold</option>
    </select>
</div>
</div>
<div>
    <input type="submit" name="btnSubmit" id="btnSubmit"
value="Sign Up!" class="btn" />
</div>
</form>
</body>
</html>
```

### *chapter06/tablefree.css*

```
h1 {
    font: 1.2em Arial, Helvetica, sans-serif;
}
input.txt {
    color: #00008B;
    background-color: #E3F2F7;
    border: 1px inset #00008B;
    width: 200px;
}
input.btn {
    color: #00008B;
    background-color: #ADD8E6;
    border: 1px outset #00008B;
}
form div {
    clear: left;
    margin: 0;
    padding: 0;
    padding-top: 0.6em;
}
form div label {
    float: left;
    width: 40%;
    font: bold 0.9em Arial, Helvetica, sans-serif;
}
```

## Обсуждение

Перед нами пример создания типичной формы. Как мы уже смогли убедиться в начале главы, формы такой структуры обычно выполняются в виде таблицы, содержащей две колонки, в которой элемент ярлыка располагается в одной ячейке, а соответствующее ему поле – в другой:

*chapter06/textonly.html (фрагмент)*

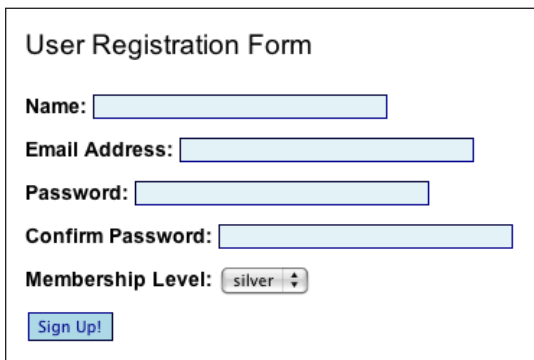
```

<form method="post" action="textonly.html">
  <table>
    <tr>
      <td><label for="fullname">Name:</label></td>
      <td><input type="text" name="fullname" id="fullname"
        class="txt" /></td>
    </tr>
    <tr>
      <td><label for="email">Email Address:</label></td>
      <td><input type="text" name="email" id="email" class="txt"/></td>
    </tr>
    <tr>
      <td><label for="password1">Password:</label></td>
      <td><input type="password" name="password1" id="password1" class="txt"
        /></td>
    </tr>
    <tr>
      <td><label for="password2">Confirm Password:</label></td>
      <td><input type="password" name="password2" id="password2" class="txt"
        /></td>
    </tr>
    <tr>
      <td><label for="level">Membership Level:</label></td>
      <td><select name="level">
        <option value="silver">silver</option>
        <option value="gold">gold</option>
      </select></td>
    </tr>
  </table>
  <p>
    <input type="submit" name="btnSubmit" id="btnSubmit"
      value="Sign Up!" class="btn" />
  </p>
</form>

```

Эта форма была сверстана с использованием таблицы, которая обеспечивает ровное расположение всех ее полей. Без таблицы поля отображались бы непосредственно после ярлыков, как показано на рис. 6.9.

При создании изображенной на рис. 6.9 формы каждый ее ряд был заключен внутри элемента `div`, и поэтому поле следует прямо за ярлыком.



The image shows a user registration form titled "User Registration Form". It contains the following elements:

- Name:** A text input field.
- Email Address:** A text input field.
- Password:** A text input field.
- Confirm Password:** A text input field.
- Membership Level:** A dropdown menu with "silver" selected.
- Sign Up!** A button.

Рис. 6.9. Форма, сверстанная без таблиц

*chapter06/tablefree.html (фрагмент)*

```
<form method="post" action="tablefree.html">
  <div>
    <label for="fullname">Name:</label>
    <input type="text" name="fullname" id="fullname" class="txt"
  />
  </div>
  <div>
    <label for="email">Email Address:</label>
    <input type="text" name="email" id="email" class="txt" />
  </div>
  :
  :
```

Чтобы созданная средствами CSS форма выглядела как таблица, не нужно вносить никаких изменений в HTML-разметку – достаточно всего лишь добавить простой CSS-код:

*chapter06/tablefree.css*

```
form div {
  clear: left;
  margin: 0;
  padding: 0;
  padding-top: 0.6em;
}
form div label {
  float: left;
  width: 40%;
  font: bold 0.9em Arial, Helvetica, sans-serif;
}
```

В данном случае мы обращаемся к элементу `label` напрямую из таблицы стилей. Ему заданы параметры шрифта, ширина и обтекание по левому краю.



Поскольку свойство `float` «вытаскивает» элемент из обычного потока элементов документа, необходимо задать нашим блокам `div` свойство `clear` со значением `left`, чтобы каждый из них начинался прямо под ярлыком предшествующего элемента `div`. Кроме того, необходимо задать нашим блокам значение свойства `padding-top`, чтобы создать небольшой зазор между рядами, вот и все!

## Группировка связанных полей формы

Заполнять большие формы гораздо удобнее, если посетитель может сразу понять, какие вопросы связаны между собой. Поэтому нужен какой-то способ показать взаимосвязи между данными – способ, который окажется полезен как для пользователей стандартных браузеров, так и для посетителей, использующих текстовые устройства и экранные дикторы.

### Решение

Для группировки связанных полей можно использовать элементы `fieldset` и `legend`:

*chapter06/fieldset.html (фрагмент)*

```
<form method="post" action="fieldset.html">
  <fieldset>
    <legend>Personal Information</legend>
    <div>
      <label for="fullname">Name:</label>
      <input type="text" name="fullname" id="fullname"
        class="txt" />
    </div>
    <div>
      <label for="email">Email Address:</label>
      <input type="text" name="email" id="email" class="txt" />
    </div>
    <div>
      <label for="password1">Password:</label>
      <input type="password" name="password1" id="password1"
        class="txt" />
    </div>
    <div>
      <label for="password2">Confirm Password:</label>
      <input type="password" name="password2" id="password2"
        class="txt" />
    </div>
  </fieldset>
  <fieldset>
    <legend>Address Details</legend>
    <div>
      <label for="address1">Address line one:</label>
      <input type="text" name="address1" id="address1">
```

```
        class="txt" />
    </div>
    <div>
        <label for="address2">Address line two:</label>
        <input type="text" name="address2" id="address2"
            class="txt" />
    </div>
    <div>
        <label for="city">Town / City:</label>
        <input type="text" name="city" id="city" class="txt" />
    </div>
    <div>
        <label for="zip">Zip / Post code:</label>
        <input type="text" name="zip" id="zip" class="txt" />
    </div>
</fieldset>
<div>
    <input type="submit" name="btnSubmit" id="btnSubmit"
        value="Sign Up!" class="btn" />
</div>
</form>
```

### *chapter06/fieldset.css*

```
h1 {
    font: 1.2em Arial, Helvetica, sans-serif;
}
input.txt {
    color: #00008B;
    background-color: #E3F2F7;
    border: 1px inset #00008B;
    width: 200px;
}
input.btn {
    color: #00008B;
    background-color: #ADD8E6;
    border: 1px outset #00008B;
}
form div {
    clear: left;
    margin: 0;
    padding: 0;
    padding-top: 5px;
}
form div label {
    float: left;
    width: 40%;
    font: bold 0.9em Arial, Helvetica, sans-serif;
}
fieldset {
    border: 1px dotted #61B5CF;
    margin-top: 1.4em;
```

```
padding: 0.6em;
}
legend {
font: bold 0.8em Arial, Helvetica, sans-serif;
color: #00008B;
background-color: #FFFFFF;
}
```

На рис. 6.10 показан вид сгруппированных элементов при просмотре документа в браузере.

The image shows a web form titled "User Registration Form". It is divided into two distinct sections, each enclosed in a dotted-line border. The first section, titled "Personal Information", contains four input fields: "Name:", "Email Address:", "Password:", and "Confirm Password:". The second section, titled "Address Details", contains four input fields: "Address line one:", "Address line two:", "Town / City:", and "Zip / Post code:". At the bottom left of the form is a "Sign Up!" button. The form is styled with a light blue background and blue text for labels and the legend.

Рис. 6.10. Разделение формы на две секции с помощью тега `<fieldset>`

## Обсуждение

Использование тегов `<fieldset>` и `<legend>` очень эффективно для группировки данных формы по категориям. Этот прием позволяет разделить различные группы элементов визуально, при этом экранные дикторы и текстовые устройства «понимают», что обозначенные элементы связаны друг с другом логически. Ситуация была бы иной, если бы вы просто вложили связанные элементы в блок `div` и задали для него стиль оформления, — их связь была бы вполне очевидна для пользователей обычных браузеров в отличие от тех, кто физически не может увидеть визуальные особенности, созданные с помощью CSS.

Для группировки связанных элементов формы достаточно разместить их между тегами `<fieldset>` и `</fieldset>`, причем непосредственно за открывающим тегом следует добавить тег `<legend>`, содержащий заголовок группы:

*chapter06/fieldset.html (фрагмент)*

```
<fieldset>
  <legend>Personal Information</legend>
  <div>
    <label for="fullname">Name:</label>
    <input type="text" name="fullname" id="fullname" class="txt"/>
  </div>
  <div>
    <label for="email">Email Address:</label>
    <input type="text" name="email" id="email" class="txt"/>
  </div>
  <div>
    <label for="password1">Password:</label>
    <input type="password" name="password1" id="password1" class="txt"/>
  </div>
  <div>
    <label for="password2">Confirm Password:</label>
    <input type="password" name="password2" id="password2"
class="txt" />
  </div>
</fieldset>
```

Стиль тегов `<fieldset>` и `<legend>`, как и других HTML-тегов, по умолчанию определяется браузером. При этом сгруппированные элементы окружаются рамкой, а содержание тега `<legend>` отображается в левом верхнем углу получившегося окошка. На рис. 6.11 представлен вид тегов `<fieldset>` и `<legend>` по умолчанию в браузере Firefox под Mac OS X.

The image shows a web browser window displaying a form titled "User Registration Form". The form is enclosed in a border and contains two main sections, each with a legend header and several input fields. The first section, "Personal Information", includes fields for Name, Email Address, Password, and Confirm Password. The second section, "Address Details", includes fields for Address line one, Address line two, Town / City, and Zip / Post code. A "Sign Up!" button is positioned at the bottom left of the form area.

Рис. 6.11. Вид тегов `<fieldset>` и `<legend>` по умолчанию

## Задание стиля для клавиш быстрого доступа

Клавиши быстрого доступа позволяют пользователю быстро переместиться к определенному месту документа или перейти по ссылке. Для этого достаточно нажать соответствующее сочетание клавиш; как правило, это клавиша Alt (или ее эквивалент) и некоторая конкретная клавиша. Конечно, пользователям нужно сообщить, какова эта конкретная клавиша!

### Решение

Во многих операционных системах буква, соответствующая необходимой клавише, отмечается в ключевом слове подчеркиванием. К примеру, под Windows комбинация клавиш Alt+F активирует выпадающее меню File. На эту возможность указывает подчеркивание буквы F в слове File, как показано на рис. 6.12.

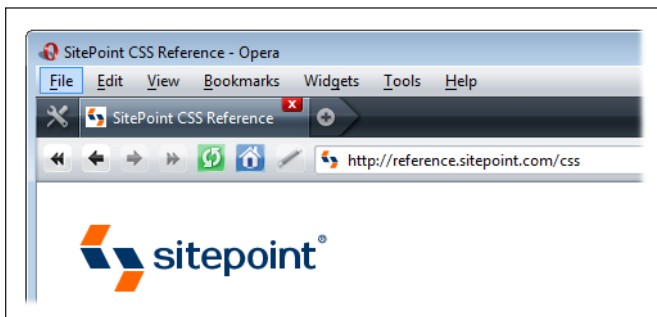


Рис. 6.12. Подчеркнутая буква F в слове File

Подобные приемы вполне подходят и для сайтов: можно использовать подчеркивание для выделения букв, соответствующих клавишам быстрого доступа:

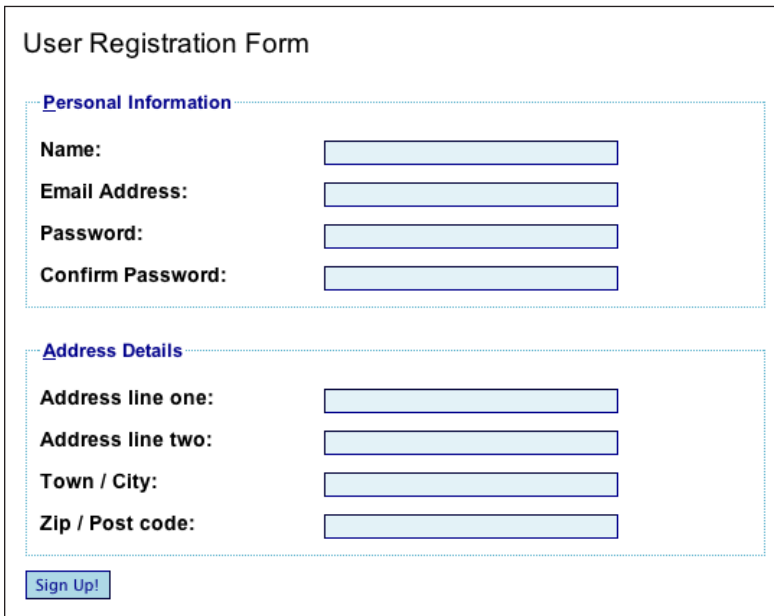
*chapter06/accesskeys.html (фрагмент)*

```
<fieldset>
  <legend><span class="akey">P</span>ersonal
  Information</legend>
  <div>
    <label for="fullname">Name:</label>
    <input type="text" name="fullname" id="fullname" class="txt" accesskey="p"
  />
</div>
```

*chapter06/accesskeys.css (фрагмент)*

```
.akey {
  text-decoration: underline;
}
```

Как видно на рис. 6.13, горячая клавиша для доступа к каждой группе полей выделена подчеркиванием.



The image shows a 'User Registration Form' with two sections: 'Personal Information' and 'Address Details'. Each section title is underlined. The 'Personal Information' section contains four input fields: 'Name:', 'Email Address:', 'Password:', and 'Confirm Password:'. The 'Address Details' section contains four input fields: 'Address line one:', 'Address line two:', 'Town / City:', and 'Zip / Post code:'. A 'Sign Up!' button is located at the bottom left of the form.

Рис. 6.13. Подчеркивание букв *P* в слове *Personal* и *A* в слове *Address* указывает на соответствующие клавиши быстрого доступа

## Обсуждение

Клавиши быстрого доступа удобны для людей, испытывающих затруднения при движении и не имеющих возможности воспользоваться мышью, а также пользователей, предпочитающих осуществлять навигацию с помощью клавиатуры. Для этих категорий посетителей можно создать клавиши быстрого доступа, позволяющие перейти к форме путем нажатия одной клавиши, а к определенному элементу формы – с помощью другой. Традиция подчеркивания буквы, соответствующей клавише быстрого доступа, хорошо знакома пользователям, привыкшим к использованию таких функций, хотя остальные пользователи могут даже не догадываться, что это означает.

Чтобы к полю формы можно было переместиться с помощью клавиши быстрого доступа, достаточно добавить соответствующему элементу атрибут `accesskey="x"`, где вместо `x` нужно подставить необходимый символ:

*chapter06/accesskeys.html (фрагмент)*

```
<div>
  <label for="fullname">Name:</label>
```

```
<input type="text" name="fullname" id="fullname" class="txt" accesskey="p" />
</div>
```

В нашем примере для доступа к первому элементу каждой группы была добавлена клавиша быстрого доступа. При нажатии на нее фокус сместится на первое поле формы, чтобы пользователи начали ее заполнение. Для выделения клавиши быстрого доступа я поместила первую букву заголовка группы полей `<legend>` внутри элемента `span` класса `akey`:

*chapter06/accesskeys.html (фрагмент)*

```
<legend><span class="akey">P</span>ersonal Information</legend>
```

Для класса `akey` задан стиль, в котором указано значение `underline` свойства `text-decoration`:

*chapter06/accesskeys.css (фрагмент)*

```
.akey {
    text-decoration: underline;
}
```

Различные браузеры и операционные системы используют разные сочетания клавиш для быстрого доступа. К примеру, в Internet Explorer и Firefox 1.5 под Windows используется клавиша `Alt`, однако во второй версии Firefox и выше используется `Alt+Shift` (на момент написания данной книги они работают только в сочетании с буквенными клавишами, в отличие от цифр). Safari использует `Ctrl`, как и Firefox под Mac OS X (опять же возможно сочетание только с буквами), а в Opera применяется сочетание `Shift+Esc`, однако пользователь может изменить данную настройку.

### Внимание

**Клавиши быстрого доступа не настолько широко доступны, как может показаться.** При создании клавиш быстрого доступа убедитесь, что они не совпадают с сочетаниям клавиш, используемыми браузером или операционной системой по умолчанию!

## Использование цветного фона для меню, созданного с помощью элементов `select`

Ранее мы уже научились изменять фоновый цвет выпадающего меню в форме. Однако можно ли использовать разный фон для каждого пункта меню, чтобы выделить различные варианты выбора?

### Решение

Чтобы различные пункты меню отображались в различных цветовых решениях, можно присвоить каждому из них различные классы. Для

пунктов меню можно изменять только значения свойств `color` и `background-color`.

### Внимание

---

**Safari не любит полосы.** Помните, что Safari не поддерживает возможность задания фонового цвета для пунктов меню `select`, поэтому описанное решение не будет работать в данном браузере.

---

Ниже представлен необходимый код:

*chapter06/select.html (фрагмент)*

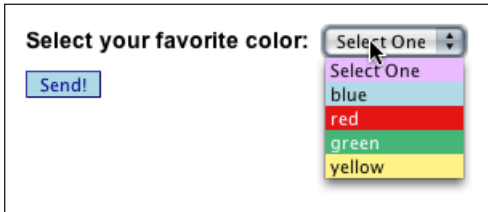
```
<form method="post" action="example8.html">
  <div>
    <label for="color">Select your favorite color:</label>
    <select name="color" id="color">
      <option value="">Select One</option>
      <option value="blue" class="blue">blue</option>
      <option value="red" class="red">red</option>
      <option value="green" class="green">green</option>
      <option value="yellow" class="yellow">yellow</option>
    </select>
  </div>
  <div>
    <input type="submit" name="btnSubmit" id="btnSubmit"
      value="Send!" class="btn" />
  </div>
</form>
```

*chapter06/select.css (фрагмент)*

```
option.blue {
background-color: #ADD8E6;
color: #000000;
}
option.red {
background-color: #E20A0A;
color: #ffffff;
}
option.green {
background-color: #3CB371;
color: #ffffff;
}
option.yellow {
background-color: #FFF280;
color: #000000;
}
```

Теперь меню выбора цвета, изображенное на рис. 6.14, действительно выглядит очень красочным.





*Рис. 6.14. Результат присваивания классов различным пунктам меню select при просмотре документа в браузере Opera*

## Обсуждение

Как правило, в CSS-коде следует избегать имен класса, связанных с определенным для них стилем. К примеру, назвать класс именем `blue` (голубой) – не самое удачное решение, поскольку в дальнейшем вы можете изменить параметры цвета в определенном для него стиле. В итоге вы получите множество заголовков класса `blue`, отображаемых при этом, скажем, в зеленом цвете, или вам придется изменять всю разметку. Однако в нашем случае, когда мы имеем дело с меню для выбора цвета, все выглядит достаточно разумно и естественно!

### Совет

**Стиль со смыслом.** Используйте различный фоновый цвет для выделения групп взаимосвязанных опций или чередуйте цветовое оформление пунктов меню `select`.

### Совет

**Альтернативный вариант: использование селекторов атрибутов.** Опять же вместо добавления классов в разметку можно воспользоваться селекторами атрибутов CSS. К примеру, вместо селектора `option.blue` можно было бы написать:

```
option[value="blue"] {
  background-color: #ADD8E6;
  color: #000000;
}
```

В этом случае также не следует забывать об отсутствии поддержки данного селектора браузером Internet Explorer 6.

## Создание таблицы стилей для формы с возможностью ввода данных, как в электронную таблицу

В подавляющем большинстве случаев формы возможно (и рекомендуется) верстать с помощью CSS, однако в определенных случаях данные

удобнее вводить в форму на основе таблицы. В качестве яркого примера можно привести веб-приложение в виде электронной таблицы.

Пользователи скорее всего хорошо знакомы с особенностями процесса ввода данных в электронные таблицы с помощью Microsoft Excel или аналогичных приложений. Это следует учитывать при дизайне интерфейса вашего приложения – схожесть интерфейса поможет пользователям быстро освоиться в новой среде. Возможно, следует создать таблицу и отформатировать ее с помощью CSS так, чтобы она напоминала электронную таблицу. Рассмотрим следующий код:

*chapter06/spreadsheet.html (фрагмент)*

```
<form method="post" action="spreadsheet.html">
<table class="formdata" summary="This table contains a form to input the
yearly income for years 1999 through 2002">
  <caption>Complete the Yearly Income 1999 - 2002</caption>
  <tr>
    <th></th>
    <th scope="col">1999</th>
    <th scope="col">2000</th>
    <th scope="col">2001</th>
    <th scope="col">2002</th>
  </tr>
  <tr>
    <th scope="row">Grants</th>
    <td><input type="text" name="grants1999" id="grants1999" /></td>
    <td><input type="text" name="grants2000" id="grants2000" /></td>
    <td><input type="text" name="grants2001" id="grants2001" /></td>
    <td><input type="text" name="grants2002" id="grants2002" /></td>
  </tr>
  <tr>
    <th scope="row">Donations</th>
    <td><input type="text" name="donations1999" id="donations1999" /></td>
    <td><input type="text" name="donations2000" id="donations2000" /></td>
    <td><input type="text" name="donations2001" id="donations2001" /></td>
    <td><input type="text" name="donations2002" id="donations2002" /></td>
  </tr>
  <tr>
    <th scope="row">Investments</th>
    <td><input type="text" name="investments1999"
      id="investments1999" /></td>
    <td><input type="text" name="investments2000"
      id="investments2000" /></td>
    <td><input type="text" name="investments2001"
      id="investments2001" /></td>
    <td><input type="text" name="investments2002"
      id="investments2002" /></td>
  </tr>
  <tr>
    <th scope="row">Fundraising</th>
    <td><input type="text" name="fundraising1999"
```

```

        id="fundraising1999" /></td>
<td><input type="text" name="fundraising2000"
        id="fundraising2000" /></td>
<td><input type="text" name="fundraising2001"
        id="fundraising2001" /></td>
<td><input type="text" name="fundraising2002"
        id="fundraising2002" /></td>
</tr>
<tr>
    <th scope="row">Sales</th>
    <td><input type="text" name="sales1999" id="sales1999" /></td>
    <td><input type="text" name="sales2000" id="sales2000" /></td>
    <td><input type="text" name="sales2001" id="sales2001" /></td>
    <td><input type="text" name="sales2002" id="sales2002" /></td>
</tr>
<tr>
    <th scope="row">Miscellaneous</th>
    <td><input type="text" name="misc1999" id="misc1999" /></td>
    <td><input type="text" name="misc2000" id="misc2000" /></td>
    <td><input type="text" name="misc2001" id="misc2001" /></td>
    <td><input type="text" name="misc2002" id="misc2002" /></td>
</tr>
<tr>
    <th scope="row">Total</th>
    <td><input type="text" name="total1999" id="total1999" /></td>
    <td><input type="text" name="total2000" id="total2000" /></td>
    <td><input type="text" name="total2001" id="total2001" /></td>
    <td><input type="text" name="total2002" id="total2002" /></td>
</tr>
</table>
<div><input type="submit" name="btnSubmit" id="btnSubmit"
    value="Add Data" /></div>
</form>

```

### *chapter06/spreadsheet.css*

```

table.formdata {
    border: 1px solid #5F6F7E;
    border-collapse: collapse;
    margin: 1em 0 2em 0;
}
table.formdata th {
    border: 1px solid #5F6F7E;
    background-color: #E2E2E2;
    color: #000000;
    text-align: left;
    font-weight: normal;
    padding: 0.2em 0.4em 0.2em 0.4em;
    margin: 0;
}

```

```
table.formdata td {
  margin: 0;
  padding: 0;
  border: 1px solid #E2E2E2;
}
table.formdata input {
  width: 80px;
  padding: 0.2em 0.4em 0.2em 0.4em;
  margin: 0;
  border: none;
}
```

На рис. 6.15 показана полученная форма, очень похожая на электронную таблицу.

Complete the Yearly Income 1999 - 2002				
	1999	2000	2001	2002
Grants				
Donations				
Investments				
Fundraising				
Sales				
Miscellaneous				
Total				

Рис. 6.15. Форма, напоминающая электронную таблицу

## Обсуждение

Наша задача состоит в создании формы, внешне напоминающей электронную таблицу, такую как таблица Excel, изображенная на рис. 6.16. Недавно мне пришлось создавать подобные формы для веб-приложения, содержащего огромное количество таблиц с данными. Клиент хотел, чтобы пользователь мог редактировать ее содержимое в соответствующем режиме; при этом вид таблицы должен был оставаться прежним.

Для достижения такого эффекта прежде всего нужно создать форму внутри структурированной таблицы, используя элементы заголовка (th), где необходимо, и добавив элемент caption и атрибут summary, чтобы она была доступна альтернативным устройствам. Полный код данной таблицы был представлен выше. Перед тем как мы добавили CSS-код, она выглядела, как показано на рис. 6.17.

The screenshot shows a Microsoft Excel window titled "Microsoft Excel - example". The spreadsheet contains the following data:

	A	B	C	D	E	F	G	H
1		1999	2000	2001	2002			
2	Grants	11,980	12,650	9,700	10,600			
3	Donations	4,780	4,989	6,700	6,590			
4	Investments	8,000	8,100	8,760	8,490			
5	Fundraising	3,200	3,120	3,700	4,210			
6	Sales	28,400	27,100	27,950	29,050			
7	Miscellaneous	2,100	1,900	1,300	1,760			
8	Total	58,460	57,859	58,110	60,700			
9								
10								
11								

Рис. 6.16. Просмотр электронной таблицы в Excel

Complete the Yearly Income 1999 - 2002

	1999	2000	2001	2002
Grants	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Donations	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Investments	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Fundraising	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Sales	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Miscellaneous	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Total	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

Рис. 6.17. Вид формы до применения таблицы стилей

Начать написание таблицы стилей для этой формы лучше всего с создания класса, содержащего все ее поля. Я назвала такой класс `formdata`:

`chapter06/spreadsheet.html` (фрагмент)

```
<table class="formdata" summary="This table contains a form to input
the yearly income for years 1999 through 2002">
```

Для данного класса определена рамка толщиной в один пиксел синего цвета, и свойству `border-collapse` задано значение `collapse`:

`chapter06/spreadsheet.css` (фрагмент)

```
table.formdata {
border: 1px solid #5F6F7E;
```

```
border-collapse: collapse;
}
```

Далее зададим стиль для заголовков колонок и рядов таблицы, т. е. для тега `<th>`. Для этого достаточно обратиться ко всем элементам `th`, расположенным в таблице класса `formdata`:

*chapter06/spreadsheet.css (фрагмент)*

```
table.formdata th {
border: 1px solid #5F6F7E;
background-color: #E2E2E2;
color: #000000;
text-align: left;
font-weight: normal;
padding: 0.2em 0.4em 0.2em 0.4em;
margin: 0;
}
```

Complete the Yearly Income 1999 - 2002				
	1999	2000	2001	2002
<b>Grants</b>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<b>Donations</b>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<b>Investments</b>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<b>Fundraising</b>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<b>Sales</b>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<b>Miscellaneous</b>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<b>Total</b>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

Рис. 6.18. Вид формы после применения стилей для элементов `table` и `th`

Для создания редактируемой таблицы необходимо скрыть рамку полей формы и добавить рамку ячейкам таблицы. Поскольку в таблице только элементы `input`, для которых нужно задать стиль, являются текстовыми полями, мы можем легко обратиться ко всем элементам `input` этой таблицы через класс `formdata`; это позволит избежать необходимости добавления классов ко всем текстовым полям.

Добавим рамку для элемента `td` и зададим свойству `border` элемента `input` значение `0`. Зададим ширину элемента `input`, поскольку нам известно, что вводимые данные не требуют много места. Затем зададим внутренние отступы, чтобы вводимый текст не приближался вплотную к краю поля:

*chapter06/spreadsheet.css (фрагмент)*

```
table.formdata td {
margin: 0;
padding: 0;
```

```
border: 1px solid #E2E2E2;
}
table.formdata input {
width: 80px;
padding: 0.2em 0.4em 0.2em 0.4em;
margin: 0;
border-width: 0;
border-style: none;
}
```

Вот и все! Если вы использовали этот прием, будьте уверены, что пользователь поймет, что таблицу можно редактировать. Удаление рамки вокруг полей формы только поможет пользователям, если они будут понимать, как заполнить форму, даже не поняв, что она присутствует!

### Примечание

**Некоторые браузеры отображают рамку вокруг элемента `input`.** Существуют браузеры (в основном это старые версии Safari, работающие под Mac), которые отображают рамку вокруг элемента `input`. В этом случае эффект будет несколько смазан, однако никак не затруднит работу с формой.

## Выделение поля формы, по которому пользователь щелкает мышью

Приложения вроде Excel выделяют поле формы, на которое пользователь смещает фокус, перемещая указатель мыши или используя клавиатуру. Можно ли создать аналогичный эффект для нашей веб-формы?

### Решение

Благодаря наличию псевдокласса `:focus` такого эффекта можно достичь исключительно средствами CSS. Предлагаемое решение будет корректно работать во всех современных браузерах, в том числе и Internet Explorer 8, но не в Internet Explorer 6 и 7:

*chapter06/spreadsheet2.css (фрагмент)*

```
table.formdata input {
width: 80px;
padding: 0.2em 0.4em 0.2em 0.4em;
margin: 0;
border: 2px solid #FFFFFF;
}
.formdata input:focus {
border: 2px solid #000000;
}
```

На рис. 6.19 показан результат обработки данного кода.

Complete the Yearly Income 1999 - 2002				
	1999	2000	2001	2002
Grants				
Donations		<input type="text"/>		
Investments				
Fundraising				
Sales				
Miscellaneous				
Total				

Рис. 6.19. Выделение поля формы, на которое наведен фокус (в Firefox)

## Обсуждение

Добавить рамку или изменить цвет фона поля формы при смещении фокуса на него достаточно просто. На самом деле для этого достаточно всего лишь воспользоваться селектором псевдокласса `:focus` для изменения стиля отображения элемента `input`, когда пользователь щелкает по нему мышью.

Как уже было сказано, к сожалению, **Internet Explorer 6 не поддерживает псевдокласс `:focus`**, поэтому часть пользователей вашего приложения не увидят данный эффект.

У этой проблемы есть решение, однако оно требует применения JavaScript. Добавьте в ваш документ после таблицы следующий код:

*chapter06/spreadsheet2.html (фрагмент)*

```
<script type="text/javascript">
var editcells =
document.getElementById('form1').getElementsByTagName('input');
for (var i = 0; i < editcells.length; i++) {
  editcells[i].onfocus = function() {
    this.className += ' hilite';
  }
  editcells[i].onblur = function() {
    this.className = this.className.replace('hilite', '');
  }
}
</script>
```

Помимо добавления данного сценария нужно добавить в файл с CSS-кодом класс `hilite`, задав для него те же правила, что и ранее для псевдокласса `:focus`:



*chapter06/spreadsheet2.css (фрагмент)*

```
.formdata input:focus, .formdata input.hilite {  
    border: 2px solid #000000;  
}
```

Теперь поле будет выделяться и в Internet Explorer 6, а кроме того, во всех браузерах, поддерживающих псевдокласс `:focus`.

## Заключение

В данной главе мы рассмотрели различные методы оформления форм с помощью CSS, начиная от изменения стиля отображения их элементов и заканчивая версткой всей формы целиком. Мы смогли убедиться в том, что CSS – **мощный инструмент, позволяющий существенно усовершенствовать внешний вид формы и сделать ее более удобной в использовании.** Мы также обсудили проблемы доступности формы для пользователей альтернативных устройств и увидели, как аккуратная разметка способствует легкости восприятия контента и более эффективной работе с сайтом.

# 7

## Кросс-браузерные решения

В данной главе мы рассмотрим приемы, применение которых обеспечит должное отображение и функционирование ваших сайтов в большинстве браузеров. Ваши пользователи используют самые различные браузеры и далеко не всегда их самые современные версии, но вам наверняка хотелось бы, чтобы использование той или иной программы не оказывало существенного влияния на работу с сайтом.

Как уже было сказано ранее, **CSS позволяет отделить структуру и содержание документа от его оформления**. Следование такому принципу дает пользователям устройств, не воспроизводящих дизайн сайта, возможность доступа к его контенту. К таким устройствам относятся карманные ПК или смартфоны, использование которых накладывает определенные ограничения с технической точки зрения, а также экранные дикторы, воспроизводящие текст для пользователей со слабым зрением, и другие подобные устройства – в силу своих функциональных особенностей. **CSS дает разработчику свободу: теперь можно разрабатывать красивые дизайнерские решения, доступные большинству пользователей, чьи браузеры поддерживают CSS, не в ущерб остальным категориям пользователей.**

Помимо подробного рассмотрения особенностей различных браузеров в данной главе содержится описание полезных приемов, помогающих исправить ошибки интерпретации **CSS различными браузерами**. Представить в данной главе все известные ошибки не представляется возможным, и, даже если бы я предприняла такую попытку, появление новых сбоях неизбежно. Я поставила перед собой цель выявить основные причины возникновения ошибок при обработке **CSS-кода браузерами** и рассказать о возможных решениях данной проблемы, а также об основных источниках информации, где вы можете получить совет. Я расскажу о том, как следует искать причину возникновения ошибки, и о том, как правильно задавать вопросы, чтобы получить желаемый ответ.

Однако есть и хорошая новость: с выходом каждого нового издания данной книги жизнь разработчика веб-страниц становится легче. Сегодня большинство возникающих проблем связано с совершенно устаревшими браузерами, такими как **Internet Explorer 6**, однако число их пользователей сокращается с каждым днем. Новые версии браузеров, как правило, хорошо поддерживают текущие стандарты: тестируя свою работу в последних версиях Internet Explorer, Safari, Firefox, Opera и Chrome, я чаще всего получаю практически одинаковый результат. Мы стремились к этому долгое время, и наконец цель почти достигнута.

## В каких браузерах следует протестировать свой сайт?

Когда-то дизайнеры беспокоились главным образом о том, чтобы их сайты хорошо выглядели в Internet Explorer и Netscape Navigator. Эти времена далеко в прошлом. **Internet Explorer все еще занимает лидирующие позиции** на рынке браузеров, однако в ходу у пользователей есть несколько других программ, среди которых и экранные дикторы, и браузеры для мобильных устройств.

### Решение

Ответ состоит в том, что следует тестировать ваши сайты в как можно большем количестве браузеров. Возможность установки различных типов браузеров зависит от доступных для вас операционных систем. В табл. 7.1 представлены основные браузеры, которые можно установить на базе **Windows, Mac OS и Linux**. **По меньшей мере, необходимо протестировать сайт в браузерах Internet Explorer 6, 7 и 8, Firefox, Opera, Safari и Chrome.**

### Примечание

**Что движет браузером?** Возможно, вы сталкивались с термином «**браузерный движок**». Если браузер – это полноценный пакет программ, включающий интерфейс и определенный набор функций, то движок – его часть, отвечающая за интерпретацию HTML- и CSS-кода и отображение веб-страниц. Некоторые движки являются отдельными программными продуктами, на основе которых создано несколько разных браузеров. К примеру, разработанный Mozilla Foundation движок Gecko<sup>1</sup> используется не только браузером Firefox, но и Camino, а также последними версиями Netscape Navigator. На движке WebKit<sup>2</sup> основаны Safari и Chrome, а сам он был создан на базе движка KHTML,<sup>3</sup> который лег в основу браузера Konqueror.

---

<sup>1</sup> <https://developer.mozilla.org/en/Gecko>

<sup>2</sup> <http://webkit.org/>

<sup>3</sup> <http://konqueror.kde.org/features/browser.php>

Вы, наверное, подумали, что если два браузера созданы на основе одного и того же движка, то достаточно протестировать сайт в одном из них. Отчасти это верно, однако при этом все равно возможны расхождения в интерпретации, особенно при использовании различных версий браузеров под разными платформами. Некоторые браузеры, например, **Internet Explorer** и **Opera**, основаны на своем собственном движке.

### Совет

**По следам редких и устаревших браузеров.** Более старые версии Internet Explorer (ниже 6), а также многие другие старые и мало распространенные браузеры, можно загрузить с <http://browsers.evolt.org/>.

Возможность установки и тестирования сайта в тех или иных браузерах зависит от вашей операционной системы. В табл. 7.1 представлен список наиболее распространенных браузеров, а также информация об используемом ими движке и адреса сайтов, где их можно скачать.

Таблица 7.1. Браузеры, движки и операционные системы

Браузер (движок)	Win	Mac	Linux	Страница загрузки
Internet Explorer версии 6, 7, 8	✓			<a href="http://www.microsoft.com/windows/internet-explorer/">http://www.microsoft.com/windows/internet-explorer/</a>
Firefox (Gecko)	✓	✓	✓	<a href="http://www.mozilla.com/">http://www.mozilla.com/</a>
Camino (Gecko)		✓		<a href="http://www.caminobrowser.org/">http://www.caminobrowser.org/</a>
Opera	✓	✓	✓	<a href="http://www.opera.com/">http://www.opera.com/</a>
Safari (WebKit)	✓	✓		<a href="http://www.apple.com/safari/">http://www.apple.com/safari/</a>
Chrome (WebKit) <sup>a</sup>	✓	✓	✓	<a href="http://www.google.com/chrome/">http://www.google.com/chrome/</a>
Konqueror (KHTML)			✓	<a href="http://konqueror.kde.org/">http://konqueror.kde.org/</a>

## Тестирование сайта в различных браузерах при наличии только одной ОС

Если в вашем офисе нет полного набора для тестирования сайтов, то, скорее всего, вы обнаружите, что не имеете возможности установить определенные браузеры, связанные с конкретными операционными системами.

<sup>a</sup> На момент написания данной книги версии для Mac OS и Linux находились в стадии разработки, и их официальный релиз только ожидается в будущем.

## Решение

Существует множество способов запуска дополнительных операционных систем на вашем компьютере, дающих вам возможность установки и использования разработанных специально для них браузеров.

### Пользователи Windows

Пользователи Windows оказываются в выигрышном положении, если речь идет о тестировании сайта в различных браузерах. По приблизительным расчетам, 60–70 процентов пользователей Интернета используют различные версии браузера **Internet Explorer**, а большинство прочих популярных браузеров имеют версию для Windows. К сожалению, когда дело касается тестирования в браузерах для Mac OS (например, Safari), их возможности, доступные под Windows, ограничены.

### Тестирование в браузерах для Mac

В настоящее время сложнее всего эмулировать платформу Mac OS X, и потому под рукой у любого серьезного веб-дизайнера должен находиться компьютер с данной операционной системой – совсем не обязательно мощный, если вы предполагаете использовать его исключительно для тестирования сайтов в Safari.

В середине 2007 года Apple вызвала переполох в сообществе веб-разработчиков, выпустив версию Safari для Windows. К сожалению, отображение страниц в этом браузере и его более старом собрате для Mac далеко не идентично. Тем не менее его можно использовать для выявления возможных проблем интерпретации.

То же самое можно сказать и о Windows-версии **Google Chrome**. Несмотря на то что он использует тот же движок, что и Safari (WebKit), следует протестировать сайт в версии Safari для Mac.

### Тестирование в браузерах для Linux

На данный момент не существует способов эмуляции Mac на компьютере с установленной системой Windows, однако есть множество возможностей просмотра сайтов в браузерах, предназначенных для Linux.

### CD-диски с Linux (Live CD)

Это версии Linux, запускаемые прямо с CD-диска. Их можно использовать для тестирования, не устанавливая Linux на жесткий диск. Одним из самых распространённых Live CD с Linux является Knoppix; его можно загрузить с сайта Knoppix.<sup>1</sup> Knoppix поставляется с настольной средой KDE, в которую входит браузер Konqueror. Еще одним популярным дистрибутивом Linux в виде Live CD является Ubuntu,<sup>2</sup> в котором

---

<sup>1</sup> <http://www.knoppix.net/>

<sup>2</sup> <http://www.ubuntu.com/>

по умолчанию используется настольная среда Gnome. Полный список других Live CD доступен на сайте FrozenTech.<sup>1</sup>

### Установка Linux в качестве второй операционной системы

Есть и другой вариант: можно установить на один компьютер две операционные системы, например, Windows и Linux, а при запуске выбрать требуемую. Подробное описание процесса установки двух операционных систем, их настройки и использования можно найти по адресу <https://help.ubuntu.com/community/WindowsDualBoot>.

### Создание виртуальной среды

Вместо того чтобы устанавливать две операционные системы, их можно запускать как виртуальные машины под текущей платформой. Для этого можно использовать Parallels Workstation<sup>2</sup> и VMWare Workstation<sup>3</sup> – коммерческие приложения, обеспечивающие запуск виртуальной машины Linux под Windows; однако они достаточно дорогие, в особенности если вам нужно только протестировать сайт в браузерах для Linux. В этом случае целесообразнее воспользоваться версией VirtualBox<sup>4</sup> для Windows – свободно распространяемым приложением с открытым кодом, с помощью которого можно в том числе и запускать виртуальные машины Linux.

### Пользователи Mac

Пользователи Mac на базе процессора Intel оказываются в таком случае в выигрыше – у них есть возможность тестирования сайта во всех трех операционных системах. Если вы как разработчик хотите работать только на одной машине, стоит задуматься о широких возможностях Mac – а это вам говорит пользователь Linux!

### Установка двух операционных систем с помощью Boot Camp

Для установки Windows в качестве второй операционной системы пользователи Mac могут воспользоваться программой Boot Camp.<sup>5</sup> Она не предназначена для запуска виртуальной машины в отдельном окне на рабочем столе; т. е., чтобы запустить Windows, вам придется выполнить перезагрузку компьютера. Однако это вполне удобно для проведения тестирования сайтов. Boot Camp включен в Mac OS X начиная с версии 10.5 (Leopard).

---

<sup>1</sup> <http://www.frozentech.com/content/livecd.php>

<sup>2</sup> <http://www.parallels.com/products/workstation/>

<sup>3</sup> <http://www.vmware.com/products/ws/>

<sup>4</sup> <http://www.virtualbox.org/>

<sup>5</sup> <http://www.apple.com/macosex/bootcamp/>

## Создание виртуальной среды

Поскольку Apple начала выпуск своих машин на базе процессора Intel, у пользователей появилась возможность запускать виртуальные машины Windows и Linux с помощью приложений, созданных сторонними разработчиками для Mac OS X. Вы сможете даже запускать различные версии Windows и тестировать свои сайты в браузерах Internet Explorer 6, 7 и 8 на одном компьютере!

Традиционно пользуется популярностью приложение Parallels Desktop для Mac OS X, изображенное на рис. 7.1.<sup>1</sup> VMWare Workstation – популярное решение для создания виртуальной среды для Windows; позже компания выпустила VMWare Fusion для Mac OS X.<sup>2</sup> Если вас интересуют свободно распространяемые решения, обратите внимание на VirtualBox с открытым исходным кодом для Mac OS X.<sup>3</sup>

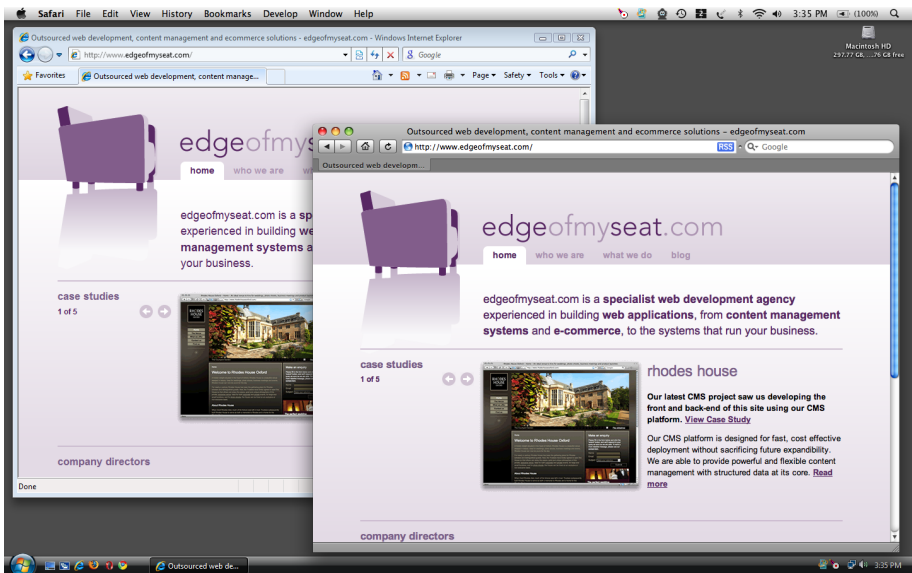


Рис. 7.1. Запуск Internet Explorer 8 и Safari на Mac OS X с помощью Parallels

## Пользователи Linux

В том, что касается тестирования сайтов в браузерах для Mac, пользователи Linux оказываются примерно в том же положении, что и пользователи Windows. Но выход существует – использование виртуальной среды и установка двух операционных систем позволяет пользователям Linux запускать различные версии Internet Explorer.

<sup>1</sup> <http://www.parallels.com/>

<sup>2</sup> <http://www.vmware.com/products/fusion/>

<sup>3</sup> <http://www.virtualbox.org/>

### Тестирование в браузерах для Mac

Для тестирования сайтов в браузерах для Mac пользователям Linux, как и пользователям Windows, нужен компьютер с установленной на нем Mac OS X.

Safari и Chrome основаны на движке WebKit, который в свою очередь был создан на базе KHTML, используемой браузером среды KDE, Konqueror (и изначально разработанной для него). Последний обычно отображает документы примерно как и Safari, и версия Chrome для Linux (на момент написания данной книги находившаяся в разработке) должна функционировать аналогичным образом. Безусловно, это нельзя считать полноценной заменой тестированию под Mac, однако с помощью этих браузеров можно приблизительно оценить вид страниц в Safari.

### Тестирование в браузерах для Windows

Для тестирования страниц в браузерах для Windows пользователям Linux проще всего установить на своем компьютере дополнительную операционную систему, однако существует множество инструментов, облегчающих процесс тестирования непосредственно в среде Linux.

#### Создание виртуальной среды

С помощью коммерческих приложений VMWare Workstation и Parallels Workstation можно запускать виртуальную машину Windows под Linux. Однако в качестве варианта также можно рассматривать свободно распространяемый VirtualBox.<sup>1</sup>

#### Установка двух операционных систем

Пользователи Linux также могут дополнительно установить на свой компьютер систему Windows, которую можно будет выбрать при включении.

## Сервисы, показывающие вид сайта в различных браузерах

Чтобы убедиться в том, что сайт нормально отображается при просмотре в различных браузерах, лучше всего протестировать его в каждом из них. Однако если в вашем офисе нет полного набора инструментов для тестирования, скорее всего вы не сможете проверить, как сайт будет отображен для пользователей определенных браузеров.

### Решение

Существует множество сервисов для проверки вида сайта в различных браузерах под различными платформами, к которым у вас может не быть доступа. Как правило, это происходит следующим образом: вы

---

<sup>1</sup> [http://www.virtualbox.org/wiki/Linux\\_Downloads](http://www.virtualbox.org/wiki/Linux_Downloads)



предоставляете URL-адрес сайта и получаете его скриншоты при его открытии в разных браузерах под разными операционными системами и с различным разрешением экрана.

Некоторые сервисы предоставляют доступ к машине, на которой вы можете проверить работу сайта на альтернативных платформах. Это особенно удобно, если вы используете JavaScript и хотите протестировать не только внешний вид сайта, но и его функциональные возможности.

Ниже приведен список доступных сервисов:

### **BrowserCam**

BrowserCam (<http://www.browsercam.com/>) – сервис создания скриншотов с возможностью удаленного доступа. Есть бесплатный пробный период.

### **Litmus**

Litmus (<http://litmusapp.com/>) – профессиональный сервис тестирования веб-страниц и HTML-документов по электронной почте, генерирующий скриншоты и предлагающий некоторые другие услуги. Можно выбрать платный или бесплатный тип учетной записи.

### **CrossBrowserTesting.com**

CrossBrowserTesting.com (<http://crossbrowsertesting.com/>) – коммерческий сервис тестирования, предоставляющий удаленный доступ к машинам, на которых установлены другие браузеры и операционные системы.

### **Browser Shots**

Browser Shots (<http://browsershots.org/>) – бесплатный, но медленно работающий сервис скриншотов, позволяющий разработчику протестировать свой документ в большом количестве браузеров.

Примечательно, что в недавнем времени Adobe и Microsoft выпустили собственное программное обеспечение для тестирования браузеров.

### **Adobe BrowserLab**

Adobe BrowserLab (<https://browserlab.adobe.com/index.html>) – сервис создания скриншотов с возможностью их сравнения, в том числе и путем наложения друг на друга с небольшой прозрачностью, благодаря которой становится легче обнаружить различия. В настоящее время он поддерживает только **Internet Explorer**, **Firefox** и **Safari**. У BrowserLab также есть расширение для Dreamweaver.

### **Microsoft SuperPreview**

Microsoft SuperPreview (<http://expression.microsoft.com/en-us/dd565874.aspx>) – приложение для **Windows**, предлагающее возможность сравнения скриншотов как при расположении рядом, так и путем наложения. В настоящее время поддерживает только **IE6**, **7** и **8**, однако в будущем планируется реализация поддержки других браузеров.

## Обсуждение

Кроме того, если у вас нет возможности просмотра сайта в каком-либо браузере, можно воспользоваться почтовой рассылкой. Пользователи большинства форумов, в том числе и форумов SitePoint Forums,<sup>1</sup> и подписчики рассылок, посвященных веб-дизайну и веб-программированию, привыкли к просьбам протестировать сайт, а в качестве благодарности вы можете помочь другим людям увидеть их страницы в доступных вам браузерах.

## Возможность поддержки нескольких версий Internet Explorer в Windows

Различные версии браузера Internet Explorer (6, 7 и 8) по-разному обрабатывают CSS-код, но Windows, как правило, не позволяет устанавливать сразу несколько версий этого браузера. Как в таком случае протестировать сайт в более старых, но все еще используемых версиях Internet Explorer?

## Решение

Программное обеспечение Virtual PC от Microsoft, распространяемое бесплатно, позволяет протестировать сайт в браузерах Internet Explorer версий 6, 7 и 8 на одном компьютере с операционной системой Windows. Для этого вам придется пройти несколько этапов, но сам способ действительно эффективен. В качестве основного браузера на вашей машине будет установлен Internet Explorer 8, а на виртуальных машинах будет доступен Internet Explorer 6 и 7.

1. Обновите Internet Explorer до версии 8, если вы еще этого не сделали.
2. Загрузите и установите Virtual PC 2007 с сайта Microsoft Virtual PC.<sup>2</sup>
3. Загрузите образ виртуальной машины Virtual PC (ограничен по времени действия) с предустановленным Microsoft Windows и Internet Explorer для IE6 и IE7 из Центра загрузок Microsoft.<sup>3</sup> Большим преимуществом использования этого образа является отсутствие необходимости в покупке дополнительной лицензии для запуска Windows.
4. Для начала работы с образом виртуальной машины распакуйте архив и запустите Virtual PC. **Включите проводник файлов образа, и отдельная версия Windows запустится в окне на вашем рабочем столе.**

---

<sup>1</sup> <http://www.sitepoint.com/forums/>

<sup>2</sup> <http://www.microsoft.com/windows/virtualpc/default.mspx>

<sup>3</sup> <http://go.microsoft.com/fwlink/?LinkId=70868>

---

**Совет**

**Режим IE7 в IE8.** На панели Developer Tools браузера Internet Explorer 8 есть функция включения режима интерпретации IE7. Она работает по принципу обратной совместимости, гарантирующему нормальное отображение сайтов, не работающих в IE8. В этом случае браузер начинает вести себя как предыдущая версия, что может решить возникшую проблему. Кроме того, это дает вам возможность быстро протестировать свой сайт в IE7, если вы по тем или иным причинам не хотите запускать образ виртуальной машины (или же запускаете одну виртуальную машину Windows в системе Mac).

---

На момент написания настоящей книги Microsoft не предоставляла образов виртуальной машины для более ранних версий Internet Explorer, однако если вам действительно крайне необходимо протестировать сайт в более ранних версиях данного браузера, чем 6, существуют их автономные версии, в которых можно проверить, насколько правильно интерпретируется ваш CSS-код. Программу установки нескольких автономных версий Internet Explorer можно скачать на сайте Tredosoft,<sup>1</sup> но учтите, что они могут сильно «капризничать». Как правило, в них бесполезно тестировать сценарии на JavaScript, поскольку для их обработки используется уже установленная на данный момент машина JScripT, а не ее более ранние версии, которые были бы установлены с теми версиями Internet Explorer. Однако такой способ вполне подойдет для разработчиков, желающих протестировать свою работу в самых старых версиях Internet Explorer.

---

**Примечание**

**Автономные версии Internet Explorer.** Существуют инсталляторы, в теории позволяющие вам запускать несколько версий Internet Explorer на одном компьютере, но я рекомендую вам воздержаться от их использования. Поскольку IE является частью Windows, невозможно запустить две абсолютно разные версии IE на одной машине, и некоторые приемы, например, использование условных комментариев, не будут работать корректно. Для тестирования сайта в различных версиях IE нужно либо несколько компьютеров, либо несколько виртуальных машин.

---

## Определение круга браузеров, для которых необходимо поддерживать все аспекты дизайна страницы

Как вы еще увидите, для старых браузеров можно написать отдельные таблицы стилей, а затем определить, какую таблицу должна использовать каждая конкретная версия, с помощью условных комментариев. Огромным преимуществом верстки страницы с применением CSS

---

<sup>1</sup> [http://tredosoft.com/Multiple\\_IE](http://tredosoft.com/Multiple_IE)

является доступность контента всем пользователям вне зависимости от того, насколько старым браузером они пользуются. При этом нужно четко осознавать, что вы не обязаны представить посетителям вашего сайта, использующим IE5, в точности то же, что пользователям IE8 или последней версии Firefox.

## Решение

Соглашение с заказчиком или политика вашей компании почти полностью определяет для вас границы реализации для различных браузеров, однако если у вас есть право голоса по этому вопросу, вы можете обратиться к схеме YUI Graded Browser Support,<sup>1</sup> которая послужит опорой для обоснования подхода к реализации поддержки определенных функций в различных браузерах. По сути, с помощью данной схемы разработчики Yahoo! определяют степень поддержки посетителей своего сайта, использующих различные браузеры. Поскольку домашняя страница Yahoo! является самой посещаемой в мире, они располагают достаточным количеством данных для обоснования своих решений.

На мой взгляд, нельзя закрывать доступ к контенту вашего сайта ни для одного пользователя; как вы увидите несколько позже, для самых старых браузеров можно создать даже простую HTML-версию сайта, вовсе не используя CSS. Для более новых браузеров, еще не поддерживающих спецификации CSS 2.1 и CSS3 (в настоящее время получающую широкое распространение), можно ограничиться более лаконичным дизайном – данный подход вполне приемлем для Internet Explorer 6. В разделе «Достижение прозрачности изображения в формате PNG в Internet Explorer 6» мы рассмотрим два способа решения проблемы, связанной с отсутствием поддержки прозрачности формата PNG браузером IE6. При этом разработчик может пойти одним из двух путей: найти способ отображения всех особенностей дизайна, включая прозрачность, или создать более простую версию сайта с использованием файлов в формате GIF. Схема YUI Graded Browser Support может помочь вам в выборе решения и предоставит необходимую информацию для его объяснения клиенту.

## Указание базовой таблицы стилей для самых старых браузеров

В настоящее время технология CSS получила настолько широкое распространение в веб-среде, что пользователям очень старых браузеров, таких как Netscape 4, доступны лишь крайне ограниченные возможности современных сайтов. Однако с ними также следует считаться, по крайней мере обеспечивая возможность нормального восприятия контента и отсутствие сбоев, вызванных использованием современных до-

---

<sup>1</sup> <http://developer.yahoo.com/yui/articles/gbs/>

стижений CSS. Для таких устаревших браузеров нужно создать специальную простую таблицу стилей, а основную таблицу стилей привязать не распознаваемыми ими средствами.

## Решение

Netscape 4, Internet Explorer 4 и другие старые браузеры не поддерживают возможность ссылки на таблицу стилей с помощью метода `@import`, а благодаря особенностям CSS эта непонятная команда будет ими проигнорирована. Мы можем воспользоваться этим для выполнения нашей задачи: более старые браузеры прочтут специально созданный для них набор стилей, а более новые браузеры, понимающие команду `@import`, смогут увидеть полную таблицу стилей.

Внутри элемента `head` вашего документа необходимо указать ссылку на базовую таблицу стилей с помощью элемента `link` – она будет прочтена всеми браузерами, поддерживающими CSS. Затем привяжите полную таблицу стилей (или таблицы стилей) с помощью метода `@import`, который будет пропущен без внимания старыми браузерами:

### *chapter07/basicstyles.html*

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <meta http-equiv="content-type" content="text/html;
      charset=utf-8" />
    <title>Serving a basic stylesheet</title>
    <link rel="stylesheet" href="basic_basic.css" type="text/css"
      media="screen" />
    <style type="text/css" media="screen">
      @import "basic_default.css";
    </style>
  </head>

  <body>
    <div class="content">
      <h1>Serving a basic style sheet to old browsers</h1>
      <p>CSS is now used so extensively on the Web that ...</p>
    </div>
  </body>
</html>

```

Представленная ниже таблица стилей `basic_basic.css` содержит несколько простых правил стилей, обеспечивающих читаемость страницы. Если у вас есть возможность тестирования в браузере Netscape 4,<sup>1</sup> таблицу

---

<sup>1</sup> Версию 4.8 данного браузера можно загрузить из архива [evolt.org](http://browsers.evolt.org/?navigator/32bit/4.8/) (<http://browsers.evolt.org/?navigator/32bit/4.8/>); она продолжает работать под Windows XP и Vista.

можно немного усложнить, не рискуя возникновением сбоев. В настоящее время такими старыми браузерами пользуются единицы. Отображение для них документа в базовом стиле – хороший результат, поскольку он по крайней мере обеспечивает им возможность прочитать содержимое сайта, чего, конечно, недостаточно для остальных пользователей.

*chapter07/basic\_basic.css*

```
body {
  background-color: #fff;
  color: #000;
  margin: 0;
  padding: 5%;
}

body, h1, h2, h3, h4, h5, h6, ol, ul, li, p {
  font-family: verdana, arial, helvetica, sans-serif;
  color: #000;
}
```

Помните, что современные браузеры прочтут обе таблицы стилей, поэтому в основной из них помимо добавления дополнительных свойств необходимо переопределить те свойства базовой таблицы стилей, которые вы хотели бы отображать по-другому в более новых браузерах.

Чтобы продемонстрировать данный подход на практике, я добавила несколько правил в следующий код; результат его обработки показан на рис. 7.2 и 7.3:

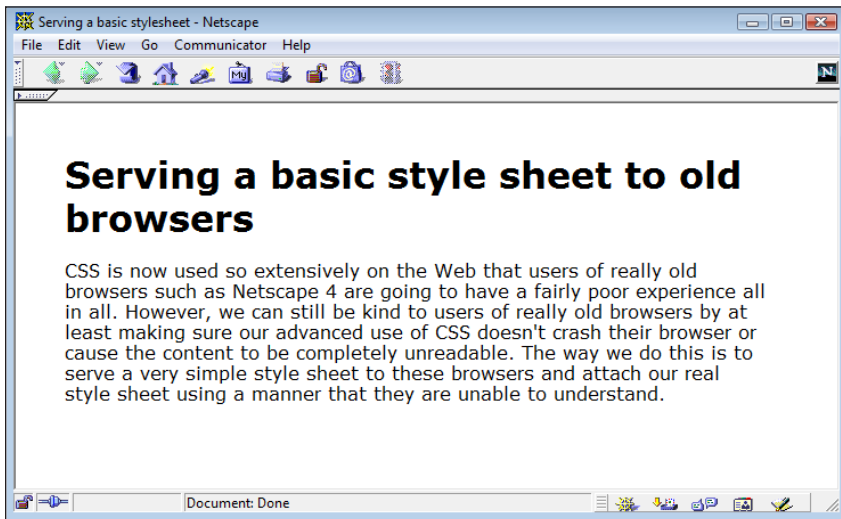


Рис. 7.2. Вид документа в Netscape 4.8

*chapter07/basic\_default.css*

```
h1 {
  color: #cc0022;
  margin: 0;
}

.content {
  background-color: #ececfc;
  padding: 0.6em;
}
```

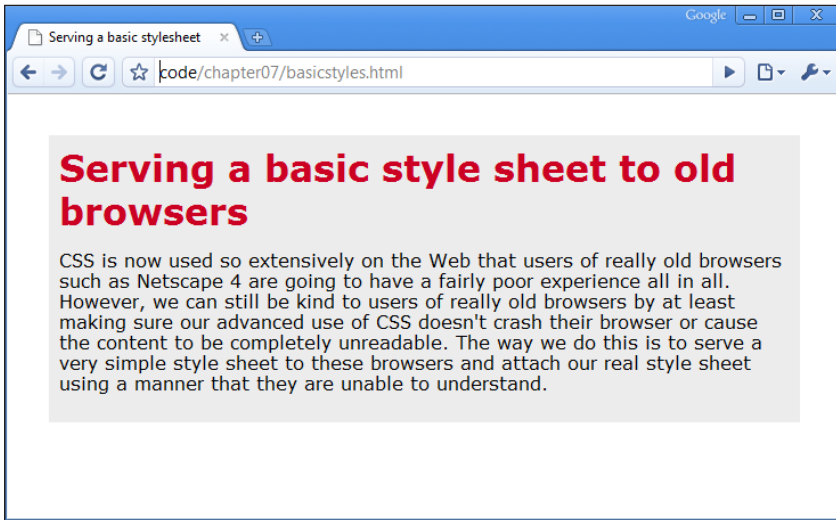


Рис. 7.3. Вид того же документа в Google Chrome

## Обсуждение

Бrowsers с минимальной поддержкой CSS доставляют больше всего хлопот, поскольку они понимают достаточно, чтобы предпринять попытку интерпретации кода, но недостаточно для того, чтобы сделать это правильно! Расширенные возможности CSS, применяемые при разработке среднего современного сайта, скорее всего будут отображены старым браузером искаженно или вообще не позволят открыть страницу. Скрытие стилей с помощью метода `@import` позволит избежать таких сбоев. На самом деле даже не обязательно добавлять базовую таблицу стилей – если вы присоединяете свою таблицу стилей с помощью `@import`, то в старых браузерах документ будет отображен в соответствии с их внутренней таблицей стилей.

Однако использование базовой таблицы стилей, связанной с документом при помощи ссылки, обладает дополнительным преимуществом:

это позволит избежать «мелькания неотформатированного контента»<sup>1</sup>. Этот досадный сбой состоит в том, что пользователи Internet Explorer на долю секунды видят неотформатированный документ (оформленный в соответствии с используемой по умолчанию таблицей стилей), пока ваша таблица стилей находится в процессе загрузки. Добавление ссылки перед выражением импорта – как в данном примере – позволяет решить описанную проблему. Таким образом, мы можем выразить свое лояльное отношение к оставшимся пользователям неуклюжих браузеров с помощью всего лишь одной уловки!

## Что такое режим совместимости и как его избежать

Вы разрабатываете сайт на XHTML и CSS, тестируете его в Internet Explorer, все, казалось бы, отлично, но... затем вы открываете тот же документ в Firefox и Safari и обнаруживаете, что все отображается совершенно по-другому. Что же происходит?

### Решение

Помимо сбоев в Internet Explorer причиной этого может послужить работа браузера в **режиме совместимости**. У многих современных браузеров есть два режима отображения. Режим совместимости отображает документ, как это сделали бы старые браузеры, такие как Netscape 4 и Internet Explorer 4 и 5, с соответствующими ошибками. **Стандартный режим** отображения обрабатывает документы в соответствии со спецификацией W3C (или максимально приближенно к ним).

- Плохо структурированные документы, содержащие устаревшую информацию в описании doctype (или вовсе не имеющие описания doctype), отображаются в режиме совместимости.
- Если в документе содержится что-либо перед выражением DOCTYPE, в том числе и пролог XML, необходимый для документов на XHTML, Internet Explorer 6 отобразит его в режиме совместимости.
- Документы, созданные с помощью строгого стандарта HTML 4 или XHTML (с соответствующей декларацией), отображаются в стандартном режиме.

Переключение между стандартным режимом и режимом совместимости в зависимости от описания типа документа называется **переключением режима отображения**. Все предельно просто: используйте описание типа документа, переводящее браузер в стандартный режим, и убедитесь, что элемент doctype идет первым по порядку, чтобы IE6 не на что было пожаловаться. Ниже приведен список описаний типа документа,

---

<sup>1</sup> <http://www.bluerobot.com/web/css/fouc.asp/>



использование которых осуществляет переключение браузеров Internet Explorer 6, Internet Explorer 5 Mac, Opera 7, Safari, Firefox и Chrome в стандартный режим.<sup>1</sup>

### HTML 4.01 Transitional

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

### HTML 4.01 Frameset

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

### HTML 4.01 Strict

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

### XHTML 1.0 Transitional

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

### XHTML 1.0 Frameset

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

### XHTML 1.0 Strict

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

### XHTML 1.1

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

Документы на XHTML должны содержать в начале пролог XML, например, такой:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
:
</html>
```

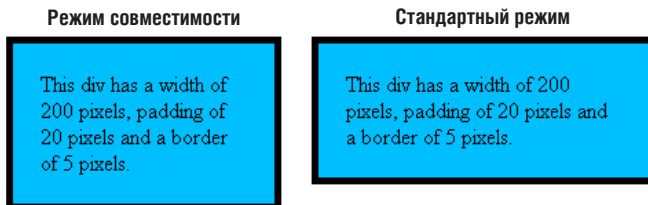
К сожалению, из-за него Internet Explorer 6 перейдет в режим совместимости, поэтому придется его убрать:

<sup>1</sup> Полный список возможного содержания doctype и информация о его влиянии на работу различных браузеров доступны по адресу <http://gutfeldt.ch/matthias/articles/doctypeswitch/table.html>.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
:
</html>
```

## Обсуждение

Самыми известными являются особенности режима совместимости **Internet Explorer**. При его включении браузер использует неверную блочную модель CSS, которая использовалась в **Internet Explorer 5** и **5.5**. Таким образом, вместо того чтобы добавить внутренний отступ и рамку к окошку путем прибавления к его ширине (200 пикселей) 20 пикселей с каждой стороны (итого 240 пикселей), браузер отображает окошко шириной в 200 пикселей и отнимает от этого значения величину отступов. На рис. 7.4 показаны различия между двумя режимами.



*Рис. 7.4. Отображение документа в режиме совместимости и в стандартном режиме в Internet Explorer*

При разработке нового сайта я рекомендую вам стремиться к соответствию требованиям стандартного режима, какое бы описание типа документа вы ни использовали. Новые браузеры, скорее всего, поддерживают стандарты W3C и выводят документ соответствующим образом вне зависимости от того, поддерживают ли они переключение режима отображения. В режиме совместимости браузеры ведут себя слишком уж странно, поэтому лучше с ним не связываться и оградить себя от проблем с самого начала.

## Задание разных таблиц стилей для Internet Explorer 6 и 7

На момент написания данной книги браузер **Internet Explorer 6** доставлял разработчикам немало головной боли, поскольку количество его пользователей все еще велико, а сам браузер крайне плохо поддерживает большую часть спецификации CSS, выдавая кучу ошибок. Microsoft исправила наиболее известные из них и реализовала поддержку спецификации CSS2.1 в **Internet Explorer 7**, однако и он во многом уступает современным браузерам, таким как **Internet Explorer 8**. Поэтому остается группа пользователей, не имеющих желания или возможности об-

новить свой браузер, и хотя среди пользователей **IE7** наблюдается тенденция к переходу к **IE8**, их количество пока остается значительным, и с этим нужно считаться.

## Решение

Самый эффективный метод связи определенных правил стиля с конкретными версиями **Internet Explorer** состоит в использовании условных комментариев. Условные комментарии – характерная особенность **Internet Explorer**; с их помощью можно выделить часть HTML-кода и определить, при каких условиях браузер должен его обработать, а при каких – проигнорировать. Условные комментарии можно использовать для того, чтобы только определенным версиям **IE** был доступен тег `<link>`.

Прежде всего необходимо создать таблицу стилей, содержащую правила для предотвращения ошибок в **Internet Explorer 6**, – **не нужно копировать** всю основную таблицу стилей, достаточно лишь переписать или добавить необходимые правила. Затем разместите внутри элемента `head` вашего документа ссылку на нее, окруженную условными комментариями, например:

```
<!--[if IE 6]>
<link rel="stylesheet" type="text/css" href="ie6fixes.css" />
<![endif]-->
```

Благодаря приведенному выше коду только **Internet Explorer 6** сможет «увидеть» таблицу стилей `ie6fixes.css`. В следующем коде таблица стилей будет доступна только **IE7**:

```
<!--[if IE 7]>
<link rel="stylesheet" type="text/css" href="ie7fixes.css" />
<![endif]-->
```

Следующий код объявляет таблицу стилей доступной для версий **Internet Explorer 7** и ниже:

```
<!--[if lte IE 7]>
<link rel="stylesheet" type="text/css" href="iefixes.css" />
<![endif]-->
```

Условные комментарии должны располагаться внутри элемента `head` вашего документа, причем после основной таблицы стилей, иначе правила, написанные специально для **IE**, будут переопределены правилами, содержащимися в основной таблице стилей.

Синтаксис такого условного комментария дает разработчику много дополнительных возможностей. Дополнительную информацию об условных комментариях можно получить в документации **SitePoint CSS**

Reference.<sup>1</sup> В последующих решениях мы рассмотрим способы использования условных комментариев для указания дополнительной таблицы стилей, а также файла со сценарием на JavaScript для Internet Explorer 6.

## Уход от наиболее распространенных ошибок в Internet Explorer 6 и 7

На текущий момент самыми «проблемными» браузерами остаются **Internet Explorer 6 (и в меньшей степени 7)**. Используя накопившиеся методические наработки, вы сможете обеспечить надежную работу своих сайтов в этих браузерах, не отказываясь при этом от создания сложных макетов страниц, позволяющих получить все преимущества поддержки технологии CSS последними версиями браузеров.

### Решение

Приведу свои соображения по поводу различных методов работы с CSS и советы по обеспечению нормального отображения сайтов **Internet Explorer 6 и 7**.

#### Процесс разработки

Следующая схема работы помогает мне избежать большинства возможных ошибок **IE6 и 7**.

#### Работайте с самым современным браузером

Изначально следует разрабатывать сайт с расчетом на современные браузеры, хорошо поддерживающие спецификацию CSS, например, последние версии **Firefox, Opera и Safari**. **Поддержка браузерами стандартов CSS становится все более полной**, а вовсе не наоборот, поэтому написанный код также должен соответствовать спецификации, поскольку в этом случае он будет корректно отображаться будущими браузерами. В процессе разработки сайта я никогда не открываю его в Internet Explorer, поскольку в этом случае может быть трудно удержаться от написания дополнительного кода для исправления найденных ошибок. Для начала нужно убедиться, что сайт нормально работает в браузерах с хорошей поддержкой CSS, а уже потом заниматься поправками для остальных браузеров.

#### Проведите валидацию HTML- и CSS-кода

Если ваша разметка соответствует стандартам W3C, то возникающие в старых браузерах ошибки будет легче исправить. Браузеры по-разному

---

<sup>1</sup> <http://reference.sitepoint.com/css/conditionalcomments/>

обрабатывают код, в который закрались ошибки, и потому незакрытый тег `<div>` или пропущенная точка с запятой в правиле стиля может привести к досадным сбоям в ходе дальнейшей разработки. Поэтому перед тем как продолжить, обязательно воспользуйтесь сервисом валидации CSS-кода и разметки документа W3C<sup>1</sup> и проверьте свой документ на наличие ошибок.

### **Протестируйте сайт в других современных браузерах**

Затем посмотрите, как сайт отображается в других современных браузерах: **Firefox, Opera, Safari, Chrome и Internet Explorer 8**. Мне довольно редко приходится вносить в код какие-либо исправления для этих браузеров, и они, как правило, незначительны и касаются реализации отдельных элементов структуры документа. Для современных браузеров я никогда не использую хитроумные приемы или альтернативные таблицы стилей, поскольку всегда можно найти более простой способ решения проблемы.

### **Проверьте сайт в Internet Explorer 6 и 7**

Если ваш CSS-код прошел валидацию и корректно обрабатывается большинством современных браузеров, то пришло время исправлять ошибки его интерпретации старыми браузерами. Я твердо придерживаюсь мнения, что не стоит нагромождать свой код дополнениями, предназначенными для устаревших браузеров.

Скорее всего, при просмотре сайта в **IE6 вы обнаружите какие-либо проблемы** – возможно, небольшие несоответствия вроде лишних отступов между элементами страницы или более серьезные недочеты – например, когда определенные части вашей страницы «уезжают» из предназначенной для них области или вовсе исчезают. В такой ситуации главное – сохранять спокойствие. Большинства сбоев IE6 можно с легкостью избежать, изменив или добавив несколько правил стилей специально для этого браузера.

То же самое можно сказать и о IE7, с тем лишь исключением, что этот браузер работает более корректно.

### **Добавьте таблицы стилей, предназначенные для конкретных браузеров, с помощью условных комментариев**

На данном этапе нужно добавить дополнительные таблицы стилей для **IE6 и/или IE7 с помощью условных комментариев**. Этот метод был рассмотрен в предыдущем разделе. Такую таблицу стилей нужно добавить в заголовок (`head`) вашего документа после уже указанных в его HTML-коде таблиц стилей, чтобы содержащиеся в них правила, написанные специально для Internet Explorer 6 и 7, переопределили аналогичные правила в основной таблице стилей.

---

<sup>1</sup> <http://validator.w3.org/>

## Исправление ошибок, возникающих в Internet Explorer

Теперь можно постепенно начать исправление возникающих в Internet Explorer 6 и 7 ошибок, добавляя необходимые правила в альтернативную таблицу стилей, будучи в полной уверенности, что они будут использованы только теми браузерами, для правильной работы которых они необходимы. Использование рассматриваемых ниже приемов позволяет избежать большинства проблем Internet Explorer 6. Для исправления сбоя Internet Explorer 7 обычно необходима лишь часть из них. В IE7 редко встречаются «баги», отсутствующие в IE6, а алгоритм их исправления, как правило, один и тот же.

### Проверьте DOCTYPE

Убедитесь, что в первой строчке вашего документа указано правильное описание типа документа. О важности использования правильного элемента `doctype` говорилось в предшествующем разделе «Что такое режим совместимости и как его избежать?». Указание неправильного типа документа может привести к тому, что ваши документы будут выглядеть довольно странно. Поэтому прежде чем начать что-то предпринимать для исправления ошибок, убедитесь, что все браузеры, в том числе IE6 и 7, отображают страницу в стандартном режиме.

### Отсутствие поддержки свойства `min-height` в IE6

Internet Explorer не поддерживает свойство `min-height`, задающее минимальную высоту элемента, а вместо этого интерпретирует значение свойства `height` в этом смысле. Таким образом, в других браузерах значение свойства `height` определяет фиксированную высоту элемента, а Internet Explorer 6 воспринимает его в качестве значения минимальной высоты, поэтому размер блока по вертикали может увеличиваться и превысить задуманные для него параметры.

Чтобы избежать такой ситуации, достаточно использовать свойство `height` в таблице стилей для IE6 в том же качестве, что и свойство `min-height` в основной таблице стилей.

### Свойство `hasLayout`

В движке IE6 и 7 есть таинственный компонент – свойство `hasLayout`, причина большого количества самых причудливых «багов». Существуют элементы, ответственные за размер и расположение содержащегося в них контента; считается, что они обладают этим свойством. Прочие элементы определяют свой размер и расположение за счет родительского элемента или более отдаленного предка. Если элемент не определяет структуру своего содержимого, это может привести к странным последствиям – например, исчезновению части контента или неправильному отображению структуры документа. Некоторые элементы, среди которых ячейки таблицы, по умолчанию обладают свойством `hasLayout`, чего, однако, нельзя сказать об элементах `div`. Если элементу задать такие CSS-свойства, как `float` со значением `left` или `right`, он также начи-

нает определять структуру документа, что приводит к решению большинства возникающих проблем. Самое главное – найти CSS-свойство, которое изменит статус элемента, но не повредит структуре документа.

Для IE6 присвоить элементу свойство `hasLayout` можно, задав значение 1% его свойству `height`. Как уже было сказано выше, IE6 воспринимает свойство `height` как `min-height`, поэтому такое значение определит минимальную высоту элемента, что не окажет никакого влияния на структуру документа, а сам элемент по-прежнему сможет вместить весь необходимый контент. Конечно, такие правила нужно указывать в таблице стилей, предназначенной специально для IE6.

IE7, напротив, совершенно верно обрабатывает свойство `height`, поэтому такой трюк в данном случае не сработает. Однако вместо этого можно задать любое, даже нулевое значение свойству `min-height`, что также изменит статус элемента в IE7. Это вполне надежный способ, поскольку значением свойства `min-height` по умолчанию также является 0.

Не всегда удастся сразу определить, какой элемент требует добавления такого «триггера» изменения статуса, но обычно виновника всех беспорядков на странице можно вычислить поступательно. Вначале я рассматриваю элементы с самой высокой степенью вложенности, т. е., если несколько блоков `div` вложены друг в друга, то я добавляю значение высоты для последнего из них, обновляю страницу и смотрю, привело ли это к каким-то изменениям. Если все осталось по-прежнему, я перехожу к `div` на уровень выше и т. д.

### Относительное позиционирование элемента

Если изменение статуса элемента другими способами не дает желаемого результата, можно попробовать присвоить элементу свойство `position` со значением `relative`. Обратите внимание, что после этого все его дочерние элементы будут располагаться относительно него. Если такой прием не нарушает дизайн сайта, вполне можно им воспользоваться.

### Прочие ошибки

Приведенные выше приемы помогут вам решить самые серьезные проблемы, однако вы можете столкнуться и с менее значительными недостатками интерпретации, такими как неправильно рассчитанные отступы или незначительное смещение элементов. Самое время вспомнить о том, что вы имеете дело со старыми браузерами, выдающими кучу ошибок, поэтому не нужно бояться изменять правила стилей в специально предназначенных для них таблицах стилей, регулируя величину отступов или перемещая элементы до тех пор, пока не будет достигнут желаемый результат. Использование условных комментариев гарантирует отсутствие каких-либо сбоев при просмотре сайта в других браузерах. Надеюсь, вам не придется делать слишком много дополнительной работы такого рода, поскольку при изменении структуры страницы, скорее всего, придется все создавать заново; иногда придется буквально по кусочкам собирать мозаику сложной страницы!

## Достижение прозрачности изображения в формате PNG в Internet Explorer 6

Одним из самых потрясающих дополнений версии 7 браузера Internet Explorer стала поддержка прозрачности изображений PNG. Как уже было сказано в главе 3 при обсуждении фоновых изображений, с помощью изображений в формате PNG действительно можно достичь прозрачности: их можно отображать поверх фона различных цветов без пиксельного гало-эффекта и использовать для создания визуальных эффектов с применением непрозрачных фоновых слоев. Однако если вы просто разместите такие изображения на странице, пользователи Internet Explorer 6 увидят непрозрачные картинки, как показано на рис. 7.5. Есть ли какой-нибудь способ обойти этот досадный недочет и подружить IE6 с PNG?

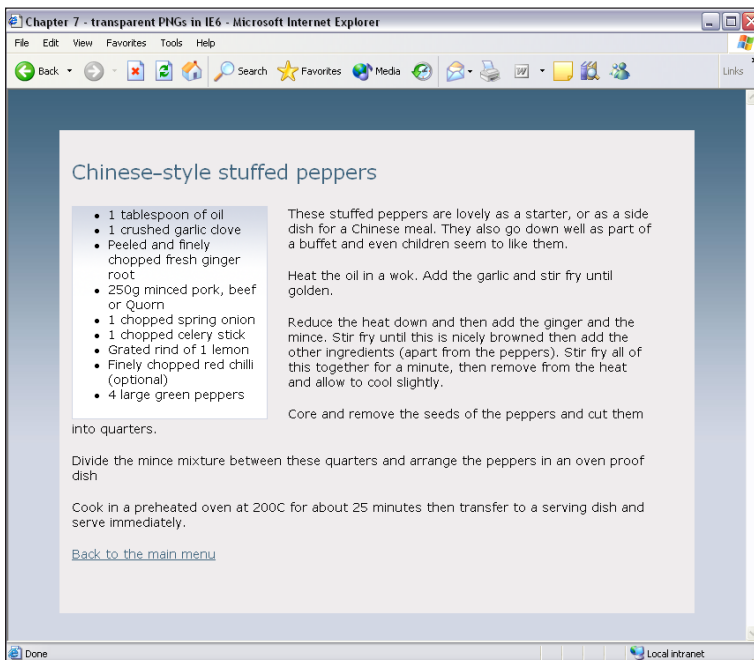


Рис. 7.5. Отображение прозрачных изображений PNG в Internet Explorer 6

### Решение

Чтобы изображения в формате PNG отображались в Internet Explorer 6 прозрачными, придется использовать небольшой сценарий на JavaScript. Он был создан Аароном Будманом (Aaron Budman)<sup>1</sup> и в дальней-

<sup>1</sup> <http://webapp.youngpup.net/?request=/snippets/sleight.xml>



шем доработан Дрю МакЛелланом (Drew McLellan) с тем, чтобы обеспечить поддержку фоновых изображений.<sup>1</sup>

Прежде всего создадим прозрачный GIF размером 1×1 px и сохраним его под именем `x.gif`.

Затем создадим новый файл сценария на JavaScript (который будет предназначен только для Internet Explorer 6) и добавим в него следующий код:

*chapter07/bgsleight.js*

```
function addLoadEvent(func) {
    var oldonload = window.onload;
    if (typeof window.onload != 'function') {
        window.onload = func;
    } else {
        window.onload = function() {
            if (oldonload) {
                oldonload();
            }
            func();
        }
    }
}

var bgsleight = function() {

    function fnLoadPngs() {
        var rslt = navigator.appVersion.match(/MSIE (\d+\.\d+)/, '');
        var itsAllGood = (rslt != null && Number(rslt[1]) >= 5.5);
        for (var i = document.all.length - 1, obj = null; (obj =
            document.all[i]); i--) {
            if (itsAllGood &&
                obj.currentStyle.backgroundImage.match(/\.png/i) != null) {
                fnFixPng(obj);
                obj.attachEvent("onpropertychange", fnPropertyChanged);
            }
            if ((obj.tagName=='A' || obj.tagName=='INPUT') &&
                obj.style.position == ''){
                obj.style.position = 'relative';
            }
        }
    }

    function fnPropertyChanged() {
        if (window.event.propertyName == "style.backgroundImage") {
            var el = window.event.srcElement;
            if (!el.currentStyle.backgroundImage.match(/x\.gif/i)) {
```

---

<sup>1</sup> <http://allinthehead.com/retro/289/sleight-update-alpha-png-backgrounds-in-ie>

```
        var bg = el.currentStyle.backgroundImage;
        var src = bg.substring(5,bg.length-2);
        el.filters.item(0).src = src;
        el.style.backgroundImage = "url(/img/shim.gif)";
    }
}
}

function fnFixPng(obj) {
    var mode = 'scale';
    var bg = obj.currentStyle.backgroundImage;
    var src = bg.substring(5,bg.length-2);
    if (obj.currentStyle.backgroundRepeat == 'no-repeat') mode =
'crop';
    obj.style.filter =
"progid:DXImageTransform.Microsoft.AlphaImageLoader(src='
+ src + '", sizingMethod=' + mode + "')";
    obj.style.backgroundImage = "url(/img/shim.gif)";
}
return {
init: function() {
    if (navigator.platform == "Win32" && navigator.appName ==
"Microsoft Internet Explorer" && window.attachEvent) {
addLoadEvent(fnLoadPngs);
    }
}
}
}
}();

bgsleight.init();
```

Чтобы данный сценарий выполнялся только в Internet Explorer 6, его нужно добавить с помощью условных комментариев:

*chapter07/bgsleight.html (фрагмент)*

```
<!--[if IE 6]>
<script type="text/javascript" src="bgsleight.js"></script>
<![endif]-->
```

Если вы сохраните страницу на данном этапе и откроете ее в Internet Explorer, окажется, что фоновое изображение элемента `div` с идентификатором `content` исчезло. Его можно вернуть, задав ему свойство `height`. Достаточно значения 1%, поскольку Internet Explorer 6 интерпретирует его как минимальный размер элемента и будет растягивать блок до тех пор, пока он не сможет вместить все вложенные элементы. Поскольку это правило стиля предназначено только для Internet Explorer, его нужно разместить либо внутри элемента `style` в заголовке `head` вашего документа, либо добавить в созданную специально для этого броузера

таблицу стилей, привязанную к документу с помощью условных комментариев:

*chapter07/bgsleight.html (фрагмент)*

```
<!--[if IE 6]>
<style type="text/css">
#content {
  height: 1%;
}
</style>
<script type="text/javascript" src="bgsleight.js"></script>
<![endif]-->
```

Обновите страницу в Internet Explorer, и вы увидите, что поверх фона будет отображаться прозрачная заливка, как показано на рис. 7.6.

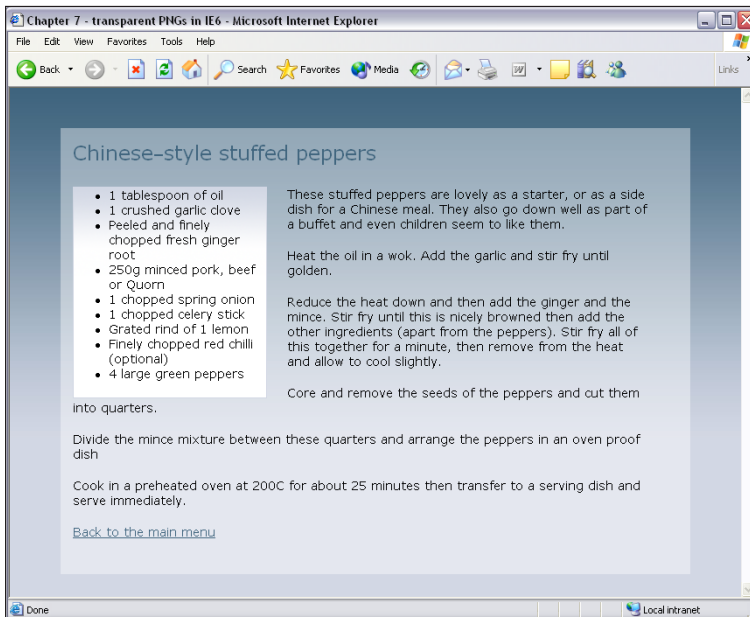


Рис. 7.6. Отображение прозрачных изображений PNG в Internet Explorer 6

## Обсуждение

Использование описанного приема может вызвать массу других неприятностей. Например, может возникнуть впечатление, будто некоторые области страницы покрыты фоновым изображением, делая ссылки недоступными для перехода, а поля для ввода текста – не принимающими фокус. Эту проблему чаще всего можно решить, добавив элементу опи-

сание `position:relative;`, однако это также усложняет нашу задачу. Тем не менее благодаря возможности использования полностью прозрачных изображений можно добиться большей гибкости дизайна, а если подходить к делу тщательно и аккуратно, все будет отлично работать.

### Совет

**Без хитростей.** Существует и другой способ решения данной проблемы: можно использовать отдельные изображения, созданные специально для Internet Explorer 6 путем указания ссылки на них в соответствующей таблице стилей, чтобы при просмотре сайта в этом браузере они отображались вместо PNG. При этом сайт будет выглядеть немного по-другому, но, учитывая, что Internet Explorer 6 отстает от современности уже на две версии и постепенно теряет своих пользователей, такое решение вполне приемлемо. В настоящее время мы часто комбинируем эти методы: где это представляется возможным, мы используем GIF, а если это слишком сложно в данной ситуации, применяем сценарий на JavaScript.

## Корректное отображение в IE 8 сайта, соответствующего стандартам W3C

Как уже было сказано ранее, **Internet Explorer 8 может отображать сайты** так же, как это сделала бы его предыдущая версия, в том числе и присваивая части из них свойство `hasLayout` и вызывая множество несоответствий, с которыми дизайнеры боролись на протяжении многих лет. Как удостовериться, что IE8 использует самый современный режим отображения, а не имитирует поведение IE7, при отображении вашего веб-сайта?

### Решение

Internet Explorer 8 – очень мощный браузер, и вам наверняка хотелось бы, чтобы он не жалел сил и отображал разрабатываемый вами новый сайт как можно лучше. Рассматривая результат обработки CSS-кода различными браузерами, я пришла к выводу, что в настоящее время можно встретить не так уж много расхождений между IE8, Firefox 3 или Safari 3 или 4. Как обычно, по умолчанию Internet Explorer 8 будет использовать переключение режимов отображения, чтобы решить, как нужно отображать страницу – в стандартном режиме или в режиме совместимости. Для поддержки обратной совместимости и в то же время пытаясь «не порвать Всемирную паутину», Microsoft ввела **вид в режиме совместимости (Compatibility View)** и головную метку `x-ua-compatible`, дающую браузеру указания по поводу того, каким образом ему следует отображать страницы – максимально соответствуя стандартам или имитируя способ обработки кода браузером Internet Explorer 7.

**Вид в режиме совместимости** можно включить в браузере, а метка `x-ua-compatible` указывается в коде документа с помощью тега `<meta />` или в заголовке HTTP, пересылаемом с сервера. Пользователь может включить отображение в режиме совместимости, нажав на кнопку `Compatibility View`; кроме того, все сайты, включенные в список `Compatibility View List` браузера Internet Explorer 8 для Windows, также будут отображаться в режиме совместимости. Данный список составлен Microsoft, и пользователи IE8 могут оформить подписку на него.<sup>1</sup>

Метка `x-ua-compatible` – это указание для браузера, переопределяющее все текущие настройки совместимости. Ниже приведен пример тега `meta`:

```
<meta http-equiv="X-UA-Compatible" content="IE=8" />
```

Его использование обеспечит отображение страницы в Internet Explorer 8 в режиме наибольшего соответствия стандартам. Чтобы при обработке документа браузер имитировал поведение Internet Explorer 7, нужно задать значение атрибута `IE=EmulateIE7`. При указании значения `IE=Edge` Internet Explorer 8 и ниже всегда будет использовать для отображения сайтов режим наибольшего соответствия стандартам.

Если вы занимаетесь разработкой сайта с нуля, я рекомендую вам обойтись без заголовка `x-ua-compatible` или выставить значение `IE=Edge`. При использовании правильного элемента `doctype` **Internet Explorer** постарается отобразить сайт как можно лучше безо всякого заголовка `x-ua-compatible` – вы ведь пишете код в соответствии со стандартами и совершенно не планируете попасть в тот злополучный список Microsoft? В противном случае – при использовании `x-ua-compatible` – рекомендую вам использовать именно значение `IE=Edge` (а не `IE=IE8` и др.). Тогда при выходе 9-й версии **Internet Explorer** ваш сайт не застынет навечно в виде для IE8.

Более подробную информацию по данной теме можно найти в замечательной статье, опубликованной Фаруком Эйтсом (Faruk Ates) в своем блоге. Она называется «IE8 and the X-UA-Compatible situation» (IE8 и ситуация с переключением режимов с помощью заголовка X-UA-Compatible).<sup>2</sup>

## Что делать, если CSS не работает

Все мы время от времени сталкиваемся с ситуациями, когда CSS-код не работает, и все тут! Вы перебрали все возможные решения, но от-

---

<sup>1</sup> Список таких сайтов можно загрузить с сайта Microsoft по адресу <http://www.microsoft.com/downloads/details.aspx?FamilyID=b885e621-91b7-432d-8175-a745b87d2588&displaylang=en>

<sup>2</sup> <http://farukat.es/journal/2009/05/245-ie8-and-the-x-ua-compatible-situation>

дельные кусочки текста продолжают исчезать и появляться в Internet Explorer 6 или же часть его перекрывает остальной контент в Safari. Прежде чем начать посыпать голову пеплом, сделайте глубокий вздох и успокойтесь. Решение есть!

## Решение

Это решение поможет вам найти решение!

### 1. Передохните

Когда все попытки исправления ошибки безуспешны, трудно сохранять ясность разума в поисках оптимального решения. Поэтому отдохните немного: прогуляйтесь, сделайте уборку на рабочем столе или займитесь домашними делами. Если ваш начальник рядом, и вам неудобно просто встать и отойти в сторону, переключитесь на выполнение другой задачи: ответьте на письма или удалите ненужные файлы. Что угодно, главное – отвлекаться от проблемы на некоторое время.

### 2. Проведите валидацию кода документа и таблицы стилей

Затем нужно проверить HTML- и CSS-код, если вы до сих пор еще этого не сделали. Возможно, проблема возникла из-за допущенных в коде ошибок, а если даже причина в другом, их наличие только все усугубляет.

### 3. Изолируйте проблему

Может ли ваша ошибка возникнуть независимо от остальных частей документа? Сбои часто возникают только при наличии определенного набора условий, поэтому выявление источника проблемы, возможно, поможет вам в поиске решения. Попробуйте перенести проблемный фрагмент кода в документ с другой разметкой.

### 4. Поищите решение в Интернете

Если ваша проблема связана с использованием определенного браузера, то она, скорее всего, возникла не впервые. Существует множество хороших сайтов, посвященных такого рода проблемам и способам их решения. Столкнувшись с проблемами в работе, я обязательно обращаюсь к следующим сайтам:

- CSS Pointers Group, <http://css.nu/pointers/bugs.html>
- Position is Everything, <http://www.positioniseverything.net/>
- Раздел Browser Bug Category на сайте css-d wiki, <http://css-discuss.incutio.com/?page=CategoryBrowserBug>

SitePoint CSS Reference<sup>1</sup> содержит немало полезной информации о поддержке различными браузерами тех или иных свойств и селек-

---

<sup>1</sup> <http://reference.sitepoint.com/css/>

торов CSS. Кроме того, попробуйте поискать решение в архивах `css-discuss`<sup>1</sup> и, конечно, воспользуйтесь поиском Google!

### 5. Обратитесь за советом к другим

Если приведенные выше советы не привели к решению проблемы, попросите помощи у других людей. Даже самые опытные разработчики порой сталкиваются с проблемами, которые не могут решить в одиночку. Иногда свежий взгляд посторонних людей может помочь найти возможные способы решения проблемы, о которых вы не подумали, или даже найти готовое решение.

При написании сообщения на форуме или в почтовой рассылке не забывайте о следующих правилах:

- Вначале следует провести поиск по архиву, если он есть, – возможно, аналогичные вопросы задают по несколько раз в день.
- Убедитесь, что ваш HTML- и CSS-код прошел валидацию, иначе вы рискуете получить ответ «проверьте ваш код на соответствие спецификации, это может помочь».
- Выложите пример вашей работы в Сети с тем, чтобы указать ссылку на него на форуме. Возможность рассмотрения проблемы независимо от сложной структуры всей страницы является дополнительным плюсом, поскольку другим будет проще разобраться в происходящем.
- Расскажите об испробованных вами способах решения проблемы, чтобы посетители форума не теряли времени, предлагая уже проверенные методы. Кроме того, это укажет на то, что вы пытались решить проблему самостоятельно перед тем, как обратиться за помощью.
- Тема сообщения должна быть информативной. Люди скорее прочитают сообщение с заголовком «Дублирование блоков в IE5», чем отреагируют на отчаянный возглас «ПОМОГИТЕ!». Правильные заголовки повышают информативность архива: достаточно взглянуть на список, чтобы получить представление об обсуждаемых темах.
- Будьте вежливы и придерживайтесь темы.
- Терпеливо ждите, пока вам ответят. Если рассылка или форум достаточно активны, но вы не получили ни одного ответа в течение дня, вполне допустимо задать вопрос снова, добавив в тему префикс REPOST. При большом объеме поступающих тем часть из них может оставаться незамеченной, и повторная публикация ненавязчиво напомнит читателям, что ваша проблема осталась нерешенной.

---

<sup>1</sup> <http://www.css-discuss.org/>

- Получив предложения по решению проблемы, примените их на практике. Не расстраивайтесь и не раздражайтесь, если они не приводят к желаемому результату или если вам кажется, что вам предлагают самые очевидные вещи. Я видела достаточно долгие дискуссии, продолжавшиеся до тех пор, пока решение не было найдено. Дайте людям возможность помочь вам!
- Если вы нашли выход из положения или решили изменить дизайн страницы, чтобы избежать возникновения проблемы, сообщите об этом в своей теме, описав все подробности, в знак уважения к тем, кто попытался вам помочь. Кроме того, это может сослужить хорошую службу тем, кто в дальнейшем будет просматривать архивы в поисках решения аналогичной проблемы. Очень обидно найти несколько предлагаемых решений проблемы, не зная, какое из них действительно эффективно (и есть ли среди них эффективные методы в принципе).

Среди подписчиков рассылок, посвященных веб-дизайну, много очень опытных разработчиков, прекрасно разбирающихся в CSS. На мой взгляд, лучшей рассылкой, посвященной CSS, является *css-discuss*.<sup>1</sup> Ее подписчики достаточно активны и дружелюбны, и вы можете почерпнуть массу полезной информации, просто читая сообщения и просматривая архивы. На сайте SitePoint также есть замечательный и очень активный форум,<sup>2</sup> на котором общается много отзывчивых и опытных людей.

## Интерпретация сообщений, выводимых инструментом W3C Validator

Валидация кода документа крайне важна для того, чтобы он корректно отображался в браузерах, поддерживающих стандарты W3C. Однако иногда выводимые в ходе этого процесса сообщения об ошибках могут сбить с толку.

### Решение

(X)HTML-документы можно проверить в режиме онлайн с помощью инструмента W3C Validator;<sup>3</sup> CSS-код можно проверить с помощью W3C CSS Validator.<sup>4</sup> Многие программы для разработки, такие как Dreamweaver,

---

<sup>1</sup> <http://www.css-discuss.org/>

<sup>2</sup> <http://www.sitepoint.com/launch/cssforum/>

<sup>3</sup> <http://validator.w3.org/>

<sup>4</sup> <http://jigsaw.w3.org/css-validator/>



имеют встроенные валидаторы, а на некоторые браузеры, например, Firefox, можно установить расширение для проверки документов.<sup>1</sup>

Проверка как HTML-, так и CSS-кода начинается от начала к концу документа. Иногда по окончании процесса валидации можно получить длинный список ошибок. Однако после исправления первой из них большинство последующих также могут исчезнуть. Это особенно характерно для проверки (X)HTML-документов. Если вы забыли закрыть тег, валидатор считает его открытым и выдает кучу ошибок, говорящих о том, что элемент А не может располагаться внутри элемента В. Закрыв тег, вы увидите, что все эти ошибки будут исправлены автоматически.

Похожая ситуация может возникнуть при проверке документов типа HTML (не XHTML), в которых разработчик закрыл тег в соответствии с правилами синтаксиса XML, например:

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

Если вы используете такой код в документе, в элементе `doctype` которого не указано, что он написан на XHTML, будет выведено сообщение о том, что закрывающий тег `</head>` расположен в неподходящем месте. Чтобы документ соответствовал стандартам HTML, достаточно удалить косую черту из тега:

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

## Ошибки и предупреждения

Если в CSS-документе содержатся синтаксические и прочие ошибки (например, пропущена точка с запятой), он не пройдет валидацию. Необходимо исправить все найденные ошибки и убедиться в том, что при обработке кода страница отображается именно так, как было задумано.

Если CSS-код не содержит ошибок, он пройдет валидацию. Однако при его проверке валидатор может вывести различные предупреждения. Вы можете принять их к сведению или проигнорировать – на ваше усмотрение. Самое распространенное предупреждение относится к неподобающему фону для определенного элемента, из-за чего часть текста на странице может оказаться непригодной для чтения. Кроме того, валидатор может указать на то, что определенный фрагмент кода может вызвать проблемы, даже если вы намеренно написали его именно таким образом (например, если вы хотите, чтобы фон одного элемента просвечивал сквозь другой элемент). Предупреждения следует воспринимать скорее как совет и напоминание о необходимости проверки кода, однако если помимо предупреждений никаких ошибок не выявлено, таблица стилей полностью прошла валидацию!

---

<sup>1</sup> <http://users.skynet.be/mgueury/mozilla/>

## Заклучение

В данной главе были рассмотрены решения проблем, с которыми вам, возможно, еще только предстоит столкнуться, особенно в том случае, если до сих пор вы создавали сайты, позиционируя элементы с помощью таблиц, а не CSS. Именно в таких условиях рождаются самые изощренные броузерные ошибки, и тестирование сайта в различных версиях различных броузеров приобретает еще более важное значение.

В данной главе я рассказала о своих методах тестирования сайтов, выявления ошибок и получения помощи в их исправлении. Кроме того, я попыталась представить вам дополнительные возможности, чтобы ваш сайт отображался подходящим образом для различных категорий пользователей. Если вы читаете книгу последовательно, глава за главой, то вскоре увидите, что все вышесказанное обретает еще большее значение в свете информации об использовании CSS для верстки страницы, представленной в главе 9.

# 8

## Доступность и альтернативные устройства

CSS позволяет отделить структуру и содержание документа от его оформления. Следование такому принципу дает возможность доступа к его контенту пользователям устройств, не воспроизводящих дизайн сайта. Причиной таких ограничений могут быть технические особенности, как в случае определенных браузеров мобильных устройств, или конкретная функциональность, как у экранных дикторов, воспроизводящих текст для пользователей с нарушениями зрения. При этом у нас остается возможность создания красивых и привлекательных с визуальной точки зрения сайтов для большинства пользователей, чьи браузеры поддерживают CSS.

Определенным пользователям с ограниченными возможностями вы облегчите доступ к сайтам, созданным с учетом вышесказанных соображений, однако следует подумать и о тех посетителях, которые могут увидеть дизайн сайта, но которым требуются дополнительные инструменты для работы с ним. Чтобы различным категориям пользователей было удобно пользоваться вашим сайтом, недостаточно просто сверстать макет страницы с использованием CSS. Например, некоторые пользователи со слабым зрением смогут прочитать только четкий текст, размер шрифта которого можно увеличить. Также в настоящей главе будут рассмотрены вопросы использования альтернативных таблиц стилей, таблиц стилей, предназначенных для определенной среды отображения (например, для печати) и написания CSS-кода для мобильных устройств.

### Аспекты доступности, о которых следует помнить при использовании CSS

Чтобы ваш сайт можно было с уверенностью назвать доступным для всех категорий пользователей, недостаточно только позаботиться об

удобстве посетителей, использующих экранные дикторы. При разработке дизайна сайта необходимо учитывать потребности самых различных пользователей: людей с нарушениями зрения, страдающих дислексией и тех, кто в силу физических ограничений не может пользоваться мышью. Такие пользователи, скорее всего, будут просматривать сайт в обычном веб-браузере, и только от вас зависит, насколько удобно им будет работать.

## Решение

Ниже представлен достаточно сжатый список полезных приемов, часть из которых уже обсуждалась в предыдущих главах книги.

### **Наряду с фоновым изображением задайте простой цвет фона**

Если текст в вашем документе отображается поверх фонового изображения – заданного, например, для ячейки таблицы или блочного элемента – не забудьте указать также и фоновый цвет. Использование фонового цвета обеспечит сохранение уровня контрастности текста в том случае, если изображение по тем или иным причинам не загрузится.

### **Задав цвет фона, задайте цвет текста и наоборот**

Чтобы читать документ было удобнее, необходимо продумать оптимальное сочетание цвета фона и текста для обеспечения должного уровня контрастности. Если вы зададите, скажем, только цвет фона, есть вероятность того, что он сольется с текстом, отображаемым в соответствии с настройками по умолчанию пользователя. Например, в настройках пользователя значится, что фон должен быть черным, а текст – белым, а вы задаете черный цвет текста. В этом случае текст просто-напросто исчезнет! Если вы хотите дать пользователю возможность самостоятельно определять настройки цвета, не следует задавать никаких цветов вообще, однако не так уж много веб-дизайнеров согласятся на такие условия!

### **Проверьте контрастность цвета**

Убедитесь, что текст выглядит достаточно контрастно на выбранном фоне. Пользователям со слабым зрением будет крайне сложно прочесть текст, если уровень его контрастности по отношению к фону невелик. Кроме того, учтите, что пользователи, страдающие дальтонизмом, могут не различать определенные сочетания цвета фона и текста. По рекомендации консорциума W3C в документации «Критерии доступности веб-контента» (WCAG 2.0 Success Criterion 1.4.3<sup>1</sup>) указано, что контрастность между текстом и изображениями должна поддерживаться в соотношении по меньшей мере 4,5:1. Для оценки уровня контрастности между выбранными вами цветами можно воспользоваться

---

<sup>1</sup> <http://www.w3.org/TR/2008/REC-WCAG20-20081211/#visual-audio-contrast-contrast>

очень удобной программой Luminosity Contrast Ratio Analyser, написанной Джемом Лемоном (Jez Lemon)<sup>1</sup>.

### **Фоновые изображения следует использовать исключительно в декоративных целях**

Добавлять фоновые изображения с помощью CSS настолько просто, что можно незаметно для себя приобрести пагубную привычку их повсеместного использования – и там, где надо, и там, где совсем не надо. Однако следует помнить, что пользователи, отключившие просмотр изображений и/или CSS, вообще не узнают о его существовании. В этом нет ничего страшного, если изображение выполняет только декоративную функцию, однако если оно несет информативную нагрузку, лучше сопроводить его описательным атрибутом alt. Тогда пользователи, не имеющие возможности его увидеть, смогут узнать, что оно существует, и прочитать его описание.

### **Используйте свойство line-height, чтобы текст было легче читать**

Благодаря увеличению высоты строки с помощью свойства line-height воспринимать текст будет еще легче, однако избегайте слишком больших отступов между строками, поскольку это может привести к обратному эффекту. Оптимальное значение данного свойства составляет от 1,2 до 1,6, причем если не указывать единицы, то высота строки будет изменяться пропорционально размеру шрифта.

## **Тестирование сайта в текстовом броузере**

Тестирование сайта в текстовом броузере – отличная возможность оценить, насколько он на самом деле удобен в использовании для различных категорий пользователей. Если в процессе навигации по сайту с помощью текстового броузера у вас не возникнет никаких затруднений, то, скорее всего, пользователи экранных дикторов также не столкнутся с какими-либо проблемами.

### **Решение**

Вы можете посмотреть, как будет выглядеть ваш сайт в текстовом броузере Lynx, причем это можно сделать прямо в режиме онлайн с помощью Lynx Viewer.<sup>2</sup> Помимо того что Lynx – удобный инструмент тестирования, его загрузка и установка бесплатна; поэтому вы можете установить себе копию Lynx. Это даст вам дополнительную возможность тестирования создаваемых страниц в процессе разработки, перед тем как загрузить их на сервер.

---

<sup>1</sup> <http://juicystudio.com/services/luminositycontrastratio.php>

<sup>2</sup> <http://www.delorie.com/web/lynxview.html>

## Пользователи Linux/UNIX

Вполне возможно, что Lynx уже установлен в вашей системе; в противном случае его можно с легкостью добавить через систему управления пакетами. Кроме того, его можно загрузить с сайта распространения программного обеспечения Lynx.<sup>1</sup>

## Пользователи Windows

Раньше установить Lynx в системе Windows было не так-то просто, но теперь можно загрузить инсталлятор с сайта csant.info.<sup>2</sup> После установки вы сможете запускать Lynx прямо из меню Start.

## Пользователи Mac OS X

Lynx для Mac OS X можно загрузить с сайта Apple.<sup>3</sup>

## Обсуждение

Lynx корректно работает под любой платформой, однако для его использования необходимо выучить несколько несложных команд. На рис. 8.1 изображен вид типичного сайта при просмотре в Lynx.

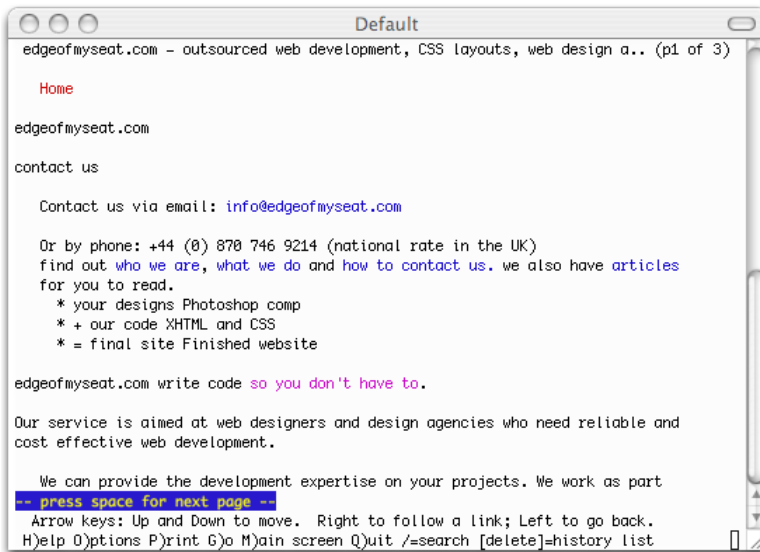


Рис. 8.1. Просмотр сайта в Lynx

<sup>1</sup> <http://lynx.isc.org/release/>

<sup>2</sup> <http://www.csant.info/lynx.htm>

<sup>3</sup> [http://www.apple.com/downloads/macosx/unix\\_open\\_source/lynxtextweb-browser.html](http://www.apple.com/downloads/macosx/unix_open_source/lynxtextweb-browser.html)

Чтобы открыть веб-страницу, нажмите G и введите ее URL-адрес. Щелкните по клавише Enter, и Lynx загрузит указанную страницу. Если на загружаемом сайте используются cookies, браузер спросит, хотите ли вы их принять. Для ответа можно использовать Y (да), N (нет), A (всегда принимать cookies с этого сайта) или V (никогда не принимать cookies с этого сайта).

Для навигации используйте клавиши со стрелками. С помощью стрелок, указывающих вверх и вниз, можно переходить по ссылкам. Щелчок по стрелке, указывающей направо, осуществляет переход по текущей ссылке, а по стрелке, указывающей налево, – переход на предыдущую страницу.

В процессе заполнения формы для навигации по ее полям используются верхняя и нижняя клавиши со стрелками, а текст вводится с помощью клавиатуры, как и в обычном браузере.

Lynx можно использовать для просмотра локальных файлов, что особенно удобно в процессе разработки. Если вы используете локальный веб-сервер – например, Apache или IIS, – то можно указать Lynx адрес через «локальный хост» (localhost/). Обратите внимание, что HTML-документы можно открывать и напрямую, указав имя файла и путь к нему.

Дополнительную информацию об использовании Lynx можно получить в справочной системе, вызываемой клавишей H; навигация по ней осуществляется так же, как и по обычным сайтам.

---

#### Совет

**Доступно на практике.** Посвятите некоторое время просмотру любимых сайтов в текстовом браузере, и вскоре вы по достоинству оцените необходимость использования атрибута alt для изображений и правильного структурирования документа.

---

## Тестирование сайта с помощью экранного диктора

Чтобы понять, каким сайт предстанет перед пользователями экранных дикторов, лучше всего поставить себя на их место на практике; однако самый известный и популярный на сегодняшний день экранный диктор, JAWS, стоит достаточно дорого (хотя вы можете воспользоваться демонстрационной версией, работающей 40 минут), а изучение особенностей его использования займет некоторое время. Какие альтернативные возможности есть у веб-разработчиков, желающих протестировать созданный сайт с помощью экранного диктора?

## Решение

Получить представление о том, как содержащаяся на сайте информация будет воспроизведена экраным диктором, можно с помощью Fire Vox, расширения для Firefox. Пользователи Firefox под Windows, Mac OS X или Linux могут загрузить его бесплатно с сайта разработчика, Чарльза Л. Чена (Charles L. Chen),<sup>1</sup> а затем установить его в своей системе, следуя инструкции. На сайте также можно найти краткое руководство по использованию Fire Vox.

## Обсуждение

Безусловно, попробовав использовать экранный диктор, вы получите некоторое представление о том, как незрячие пользователи работают с Интернетом, однако человек с нормальным зрением никогда не сможет полностью прочувствовать положение такого пользователя и, учитывая ограниченное использование экранных дикторов при тестировании сайта, так же глубоко разобраться во всех тонкостях данной программы, как те, для кого это – единственное средство получения информации из Сети. Поэтому если у вас нет времени на тщательное изучение особенностей экранного диктора, тестирование сайта с его помощью следует скорее считать попыткой лучше представить себе работу незрячих пользователей, чем полноценной проверкой сайта на совместимость с экраным диктором.

## Создание отдельных таблиц стилей для различных устройств

Отдельные таблицы стилей можно создавать для разных браузеров, а можно ли для разных устройств?

## Решение

В спецификации CSS2.1 введено понятие **среды отображения**, которое позволяет веб-разработчикам создавать отдельные таблицы стилей или отвести часть общей таблицы стилей для применения к документу при его отображении различными устройствами.

В теге, привязывающем таблицу стилей к документу, можно указать среду отображения, для которой она предназначена. К примеру, в следующем теге осуществляется привязка таблицы стилей, которая будет использоваться только для отображения сайта браузером на экране компьютера:

---

<sup>1</sup> <http://www.firevox.clcworld.net/downloads.html>



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

Для размещенных внутри элемента `head` правил стилей также можно указать такую метку:

```
<style type="text/css" media="all">
  :
</style>
```

В обоих примерах значением атрибута `media` служит название среды отображения, для которой предназначена таблица стилей. Она будет использоваться только устройствами, поддерживающими указанную среду.

## Обсуждение

В спецификации CSS 2.1 указаны следующие типы среды отображения:<sup>1</sup>

<code>all</code>	подходит для всех устройств
<code>braille</code>	для тактильных устройств, например, брайлеровских дисплеев
<code>embossed</code>	для постраничной печати по системе Брайля
<code>handheld</code>	для портативных устройств (как правило, с небольшим экраном и ограниченной пропускной способностью полосы)
<code>print</code>	для документов с постраничной организацией и документов, отображаемых на экране в режиме предпросмотра перед печатью (Print Preview)
<code>projection</code>	для презентаций, представляемых с помощью проектора (используется Opera в полноэкранном режиме)
<code>screen</code>	предназначается прежде всего для цветных компьютерных мониторов
<code>speech</code>	для синтезаторов речи (обратите внимание, что в CSS2 с этой целью использовалась среда отображения <code>aural</code> )
<code>tty</code>	для устройств с фиксированной сеткой символов (телетайпов, терминалов или портативных устройств с ограниченными возможностями отображения); при написании таблицы стилей для данной среды отображения не следует использовать в качестве единицы измерения пиксели
<code>tv</code>	для телевизионных дисплеев (цветных, с низким разрешением, ограниченными возможностями прокрутки и функцией воспроизведения звука)

---

<sup>1</sup> <http://www.w3.org/TR/CSS21/media.html#media-types>

Помимо использования атрибута `media`, как было описано выше, у разработчика есть возможность указания стилей для различных устройств в единой таблице стилей с помощью правила `@media`.

Ниже представлен пример применения этого приема на практике. В рассматриваемой таблице стилей указано, что при печати документа должен использоваться шрифт размером в десять пунктов, а при отображении его на экране – в 12 пикселей. В обоих случаях цвет текста будет черным:

```
@media print {
  body {
    font-size: 10pt;
  }
}
@media screen {
  body {
    font-size: 12px;
  }
}
@media screen, print {
  body {
    color: #000000;
  }
}
```

В настоящее время не так много устройств полностью поддерживают различные среды отображения, и метод указания конкретных стилей для различных устройств в будущем, скорее всего, будет заменен модулем запросов среды CSS3.<sup>1</sup> К примеру, Opera Mobile и Safari для iPhone прекратили поддержку среды отображения `handheld`, поскольку компании-разработчики считают их полнофункциональными браузерами в отличие от типичных маломощных браузеров, установленных на портативных устройствах. Вместо этого они сделали выбор в пользу поддержки запросов среды, что гораздо эффективнее для указания различных стилей для современных мобильных устройств, корректно обрабатывающих CSS-код, но обладающих некоторыми ограничениями, обусловленными небольшим размером экрана.<sup>2</sup>

---

<sup>1</sup> <http://reference.sitepoint.com/css/mediaqueries/>

<sup>2</sup> С помощью запросов среды можно, например, указать определенные правила стилей для устройств с максимальной шириной экрана в 480 пикселей. Более подробную информацию можно получить на сайте Сообщества разработчиков Opera (Opera Developer Community) по адресу <http://dev.opera.com/articles/view/opera-mobile-9-5-the-developer-angle/> и на сайте Центра разработчиков Apple Safari (Apple Safari DevCenter) по адресу <http://developer.apple.com/safari/>.

Однако указание различных типов среды все еще остается актуальным; одним из самых полезных и поддерживаемых современными браузерами типов среды отображения является `print`. В следующем разделе мы рассмотрим способ его использования для создания версии документа для печати.

### Совет

**Не нужно начинать с самого начала.** При создании таблицы стилей для другой среды отображения проще всего скопировать уже существующий файл и сохранить его под новым именем. Таким образом, все селекторы сразу окажутся у вас под рукой, и останется всего лишь изменить некоторые из указанных для них правил.

## Создание таблицы стилей для печатной версии документа

Веб-страницы редко удается напечатать удачно, поскольку принципы создания страницы, хорошо смотрящейся на экране, обычно отличаются от принципов создания страницы, хорошо смотрящейся на бумаге. Однако, используя различные типы среды CSS, можно создать таблицу стилей, применяемую к документу при его печати.

### Решение

Можно создать специальную таблицу стилей для печати, например:

*chapter08/print-stylesheet.html*

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
<head>
<title>Print Style Sheet</title>
<meta http-equiv="content-type"
  content="text/html; charset=utf-8" />
<link rel="stylesheet" type="text/css" href="main.css"
  title="default" />
<link rel="stylesheet" type="text/css" href="print.css"
  media="print" />
</head>
<body>
<div id="banner"></div>
<div id="content">
  <h1>Chinese-style stuffed peppers</h1>
  <p>These stuffed peppers are lovely as a starter, or as a side dish for a
  Chinese meal. They also go down well as part of a buffet and even children
  seem to like them.</p>
  <h2>Ingredients</h2>
  ⋮
  ⋮
```

```
</div>
<div id="navigation">
  <ul id="mainnav">
    <li><a href="#">Recipes</a></li>
    <li><a href="#">Contact Us</a></li>
    <li><a href="#">Articles</a></li>
    <li><a href="#">Buy Online</a></li>
  </ul>
</div>
</body>
</html>
```

### *chapter08/main.css*

```
body, html {
  margin: 0;
  padding: 0;
}
#navigation {
  width: 200px;
  font: 90% Arial, Helvetica, sans-serif;
  position: absolute;
  top: 41px;
  left: 0;
}
#navigation ul {
  list-style: none;
  margin: 0;
  padding: 0;
  border: none;
}
#navigation li {
  border-bottom: 1px solid #ED9F9F;
  margin: 0;
}
#navigation li a:link, #navigation li a:visited {
  display: block;
  padding: 5px 5px 5px 0.5em;
  border-left: 12px solid #711515;
  border-right: 1px solid #711515;
  color: #FFFFFF;
  background-color: #b51032;
  text-decoration: none;
}
#navigation li a:hover {
  color: #FFFFFF;
  background-color: #711515;
}
#content {
  margin-left: 260px;
  margin-right: 60px;
}
```

```
#banner {
  height: 40px;
  background-color: #711515;
  border-bottom: 1px solid #ED9F9F;
  text-align: right;
  padding-right: 20px;
  margin-top: 0;
}
#banner ul {
  margin: 0;
  padding: 0;
}
#banner li {
  display: inline;
}
#banner a:link, #banner a:visited {
  font: 80% Arial, Helvetica, sans-serif;
  color: #FFFFFF;
  background-color: transparent;
}
#content p, #content li {
  font: 80%/1.6em Arial, Helvetica, sans-serif;
}
#content p {
  margin-left: 1.5em;
}
#content h1, #content h2 {
  font: 140% Georgia, "Times New Roman", Times, serif;
}
#content h2 {
  font: 120% Georgia, "Times New Roman", Times, serif;
  padding-bottom: 3px;
}
}
```

### *chapter08/print.css*

```
body, html {
  margin: 0;
  padding: 0;
}
#navigation {
  display: none;
}
#content {
  margin-left: 20pt;
  margin-right: 30pt;
}
#banner {
  display: none;
}
#content p, #content li {
  font: 12pt/20pt "Times New Roman", Times, serif;
```

```

}
#content p {
  margin-left: 20pt;
}
#content h1, #content h2 {
  font: 16pt Georgia, "Times New Roman", Times, serif;
  color: #4b4b4b;
  background-color: transparent;
}
#content h2 {
  font: 14pt Georgia, "Times New Roman", Times, serif;
  padding-bottom: 2pt;
  border-bottom: 1pt dotted #CCCCCC;
}
}

```

## Обсуждение

При наличии отдельной таблицы стилей для печати посетителям будет гораздо удобнее работать с вашим сайтом, в особенности если вы используете много графики. Печать документа, содержащего множество изображений, может оказаться затратной по количеству чернил и времени при печати на старых принтерах. Кроме того, некоторые сайты плохо выглядят на бумаге из-за неудачно подобранных цветов или особенностей структуры документа. На рис. 8.2 изображена страница, состоящая из двух колонок, созданных с помощью CSS, с навигационной панелью сбоку и основной областью, содержащей рецепт.



Рис. 8.2. Вид страницы, состоящей из двух колонок, в браузере

На рис. 8.3 изображена та же страница в режиме Print Preview.

Эти рисунки позволяют четко увидеть очевидные отличия между отображением на экране и отпечатком. Стандартный лист бумаги формата А4 имеет ограниченную ширину, поэтому если на нем разместится меню, то на печать рецепта останется только половина ширины листа.

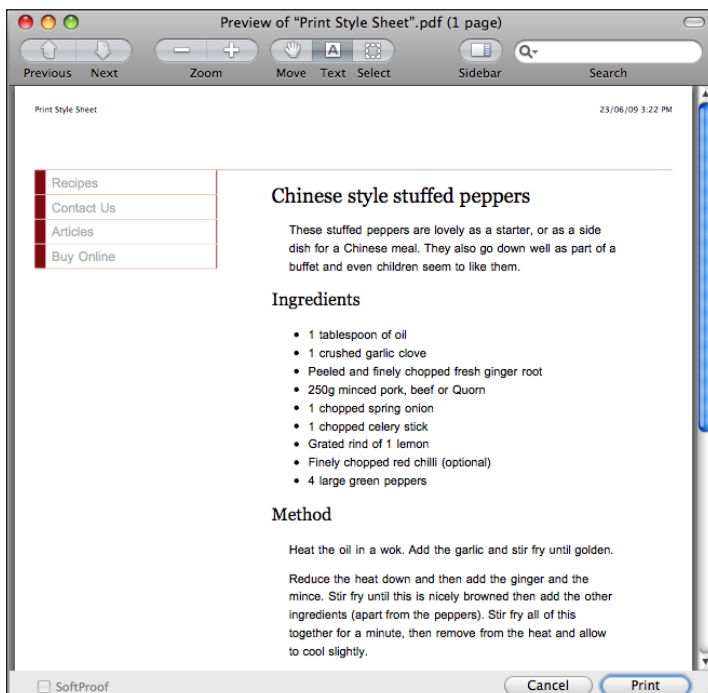


Рис. 8.3. Вид страницы в режиме Print Preview

Это может привести к тому, что длинные рецепты придется печатать на нескольких страницах.

Обычно на сайтах имеется версия для печати тех страниц, которые, по предположениям разработчиков, пользователь может захотеть распечатать. Однако применение такого подхода приводит к необходимости обслуживания сразу нескольких версий документа. Кроме того, пользователь должен найти кнопку Print (Печать) на самой странице вместо того, чтобы воспользоваться соответствующей кнопкой браузера. При использовании CSS страница будет отображена в соответствии с правилами, указанными в таблице стилей для печати, вне зависимости от того, по какой из кнопок щелкнет пользователь – на странице или в браузере.

Рассмотрим процесс создания таблицы стилей для печати и ее привязки к документу более подробно.

## Привязка таблицы стилей для печати

Откройте текущую таблицу стилей и сохраните ее под именем **print.css**. Это наша новая таблица стилей для печатной версии документа. В коде документа необходимо указать ссылку на нее, предназначенную для среды отображения print:

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

## Создание правил стилей для печати

После сохранения существующей таблицы стилей под новым именем ее можно взять за основу новой таблицы, изменив уже записанные в ней правила. В моем документе навигационная панель расположена внутри элемента div; для него задано следующее правило стиля:

*chapter08/main.css (фрагмент)*

```
#navigation {
  width: 200px;
  font: 90% Arial, Helvetica, sans-serif;
  position: absolute;
  top: 41px;
  left: 0;
}
```

Поскольку в печатной версии навигация совершенно бесполезна, ее нужно скрыть. Для этого заменим содержимое приведенного выше блока описания на единственное свойство с заданным значением `display:none`:

*chapter08/print.css (фрагмент)*

```
#navigation {
  display: none;
}
```

Теперь можно удалить все правила, применявшиеся к элементам, расположенным внутри блока `navigation`.

Кроме того, можно расширить область с основным контентом, чтобы она заняла все свободное пространство страницы. Найдите в таблице стилей раздел, посвященный элементу `content`:

*chapter08/main.css (фрагмент)*

```
#content {
  margin-left: 260px;
  margin-right: 60px;
}
```

Внешний отступ слева можно уменьшить, поскольку теперь не нужно выделять дополнительное пространство для размещения панели навигации. Кроме того, не мешает изменить единицы измерения с пик-



селов (для отображения на экране) на пункты (для печати), о чем мы подробно говорили в разделе «Выбор единиц измерения для задания размера шрифтов: пиксели, пункты, пики или что-то другое» главы 2:

*chapter08/print.css (фрагмент)*

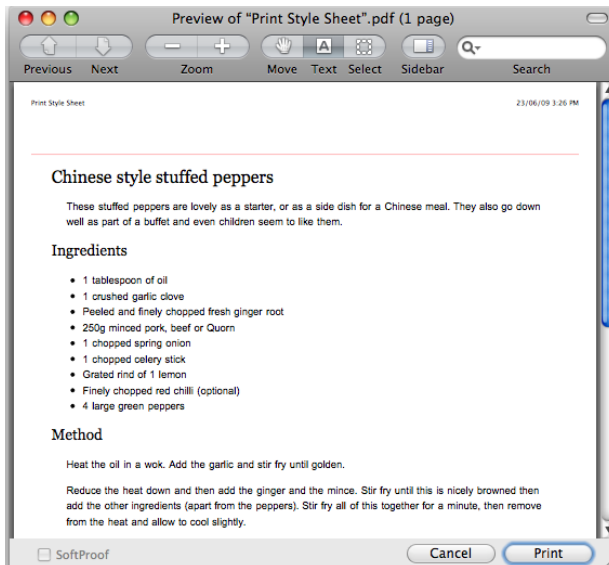
```
#content {
  margin-left: 20pt;
  margin-right: 30pt;
}
```

При просмотре документа в режиме Print Preview, как показано на рис. 8.4, или при выводе на печать с помощью браузера мы увидим, что панель навигации исчезла и пространство под основной контент используется на странице более эффективно.

На рис. 8.4 видно, что в верхней части страницы расположена линия. Это нижняя часть рамки баннера. С последним можно поступить так же, как и с панелью навигации. Прежде всего для этого нужно найти применяемое по отношению к баннеру правило в таблице стилей:

*chapter08/main.css (фрагмент)*

```
#banner {
  height: 40px;
  background-color: #711515;
  border-bottom: 1px solid #ED9F9F;
}
```



*Рис. 8.4. После удаления панели навигации страница гораздо лучше будет выглядеть на бумаге*

По аналогии с панелью навигации зададим для баннера свойство `display` со значением `none` и удалим оставшиеся правила для элемента с указанным идентификатором.

*chapter08/print.css (фрагмент)*

```
#banner {
    display: none;
}
```

Наконец, приступим к форматированию текста. Обычно в версии для печати я изменяю цветной текст на черный или серый, за исключением тех случаев, когда текст по тем или иным причинам должен сохранить цветовое оформление. Размер шрифта укажем в пунктах, чтобы он оставался постоянной величиной в различных системах.

Кроме того, текст, предназначенный для печати, лучше оформить шрифтом семейства `serif` – так он будет лучше восприниматься на бумаге. Ниже приведен измененный код:

*chapter08/print.css (фрагмент)*

```
#content p, #content li {
    font: 12pt/20pt "Times New Roman", Times, serif;
}
#content p {
    margin-left: 20pt;
}
#content h1, #content h2 {
    font: 16pt Georgia, "Times New Roman", Times, serif;
    color: #4b4b4b;
    background-color: transparent;
}
#content h2 {
    font: 14pt Georgia, "Times New Roman", Times, serif;
    padding-bottom: 2pt;
    border-bottom: 1pt dotted #CCCCCC;
}
```

Теперь страница выглядит еще проще, зато читать текст стало значительно удобнее. Ее окончательный вид представлен на рис. 8.5.

---

### Совет

**Таблицы стилей для печати и просто таблицы.** Таблицу стилей для печати, предназначенную для сверстанного на CSS документа, создать несложно, однако помимо этого можно создавать такие таблицы стилей и для макетов на основе таблиц, что особенно удобно в том случае, если ширина ячеек была задана с помощью CSS. При этом можно скрыть ячейки, содержащие панель навигации, таким же образом, как мы поступили с элементом `div` в приведенном выше примере.

---

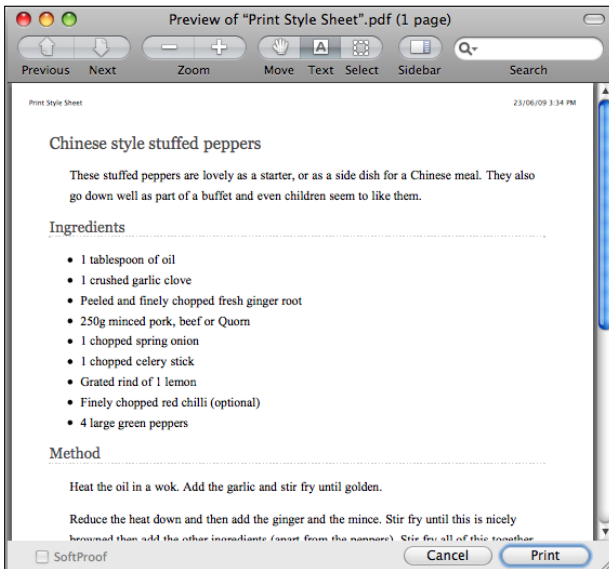


Рис. 8.5. Просмотр страницы в режиме Print Preview после внесения всех изменений

## Добавление на сайт альтернативных таблиц стилей

Некоторые современные браузеры дают пользователю возможность увидеть список привязанных к документу таблиц стилей и самостоятельно выбрать одну из них. Таким образом можно с легкостью добавить, например, таблицу стилей, инвертирующую цвета элементов страницы.

### Решение

Укажите в коде документа ссылку на альтернативную таблицу стилей, присвоив атрибут `rel="alternate stylesheet"` элементу `link` и задайте содержащему ее файлу имя, которое будет отображено в меню браузера. Оно должно быть описательным (например, `high contrast` – «высокая контрастность»), чтобы пользователю было легче сделать выбор. Кроме того, основной таблице стилей также нужно задать имя, чтобы ее можно было отличить от альтернативной:

*chapter08/alternate-stylesheets.html (фрагмент)*

```
<link rel="stylesheet" type="text/css" href="main.css"
  title="default" />
<link rel="stylesheet" type="text/css" href="print.css"
  media="print" />
<link rel="alternate stylesheet" type="text/css"
  href="highcontrast.css" title="high contrast" />
```

*chapter08/highcontrast.css*

```
body, html {
    margin: 0;
    padding: 0;
    background-color: #000000;
    color: #FFFFFF;
}
#navigation {
    width: 200px;
    font: 90% Arial, Helvetica, sans-serif;
    position: absolute;
    top: 41px;
    left: 0;
}
#navigation ul {
    list-style: none;
    margin: 0;
    padding: 0;
    border: none;
}
#navigation li {
    border-bottom: 1px solid #ED9F9F;
    margin: 0;
}
#navigation li a:link, #navigation li a:visited {
    display: block;
    padding: 5px 5px 5px 0.5em;
    border-left: 12px solid #711515;
    border-right: 1px solid #711515;
    color: #FFFFFF;
    background-color: #b51032;
    text-decoration: none;
}
#navigation li a:hover {
    color: #FFFFFF;
    background-color: #711515;
}
#content {
    margin-left: 260px;
    margin-right: 60px;
}
#banner {
    height: 40px;
    background-color: #711515;
    border-bottom: 1px solid #ED9F9F;
    text-align: right;
    padding-right: 20px;
    margin-top: 0;
}
#banner ul {
    margin: 0;
```

```

padding: 0;
}
#banner li {
display: inline;
}
#banner a:link, #banner a:visited {
font: 80% Arial, Helvetica, sans-serif;
color: #FFFFFF;
background-color: transparent;
}
#content p, #content li {
font: 80%/1.6em Arial, Helvetica, sans-serif;
}
#content p {
margin-left: 1.5em;
}
#content h1, #content h2 {
font: 140% Georgia, "Times New Roman", Times, serif;
color: #FFFFFF;
background-color: transparent;
}
#content h2 {
font: 120% Georgia, "Times New Roman", Times, serif;
padding-bottom: 3px;
border-bottom: 1px dotted #CCCCCC;
}
}

```

На рис. 8.6 показан вид страницы при выборе пользователем альтернативной таблицы стилей из меню View браузера Firefox.

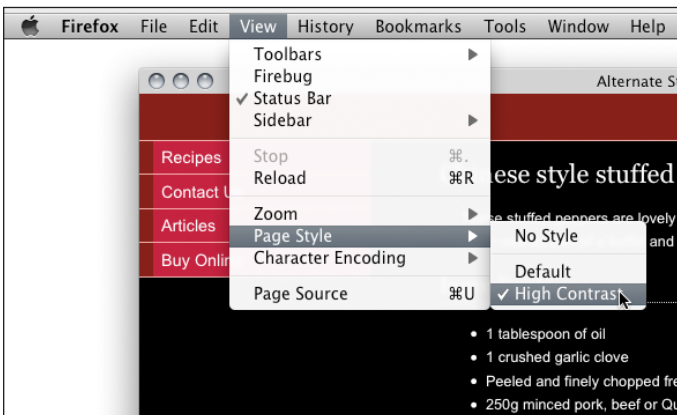


Рис. 8.6. Вид страницы при переключении к таблице стилей High Contrast в Firefox

## Обсуждение

Эта функция проста в использовании и значительно расширяет возможности пользователя, требуя минимальных затрат со стороны разработчика. Как правило, создание таблицы стилей, изменяющей цветовое оформление документа, занимает совсем немного времени. Для этого достаточно сохранить существующий файл со стилями под новым именем и поменять значения свойств, определяющих параметры шрифта, цвет и положение элементов.

К сожалению, поддержка данной функции во многих браузерах ограничена, а в Internet Explorer отсутствует вовсе. Однако пользователи, для которых данная возможность представляет интерес, могут выбрать для работы соответствующий браузер.

### Совет

**Видите, как я предусмотрителен!** В настоящее время крайне мало сайтов предлагают своим посетителям подобную возможность, и потому не помешает сообщить им, что на вашем сайте могут быть использованы альтернативные таблицы стилей. Можно создать специальную страницу с информацией о возможностях сайта и указать ссылку на нее на главной странице.

## Масштабируемый макет

Следующим этапом после написания таблицы стилей для крупной печати является создание **масштабируемого макета** страницы. Данная концепция получила широкое распространение благодаря Джо Кларку (Joe Clark); она состоит в изменении структуры страницы путем расположения всех ее элементов в одну колонку с помощью CSS и использовании контрастных цветов.<sup>1</sup> Это особенно важно для пользователей, использующих функцию увеличения масштаба страницы в современных браузерах (поскольку при этом крупнее становится не только текст, но и все остальные элементы страницы) или пользующихся специальными программами, подобными экранной лупе, для облегчения зрительного восприятия текста. При увеличении масштаба страницы таким способом боковые панели смещаются за пределы поля зрения, в результате чего на странице остается только основной контент.

Благодаря увеличению размера шрифта и отображению светлого текста на темном фоне (такое сочетание облегчает восприятие информации) при использовании масштабируемого макета пользователям со слабым зрением будет гораздо удобнее работать с сайтом. Таблица стилей, создающая масштабируемый макет страницы, с которой мы работали на протяжении этой главы, может содержать нижеследующие правила, а на рис. 8.7 показан вид страницы при просмотре в браузере.

<sup>1</sup> <http://joelclark.org/access/webaccess/zoom/>



*Рис. 8.7. Вид страницы с применением таблицы стилей для zoom layout*

#### *chapter08/zoom.css*

```

body, html {
    margin: 1em 2em 2em 2em;
    padding: 0;
    font-size: 140%;
    background-color: #333;
    color: #FFFFFF;
}

#navigation ul {
    list-style: none;
    margin: 0;
    padding: 0;
    border: none;
}

#navigation li {
    float: left;
    width: 20%;
}

#navigation li a:link, #navigation li a:visited {
    color: #FFFF00;
}

#navigation li a:hover {
    text-decoration: none;
}

```

```
}

#content {
  padding: 1em 0 0 0;
  clear:left;
}

#content p, #content li {
  line-height: 1.6em;
}

#content h1, #content h2 {
  font: 140% Georgia, "Times New Roman", Times, serif;
  color: #FFFFFF;
  background-color: transparent;
}

#content h2 {
  font: 120% Georgia, "Times New Roman", Times, serif;
}
```

## Нужно ли отображать на сайте инструменты для изменения размера шрифта или переключения между различными таблицами стилей?

Вам, вероятно, встречались сайты с инструментами для увеличения или уменьшения размера шрифта, осуществляемого путем переключения между таблицами стилей. В данной главе уже были рассмотрены способы создания альтернативных таблиц стилей; нужно ли добавить на сайт ссылку для переключения между ними?

### Решение

Создание на сайте инструментов для изменения размера шрифта – часто представленных в виде нескольких букв А разной величины – совершенно излишне, если сайт построен правильно, с использованием методов задания величины шрифта, обеспечивающих возможность ее изменения во всех браузерах. Гораздо важнее, чтобы пользователи научились увеличивать и уменьшать шрифт предлагаемыми браузером средствами, поскольку это дает возможность регулирования размера текста на большинстве сайтов, а не только на тех, где для этого имеются специальные инструменты.

При создании масштабируемого макета страницы или использовании иных альтернативных методов структурирования документа механизм регулирования его размера менее очевиден. Пользователям современных браузеров доступна функция изменения масштаба всей страницы, а не только размера шрифта. В идеале любой дизайн должен масштабироваться корректно, избавляя разработчиков от необходимости исполь-



зования альтернативных таблиц стилей. Однако при увеличении масштаба макета с фиксированной шириной часть его элементов нередко исчезает за пределами экрана.

На мой взгляд, в данной ситуации хорошим решением будет предложить пользователям другой дизайн, соответствующий их потребностям. Для этого достаточно создать альтернативную таблицу стилей, как показано выше, а на странице, посвященной вопросам доступности сайта, разместить подробную информацию о том, какие браузеры позволяют использовать альтернативные таблицы стилей и каким образом можно переключиться к масштабируемой версии. Или же можно создать на странице переключатель с помощью **JavaScript** или сценария, выполняющегося на сервере. Однако эту функцию следует добавлять только при большой необходимости. Не нужно думать, что вы *обязаны* предоставить пользователю переключатели стилей страницы или что их использование избавляет вас от необходимости адаптации основного дизайна для всех категорий пользователей.

## Создание альтернативных таблиц стилей без копирования кода из основной таблицы

В предыдущих примерах создание альтернативной таблицы стилей происходило путем копирования правил из основной таблицы и изменения значений некоторых свойств. Действительно ли так необходимо использовать в альтернативной таблице стилей все эти правила или достаточно только тех, которые *нужно* изменить?

### Решение

Чтобы ответить на этот вопрос, создадим несколько таблиц стилей. Первая из них будет содержать свойства, в изменении которых нет необходимости, вторая – свойства, значения которых будут меняться, а третья – измененные значения этих свойств:

*alternate-stylesheets2.html (фрагмент)*

```
<link rel="stylesheet" type="text/css" href="main2.css" />
<link rel="stylesheet" type="text/css" href="defaultcolors.css"
title="Default" />
<link rel="stylesheet" type="text/css" href="print.css"
media="print" />
<link rel="alternative stylesheet" type="text/css"
href="highcontrast2.css" title="High Contrast" />
```

*chapter08/main2.css*

```
body, html {
margin: 0;
padding: 0;
```

```
}
#navigation {
  font: 90% Arial, Helvetica, sans-serif;
  position: absolute;
  left: 0;
  top: 41px;
}
#navigation ul {
  list-style: none;
  margin: 0;
  padding: 0;
  border: none;
}
#navigation li {
  border-bottom: 1px solid #ED9F9F;
  margin: 0;
}
#navigation li a:link, #navigation li a:visited {
  display: block;
  padding: 5px 5px 5px 0.5em;
  border-left: 12px solid #711515;
  border-right: 1px solid #711515;
  background-color: #B51032;
  color: #FFFFFF;
  text-decoration: none;
}
#navigation li a:hover {
  background-color: #711515;
  color: #FFFFFF;
}
#banner {
  background-color: #711515;
  border-bottom: 1px solid #ED9F9F;
  text-align: right;
  padding-right: 20px;
  margin-top: 0;
}
#banner ul {
  margin: 0;
}
#banner li {
  display: inline;
}
#banner a:link, #banner a:visited {
  font: 80% Arial, Helvetica, sans-serif;
  color: #FFFFFF;
  background-color: transparent;
}
#content p, #content li {
  font: 80%/1.6em Arial, Helvetica, sans-serif;
}
```

```
#content p {
  margin-left: 1.5em;
}
#content h1, #content h2 {
  font: 140% Georgia, "Times New Roman", Times, serif;
  color: #B51032;
  background-color: transparent;
}
#content h2 {
  font: 120% Georgia, "Times New Roman", Times, serif;
  padding-bottom: 3px;
  border-bottom: 1px dotted #ED9F9F;
}
```

#### *chapter08/defaultcolors.css*

```
body, html {
  background-color: #FFFFFF;
  color: #000000;
}

#content h1, #content h2 {
  color: #B51032;
  background-color: transparent;
}

#content h2 {
  border-bottom: 1px dotted #ED9F9F;
}
```

#### *chapter08/highcontrast2.css*

```
body, html {
  background-color: #000000;
  color: #FFFFFF;
}

#content h1, #content h2 {
  color: #FFFFFF;
  background-color: transparent;
}

#content h2 {
  border-bottom: 1px dotted #CCCCCC;
}
```

## Обсуждение

Для создания файла **highcontrast.css**, с которым мы работали в разделе «Добавление на сайт альтернативных таблиц стилей», я изменила всего несколько свойств исходной таблицы стилей. Поменялись значения цвета фона и шрифта:

*chapter08/main.css (фрагмент)*

```
body, html {
  margin: 0;
  padding: 0;
  background-color: #FFFFFF;
  color: #000000;
}
```

*chapter08/highcontrast.css (excerpt)*

```
body, html {
  margin: 0;
  padding: 0;
  background-color: #000000;
  color: #FFFFFF;
}
```

Кроме того, я изменила цвет заголовков первого и второго уровня:

*chapter08/main.css (фрагмент)*

```
#content h1, #content h2 {
  font: 140% Georgia, "Times New Roman", Times, serif;
  color: #B51032;
  background-color: transparent;
}

#content h2 {
  font: 120% Georgia, "Times New Roman", Times, serif;
  padding-bottom: 3px;
  border-bottom: 1px dotted #ED9F9F;
}
```

*chapter08/highcontrast.css (фрагмент)*

```
#content h1, #content h2 {
  font: 140% Georgia, "Times New Roman", Times, serif;
  color: #FFFFFF;
  background-color: transparent;
}

#content h2 {
  font: 120% Georgia, "Times New Roman", Times, serif;
  padding-bottom: 3px;
  border-bottom: 1px dotted #CCCCCC;
}
```

Чтобы не копировать все содержимое таблицы стилей для создания файла **highcontrast.css**, можно выделить только те свойства, значение которых точно придется менять. Затем их нужно разместить в новом файле, который задаст цветовую схему страницы по умолчанию; таб-

лица стилей для создания версии сайта повышенной контрастности будет содержать те же свойства с измененными значениями. Таким образом можно избежать необходимости работы с несколькими версиями по сути одной таблицы стилей.

## Заключение

В настоящей главе мы рассмотрели несколько способов использования таблиц стилей, благодаря которым ваш сайт станет более доступным для различных категорий пользователей. Правильное построение структуры документа уже само по себе облегчает работу с сайтом для пользователей экранных дикторов, а использование альтернативных таблиц стилей дает разработчику новые инструменты для обеспечения удобства пользователей с ограниченными возможностями.

# 9

## Позиционирование элементов с помощью CSS

Отбросив проблемы, связанные с ошибками браузеров, можно утверждать, что основные принципы позиционирования элементов страницы с помощью CSS достаточно просты. Прочно усвоив основные правила, вы вскоре обнаружите, что решение сложных задач, по сути, заключается в применении этих правил по отношению к различным частям документа.

Данная глава начинается с рассмотрения основных методов построения макета страницы с помощью CSS, а также представляет полезные приемы, которые помогут вам создавать по-настоящему красивые и оригинальные сайты. Они послужат фундаментом для дальнейшей реализации ваших творческих идей. После прочтения данной главы вы получите полное представление об инструментах CSS, необходимых для создания работоспособных сайтов. Во второй части главы мы углубимся в более частные задачи верстки макетов, так что, если с основными понятиями вы разобрались, вы сможете найти в них необходимые для решения собственных задач приемы.

### **В каких случаях следует использовать классы, а в каких – идентификатор**

На первый взгляд, использование классов и идентификаторов практически не отличается друг от друга: и тому и другому можно присваивать CSS-свойства с необходимыми значениями для изменения вида элементов (X)HTML-документа. Однако при каких обстоятельствах эффективнее использовать классы, а при каких – идентификаторы?

## Решение

Самое важное правило применения классов и идентификаторов состоит в том, что идентификатор можно использовать *только один раз* в документе, поскольку он выполняет функцию выделения единственного элемента. Присвоенный элементу уникальный идентификатор больше нельзя использовать в том же документе.

Классы, напротив, можно использовать в документе произвольное количество раз. Поэтому, если некоторые свойства присущи сразу нескольким элементам страницы, использование классов можно считать идеальным решением.

Любому элементу можно одновременно присвоить и класс, и идентификатор. В качестве примера такого элемента можно привести все расположенные на странице поля для ввода текста; для обращения к ним в сценарии на JavaScript каждому полю придется также задать уникальный идентификатор. При этом не нужно определять для него какие-либо стили.

Как правило, я присваиваю идентификатор самым основным элементам, играющим важную роль в формировании структуры документа. Таким образом, в моем коде чаще всего встречаются следующие идентификаторы: `header`, `content`, `nav` и `footer`. Ниже приведен пример кода:

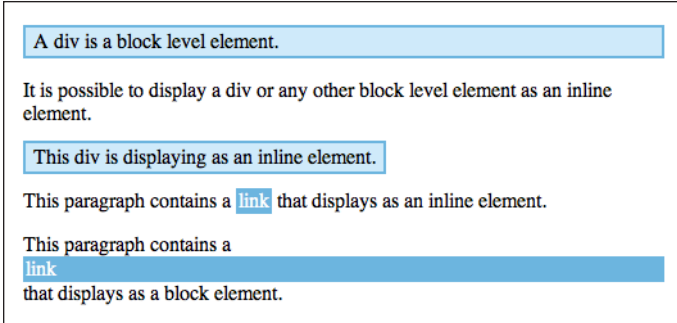
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
<head>
<title>Absolute positioning</title>
<meta http-equiv="content-type"
content="text/html; charset=utf-8" />
<link rel="stylesheet" type="text/css" href="position2.css" />
</head>
<body>
<div id="header"> ... </div>
<div id="content">
:Main page content here ...
</div>
<div id="nav"> ... </div>
</body>
</html>
```

## Отображение строкового элемента как блочного, и наоборот

Иногда необходимо, чтобы браузер обрабатывал определенные HTML-элементы несколько иным образом, нежели по умолчанию.

## Решение

На рис. 9.1 видно, что элемент `div` отображается в непрерывном потоке элементов страницы, а ссылка отображается как блочный элемент.



*Рис. 9.1. Отображение блочного элемента в качестве строкового, а строкового – в качестве блочного*

Такой эффект был достигнут с помощью следующего кода:

*chapter09/inline-block.html*

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
  <head>
    <title>Inline and block level elements</title>
    <meta http-equiv="content-type"
      content="text/html; charset=utf-8" />
    <style type="text/css">
      #one {
        background-color: #CFEAF4;
        border: 2px solid #6CB5DF;
        padding: 2px 6px 2px 6px;
      }
      #two {
        background-color: #CFEAF4;
        border: 2px solid #6CB5DF;
        padding: 2px 6px 2px 6px;
        display: inline;
      }
      a {
        background-color: #6CB5DF;
        color: #FFFFFF;
        text-decoration: none;
        padding: 1px 2px 1px 2px;
      }
      a.block {

```



```
        display: block;
    }
</style>
</head>
<body>
  <div id="one">A div is a block level element.</div>
  <p>It is possible to display a div or any other block level element as
an inline element. </p>
  <div id="two">This div is displaying as an inline element.
</div>
  <p>This paragraph contains a
  <a href="http://www.sitepoint.com/">link</a> that
displays as an inline element.</p>
  <p>This paragraph contains a
  <a class="block" href="http://www.sitepoint.com/">link</a> that
displays as a block element.</p>
</body>
</html>
```

## Обсуждение

Блочные элементы отличаются от строковых тем, что они могут содержать другие элементы любого типа. Они также форматируются иным образом: по умолчанию для их размещения отводится прямоугольная область, по ширине занимающая всю страницу. Строковые элементы размещаются непосредственно в тексте, причем для того чтобы они уместились внутри содержащего их блока, осуществляется переход на новую строку. По умолчанию следующие HTML-элементы считаются блочными: заголовки (h1, h2, h3,...), абзацы (p), списки (ul, ol) и различные контейнеры (div, blockquote).

В приведенном выше примере есть элемент div, отображаемый в соответствии с его типом. Будучи блочным элементом, по ширине он занимает все доступное пространство родительского элемента – в данном случае body. Если бы он располагался внутри другого div или в ячейке таблицы, его ширина соответствовала бы ширине этого элемента.

Чтобы div отображался по-другому, можно превратить его в строковый элемент путем изменения следующего свойства:

```
display: inline;
```

Аналогичным образом можно превратить строковый элемент в блочный. Обратите внимание, что в рассмотренном ранее примере элемент a по умолчанию является строковым. У разработчика достаточно часто возникает потребность в изменении его типа на блочный, например при создании навигационной панели с помощью CSS. Для этого нужно задать свойству display значение block. В предыдущем примере благодаря использованию такого приема серое окошко со ссылкой по ширине занимает весь экран.

## Задание внешних и внутренних отступов с помощью CSS

Чем отличаются свойства `margin` и `padding` и в чем выражается результат их использования?

### Решение

Свойство `margin` добавляет пустое пространство *за пределами* элемента. Такой внешний отступ можно задавать независимо с каждой стороны:

```
margin-top: 1em;
margin-right: 2em;
margin-bottom: 0.5em;
margin-left: 3em;
```

То же самое можно записать в сокращенной форме:

```
margin: 1em 2em 0.5em 3em;
```

Если отступы со всех сторон одинаковы, можно использовать следующее простое правило:

```
margin: 1em;
```

В результате его применения отступ между границей элемента и другими элементами составит `1em`.

На рис. 9.2 показан вид блочного элемента после добавления к нему внешних отступов. Код документа таков:

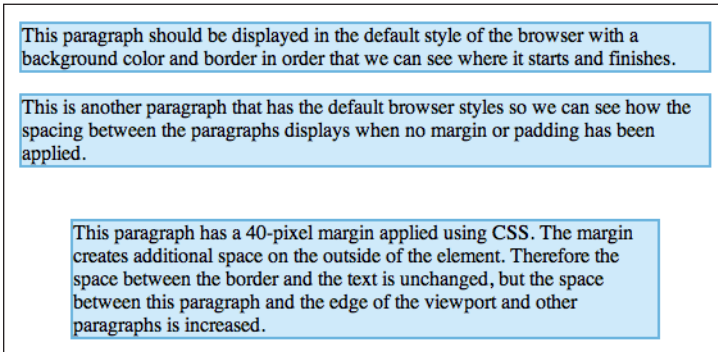
*chapter09/margin.html*

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
  <head>
    <title>Margins</title>
    <meta http-equiv="content-type"
      content="text/html; charset=utf-8" />
    <style type="text/css">
      p {
        background-color: #CFEAF4;
        border: 2px solid #6CB5DF;
      }
      p.margintest {
        margin: 40px;
      }
    </style>
  </head>
  <body>
    <p>This paragraph should be displayed in the default ...</p>
    <p>This is another paragraph that has the default ...</p>
```

```

    <p class="margintest">This paragraph has a 40-pixel ...</p>
  </body>
</html>

```



**Рис. 9.2.** Добавление к элементу внешних отступов с помощью CSS

Свойство `padding` добавляет отступы *внутри* элемента – между его границей и дочерними элементами. Внутренние отступы можно задавать по отдельности с каждой стороны:

```

padding-top: 1em;
padding-right: 1.5em;
padding-bottom: 0.5em;
padding-left: 2em;

```

То же самое можно записать в сокращенной форме:

```
padding: 1em 1.5em 0.5em 2em;
```

Как и при использовании свойства `margin`, если внутренние отступы со всех сторон одинаковы, можно использовать следующее простое правило:

```
padding: 1em;
```

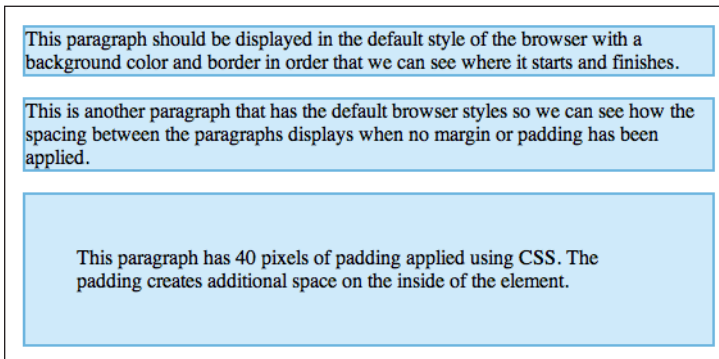
На рис. 9.3 показан вид блочного элемента после добавления к нему внутренних отступов. Код документа представлен ниже. Сравните его с документом, изображенным на рис. 9.2, – разница между свойствами `margin` и `padding` очевидна.

*chapter09/padding.html*

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
  <head>
    <title>Padding</title>
    <meta http-equiv="content-type"
      content="text/html; charset=utf-8" />
    <style type="text/css">

```



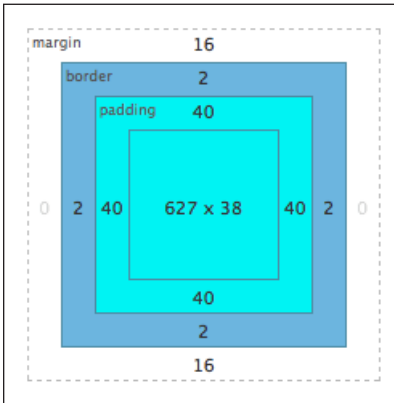
*Рис. 9.3. Добавление элементу внутренних отступов с помощью CSS*

```
p {
    border: 2px solid #AAAAAA;
    background-color: #EEEEEE;
}
p.paddingtest {
    padding: 40px;
}
</style>
</head>
<body>
  <p>This paragraph should be displayed in the default ...</p>
  <p>This is another paragraph that has the default ...</p>
  <p class="paddingtest">This paragraph has 40 pixels ...</p>
</body>
</html>
```

## Обсуждение

Приведенное решение наглядно демонстрирует назначение свойств `margin` и `padding`. Как мы видим, первое из них создает дополнительное пустое пространство между элементом и окружающими элементами, а второе – внутри элемента, между его границей и содержимым. Рисунок 9.4 служит прекрасной иллюстрацией к вышесказанному.

При одновременном добавлении как внешних, так и внутренних отступов, к элементу с фиксированной шириной его размеры будут увеличены соответствующим образом. К примеру, у нас есть элемент с шириной в 400 пикселей, к которому мы добавляем внутренний отступ величиной в 40 пикселей со всех сторон. В этом случае его ширина составит 480 пикселей (изначально ширина в 400 пикселей плюс по 40 пикселей с каждой стороны). Добавьте к этому внешний отступ в 20 пикселей, и в итоге элемент займет по ширине 520 пикселей (видимая часть составит 480 пикселей, а пустое пространство со всех сторон – 20 пикселей). Если макет вашего документа основан на точном расчете, обязатель-



*Рис. 9.4. Указание внешних и внутренних отступов, а также параметров рамки элемента*

но просчитывайте величину каждого элемента, не забывая о внешних и внутренних отступах.

#### Примечание

**Путаница в режиме совместимости.** Очень старые версии Internet Explorer, а именно 5 и 5.5, считают, что размер рамки и внутреннего отступа уже включен в указанную ширину элемента; в этих браузерах ширина описанных выше элементов сохранит значение 400 пикселей, а из-за добавления отступов и рамки размер видимой части элемента уменьшится. Чтобы избежать возникновения такой ситуации, можно добавить внутренний отступ родительскому элементу вместо того, чтобы задавать внешний отступ самому элементу. В настоящее время большинство дизайнеров не считают обеспечение нормального отображения сайта в Internet Explorer 5 приоритетной задачей, поэтому единственное, о чем следует задуматься, – как избежать перехода Internet Explorer 6 в режим совместимости. Этот вопрос был рассмотрен в главе 7.

## Обтекание текстом изображения

Изображение можно окружить текстом средствами HTML, а именно с помощью атрибута `align`. В настоящее время использовать его не рекомендуется, однако CSS предлагает полноценную замену этому атрибуту!

### Решение

Для реализации обтекания текстом изображения слева или справа используется CSS-свойство `float`, как показано на рис. 9.5.

Код страницы будет таким:


*chapter09/float.html*

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
  <head>
    <title>Float</title>
    <meta http-equiv="content-type"
      content="text/html; charset=utf-8" />
    <style type="text/css">
      .featureimg {
        float: left;
        width: 319px;
      }
    </style>
  </head>
  <body>
    <h1>Chinese-style stuffed peppers</h1>
    
    <p>These stuffed peppers are lovely as a starter, or as a ...</p>
    : More paragraphs
  </body>
</html>

```

## Chinese-style stuffed peppers



These stuffed peppers are lovely as a starter, or as a side dish for a Chinese meal. They also go down well as part of a buffet and even children seem to like them.

Heat the oil in a wok. Add the garlic and stir fry until golden.

Reduce the heat and add the ginger and the mince. Stir fry until this is nicely browned then add the other ingredients (apart from the peppers). Stir fry all of this together for a minute, then remove from the heat and allow to cool slightly.

Core and remove the seeds of the peppers and cut them into quarters.

Divide the mince mixture between these quarters and arrange the peppers in an oven proof dish

*Рис. 9.5. Обтекание изображения текстом с помощью свойства float*

## Обсуждение

Свойство `float`<sup>1</sup> удаляет элемент из обычного потока и размещает его у правого или левого края родительского блока. Остальные блочные

<sup>1</sup> Блоки, для которых задано свойство `float`, называют «плавающими», а сам эффект — «обтеканием». — *Прим. перев.*

элементы будут вести себя так, как будто указанного элемента вообще не существует. Однако строковые элементы, такие как контент, будут смещаться, освобождая для него пространство, и потому использование свойства `float` создает эффект обтекания изображения текстом.

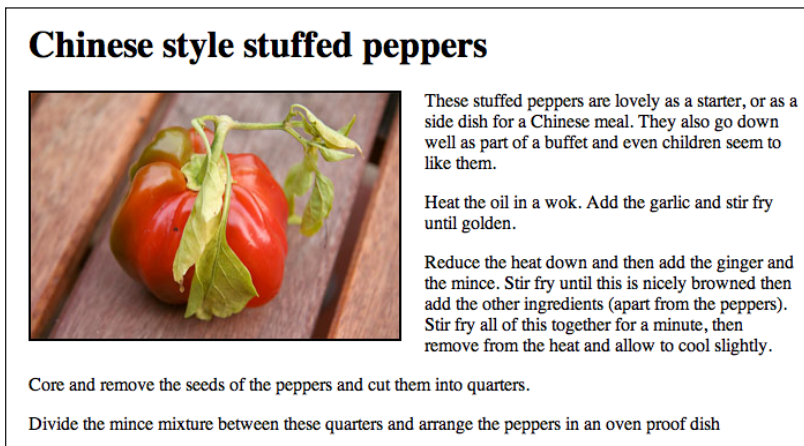
Как видно на рис. 9.5, текст расположен вплотную к границе изображения. При добавлении к изображению рамки он будет соприкасаться с ней.

Для добавления пустого пространства между текстом и изображением необходимо задать для изображения внешний отступ. Поскольку оно смещено к левой границе родительского элемента, достаточно будет добавит отступ справа и снизу, чтобы слегка отодвинуть текст от него:

*chapter09/float2.html (фрагмент)*

```
.featureimage {
  float: left;
  width: 319px;
  border: 2px solid #000000;
  margin-right: 20px;
  margin-bottom: 6px;
}
```

На рис. 9.6 показан результат обработки данного кода: теперь между изображением и текстом появился просвет.



*Рис. 9.6. После добавления дополнительных отступов справа и снизу документ выглядит привлекательнее*

## Как избежать смещения следующего элемента вверх при использовании свойства `float`

После присвоения изображению или иному элементу свойства `float` остальные блочные элементы будут вести себя, как будто его не суще-

ствует, при этом содержащиеся в них текст и строчные изображения будут этот плавающий элемент обтекать. Как заставить элементы страницы отобразиться ниже такого плавающего блока?

## Решение

Использование свойства `clear` позволяет отобразить заданный элемент под любыми элементами, для которых задано свойство `float`, как если бы они оставались частью потока элементов. К примеру, зададим это свойство со значением `both` для первого абзаца, следующего за списком ингредиентов:

### *chapter09/float-clear.html*

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
  <head>
    <title>float and clear</title>
    <meta http-equiv="content-type"
      content="text/html; charset=utf-8" />
    <style type="text/css">
      .featureimg {
        float: right;
        width: 319px;
        margin-left: 20px;
        margin-bottom: 6px;
        border: 1px solid #000000;
      }
      .clear {
        clear: both;
      }
    </style>
  </head>
  <body>
    <h1>Chinese style stuffed peppers</h1>
    
    <ul>
      <li>1 tablespoon of oil</li>
      <li>1 crushed garlic clove</li>
      <li>Peeled and finely chopped fresh ginger root</li>
      <li>250g minced pork, beef or Quorn</li>
      <li>1 chopped spring onion</li>
      <li>1 chopped celery stick</li>
      <li>Grated rind of 1 lemon</li>
      <li>Finely chopped red chilli (optional)</li>
      <li>4 large green peppers</li>
    </ul>
    <p class="clear">These stuffed peppers are lovely as a...</p>
    : More paragraphs
  </body>
</html>
```



```
</body>
</html>
```

Как видно на рис. 9.7, на котором мы сдвинули плавающий элемент вправо, при такой разметке абзац сдвигается вниз и располагается под плавающим изображением.

## Chinese style stuffed peppers

- 1 tablespoon of oil
- 1 crushed garlic clove
- Peeled and finely chopped fresh ginger root
- 250g minced pork, beef or Quorn
- 1 chopped spring onion
- 1 chopped celery stick
- Grated rind of 1 lemon
- Finely chopped red chilli (optional)
- 4 large green peppers



These stuffed peppers are lovely as a starter, or as a side dish for a Chinese meal. They also go down well as part of a buffet and even children seem to like them.

Heat the oil in a wok. Add the garlic and stir fry until golden.


Рис. 9.7. Первый абзац отображается под изображением

## Обсуждение

Свойство `float` удаляет элемент из потока, и последующие блочные элементы просто-напросто игнорируют его. Для более наглядной демонстрации этого механизма можно добавить элементам документа рамки, в нашем случае – элементам `ul` и `p`, как показано на рис. 9.8.

## Chinese style stuffed peppers

- 1 tablespoon of oil
- 1 crushed garlic clove
- Peeled and finely chopped fresh ginger root
- 250g minced pork, beef or Quorn
- 1 chopped spring onion
- 1 chopped celery stick
- Grated rind of 1 lemon
- Finely chopped red chilli (optional)
- 4 large green peppers



These stuffed peppers are lovely as a starter, or as a side dish for a Chinese meal. They also go down well as part of a buffet and even children seem to like them.

Heat the oil in a wok. Add the garlic and stir fry until golden.

Рис. 9.8. Добавление рамки шириной в два пиксела к элементам «ul» и «p»

Плавающее изображение отображается поверх элементов блочного уровня. Содержащийся в них текст окружает его, однако сами элементы не обращают внимания на его наличие. Это означает, что, если высота изображения в нашем примере превысит высоту списка ингредиентов, следующий за списком абзац будет обтекать изображение, как показано на рис. 9.8.

Для размещения абзаца под нижней границей изображения можно воспользоваться свойством `clear`:

*chapter09/float-clear.html (фрагмент)*

```
.clear {
  clear: both;
}
```


Этот CSS-класс нужно присвоить первому тегу `<p>`, следующему за списком ингредиентов:

*chapter09/float-clear.html (фрагмент)*

```
<p class="clear">These stuffed peppers are lovely as a
starter, or as a side dish for a Chinese meal. They also go
down well as part of a buffet and even children seem to like
them.</p>
```

Оставим добавленные ранее рамки и обновим документ. Его нынешний вид показан на рис. 9.9; как видите, указанный абзац начинается под изображением перца, что особенно наглядно демонстрирует его рамка.

## Chinese style stuffed peppers

<ul style="list-style-type: none"><li>• 1 tablespoon of oil</li><li>• 1 crushed garlic clove</li><li>• Peeled and finely chopped fresh ginger root</li><li>• 250g minced pork, beef or Quorn</li><li>• 1 chopped spring onion</li><li>• 1 chopped celery stick</li><li>• Grated rind of 1 lemon</li><li>• Finely chopped red chilli (optional)</li><li>• 4 large green peppers</li></ul>	
--	--

These stuffed peppers are lovely as a starter, or as a side dish for a Chinese meal. They also go down well as part of a buffet and even children seem to like them.

Heat the oil in a wok. Add the garlic and stir fry until golden.

Рис. 9.9. Размещение абзаца под изображением с помощью свойства `clear`

Свойство `clear` также может принимать значения `left` и `right`, что может пригодиться в случае, если вы хотите, чтобы элемент игнорировал свой-

ство `float` со значениями влево или вправо соответственно. Однако чаще всего вам потребуется значение `both`. Имейте в виду, что использование свойств `float` и `clear` может привести к ошибкам интерпретации, особенно в Internet Explorer – достаточно вспомнить «исчезающий контент» в Internet Explorer 6, о котором мы говорили в главе 7.

## Как расположить логотип сайта слева, а слоган – справа

Если вы когда-нибудь верстали макет страницы с помощью таблиц, то вам хорошо известно, насколько просто достичь изображенного на рис. 9.10 эффекта. Достаточно выровнять содержимое левой ячейки таблицы, состоящей из двух колонок, по левому краю, а правой – по правому. К счастью, того же результата можно добиться и с помощью CSS.

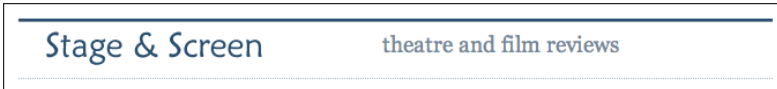


Рис. 9.10. Расположение логотипа слева, а слогана – справа с помощью CSS

### Решение

Для позиционирования элементов указанным образом можно воспользоваться свойством `float`:

*chapter09/slogan-align.html (фрагмент)*

```

:
<body>
  <div id="header">
    
    <span class="slogan">theatre and film reviews</span>
  </div>
</body>
:

```

*chapter09/slogan-align.css*

```

body {
  margin: 0;
  padding: 0;
  background-color: #FFFFFF;
  color: #000000;
  font-family: Arial, Helvetica, sans-serif;
  border-top: 2px solid #2A4F6F;
}
#header {
  border-top: 1px solid #778899;

```

```

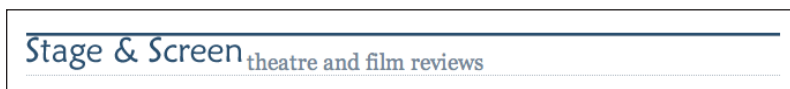
border-bottom: 1px dotted #B2BCC6;
height: 3em;
}
#header .slogan {
font: 120% Georgia, "Times New Roman", Times, serif;
color: #778899;
background-color: transparent;
float: right;
width: 300px;
margin-right: 2em;
margin-top: 0.5em;
}
#header .logo {
float: left;
width: 187px;
margin-left: 1.5em;
margin-top: 0.5em;
}
}

```

## Обсуждение

С помощью свойства `float` можно выравнивать элементы «шапки» по любой стороне. Перед добавлением данного свойства элементы будут отображаться рядом друг с другом, как показано на рис. 9.11.

Такое позиционирование элементов обусловлено HTML-разметкой, в которой не содержится никаких указаний относительно их расположения. Именно поэтому они просто следуют друг за другом.



*Рис. 9.11. Расположение элементов по умолчанию*

Рассмотрим HTML-код, управляющий расположением слогана:

*chapter09/slogan-align.html (фрагмент)*

```

<div id="header">
  
  <span class="slogan">theatre and film reviews</span>
</div>

```

С помощью свойства `float` можно разместить элементы класса `logo` слева, а элементы класса `slogan` – справа. Кроме того, я добавила правило для выравнивания текста слогана вправо, без которого он прижимался бы к левому краю содержащего его элемента `span`, который выровнен вправо! Результат проведенных действий показан на рис. 9.12.



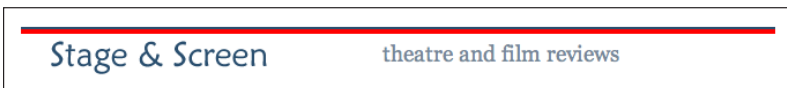
*Рис. 9.12. Применение свойства `float` для позиционирования элементов*

Для создания дополнительного пустого пространства вокруг элементов добавим к логотипу внешний отступ сверху и слева, а к слогану – сверху и справа:

*chapter09/slogan-align.css (фрагмент)*

```
#header .slogan {
  font: 120% Georgia, "Times New Roman", Times, serif;
  color: #778899;
  background-color: transparent;
  float: right;
  width: 300px;
  text-align: right;
  margin-right: 2em;
  margin-top: 0.5em;
}
#header .logo {
  float: left;
  width: 187px;
  margin-left: 1.5em;
  margin-top: 0.5em;
}
```

При использовании такого подхода следует учесть следующее: если вы делаете все элементы в контейнере плавающими, то ничто не заставляет контейнер оставаться открытым, и его высота уменьшается до нуля, как показано на рис. 9.13.



*Рис. 9.13. При указании свойства `float` для дочерних элементов «шапки» она полностью сворачивается*

Для большей наглядности я добавила к контейнеру логотипа и слогана рамку. На рис. 9.14. показан вид контейнера до присваивания дочерним элементам свойства `float`.



*Рис. 9.14. Размер «шапки» до присваивания дочерним элементам свойства `float`*

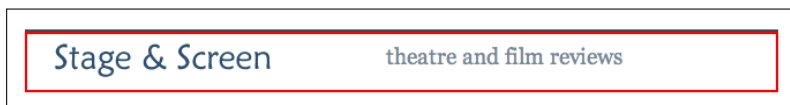
После присваивания дочерним элементам свойства `float` высота контейнера уменьшается до нуля, поскольку теперь они удалены из общего потока элементов в документе, в результате чего рамка контейнера превратилась в сплошную красную линию, как видно на рис. 9.13.

Во избежание такой ситуации можно задать высоту блока с помощью свойства `height`:

*chapter09/slogan-align.css (фрагмент)*

```
#header {
  border-top: 1px solid #778899;
  border-bottom: 1px dotted #B2BCC6;
  height: 3em;
}
```

Теперь блок занимает отведенную ему область страницы, как показано на рис. 9.15.



*Рис. 9.15. После задания свойства `height` для `div` с «шапкой» страница отображается совершенно нормально*

При выборе значения свойства `height` в такой ситуации следует учитывать возможные изменения в документе при увеличении размера шрифта пользователем. В качестве единиц измерения при этом лучше всего использовать `em`, поскольку при этом высота элемента будет увеличиваться в соответствии с размером текста, и он по-прежнему сможет вместить его, не «выталкивая» плавающие элементы.

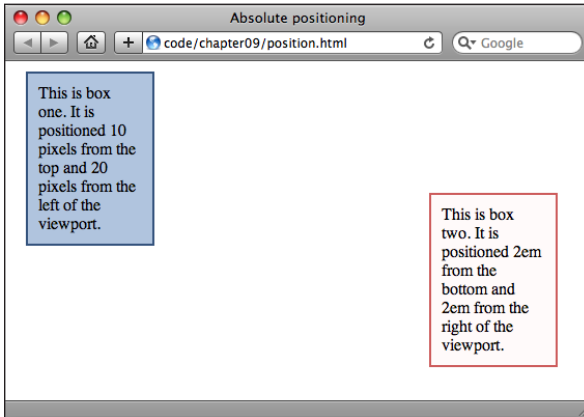
Если бы объем текста в данном контейнере был неизвестен, то следовало бы воспользоваться свойством `clear`, о котором мы говорили чуть ранее.

## Позиционирование элемента на странице с помощью CSS

С помощью CSS можно указать точное расположение элемента на странице.

### Решение

CSS дает разработчику возможность **абсолютного позиционирования** элемента путем указания расстояния от него до верхнего, левого, правого или нижнего края документа. Два блока, изображенных на рис. 9.16, позиционированы абсолютно.



**Рис. 9.16.** Абсолютное позиционирование блоков

Код страницы будет таким:

*chapter09/position.html*

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
  <head>
    <title>Absolute positioning</title>
    <meta http-equiv="content-type"
      content="text/html; charset=utf-8" />
    <link rel="stylesheet" type="text/css" href="position.css" />
  </head>
  <body>
    <div id="box1">This is box one. It is positioned 10 pixels
      from the top and 20 pixels from the left of the viewport.
    </div>
    <div id="box2">This is box two. It is positioned 2em from the bottom and
      2em from the right of the viewport.</div>
  </body>
</html>

```

*chapter09/position.css*

```

#box1 {
  position: absolute;
  top: 10px;
  left: 20px;
  width: 100px;
  background-color: #B0C4DE;
  border: 2px solid #34537D;
}
#box2 {
  position: absolute;
  bottom: 2em;

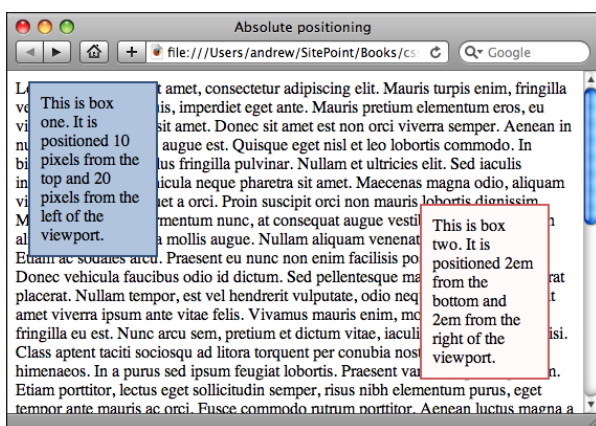
```

```

right: 2em;
width: 100px;
background-color: #FFFAFA;
border: 2px solid #CD5C5C;
}
    
```

## Обсуждение

Задание элементу свойства `position` со значением `absolute` полностью удаляет его из основного потока элементов страницы. К примеру, при добавлении нескольких абзацев текста в рассмотренный выше документ два блока будут отображаться поверх него, как показано на рис. 9.17.



*Рис. 9.17. Два абсолютно позиционированных блока больше не принадлежат основному потоку элементов*

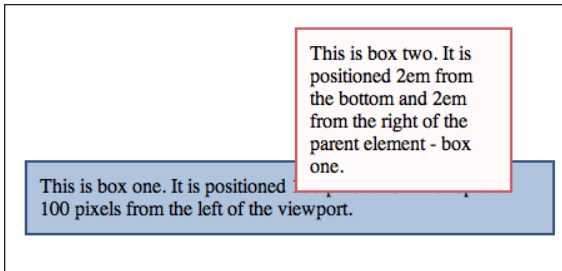
В HTML-коде документа абзацы располагаются под абсолютно позиционированными элементами `div`, однако последние удалены из основного потока, поэтому при обработке кода текст начинается непосредственно в верхнем левом углу страницы, как если бы блоков вообще не существовало.

Как мы увидим позже в разделе «Создание «резинового» макета с двумя колонками, в котором слева расположено меню, а справа – основная область с контентом», путем добавления внешних и внутренних отступов для других элементов с помощью свойств `margin` и `padding` можно создать дополнительное пространство для размещения абсолютно позиционированных элементов. Возможно, из рассмотренного примера это напрямую не следует, но элементы можно абсолютно позиционировать не только по отношению к краям документа (хотя такой подход наиболее широко распространен), но и по отношению к краям родительского элемента.

На рис. 9.18 изображен документ, содержащий два блока, причем розовый блок вложен в синий. Поскольку последний также позициониру-



ван абсолютно, то именно по отношению к его краям рассчитывается позиция розового блока при абсолютном позиционировании.



*Рис. 9.18. Позиционирование одного блока по отношению к другому*

При этом был использован следующий код:

*chapter09/position2.html (фрагмент)*

```
<div id="box1">This is box one. It is positioned 100 pixels from the top and
100 pixels from the left of the viewport.
  <div id="box2">This is box two. It is positioned 2em from the bottom and
2em from the right of the parent element - box one.
  </div>
</div>
```

*chapter09/position2.css*

```
#box1 {
  position: absolute;
  top: 100px;
  left: 100px;
  width: 400px;
  background-color: #B0C4DE;
  border: 2px solid #34537D;
}
#box2 {
  position: absolute;
  bottom: 2em;
  right: 2em;
  width: 150px;
  background-color: #FFFafa;
  border: 2px solid #CD5C5C;
}
```

Для большей наглядности присвоим элементу box1 свойство height со значением 300 пикселей:

*chapter09/position3.css (фрагмент)*

```
#box1 {
  position: absolute;
  top: 100px;
```

```
left: 100px;  
width: 400px;  
height: 300px;  
background-color: #B0C4DE;  
border: 2px solid #34537D;  
}
```

Как видно на рис. 9.19, розовый блок теперь целиком располагается внутри синего, а не «вылезает» из него сверху. Такой эффект достигается позиционированием розового блока по отношению к нижнему и правому краям синего блока.

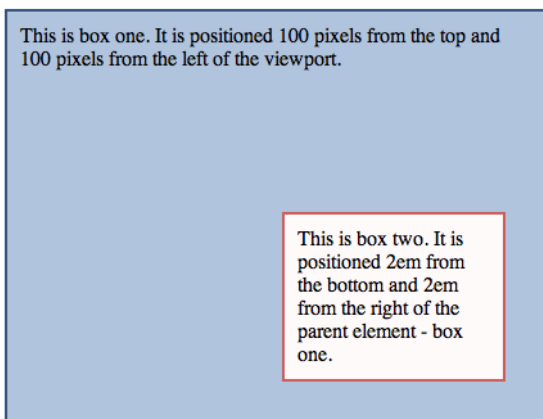


Рис. 9.19. Розовый блок отображается внутри синего

### Примечание

**Позиционирование от родителя.** Важно помнить, что дочерние элементы (box2) могут позиционироваться по отношению к родительскому элементу (box1), только если расположение последнего также задано средствами CSS.

Если для родительского элемента не задано свойство `position`, дочерний элемент будет позиционироваться по отношению к краям первого позиционированного предка – родителя родителя и т. д. – или в итоге элемента `body` (иными словами, по отношению к краям документа). В приведенном выше примере при отсутствии позиционирования родительского элемента расположение дочернего элемента будет рассчитываться по отношению к краям документа, поскольку он является предком следующего уровня.

## Центрирование блока на странице

В Сети достаточно часто встречаются сайты, контент которых расположен в центрированном блоке с фиксированной шириной, например, как на рис. 9.20. Как достичь такого эффекта средствами CSS?

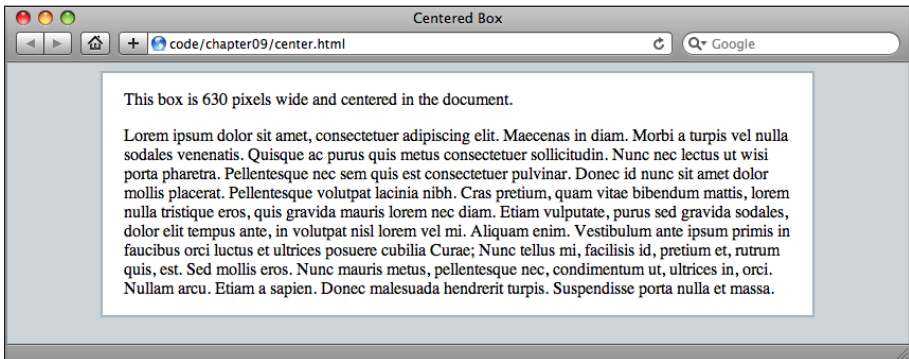


Рис. 9.20. Центрирование блока с фиксированной шириной с помощью CSS

## Решение

Для центрирования блока с фиксированной шириной можно задать для него внешний отступ по бокам с помощью CSS-свойства `margin` со значением `auto`:

### *chapter09/center.html*

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
  <head>
    <title>Centered Box</title>
    <meta http-equiv="content-type"
      content="text/html; charset=utf-8" />
    <link rel="stylesheet" type="text/css" href="center.css" />
  </head>
  <body>
    <div id="content">
      <p>This box is 630 pixels wide and centered in the document.
      </p>
      <p>Lorem ipsum dolor sit amet, consectetur adipiscing ...
      </p>
    </div>
  </body>
</html>
```

### *chapter09/center.css*

```
body {
  background-color: #CCD3D9;
  color: #000000;
}
#content {
  width: 630px;
  margin-left: auto;
```

```
margin-right: auto;
border: 2px solid #A6B2BC;
background-color: #FFFFFF;
color: #000000;
padding: 0 20px 0 20px;
}
```

## Обсуждение

Таким образом можно с легкостью центрировать блоки; использование такого подхода оптимально для расположения области с контентом в центре страницы.

При задании свойству `margin` значения `auto` браузер создаст одинаковые отступы для блока как справа, так и слева, так что он будет расположен в центре. В разделе «Создание «резинового» макета с двумя колонками, в котором слева расположено меню, а справа – основная область с контентом» мы познакомимся со способом расположения элементов внутри централизованного таким образом контейнера.

### Примечание

**Internet Explorer 5.x.** Раньше при использовании такого приема нужно было указать несколько дополнительных CSS-правил, чтобы избежать возникновения ошибок в Internet Explorer 5.x, поскольку в этих версиях браузера не получалось центрировать контент с помощью внешних отступов. Чтобы перейти браузер, нужно было задать свойство `text-align: center` для элемента `body` и `text-align: left` для блока `div`, содержащего контент. Если вам необходимо обеспечить нормальное отображение сайта с такой структурой в этом бедном старом браузере, воспользуйтесь таким приемом.

## Создание блока с закругленными краями

Существует несколько методов создания закругленных краев для блока. Рассмотрим несколько из них:

### Решение 1: CSS3-свойство `border-radius`

Существует свойство `border-radius`, с помощью которого можно задать степень закругленности рамки вокруг блочного элемента. Это свойство будет включено в спецификацию CSS3, когда процесс ее разработки будет завершен.<sup>1</sup> К сожалению, в настоящее время ни один браузер не поддерживает свойство `border-radius`, однако разработчики Safari и Firefox реализовали экспериментальную поддержку такой возможности путем создания расширений, предназначенных для конкретных браузеров.<sup>2</sup>

<sup>1</sup> <http://www.w3.org/TR/css3-border/#the-border-radius>

<sup>2</sup> <http://reference.sitepoint.com/css/vendorspecific/>

Более того, поскольку эти расширения на самом деле являются частью движков соответствующих браузеров, поэтому любой браузер, основанный на движке Gecko (как Camino) или WebKit (как Chrome), также поддерживает эти свойства. Данное решение будет работать только в самых последних версиях Safari, Firefox Camino и Chrome, но не в Internet Explorer и Opera. Ниже приведен HTML- и CSS-код, а результат его обработки показан на рис. 9.21:

*chapter09/corners1.html*

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
  <head>
    <title>Rounded Corners</title>
    <meta http-equiv="content-type"
      content="text/html; charset=utf-8" />
    <link rel="stylesheet" type="text/css" href="corners1.css" />
  </head>
  <body>
    <div class="curvebox">
      <p>Lorem ipsum dolor sit amet, consectetur adipiscing</p>
    </div>
  </body>
</html>

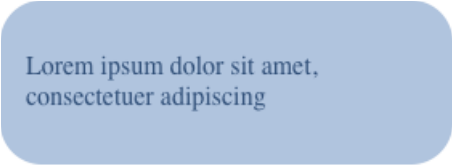
```

*chapter09/corners1.css*

```

.curvebox {
  width: 250px;
  padding: 1em;
  background-color: #B0C4DE;
  color: #33527B;
  -moz-border-radius: 25px;
  -webkit-border-radius: 25px;
}

```



Lorem ipsum dolor sit amet,  
consectetur adipiscing

**Рис. 9.21.** Блок с закругленными краями, созданными с помощью CSS

В этом примере был создан блок с закругленными краями без единого изображения! Эти аккуратные уголки были созданы благодаря использованию следующего свойства:

```

-moz-border-radius: 25px;
-webkit-border-radius: 25px;

```

Если удалить эти свойства из таблицы стилей, получим блок с обычными прямыми углами, как показано на рис. 9.22. Точно так же блок будет отображен броузерами, не поддерживающими данное свойство.

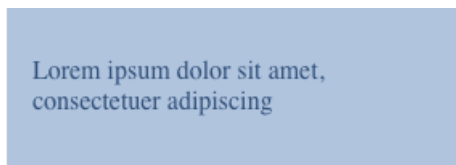


Рис. 9.22. Вид блока в броузерах, не поддерживающих свойство `border-radius`

Безусловно, поскольку данное решение подходит только для пользователей броузеров, основанных на движке Gecko или WebKit, большинство веб-дизайнеров предпочитают пользоваться альтернативными методами.

## Решение 2: использование изображений (с изменением разметки)

Желаемого результата можно добиться путем изменения HTML-кода документа и добавления нескольких изображений. При этом блок будет отображаться с закругленными краями в большинстве броузеров. Итак, прежде всего создадим изображения закругленных уголков в графической программе. Это проще всего сделать путем разделения круга на четверти, как показано на рис. 9.23. Каждый уголок будет представлен с помощью отдельного небольшого изображения.



Рис. 9.23. Изображения закругленных уголков

Ниже представлена разметка документа, в которую уже включены изображения верхнего и нижнего левого уголков, вложенные в верхний и нижний элементы `div`:

*chapter09/corners2.html*

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
  <head>
    <title>Rounded corners</title>
    <meta http-equiv="Content-Type"
      content="text/html; charset=utf-8" />
```

```

    <link rel="stylesheet" type="text/css" href="corners2.css" />
  </head>
  <body>
    <div class="rndbox">
      <div class="rndtop"></div>
      <p>Lorem ipsum dolor sit amet, consectetur adipiscing</p>
      <div class="rndbottom"></div>
    </div>
  </body>
</html>

```

Изображения правого нижнего и верхнего уголка используются в качестве фоновых для соответствующих элементов `div` класса `rndtop` и `rndbottom` путем добавления соответствующих правил в таблицу стилей:

*chapter09/corners2.css (фрагмент)*

```

.rndbox {
  background: #C6D9EA;
  width: 300px;
  font: 0.8em Verdana, Arial, Helvetica, sans-serif;
  color: #000033;
}
.rndtop {
  background: url(topright.gif) no-repeat right top;
}
.rndbottom {
  background: url(bottomright.gif) no-repeat right top;
}
.rndbottom img {
  display:block;
}
.rndbox p {
  margin: 0 0.4em 0 0.4em;
}

```

Такое сочетание HTML- и CSS-кода создает блок со сглаженными углами, показанный на рис. 9.24.

Lorem ipsum dolor sit amet, consectetur adipiscing

**Рис. 9.24.** Блок с закругленными краями, созданный путем изменения HTML-разметки и добавления изображений

## Решение 3: использование JavaScript

Возможно, использование дополнительного кода и изображений – не самый лучший вариант, особенно если в вашем документе должно быть много блоков с закругленными краями. Для решения этой задачи многие разработчики выбрали для создания закругленных краев блока использование JavaScript. Это вполне выигрышный вариант, поскольку даже если пользователь отключит JavaScript, это никак не повлияет на возможность работы с сайтом – а просто отключит отображение дополнительных декоративных элементов.

Создать закругленные края блока с помощью JavaScript можно несколькими способами, однако мы рассмотрим лишь один из них – NiftyCube, – поскольку его можно быстро добавить в код документа, чтобы все заработало. Необходимый сценарий включен в архив кода для данной книги, однако его последнюю версию в виде zip-файла можно загрузить с сайта NiftyCube.<sup>1</sup> После распаковывания архива вы увидите множество страниц, демонстрирующих результат использования данного метода, но для того чтобы самостоятельно применить его на практике, вам понадобятся только два файла – **niftycube.js** со сценарием на JavaScript и **niftyCorners.css** с кодом на CSS. Скопируйте эти файлы в директорию с вашим сайтом. За основу возьмем блок с прямоугольными краями, созданный с помощью следующей разметки:

### *chapter09/corners3-start.html*

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
  <head>
    <title>Rounded Corners</title>
    <meta http-equiv="content-type"
      content="text/html; charset=utf-8" />
    <link rel="stylesheet" type="text/css" href="corners3.css" />
  </head>
  <body>
    <div class="curvebox">
      <p>Lorem ipsum dolor...</p>
    </div>
  </body>
</html>
```

Вы можете задать любой стиль оформления данного блока, придерживаясь всего лишь одного правила: внутренний отступ должен *обязательно* быть указан в пикселах. При использовании других единиц, например em, блок будет отображен некорректно в Internet Explorer. Результат проделанной работы показан на рис. 9.25.

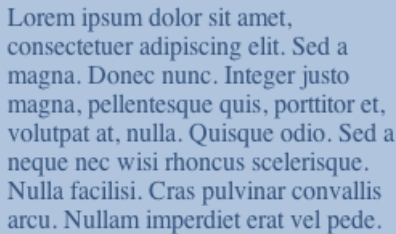
---

<sup>1</sup> <http://www.html.it/articoli/niftycube/>



*chapter09/corners3.css*

```
.curvebox {
  width: 250px;
  padding: 20px;
  background-color: #B0C4DE;
  color: #33527B;
}
```



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed a magna. Donec nunc. Integer justo magna, pellentesque quis, porttitor et, volutpat at, nulla. Quisque odio. Sed a neque nec wisi rhoncus scelerisque. Nulla facilisi. Cras pulvinar convallis arcu. Nullam imperdiet erat vel pede.

*Рис. 9.25. Блок с прямоугольными краями*

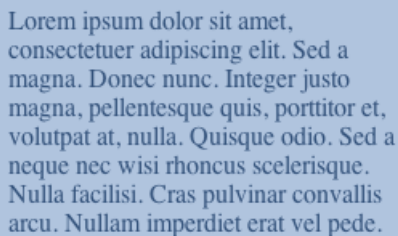
Чтобы края блока стали закругленными, укажите ссылку на файл со сценарием на JavaScript внутри элемента `head` вашего документа, а затем добавьте простую функцию, передающую в сценарий информацию о том, что необходимо закруглить углы элементов класса `curvebox`:

*chapter09/corners3.html*

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
  <head>
    <title>Rounded Corners</title>
    <meta http-equiv="content-type"
      content="text/html; charset=utf-8" />
    <link rel="stylesheet" type="text/css" href="corners3.css" />
    <script type="text/javascript" src="niftycube.js">
    </script>
    <script type="text/javascript">
      window.onload=function(){
        Nifty("div.curvebox");
      }
    </script>
  </head>
  <body>
    <div class="curvebox">
      <p>Lorem ipsum dolor sit amet, consectetur adipiscing...</p>
```

```
</div>  
</body>  
</html>
```

В результате получим блок, изображенный на рис. 9.26.



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed a magna. Donec nunc. Integer justo magna, pellentesque quis, porttitor et, volutpat at, nulla. Quisque odio. Sed a neque nec wisi rhoncus scelerisque. Nulla facilisi. Cras pulvinar convallis arcu. Nullam imperdiet erat vel pede.

*Рис. 9.26. Создание блока с закругленными углами без использования изображений или дополнительного HTML-кода*

## Обсуждение

Существует множество методов создания блоков с закругленными краями без использования JavaScript, однако все они подразумевают добавление дополнительной разметки или структурирования документа определенным образом.<sup>1</sup> Если ваш документ содержит лишь несколько блоков, края которых вам хотелось бы закруглить – например, основной контейнер или пару крупных блоков, – то использование дополнительных изображений и разметки не так уж критично. Однако если таких блоков много, то портить разметку многочисленными дополнительными элементами `div` и изображениями крайне нежелательно. Использование JavaScript позволяет сохранить ясность и четкость HTML-кода, но, как и все решения с использованием сценариев, оно будет работать только при условии, что пользователь не отключил их поддержку.

На мой взгляд, применение JavaScript в таких случаях – для компенсации недостаточной поддержки CSS-свойств – вполне оправдано. Если вашим сайтом удобно пользоваться и без закругленных углов, то пользователи, отключившие JavaScript, останутся на нем. Перед тем как принять окончательное решение об использовании сценариев в вашем проекте, обязательно протестируйте сайт в браузере с отключенным JavaScript, чтобы убедиться в удобстве его использования при таких условиях.

---

<sup>1</sup> Один из методов создания сглаженных углов с использованием семантически верной разметки без JavaScript называется Spanky Corners (<http://tools.sitepoint.com/spanky/>). Он был разработан Алексом Уолкером (Alex Walker) с сайта SitePoint.

## Создание «резинового» макета: слева – меню, а справа – область с контентом

Сайты с меню слева и большой областью с контентом справа, как на рис. 9.27, крайне часто встречаются в Сети. Рассмотрим способ создания такого макета с помощью CSS.



Рис. 9.27. Создание «резинового» макета с двумя колонками с помощью CSS

### Решение

Для достижения показанного на рис. 9.27 эффекта использовались следующая разметка и CSS-код:

*chapter09/2col.html*

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
  <head>
    <title>Stage & Screen - theatre and film reviews</title>
    <meta http-equiv="content-type"
      content="text/html; charset=utf-8" />
    <link rel="stylesheet" type="text/css" href="2col.css" />
  </head>
  <body>

```

```

<div id="header">
  
  <span class="strapline">theatre and film reviews</span>
</div>
<div id="content">
  <h1>Welcome to Stage & Screen</h1>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing ...</p>
  :
</div>
<div id="nav">
  <ul>
    <li><a href="#">Play Reviews</a></li>
    <li><a href="#">Film Reviews</a></li>
    <li><a href="#">Post a Review</a></li>
    <li><a href="#">About this site</a></li>
    <li><a href="#">Contact Us</a></li>
  </ul>
  <h2>Latest Reviews</h2>
  <ul>
    <li><a href="#">Angels & Demons</a></li>
    <li><a href="#">Star Trek</a></li>
    <li><a href="#">Up</a></li>
    <li><a href="#">Land of the Lost</a></li>
  </ul>
</div>
</body>
</html>

```

**chapter09/2col.css**

```

body {
  margin: 0;
  padding: 0;
  background-color: #FFFFFF;
  color: #000000;
  font-family: Arial, Helvetica, sans-serif;
  border-top: 2px solid #2A4F6F;
}

#header {
  border-top: 1px solid #778899;
  border-bottom: 1px dotted #B2BCC6;
  height: 3em;
}

#header .strapline {
  font: 120% Georgia, "Times New Roman", Times, serif;
  color: #778899;
  background-color: transparent;
  float: right;
}

```

```
width: 300px;
text-align: right;
margin-right: 2em;
margin-top: 0.5em;
}

#header .logo {
float: left;
width: 187px;
margin-left: 1.5em;
margin-top: 0.5em;
}

#nav {
position: absolute;
top: 5em;
left: 1em;
width: 14em;
}

#nav ul {
list-style: none;
margin: 0;
padding: 0;
}

#nav li {
font-size: 80%;
border-bottom: 1px dotted #B2BCC6;
margin-bottom: 0.3em;
}

#nav a:link, #nav a:visited {
text-decoration: none;
color: #2A4F6F;
background-color: transparent;
}

#nav a:hover {
color: #778899;
}

#nav h2 {
font: 110% Georgia, "Times New Roman", Times, serif;
color: #2A4F6F;
background-color: transparent;
border-bottom: 1px dotted #CCCCCC;
}

#content {
margin-left: 16em;
```

```

margin-right: 2em;
}

h1 {
font: 150% Georgia, "Times New Roman", Times, serif;
}

#content p {
font-size: 80%;
line-height: 1.6em;
}

```

## Обсуждение

За основу макета возьмем «шапку», созданную в разделе «Как расположить логотип сайта слева, а слоган – справа?». Добавив немного контента, разместим его внутри блока `div` с ID `content`. Меню навигации состоит из двух маркированных списков, расположенных в блоке с ID `nav`. Как вы наверняка уже поняли, при отсутствии каких-либо правил позиционирования эти блоки будут расположены под «шапкой» в том же порядке, в котором они следуют друг за другом в документе (как показано на рис. 9.28).

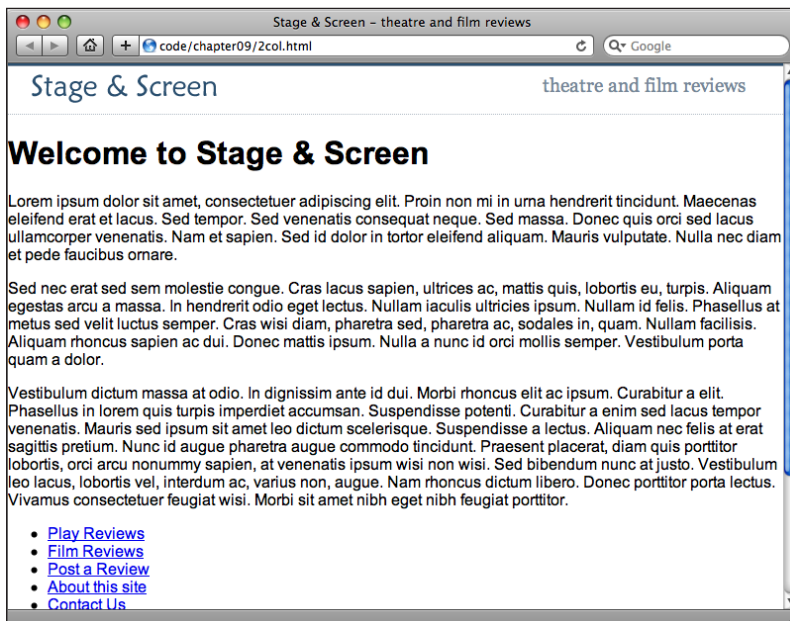


Рис. 9.28. Отображение контента и навигации без позиционирования

На данном этапе CSS-код выглядит следующим образом:

*chapter09/2col.css (фрагмент)*

```
body {
    margin: 0;
    padding: 0;
    background-color: #FFFFFF;
    color: #000000;
    font-family: Arial, Helvetica, sans-serif;
    border-top: 2px solid #2A4F6F;
}

#header {
    border-top: 1px solid #778899;
    border-bottom: 1px dotted #B2BCC6;
    height: 3em;
}

#header .slogan {
    font: 120% Georgia, "Times New Roman", Times, serif;
    color: #778899;
    background-color: transparent;
    float: right;
    width: 300px;
    text-align: right;
    margin-right: 2em;
    margin-top: 0.5em;
}

#header .logo {
    float: left;
    width: 187px;
    margin-left: 1.5em;
    margin-top: 0.5em;
}
```

**Задание размера и позиционирование меню**

Для размещения меню непосредственно под панелью с заголовком воспользуемся методом абсолютного позиционирования, а также зададим его ширину:

*chapter09/2col.css (фрагмент)*

```
#nav {
    position: absolute;
    top: 5em;
    left: 1em;
    width: 14em;
}
```

Как видно на рис. 9.29, после добавления этого правила меню располагается поверх текста, поскольку при абсолютном позиционировании оно было удалено из общего потока элементов документа.

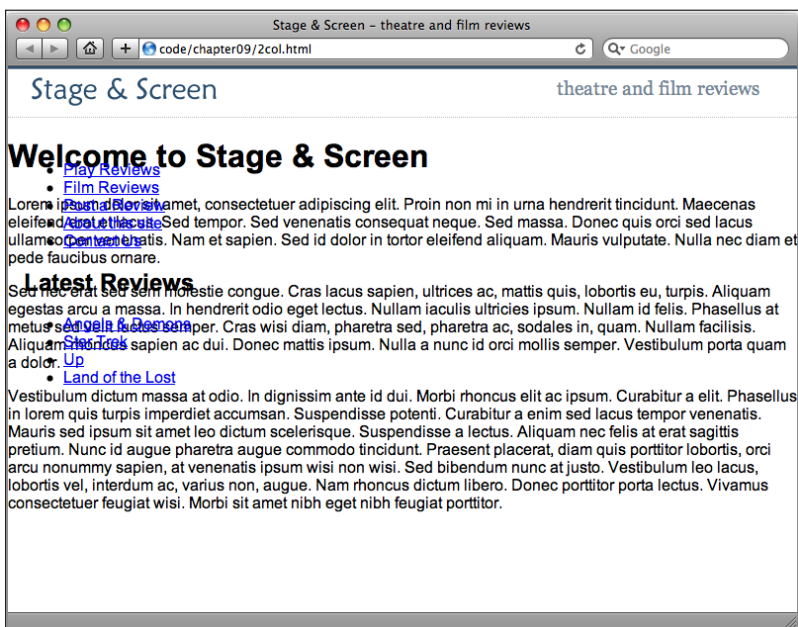


Рис. 9.29. Абсолютное позиционирование меню

## Позиционирование блока с контентом

Поскольку перед нами стоит задача создания «резинового» макета, нежелательно задавать ширину области с контентом, но, впрочем, это в любом случае не обязательно. Однако в настоящее время область с контентом размещается там, где должно быть меню. Чтобы все встало на свои места, достаточно задать для области с контентом свойство `margin`, освободив пространство для меню. Результат показан на рис. 9.30.

```
#content {
  margin-left: 16em;
  margin-right: 2em;
}
```

Теперь все элементы расположены аккуратно по отношению друг к другу, и можно перейти к заданию стиля каждого из них с помощью CSS, чтобы получить результат, изображенный на рис. 9.27. Использованная таблица стилей приведена полностью в начале раздела.

### Совет

**Использование `em` для позиционирования блоков с текстом.** В качестве единиц измерения для позиционирования элементов документа я использовала `em`. Это обеспечит пропорциональное изменение размеров блока при изменении пользователем размера шрифта, позволяя избежать перекрытия элементов. Такое решение оптимально для задания ширины блоков и отступов на



сайте, содержащем много текстовой информации. Однако если дизайн сайта подразумевает использование большого количества изображений, этот подход следует использовать с осторожностью, ведь изображения не изменяют свой размер наряду с текстом. Если вам необходим точный контроль за расположением элементов на странице, возможно, следует использовать в качестве единиц измерения пиксели.



Рис. 9.30. Добавление внешних отступов для области с контентом

## Изменение расположения элементов макета на противоположное, чтобы меню было справа

Можно ли с помощью метода, описанного в разделе «Создание «резинового» макета с двумя колонками, в котором слева расположено меню, а справа – основная область с контентом», создать макет, в котором меню расположено справа?

### Решение

Конечно, это можно сделать с помощью того же метода! Для этого нужно переместить меню слева направо и задать для области с контентом большой внешний отступ справа, чтобы освободить для меню место. Результат показан на рис. 9.31.

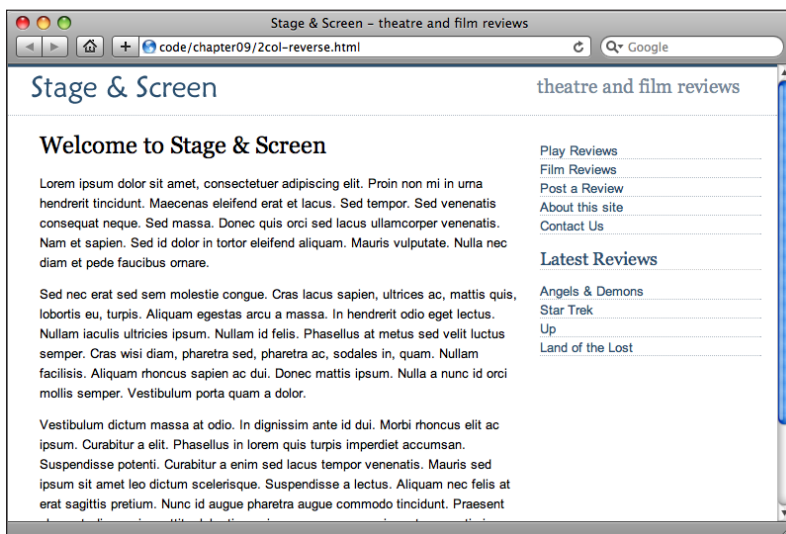


Рис. 9.31. Создание двухколоночного макета с меню справа

## Обсуждение

Чтобы разместить меню справа, не нужно вносить никаких изменений в разметку документа — достаточно всего лишь изменить свойства, задающие позиционирование элемента `nav` и внешние отступы для элемента `content`:

*chapter09/2col-reverse.css*

```
#nav {
    position: absolute;
    top: 5em;
    right: 1em;
    width: 14em;
}
#content {
    margin-left: 2em;
    margin-right: 16em;
}
```

В этом случае преимущества использования абсолютного позиционирования налицо. Неважно, где наше меню находится в разметке документа — при использовании абсолютного позиционирования оно будет удалено из основного потока элементов, и его можно разместить в любом месте страницы. Это очень удобно с точки зрения доступности сайта, поскольку разработчик может разместить менее важные элементы (например, список ссылок на другие сайты, рекламу и т. д.) в конце кода документа. Таким образом, пользователям экранных дикторов не

придется тратить время на прослушивание этой маловажной информации при каждом обращении к странице. В то же время для достижения желаемого визуального эффекта вы можете позиционировать элементы в любом месте страницы.

## Макет фиксированной ширины с двумя колонками по центру страницы

С помощью CSS можно создать макет с двумя колонками, расположенными внутри центрированного блока.

### Решение

Создание центрированного макета фиксированной ширины, состоящего из двух колонок, – чуть более сложная задача, чем создание фиксированного макета, выровненного по левому краю, или «резинового» макета, поскольку при этом отсутствует абсолютная точка отсчета справа или слева, которую можно использовать для позиционирования элементов по горизонтали. Тем не менее есть несколько способов решения этой проблемы, с помощью которых можно создать страницу, подобную той, что мы увидим на иллюстрации ниже.

Вне зависимости от избранного вами метода HTML-код будет одинаковым:

#### *chapter09/2col-fixedwidth.html*

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
  <head>
    <title>Recipe for Success | Perfect Pizza</title>
    <link href="2col-fixedwidth.css" rel="stylesheet"
      type="text/css" />
    <meta http-equiv="content-type"
      content="text/html; charset=utf-8" />
  </head>
  <body>
    <div id="wrapper">
      <div id="header">
        <h1>Perfect Pizza</h1>
      </div>
      <div id="content">
        <h2>Choosing Your Toppings</h2>
        <p>Sed nec erat sed sem molestie congue. Cras lacus ...</p>
        :
      </div>
      <div id="nav">
        <ul>
```

```
<li><a href="#">Prepare the Dough</a></li>
<li class="cur"><a href="#">Choose Your Toppings</a></li>
<li><a href="#">Pizza Ovens</a></li>
<li><a href="#">Side Salads</a></li>
</ul>
</div>
</div>
</body>
</html>
```

**Для решения поставленной задачи проще всего разместить область с контентом и навигационное меню внутри централизованного блока с помощью абсолютного и относительного позиционирования соответственно:**

#### *chapter09/2col-fixedwidth.css*

```
body {
  margin: 0;
  padding: 0;
  font-family: Arial, Helvetica, sans-serif;
  background-color: #FFFFFF;
}
#wrapper {
  position: relative;
  width: 760px;
  margin-right: auto;
  margin-left: auto;
  margin-bottom: 1em;
}
#header {
  background-image: url(kitchen.jpg);
  background-repeat: no-repeat;
  height: 200px;
  position: relative;
}
#header h1 {
  margin: 0;
  padding: 0.3em 10px 0.3em 0;
  text-align: right;
  width: 750px;
  font-weight: normal;
  color: #FFFFFF;
  font-size: 190%;
  position: absolute;
  bottom: 0;
  left: 0;
  background-image: url(black80percent.png);
}
#content {
  margin-left: 250px;
```

```
width: 500px;
padding: 0 10px 0 0;
}
#content h2 {
font-size: 120%;
color: #3333FF;
background-color: transparent;
margin: 0;
padding: 1.4em 0 0 0;
}
#content p {
font-size: 80%;
line-height: 1.6;
}
#nav {
position: absolute;
top: 200px;
left: 0;
width: 230px;
}
#nav ul {
list-style: none;
margin: 3em 0 0 0;
padding: 0;
border: none;
}
#nav li {
font-size: 85%;
}
#nav a:link, #nav a:visited {
color: #555555;
background-color: transparent;
display: block;
padding: 1em 0 0 10px;
text-decoration: none;
min-height: 40px;
}
#nav a:hover, #nav li.cur a:link, #nav li.cur a:visited {
color: #FFFFFF;
background-image: url(arrow.gif);
background-repeat: no-repeat;
}
```

Как видно на рис. 9.32, в результате получится достаточно простой макет. Если вы хотите усложнить дизайн страницы, добавив некоторые декоративные элементы – например, задав фоновое изображение для области с контентом или рамку для всего макета, – придется использовать другой метод.

В качестве альтернативного варианта можно просто задать свойство `float` для меню навигации и для блока с контентом, чтобы они распола-

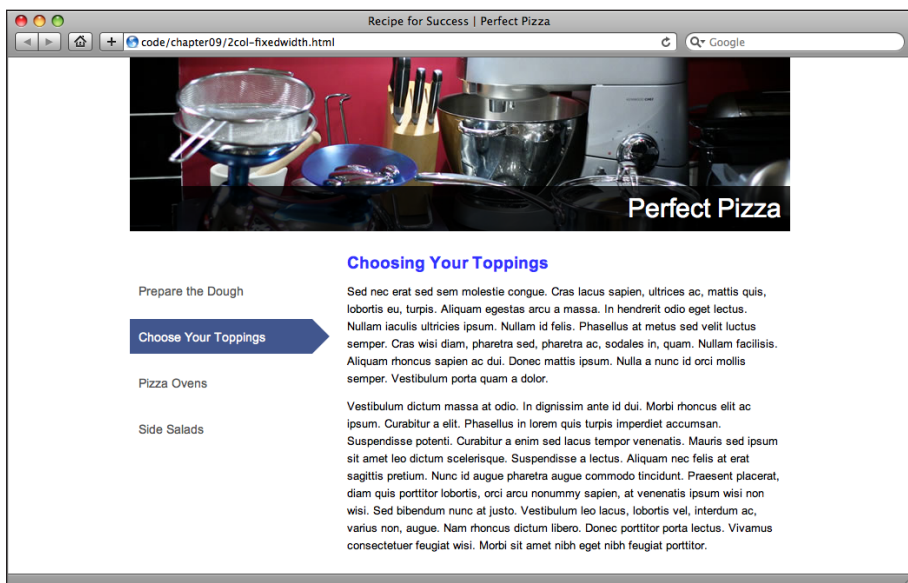


Рис. 9.32. Центрированный макет с фиксированной шириной

гались с левой и правой стороны центрированного блока соответственно. Использование данного свойства дает разработчику возможность гибкого внесения дальнейших изменений в макет, например, если затем потребуется добавить блок в нижнюю часть страницы (для этого достаточно задать для него свойство `clear` со значением `both`, чтобы он располагался под двумя колонками независимо от их высоты, что невозможно при их абсолютном позиционировании) или задать для макета обрамление. Воспользовавшись преимуществами данного способа, мы добавили рамку вокруг макета:

*chapter09/2col-fixedwidth-float.css*

```
body {
    margin: 0;
    padding: 0;
    font-family: Arial, Helvetica, sans-serif;
    background-color: #FFFFFF;
}
#wrapper {
    position: relative;
    width: 760px;
    margin-right: auto;
    margin-left: auto;
    margin-bottom: 1em;
    background-image: url(sidebar.gif);
    background-repeat: repeat-y;
```

```
border-right: 1px solid #888888;
border-bottom: 1px solid #888888;
}
#header {
background-image: url(kitchen.jpg);
background-repeat: no-repeat;
height: 200px;
position: relative;
}
#header h1 {
margin: 0;
padding: 0.3em 10px 0.3em 0;
text-align: right;
width: 750px;
font-weight: normal;
color: #FFFFFF;
font-size: 190%;
position: absolute;
bottom: 0;
left: 0;
background-image: url(black80percent.png);
}
#content {
float: right;
width: 500px;
padding: 0 10px 0 0;
}
#content h2 {
font-size: 120%;
color: #3333FF;
background-color: transparent;
margin: 0;
padding: 1.4em 0 0 0;
}
#content p {
font-size: 80%;
line-height: 1.6;
}
#nav {
float: left;
width: 230px;
}
#nav ul {
list-style: none;
margin: 3em 0 0 0;
padding: 0;
border: none;
}
#nav li {
```

```

    font-size: 85%;
  }
  #nav a:link, #nav a:visited {
    color: #555555;
    background-color: transparent;
    display: block;
    padding: 1em 0 0 10px;
    text-decoration: none;
    min-height: 40px;
  }
  #nav a:hover, #nav li.cur a:link, #nav li.cur a:visited {
    color: #FFFFFF;
    background-image: url(arrow.gif);
    background-repeat: no-repeat;
  }
  #footer {
    clear: both;
    font-size: 80%;
    padding: 1em 0 1em 0;
    margin-left: 250px;
    color: #999999;
    background-color: transparent;
  }
}

```

Результат обработки данного кода показан на рис. 9.33.

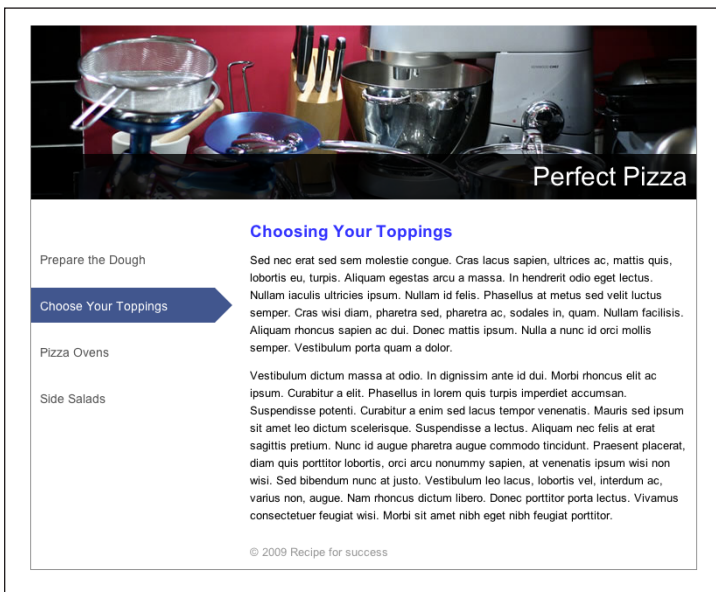


Рис. 9.33. Создание центрированного макета фиксированной ширины с рамкой с использованием свойства *float*



## Обсуждение

Чтобы не отвлекаться от основной темы, не будем заниматься заданием свойств для цветового оформления, размера и типа шрифта, сконцентрировав все внимание на методах создания макета.

В обеих версиях макета мы будем использовать центрированный `div`, как в примерах, с которыми мы работали в разделе «Центрирование блока на странице». Для него задан ID со значением `wrapper`:

*chapter09/2col-fixedwidth.css или chapter09/2col-fixedwidth-float.css (фрагмент)*

```
body {
    margin: 0;
    padding: 0;
    :
}

#wrapper {
    width: 760px;
    margin-right: auto;
    margin-left: auto;
    :
}
```

Результат применения этих стилей показан на рис. 9.34.



Рис. 9.34. Центрирование контента на странице

В разделе «Создание «резинового» макета с двумя колонками, в котором слева расположено меню, а справа – основная область с контентом» мы узнали, что для расположения меню навигации можно использо-

вать абсолютное позиционирование, а во избежание его перекрытия с областью контента последней следует задать большой внешний отступ. В данном случае единственное отличие от того примера состоит в том, что навигационное меню должно располагаться внутри централизованного блока `wrapper`, поэтому его нельзя позиционировать на странице абсолютно.

При задании элементу свойства `position` вместо значения `absolute` можно указать `relative` – при этом элемент не будет удален из общего потока; вместо этого он будет смещен по отношению к своей позиции на странице по умолчанию (однако если никаких параметров смещения указано не будет, он будет отображен как если бы свойство `position` не было задано). Однако в отличие от элементов, для которых свойство `position` не задано, относительно позиционированный элемент послужит в качестве **системы координат** для размещения любого дочернего элемента с абсолютным позиционированием.

По сути, позиция элемента, для которого задано свойство `position: absolute`, расположенного внутри элемента со свойством `position: relative`, будет рассчитываться по отношению к краям родительского элемента. Именно это нам и нужно для позиционирования меню навигации внутри централизованного блока в данном примере.

Прежде всего следует задать элементу `wrapper` свойство `position` со значением `relative`:

*chapter09/2col-fixedwidth.css (фрагмент)*

```
#wrapper {
  position: relative;
  text-align: left;
  width: 760px;
  margin-right: auto;
  margin-left: auto;
  :
}
}
```

Затем позиционируем блок с навигацией абсолютно:

*chapter09/2col-fixedwidth.css (фрагмент)*

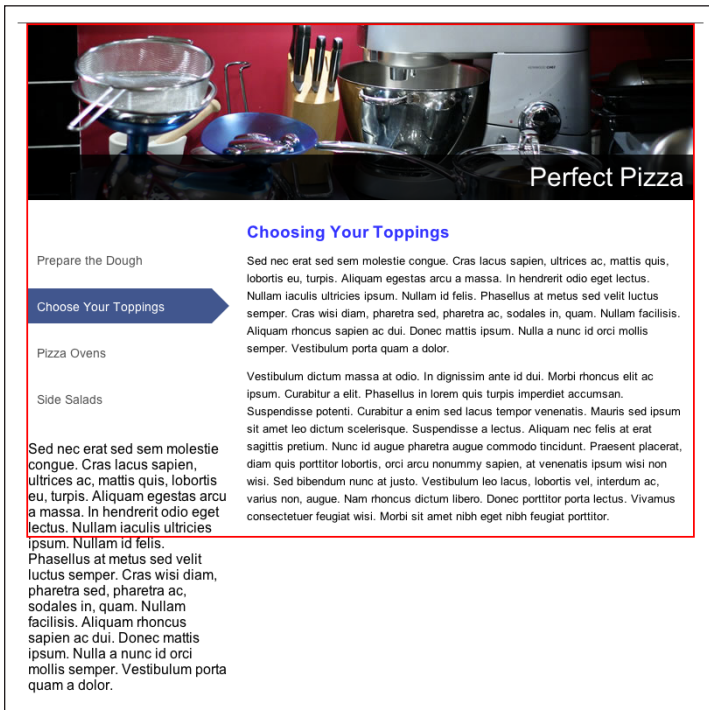
```
#nav {
  position: absolute;
  top: 200px;
  left: 0;
  width: 230px;
}
```

Наконец, добавим для области с основным контентом внешний отступ, чтобы освободить место для только что размещенного меню навигации:

*chapter09/2col-fixedwidth.css (фрагмент)*

```
#content {
  margin-left: 250px;
  padding: 0 10px 0 0;
}
```

Такой макет выглядит идеально при условии, что область с основным контентом занимает больше пространства по вертикали, чем панель навигации. К сожалению, абсолютно позиционированный блок `nav` не может оказывать никакого влияния на высоту блока `wrapper`, поэтому если область с контентом будет короче панели навигации, блок `wrapper` не сможет вместить все содержащиеся в панели навигации ссылки. Чтобы убедиться в этом на практике, можно добавить для элемента `wrapper` цветную рамку шириной в два пиксела, а на панель навигации добавить больше текста, чтобы она превысила область с контентом по высоте. На рис. 9.35 видно, что содержимое панели навигации выходит за пределы элемента `wrapper`.



**Рис. 9.35.** Содержимое панели навигации выходит за пределы блока «wrapper»

Другое решение поставленной задачи состоит в использовании плавающих блоков. Его сложнее реализовать, но оно позволяет преодолеть указанное ограничение предыдущего метода, и вы сможете добавлять

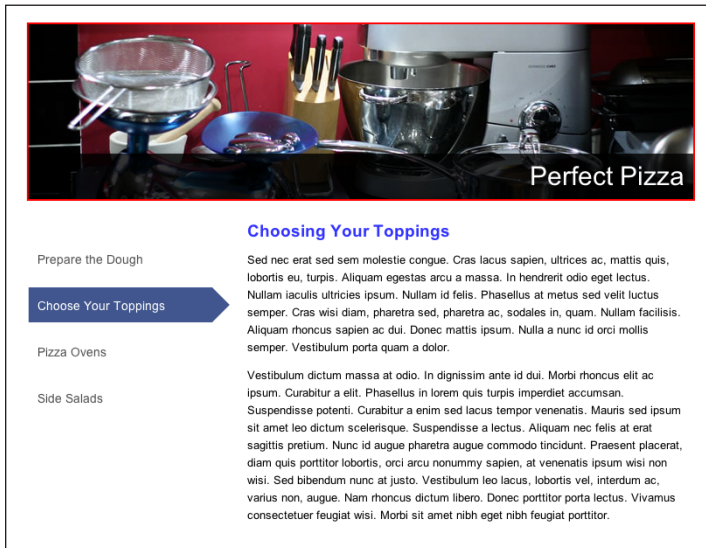
элементы в нижнюю часть страницы вне зависимости от того, какая из колонок длиннее. Прежде всего разместим блок навигации слева, а блок content – справа с помощью свойства float:

*chapter09/2col-fixedwidth-float.css (фрагмент)*

```
#content {
  float: right;
  width: 500px;
  padding: 0 10px 0 0;
}

#nav {
  float: left;
  width: 230px;
}
```

Данный код позволяет получить тот же результат, что и при использовании рассмотренного выше метода. Однако теперь при добавлении цветной рамки для элемента wrapper вы увидите, что она обрамляет только «шапку» страницы. Оберткой wrapper больше ничего не обернуто!



**Рис. 9.36.** Расположение плавающих блоков справа и слева

Основной причиной использования плавающих блоков послужила необходимость добавления блока в нижнюю часть страницы. Для этого в код документа под плавающими блоками нужно добавить следующее:

*chapter09/2col-fixedwidth-float.html (фрагмент)*

```
<div id="nav">
  <ul>
```

```

    <li><a href="#">Prepare the Dough</a></li>
    <li class="cur"><a href="#">Choose Your Toppings</a></li>
    <li><a href="#">Pizza Ovens</a></li>
    <li><a href="#">Side Salads</a></li>
  </ul>
</div>
<div id="footer">&copy; 2009 Recipe for success</div>
</div>
</body>
</html>

```

После обновления страницы вы увидите, что теперь рамка блока `wrapper` окружает весь контент страницы, как показано на рис. 9.37. Дело в том, что после удаления плавающих блоков из основного потока элементов там остался единственный элемент – блок `footer`, отодвинутый плавающими блоками в левый нижний угол блока `wrapper`.

Если для блока `footer` задать свойство `clear` со значением `both`, то он будет размещен ниже обоих плавающих блоков, и потому `wrapper` вместит и панель навигации, и контент, причем неважно, какой из элементов окажется длиннее. Текущий вид страницы показан на рис. 9.38:

*chapter09/2col-fixedwidth-float.css (фрагмент)*

```

#footer {
  clear: both;
  :
  }

```

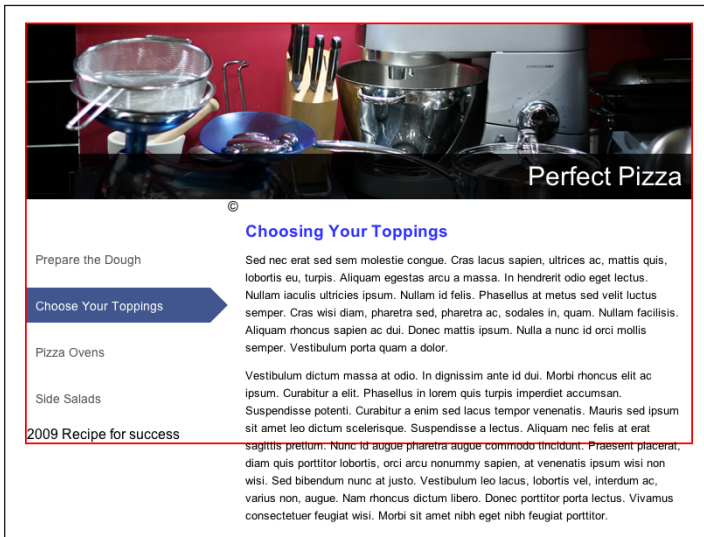


Рис. 9.37. Страница после добавления блока `footer`

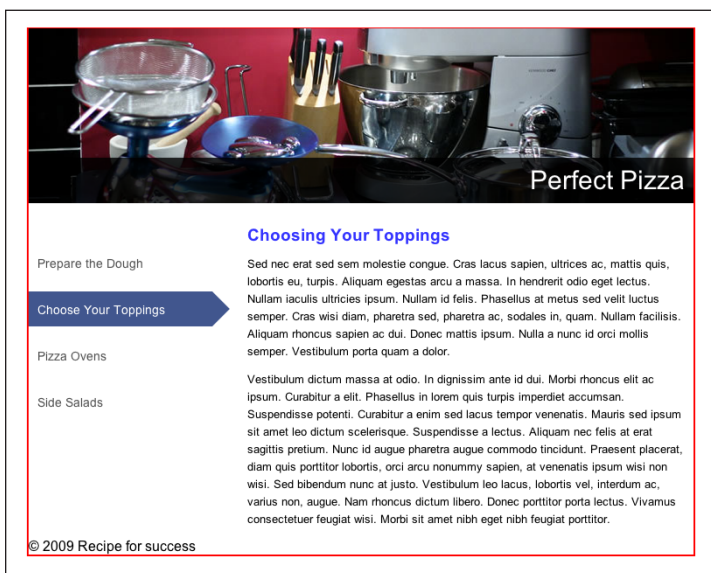


Рис. 9.38. Для блока footer задано свойство `clear: both`

## Создание колонки, занимающей все доступное пространство по высоте

Если вы когда-нибудь пробовали задать фон для боковой колонки, подобной той, что мы создали в разделе «Создание макета фиксированной ширины с двумя колонками, расположенного по центру страницы», то наверняка убедились в невозможности задания для нее той же высоты, что и у расположенной рядом более длинной колонки, из-за чего добавленный фон выглядит немного странно. К примеру, при задании фонового цвета для блока с панелью навигации он займет по высоте только область, в которой расположены ссылки, а не всю область с контентом, как показано на рис. 9.39.

### Решение

Для решения этой проблемы можно добавить фоновое изображение к элементу страницы, длина которого совпадает с длиной второй колонки, а ширина соответствует ширине панели навигации. При этом создается впечатление, будто фон принадлежит самой панели навигации. В нашем случае можно добавить фоновое изображение для элемента `wrapper`, как показано на рис. 9.40:

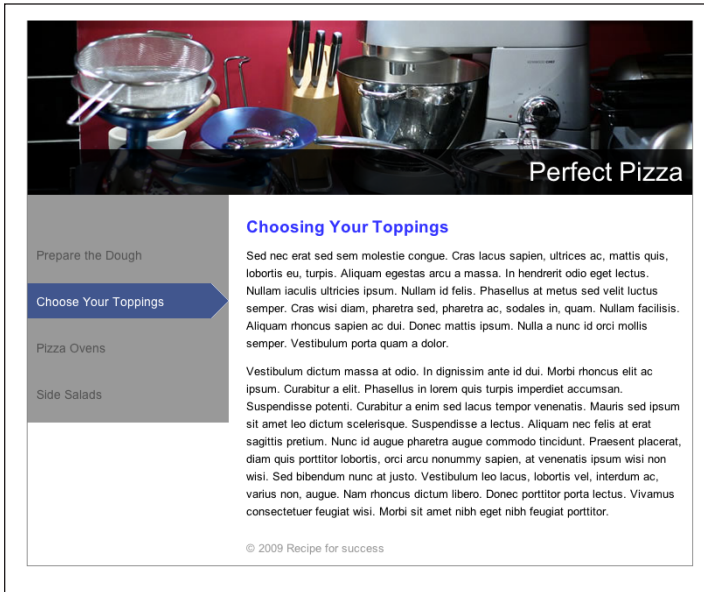


Рис. 9.39. Серый фоновый цвет окрашивает только область со ссылками

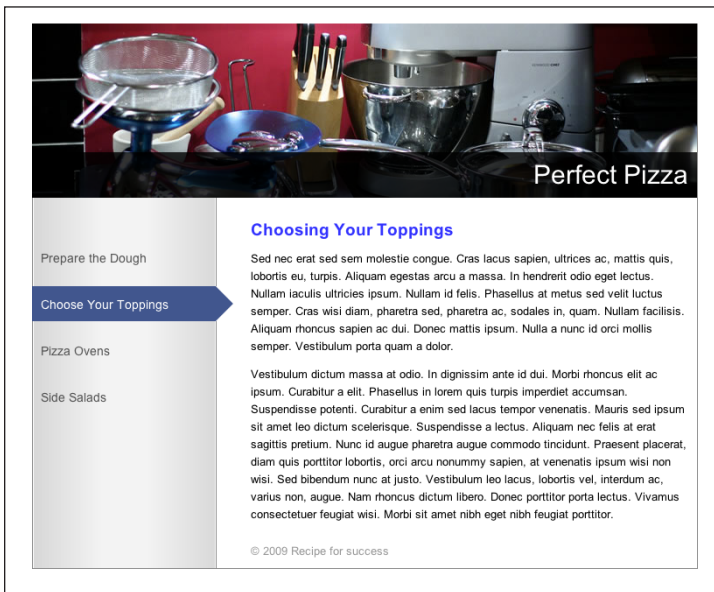


Рис. 9.40. Фон как бы привязан к панели навигации

*chapter09/2col-fixedwidth-float.css (фрагмент)*

```
#wrapper {
  position: relative;
  text-align: left;
  width: 760px;
  margin-right: auto;
  margin-left: auto;
  margin-bottom: 1em;
  background-image: url(sidebar.gif);
  background-repeat: repeat-y;
  border-right: 1px solid #888888;
  border-bottom: 1px solid #888888;
}
```

## Обсуждение

Этот несложный прием можно с успехом применять при создании различных макетов страниц. В данном примере фоновый цвет был задан для блока `wrapper`, поскольку фон должен простирается вниз до конца области с контентом; изображение «шапки» закрывает фоновое изображение панели навигации в верхней части страницы. Аналогичным образом можно сделать так, чтобы фоновое изображение распространялось до нижнего блока или после определенной области контента: для этого достаточно задать фоновое изображение для элемента, содержащего необходимую область.

### Совет

**Использование фонового градиента.** В данном примере было использовано одинаковое повторяющееся изображение, однако вместо него можно было бы использовать вертикальное изображение с градиентным переходом, плавно сливающееся с цветом фона страницы, задав для него значение `no-repeat`.

## Добавление тени к блоку

На многих страницах можно встретить элементы, отбрасывающие тень, – чаще всего это область размещения основного контента. Добавим такую тень к макету фиксированной ширины, подобному тому, с которым мы работали в разделе «Создание колонки, занимающей все доступное пространство по высоте».

### Решение

Для добавления тени к краям блока нам потребуется два изображения: одно послужит в качестве фона, другое создаст тень внизу. На рис. 9.41 показан создаваемый эффект.



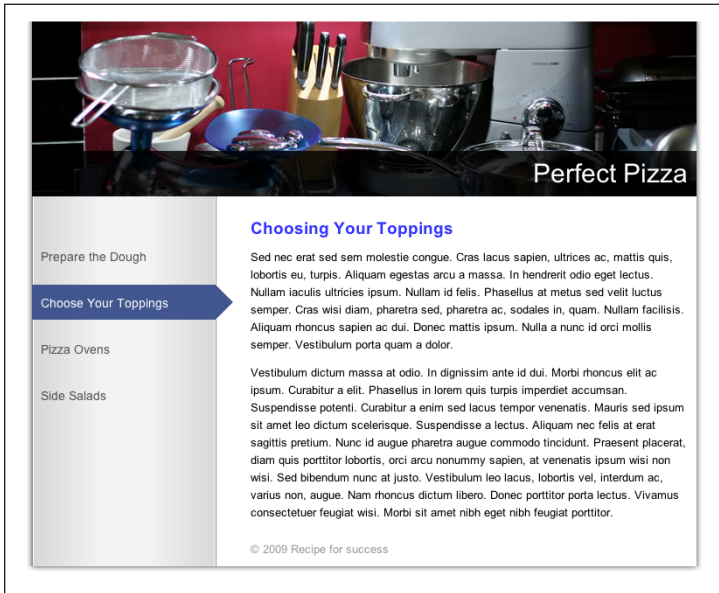


Рис. 9.41. Блок, отбрасывающий тень

Для создания такого эффекта нужно добавить в разметку документа дополнительный код, который послужит каркасом для размещения изображений.

Первое изображение с именем **shadow-bg.jpg**, показанное на рис. 9.42, послужит в качестве фонового изображения для блока `div` с ID `wrapper`. Оно будет создавать тень по бокам по всей высоте страницы. Второе изображение, **shadow-bottom.jpg**, будет использовано для создания тени в нижней части блока.

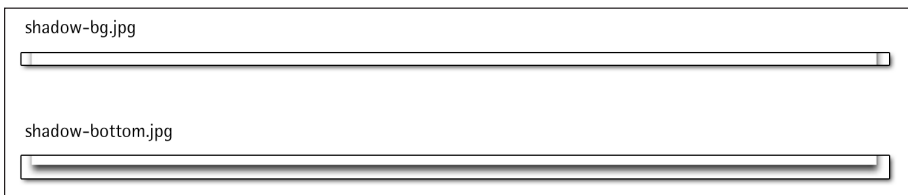


Рис. 9.42. Файлы для создания эффекта тени

Ширина блока `wrapper` увеличена на 20 пикселей, поскольку я хочу, чтобы размер области с контентом оставался прежним, но сам контент не соприкасался с ее краями. Кроме того, я добавила первое изображение в качестве фонового для данного элемента:

*chapter09/2col-fixedwidth-shadow.css (фрагмент)*

```
#wrapper {
  position: relative;
  text-align: left;
  width: 780px;
  margin-right: auto;
  margin-left: auto;
  background-image: url(shadow-bg.jpg);
  background-repeat: repeat-y;
}
```

Затем создадим дополнительный блок `div` с именем `wrapper2`, содержащий элементы `content`, `nav` и `footer` и расположенный внутри блока `wrapper`.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
  <head>
    <title>Recipe for Success | Perfect Pizza</title>
    <link href="2col-fixedwidth-shadow.css" rel="stylesheet"
      type="text/css" />
    <meta http-equiv="content-type"
      content="text/html; charset=utf-8" />
  </head>
  <body>
    <div id="wrapper"><div id="wrapper2">
      <div id="header">
        <h1>Perfect Pizza</h1>
      </div>
      <div id="content">
        <h2>Choosing Your Toppings</h2>
        <p>Sed nec erat sed sem molestie congue. Cras lacus ...</p>
        <p>Vestibulum dictum massa at odio. In dignissim ...</p>
      </div>
      <div id="nav">
        <ul>
          <li><a href="#">Prepare the Dough</a></li>
          <li class="cur"><a href="#">Choose Your Toppings</a></li>
          <li><a href="#">Pizza Ovens</a></li>
          <li><a href="#">Side Salads</a></li>
        </ul>
      </div>
      <div id="footer">&copy; 2009 Recipe for success</div>
    </div></div>
  </body>
</html>
```

Для добавленного блока задано фоновое изображение, отображаемое в области панели навигации (вместо внешнего `wrapper`). Кроме того, ему задано свойство `padding`, чтобы немного отодвинуть контент от его края:

*chapter09/2col-fixedwidth-shadow.css (фрагмент)*

```
#wrapper2 {
    background-image: url(sidebar.gif);
    background-repeat: repeat-y;
    margin: 0 10px 0 10px;
}
```

Наконец, нужно добавить изображение для создания тени снизу. Для этого создадим еще один блок `div` с идентификатором `btm` прямо перед закрывающим тегом `</div>` внешнего блока `wrapper`.

*chapter09/2col-fixedwidth-shadow.html (фрагмент)*

```
<div id="footer">&copy; 2009 Recipe for success</div>
</div>
<div id="btm"></div></div>
</body>
</html>
```

Теперь достаточно задать фоновое изображение для данного блока и задать блоку высоту, равную высоте изображения.

*chapter09/2col-fixedwidth-shadow.css (фрагмент)*

```
#btm {
    background-image: url(shadow-bottom.jpg);
    background-repeat: no-repeat;
    display: block;
    height: 13px;
}
```

Вот и все, тень готова!

## Создание макета с тремя колонками средствами CSS

Дизайн многих страниц предполагает разделение пространства на три части. Как видно на рис. 9.43, одна колонка содержит панель навигации, другая – основной контент, а третья – дополнительную информацию, например рекламные объявления. Рассмотрим способ создания такого макета с помощью CSS.

### Решение

Для создания «резинового» макета с тремя колонками используется тот же простой метод, который мы использовали в разделе «Создание «резинового» макета с двумя колонками, в котором слева расположено меню, а справа – основная область с контентом»:

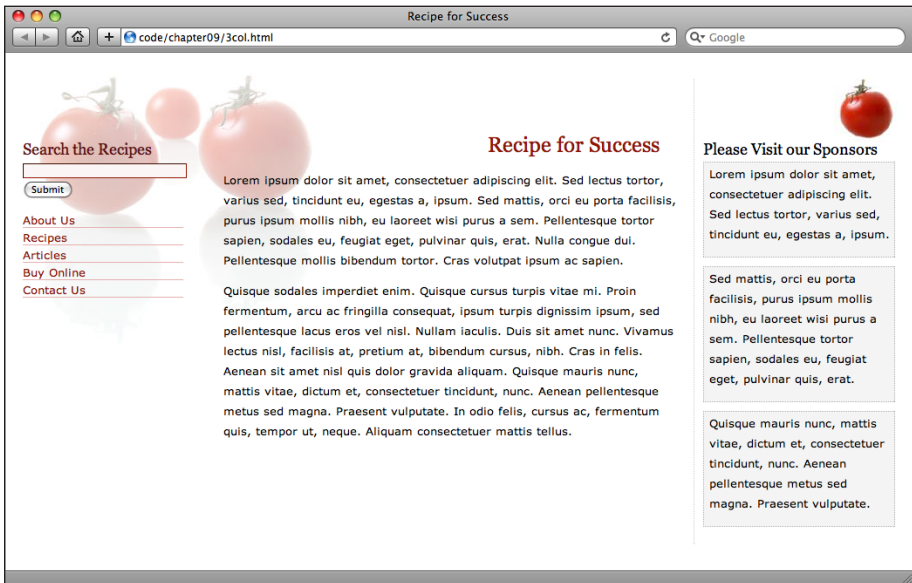


Рис. 9.43. Макет из трех колонок, созданный с помощью CSS

### chapter09/3col.html

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
  <head>
    <title>Recipe for Success</title>
    <meta http-equiv="content-type"
      content="text/html; charset=utf-8" />
    <link rel="stylesheet" type="text/css" href="3col.css" />
  </head>
  <body>
    <div id="content">
      <h1>Recipe for Success</h1>
      <p>Lorem ipsum dolor sit amet, consectetur adipiscing ...</p>
      <p>Quisque sodales imperdiet enim. Quisque cursus ...</p>
    </div>
    <div id="side1">
      <form method="post" action="" id="searchform">
        <h3><label for="keys">Search the Recipes</label></h3>
        <div>
          <input type="text" name="keys" id="keys" class="txt" />
          <br />
          <input type="submit" name="Submit" value="Submit" />
        </div>
      </form>
    </div>
  </body>
</html>

```

```

        <ul>
          <li><a href="#">About Us</a></li>
          <li><a href="#">Recipes</a></li>
          <li><a href="#">Articles</a></li>
          <li><a href="#">Buy Online</a></li>
          <li><a href="#">Contact Us</a></li>
        </ul>
      </div>
    <div id="side2">
      <h3>Please Visit our Sponsors</h3>
      <div class="adbox"><p>Lorem ipsum dolor sit amet ...</p></div>
      <div class="adbox"><p>Sed mattis, orci eu porta ...</p></div>
      <div class="adbox"><p>Quisque mauris nunc, mattis ...</p></div>
    </div>
  </body>
</html>

```

### *chapter09/3col.css*

```

body {
  margin: 0;
  padding: 0;
  background-image: url(tomato_bg.jpg);
  background-repeat: no-repeat;
  background-color: #FFFFFF;
}
p {
  font: 80%/1.8em Verdana, Geneva, Arial, Helvetica, sans-serif;
  padding-top: 0;
  margin-top: 0;
}
form {
  margin: 0;
  padding: 0;
}
#content {
  margin: 66px 260px 0px 240px;
  padding: 10px;
}
#content h1 {
  text-align: right;
  padding-right: 20px;
  font: 150% Georgia, "Times New Roman", Times, serif;
  color: #901602;
}
#side1 {
  position: absolute;
  width: 200px;
  top: 30px;
  left: 10px;
  padding: 70px 10px 10px 10px;
}

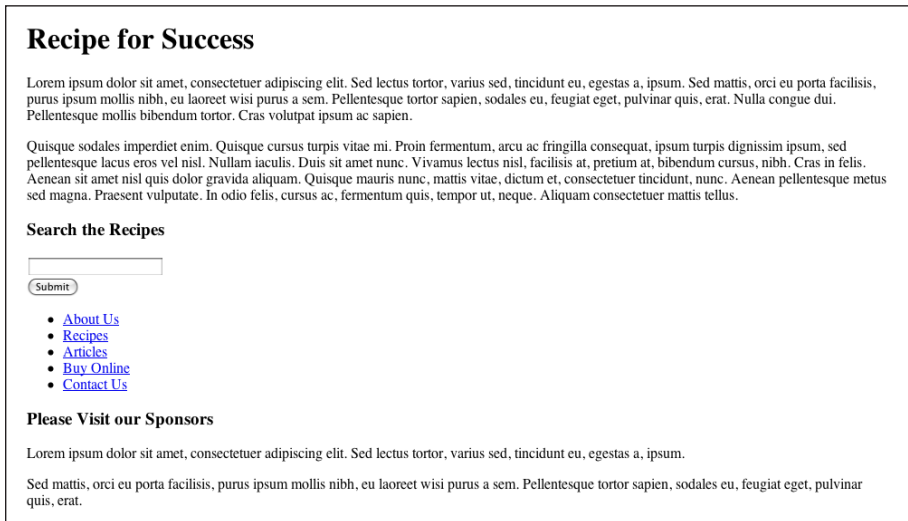
```

```
#side2 {
  position: absolute;
  width: 220px;
  top: 30px;
  right: 10px;
  padding: 70px 10px 10px 10px;
  border-left: 1px dotted #CCCCCC;
  background-image: url(sm-tomato.jpg);
  background-position: top right;
  background-repeat: no-repeat;
}
#side2 h3 {
  font: 110% Georgia, "Times New Roman", Times, serif;
  margin: 0;
  padding-bottom: 4px;
}
.adbox {
  padding: 2px 4px 2px 6px;
  margin: 0 0 10px 0;
  border: 1px dotted #B1B1B1;
  background-color: #F4F4F4;
}
#side1 h3 {
  font: 110% Georgia, "Times New Roman", Times, serif;
  color: #621313;
  background-color: transparent;
  margin: 0;
  padding-bottom: 4px;
}
#side1 .txt {
  width: 184px;
  background-color: #FCF5F5;
  border: 1px inset #901602;
}
#side1 ul {
  list-style: none;
  margin-left: 0;
  padding-left: 0;
  width: 184px;
}
#side1 li {
  font: 80% Verdana, Geneva, Arial, Helvetica, sans-serif;
  margin-bottom: 0.3em;
  border-bottom: 1px solid #E7AFAF;
}
#side1 a:link, #side1 a:visited {
  text-decoration: none;
  color: #901602;
  background-color: transparent;
}
```

```
#side1 a:hover {
  color: #621313;
}
```

## Обсуждение

Создать такой макет совсем не сложно. За основу мы возьмем неформатированный документ, показанный на рис. 9.44, который содержит три блока `div`: один с идентификатором `content`, второй с идентификатором `side1` и третий с идентификатором `side2`.



*Рис. 9.44. Неформатированный XHTML-документ*

Три колонки были созданы с помощью приведенного ниже CSS-кода. Левая и правая колонки позиционированы абсолютно: `side1` размещается в левой части страницы, а `side2` — в правой. Кроме того, для них задан большой отступ сверху, чтобы освободить пространство для фоновых изображений, которые будут выступать в качестве заголовков:

*chapter09/3col.css (фрагмент)*

```
#side1 {
  position: absolute;
  width: 200px;
  top: 30px;
  left: 10px;
  padding: 70px 10px 10px 10px;
}
```

*chapter09/3col.css (фрагмент)*

```
#side2 {
  position: absolute;
```

```
width: 220px;
top: 30px;
right: 10px;
padding: 70px 10px 10px 10px;
:
}
```

Блок `content` расположен между двумя абсолютно позиционированными колонками, для которых освобождено достаточно пространства путем задания для `content` внешних отступов с помощью свойства `margin`:

*chapter09/3col.css (фрагмент)*

```
#content {
margin: 66px 260px 0px 240px;
padding: 10px;
}
```

На рис. 9.45 показан вид страницы после задания свойств для позиционирования элементов.

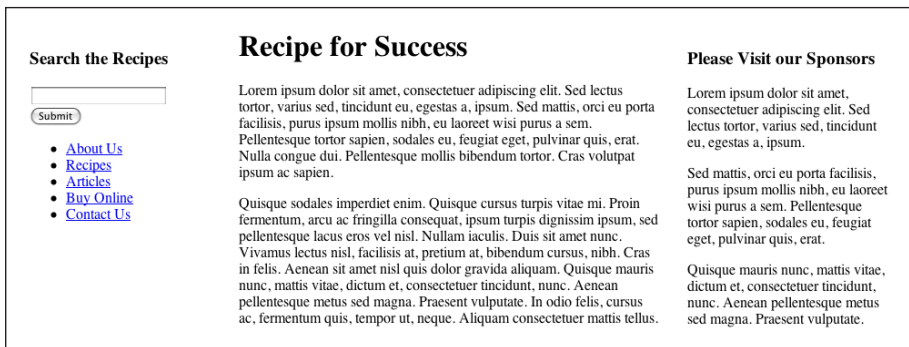


Рис. 9.45. Три колонки, позиционированные с помощью CSS

Теперь, когда все колонки на месте, можно задать стиль оформления для отдельных элементов, чтобы они соответствовали дизайну страницы. Я добавила фоновое изображение с томатами для элементов `body` и `side2`, в результате чего получилась страница, которую мы уже видели на рис. 9.43.

## Добавление к «резиновому» макету нижнего блока

Если вы уже каким-то образом экспериментировали с абсолютным позиционированием, то у вас наверняка закралось подозрение, что верстка макета, основанная на абсолютном позиционировании, не позволит добавить нижний блок, который будет отображаться ниже всех



трех колонок независимо от их высоты. Надо признать, чутье вас не подвело.

Чтобы добавить нижний блок к такому трехколоночному макету, нужно использовать плавающие блоки. Однако при создании «резинового» макета с плавающими блоками возникают дополнительные проблемы в отличие от использования аналогичного макета фиксированной ширины. При задании элементу свойства `float` ему нужно задать также и ширину. Если нам известна ширина всего макета, это не представляет никаких сложностей. При работе с «резиновым» макетом, как в разделе «Создание макета с тремя колонками средствами CSS», у нас есть две колонки, ширина которых нам известна (боковых панелей), однако ширину области с контентом мы не знаем – она меняется, заполняя все свободное пространство.

## Решение

Для решения проблем, возникающих при наличии колонки с переменными размерами в «резиновом» макете, придется немного усложнить макет страницы, используя отрицательные отступы для освобождения пространства для размещения колонки с фиксированной шириной в гибко изменяющей свои размеры области с контентом. Кроме того, в разметку документа необходимо добавить следующий код для создания самих плавающих элементов:

### *chapter09/3col-alt.html*

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
  <head>
    <title>Recipe for Success</title>
    <meta http-equiv="content-type"
      content="text/html; charset=utf-8" />
    <link rel="stylesheet" type="text/css" href="3col-alt.css" />
  </head>
  <body>
    <div id="wrapper">
      <div id="content">
        <div id="side1">
          <form method="post" action="" id="searchform">
            <h3><label for="keys">Search the Recipes</label></h3>
            <div>
              <input type="text" name="keys" id="keys"
                class="txt" /><br />
              <input type="submit" name="Submit" value="Submit" />
            </div>
          </form>
        </div>
        <ul>
          <li><a href="#">About Us</a></li>

```

```

        <li><a href="#">Recipes</a></li>
        <li><a href="#">Articles</a></li>
        <li><a href="#">Buy Online</a></li>
        <li><a href="#">Contact Us</a></li>
    </ul>
</div>
<div id="main">
    <h1>Recipe for Success</h1>
    <p>Lorem ipsum dolor sit amet, consectetur ...</p>
    <p>Quisque sodales imperdiet enim. Quisque ...</p>
</div>
</div>
</div>

<div id="side2">
    <h3>Please Visit our Sponsors</h3>
    <div class="adbox"><p>Lorem ipsum dolor sit amet ...</p></div>
    <div class="adbox"><p>Sed mattis, orci eu porta ...</p></div>
    <div class="adbox"><p>Quisque mauris nunc, mattis ...</p></div>
</div>
<div id="footer">
    : footer content...
</div>
</body>
</html>

```

В таблице стилей блоку `wrapper` задана ширина 100% и отрицательный внешний отступ – 230 пикселей. Благодаря использованию отрицательных отступов можно задать боковой панели изменяемую ширину (на 230 пикселей меньше окна браузера).

Затем можно разместить боковые панели справа и слева от области с контентом с помощью свойства `float`:

*chapter09/3col-alt.css (фрагмент)*

```

body {
    margin: 0;
    padding: 0;
}
#wrapper {
    width: 100%;
    float: left;
    margin-right: -230px;
    margin-top: 66px;
}
#content {
    margin-right: 220px;
}
#main {
    margin-left: 220px;
}

```

```
#side1 {
  width:200px;
  float:left;
  padding: 0 10px 0 10px;
}
#side2 {
  width: 190px;
  padding: 80px 10px 0 10px;
  float:right;
}
#footer {
  clear:both;
  border-top: 10px solid #CECECE;
}
```

Как видно на рис. 9.46, при использовании такой таблицы стилей колонки располагаются именно так, как было задумано, а нижний блок размещается прямо под всеми тремя колонками. Аналогичным образом можно создать и макет, состоящий из двух колонок: их порядок можно изменить путем изменения значения свойства `float` с `left` на `right`. Если вы немного поэкспериментируете с кодом, то добьетесь необходимого расположения элементов, даже если поначалу кажется, что эти действия совсем не логичны.

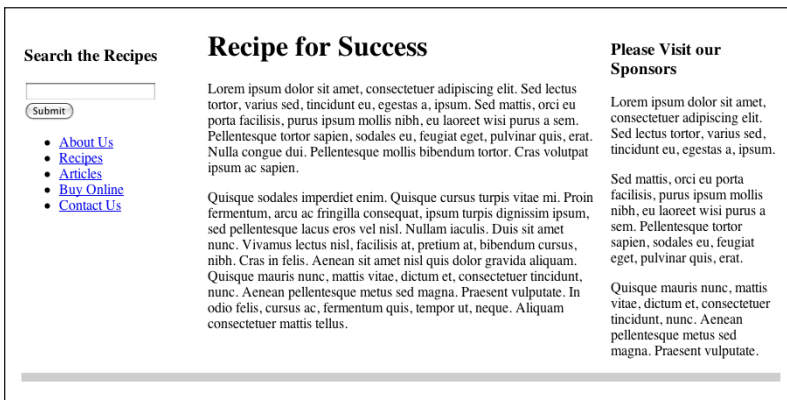


Рис. 9.46. Все колонки расположены в предназначенном для них месте

## Создание галереи миниатюр

Для отображения коллекции изображений, например фотоальбома, проще всего использовать таблицы. Однако документ, изображенный на рис. 9.47, был создан средствами CSS, что дает разработчику дополнительные преимущества, которых нет в табличной версии.

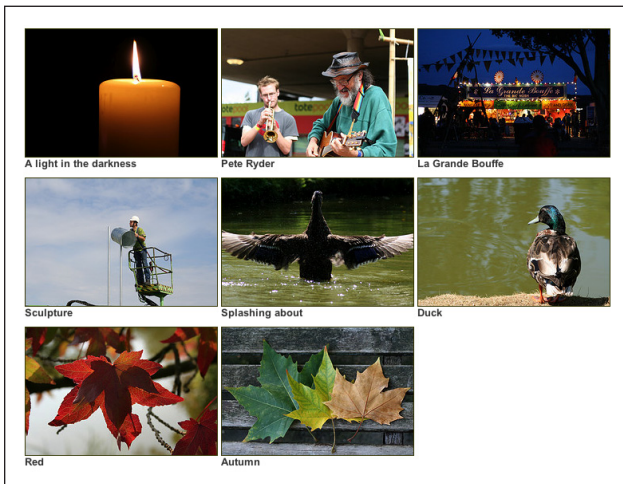


Рис. 9.47. Создание галереи изображений с помощью CSS

## Решение

Создадим простой список изображений альбома и позиционируем их с помощью CSS:

*chapter09/gallery.html*

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
  <head>
    <title>CSS photo album</title>
    <meta http-equiv="content-type"
      content="text/html; charset=utf-8" />
    <link href="gallery.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <ul id="albumlist">
      <li>A light in the darkness</li>
      <li>Pete Ryder</li>
      <li>La Grande Bouffe</li>
      <li>Sculpture</li>
      <li>Splashing about</li>
      <li>Duck</li>
    </ul>
  </body>
</html>
```

```

    <li>Red</li>
    <li>Autumn</li>
  </ul>
</body>
</html>

```

### *chapter09/gallery.css*

```

body {
  background-color: #FFFFFF;
  color: #000000;
  margin: 0;
  padding: 0;
}
#albumlist {
  list-style-type: none;
}
#albumlist li {
  float: left;
  width: 240px;
  margin-right: 6px;
  margin-bottom: 10px;
  font: bold 0.8em Arial, Helvetica, sans-serif;
  color: #333333;
}
#albumlist img {
  display: block;
  border: 1px solid #333300;
}

```

## Обсуждение

За основу альбома мы возьмем маркированный список, пунктами которого служат изображения с описаниями. Без применения CSS список будет отображен, как показано на рис. 9.48.

### *chapter09/gallery.html (фрагмент)*

```

<ul id="albumlist">
  <li>A light in the darkness</li>
  <li>Pete Ryder</li>
  <li>La Grande Bouffe</li>
  <li>Sculpture</li>
  <li>Splashing about</li>
  <li>Duck</li>

```

```

<li>Red</li>
<li>Autumn</li>
</ul>

```

Обратите внимание, что для списка изображений задан атрибут `id` со значением `albumlist`.

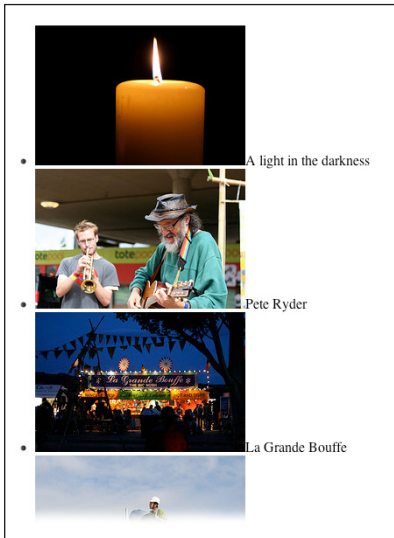


Рис. 9.48. Вид списка изображений до применения каскадных таблиц стилей

Чтобы расположить изображения по сетке, зададим свойство `float` содержащим их элементам `li`. Добавим соответствующие правила в таблицу стилей:

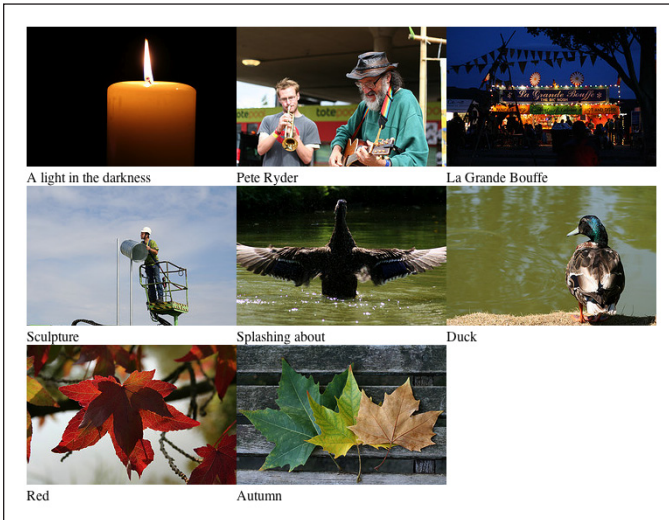
```

#albumlist {
  list-style-type: none;
}
#albumlist li {
  float: left;
  width:240px;
}
#albumlist img {
  display: block;
}

```

Добавленные правила удаляют маркеры пунктов списка и задают для них свойство `float` со значением `left`, как показано на рис. 9.49. Кроме того, задав изображениям отображение в виде блочных элементов, мы добились того, что описания располагаются под ними.

Теперь все изображения позиционированы. При изменении размеров окна вы увидите, что они заполняют все доступное пространство по ширине, а если окно становится слишком узким для того, чтобы вместить все изображения, оставшиеся изображения переходят на следующую строку.



**Рис. 9.49.** Вид страницы после задания изображениям свойства *float* со значением *left*

Каркас страницы готов – теперь можно сделать его привлекательнее. К примеру, можно добавить правила для создания зазоров между изображениями и задать красивый шрифт для их описаний:

*chapter09/gallery.css (фрагмент)*

```
#albumlist li {
  float: left;
  width:240px;
  margin-right: 6px;
  margin-bottom: 10px;
  font: bold 0.8em Arial, Helvetica, sans-serif;
  color: #333333;
}
```

Можно задать рамку для изображений:

*chapter09/gallery.css (фрагмент)*

```
#albumlist img {
  border: 1px solid #333300;
}
```

Уверена, вы по достоинству оцените гибкость данного метода, в особенности при добавлении изображений из базы данных – при этом не нужно подсчитывать их количество, и можно динамически добавлять в список новые пункты.

Тем не менее в определенных случаях переход изображений на новую строку может быть нежелателен. Этого можно избежать, задав ширину для элемента `ul`:

```
#albumlist {  
  list-style-type: none;  
  width: 600px;  
}
```

Это правило задает ширину, по достижении которой изображения могут переходить на следующую строку, как показано на рис. 9.50.



*Рис. 9.50. После указания ширины родительского элемента `ul` изображения не переходят на следующую строку*

## Создание макета страницы с помощью CSS-таблиц

В разделе «Создание колонки, занимающей все доступное пространство по высоте» было сказано, что с помощью CSS это невозможно. Возмож-



но, следовало бы сказать, что не существует метода создания таких колонок, поддерживаемого в настоящее время всеми распространенными браузерами, поскольку в этом разделе нам предстоит познакомиться с таблицами CSS, с помощью которых можно без труда добиться такого эффекта.

## Решение

Понятие **CSS-таблиц** подразумевает определенные в спецификации CSS 2.1 значения свойства `display`, относящиеся к элементам таблиц: `table`, `table-row` и `table-cell`. При их использовании HTML-элемент будет вести себя как соответствующий элемент таблицы.

CSS-таблицы поддерживаются версиями браузеров не ниже Firefox 2, Opera 9.5, Safari 3, Chrome 1 и Internet Explorer 8. К сожалению, Internet Explorer 6 или 7 не обрабатывает эти значения, поэтому придется решить для себя, насколько важно для вас их использование.

С помощью CSS-таблиц можно разместить элементы в сетке, а также решить проблему создания фона по всей высоте страницы. Элемент, для которого задано свойство `display` со значением `table`, и его потомки будут отображаться как элементы таблицы, что позволяет нам определять их границы по отношению к соседним элементам.

Вернемся к макету фиксированной ширины с двумя колонками и создадим колонку, занимающую все пространство по высоте, не прибегая к описанным ранее хитроумным уловкам. Для элементов приведенного ниже HTML-кода можно задать отображение в виде элементов таблицы:

*chapter09/2col-fixedwidth-tables.htm (фрагмент)*

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
  <head>
    <title>Recipe for Success | Perfect Pizza</title>
    <link href="2col-fixedwidth-table.css" rel="stylesheet"
          type="text/css" />
    <meta http-equiv="content-type"
          content="text/html; charset=utf-8" />
  </head>
  <body>
    <div id="wrapper">
      <div id="header">
        <h1>Perfect Pizza</h1>
      </div>
      <div id="main">
        <div id="nav">
          <ul>
            <li><a href="#">Prepare the Dough</a></li>
            <li class="cur"><a href="#">Choose Your Toppings</a>
            </li>
          </ul>
        </div>
      </div>
    </div>
  </body>
</html>
```

```

        <li><a href="#">Pizza Ovens</a></li>
        <li><a href="#">Side Salads</a></li>
    </ul>
</div>
<div id="content">
    <h2>Choosing Your Toppings</h2>
    <p>Sed nec erat sed sem molestie congue. Cras lacus ...</p>
    <p>Vestibulum dictum massa at odio. In dignissim ...</p>
    <div id="footer">&copy; 2009 Recipe for success</div>
</div>
</div>
</div>
</body>
</html>

```

Как видите, в разметку был добавлен дополнительный блок `div` с идентификатором `main`, обрамляющий элементы `content` и `nav`. Кроме того, разметка меню навигации расположена в коде перед разметкой области с контентом: единственный недостаток использования **CSS-таблиц** заключается в необходимости размещения содержимого колонки строго в том же порядке, в котором оно должно отображаться. Теперь не нужно задавать свойство `clear: both;` для элемента `footer`, чтобы он отображался ниже всех трех колонок, поэтому его можно разместить внутри основного блока `div`.

В **CSS**-код достаточно внести лишь несколько изменений. Во-первых, нужно удалить фоновое изображение элемента `wrapper`, поскольку теперь изображение можно задать для самой колонки; для элемента `main` нужно задать свойство `display` со значением `table`, а для элементов `content` и `nav` — со значением `table-cell`. Теперь фоновое изображение для боковой панели можно задать прямо элементу `nav`, удалить свойства `margin-left` и `clear` для элемента `footer`, поскольку теперь в них нет необходимости как при создании макета на основе плавающих блоков. Ниже представлен измененный **CSS**-код:

#### *chapter09/2col-fixedwidth-tables.css*

```

#wrapper {
    position: relative;
    text-align: left;
    width: 760px;
    margin-right: auto;
    margin-left: auto;
    margin-bottom: 1em;
    border-right: 1px solid #888888;
    border-bottom: 1px solid #888888;
}
:
#main {
    display: table;
}

```

```
#content {
  display: table-cell;
  width: 500px;
  padding: 0 10px 0 20px;
}
:
#nav {
  display: table-cell;
  width: 230px;
  background-image: url(sidebar.gif);
  background-repeat: repeat-y;
}
:
#footer {
  font-size: 80%;
  padding: 1em 0 1em 0;
  color: #999999;
  background-color: transparent;
}
}
```

Теперь макет страницы выглядит так же, как и созданный нами ранее макет в разделе «Создание колонки, занимающей все доступное пространство по высоте», — без использования плавающих блоков и возникновения соответствующих проблем. Версия с CSS-таблицами гораздо проще и логичнее.

Если помните, в разделе «Добавление к «резиновому» макету нижнего блока» мы столкнулись со значительными трудностями при попытке добавления нижнего блока к «резиновому» макету с тремя колонками, в результате чего пришлось прибегнуть к довольно неестественному способу с использованием отрицательных внешних отступов. Ниже представлена разметка «резинового» макета с тремя колонками, основанного на использовании CSS-таблиц:

*chapter09/3col-table.html (фрагмент)*

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
  <head>
    <title>Recipe for Success</title>
    <meta http-equiv="content-type"
      content="text/html; charset=utf-8" />
    <link rel="stylesheet" type="text/css" href="3col-table.css" />
  </head>
  <body>
    <div id="wrapper">
      <div id="content">
        <div id="side1">
          <form method="post" action="" id="searchform">
            <h3><label for="keys">Search the Recipes</label></h3>
```

```

        <div>
            <input type="text" name="keys" id="keys"
                class="txt" /><br />
            <input type="submit" name="Submit" value="Submit" />
        </div>
    </form>
    <ul>
        <li><a href="#">About Us</a></li>
        <li><a href="#">Recipes</a></li>
        <li><a href="#">Articles</a></li>
        <li><a href="#">Buy Online</a></li>
        <li><a href="#">Contact Us</a></li>
    </ul>
</div>
<div id="main">
    <h1>Recipe for Success</h1>
    <p>Lorem ipsum dolor sit amet, consectetur ...</p>
    <p>Quisque sodales imperdiet enim. Quisque ...</p>
</div>
<div id="side2">
    <h3>Please Visit our Sponsors</h3>
    <div class="adbox"><p>Lorem ipsum dolor sit ...</p></div>
    <div class="adbox"><p>Sed mattis, orci eu ...</p></div>
    <div class="adbox"><p>Quisque mauris nunc, ...</p></div>
</div>
</div>
</div>
<div id="footer">
    : footer content...
</div>
</body>
</html>

```

**Четкость и последовательность** данного кода сразу бросается в глаза при сравнении с версией того же документа, созданной на основе плавающих блоков. CSS-код будет таким:

*chapter09/3col-table.css (фрагмент)*

```

#wrapper {
    width:100%;
}
#content {
    display: table;
    width: 100%;
}
#main {
    display: table-cell;
}
#side1 {
    display: table-cell;
}

```

```
width:200px;
padding: 0 10px 0 10px;
}
#side2 {
display: table-cell;
width: 190px;
padding: 80px 10px 0 10px;
}
#footer {
border-top: 10px solid #CECECE;
}
```

CSS-код также отличается простотой и логичностью: не нужно задавать отступы, обтекание и т. д. При этом внешне макет ничем не отличается от таблицы с одной строкой и тремя колонками. Нижний блок аккуратно располагается под колонками.

## Обсуждение

Вы, возможно, заметили, что ни одному элементу не задано свойство для отображения в качестве ряда таблицы. При создании документа несложной структуры браузер добавит мнимый ряд таблицы, называемый **неопределенным элементом таблицы**, окружающий ячейки. Однако при создании более сложных макетов рекомендуется использовать дополнительные элементы, отображаемые в качестве рядов таблицы. Это поможет избежать лишних ошибок.

Описанный метод так и хочется применить на практике, но помните об отсутствии поддержки необходимых свойств браузерами Internet Explorer 6 и 7. Если вы хотите, чтобы сайт корректно отображался в любых браузерах, можно добавить альтернативный код для IE6 и 7 с помощью условных комментариев. Думаю, в будущем все больше сайтов будет создаваться с использованием CSS-таблиц – как только число пользователей Internet Explorer 6 и 7 начнет стремительно уменьшаться. Более подробную информацию об использовании данного метода и адаптации кода для старых браузеров вы найдете в книге, написанной мной совместно с Кевином Янком (Kevin Yank) «Everything You Know About CSS Is Wrong», опубликованной издательством SitePoint.<sup>1</sup>

## Заключение

В данной главе были рассмотрены основные принципы построения макетов страниц. Комбинируя их с другими рассмотренными в данной книге решениями, например, различными методами создания навигации и оригинальными способами использования изображений, вы сможете воплотить свои творческие идеи на практике, создавая собствен-

---

<sup>1</sup> <http://www.sitepoint.com/books/csswrong1/>

ные страницы с уникальным дизайном. Большая часть созданных с помощью каскадных таблиц стилей макетов, по сути, основана на одних и тех же приемах, что можно сказать и о макетах, сверстанных таблицами.

Как только вы усвоите основные принципы и правила использования CSS, то увидите, что простор для фантазии при разработке дизайна сайта поистине безграничен. Убедиться в потрясающих возможностях CSS можно, посмотрев макеты, создаваемые разными разработчиками на сайте CSS Zen Garden.<sup>1</sup>

---

<sup>1</sup> <http://www.csszengarden.com/>

# Алфавитный указатель

## А

<a>, элемент, 30  
accesskey, атрибут, 185  
Adobe BrowserLab, сервис, 204  
all, среда отображения, 236  
alt, атрибут, 232

## В

background-attachment, свойство, 79  
background-color, свойство, 168  
background-image, свойство, 72  
background-position, свойство, 76  
background-repeat, свойство, 74  
background, свойство, 79  
<blockquote>, элемент, 59  
Boot Camp, программа, 201  
border-collapse, свойство, 139  
border-radius, свойство, 279  
border, свойство, 72, 105, 168  
braille, среда отображения, 236  
BrowserCam, сервис, 204  
Browser Shots, сервис, 204

## С

<caption>, элемент, 135  
clear, свойство, 267  
<col>, элемент, 150  
<colgroup>, элемент, 152  
color, свойство, 168  
CrossBrowserTesting.com, сервис, 204  
CSS-таблицы, 324  
cursor, свойство, 116

## D

display, свойство, 66, 96, 104, 171, 260, 324  
<!DOCTYPE>, тег, 211

## Е

em, единица измерения, 39  
    использование для позиционирования, 291

embossed, среда отображения, 236

## F

<fieldset>, элемент, 180  
Fire Vox, расширение для Firefox, 235  
first-child, псевдокласс, 31, 50  
float, свойство, 110, 264  
:focus, псевдокласс, 194  
font, свойство, 168  
font-family, свойство, 35  
font-size, свойство, 37  
for, атрибут, 175

## G

Gecko, движок, 198  
GIF, формат, 87

## Н

handheld, среда отображения, 236  
hasLayout, свойство, 217  
:hover, псевдокласс, 97  
HTML, 22  
HTML 4.01 Frameset, 212  
HTML 4.01 Strict, 212  
HTML 4.01 Transitional, 212

## I

id, атрибут, 27  
important, ключевое слово, 32  
@import, команда, 208  
<input>, элемент, 168  
Internet Explorer, браузер, 205  
    автономные версии, 206

## J

JAWS, экранный диктор, 234  
jQuery, библиотека, 147

## К

KHTML, движок, 198  
Knopix, дистрибутив, 200  
Konqueror, браузер, 203

**L**

<label>, элемент, 173  
<legend>, элемент, 180  
line-height, свойство, 232  
list-style-image, свойство, 64  
list-style-type, свойство, 62  
Litmus, сервис, 204  
localhost, 234  
Лунх, браузер, 232

**M**

margin, свойство, 55, 261  
media, атрибут, 236  
@media, правило, 237  
Microsoft SuperPreview, сервис, 204  
min-height, свойство, 217  
mouseout, обработчик событий, 148  
mouseover, обработчик событий, 148

**N**

NiftyCube, сценарий, 283  
nth child, псевдокласс, 146

**P**

padding, свойство, 55, 168, 261  
Parallels Desktop, приложение, 202  
Parallels Workstation, приложение, 201  
PNG, формат, 87  
    прозрачность, 219  
position, свойство, 218, 275

**R**

rel, атрибут, 24

**S**

<select>, элемент  
    фоновый цвет, 186  
Spanky Corners, метод, 285  
style, атрибут, 22  
<style>, элемент, 21  
summary, атрибут, 135

**T**

text-align, свойство, 57, 60  
text-decoration, свойство, 44, 52  
text-height, свойство, 56  
text-indent, свойство, 60  
text-transform, свойство, 61  
<th>, элемент, 136

**U**

Ubuntu, дистрибутив, 200

**V**

VirtualBox, приложение, 201, 202  
VMWare Fusion, приложение, 202  
VMWare Workstation, приложение, 201, 202

**W**

W3C CSS Validator, валидатор, 227  
W3C Validator, валидатор, 227  
WebKit, движок, 198  
width, свойство, 168

**X**

XHTML, 22  
XHTML 1.0 Frameset, 212  
XHTML 1.0 Strict, 212  
XHTML 1.0 Transitional, 212  
XHTML 1.1, 212  
XML, 212  
x-ua-compatible, метка, 223

**A**

альтернативные таблицы стилей, 246  
альтернативные устройства, 230

**Б**

блок описаний, 24  
блочный элемент, 96  
браузер, 198  
браузерный движок, 198

**В**

валидация, 215  
вес стиля, 32  
виртуальная машина, 201  
виртуальная среда  
    создание, 202  
вкладки, 107  
    выделение текущей, 112  
внешние таблицы стилей, 23  
    ссылка на, 23  
внутритекстовые стили, 22  
выделение текста, 55  
выпадающее меню, 125

**Г**

глобальное удаление промежутков, 68  
горизонтальная линия, 58  
горизонтальное меню, 102



**Д**

динамические селекторы псевдоклассов, 31  
доступность, 230

**Е**

единицы измерения, 37, 39, 41, 244

**З**

заголовок, фон, 51  
закругленные края блока, 279

**И**

идентификатор, 257  
изображение, рамка, 70  
    двойная, 89  
имена классов, 188  
интервал, межстрочный, 56  
источник стиля, 32

**К**

календарь  
    создание с помощью CSS, 153  
карта сайта, 122  
каскадирование, 27, 32  
клавиши быстрого доступа, 184  
классы, 26, 257  
ключевые слова, 41, 77  
    абсолютные, 41  
    относительные, 42  
кнопки, 105  
    подтверждения, 172  
комментарии, добавление, 69  
конкретность, 34  
контрастность, 231  
критерии доступности веб-контента, 231  
курсор, изменение вида, 116

**М**

макет  
    масштабируемый, 249  
    с тремя колонками, 310  
    фиксированной ширины, 294  
межстрочный интервал, 56

**Н**

навигационное меню  
    на основе изображений, 126  
    с кнопками, 105  
    смена изображений, 118  
    создание с помощью списка, 93  
навигация, 92

наследование, 43  
ненавязчивый JavaScript, 149  
неопределенный элемент таблицы, 328

**О**

обратная совместимость, 206  
обтекание, 265  
описание стиля, 20, 24  
относительное позиционирование, 218  
отступы  
    отрицательные, 55  
    удаление, 66  
оформление по умолчанию, 21

**П**

переключение режима отображения, 211  
переносы строки, 171  
пики, 38  
пиксели, 38  
плавающие блоки, 265  
позиционирование, 273, 277  
    абсолютное, 273  
    относительное, 301  
порядок следования, 34  
последний элемент в цепочке, 54  
правило стиля, 24  
    способы записи, 24  
предупреждения валидатора, 228  
прозрачность, эффект, 87  
проценты, 41  
псевдокласс, 30  
пункты, 38  
пустое пространство между ячейками  
    удаление, 139

**Р**

раздвижных дверей метод, 109  
размер элемента  
    относительное задание, 43  
рамка, 137  
режим совместимости, 42, 211, 264  
«резиновый» макет, 78, 286

**С**

свойство, 24  
селектор, 20, 24  
    ID, 27  
    атрибутов, 115, 170, 188  
    дочерний, 28  
    класса, 26  
    контекстный, 28  
    псевдоклассов для ссылок, 30

- смежный, 29
- типа, 25
- синтаксис CSS, 24
- списки
  - вложенные, 99
  - маркированные
    - стиль маркеров, 62
  - отступы, 64
  - размещение пунктов по горизонтали, 66
- среда отображения, 236
- ссылки
  - изменение вида при наведении указателя мыши, 96, 97
  - изменение цвета, 46
  - подчеркивание, 44
- стандартный режим отображения, 211
- стили встраиваемые, 22

## Т

- таблица стилей для печати, 38, 238
- таблицы
  - рамка, 137
  - фоновый цвет колонок, 150
- табличные данные, 132, 133
- текст
  - выравнивание, 57
  - отображение заглавными буквами, 61
  - отступы, 59
  - размещение поверх изображения, 84
  - центрирование, 60
- текстовый броузер, 232
- тень, добавление, 307
- тестирование сайта, 199

## У

- универсальный селектор, 68
- условные комментарии, 214

## Ф

- фоновое изображение, 72
  - способ размножения, 74
  - фиксация, 79
- фоновый градиент, использование, 307
- формы, 164
  - ввод данных, 188
  - в две колонки, 176
  - вид элементов, 165
  - группировка полей, 180
  - нежелательное пустое пространство, 171
  - фоновый цвет, 168

- фотоальбом, создание, 318

## Ц

- центрирование блока, 277

## Ш

- шрифт, 35
  - размер, 37, 39, 41, 244
  - семейства шрифтов, 36

## Э

- экранный диктор, 235
- элементы
  - блочные, 258
  - строковые, 258
- эффект гало, 87

## Я

- явные ярлыки, 175

По договору между издательством «Символ-Плюс» и Интернет-магазином «Books.Ru – Книги России» единственный легальный способ получения данного файла с книгой ISBN 978-5-93286-182-0, название «CSS: 100 и 1 совет, 3-е издание» – покупка в Интернет-магазине «Books.Ru – Книги России». Если Вы получили данный файл каким-либо другим образом, Вы нарушили международное законодательство и законодательство Российской Федерации об охране авторского права. Вам необходимо удалить данный файл, а также сообщить издательству «Символ-Плюс» ([piracy@symbol.ru](mailto:piracy@symbol.ru)), где именно Вы получили данный файл.