Эрик Фримен, Элизабет Фримен

Изучаем HTML XHTML и CSS



Начните свою веб-карьеру, прочитав всего одну главу

Руководство по созданию веб-страниц, основанных на стандартах



Узнайте все хитрости работы с HTML и CSS



Узнайте, каким на самом деле должен быть стиль

Решите около 100 упражнений и головоломок





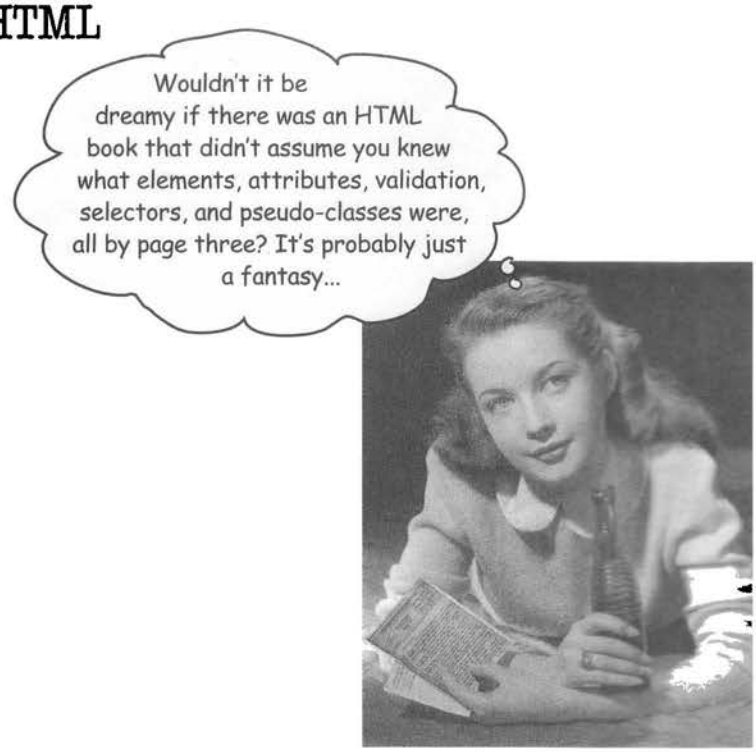
Научитесь избегать ошибок при проверке кода

O'REILLY®

ПИТЕР®



Head First HTML
with CSS & XHTML



Elisabeth Freeman
Eric Freeman

O'REILLY®

Beijing • Cambridge • Köln • Sebastopol • Taipei • Tokyo

Изучаем HTML, CSS и XHTML



Элизабет Фримен
Эрик Фримен

ПИТЕР®

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

2012

Эрик Фримен, Элизабет Фримен

Изучаем HTML, XHTML и CSS

Серия «Бестселлеры O'Reilly»

Перевели с английского И. Дубенок, В. Квиткевич

Заведующий редакцией	*А. Громаковский*
Ведущий редактор	*Н. Гринчик*
Художник	*А. Татарко*
Корректор	*Е. Павлович*
Верстка	*Д. Коршук*

ББК 32.988.02

УДК 004.738.5

Эрик Фримен, Элизабет Фримен

Ф88 Изучаем HTML, XHTML и CSS. — СПб.: Питер, 2012. — 656 с.: ил. — (Серия «Бестселлеры O'Reilly»).

ISBN 978-5-459-01060-2

Устали от чтения таких книг по HTML, которые понятны только специалистам в этой области? Тогда самое время взять в руки наше издание. Хотите изучить HTML так, чтобы уметь создавать веб-страницы, о которых вы всегда мечтали? Так, чтобы более эффективно общаться с друзьями, семьей и привередливыми клиентами? Хотите действительно обслуживать и улучшать HTML-страницы по прошествии времени, чтобы они работали во всех браузерах и мобильных устройствах? Тогда эта книга для вас. Прочитав ее, вы узнаете все секреты создания веб-страниц.

Благодаря ей вам больше не придется думать, какие цвета нужно использовать, чтобы они сочетались между собой, как правильно применять шрифты, чтобы они не «плавали» по экрану и верно отображались в различных браузерах. Вы узнаете, как работают профессионалы, чтобы получить визуально привлекательный дизайн, и как максимально эффективно использовать HTML, CSS и XHTML, чтобы создавать такие веб-страницы, мимо которых не пройдет ни один пользователь.

Права на издание получены по соглашению с O'Reilly.
Все права защищены. Никакая часть данной книги не может быть воспроизведена в какой бы то ни было форме без письменного разрешения владельцев авторских прав.

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

ISBN 978-0-59-610197-8

ISBN 978-5-459-01060-2

© 2006 O'Reilly Media, Inc. All rights reserved

© Перевод на русский язык ООО Издательство «Питер», 2012

© Издание на русском языке, оформление ООО Издательство «Питер», 2012

ООО «Мир книг», 198206, Санкт-Петербург, Петергофское шоссе, 73, лит. А29

Налоговая льгота — общероссийский классификатор продукции ОК 005-93, том 2: 95 3005 — литература учебная.

Подписано в печать 16.12.11. Формат 84x108/16. Усл. п. л. 68,880. Доп. тираж. Заказ 27020.

Отпечатано по технологии CtP в ОАО «Первая Образцовая типография», обособленное подразделение «Печатный двор». 197110, Санкт-Петербург, Чкаловский пр., 15.

Войны между браузерами. Читайте об этом в главе 6.

Посвящается Консорциуму Всемирной паутины (W3C) за прекращение войн между браузерами и четкое разграничение структуры (HTML) и презентации (CSS)...

Для того чтобы грамотно использовать HTML, CSS и XHTML в комплексе, необходимо прочитать эту книгу.

Об авторах

Элизабет Фримен (Elisabeth Freeman)



Элизабет – автор и разработчик программного обеспечения. Она с самого детства имеет дело с Интернетом. Элизабет – соучредитель The Ada Project (TAP), сайта для женщин, который посвящен работе с компьютером и ныне принят ACM (Ассоциацией по вычислительной технике). Кроме того, Элизабет возглавляет работы по исследованию и развитию цифровых носителей информации в Walt Disney Company, где она стала одним из изобретателей Motion, системы управления содержимым, которая каждый день доставляет терабайты видео пользователям Disney, ESPN и Movies.com.

В глубине души Элизабет – исследователь в области компьютеров. Она имеет научные степени по компьютерным наукам Йельского и Индианского университетов. Помимо всего прочего, Элизабет работала над языком визуального общения, RSS (формат представления новостей), с интернет-системами. Она также активно выступает в защиту женщин, участвующих в компьютеризации, и разрабатывает специальные программы.

Сегодня вы найдете ее за своим ноутбуком, пишущей на Java и Cocoa, хотя она мечтает о временах, когда весь мир будет использовать язык Scheme.

Со времен своего детства в Шотландии Элизабет любит туризм и отдых на открытом воздухе. Она также заядлая велосипедистка, вегетарианка и любитель животных.

Вы можете отправить ей письмо на beth@oreilly.com.



Эрик Фримен (Eric Freeman)

Эрик – исследователь в области компьютеров, страстно увлекающийся работой с информацией медиатипа и архитектурой систем программного обеспечения. В течение последних четырех лет он занимался реализацией своей мечты, управляя разработкой широкополосного Интернета и беспроводного доступа в Disney. Теперь же он снова вернулся к созданию первоклассного программного обеспечения и усердно трудится на Java в Mac.

Эрик провел большую часть 1990-х годов, работая над альтернативными моделями «Рабочего стола» вместе с Дэвидом Джелернтером (David Gelernter) (и они оба до сих пор задаются вопросом: «Почему нужно давать файлам имена?»). Благодаря этой работе Эрик получил ученую степень доктора философии в Йельском университете в 1997 году. Он также стал соучредителем корпорации Mirror Worlds Technologies, чтобы создать коммерческую версию своей диссертации, Lifestreams.

В прошлой жизни Эрик создавал программное обеспечение для сетей и суперкомпьютеров. Вы можете знать о нем по такой книге, как *JavaSpaces Principles Patterns and Practice*. Он изобрел запоминающие устройства для систем представления данных в форме кортежей для вычислительных машин CM-5 и создал некоторые из первых информационных систем для НАСА в конце 1980-х годов.

В настоящее время Эрик живет на острове Бэйнбридж. В свободное время (когда он не занят программированием) больше шансов застать его не за просмотром кинофильмов, а за тонкой настройкой программ (завершающими операциями по усовершенствованию аппаратного или программного средства) или за попыткой восстановить видеоигру Dragon's Lair («Логово Дракона»), продолжающейся примерно с начала 1980-х годов. Он также не против поработать ночью в качестве диджея электронной музыки.

Пишите ему на eric@oreilly.com или посетите его блог по адресу http://www.ericfreeman.com.

Содержание (сводка)

	Введение	23
1	Язык Сети: *знакомство с HTML*	35
2	Знакомство с гипертекстом: *идем дальше – используем гипертекст*	75
3	Конструирование веб-страниц: *строительные блоки*	107
4	Путешествие в Webville: *соединение*	151
5	Знакомство с медиа: *добавление изображений на страницы*	189
6	Серьезный HTML: *соответствие стандартам*	245
7	Добавление X к HTML: *переходим к XHTML*	285
8	Начнем работать над дизайном: *приступаем к работе с CSS*	303
9	Увеличиваем словарный запас: *меняем шрифты и цвета*	357
10	Познакомимся с элементами поближе: *блочная модель*	399
11	Современная веб-конструкция: *элементы div и span*	441
12	Расставим элементы по местам: *разметка и позиционирование*	499
13	Представление в табличной форме: *таблицы и большие списки*	557
14	Переход на интерактивный режим: *XHTML-формы*	597
Приложение. Топ-10 тем, которые не были освещены в этой книге		643

Содержание (настоящее)

Введение

Поведение вашего мозга при изучении HTML и CSS. Когда вы пытаетесь что-либо выучить, ваш мозг неустанно следит за тем, чтобы процесс изучения не остановился. Он думает: «Лучше оставить место для более важной информации, чтобы, например, знать, встречи с какими дикими животными следует избегать. Или знать, что катание на сноуборде без специального снаряжения — не самая удачная идея». Как же убедить свой мозг в том, что для вас так же важно знать HTML и CSS?

Для кого написана эта книга?	24
Кому не стоит читать эту книгу?	24
Мы знаем, что вы думаете	25
Мы также знаем, о чем думает ваш мозг	25
Метапознание: учимся учиться	27
Вот что МЫ делали	28
А вот что можете сделать ВЫ, чтобы заставить свой мозг работать	29
Примите к сведению	30
Технические рецензенты	32
Благодарности	33

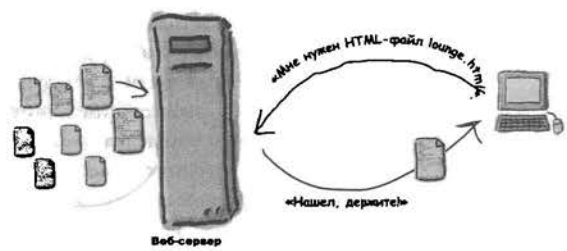
1 Знакомство с HTML

Язык Сети

Единственное, что необходимо для того, чтобы успешно работать в Сети, — научиться говорить на ее специфическом языке: HyperText Markup Language (язык гипертекстовой разметки), или сокращенно HTML. Итак, приготовьтесь к нескольким урокам языка. После этой главы вы не только узнаете некоторые базовые понятия HTML, но и сможете разговаривать на этом языке, используя определенный **стиль**. Черт возьми, к концу этой книги вы сможете говорить на языке HTML так, будто выросли в Сети!

Сеть убила радиозвезду	36
Что делает веб-сервер	37
Что делает браузер	37
Что пишете вы (HTML-код)	38
Что создает браузер	39
Большая перемена в кафе Starbuzz	43
Создание веб-страницы для Starbuzz	45
Создание HTML-файла (Mac)	46
Создание HTML-файла (Windows)	48
Между тем вернемся к кафе Starbuzz	51
Сохранение работы	52
Открытие веб-страницы в браузере	53
Тестирование страницы	54
Еще один тест	58
Разделение тегов	59
Познакомьтесь с элементом <style>	63
Придание определенного стиля странице Starbuzz	64





2 Знакомство с гипертекстом

Кто-то сказал «гипертекст»? Что это? О, только чистая основа Сети. В главе 1 мы привели основные сведения о языке HTML, и, надеемся, вы пришли к выводу, что это хороший язык для разметки текста, используемый для описания структуры веб-страниц. Сейчас наша задача — разобраться с гипертекстом, который позволит освободиться от одиночных страниц и ссылаться на другие страницы. В процессе этого мы познакомимся с новым элементом <a> и поймем, какая превосходная штука — взаимосвязь страниц. Итак, пристегните ремни безопасности, вы вот-вот начнете изучение гипертекста.



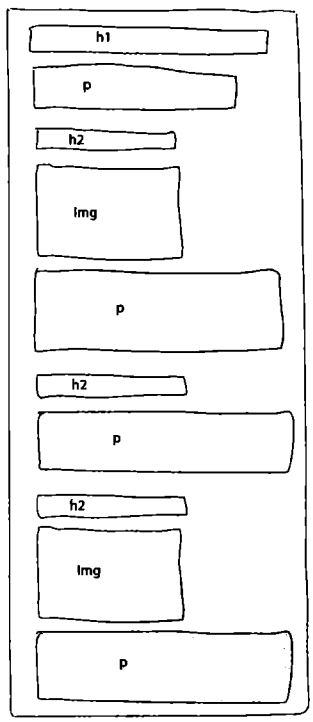
Новая и усовершенствованная гостевая	76
Создание новой гостевой	78
Что делали мы	80
Что делает браузер	81
Что такое атрибуты	83
Технические трудности	90
Планирование путей	92
Восстановление «отсутствующих изображений»	98



3 строительные блоки

Конструирование веб-страниц

Мы говорили вам, что в этой книге вы действительно будете создавать веб-страницы. Конечно, уже многое выучено: теги, элементы, ссылки, пути, однако все это бесполезно, если, используя полученные знания, не попробовать создать парочку потрясающих веб-страниц. В этой главе мы будем расширять строительство веб-страниц: перейдем от их общего осмысления к проектированию, зальем фундамент, построим их и даже выполним кое-какую отделку. Все, что вам нужно, — это усердие, трудолюбие и пояс для рабочих инструментов, так как мы будем добавлять некоторые новые инструменты и давать вам информацию, как пользоваться ими.



От дневника к сайту на скорости 12 миль в час	109
Черновик	110
От черновика к плану	111
От плана к веб-странице	112
Тестирование страницы Тони	114
Добавление новых элементов	115
Знакомство с элементом <q>	116
...и его тестирование	116
Дли-и-и-инные цитаты	120
Добавление элемента <blockquote>	121
Полное разоблачение тайны <q> и <blockquote>	124
Тем временем вернемся к сайту Тони...	132
Разработка HTML-списков в два этапа	134
Тестирование списков на примере перечня городов	136
Используйте вложенность, чтобы убедиться в соответствии тегов	141

соединение

4 Путешествие в Webville

Веб-страницы предназначены для того, чтобы располагаться и обслуживаться в Интернете. До сих пор вы создавали веб-страницы, которые «жили» только в вашем собственном компьютере. Вы также создавали ссылки только на те страницы, которые хранятся на вашем компьютере. Мы вот-вот изменим это навсегда. В этой главе мы научим вас размещать веб-страницы в Интернете, где все ваши родные, друзья и покупатели действительно смогут их увидеть. Мы также раскроем тайну создания ссылок на другие страницы, взломав код h, t, t, p, :, /, /, w, w, w. Итак, собирайте свои вещи, следующая остановка — Webville.

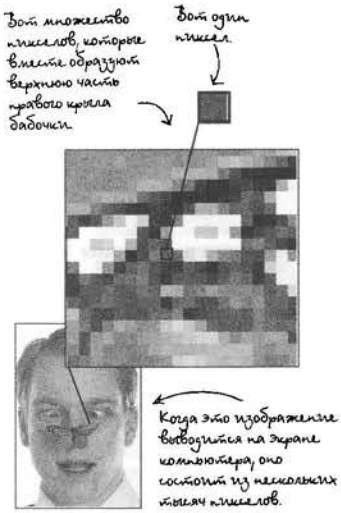
Размещение сайта Starbuzz (или вашего собственного сайта) в Сети 152
Поиск хостинговой компании 153
Привет, мое доменное имя... 154
Как можно получить доменное имя 154
Заселение 156
Перемещение файлов в корневую папку 157
Столько информации об FTP, сколько может поместиться на две страницы 158
Вернемся к делу... 161
Что такое HTTP-протокол 163
Что такое абсолютный путь 164
Как работают страницы, выдаваемые по умолчанию 167
Как мы создаем ссылки на другие сайты 170
Создание ссылки на страницу о кофеине 171
А теперь протестируем... 172
Наивысший уровень качества веб-страниц 175
Тестирование атрибута title 176
Создание ссылки внутрь страницы 177
Использование элемента <a> для указания пункта назначения 178
Как сослаться на якорь 179
Переход по ссылке в новое окно 183
Открытие нового окна с использованием атрибута target 184



Добавление изображений на страницы

5 Знакомство с медиа

Улыбнитесь и скажите «сыр». Теперь улыбнитесь и скажите «gif», «jpg» или «png» — это те форматы файлов, которые вы выберете, создавая рисунки для Сети. В этой главе вы узнаете все о том, как добавить на веб-страницу свой первый медиафайл — изображение. У вас есть парочка цифровых фотографий, которые вы хотите поместить в Сеть? Никаких проблем. У вас есть логотип, который нужен на веб-странице? И это легко. Или, может быть, сначала вы хотите более близко познакомиться с элементом <img>? К концу этой главы вы будете знать все мельчайшие подробности того, как использовать этот элемент и его атрибуты. Вы также узнаете, как этот небольшой элемент побуждает браузер делать такую серьезную работу по поиску и отображению ваших картинок.



Как браузер работает с изображениями	190
Как работают рисунки	193
<img>: теперь не только относительные ссылки	198
Всегда имейте запасной вариант	200
Определение размеров изображений	201
Создание сайта для самых больших фанатов: myPod	202
Доработка файла index.html для сайта myPod	203
Ого! Рисунок слишком большой	205
Изменение размера изображения	207
Открытие изображения	209
Размеры изменены, теперь сохраняем	213
Исправление HTML-кода для myPod	215
А сейчас протестируем...	215
Еще больше фотографий для myPod	217
Еще один тест для myPod	218
Доработка сайта таким образом, чтобы использовались эскизы	219
Создание эскизов	220
И снова тест для myPod	222
Превращение эскизов в ссылки	223
Создание индивидуальных страниц для фотографий	224
Итак, как же создать изображения-ссылки?	225
Открытие логотипа myPod	229
Какой формат использовать?	230
Использовать прозрачность или нет?	231
Сохранение прозрачного GIF-изображения	232
Минуточку, а как узнать цвет фона веб-страницы?	233
Установка цвета подложки	233
Рассмотрим логотип с подложкой	234
Сохранение логотипа	235
Добавление логотипа на веб-страницу myPod	235
А теперь последний тест	236

соответствие стандартам

6 Серьезный HTML

Что же еще нужно знать об HTML? Вы уже довольно неплохо справляетесь с написанием HTML-страниц. Не настало ли время перейти к CSS и научиться придавать всей этой разметке еще и ошеломительный внешний вид? Перед тем как сделать это, мы должны убедиться, что ваши знания о HTML действительно на должном уровне. Мы планируем это сделать, серьезно подойдя к рассмотрению HTML. Не поймите нас неправильно, вы и так создавали первоклассный HTML, но есть еще несколько моментов, о которых вам нужно знать, чтобы помочь браузеру корректно отображать ваши страницы. Кроме того, вы должны быть уверены, что в вашем коде не будут появляться мелкие ошибки. Что это вам даст? Страницы, которые отображаются в различных браузерах более или менее одинаково (и даже на мобильных устройствах и с помощью экранных дикторов для людей со слабым зрением), страницы, которые быстрее загружаются, и страницы, которые гарантированно хорошо обращаются с CSS. Приготовьтесь, в этой главе вы из любителя превратитесь в профессионала.



История развития HTML	248
Нужно позаботиться о том, чтобы браузеры не использовали режим обратной совместимости для наших страниц!	251
Добавление определения типа документа	253
Тест для DOCTYPE	254
Познакомьтесь с W3C-валидатором	256
Валидация гостевой Head First	257
Хьюстон, у нас проблема...	258
Исправление этой ошибки	259
Добавление тега <meta> для определения типа кодировки документа	262
Изменение переходного DOCTYPE на строгий	268
Процедура валидации прошла успешно?	269
Исправление ошибки вложенности	271
Еще один шанс стать строгим	272
Строгий HTML 4.01; вам срочно нужен путеводитель	274

7 переходим к XHTML

Добавление X к HTML

Мы вас обманули. Мы знаем, что вы думали, будто покупаете книгу по HTML, но на самом деле это лишь маскировка для книги по XHTML. В действительности все это время мы в основном учили вас XHTML. Вам, наверное, интересно, что это такое? Ладно, познакомьтесь с eXtensible (расширяемым) HTML, также известным как XHTML, следующим этапом в развитии HTML. Он более «скромный» и еще сильнее нацелен на совместимость с браузерами на широком спектре устройств. В этой небольшой главе мы планируем перейти от HTML к XHTML в три простых этапа. Итак, переверните страницу и приступайте... (а после этого мы примемся за CSS).

Что такое XML?	287
Что произойдет с HTML?	288
Итак, в чем же преимущества использования XHTML?	290
Вы совсем близки к тому, чтобы начать использовать XHTML	292
Переход от строгого HTML к XHTML 1.0 за три шага	294
Валидация используется не только для HTML	297
HTML или XHTML? Выбор за вами	302



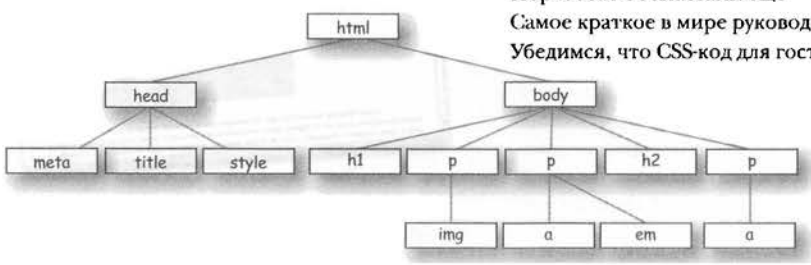
8 Начнем работать над дизайном

Раньше говорилось, что в книге будет материал про CSS. До сих пор мы изучали XHTML, применяемый для создания структуры веб-страниц. Но, как видите, манера браузеров оформлять страницы оставляет желать лучшего. Конечно же, можно было позвать полицию моды, но нам это не нужно. Мы отлично справимся с дизайном страниц с помощью CSS, часто даже не меняя XHTML-код. Действительно ли это так легко? Ну, придется выучить новый язык; в конце концов, Webville — это двуязычный город. После прочтения этой главы, посвященной CSS, вы будете в состоянии поддерживать разговор, находясь на любой из сторон Мейнстрит.

Вы больше не в Канзасе 304
Послушаем, что происходит в реалити-шоу «Квартира соседа» в Webville 306
Использование CSS вместе с XHTML 307
Добавление CSS в ваш XHTML 309
Добавление стиля в гостевую 310
Тестирование стиля 311
Стилизация заголовков 312
Подчеркнем заголовок с приветствием 313
Существует особая технология: указание второго правила, только для <h1> 314
Как же на самом деле все работает 315
Визуальное представление селекторов 318
Присвоение страницам с напитками и указателями стиля основной страницы гостевой 321
Создание файла lounge.css 322
Создание ссылки из lounge.html на внешний CSS-файл 323
Создание ссылок на внешние таблицы стилей из файлов elixir.html и directions.html 324
Тестирование всего сайта 325
Пришло время поговорить о наследовании 329
Что будет, если мы переместим font вверх по дереву? 330
Протестируем новый CSS-код 331
Переопределение наследуемых свойств 332
Тест 333
Создание класса в файле elixir.html 335
Создание селектора для класса 336
Тестирование класса greentea 337
Поработаем с классами еще 338
Самое краткое в мире руководство по применению классов 340
Убедимся, что CSS-код для гостевой валидный 347

Пятиминутная головоломка



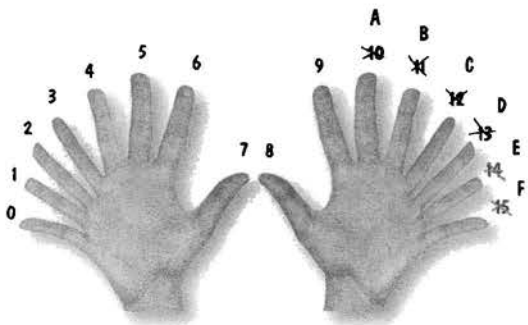


9 Меняем шрифты и цвета

Увеличиваем словарный запас

Ваше изучение языка CSS проходит успешно. Вы уже ознакомились с основами CSS и знаете, как создавать правила, выбирать элементы и определять для них стили. Теперь настало время увеличить ваш словарный запас, а это означает, что вам нужно познакомиться с некоторыми новыми свойствами и узнать, что они могут делать. В настоящей главе мы поработаем с несколькими наиболее используемыми свойствами, которые влияют на оформление текста. Для этого вам придется кое-что узнать о цветах и шрифтах. Вы поймете, что совершенно не обязательно устанавливать те шрифты, которые применяются повсеместно, или те размеры и стили, что по умолчанию используются браузерами для абзацев и заголовков. Вы также узнаете, что существует намного больше цветов, чем может различить ваш глаз.

Самое главное о тексте и шрифтах 358
Итак, что такое семейство шрифтов? 360
Определение семейств шрифтов в CSS 363
Как работает свойство font-family 363
Вновь поработаем с дневником Тони 364
Задаем новое свойство font-family 365
Тестирование новых шрифтов страницы Тони 366
Как быть, если у разных пользователей установлены различные шрифты? 367
Размеры шрифта 368
Поменяем размеры шрифтов для веб-страницы Тони 372
Тестирование страницы с новыми размерами шрифтов 373
Настройка насыщенности шрифтов 375
Заголовки с плотностью normal. Тестирование страницы 376
Стилизация шрифтов 377
Стилизация цитаты курсивом на странице Тони 378
Как работают «безопасные» цвета? 380
Как задаются «безопасные» цвета? Рассмотрим разные способы... 383
Двухминутное руководство по использованию шестнадцатеричных кодов 386
Объединим все вместе 388
Где найти «безопасные» цвета 388
Вернемся к странице Тони... Сделаем заголовки оранжевыми и подчеркнем их 391
Тестирование оранжевых заголовков на странице Тони 392
Все, что вы хотели знать о декорировании текста 393
Удаление подчеркивания 394



10 Познакомимся с элементами поближе

Чтобы создавать современные веб-сооружения, вам нужно на самом деле хорошо разбираться в строительных материалах. В этой главе мы подробно рассмотрим наши строительные материалы — элементы XHTML. Мы буквально под микроскопом изучим, из чего сделаны все эти блочные и строчные элементы. Вы узнаете, как можно управлять практически всеми возможностями оформления элементов в CSS. Однако на этом мы не остановимся — вы также узнаете, как можно присваивать элементам уникальные идентификаторы. И, если этого будет недостаточно, вы выучите, в каком случае и как использовать несколько таблиц стилей. Итак, переворачивайте страницу и познакомьтесь с элементами поближе.



Модернизация гостевой	400
Подготовка к работе с новой гостевой	402
Начнем с нескольких простых изменений	402
Поработаем с межстрочными интервалами	404
Подготовка к главной реконструкции	405
Рассмотрим блочную модель более подробно	406
Что можно делать с блоками	408
Тем временем вернемся к гостевой	411
Создание стиля оформления для «абзаца с гарантией»	413
Тест для границы абзаца	414
Отступ, граница и поля «абзаца с гарантией»	415
Добавление отступов	415
Тест для отступов	416
Теперь добавим поля	416
Тест для полей	417
Добавление фонового рисунка	418
Тест для фонового рисунка	420
Закрепление фонового изображения	421
Еще один тест для фонового изображения	422
Как увеличить отступ только с левой стороны?	422
Как увеличить размер поля только с правой стороны?	423
Двухминутное руководство по границам	424
Доведение границы до совершенства	426
Атрибут id	430
Как же селектор идентификатора применяется в CSS	431
Использование идентификатора для гостевой	432
Смешивание нескольких таблиц стилей	434
Использование нескольких таблиц стилей	435
Таблицы стилей — теперь не только для представления в окнах браузеров	436

11 элементы div и span

Современная веб-конструкция

Пришло время для подготовки массивной конструкции. В этой главе мы займемся такими XHTML-элементами, как и . Это вам уже не мелкие прутья, а большие стальные балки. С помощью и вы построите серьезные опорные конструкции и, расставив их по местам, сможете стилизовать новыми, более действенными методами. Обратите внимание, что ваш пояс для CSS-инструментов практически заполнен, так что настало время показать вам несколько приемов для быстрого доступа к ним, что очень облегчит определение всех этих свойств. В эту главу мы также пригласили специальных гостей — псевдоклассы, с помощью которых вы сможете создавать очень интересные селекторы.

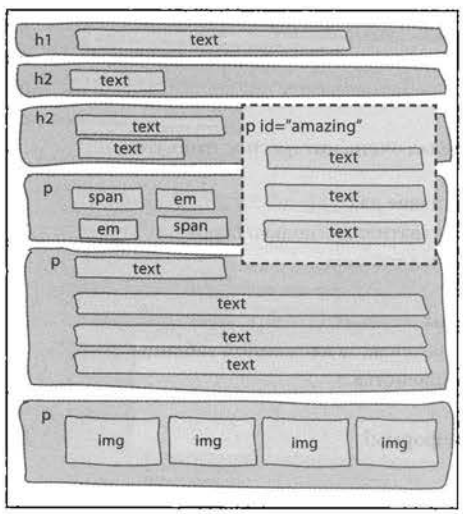


Рассмотрим XHTML с описанием напитков	443
Выясним, как можно разбить страницу на логические разделы	445
Тем временем вернемся в гостевую	450
Тест для	451
Добавление границы	452
Тест для границы	452
Что осталось добавить в стиль раздела с напитками	453
План действий	454
Поработаем над шириной раздела с напитками	454
Протестируем ширину раздела с напитками	455
Стилизация раздела с напитками	459
Тест для новых стилей	460
Мы почти закончили...	463
Нам нужен способ выбрать потомков	465
Изменение цвета для заголовков раздела с напитками	467
Быстрый тест	467
Решение проблемы с межстрочными интервалами	468
Посмотрите, что получилось	469
Пришло время воспользоваться сокращениями	470
Добавление элементов за три простых шага	476
Тестирование элементов	477
Элемент <a> и его разносторонняя личность	480
Как можно по-разному стилизовать элементы с учетом их состояния?	481
Применение псевдоклассов на практике	483
Тест для ссылок	484
Не пора ли поговорить о каскадности?	485
Поиграем в игру «Каков мой приоритет?»	488



12 Расставим элементы по местам

Пришло время обучить XHTML-элементы новым трюкам. Пора разбудить их и заставить помочь нам создавать страницы с реальными схемами размещения элементов. Каким образом? Ну, вы хорошо разобрались со структурными элементами и и теперь знаете, как работает блочная модель. Пришла пора применить эти знания и создать несколько дизайнов. Мы говорим не просто о цвете фона и шрифта, а о полностью профессиональном дизайне, в котором используется многоколоночная разметка.



Вы сделали упражнение «Мозговой штурм» повышенной сложности? 500
Используй поток, Люк 501
Как насчет строчных элементов? 503
Как создать плавающий элемент 507
Поэкспериментируем над плавающим элементом в гостевой 509
Новый сайт для Starbuzz 511
Посмотрим на разметку 512
А теперь посмотрим на стиль 514
Переведем Starbuzz на следующий уровень 515
Тест для Starbuzz 518
Решение проблемы двух колонок 519
Задание поля для области с основным содержимым 520
Тест 521
Ох, у нас появилась еще одна проблема 521
Вернемся к решению проблемы с наложением 523
Тест 524
Левее, выше, правее... 526
Быстрый тест 528
Дизайны с фиксированной и непостоянной шириной 529
Тест для фиксированного дизайна 530
Что находится между разметками с фиксированной и непостоянной шириной? Гибкая разметка, конечно же! 530
Тест для гибкой разметки 531
Как работает абсолютное позиционирование 532
Изменение CSS для Starbuzz-страницы 535
Теперь нужно подкорректировать с идентификатором main 536
Протестируем страницу с абсолютным позиционированием 537
Что мы можем сделать, или Не могли бы вы рассказать, как создать двухколоночную разметку, которая не будет постоянно портиться? 538
Один компромисс, на который вы можете пойти, чтобы решить проблему с нижним колонтитулом 539
Позиционирование награды 541
Еще кое-что, что вы должны знать об абсолютном позиционировании 544
Как работает фиксированное позиционирование 547
Размещение купона на странице 548
Использование отрицательного значения свойства left 549
Положительный тест для отрицательных значений 550
Знакомство с относительным позиционированием 551
Тест 552
Три колонки и более... 553

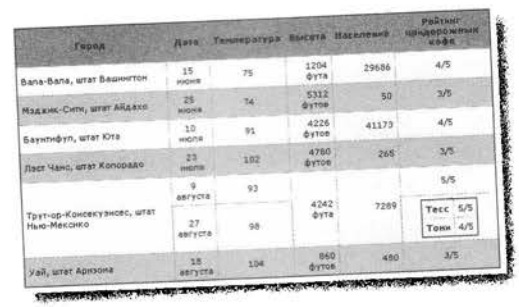
13 таблицы и большие списки

Представление в табличной форме

Если данные лучше оформить в виде таблицы... Пришло время научиться работать с устрашающими табличными данными. Каждый раз, когда вам нужно создать страницу, на которой выводится список документов вашей компании за последний год или перечень вашей коллекции чучел Beanie Baby (не беспокойтесь, мы никому не расскажем об этом), вы знаете, что нужно использовать XHTML. Но как? Мы готовы заключить с вами соглашение: выполняйте все наши инструкции — и за одну главу вы узнаете все секреты, позволяющие поместить данные прямо в XHTML-таблицы. Но есть кое-что еще: с каждой инструкцией мы будем представлять вам информацию из нашего эксклюзивного руководства по стилизации XHTML-таблиц. Если вы начнете прямо сейчас, то в качестве специального бонуса мы представим вам руководство по стилизации XHTML-списков. Не сомневайтесь, соглашайтесь прямо сейчас!

Как мы создаем таблицы в XHTML	559
Что создает браузер	561
Разделение таблицы	562
Добавление заголовка и краткого описания таблицы	565
Сделаем тест... и подумаем о стиле	566
Перед тем как мы перейдем к стилизации, поместим таблицу на страницу Тони	567
Стилизуем таблицу	568
Тест для стилизованной таблицы	569
Добавление границ	572
Как насчет цвета?	574
Мы говорили, что Тони сделал очень интересное открытие в Трут-ор-Консекуэнсес?	575
Посмотрим на таблицу Тони еще раз	576
Как сделать, чтобы ячейка охватила несколько строк	577
Новая и усовершенствованная таблица	579
Проблема в раю?	580
Тест для вложенной таблицы	582
Переопределение CSS для заголовков вложенной таблицы	584
Доведем сайт Тони до совершенства	585
Стилизация списка	586
Если нужен маркер особой формы?	587

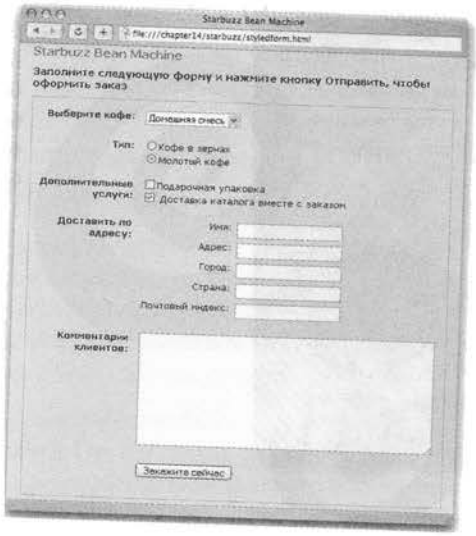




14 XHTML-формы

Переход на интерактивный режим

До сих пор ваше веб-общение было односторонним: от веб-страницы к пользователям. Разве не было бы здорово, если бы пользователи смогли отвечать вам? Именно на этом этапе вступают в действие формы XHTML. Как только вы снабдите свои страницы формами (прибегнув к определенной помощи веб-сервера), вы сможете собирать отзывы клиентов, принимать по Сети заказы, делать следующий ход в игре в режиме онлайн или проводить интернет-голосования. В этой главе вы познакомитесь с целой группой XHTML-элементов, предназначенных для создания веб-форм. Вы также узнаете кое-что о том, что происходит на сервере для поддержки форм и как сделать сами формы стильными.



Как работают формы	598
Как формы работают в браузере	599
Что вы пишете в XHTML	600
Что создает браузер	601
Как работает элемент <form>	602
Что может входить в форму	604
Подготовка к созданию формы для Bean Machine	610
Из чего состоит элемент <form>	610
Добавление элемента <form>	611
Как работают атрибуты name элементов формы	612
Вернемся к размещению элементов <input> в XHTML	614
Тест для формы	615
Добавим в форму еще несколько элементов <input>	615
Добавление элемента <select>	616
Тест для элемента <select>	617
Предоставьте клиенту выбор, хочет он молотый кофе или кофе в зернах	618
Стилизация переключателей	619
Дополнение формы	620
Добавление флажков и многострочного текстового поля	621
Завершающий тест	622
Проверим GET на практике	627
Использовать таблицы или нет	631
Размещение элементов формы в таблице	632
Тест для формы в виде таблицы	634
Стилизация формы и таблицы с помощью CSS	635
Последний тест	636
Что еще может входить в форму	637

Приложение

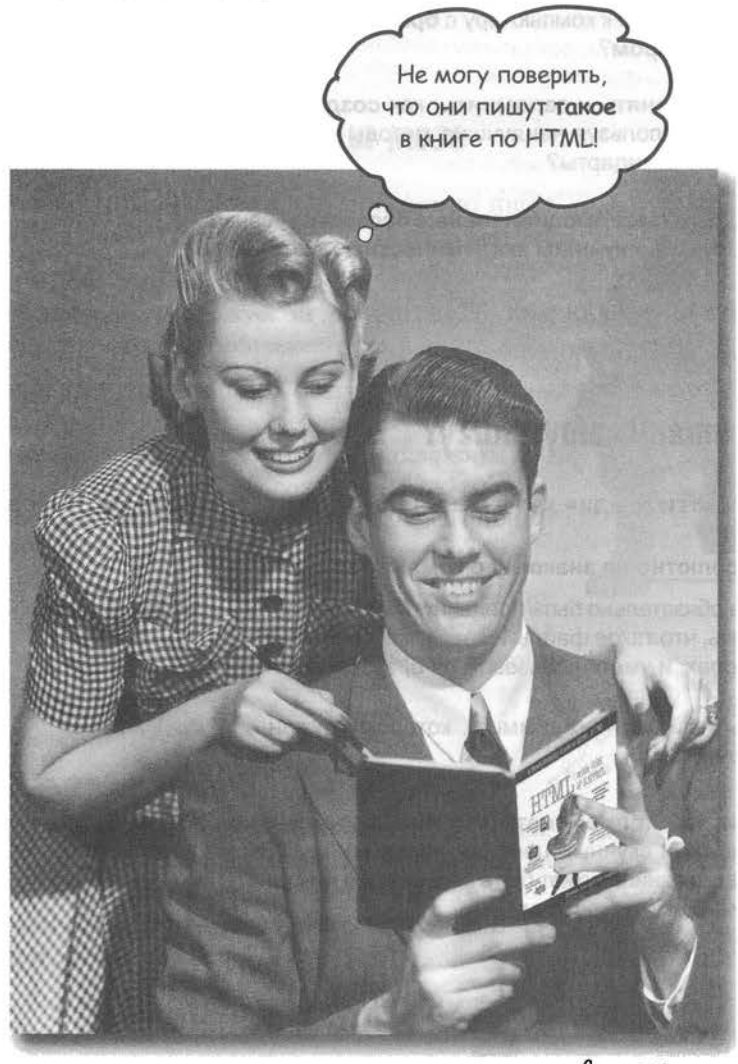
Топ-10 тем, которые не были освещены в этой книге

В книге мы рассмотрели очень много тем, и вы уже почти закончили свое обучение по ней. Однако мы не можем со спокойной душой отпустить вас, не рассказав еще кое-что напоследок. Как бы мы ни старались, но не сможем уместить в это маленькое приложение все то, что вам нужно знать, поэтому мы сначала включили в него все, что вам нужно знать об XHTML и CSS (все то, что не было освещено в предыдущих главах), уменьшив размер шрифта на 0,00004. Все вместилось, но прочесть это было нереально. В итоге мы все же выбросили большую часть информации и поместили в приложение только десять самых важных тем.



#1 Дополнительные типы селекторов 644
#2 Фреймы 646
#3 Мультимедиа и флэш 647
#4 Сервисные программы для создания веб-страниц 648
#5 Клиентские сценарии 649
#6 Серверные сценарии 650
#7 Поговорим о поисковых службах 651
#8 Подробнее о таблицах стиля для печати 652
#9 Страницы для мобильных устройств 653
#10 Блоги 654

Введение



В этой части мы ответим на насущный вопрос: «Итак, почему они пишут такое в книге по HTML?»

Для кого написана эта книга?

Если вы ответите «да» на все следующие вопросы...

1. У вас есть доступ к компьютеру с **браузером** и **текстовым редактором**?
2. Вы хотите **понять** и **запомнить**, как **создавать** веб-страницы, используя **наилучшие методы** и самые современные стандарты?
3. Вы предпочитаете **эмоциональные беседы** на званых обедах **сухим, скучным академическим лекциям**?

Если у вас есть доступ к любому компьютеру не более чем десятилетней давности выпуска, ответ — «да».

...значит, эта книга для вас.

Кому не стоит читать эту книгу?

Если вы ответите «да» хотя бы на один из следующих вопросов...

1. **Вы абсолютно не знакомы с компьютерами?**

   (Вам не обязательно быть продвинутым пользователем, но вы должны понимать, что такое файлы и папки, как работать в простых текстовых редакторах, и уметь пользоваться браузером.)

2. Вы опытный веб-программист, которому нужен **справочник**?

3. Вы **боитесь попробовать что-нибудь новенькое**? Вы предпочитаете один скучный цвет рисунку в клетку? Вы не верите, что техническая книга может быть серьезной и что теги HTML наделены человеческими качествами?



...то эта книга не для вас.

[Заметка от отдела маркетинга: вообще-то эта книга для любого, у кого есть деньги.]

Мы знаем, что вы думаете

«Как эта книга может быть серьезной?»

«Зачем здесь столько картинок?»

«Неужели так можно чему-то научиться?»

Мы также знаем, о чем думает ваш мозг

Ваш мозг страстно желает нововведений. Он постоянно ищет и *ждет* чего-нибудь необычного. Он так устроен, и это помогает вам выжить.

Сегодня у вас меньше шансов стать закуской для тигра. Но ваш мозг все еще начеку, а вы просто не замечаете этого.

Итак, что же ваш мозг делает с рутинными, обычными, нормальными ситуациями, с которыми вы сталкиваетесь? Все, что он *может*, — остановить их *вмешательство* в действительно *важную* для себя работу. Он не утруждает себя, чтобы сохранить что-то скучное. Все неинтересное и обычное просто проходит через фильтр «абсолютно неважных вещей».

Каким же образом ваш мозг узнает, что важно, а что нет? Предположим, вы выбрались на прогулку за город и перед вами выскочил тигр. Что произойдет в вашей голове и в вашем теле? Возбуждение нервных клеток. *Эмоциональный всплеск.*

И тогда ваш мозг понимает...

Это очень важно! Не забудь это!

Но представьте, что вы дома или в библиотеке. Здесь сухо и тепло и нет никаких тигров. Вы учитесь. Готовитесь к экзамену. Или пытаетесь разобраться в очень сложной технической теме, на что, как полагает ваш начальник, достаточно недели или в крайнем случае десяти дней.

Но вот одна проблема. Ваш мозг пытается оказать вам большую услугу. Он пытается *обеспечить сохранение* этой явно неважной информации таким образом, чтобы не загромождать и без того редкие ресурсы для ее хранения. Ресурсы, которые лучше потратить для хранения чего-то действительно *важного*. Как тигры. Как опасность огня. Как то, что вы больше никогда не должны кататься на лыжах в шортах.

И нет простого способа сказать мозгу: «Эй, мозг, большое спасибо, но какой бы скучной тебе ни казалась эта книга и что бы ты ни думал сейчас, я действительно хочу, чтобы ты запомнил все это».



Мы считаем, что читатель этой книги учится

Итак, что же нужно для того, чтобы *научиться* чему-либо. Сначала вы должны это *узнать*, а затем убедиться, что не *забыли*. Дело вовсе не в зубрежке. Последние исследования в областях когнитологии, нейробиологии и педагогической психологии показали, что для *обучения* необходимо намного больше, чем просто текст на странице. Мы знаем, что заставит ваш мозг заработать в полную силу.

Некоторые принципы этой книги

Добиваться визуализации. Изображения запоминаются намного лучше, чем слова, и делают процесс обучения более эффективным. Кроме того, визуализация делает предмет изучения более понятным.



Лучше поместить надписи внутри графиков, к которым они имеют отношение, или рядом с ними, чем внизу текущей страницы или на следующей странице. Тогда учащиеся будут в состоянии в два раза быстрее решить задачу по этому материалу.

Использовать разговорный стиль и обращение к читателю. Новейшие исследования показали, что студенты выполняют тесты после изучения материала до 40 % лучше, если в книге идет обращение напрямую к читателю, от первого лица и с использованием разговорного стиля вместо официального. Рассказывай историю вместо чтения лекции. Используй шутливый язык. Не будь слишком серьезным. Кто сильнее привлечет внимание: интересный собеседник на званом обеде или лектор?



Стимулировать ученика мыслить более глубоко. Иными словами, пока вы активно напрягаете свои извилины для обучения, в вашей голове больше ничего не происходит. Читатель должен иметь хорошую мотивацию учиться, быть заинтересованным, любознательным. Он должен с воодушевлением решать задачи, делать выводы и осваивать новые темы. Для этого ему нужны задачи, вопросы и упражнения, в которых есть над чем задуматься, и действия, для которых необходимо напрягать оба полушария головного мозга и даже использовать различные органы чувств.



Захватывать и удерживать внимание читателя. Каждый из нас знаком с таким: «Я действительно хочу это выучить, но засыпаю после первой же страницы». Ваш мозг обращает внимание на все необыкновенное, интересное, странное, броское, неожиданное. Изучение новых, трудных для понимания технических вопросов не должно быть скучным. Ваш мозг намного быстрее освоит новую информацию, если она не **будет нудной.**

Использовать эмоциональную составляющую. Вы знаете, что способность запоминать что-либо намного выше, если затрагиваются чувства? Вы запоминаете то, что вас волнует. Вы запоминаете свои ощущения. Нет, мы, конечно же, не говорим о душещипательных историях про мальчика и его собаку. Мы говорим об удивлении, любознательности, забаве, о возникающем у вас вопросе «О, как же это возможно?..» и таком чувстве, как «Я гений!», которое приходит после того, как вы разрешили головоломку, выучили что-то, что остальным кажется очень сложным, или реализовали что-то, что инженер Боб, считающий себя намного более продвинутым в этой области, не смог сделать.



Метапознание: учимся учиться



Если вы действительно хотите чему-то научиться, и притом научиться быстрее и глубже, подумайте над тем, как вы думаете. Изучите то, как вы учите.

Большинство из нас в детстве не проходили курса по метапознанию, или теории обучения. *Предполагалось*, что мы будем обучаться, но вряд ли нас *учили* это делать.

Но мы предполагаем, что если вы держите в руках эту книгу, то действительно хотите научиться создавать веб-страницы. И, вероятно, вы не хотите тратить много времени, а хотите *запомнить* то, что прочитаете, и затем суметь применить это на практике. А для этого нужно *понять* прочитанное. Для того чтобы вынести как можно больше из этой (или любой другой) книги или иного источника знаний, возьмите на себя ответственность за работу своего мозга.

Хитрость заключается в том, чтобы внушить мозгу, что новый материал, который вы изучаете, действительно важен. Является ключевым для вашего благополучия. Настолько же важен, как тигр. В противном случае вы будете постоянно воевать со своим мозгом, прилагая все усилия для запоминания новой информации и ее сохранения.

Итак, как же внушить мозгу, что HTML и CSS так же важны, как и тигр?

Существует два способа: нудный и медленный и более быстрый и эффективный. Медленный способ использует постоянное повторение. Вы наверняка знаете, что способны выучить и запомнить даже самый скучный материал, если постоянно повторять одно и то же. При достаточном количестве повторений ваш мозг скажет: «Я не думаю, что это важно для него, но он постоянно смотрит на это. Поэтому я допускаю, что это может иметь для него значение».

Быстрый способ – сделать что-нибудь, что ***увеличит активность работы мозга***, и особенно различные способы его работы. Описанное выше – важная составляющая решения проблемы, и уже доказано, что все это поможет вашему мозгу оказать вам услугу. Например, исследования показали, что если расположить слова *внутри* картинки, которую они описывают (вместо того чтобы помещать их в какой-то другой части страницы, например в заголовке или основном тексте), то это мотивирует мозг попытаться понять, как же взаимосвязаны слова и картинка, что, в свою очередь, приводит к возбуждению большего количества нервных клеток. Чем больше нервных клеток возбудится, тем больше шансов, что мозг воспримет информацию как ту, на которую нужно обратить внимание и, возможно, сохранить.

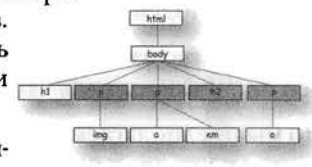
Разговорный стиль помогает усвоить информацию, так как люди имеют тенденцию лучше концентрироваться, если осознают, что беседуют с кем-то, а не когда они изучают материал самостоятельно.

Удивительно, но вашему мозгу неважно, что беседа идет между вами и книгой! С другой стороны, если стиль изложения официальный и скучный, то мозг воспринимает информацию так, будто вы сидите на лекции в зале, полном пассивных слушателей. И нет необходимости бодрствовать.

Но картинки и разговорный стиль – это только начало.

Вот что МЫ делали

Мы использовали ***рисунки***, потому что ваш мозг больше настроен на восприятие изображений, а не текста. С точки зрения мозга, рисунок действительно стоит 1024 слов. А еще лучше, если текст и изображение работают вместе. Мы добавляли текст внутрь картинки, потому что мозг работает более эффективно, если текст находится внутри того, что он описывает, а не, например, в названии или еще где-то.



Мы использовали ***повторение***, говоря об одном и том же разными способами и применяя различные формы представления информации. Так повышается вероятность того, что информация будет закодирована в нескольких зонах вашего мозга.

Мы использовали ***необычные*** понятия и рисунки, так как ваш мозг хорошо усваивает все новое. Рисунки и идеи наделялись хотя бы некоторым ***эмоциональным*** содержанием, потому что ваш мозг тесно связан с биохимией эмоций. Если вы что-то почувствуете, то есть больше вероятности, что вы это запомните. Пусть даже это всего лишь ***шутка***, ***сюрприз*** или ***интересный факт***.



Мы использовали личный ***разговорный стиль***, потому что мозг больше настроен на усвоение информации, если полагает, что вы с кем-то беседуете, а не пассивно слушаете. Это происходит даже тогда, когда вы читаете.

Мы включили в книгу более 100 ***упражнений***, так как ваш мозг лучше учит и запоминает, когда вы что-то ***делаете***. Мы сделали упражнения сложными, но выполнимыми, так как именно такие упражнения предпочитает большинство людей.



Мы использовали ***различные стили обучения***, потому как одни предпочитают изучение шаг за шагом, в то время как другие хотят сначала получить общее представление о проблеме, а третьи просто хотят увидеть пример. Однако, несмотря на личные предпочтения, каждому будет полезно ознакомиться с одним и тем же материалом различными способами.



Мы включили в книгу информацию для ***обоих полушарий мозга***, потому что чем большую часть мозга вы используете, тем более успешно вы учитесь и запоминаете, и тем дольше можете концентрироваться на изучаемой теме. Поскольку при работе одного полушария головного мозга второе может отдохнуть, вы можете эффективно учиться более продолжительное время.

Мы также включили в издание ***истории***, которые представляют ***несколько точек зрения***, так как мозг начинает лучше усваивать материал, если вынужден оценивать и анализировать.

Мы включили в книгу ***сложные задачи***, упражнения и ***вопросы***, на которые не всегда есть однозначный ответ, потому что ваш мозг лучше учится и запоминает, если ему есть над чем поработать. Согласитесь, вы не можете привести свое тело в спортивную форму, просто наблюдая за тем, как другие занимаются в спортзале. Мы сделали все возможное, чтобы обеспечить для вас ***максимальный результат при условии вашей прилежной работы***. Иными словами, вам не придется тратить время на разбор сложных для понимания упражнений, трудного, нагруженного непонятными терминами или излишне сжатого текста.



Мы использовали ***людей*** в историях, примерах, рисунках и т. д., потому что вы тоже человек. И ваш мозг обращает больше внимания на людей, чем на предметы.

Мы использовали принцип ***80/20***. Мы полагаем, что если вы собираетесь стать опытным веб-программистом, то эта книга не станет для вас единственным источником знаний. Поэтому мы не рассказываем обо всем. Речь пойдет лишь о том, что действительно важно.



Вырежьте листок с напоминанием и наклейте его на холодильник.

А вот что можете сделать ВЫ, чтобы заставить свой мозг работать

Итак, мы свою часть работы сделали. Остальное зависит от вас. Следующие подсказки — лишь отправная точка; прислушивайтесь к своему мозгу и выясняйте, что для вас работает, а что — нет. Пробуйте какие-то новые способы.

1 Не торопитесь все прочитать. Чем больше вы поймете, тем меньше вам придется запоминать.

Не *читайте* просто так. Делайте паузы и вдумывайтесь. Если в книге вам задан вопрос, не пропускайте его. Представьте, что кто-то действительно обращается к вам с вопросом. Чем более глубоко вы заставляете свой мозг вникать в суть проблемы, тем выше вероятность того, что вы запомните и усвоите материал.

2 Выполняйте упражнения. Делайте свои собственные заметки.

Мы включили в книгу упражнения. Но если мы будем выполнять их вместо вас, то это то же самое, как если бы кто-то вместо вас занимался спортом. Не *смотрите* на упражнения просто так. Пользуйтесь карандашом. Существует множество доказательств тому, что физическая активность *во время* обучения повышает его эффективность.

3 Читайте «Часто задаваемые вопросы»

Все вопросы важны. ***Они являются частью основного материала!*** Не пропускайте их.

4 Не читайте других книг перед сном. Или, по крайней мере, других *интересных* книг.

Часть обучения (особенно сохранение информации в долговременную память) происходит уже *после* того, как вы отложили книгу. Вашему мозгу нужно время для обработки и запоминания информации. Если вы в это время добавите какую-то новую информацию, то часть предыдущей будет потеряна.

5 Пейте воду. Много воды.

Ваш мозг лучше всего работает в жидкой среде. Обезвоживание (которое может случиться еще до того, как вы почувствуете жажду) снижает функцию запоминания.

6 Проговаривайте все, что учите. Говорите громко.

За речевую активность отвечает отдельная зона мозга. Если вы пытаетесь что-то понять или хотите лучше что-то запомнить, громко проговорите это. А лучше попытаться объяснить это вслух кому-то другому. В результате вы будете учиться быстрее и, возможно, обнаружите что-то новое, чего не замечали в процессе чтения.

7 Прислушивайтесь к своему мозгу.

Следите за тем, чтобы мозг не перегружался информацией. Если вы ловите себя на том, что читаете невдумчиво или забываете то, что сейчас прочитали, значит, настало время отдохнуть. Перечитывая одно и то же снова и снова, пытаясь впихнуть в себя больше информации, вы не выучите быстрее и даже, возможно, навредите всему процессу обучения.

8 *Пытайтесь* прочувствовать прочитанное!

Вашему мозгу необходимо осознавать, что информация *важна*. Пытайтесь увлечься сюжетом. Выдумывайте свои собственные комментарии к фотографиям. Лучше, например, поворчать, что шутка несмешная, чем остаться полностью равнодушным.

9 *Создавайте* что-нибудь!

Применяйте прочитанное для проектирования чего-либо нового или для переработки старых проектов. Просто делайте *что-нибудь* (кроме заданий и упражнений из книги), чтобы приобрести опыт. Все, что вам нужно, — это карандаш и задача, которую нужно решить. Задача, которая может быть полезна для изучения HTML и CSS.

Примите к сведению

Эта книга — учебник, а не просто справочник. Мы рассмотрели все ситуации, которые могут возникнуть в процессе обучения, и создали книгу с учетом этого. Читая ее первый раз, вы должны начать с самого начала, поскольку мы постоянно предполагаем, что вы уже знаете все вышеизложенное.

Мы начинаем с изучения исходных положений HTML, затем переходим к HTML 4.01, основанному на общепринятых стандартах, и только потом к XHTML.

Чтобы писать на HTML, основанном на общепринятых стандартах, или на XHTML, нужно понимать много технических деталей, которые бесполезны, пока вы еще только знакомитесь с основами языка. Наш подход заключается в том, чтобы сначала научить вас основным понятиям HTML (не останавливаясь на деталях), а затем, когда будете все хорошо понимать, научить вас писать на HTML, основанном на общепринятых стандартах, и на XHTML. В этом есть дополнительная польза, так как технические детали имеют больше смысла после того, как изучены основы.

Важно также, что к моменту использования CSS вы уже будете писать на HTML, основанном на общепринятых стандартах, и на XHTML.

Мы не описываем все подряд элементы и атрибуты HTML и свойства CSS, которые когда-либо были созданы.

Существует *множество* элементов и атрибутов HTML и свойств CSS. Бесспорно, все они интересны, но нашей целью было написание книги, которая весит меньше, чем ее читатель, поэтому мы не описываем их все. Мы обращаем внимание только на основные элементы HTML и свойства CSS, которые действительно *важны* для начинающих, и надеемся, что вы по-настоящему поймете, как и когда их использовать. В любом случае после прочтения этой книги вы сможете взять любой справочник и быстро ознакомиться со всеми элементами и свойствами, которые мы пропустили.

В книге четко разделены понятия структуры и представления ваших страниц.

В наши дни при создании серьезных веб-страниц применяется HTML и XHTML для структурирования их содержимого, а также CSS для стилизации и дизайна. В веб-страницах 1990-х годов часто использовалась другая модель, где HTML применялся и для структуризации, и для стилизации. Мы научим вас использовать HTML для структуризации, а CSS для стилизации; мы не видим смысла учить вас дурным устаревшим привычкам.

Мы поощряем использование нескольких браузеров.

Пока мы учим вас писать на HTML, CSS и XHTML, основанных на стандартах, вы по-прежнему (и, скорее всего, и далее) будете сталкиваться с небольшими различиями в способах отображения страниц разными браузерами. Поэтому

мы советуем выбрать по крайней мере два современных браузера и тестировать в них веб-страницы. Это научит вас понимать различия между браузерами и создавать страницы, которые хорошо работают во множестве браузеров.

Мы часто используем название «элемент» вместо названия «тег».

Вместо того чтобы говорить «тег **<a>**», мы говорим «элемент **<a>**». Пусть формально это некорректно (так как **<a>** — это открывающий тег, а не целый элемент), но это улучшает читабельность текста, и мы всегда используем слово «элемент» во избежание путаницы.

Упражнения обязательны для выполнения.

Задания и упражнения — это не дополнения, а часть основного материала книги. Одни из них предназначены для облегчения запоминания, другие — для применения выученного материала. ***Не пропускайте упражнения.***

Повторения используются специально, и это важно.

Одна отличительная особенность книги — то, что мы *действительно* хотим донести информацию до читателя. Мы также хотим, чтобы читатель ничего не забыл после прочтения книги. Большинство справочников не ставят своей целью запоминание и воспроизведение читателем содержащейся в них информации, но эта книга готова *обучать*, поэтому некоторые понятия появляются не один раз.

Примеры настолько компактны, насколько это возможно.

Наши читатели жаловались на то, что в некоторых примерах приходится просматривать 200 строк листинга, чтобы добраться до двух действительно важных для понимания. Большинство примеров в этой книге очень краткие, и их суть понятна сразу. Не ожидайте, что все примеры будут корректными, они пишутся специально для обучения и не всегда работают полностью правильно.

Все файлы с примерами мы выложили в Интернете, и вы можете скачать их на сайте `http://www.headfirstlabs.com/books/hfhtml/`.

К упражнениям «Мозговой штурм» ответы не прилагаются.

У некоторых из них правильных ответов вовсе не существует, в других случаях используйте свой опыт, чтобы определить правильный ответ. В некоторых примерах повышенной сложности есть подсказки, которые приведут к правильному ответу.

Технические рецензенты



Наши рецензенты

Выражаем большую благодарность нашей команде технических рецензентов. **Йоханнес де Йонг** (Johannes de Jong), организовавший и возглавивший проект, исполнял роль «папы» и сделал всю работу слаженной. **Паулина Макнамара** (Pauline McNamara), соуправляющий проекта, привела все в систему и была первой, кто обратил внимание на то, что примеры не совсем актуальны для наших дней. Вся команда подтвердила, как сильно нам помогли их техническая экспертная оценка и внимание к деталям. **Валентин Креттас** (Valentin Crettaz), **Барни Мариспини** (Barney Marispini), **Маркус Грин** (Marcus Green), **Айк Ван Атта** (Ike Van Atta), **Дэвид О'Мера** (David O'Meara), **Джо Коньор** (Joe Conior) и **Кори Макглоун** (Corey McGlone) не пропустили ни одной мелочи, анализируя книгу, что сделало ее намного лучше. Ребята, вы молодцы! Благодарим также Кори (Corey) и Паулину (Pauline), которые не давали нам незаметно перейти к слишком официальному стилю изложения. Отдельная благодарность JavaRanch за услуги размещения информации в Сети.

Огромное спасибо **Луизе Барр** (Louise Barr), нашему веб-дизайнеру, которая упорно следила за дизайном и за использованием нами XHTML и CSS.

Благодарности*



И еще про технические рецензии

Мы также выражаем большую благодарность нашему уважаемому техническому рецензенту ***Дэвиду Пауэрсу*** (David Powers). У нас были действительно сложные отношения с ним, потому что он заставлял нас усердно работать, зато какой результат! Поэтому *оно того стоило.* Честно говоря, основываясь на его комментариях, мы сделали существенные изменения в этой книге, и в техническом плане это сделало ее в два раза лучше.

В O'Reilly:

Выражаем большую благодарность нашему редактору ***Бретту Маклафлину*** (Brett McLaughlin), который отредактировал всю книгу так, что она стала ясной и понятной и может быть быстро освоена всей семьей. Бретт потратил много времени, усердно редактируя книгу (что нелегко для этой серии). Спасибо, Бретт, этой книги не было бы без тебя.



Наша искренняя благодарность всей команде O'Reilly: ***Грег Коррин*** (Greg Corrin), ***Гленн Бизиньяни*** (Glenn Bisignani), ***Тони Артузо*** (Tony Artuso), ***Кайл Харт*** (Kyle Hart) возглавили отдел маркетинга, и мы признательны им за их замечательный новый подход. Благодарим ***Элли Волкхаусен*** (Ellie Volkhausen) за притягательный дизайн обложки, который до сих пор нам служит, а также ***Карэн Монтгомери*** (Karen Montgomery) за то, что поспособствовала тому, чтобы использовалась именно эта обложка. И, конечно же, благодарим ***Колена Гормана*** (Colleen Gorman) за основательное редактирование текста (и за сохранение шутливого стиля изложения). Книга не была бы столь красочной, если бы не ***Сью Виллинг*** (Sue Willing) и ***Клэр Клотье*** (Claire Cloutier). Кроме того, выражение признательности всем участвующим в создании книги было бы неполным без благодарности ***Майку Лукайдсу*** (Mike Loukides) за создание целой серии и ***Тиму О'Рейлли*** (Tim O'Reilly) за то, что он всегда был и продолжает быть с нами. Наконец, благодарим ***Майка Хендриксона*** (Mike Hendrickson) за то, что привлек нас всех в дружную семью создателей этой серии и доверил нам работу с ней.



И последнее, но далеко не маловажное: благодарим ***Кэти Сьерра*** и ***Берта Бэйтса*** – наших партнеров, которые создали серию. Спасибо вам, ребята, за доверие. Мы надеемся, что оправдали его. Трехдневная сессия была ярким моментом написания книги, и мы надеемся повторить ее снова. О, и не могли бы вы в следующий раз позвонить LTJ и пригласить его приехать в Сиэтл?

* Мы выражаем благодарность такому большому количеству людей, так как проверяем теорию, будто каждый из них купит по крайней мере один экземпляр книги, а возможно, больше, учитывая их родственников и знакомых. Если вы хотите, чтобы мы выразили вам благодарность в нашей следующей книге, и у вас большая семья, пишите нам.

Язык Сети



Единственное, что необходимо для того, чтобы успешно работать в Сети, — научиться говорить на ее специфическом языке: HyperText Markup Language (язык гипертекстовой разметки), или сокращенно HTML. Итак, приготовьтесь к нескольким урокам языка. После этой главы вы не только узнаете некоторые **базовые понятия** HTML, но и сможете разговаривать на этом языке, используя определенный **стиль**. Черт возьми, к концу этой книги вы сможете говорить на языке HTML так, будто выросли в Сети!

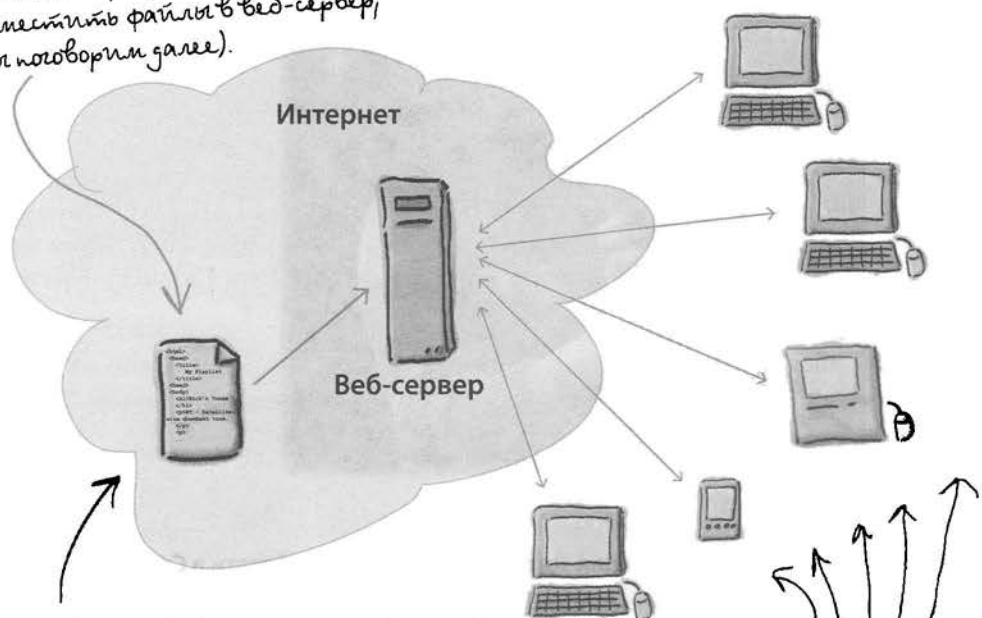
Сеть
~~Видео~~ а убило радиозвезду

Хотите поделиться своей новой идеей? Продать что-то? Просто нужен творческий выход? Обратитесь к Сети – нет необходимости говорить вам, что она стала универсальной формой коммуникации. Кроме того, это форма коммуникации, в которой **ВЫ** можете участвовать.

Но если вы действительно хотите эффективно использовать возможности Сети, то вам необходимо узнать кое-какие вещи про **HTML**, не говоря уже о том, что нужно понимать, как работает сама Сеть. Начнем с самого главного.

Для получения веб-страницу вы создаете файлы, написанные на языке HyperText MarkupLanguage (сокращенно HTML) и помещаете их в веб-сервер (о том, как поместить файлы в веб-сервер, мы поговорим далее).

После того как вы поместили созданные файлы в веб-сервер, любой браузер сможет через Интернет отыскать ваши веб-страницы.

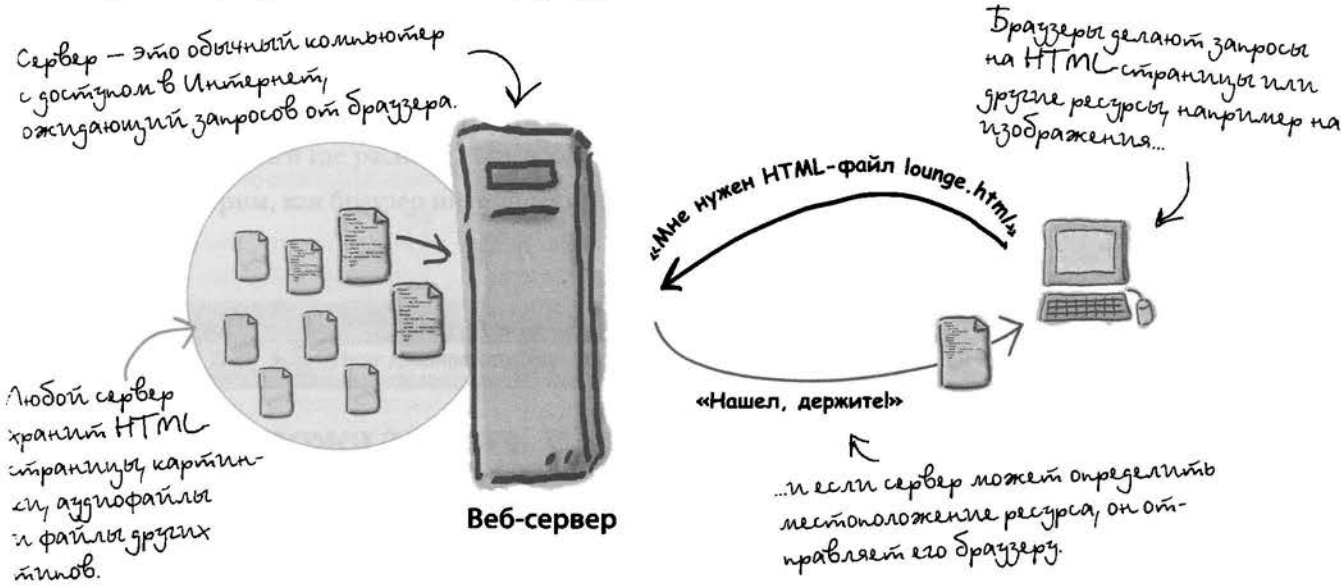


Язык HTML вашей веб-страницы говорит браузеру все, что ему необходимо знать для отображения этой страницы. А если вы выполнили свою работу хорошо, то ваши страницы будут так же корректно отображаться в карманных компьютерах и мобильных устройствах и смогут работать с речевыми браузерами и экранными дикторами для людей с плохим зрением.

Существует множество компьютеров и устройств, подключенных к Интернету и отображающих веб-страницы. Очень важно, что ваши друзья, члены семьи, болельщики и потенциальные покупатели могут работать за этими компьютерами.

Что делает Веб-сервер

Веб-серверы непрерывно работают в Интернете, неустанно ожидая запросов от браузеров. Каких именно запросов? Запросов на веб-страницы, изображения, аудиофайлы и, возможно, даже видеоролики. Когда сервер получает запрос на какой-нибудь из таких ресурсов, он находит этот ресурс и высылает его браузеру.



Что делает браузер

Вы уже знаете, как работает браузер: вы бродите по различным сайтам в Сети, щелкая на ссылках для перехода на различные страницы. Такой щелчок служит поводом для того, чтобы ваш браузер сделал запрос на HTML-страницу веб-серверу, получил ответ на свой запрос и отобразил эту страницу в своем окне.



Но каким образом браузер узнает о том, как именно отображать страницу? Вот здесь начинает работать язык HTML. Он говорит браузеру все о содержании и структуре страницы. Посмотрим, как это работает...

Что пишете Вы (HTML-код)

Итак, вы знаете, что HTML – это ключ, благодаря которому браузер отображает страницы, но как именно выглядит HTML и что он делает?

Давайте посмотрим на небольшой фрагмент HTML-кода. Представьте, что вы собираетесь создать веб-страницу – рекламную *гостевую страничку* с хорошими мелодиями, освежающими напитками и беспроводным доступом. Рассмотрим код, который вы напишете на HTML:

```
<html>
  <head>
    <title>Гостевая Head First</title>
  </head>
  <body>
    <h1>Добро пожаловать в гостевую Head First</h1>
    <img src="drinks.gif">
    <p>
      Заходите к нам каждый вечер, чтобы попробовать
      освежающие эликсиры, поболтать и, возможно,
      <em>станцевать разок-другой</em>.
      Всегда обеспечен беспроводной доступ
      (захватите с собой свой ноутбук).
    </p>
    <h2>Указатели</h2>
    <p>
      Вы найдете нас в самом центре Webville.
      Присоединяйтесь к нам!
    </p>
  </body>
</html>
```



Мы не ожидаем, что вы уже знаете HTML.

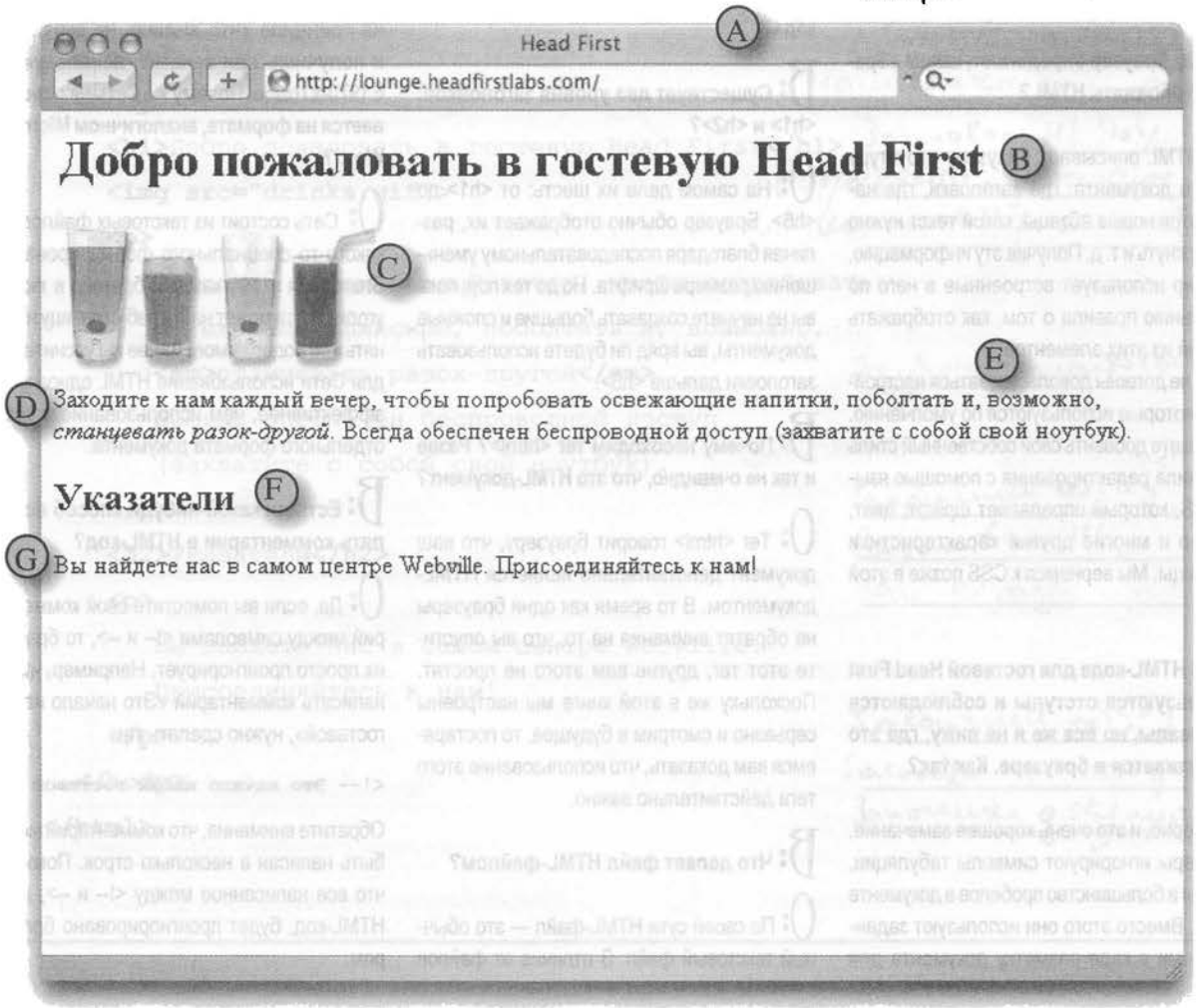
На данном этапе вы просто должны представлять, как выглядит HTML. Далее мы постепенно опишем все более детально. А пока изучайте HTML-код и наблюдайте, как он отображается в окне браузера. Обязательно обращайте внимание на каждое замечание и ссылку, а также на то, как и где они отображаются в браузере.

Что создает браузер

Когда браузер читает вашу HTML-страницу, он интерпретирует все *теги*, которые окружают основной текст программы. Теги — это обычные слова или символы в угловых скобках, например, **<head>**, **<p>**, **<h1>** и т. д. С помощью тегов браузер распознает *структуру и значение* вашего текста. Итак, вместо того, чтобы просто передавать браузеру кусок текста, благодаря HTML вы можете использовать теги, чтобы сказать браузеру, какая часть текста является названием, где начинается новый абзац, что нужно подчеркнуть и где расположить изображения.

Давайте проверим, как браузер интерпретирует теги в гостевой Head First.

Обратите внимание, как каждый тег в HTML соответствует тому, что отображается в браузере.



дальше ▸

Часто Задаваемые Вопросы

В: Итак, HTML — это всего лишь набор тегов, которые я расставляю вокруг основного текста?

О: Для начинающих. Запомните, что HTML — это HyperText *Markup* Language (язык гипертекстовой *разметки*), то есть он дает возможность «пометить» ваш текст тегами, которые описывают браузеру его структуру. Помимо этого, у HTML есть *гипертекстовый* аспект, о котором мы поговорим далее в книге.

В: Как браузер определяет, каким образом отображать HTML?

О: HTML описывает браузеру структуру вашего документа: где заголовки, где начинаются новые абзацы, какой текст нужно подчеркнуть и т. д. Получив эту информацию, браузер использует встроенные в него по умолчанию правила о том, как отображать каждый из этих элементов.

Но вы не должны довольствоваться настройками, которые используются по умолчанию. Вы можете добавить свой собственный стиль и правила редактирования с помощью языка CSS, который определяет шрифт, цвет, размер и многие другие характеристики страницы. Мы вернемся к CSS позже в этой главе.

В: В HTML-коде для гостевой Head First используются отступы и соблюдаются интервалы, но все же я не вижу, где это отображается в браузере. Как так?

О: Верно, и это очень хорошее замечание. Браузеры игнорируют символы табуляции, абзацы и большинство пробелов в документе HTML. Вместо этого они используют заданную вами в коде разметку документа для определения мест, где появляется разрыв строки и конец абзаца.

Спрашивается, для чего же мы форматируем текст, если браузер все равно проигнорирует это? Для того чтобы нам было легче читать HTML-документ, когда мы его редактируем. Поскольку обычно HTML-документ со временем усложняется, то, обнаружив в нем при редактировании пробелы, символы табуляции и абзацы, вы поймете, что они действительно улучшают читабельность кода.

В: Существует два уровня заголовков: <h1> и <h2>?

О: На самом деле их шесть: от <h1> до <h6>. Браузер обычно отображает их, различая благодаря последовательному уменьшению размера шрифта. Но до тех пор, пока вы не начнете создавать большие и сложные документы, вы вряд ли будете использовать заголовки дальше <h3>.

В: Почему необходим тег <html>? Разве и так не очевидно, что это HTML-документ?

О: Тег <html> говорит браузеру, что ваш документ действительно является HTML-документом. В то время как одни браузеры не обратят внимания на то, что вы опустите этот тег, другие вам этого не простят. Поскольку же в этой книге мы настроены серьезно и смотрим в будущее, то постараемся вам доказать, что использование этого тега действительно важно.

В: Что делает файл HTML-файлом?

О: По своей сути HTML-файл — это обычный текстовый файл. В отличие от файлов специальных текстовых редакторов, он не имеет возможности форматирования. Обычно в конце имени файла добавляется .html или .htm (в системах, которые поддерживают только три буквы в расширении файла), чтобы система определила тип файла. Но, как вы понимаете, действительно имеет значение лишь содержимое файла.

В: Мне кажется, что размечать документ — глупо. Программы, основанные на принципе «что видишь на экране, то и получишь при печати», используются с 1970-х годов. Почему же Сеть не основывается на формате, аналогичном Microsoft Word?

О: Сеть состоит из текстовых файлов без какого-то специального форматирования. Благодаря этому каждый браузер в любом уголке света может найти веб-страницу и «понять» ее содержимое. Далее вы уясните, что для Сети использование HTML однозначно эффективнее, чем использование любого отдельного формата документа.

В: Есть ли какой-нибудь способ вставлять комментарии в HTML-код?

О: Да, если вы поместите свой комментарий между символами <!-- и -->, то браузер их просто проигнорирует. Например, чтобы написать комментарий «Это начало нашей гостевой», нужно сделать так:

```
<!-- Это начало нашей гостевой -->
```

Обратите внимание, что комментарий может быть написан в несколько строк. Помните, что все написанное между <!-- и -->, даже HTML-код, будет проигнорировано браузером.

Возьми в руку карандаш

Вы ближе к изучению HTML, чем думаете...

Снова рассмотрим HTML-код для гостевой Head First. Взгляните на теги. Можете догадаться, что они говорят браузеру о содержимом документа? Напишите свои ответы справа или где-нибудь еще в свободном месте. Мы начали это делать за вас...

```
<html>                                                    Говорит браузеру, что это начало HTML-кода.
  <head>                                                  Начинается название веб-страницы (подробнее об этом далее).
    <title>Гостевая Head First</title>                    _____
  </head>                                                 _____
  <body>                                                  _____
    <h1>Добро пожаловать в гостевую Head First</h1>       _____
    <img src="drinks.gif">                                _____
    <p>                                                   _____
       Заходите к нам каждый вечер, чтобы попробовать
       освежающие эликсиры, поболтать и, возможно,
       <em>станцевать разок-другой</em>.                  _____
       Всегда обеспечен беспроводной доступ
       (захватите с собой свой ноутбук).
    </p>                                                  _____
    <h2>Указатели</h2>                                    _____
    <p>                                                   _____
       Вы найдете нас в самом центре Webville.
       Присоединяйтесь к нам!
    </p>                                                  _____
  </body>                                                 _____
</html>                                                   _____
```



Возьми в руку карандаш

Ответы

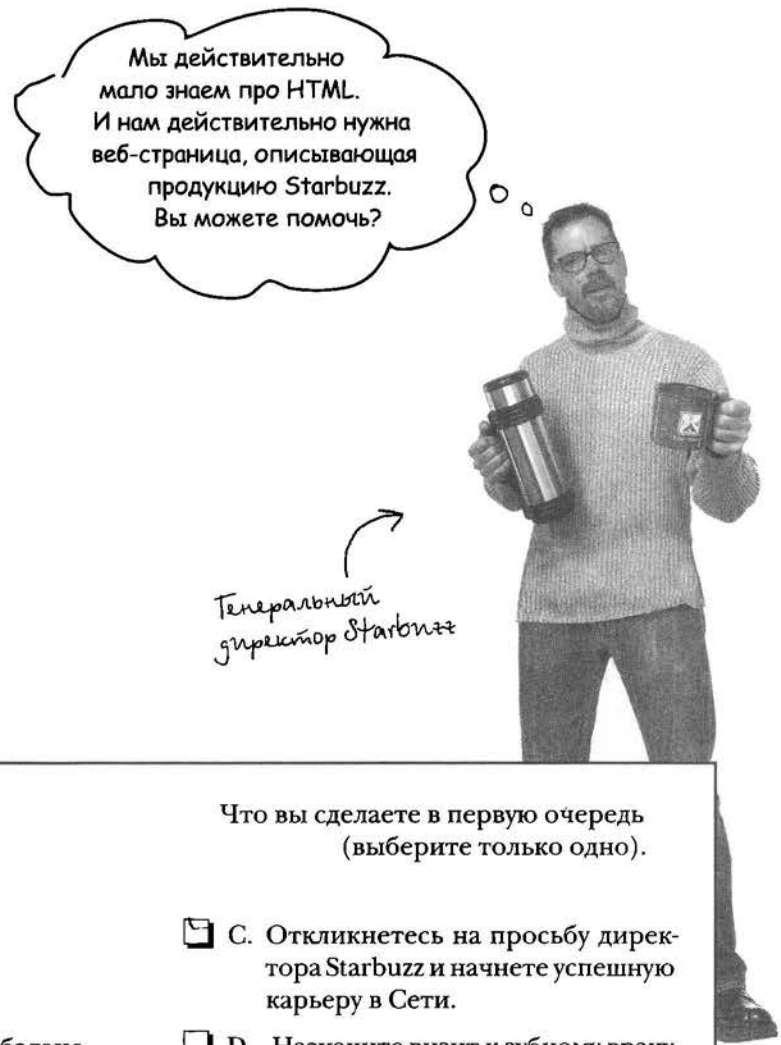
```
<html>                                          Говорит браузеру, что это начало HTML-кода.
  <head>                                        Начинается название веб-страницы.
    <title>Гостевая Head First</title>          Начинается заглавие страницы.
  </head>                                       Конец заголовка.
  <body>                                        Начинается основная часть страницы.
    <h1>Добро пожаловать в гостевую Head First</h1>   Говорит браузеру, что это заглавие.
    <img src="drinks.gif">                      Добавляет изображение drinks.gif.
    <p>                                         Начинается абзац.
      Заходите к нам каждый вечер, чтобы попробовать
      освежающие эликсиры, поболтать и, возможно,
      <em>станцевать разок-другой</em>.         Используется курсивный шрифт
      Всегда обеспечен беспроводной доступ      для предложения потанцевать.
      (захватите с собой свой ноутбук).
    </p>                                        Заканчивается абзац.
    <h2>Указатели</h2>                          Говорит браузеру, что это подзаголовок.
    <p>                                         Начинается новый абзац.
      Вы найдете нас в самом центре Webville.
      Присоединяйтесь к нам!
    </p>                                        Заканчивается абзац.
  </body>                                       Заканчивается основная часть страницы.
</html>                                         Говорит браузеру, что это конец HTML-кода.
```



Большая перемена в кафе Starbuzz

Кафе Starbuzz известно как самое быстроразвивающееся кафе. Если вы увидели одно такое кафе в своем районе, то, оглянувшись, скорее всего, заметите и второе.

На самом деле, они растут так быстро, что даже умудрились обойтись без веб-страницы в Интернете... Поэтому не исключено, что, пока вы покупаете чай Starbuzz Chai, вы случайно сталкиваетесь с директором Starbuzz...



МОЗГОВОЙ ШТУРМ

Что вы сделаете в первую очередь (выберите только одно).

- ❑ A. Купите собаку.
- ❑ B. Наконец подведете баланс на своем счете.
- ❑ C. Откликнетесь на просьбу директора Starbuzz и начнете успешную карьеру в Сети.
- ❑ D. Назначите визит к зубному врачу.



Замечательно! Мы так рады, что вы решили помогать нам*. Вот что нам нужно на первой странице...

Генеральный директор черкнул что-то на салфетке и протянул ее вам...

Starbuzz Coffee

Спасибо, что согласились помочь! Нам нужна простая веб-страница (смотрите ниже), на которой будут названия, цены и описания напитков.

Домашняя смесь, $1,49
Мягкая, нетерпкая смесь различных сортов кофе из Мексики, Боливии и Гватемалы

Кофе мокко, $2,35
Эспрессо, взбитое молоко и шоколадный сироп

Капучино, $1,89
Смесь эспрессо и взбитого молока с добавлением пены

Чай, $1,85
Ароматный напиток из черного чая, специй, молока и меда

Возьми в руку карандаш

Взгляните на салфетку. Можете определить на ней *структуру* текста? Иными словами, вы четко видите заголовки? Абзацы? Не отсутствует ли название?

Зафиксируйте на салфетке (карандашом) структуру, которую видите, и добавьте все, что пропущено.

Ищите ответы в конце главы 1.

* Если же вы выбрали пункты A, B или D на предыдущей странице, то мы рекомендовали бы вам подарить эту книгу какой-нибудь хорошей библиотеке, использовать ее для разжигания огня этой зимой или, так и быть, продать ее и получить какие-то деньги.

Создание Веб-страницы для Starbuzz

Конечно же, основная проблема в том, что вы никогда еще не создавали веб-страницы. Но ведь именно для этого вы начали учить HTML по нашей книге?

Не беспокойтесь, мы расскажем вам, что нужно делать дальше.

1. **Создайте HTML-файл, используя любимый текстовый редактор.**
2. **Наберите на клавиатуре меню, которое директор Starbuzz написал на салфетке.**
3. **Сохраните файл с именем index.html.**
4. **Откройте файл index.html в своем любимом браузере, сделайте шаг назад, и вы увидите, что случилось чудо.**



Создание HTML-файла (Mac)

Все HTML-файлы – это текстовые файлы. Для создания текстового файла нужна программа, которая позволяет создавать обыкновенный текст без использования причудливого форматирования и специальных символов. Вам нужен простой, чистый текст.

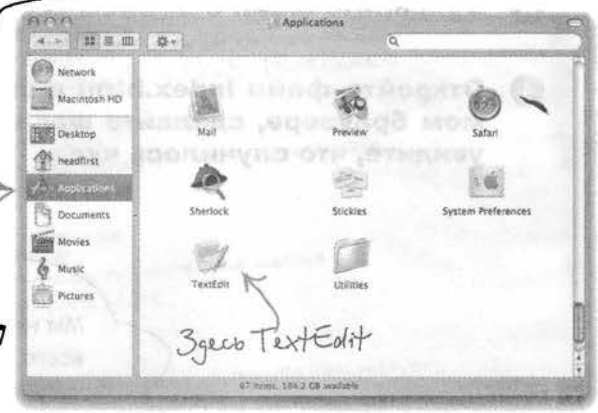
В этой книге мы используем редактор TextEdit в Mac. Тем не менее, если вы предпочитаете другой текстовый редактор, все тоже будет хорошо работать. А если вы работаете в Windows, то можете пропустить пару страниц и перейти сразу к инструкциям для Windows.



Шаг первый

Направляйтесь в папку Applications (Приложения).

Ярлык программы TextEdit находится в папке Applications (Приложения). Самый простой способ туда попасть – выбрать пункт New Finder Window (Поиск в новом окне) в меню Finder's File (Файлы поиска) и затем искать папку Applications (Приложения) прямо в ярлыках. Когда найдете ее, щелкните на ней кнопкой мыши.



Шаг второй

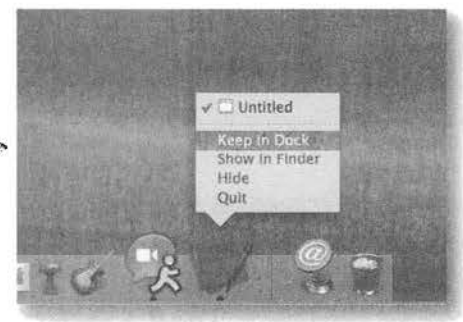
Найдите и запустите TextEdit.

Вероятно, в вашей папке Applications (Приложения) будет много программ. Прокручивайте список вниз до тех пор, пока не увидите TextEdit. Для запуска программы дважды щелкните на ее значке.

Шаг третий (необязательный)

Держите TextEdit на панели запуска.

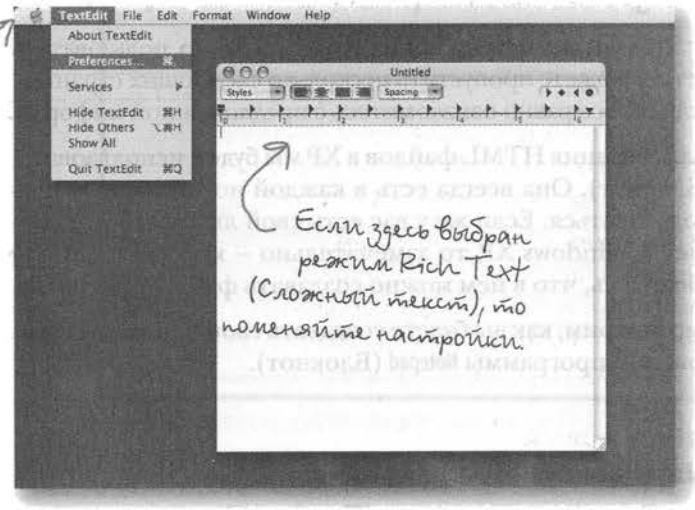
Щелкните (и держите кнопку мыши нажатой) на значке TextEdit на панели запуска (этот значок там появляется сразу после запуска программы). В появившемся меню выберите пункт Keep in Dock (Установить на панель запуска). Таким образом, значок TextEdit всегда будет на панели запуска и вам не придется каждый раз, когда нужно открыть эту программу, искать ее в папке Applications (Приложения).



Шаг четвертый

Поменяйте настройки в TextEdit.

По умолчанию в TextEdit выбран режим Rich Text (Сложный текст), это означает, что в ваш файл при сохранении будут добавляться форматирование и специальные символы. Однако вам это не нужно. Поэтому необходимо поменять настройки в TextEdit, чтобы текст в вашем файле сохранялся в обычном виде. Для этого в меню TextEdit выберите подменю Preferences (Настройки).



Шаг пятый

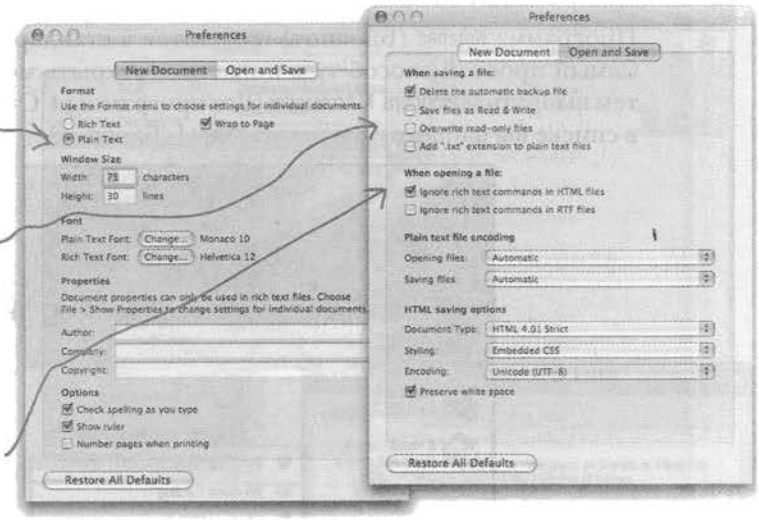
Установите режим Plain Text (Обычный текст).

В окне Preferences (Настройки) сделайте следующее.

Во-первых, выберите на вкладке New Document (Создание документа) в качестве используемого по умолчанию режим Plain Text (Обычный текст).

Во-вторых, убедитесь, что на вкладке Open and Save (Открытие и сохранение) снят флажок Add .txt extension to plain text files (Добавлять расширение .txt к файлам с обычным текстом).

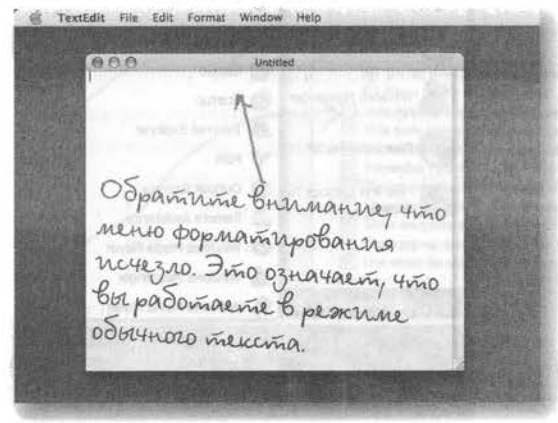
И последнее, убедитесь, что установлен флажок Ignore rich text commands in HTML files (Игнорировать команды усложнения текста в HTML-файлах).



Шаг шестой

Закройте программу и откройте заново.

Сейчас, выбрав в TextEdit меню пункт Quit (Выход), выйдите из программы TextEdit, и потом заново ее запустите. На этот раз вверху не будет причудливых меню форматирования. Можно приступать к созданию HTML-файлов.



Создание HTML-файла (Windows)

Или другой версии Windows.

Если вы читаете эту страницу, то вы, вероятно, пользователь Windows XP. Если нет, то можете пропустить несколько следующих страниц. Но мы не против, если вы все равно ознакомитесь с предложенной информацией.

Блокнот есть и в любой другой версии Windows.

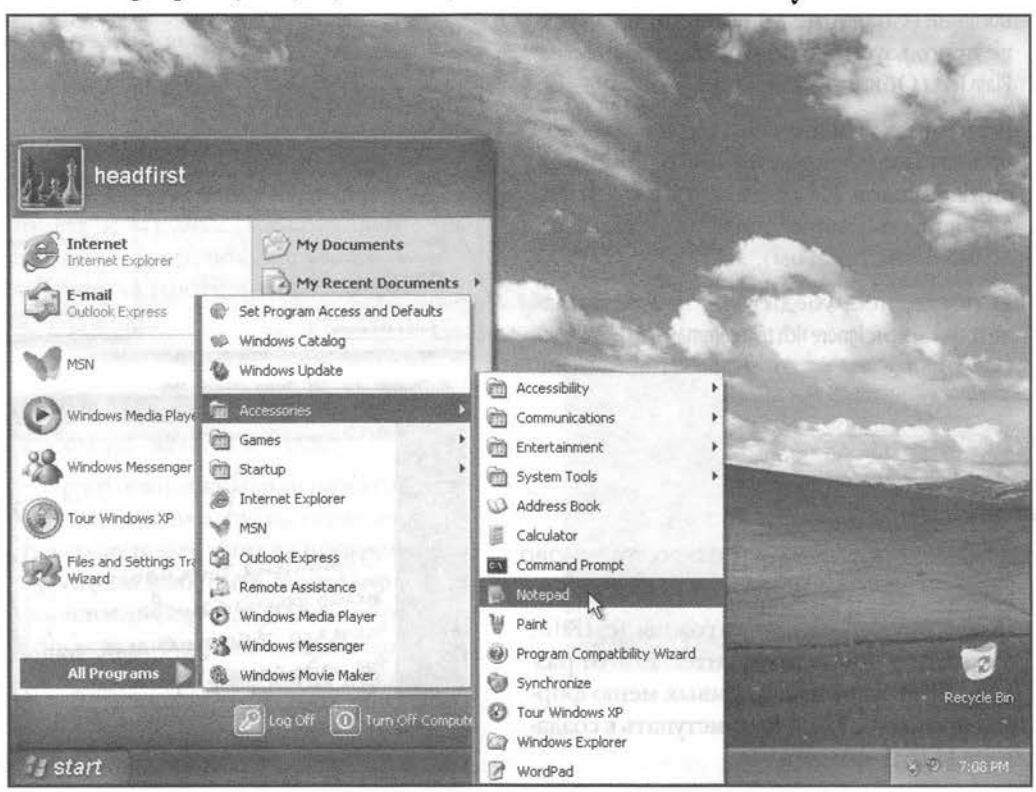
Для создания HTML-файлов в XP мы будем использовать программу Notepad (Блокнот). Она всегда есть в каждой новой версии Windows, и ею легко пользоваться. Если же у вас есть свой любимый редактор, который работает в Windows XP, то замечательно – можете использовать его. Только убедитесь, что в нем можно создавать файлы с расширением HTML.

Рассмотрим, как вы будете создавать свой первый HTML-файл при использовании программы Notepad (Блокнот).

Шаг первый

Откройте меню Start (Пуск) и найдите программу Notepad (Блокнот).

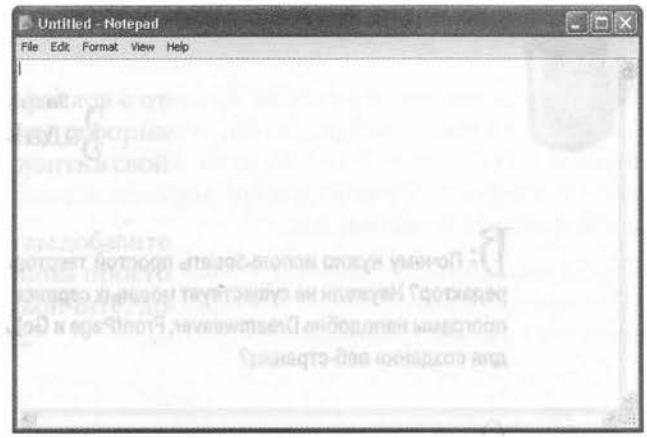
Программа Notepad (Блокнот) находится в стандартных программах. Самый простой способ туда попасть – открыть меню Start (Пуск), затем выбрать All Programs ▸ Accessories (Все программы ▸ Стандартные). Здесь в списке вы найдете программу Notepad (Блокнот).



Шаг второй

Откройте Notepad (Блокнот).

Выбрав в папке Accessories (Стандартные) программу Notepad (Блокнот), щелкните на ней кнопкой мыши. Откроется пустое окно, в котором вы можете начать набирать HTML-код.



Но рекомендуемый.

Шаг третий (необязательный)

Не прячьте расширения файлов.

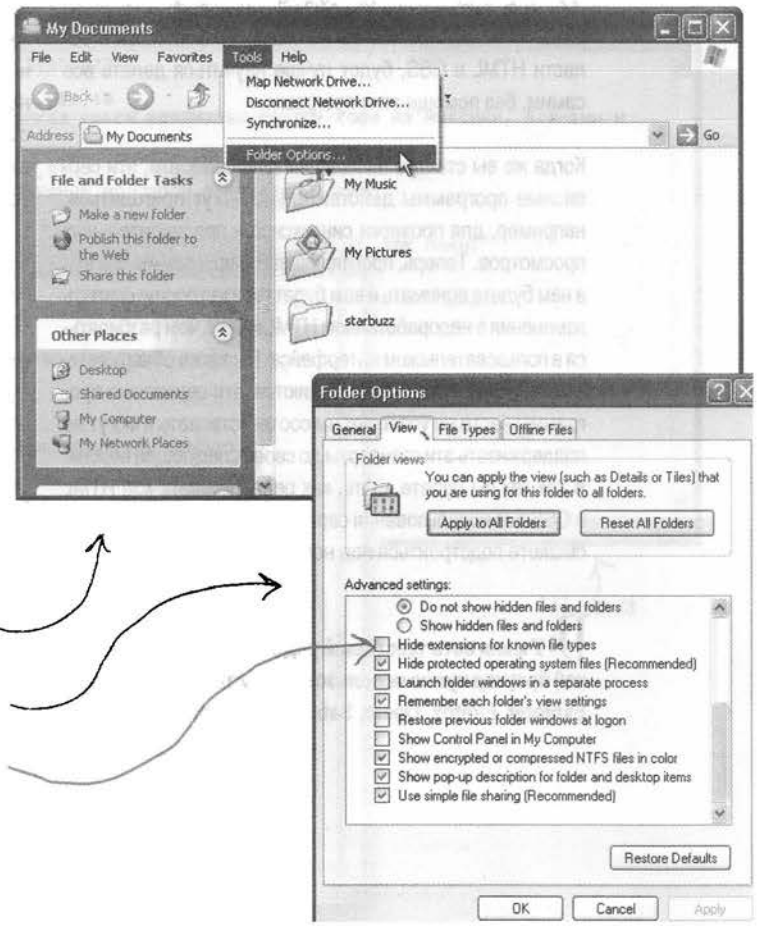
По умолчанию Explorer (Проводник) XP прячет зарегистрированные расширения файлов. Например, файл `Irule.html` будет показан как `Irule` без расширения `.html`.

Намного меньше путаницы будет, если вы укажете системе показывать расширения, поэтому мы расскажем вам, как это сделать.

Во-первых, в любом окне Explorer (Проводник) в меню Tools (Сервис) выберите пункт Folder Options (Свойства папки).

Затем на вкладке View (Вид) прокрутите список Advanced settings (Дополнительные параметры) вниз, пока не увидите флажок Hide extensions for known file types (Скрывать расширения для зарегистрированных типов файлов). *Снимите* его.

Все. Нажмите кнопку OK, чтобы применить настройки, и теперь в Explorer (Проводник) будут отображаться все типы файлов.



Часто Задаваемые Вопросы

В: Почему нужно использовать простой текстовый редактор? Неужели не существует мощных сервисных программ наподобие Dreamweaver, FrontPage и GoLive для создания веб-страниц?

О: Вы читаете эту книгу потому, что хотите понять действительно правильные технологии для создания веб-страниц, не так ли? Сервисные программы, о которых вы говорите, действительно хороши. Но они делают много работы за вас. Пока же вы не стали специалистом в области HTML и CSS, будет лучше научиться делать все самим, без помощи этих программ.

Когда же вы станете настоящим специалистом, эти сервисные программы действительно могут пригодиться, например, для проверки синтаксиса и предварительных просмотров. Теперь, проглядывая код программы, вы все в нем будете понимать и вам будет гораздо проще сделать изменения в недоработанном HTML и CSS, чем разбираться в пользовательском интерфейсе. Вы также обнаружите, что, поскольку стандарты меняются, эти сервисные программы часто не успевают им соответствовать и могут не поддерживать эти стандарты до своей следующей версии. А так как вы будете знать, как редактировать код HTML и CSS без использования сервисных программ, то всегда сможете подстроиться под новые стандарты.

В: У меня есть текстовый редактор, но я не знаю, какой браузер нужно использовать. Их так много: Internet Explorer, Firefox, Opera, Safari. Что делать?

О: Простой ответ: используйте тот браузер, который вам больше всего нравится. Дело в том, что все браузеры пытаются поддерживать HTML и CSS одинаково (убедитесь только, что используете новейшую версию браузера, для лучшей совместимости).

Развернутый ответ: в действительности есть небольшие различия в том, как разные браузеры обрабатывают веб-страницы. Если вы знаете, что пользователи будут открывать ваши веб-страницы в различных браузерах, тестируйте свой код во всех этих браузерах. Одни страницы будут выглядеть абсолютно одинаково, другие — нет. Чем более продвинутым в HTML и CSS вы будете становиться, тем больше эти различия будут иметь для вас значение. Мы еще рассмотрим некоторые из этих тонкостей в данной книге.

Если вы ищете хороший браузер, то попробуйте Mozilla Firefox. Он очень хорошо поддерживает HTML и CSS.

В: Я создаю эти файлы на своем компьютере. Как мне просмотреть их в Интернете?

О: Одно из преимуществ HTML состоит в том, что вы можете создавать и тестировать файлы на своем компьютере, а потом внедрять их в Сеть. На данный момент мы ставим задачу узнать, как создаются эти файлы и что у них внутри. К размещению их в Сети мы вернемся чуть позже.

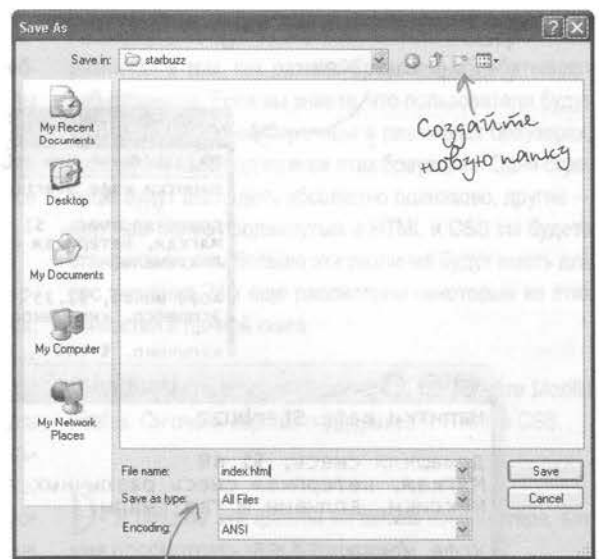
Сохранение работы

После того как вы перенесли информацию о напитках с салфетки в текстовый редактор, нужно сохранить работу в файле под названием `index.html`. Перед тем как это сделать, создайте папку под названием starbuzz, чтобы хранить в ней файлы для сайта кафе.

Чтобы выполнить это, выберите в меню File (Файл) пункт Save (Сохранить), в результате чего откроется окно Save As (Сохранить как). Рассмотрим, как это сделать.

Windows

① Создайте папку starbuzz для хранения всех файлов, имеющих отношение к кафе Starbuzz. Это делается с помощью кнопки New Folder (Новая папка).



Mac



Создайте новую папку

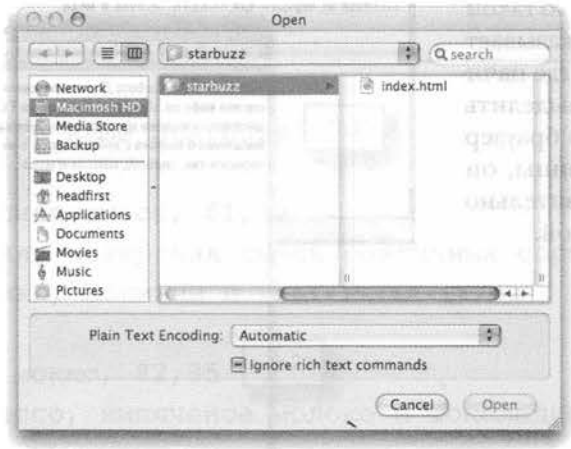
При использовании Windows вам также нужно выбрать тип файла All Files (Все файлы), иначе программа Notepad (Блокнот) добавит .txt к имени файла.

② Откройте только что созданную папку starbuzz, введите index.html в поле File name (Имя файла) и нажмите кнопку Save (Сохранить).

Открытие веб-страницы в браузере

Вы готовы открыть вашу первую веб-страницу? В своем любимом браузере выберите в меню File (Файл) пункт Open File (Открыть файл) (или Open (Открыть), если используете Windows XP и Internet Explorer) и найдите свой файл `index.html`. Выберите его и нажмите кнопку Open (Открыть).

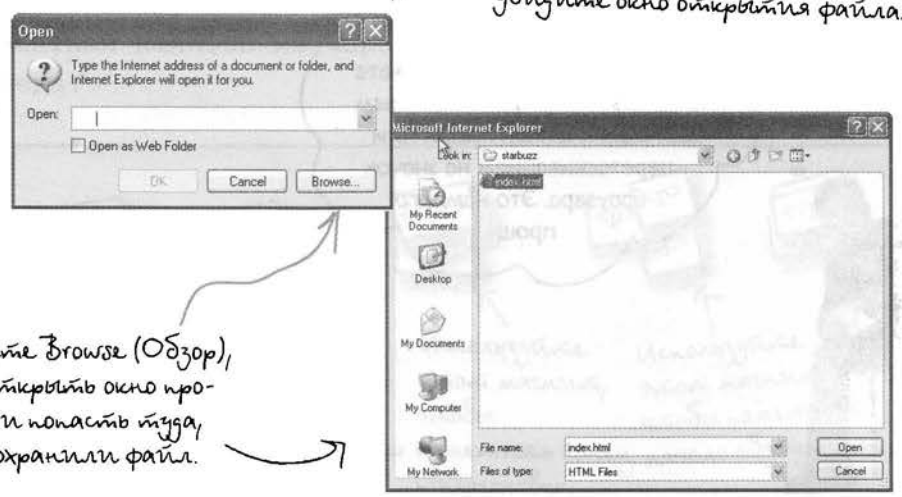
Mac



Найдите свой файл и выберите его, щелкнув кнопкой мыши на его значке и затем нажав кнопку Open (Открыть).

В Windows Internet Explorer это выполняется в два этапа. Сначала вы увидите окно открытия файла.

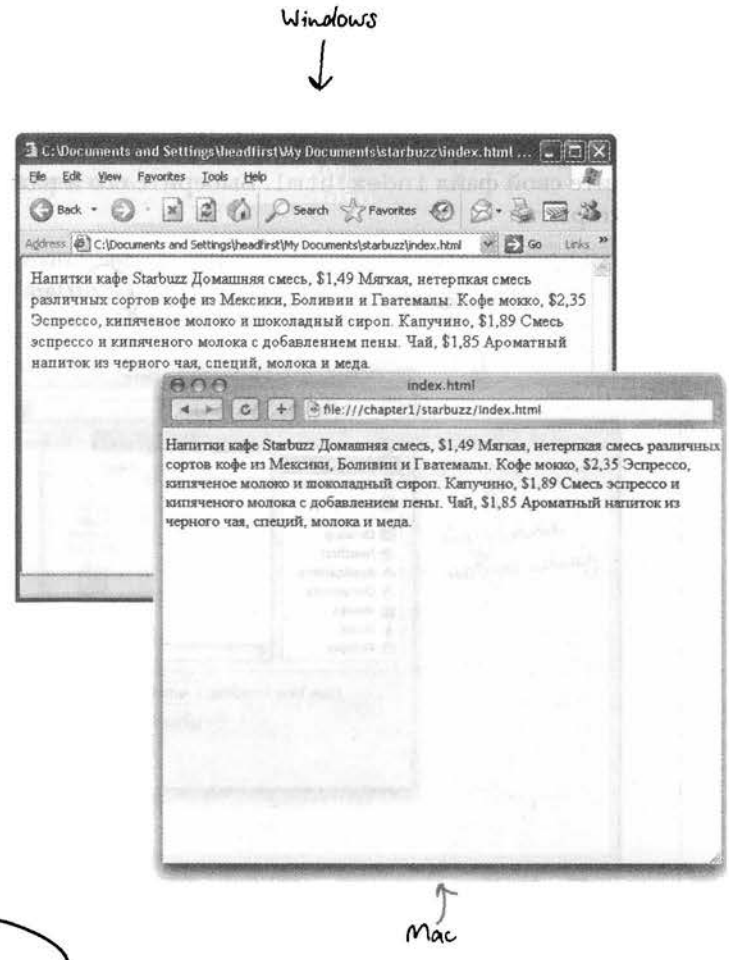
Windows



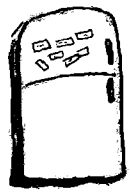
Нажмите Browse (Обзор), чтобы открыть окно просмотра и попасть туда, где вы сохранили файл.

Тестирование страницы

Ура! Вы загрузили страницу в браузер, хотя результат немного... ох... неудовлетворительный. Но это случилось потому, что мы забежали вперед, чтобы узнать механизм создания веб-страницы и ее просмотра в браузере. Ведь к настоящему времени вы напечатали только *содержание* веб-страницы. Теперь в действие вступает HTML. Он дает вам возможность рассказать браузеру о *структуре* страницы. Какой структуре? Как вы уже поняли, это такой способ разметки текста, который показывает браузеру, что является заголовком, где начинается новый абзац, какой текст выделить как подзаголовок и т. д. Как только браузер немного узнает о структуре страницы, он сможет отобразить ее более выразительно и страница станет более читабельной.





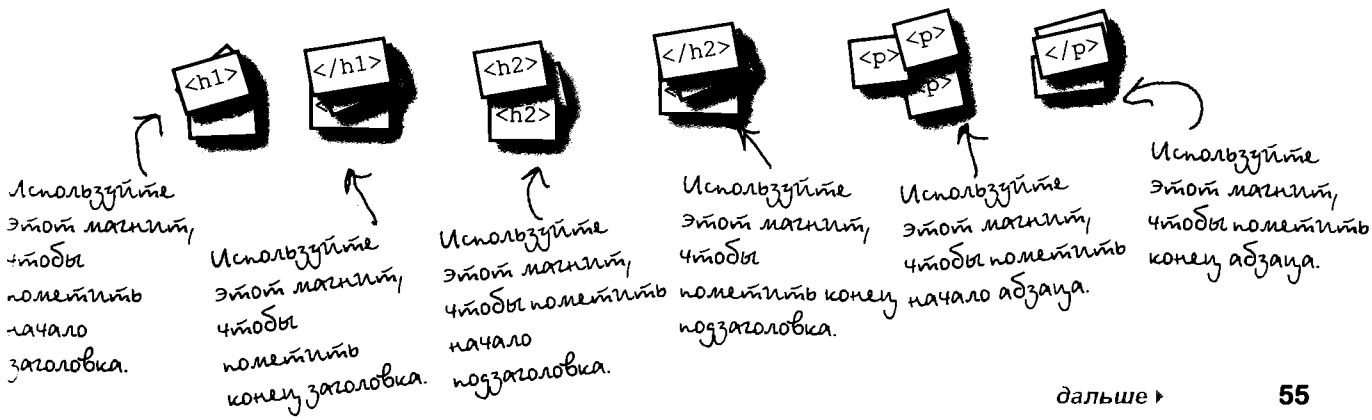


Магниты для разметки

Итак, давайте добавим структуру...

Ваша задача — структурировать текст с салфетки Starbuzz. Используйте магниты для холодильника (ищите их внизу этой страницы) для разметки текста таким образом, чтобы показать, какая его часть является заголовком, какая — подзаголовком и где начинается новый абзац. Кое-что мы сделали сами, чтобы вам было немного проще начать. Учтите, что для этой работы вам не понадобятся все магниты. Некоторые из них останутся незадействованными.

```
<h1>Напитки кафе Starbuzz</h1>

    Домашняя смесь, $1,49
    Мягкая, нетерпкая смесь различных сортов кофе из
    Мексики, Боливии и Гватемалы.

<h2>Кофе мокко, $2,35</h2>
<p>Эспрессо, кипяченое молоко и шоколадный сироп.</p>

    Капучино, $1,89
    Смесь эспрессо и кипяченого молока с добавлением пены.

    Чай, $1,85
    Ароматный напиток из черного чая, специй, молока
    и меда.
```





Поздравляем!
Вы только что написали свой первый HTML-код.

Возможно, это напоминало наклеивание магнитов на холодильник, но на самом деле вы делали разметку текста с помощью HTML. Только, как вы поняли, мы говорили о магнитах, а имели в виду *теги*.

Посмотрите на разметку, приведенную ниже, и сравните ее со своими магнитами с предыдущей страницы.

Теги <h1> и </h1> используются для выделения заголовка. Весь текст внутри этих тегов — заголовок.

```
<h1>Напитки кафе Starbuzz</h1>

<h2>Домашняя смесь, $1,49</h2>
<p>Мягкая, нетерпкая смесь различных
сортов кофе из Мексики, Боливии и Гватемалы.</p>

<h2>Кофе мокко, $2,35</h2>
<p>Эспрессо, кипяченое молоко и шоколадный сироп.</p>

<h2>Капучино, $1,89</h2>
<p>Смесь эспрессо и кипяченого молока
с добавлением пены.</p>

<h2>Чай, $1,85</h2>
<p>Ароматный напиток из черного чая,
специй, молока и меда.</p>
```

Теги <h2> и </h2> окружают подзаголовок. Надо понимать, что <h2> — это подзаголовок заголовка <h1>.

Теги и окружают часть текста, которая является отдельным абзацем. Это может быть одно или несколько предложений.

Заметьте, что не обязательно указывать теги в одной строке. Вы можете вставлять между ними столько текста, сколько хотите.

Мы все еще здесь?

У вас есть HTML-файл с разметкой. Значит ли это, что мы имеем готовую веб-страницу? Почти. Вы уже видели теги **<html>**, **<head>**, **<title>** и **<body>**, и мы просто должны их применить, чтобы создать первую HTML-страницу...

Во-первых, окружите свой код тегами <html> и </html>. Это позволит сказать браузеру, что в файле содержится HTML-код.

Затем добавьте теги <head> и </head>. Этот раздел (назовем его «шапкой») содержит описание вашей веб-страницы. Пока думайте про это так: «шапка» позволяет рассказать браузеру о веб-странице.

Продолжайте дальше и поместите название внутрь «шапки». Заголовок обычно появляется вверху окна браузера.

```
<html>

<head>
      <title>Кафе Starbuzz</title>
</head>

<body>
      <h1>Напитки кафе Starbuzz</h1>
      <h2>Домашняя смесь, $1,49</h2>
      <p>Мягкая, нетерпкая смесь различных сортов кофе из
      Мексики, Боливии и Гватемалы.</p>

      <h2>Кофе мокко, $2,35</h2>
      <p>Эспрессо, кипяченое молоко и шоколадный сироп.</p>

      <h2>Капучино, $1,89</h2>
      <p>Смесь эспрессо и кипяченого молока с добавлением
      пены.</p>
      <h2>Чай, $1,85</h2>
      <p>Ароматный напиток из черного чая, специй, молока
      и меда.</p>
</body>

</html>
```

«Шапка» состоит из тегов <head> и </head> и всего, что находится между ними.

Основная часть страницы состоит из тегов <body> и </body> и всего, что находится между ними.

Основная часть содержит всю информацию и структуру веб-страницы, части, которые вы видите в браузере.



дальше ›

Разделение тегов



Итак, вы уже немного знакомы с разметкой страницы, так давайте присмотримся и разберемся, как на самом деле работают теги...

Обычно вы ставите теги вокруг какой-то части текста. Например, мы используем теги, чтобы сказать браузеру, что часть текста «Напитки кафе Starbuzz» — это заголовок первого уровня.

Вот открывающий тег, который начинает заголовок.

Закрывающий тег завершает заголовок; в нашем примере тег </h1> завершает заголовок <h1>. Вы понимаете, что это закрывающий тег так как он идет после содержимого и перед «h1» указано «/». Все закрывающие теги содержат символ «/».

<h1> Напитки кафе Starbuzz </h1>

Тег состоит из имени элемента, взятого в угловые скобки < и >. Обычно вы ставите теги вокруг какой-то части текста. Например, мы используем теги, чтобы сказать браузеру, что часть текста «Напитки кафе Starbuzz» — это заголовок первого уровня.

Вся эта часть называется элементом. В нашем примере мы назовем его элементом <h1>. Элемент состоит из ограждающих тегов и содержимого.

Мы называем открывающий и его закрывающий тег соответственными.

Используйте пары тегов вокруг содержимого, чтобы описать браузеру структуру вашей страницы.

Помните:

Элемент = открывающий тег + содержимое +

+ закрывающий тег

Часто Задаваемые Вопросы

В: Итак, соответственные теги не обязательно должны быть расположены на одной строке?

О: Нет. Помните, что браузер не обращает внимания на символы табуляции, абзаца и пробелы, поэтому ваши теги могут начинаться и заканчиваться где угодно, на одной строке или даже на разных строках. Просто убедитесь, то первым идет открывающий тег, как <h2>, а вторым — закрывающий тег, как </h2>.

В: Почему в закрывающих тегах используется символ «/»?

О: Этот символ в закрывающих тегах помогает и вам, и браузеру понять, где заканчивается отдельный кусок структурированного кода. Если бы не это, то закрывающий тег выглядел бы точно так же, как открывающий, верно?

В: Я заметил, что в некоторых HTML-кодах открывающему тегу не ставится в соответствие закрывающий.

О: Вообще подразумевается, что теги *должны быть* соответственными. Браузеры достаточно хорошо выполняют работу по распознаванию того, что вы имели в виду, даже если написали HTML-код неправильно. Кроме того, в наши дни есть множество средств, помогающих написать код без ошибок. Существует множество сервисных программ, проверяющих ваш код перед тем, как вы поместите страницу в веб-сервер и весь мир сможет ее увидеть. Однако если вы будете нервничать, то никогда не сможете достичь в HTML совершенства, так что успокойтесь и просто возьмите себе в привычку всегда ставить в соответствие открывающим тегам закрывающие.

В: Хорошо, а что это за тег <img src="drinks.gif"> в примере про гостевую? Вы забыли закрывающий тег?

О: Класс, тонкое замечание! Существуют такие элементы, которые используют сокращенную форму записи только с одним тегом. Пока закиньте это в дальний уголок своего сознания, а мы вернемся к этому в следующих главах.

В: Элемент — это открывающий тег + + содержимое + закрывающий тег, а может ли один тег находиться внутри другого, как «шапка» и основная часть находятся внутри тега <html>?

О: Да, HTML-теги часто «вложены» таким образом. Если подумать, то для HTML-страниц естественно иметь основную часть, состоящую из абзацев и других элементов. Поэтому одни HTML-элементы содержат другие внутри своих тегов. Мы подробно разберем это в следующих главах, а пока просто обратите внимание, как элементы на веб-странице связаны между собой.

Мозговой штурм

Теги могут быть по-разному оформлены, а не только так, как вы видели до сих пор. Вот тег абзаца с небольшим дополнением в нем. Как вы думаете, что оно делает?

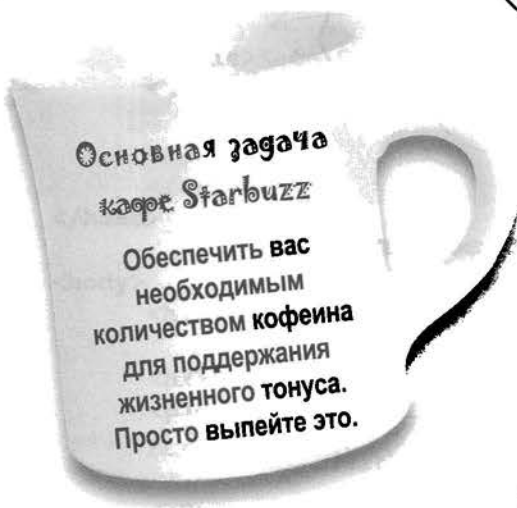
```
<p id="houseblend">Мягкая, нетерпкая
смесь различных сортов кофе из
Мексики, Боливии и Гватемалы.</p>
```

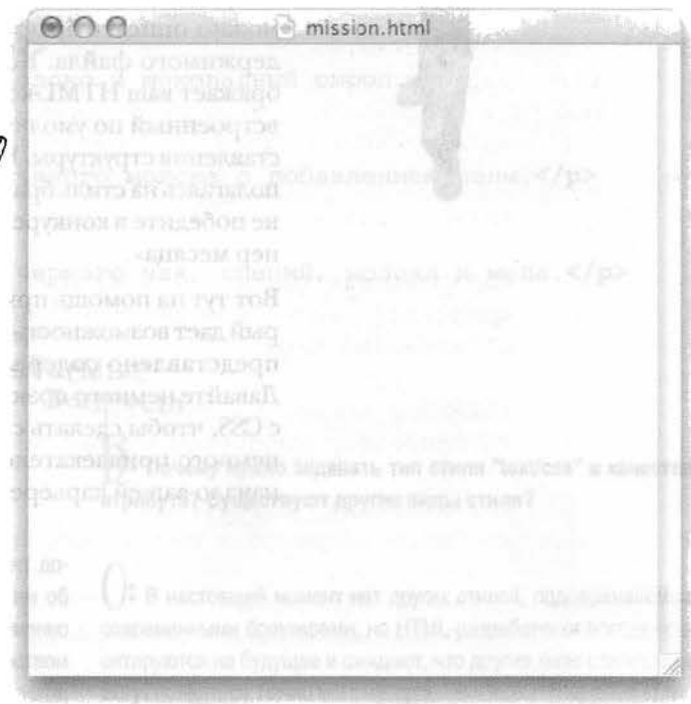


Упражнение

О, я забыл сказать, что нужно указать основную задачу нашего кафе на веб-странице. Вот постановка нашей задачи на примере одной чашки кофе. Поместите ее на отдельную страницу...







1. Напишите HTML-код для новой страницы `mission.html`.
2. Введите его, используя текстовый редактор, и сохраните страницу `mission.html` в той же папке, где и `index.html`.
3. Когда справитесь с этим, откройте файл `mission.html` в своем браузере.
4. Проверьте себя в конце этой главы, перед тем как читать дальше...



Итак, кажется, что-то стало проясняться. У вас уже есть основная страница и страница с постановкой задачи. Но не забывайте, что директор также хотел, чтобы сайт выглядел великолепно. Подумайте, может, нужно немного стилизовать страницы?

Хорошо. Мы уже разобрались со структурой, теперь настало время сконцентрироваться на дизайне страниц.

Вы уже знаете, что с помощью HTML можно описать браузеру структуру содержимого файла. Когда браузер отображает ваш HTML-код, он использует встроенный по умолчанию стиль представления структуры. Но очевидно, что, полагаясь на стиль браузера, вы никогда не победите в конкурсе «Лучший дизайнер месяца».

Вот тут на помощь приходит CSS, который дает возможность указать, как будет представлено содержимое страницы. Давайте немного поэкспериментируем с CSS, чтобы сделать страницу Starbuzz немного привлекательнее (и положим начало вашей карьере в этом деле).

CSS – это аббревиатура для Cascading Style Sheets (каскадные таблицы стилей). Мы остановимся на этом более подробно чуть позже, а пока просто знайте, что CSS дает вам возможность описать браузеру внешний вид веб-страницы.

Познакомьтесь с элементом <style>

Чтобы стилизовать страницу, нужно добавить на нее новый Э-Л-Е-М-Е-Н-Т (проговорите это вместе с нами) – элемент **<style>**. Давайте вернемся к странице Starbuzz и придадим ей определенный стиль. Вот, взгляните...

Элемент <style> находится внутри «шапки» вашей HTML-страницы

Как и у других элементов, у него есть открывающий тег <style> и закрывающий тег </style>...

...однако тег <style> также нуждается в атрибуте, который называется type и говорит браузеру, какой тип стиля использовать. Поскольку вы будете применять CSS, вам необходимо задать тип «text/css».

Здесь задаются стили страницы

```
<html>
  <head>
    <title>Кафе Starbuzz</title>
    <style type="text/css">

    </style>
  </head>

  <body>
    <h1>Напитки кафе Starbuzz</h1>

    <h2>Домашняя смесь, $1,49</h2>
    <p>Мягкая, нетерпкая смесь различных сортов кофе из Мексики,
    Боливии и Гватемалы.</p>

    <h2>Кофе мокко, $2,35</h2>
    <p>Эспрессо, кипяченое молоко и шоколадный сироп.</p>

    <h2>Капучино, $1,89</h2>
    <p>Смесь эспрессо и кипяченого молока с добавлением пены.</p>

    <h2>Чай, $1,85</h2>
    <p>Ароматный напиток из черного чая, специй, молока и меда.</p>
  </body>
</html>
```

Часто Задаваемые Вопросы

В: Элемент может иметь атрибут? Что это значит?

О: С помощью атрибутов вы можете снабжать элемент дополнительной информацией. Например, если мы говорим об элементе <style>, то атрибут позволяет уточнить, какой именно стиль имеется в виду. Вы еще познакомитесь со множеством других атрибутов для различных элементов; просто запомните, что они несут дополнительную информацию об элементе.

В: Почему нужно задавать тип стиля "text/css" в качестве атрибута? Существуют другие виды стиля?

О: В настоящий момент нет других стилей, поддерживаемых современными браузерами, но HTML-разработчики всегда ориентируются на будущее и ожидают, что другие типы стилей еще могут появиться. Лично мы с нетерпением ждем появления стиля <style type="50sKitsch"».

Придание определенного стиля странице Starbuzz

Теперь, когда вы знаете, что такое элемент **<style>**, все, что вам нужно сделать, – это добавить немного CSS в веб-страницу, чтобы придать ей более привлекательный вид. Ниже вы найдете CSS-код, который мы создали для вас. Каждый раз, увидев логотип Готовая выпечка, вы можете посмотреть результат работы HTML и CSS в чистом виде. Доверяйте нам. Как работает разметка, вы узнаете уже *после того*, как увидите результат.

Итак, взгляните на CSS-код, а затем добавьте его в файл **index.html**. Когда вы наберете код, сохраните файл.



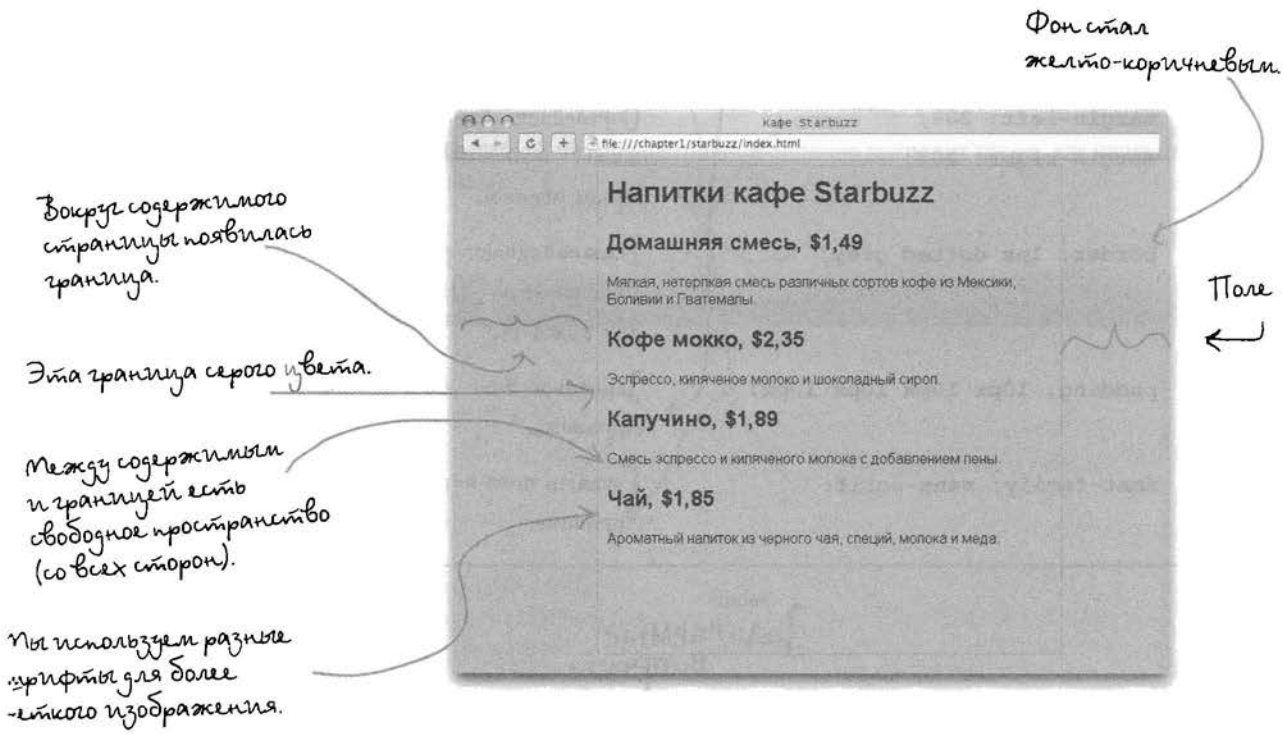
Готовый CSS-код

```
<html>
  <head>
    <title>Кафе Starbuzz</title>
    <style type="text/css">
      body {
        background-color: #d2b48c;
        margin-left: 20%;
        margin-right: 20%;
        border: 1px dotted gray;
        padding: 10px 10px 10px 10px;
        font-family: sans-serif;
      }
    </style>
  </head>

  <body>
    <h1>Напитки кафе Starbuzz</h1>

    <h2>Домашняя смесь, $1,49</h2>
    <p>Мягкая, нетерпкая смесь различных сортов кофе из Мексики, Боливии
    и Гватемалы.</p>

    <h2>Кофе мокко, $2,35</h2>
    <p>Эспрессо, кипяченое молоко и шоколадный сироп.</p>

    <h2>Капучино, $1,89</h2>
    <p>Смесь эспрессо и кипяченого молока с добавлением пены.</p>

    <h2>Чай, $1,85</h2>
    <p>Ароматный напиток из черного чая, специй, молока и меда.</p>
  </body>
</html>
```

CSS использует синтаксис, сильно отличающийся от синтаксиса HTML.

Путешествие в страну стиля

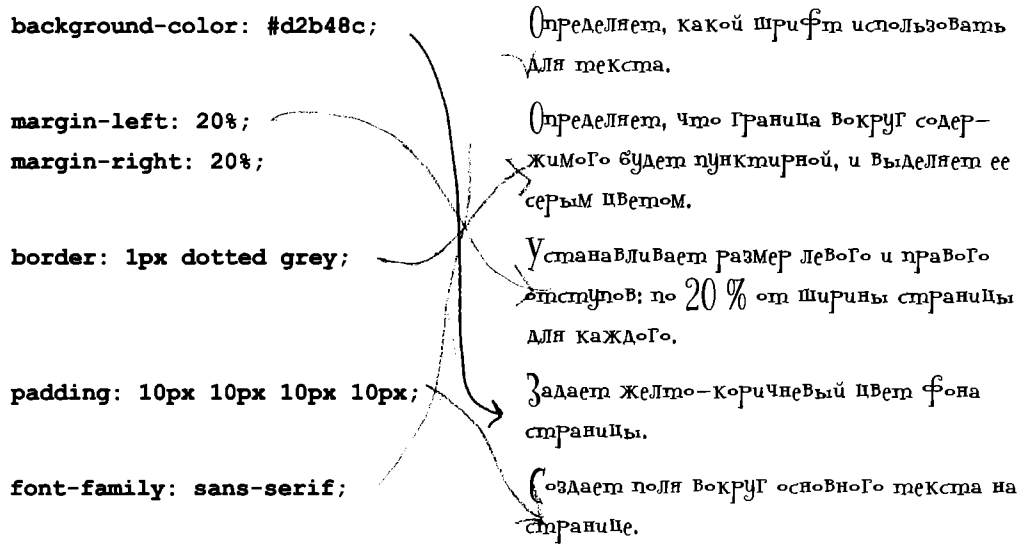
Настало время для следующего теста. Итак, обновите файл `index.html` еще раз. Теперь веб-страница Starbuzz будет выглядеть совсем иначе.





Кто и что делает?

Пока вы лишь мельком взглянули на CSS-код, но уже можете представить, на что он способен. Поставьте в соответствие каждой строке из определения стиля то, что она делает.



Часто Задаваемые Вопросы

В: CSS и HTML выглядят абсолютно по-разному. Зачем нужны два языка? Просто для того, чтобы мне нужно было больше выучить, так?

О: Вы совершенно правы в том, что HTML и CSS — абсолютно разные языки, но это потому, что они предназначены для абсолютно разных целей. Точно так же, как вы не станете использовать английский язык для подведения баланса на чековой книжке или язык математики для написания стихотворения, вы не будете использовать CSS для создания структуры или HTML для создания стиля, потому что это не то, для чего они были разработаны. Это значит, что вам нужно выучить оба языка. И в процессе обучения вы поймете, что каждый язык хорош именно в том, для чего предназначен. В результате легче выучить их оба, чтобы использовать каждый по назначению, а не применять один язык для обеих задач.

В: Мне кажется, что #d2b48c не похоже на цвет. Как это может означать желто-коричневый цвет?

О: Существует два способа задания цвета в CSS. Самый известный основан на применении шестнадцатеричного кода, которым и является значение #d2b48c. На самом деле это желто-коричневый цвет. Пока просто смиритесь с этим, а чуть позже мы подробно расскажем вам, как #d2b48c может быть цветом.

В: Что это за body перед CSS? Что оно означает?

О: Слово body в CSS означает, что весь код между { и } применяется к содержимому HTML-элемента <body>. Итак, когда вы устанавливаете шрифт sans-serif, вы говорите, что по умолчанию он будет использоваться для основной части веб-страницы.

В скором времени мы углубимся во множество других деталей работы CSS, поэтому продолжайте читать. Далее вы узнаете, что есть много особенностей применения этих правил, и, пользуясь этим, сможете создавать красивые и стильные страницы.



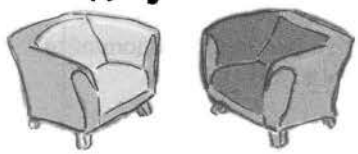
Упражнение

После того как вы немного стилизовали страницу Starbuzz `index.html`, продолжайте двигаться вперед и создайте такой же стиль для страницы `mission.html`.

1. Напишите HTML-код для страницы `mission.html`, а затем добавьте новый CSS-код.
2. Сохраните файл `mission.html`, чтобы включить в него новый код.
3. Когда справитесь с этим, обновите страницу `mission.html` в браузере.
4. Убедитесь, что ваша страница-задание выглядит так же, как наша (см. в конце главы).



Беседа у камина



Вечерний диалог: **HTML и CSS — про содержимое и стиль.**

HTML

Привет, CSS! Я рад, что ты здесь, потому что я хочу прекратить кое-какие сплетни про нас.

Многие люди думают, что мои теги говорят браузеру, как *отображать* содержимое. Но это же неправда! Я отвечаю только за *структуру*, а не за дизайн.

Ну, ты понимаешь, как некоторые люди могут войти в заблуждение. В конце концов, можно использовать HTML без CSS и все же получить приличного вида страницу.

Эй, я тоже достаточно силен. Придать содержимому структуру намного важнее, чем задать хороший внешний вид. Стиль так поверхностен; действительно важна структура текста.

О, какое самолюбие! Мне не стоило ожидать от тебя чего-то большего. Ты просто пытаешься придать важности стилю, о котором все время говоришь.

CSS

Правда? Какие сплетни?

Да, я тоже не хочу, чтобы люди думали, что ты выполняешь мою работу!

«Приличного» — это все-таки преувеличение, ты так не думаешь? Я имею в виду, что большинство браузеров отображают простой HTML-код достаточно скверно. Люди должны уяснить, как я силен и как легко могу придать их страницам отличный стиль.

Смотри на вещи трезво! Без меня веб-страницы были бы достаточно скучными и ничтожными. Кроме того, если убрать возможность стилизации страниц, то никто не станет воспринимать их всерьез. Они будут выглядеть грубо и непрофессионально.

HTML



CSS

Придать важности? Хороший дизайн и компоновка могут оказывать огромное влияние на читабельность и простоту использования страниц. И ты должен быть счастлив, что мои правила гибкого стиля позволяют дизайнерам экспериментировать с твоими элементами, не нарушая структуру.

Верно. На самом деле мы абсолютно разные языки, что замечательно, потому что я не хочу, чтобы твои дизайнеры стиля портили мои структурные элементы.

Не волнуйся, мы живем в разных вселенных.

Да, для меня это становится очевидным, когда я смотрю на CSS. Это инопланетный язык.

Да как вообще HTML может называться языком! Никто никогда не видел ничего более неуклюжего, чем все эти действия с тегами!

Миллионы веб-разработчиков с тобой не согласятся. Мой синтаксис чистый и ясный и хорошо подходит для содержимого.

Ты только взгляни на CSS: он такой изящный и простой, никаких бестолковых угловых скобок <вокруг><всего>. <Посмотри><,><я><могу><разговаривать> <только> <как> <господин><HTML><,> <взгляни> <на> <меня><!>

Эй, глупый, ты слышал когда-нибудь о закрывающих тегах?

Просто заметь, что, куда бы ты ни пошел, я найду и окружу тебя тегами **<style>**. Удачи, беглец!

Ха! Я тебе покажу... Догадываешься как? Я могу убежать...



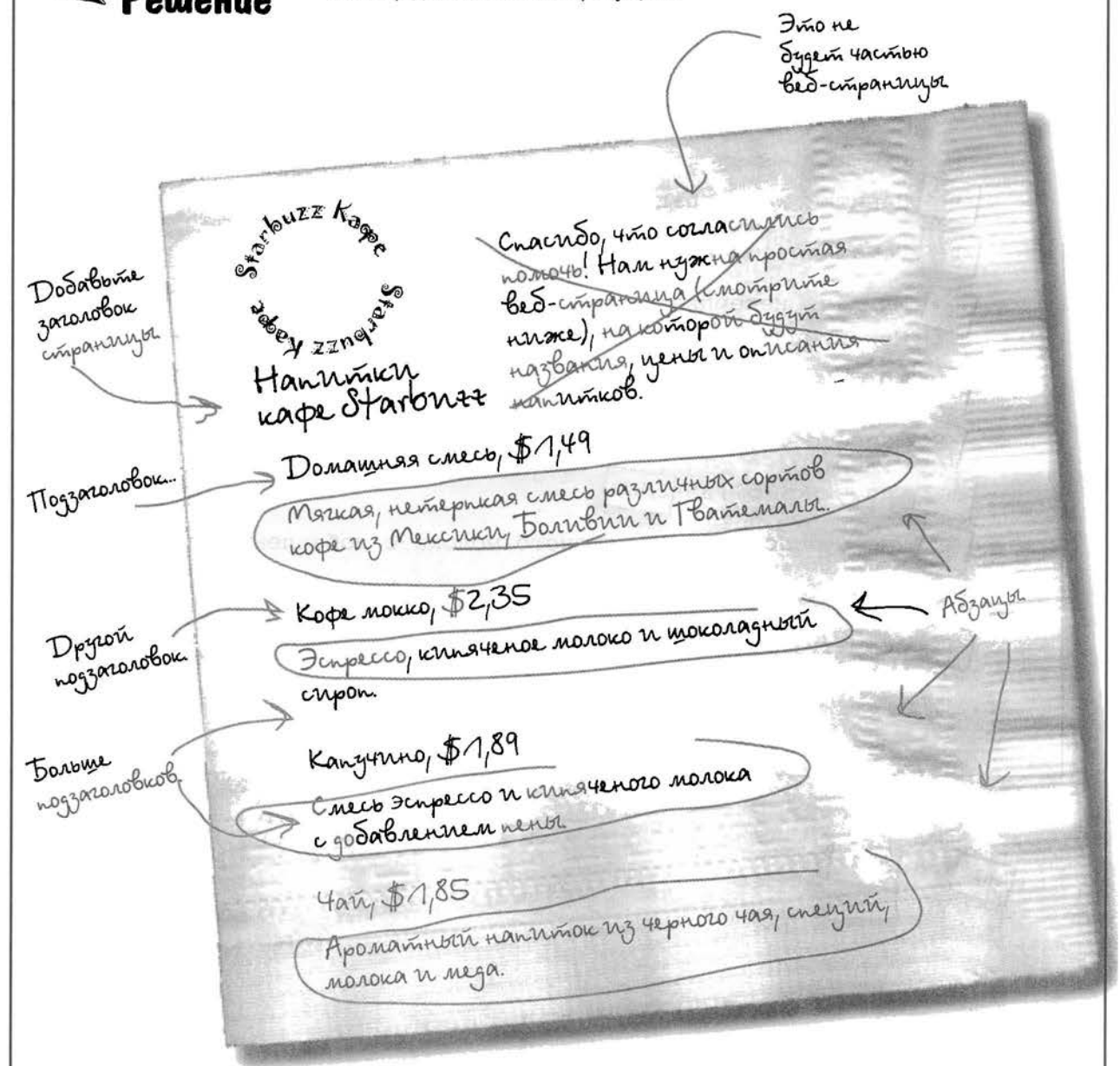
ПОВТОРИМ выученное

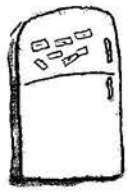
- HTML и CSS — языки, которые мы используем для создания веб-страниц.
- Веб-серверы обслуживают и хранят веб-страницы, которые созданы с помощью HTML и CSS. Браузеры извлекают страницы и визуализируют их содержимое, основанное на HTML и CSS.
- HTML — это аббревиатура HyperText Markup Language (язык гипертекстовой разметки). Этот язык используется для структуризации веб-страниц.
- CSS — это аббревиатура Cascading Style Sheets. Он используется для управления отображением вашего HTML.
- Используя HTML, мы помечаем содержимое тегами, чтобы придать ему структуру. Мы называем соответствующие теги и их содержимое элементом.
- Элемент состоит из трех частей: открывающего тега, содержимого и закрывающего тега. Но есть несколько элементов, например **<img>**, которые являются исключением из этого правила.
- Открывающие теги могут иметь атрибуты. Мы уже видели парочку: **type** и **align**.
- Чтобы закрывающие теги отличались от открывающих, в них после левой угловой скобки перед именем тега ставится символ «/».
- В ваших страницах всегда должен присутствовать элемент **<html>** с элементами **<head>** и **<body>** внутри.
- Информация про веб-страницу входит в элемент **<head>**.
- То, что вы добавляете внутрь элемента **<body>**, вы видите в браузере.
- Большинство служебных символов (символы табуляции, абзаца, пробелы) игнорируются браузером, но вы можете использовать их, чтобы сделать HTML-код более читабельным.
- В HTML-код веб-страницы можно добавить CSS-код, если поместить правила CSS внутрь элемента **<style>**. Элемент **<style>** всегда должен находиться внутри элемента **<head>**.
- С помощью CSS вы определяете свойства стиля элементов HTML.

Возьми в руку карандаш

Решение

Продолжайте дальше и сделайте разметку структуры на салфетке (карандашом), добавив то, что пропущено.





Магниты для разметки: решение

Вашим заданием было — структурировать текст на салфетке Starbuzz. Вы должны были использовать магниты для холодильника, расположенные внизу страницы, для разметки текста, чтобы указать, что является заголовком, что подзаголовком, а где начинается новый абзац. Как видите, некоторые магниты остались неиспользованными.

<h1>Напитки кафе Starbuzz</h1>

<h2>Домашняя смесь, $1,49</h2>

Мягкая, нетерпкая смесь различных сортов кофе из Мексики, Боливии и Гватемалы.

<h2>Кофе мокко, $2,35</h2>

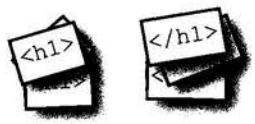
Эспрессо, кипяченое молоко и шоколадный сироп.

<h2>Капучино, $1,89</h2>

Смесь эспрессо и кипяченого молока с добавлением пены.

<h2>Чай, $1,85</h2>

Ароматный напиток из черного чая, специй, молока и меда.





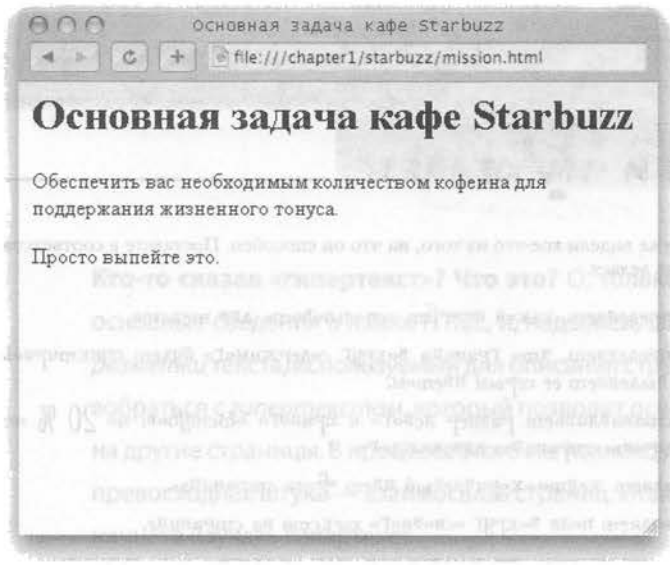
Решение упражнений



mission.html

```
<html>
  <head>
    <title>Основная задача кафе Starbuzz</title>
  </head>
  <body>
    <h1>Основная задача кафе Starbuzz</h1>
    <p>Обеспечить вас необходимым количеством
    кофеина для поддержания жизненного тонуса.</p>
    <p>Просто выпейте это.</p>
  </body>
</html>
```

Это HTML-код.



Это HTML, отображаемый в браузере



Решение упражнений

mission.html

```
<html>
  <head>
    <title>Основная задача кафе Starbuzz</title>
    <style type="text/css">
      body {
            background-color: #d2b48c;
            margin-left: 20%;
            margin-right: 20%;
            border: 1px dotted gray;
            padding: 10px 10px 10px 10px;
            font-family: sans-serif;
      }
    </style>
  </head>
  <body>
    <h1>Основная задача кафе Starbuzz</h1>
    <p>Обеспечить вас необходимым количеством кофеина для поддержания
жизненного тонуса</p>
    <p>Просто выпейте это.</p>
  </body>
</html>
```



Кто и что делает?

Хотя вы только один раз взглянули на CSS, вы уже видели кое-что из того, на что он способен. Поставьте в соответствие каждой строке с определением стиля то, что она делает.

```
background-color: #d2b48c;
margin-left: 20%;
margin-right: 20%;
border: 1px dotted grey;
padding: 10px 10px 10px 10px;
font-family: sans-serif;
```

Определяет, какой шрифт использовать для текста.

Определяет, что граница вокруг содержимого будет пунктирной, и выделяет ее серым цветом.

Устанавливает размер левого и правого отступов: по 20 % от ширины страницы для каждого.

Задает желто-коричневый цвет фона страницы.

Создает поля вокруг основного текста на странице.

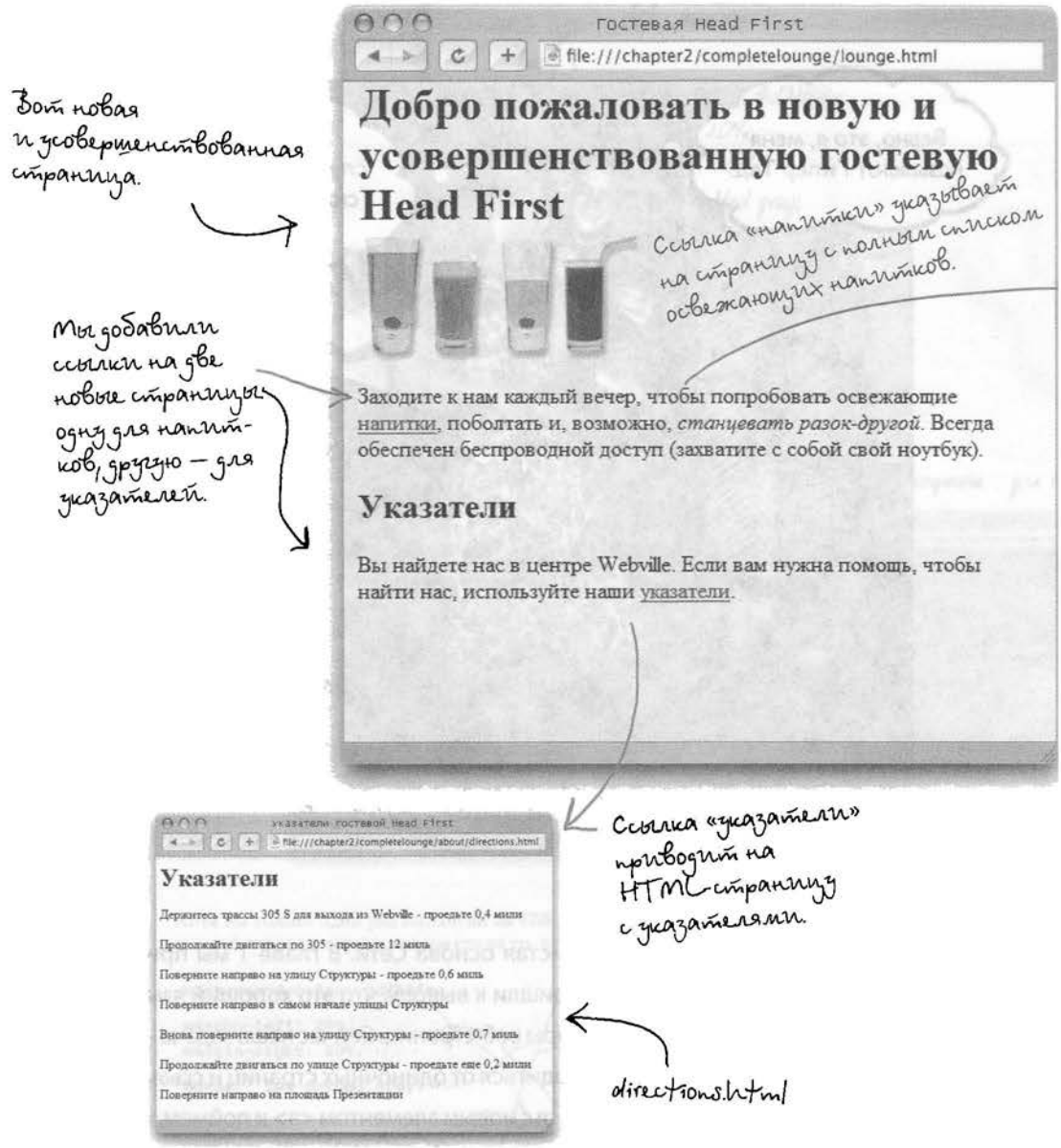
Знакомство с гипертекстом



Кто-то сказал «гипертекст»? Что это? О, только чистая основа Сети. В главе 1 мы привели основные сведения о языке HTML, и, надеемся, вы пришли к выводу, что это хороший *язык для разметки* текста, используемый для описания структуры веб-страниц. Сейчас наша задача — разобраться с *гипертекстом*, который позволит освободиться от одиночных страниц и ссылаться на другие страницы. В процессе этого мы познакомимся с новым элементом <a> и поймем, какая превосходная штука — взаимосвязь страниц. Итак, пристегните ремни безопасности, вы вот-вот начнете изучать гипертекст.

Новая и усовершенствованная гостевая

Помните гостевую Head First? Замечательный сайт, но не будет ли лучше, если покупатели смогут просмотреть список освежающих напитков? Кроме того, можно дать покупателям кое-какие указатели, чтобы они смогли нас найти.





elixir.html

Страница содержит список полезных освежающих напитков. Не стесняйтесь, отведайте один перед тем, как продолжить работу.

Создание новой и усовершенствованной гостевой в три этапа...

Переделаем оригинальную страницу гостевой Head First так, чтобы она ссылалась на две новые страницы.

1. Первый этап будет легким, потому что мы уже создали для вас файлы `directions.html` и `elixir.html`. Вы найдете их в исходных файлах для книги, которые доступны на сайте `http://www.headfirstlabs.com`.



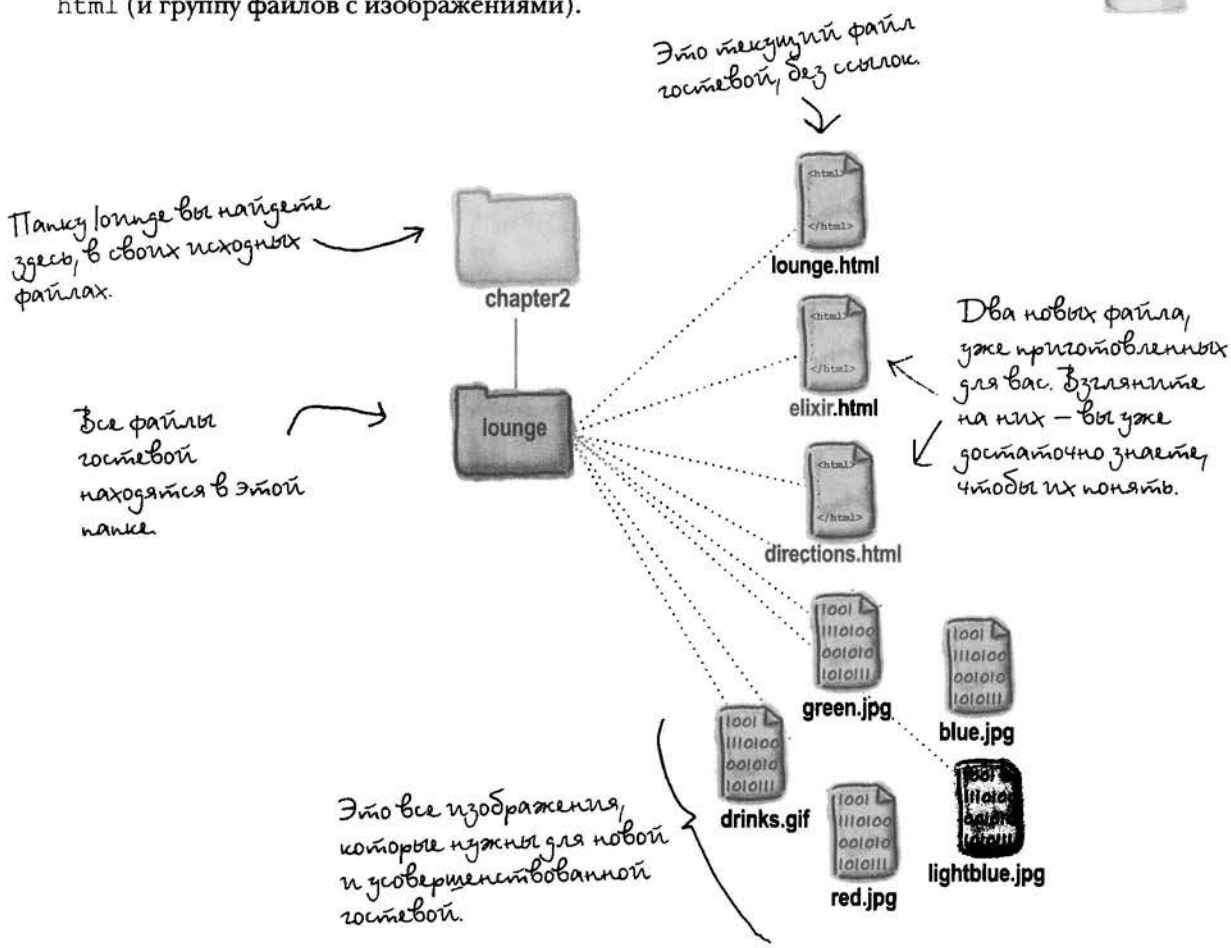
2. На втором этапе вам нужно отредактировать файл `lounge.html` и добавить в HTML-код ссылки на файлы `directions.html` и `elixir.html`.
3. И последнее — вы протестируете страницы и проверите работу ссылок. Когда вы закончите, мы сядем и посмотрим, как это работает.

Создание новой гостевой



1 Получение исходных файлов

Продолжите работу и возьмите исходные файлы на сайте `http://www.headfirstlabs.com`. Когда вы их скачаете, зайдите в папку `chapter2/lounge` и найдите файлы `lounge.html`, `elixir.html` и `directions.html` (и группу файлов с изображениями).



МОЗГОВОЙ ШТУРМ

Гостевая Head First увеличивается. Как вы думаете, хорошо ли для организации сайта держать все его файлы в одной папке? Что можно придумать вместо этого?

2 Редактирование страницы lounge.html

Откройте файл `lounge.html` в своем текстовом редакторе. Добавьте новый текст и HTML-код, выделенный ниже. После того как вы все сделаете, мы вернемся и посмотрим, как это работает на странице.

```
<html>
  <head>
    <title>Гостевая Head First</title>
  </head>
  <body>
    <h1>Добро пожаловать в новую и усовершенствованную гостевую Head
    First</h1>
    <img src="drinks.gif">
    <p>
      Заходите к нам каждый вечер, чтобы попробовать
      освежающие<a href="elixir.html">напитки</a>,
      поболтать и, возможно,
      <em>станцевать разок-другой</em>.
      Всегда обеспечен беспроводной доступ
      (захватите с собой свой ноутбук).
    </p>
    <h2>Указатели</h2>
    <p>
      Вы найдете нас в центре Webville.
      Если вам нужна помощь, чтобы найти нас, используйте
      наши<a href="directions.html">указатели</a>.
    </p>
  </body>
</html>
```

Давайте добавим «новую и усовершенствованную» в «шапку».

Здесь мы добавим HTML-код для ссылки на список с напитками.

Для создания ссылки мы используем элемент <a>. Как этот элемент работает, вы увидите через секундочку...

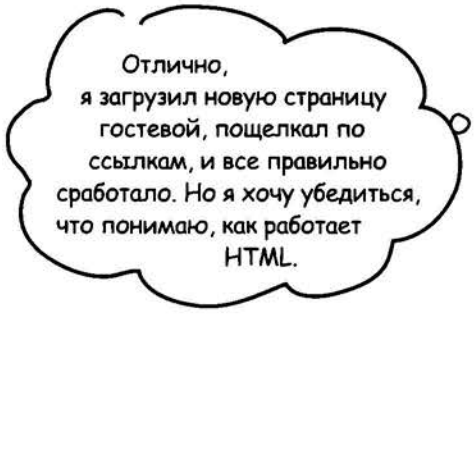
Мы должны добавить сюда какой-нибудь текст, направляющий покупателей в наше кафе.

Здесь мы добавим ссылку на указатели, вновь используя элемент <a>.

3 Сохранение страницы lounge.html и ее тестирование в браузере

Когда вы внесете изменения, сохраните файл `lounge.html` и откройте его в своем браузере. Попробуйте сделать следующее.

1. **Щелкните кнопкой мыши на ссылке «напитки», и появится новая страница со списком напитков.**
2. **Нажмите кнопку Назад в браузере — вновь должна будет открыться страница lounge.html.**
3. **Щелкните кнопкой мыши на ссылке «указатели», и откроется новая страница с указателями.**





Позади сцены

Что делали мы

1 Давайте подробно разберем процедуру создания HTML-ссылок. В первую очередь нам нужно поместить текст, которой мы сделаем ссылкой, в элемент <a>. Это делается таким образом:

```
<a>напитки</a>          <a>указатели</a>
```

Элемент <a> применяется для создания ссылки на другую страницу.

Содержимое элемента <a> выступает в качестве метки для ссылки. В браузере метка отображается в виде подчеркнутого текста, чтобы показать вам, что можно щелкнуть на нем кнопкой мыши.

2 Теперь, когда у нас есть метка для каждой ссылки, нужно добавить кое-какой HTML-код, чтобы сказать браузеру, на что ссылка указывает:

```
<a href="elixir.html">напитки</a>
```

Для этой ссылки браузер отобразит метку «напитки», после щелчка на которой пользователь попадет на страницу elixir.html.

Атрибут href задает назначение ссылки.

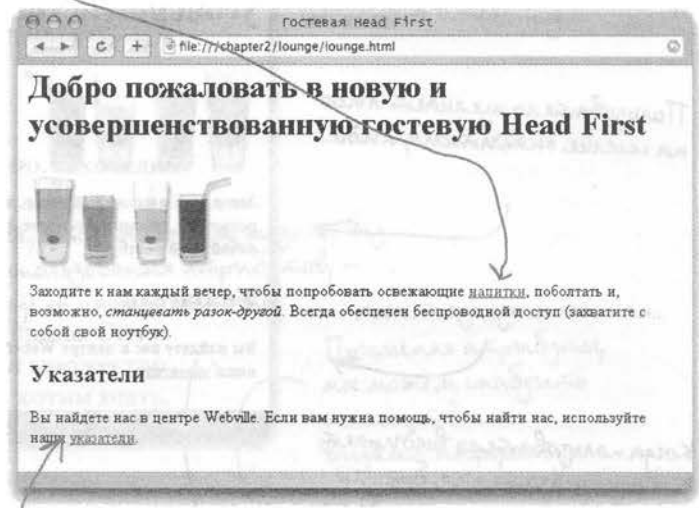
А для этой ссылки браузер отобразит метку «указатели», после щелчка на которой пользователь попадет на страницу directions.html.

```
<a href="directions.html">указатели</a>
```

Что делает браузер

❶ Во-первых, поскольку браузер формирует изображение страницы, то, сталкиваясь с элементом <a>, он выводит его содержимое как ссылку, на которой можно щелкнуть кнопкой мыши.

```
<a href="elixir.html">напитки</a>
```



И «напитки», и «указатели» находятся между открывающим и закрывающим тегами <a>, поэтому они становятся на веб-страницах ссылками, на которых можно щелкнуть кнопкой мыши.

```
<a href="directions.html">указатели</a>
```

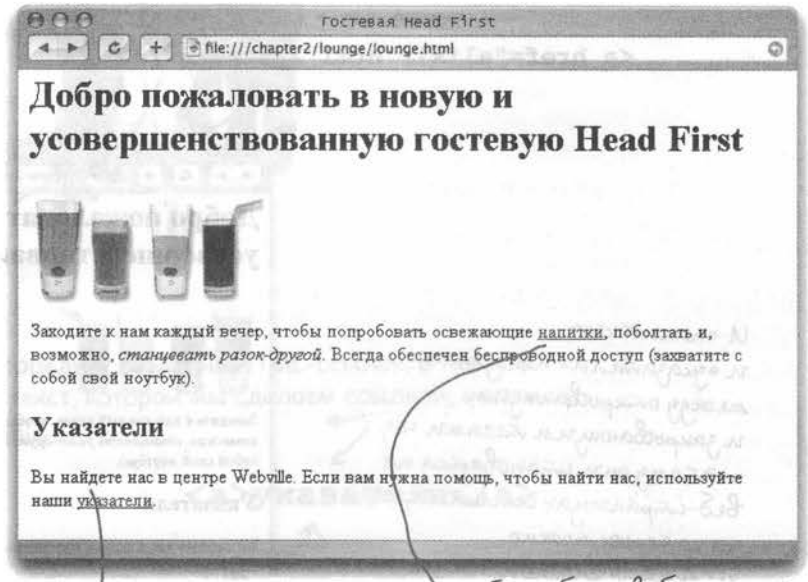
Используйте элемент <a> для создания гипертекстовых ссылок на другую страницу.

Содержимое элемента <a> можно активизировать щелчком кнопкой мыши на веб-странице.

Атрибут href указывает браузеру адрес назначения ссылки.

Позади сцены

2 Затем, когда пользователь щелкает кнопкой мыши на ссылке, браузер проверяет атрибут href, чтобы определить страницу, на которую указывает ссылка.



Пользователь щелкает либо на ссылке «напитки», либо...

...на ссылке «указатели».

Когда пользователь выбирает ссылку «указатели», браузер находит значение атрибута href — в данном случае это directions.html...

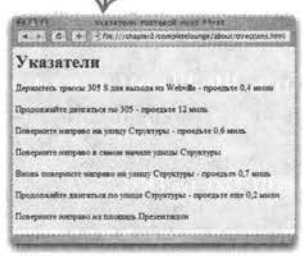
Если была выбрана ссылка «напитки», браузер находит значение elixir.html...

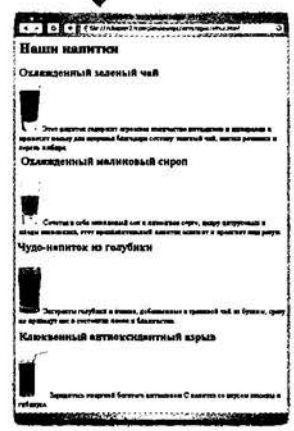
```
<a href="elixir.html">напитки</a>
```

...и отображает соответствующую страницу.

```
<a href="directions.html">указатели</a>
```

...и загружает страницу directions.html.





Что такое атрибуты



С помощью атрибутов можно указать дополнительную информацию об элементе. Несмотря на то что мы еще подробно не разбирали атрибуты, вы уже видели несколько их примеров:

```
<style type="text/css">
<a href="irule.html">
<img src="sweetphoto.gif">
```

Атрибут type указывает на то, какой язык стиля мы используем в данном случае CSS.

Атрибут href указывает на адрес назначения ссылки.

Атрибут src определяет имя файла с изображением, которое задается в теге img.

Давайте на примере посмотрим, как работают атрибуты, чтобы вы лучше разобрались с этой темой.

Если бы был элемент <car>...

Если бы был элемент **<car>**, то вы, естественно, захотели бы написать что-то вроде:

```
<car>Моя красная Мини</car>
```

Все, что мы можем указать без использования атрибутов, — модель нашей машины.

Но этот элемент **<car>** задает только марку машины. Он не позволяет узнать год выпуска, точную марку и множество других подробностей, которые мы, возможно, хотим знать. Итак, если бы **<car>** действительно был элементом, мы бы примерно так использовали атрибуты:

Применяя атрибуты, мы можем снабдить элемент всеми видами информации.

```
<car make="BMW" model="Mini Cooper" convertible="нет">Моя красная Мини</car>
```

Не правда ли, лучше? Теперь эта запись говорит нам намного больше.



ПРАВИЛА БЕЗОПАСНОСТИ

Атрибуты всегда пишутся одинаково: сначала идет имя атрибута, затем знак равенства, затем значение атрибута, взятое в двойные кавычки. Возможно, в Сети вы увидите небрежно написанные HTML-страницы, в которых пропущены эти двойные кавычки, но не ленитесь ставить их сами. Ваша небрежность может доставить вам массу проблем (далее в книге вы убедитесь в этом сами).

Делайте так (правильный вариант)

```
<a href="top10.html">Отличные фильмы</a>
```

имя атрибута — знак равенства — двойные кавычки — значение атрибута — двойные кавычки

Не делайте так (неправильный вариант)

```
<a href=top10.html>Отличные фильмы</a>
```

ОШИБКА — значение атрибута не взято в двойные кавычки.



Часто Задаваемые Вопросы

В: Могу ли я просто выдумать новый атрибут для элемента HTML?

О: Нет, потому что браузеры распознают только предопределенный набор атрибутов для каждого элемента. Если же вы просто выдумали атрибут, то браузер не будет знать, что с ним делать. Принято говорить, что браузер «поддерживает» элемент или атрибут, если распознает их. Вы можете использовать только те атрибуты, которые точно поддерживаются браузерами.

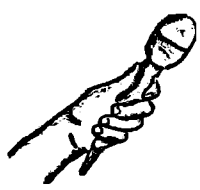
В: А кто решает, что поддерживается, а что — нет?

О: Существуют комитеты по стандартам, которые занимаются элементами и атрибутами HTML. Эти комитеты состоят из людей, которые отдают огромное количество своего времени и энергии, чтобы убедиться, что существует общий для всех версий HTML план, который все компании могут использовать для реализации своих браузеров.

В: Как мне узнать, что элементы и атрибуты поддерживаются? И можно ли все атрибуты применить к одному элементу?

О: Для каждого элемента есть свой собственный набор атрибутов. Иными словами, для элемента используются только те атрибуты, которые имеют для него смысл и поддерживаются им.

Далее в книге мы расскажем, какие именно атрибуты какими элементами поддерживаются. Чтобы освежить память, после прочтения этой книги можно пользоваться множеством справочников, например изданием *HTML & XHTML: The Definitive Guide* (O'Reilly).



УЯЗВИМОСТЬ АТРИБУТОВ

Интервью, взятое на этой неделе.
Признания атрибута href

Head First: Добро пожаловать, Href. Очень приятно брать интервью у такого важного атрибута, как вы.

Href: Спасибо. Я очень рад, что освободился от создания ссылок и нахожусь здесь; эта работа может изнурить любой атрибут. Догадываетесь, кто говорит браузеру, куда идти дальше, каждый раз, как только кто-то щелкнет на ссылке? Это я.

Head First: Мы рады, что вы выделили время для нас в своем очень плотном графике работы. Может быть, расскажете нам все с самого начала? Что значит быть атрибутом?

Href: Конечно. Атрибут используется, чтобы видоизменить элемент. Нет ничего проще, чем окружить кусок текста парочкой тегов **<a>**. Например, для выражения «Запишись сейчас!» это будет выглядеть так: **<a>Запишись сейчас!**. Но без меня, атрибута `href`, вы не сможете указать элементу **<a>** место назначения ссылки.

Head First: Пока понятно...

Href: ...благодаря атрибуту вы можете снабдить элемент дополнительной информацией. В моем случае это информация о том, куда указывает ссылка: ****`Запишись сейчас!`****. Это означает, что элемент **<a>**, отображаемый на странице меткой «Запишись сейчас!», ссылается на страницу `signup.html`. В настоящее время в мире есть множество других атрибутов, но только меня используют с элементом **<a>**, чтобы показать, куда он ссылается

Head First: Здорово. Но сейчас мы должны спросить и надеемся, что это вас не оскорбит: что означает ваше имя?

Href: Это имя старого интернет-семейства. Оно означает «гипертекстовая ссылка», но друзья сокращенно называют меня просто Href.

Head First: А что это?

Href: Гипертекстовая ссылка — это просто другое название ресурса, который находится в Интернете или в вашем компьютере. Обычно ресурс — это другая страница, но я также могу указывать на аудио, видео... и на все остальное.

Head First: Интересно. Наши читатели до сих пор успели познакомиться только со ссылками на их собственные страницы. А как ссылаться на другие страницы и ресурсы в Сети?

Href: Эй! Я должен снова возвращаться к работе, вся Сеть не может без меня нормально работать. Более того, разве это не ваша работа — учить этому читателей?

Head First: Хорошо, хорошо, да, мы вернемся к этому чуть позже. Спасибо, что уделили нам время, Href.



Упражнение

Вы создали ссылки, чтобы попасть со страницы `lounge.html` на страницы `elixir.html` и `directions.html`. Сейчас мы поработаем над созданием обратных ссылок. Ниже приведен HTML-код для файла `elixir.html`. Добавьте ссылку с меткой Назад в гостевую, которая будет указывать назад на страницу `lounge.html`.

```
<html>
  <head>
    <title>Напитки гостевой Head First</title>
  </head>
  <body>
    <h1>Наши напитки</h1>

    <h2>Охлажденный зеленый чай</h2>
    <p>
                <img src="green.jpg">
                Этот напиток содержит огромное количество витаминов и
                минералов и приносит пользу для здоровья благодаря
                составу: зеленый чай, цветки ромашки и корень имбиря.
    </p>
    <h2>Охлажденный малиновый сироп</h2>
    <p>
                <img src="lightblue.jpg">
                Сочетая в себе малиновый сок и лимонное сорго, цедру
                цитрусовых и плоды шиповника, этот прохладительный
                напиток освежит и прояснит ваш разум.
    </p>
    <h2>Чудо-напиток из голубики</h2>
    <p>
                <img src="blue.jpg">
                Экстракты голубики и вишни, добавленные в травяной чай
                из бузины, сразу же приведут вас в состояние покоя
                и блаженства.
    </p>
    <h2>Клюквенный антиоксидантный взрыв</h2>
    <p>
                <img src="red.jpg">
                Зарядитесь энергией богатого витамином С напитка
                со вкусом клюквы и гибикуса.
    </p>

Ваш новый
HTML-код
будет здесь.

  </body>
</html>
```

Когда справитесь с этим, сделайте то же самое с файлом directions.html.



Нам нужны несколько элементов <a> для компоновки и перекомпоновки. Надеемся, вам помогут новые знания про элемент <a>. В каждой строке следующей таблицы вы найдете некоторые сочетания метки, адреса и полного элемента **<a>**. Заполните пропущенные ячейки. Ячейку в первом ряду мы уже заполнили за вас.

Метка	Адресат	Что вы пишете в HTML
Горячо или нет?	hot.html	`<a href="hot.html">Горячо или нет?</a>`
Резюме	cv.html	
	candy.html	`<a href="_____">Вкусная конфета</a>`
Посмотри на мою Мини	mini-cooper.html	
Давайте поиграем		`<a href="millionaire.html">_____</a>`

Часто Задаваемые Вопросы

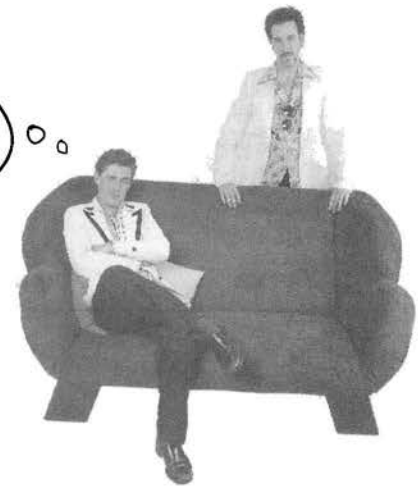
В: Я видел много страниц, в которых можно щелкнуть кнопкой мыши на изображении, а не на тексте. Можно ли использовать для этого элемент <a>?

О: Да, если вы поместите элемент <img> между тегами <a>, то ваш рисунок станет такой же ссылкой, как и текст. Мы не будем подробно на этом останавливаться, но рисунки отлично работают в качестве ссылок.

В: Итак, я могу поместить что угодно между тегами <a>, и на этом можно будет щелкнуть кнопкой мыши? Скажем, целый абзац?

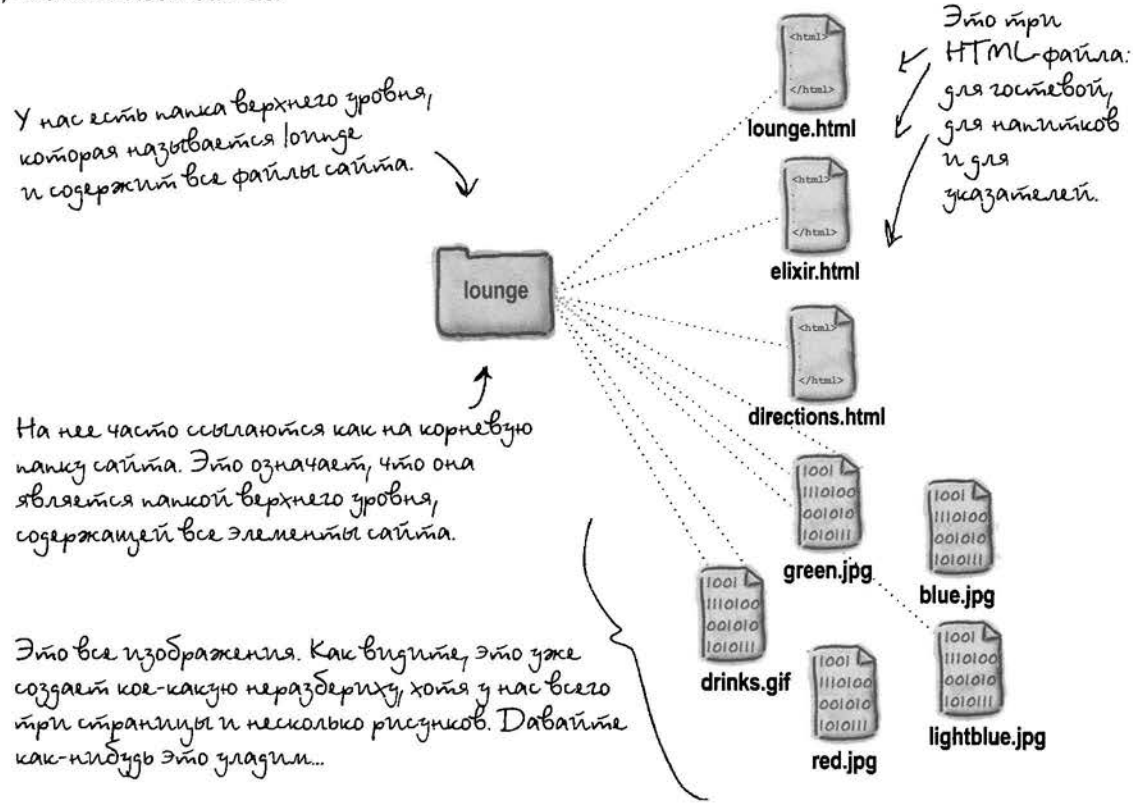
О: Тише, тише. Не так быстро. Не любой элемент может быть помещен внутрь элемента <a>. Вообще, вы будете использовать только текст или рисунки (или и то и другое) внутри элемента <a>. Какие теги могут быть использованы внутри других — это совсем другая тема, но не волнуйтесь, мы очень скоро к ней вернемся.

Ваша работа над гостевой Head First уже сполна окупилась. Благодаря этим притягательным напиткам и указателям многие люди часто находят наш сайт и посещают его в Сети. Теперь в наших планах — всеми возможными способами расширить online-содержимое гостевой.



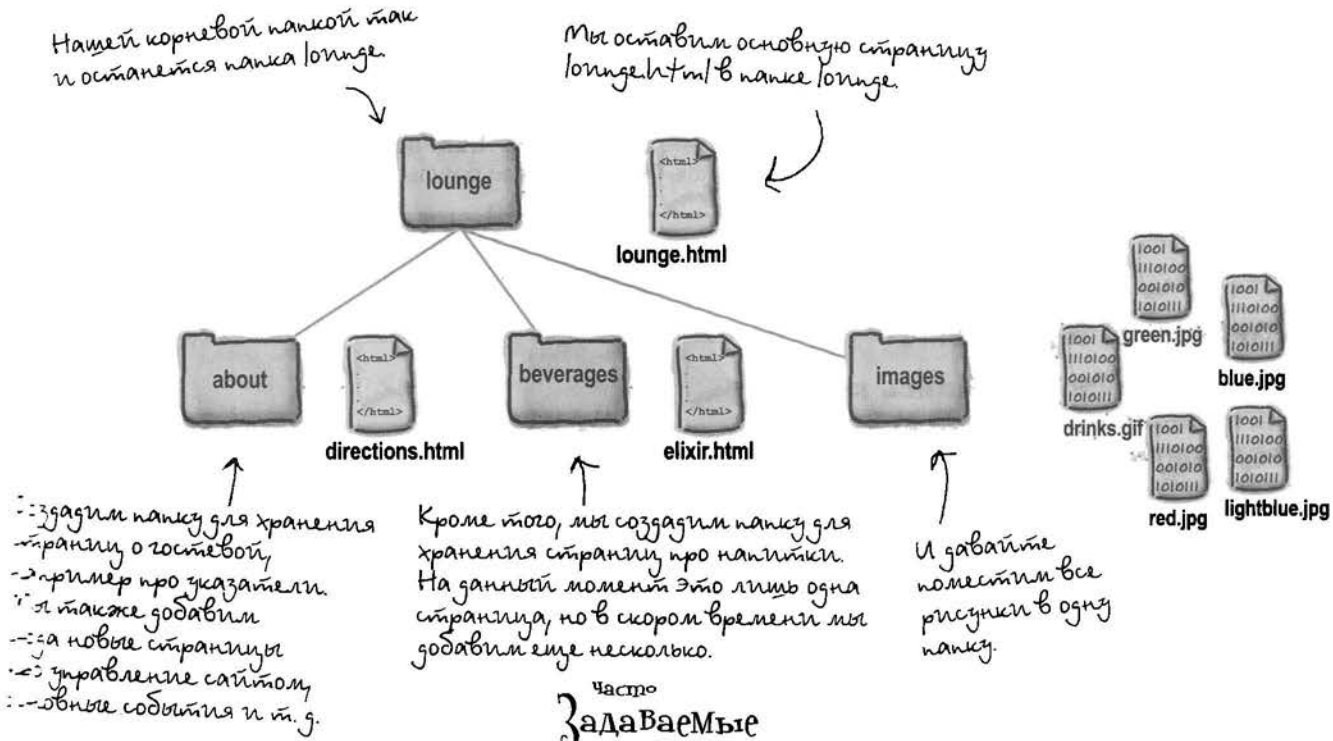
Наведение порядка

Перед тем как продолжить создавать HTML-страницы, нужно привести все в порядок. До сих пор мы складывали все файлы и изображения в одну папку. Далее вы узнаете, что даже небольшими сайтами намного проще управлять, если разложить веб-страницы, рисунки и другие ресурсы в различные папки. Рассмотрим, что мы имеем сейчас:



Приведение гостевой страницы в порядок

Создадим для гостевой какую-нибудь выразительную структуру. Существует множество способов привести сайт в порядок, но мы начнем с простого и создадим парочку папок для страниц, а затем разместим все изображения в одном месте.



Часто Задаваемые Вопросы

В: Раз уж мы создали папку для рисунков, почему бы нам не завести другую под названием html и не сложить в нее все HTML-файлы?

О: Мы можем это сделать. Не существует единственно правильного способа приведения файлов в систему, но мы ведь хотим систематизировать их наиболее удобным способом для себя и для пользователей сайта. Как и в большинстве дизайнерских решений, мы хотим выбрать достаточно гибкую систему организации файлов, учитывая то, что сайт может расти, а система файлов будет оставаться ясной и понятной.

В: О, а почему бы не помещать папки с рисунками во все другие папки, такие как about и beverages?

О: Опять же мы можем так сделать. Но мы предполагаем, что некоторые рисунки будут использованы на нескольких страницах, поэтому мы помещаем их все вместе в корневую папку. Если в вашем сайте используется множество изображений на различных его страницах, вы захотите, чтобы каждая ветвь имела свою собственную папку с картинками.

В: Каждая ветвь?



О: Способ, с помощью которого мы описываем расположение папок, можно сравнить с рассматриванием дерева сверху вниз. Вверху находится корень, а каждый путь от него к файлу или папке — это ветвь.



Упражнение

Сейчас вам нужно создать структуру папок и файлов, описанную на предыдущей странице. Рассмотрим, что именно вы должны сделать.

1. Определите место для вашей папки lounge и создайте в ней три новые папки нижележащего уровня — about, beverages и images.
2. Переместите файл **directions.html** в папку about.
3. Переместите файл **elixir.html** в папку beverages.
4. Переместите все рисунки в папку images.
5. И наконец, загрузите файл **lounge.html** и протестируйте работу всех ссылок. Проверьте свои результаты.

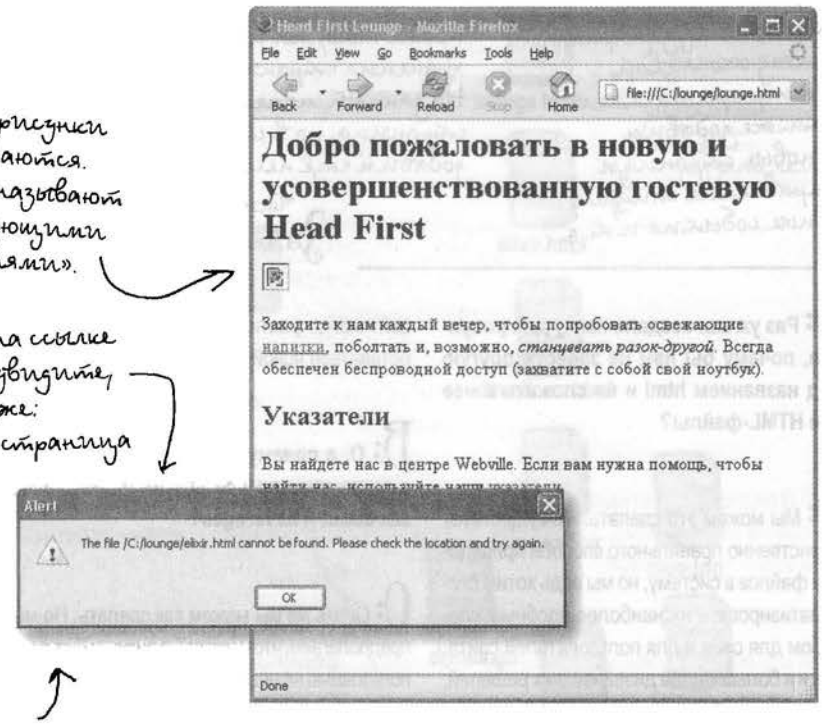
Технические трудности

Кажется, у нас появились кое-какие проблемы с гостевой страницей после приведения файлов и папок в систему...

Некоторые рисунки не отображаются. Обычно их называют «отсутствующими изображениями».

Если вы щелкнете кнопкой мыши на ссылке «напитки» (или «указатели»), то увидите, что все стало выглядеть намного хуже: появляется окно с сообщением, что страница не может быть найдена.

Одни браузеры отображают эту ошибку прямо в своем окне, другие — в отдельном диалоговом окне.





Правильно. Мы должны указать браузеру новое расположение страниц.

До сих пор вы использовали атрибут **`href`**, значения которого указывали на страницы *в этой же папке*. Но обычно сайты более сложные, и вы должны уметь указывать путь к страницам *из других папок*.

Для этого вы должны прослеживать путь от основной страницы к целевому файлу. Это может означать, что необходимо опуститься вниз или подняться вверх на папку или две, но в любом случае нужно получить *относительный путь* к файлу, который затем следует поместить в **`href`**.

Планирование путей



Что вы обычно делаете, когда планируете отдохнуть с семьей и отправиться в путешествие? Вы достаете карту и определяете путь к пункту назначения. При этом пути связаны с вашим нынешним месторасположением: ведь, если бы вы находились в другом городе, это были бы совершенно другие пути, верно?

Хорошо, вы действительно воспользуетесь картами Google, но пока поработайте здесь с нами!

Когда нужно выяснить относительный путь для ссылок, все делается точно так же. Вы начинаете со страницы, на которой расположена нужная ссылка, а затем определяете путь, проходя через все папки, пока не найдете файл.

Существуют также другие типы путей. Мы вернемся к этому в следующих главах.

Давайте проработаем парочку относительных путей к файлу (и заодно исправим гостевую страницу).

Формирование ссылок на вложенные папки

① Создание ссылки от файла lounge.html к файлу elixir.html

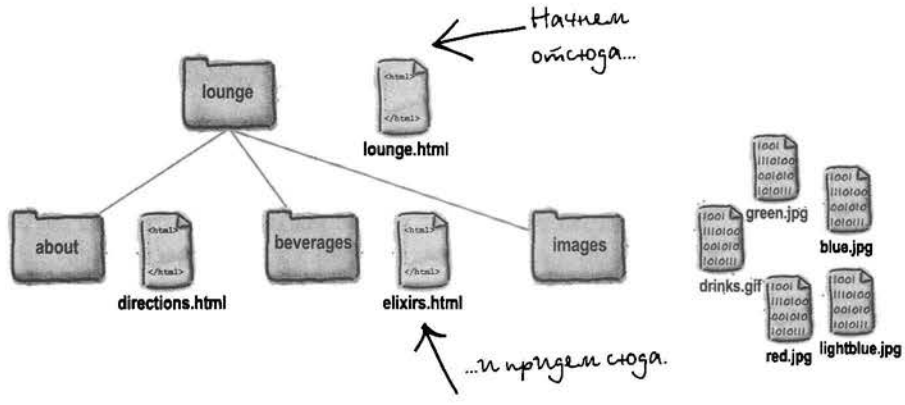
Наша задача – исправить ссылку напитки на странице `lounge.html`. Вот как элемент **<a>** выглядит сейчас:

```
<a  href="elixir.html">напитки</a>
```

Мы используем просто имя elixir.html, что говорит браузеру искать этот файл в той же папке, где и файл lounge.html.

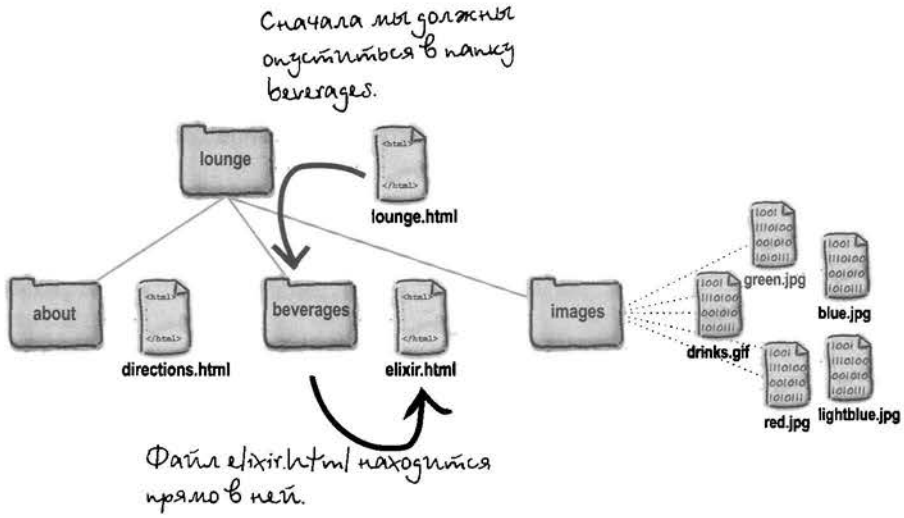
② Определение источника и адреса назначения

Когда мы реорганизовали гостевую, мы оставили файл `lounge.html` в папке lounge и поместили файл `elixir.html` в папку beverages, которая находится внутри папки lounge.



③ Прорисовка пути от источника к адресу назначения

Давайте прорисуем путь. Чтобы попасть от файла `lounge.html` к файлу `elixir.html`, мы должны войти в папку beverages.



④ Изменение значения атрибута href для указания прорисованного нами пути

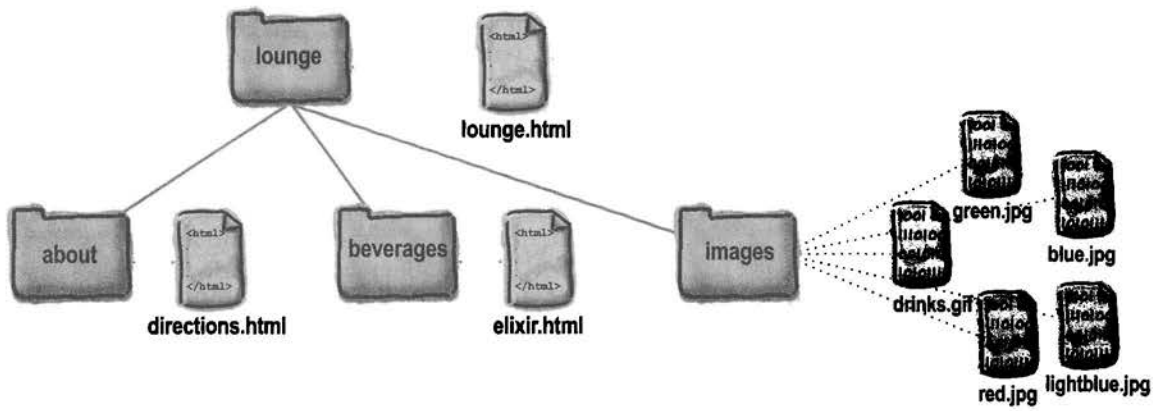
Теперь, когда мы знаем путь, мы должны преобразовать его в формат, распознаваемый браузером.



В качестве значения href мы используем относительный путь. Теперь, после того как вы выберете ссылку, браузер будет искать файл elixir.html в папке beverages.

Возьми в руку карандаш

Теперь ваша очередь: прорисуйте относительный путь от файла `lounge.html` к файлу `directions.html`. Когда справитесь с этим, дополните элемент **<a>**, приведенный ниже. Проверьте свой ответ в конце главы, а затем продолжайте работу и измените оба элемента **<a>** в файле `lounge.html`.




```
<a  href="_____">указатели</a>
```

ЗДЕСЬ БУДЕТ ВАШ ОТВЕТ

Идем другой дорогой: формирование ссылок на родительскую папку

1 Создание ссылки от файла directions.html к файлу lounge.html

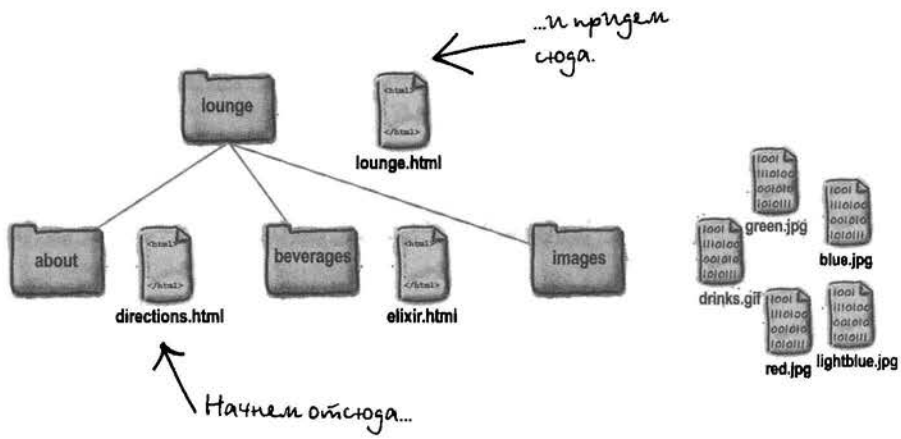
Теперь нам нужно исправить ссылку Назад в гостевую. Вот как сейчас выглядит элемент **<a>** в файле `directions.html`:

```
<a  href="lounge.html">Назад в гостевую</a>
```

Сейчас мы используем имя файла lounge.html, что говорит браузеру искать его в той же папке, где находится файл directions.html. Это уже не сработает правильно.

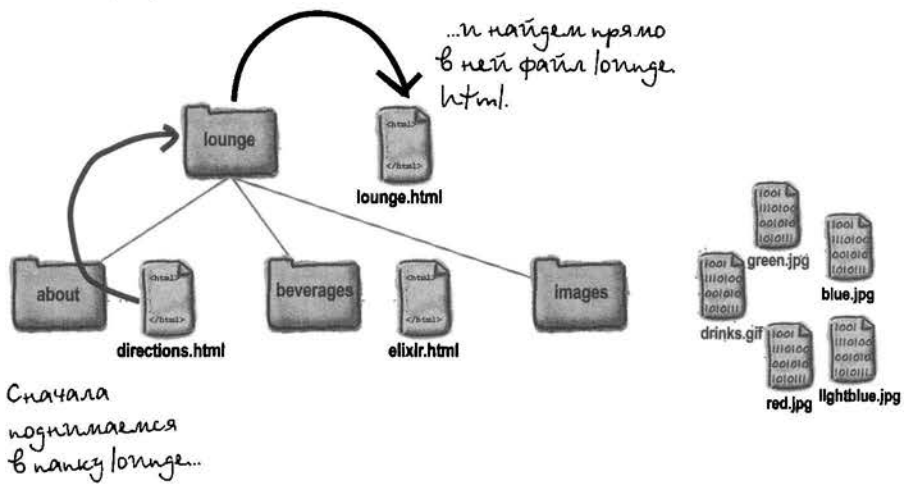
2 Определение источника и адреса назначения

Давайте посмотрим на файл-источник и целевой файл. Сейчас источник – это файл `directions.html`, который находится в папке about. Целевой файл – `lounge.html`, который находится на уровень выше папки about.



3 Прорисовка пути от источника к адресу назначения

Давайте прорисуем путь. Чтобы попасть от файла `directions.html` к файлу `lounge.html`, мы должны подняться на одну папку вверх, в папку lounge, где мы найдем файл `lounge.html`.



④ Изменение значения атрибута href для указания прорисованного нами пути

Мы почти закончили. Теперь, когда мы знаем путь, мы должны преобразовать его в формат, распознаваемый браузером. Поработаем над этим.

Сначала поднимемся на одну папку вверх. Как это сделать? С помощью символов «..». Верно, это две точки.

Выделяем каждую часть пути с помощью символа «/».

В конце пишем имя файла.

```
.. / lounge.html
```

Соберем все это вместе...

```
<a href="../lounge.html">Назад в гостевую</a>
```

Теперь, после того как вы выберете ссылку, браузер будет искать файл lounge.html в папке уровнем выше (в родительской папке).





Часто Задаваемые Вопросы

В: Что такое родительская папка? Если у меня есть папка под названием яблоки внутри папки фрукты, будет ли папка фрукты родительской для папки яблоки?

О: Да, точно. Папки (их еще называют директориями или каталогами) часто описываются с помощью терминов наследственных связей. Рассмотрим это, используя ваш пример. Папка фрукты — это родитель каталога яблоки, а каталог яблоки — ребенок (дочерний каталог) папки фрукты. Если бы у вас была еще папка груши, тоже ребенок папки фрукты, то она была бы сестрой каталога яблоки. Просто проводите аналогии с генеалогическим древом.

В: Хорошо, с родительскими папками понятно, но что это за «..»?

О: Если вам нужно сказать браузеру, что файл, на который вы ссылаетесь, находится в родительской папке, используйте символы «..». Они как бы говорят: «Двигайтесь вверх, в родительскую папку». Другими словами, так браузер обратится к родительской папке.
В нашем примере мы хотели попасть из файла directions.html, который располагается в папке about, в файл lounge.html, который находится в папке lounge, родительской для about. Поэтому нам нужно было сказать браузеру смотреть на одну папку выше. Использование символов «..» — это способ, которым мы говорим браузеру идти ВВЕРХ.

В: А что делать, если нужно подняться на две, а не на одну папку вверх?

О: Символы «..» можно использовать для каждой родительской папки, в которую вы хотите войти. Каждый раз, используя «..», вы поднимаетесь на одну папку вверх. Итак, если вы хотите подняться на две папки, введите «../..». Как видите, в этом случае необходимо отделять каждую часть пути с помощью символа «/». Не забывайте делать это, так как браузер не поймет, что означает «....»!

В: Когда я поднимусь на две папки вверх, как мне сказать браузеру, где искать файл?

О: Добавьте к символам «../..» имя файла. Так, чтобы сослаться на файл fruit.html, который находится в папке двумя уровнями выше, нужно написать ../../fruit.html. Вы, наверное, ожидаете, что мы назовем папку, подразумеваемую под символами «../..», «папкой-бабушкой», но обычно вместо этого мы говорим «родитель родительской папки».

В: Есть ли какие-нибудь ограничения в том, насколько высоко я могу подняться?

О: Вы можете подниматься до тех пор, пока не придете в корневую папку своего сайта. В нашем примере корневой папкой была папка lounge. Иными словами, вы не можете подняться выше этой папки.

В: А как насчет обратного направления? Есть ли какие-нибудь ограничения в том, насколько глубоко я могу опуститься вниз?

О: Вы можете опуститься вниз ровно на столько папок, сколько создали. Если вы создали папку на 10 уровней ниже корневой, то можете указать путь, который приведет вас вниз на 10 папок. Но мы не рекомендуем этого делать: если у вас так много уровней, это, скорее всего, означает, что организация вашего сайта слишком сложна!
К тому же существует ограничение на количество символов в пути: их должно быть не больше 255. Это достаточно большое количество символов, и вряд ли вам когда-либо столько понадобится. Однако если ваш сайт слишком большой, то старайтесь не забывать об этом ограничении.

В: Моя операционная система в качестве разделителя папок использует символ «\»; должен ли я тоже использовать его вместо символа «/»?

О: Нет; для веб-страниц всегда используется символ «/». Не используйте «\». В разных операционных системах применяются различные разделители (например, Windows использует «\» вместо «/»), но когда речь идет о Сети, мы выбираем общий для всех разделитель и привязываемся к нему. Поэтому неважно, какая у вас операционная система: Mac, Windows, Linux или какая-то еще. Всегда используйте символ «/», указывая путь в HTML.

МОЗГОВОЙ ШТУРМ

Ваша очередь: прорисуйте относительный путь от файла `elixir.html` к файлу `lounge.html` со ссылки Назад в гостевую. Чем он отличается от такой же ссылки в файле `directions.html`?

Ответ: Ничем, он абсолютно такой же.

Восстановление «отсутствующих изображений»

Ваша гостевая снова в почти рабочем состоянии. Все, что осталось исправить, – это те рисунки, которые не отображаются.

Мы пока не рассматривали подробно элемент **<img>** (сделаем это через пару глав). Главное для вас сейчас – знать, что атрибут **src** элемента **<img>** может содержать относительный путь точно так же, как атрибут **href**.

Вот элемент **<img>** файла `lounge.html`:

```
<img src="drinks.gif">
```

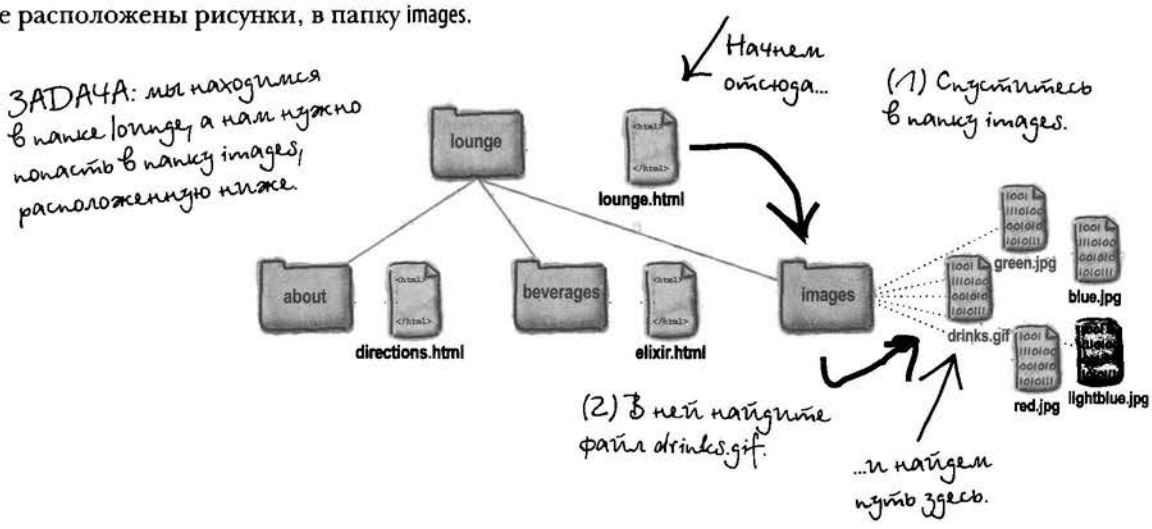
Это относительный путь, который говорит браузеру, где расположен рисунок. Мы указали его точно так же, как в атрибуте href элемента <a>.

Эй, это хорошо, что вы исправили все ссылки, но, кажется, вы кое-что забыли. Все наши изображения отсутствуют! Не томите нас, времени не так много.



Поиск пути от страницы lounge.html к рисунку drinks.gif

Чтобы найти путь, мы должны идти от файла `lounge.html` к месту, где расположены рисунки, в папку images.

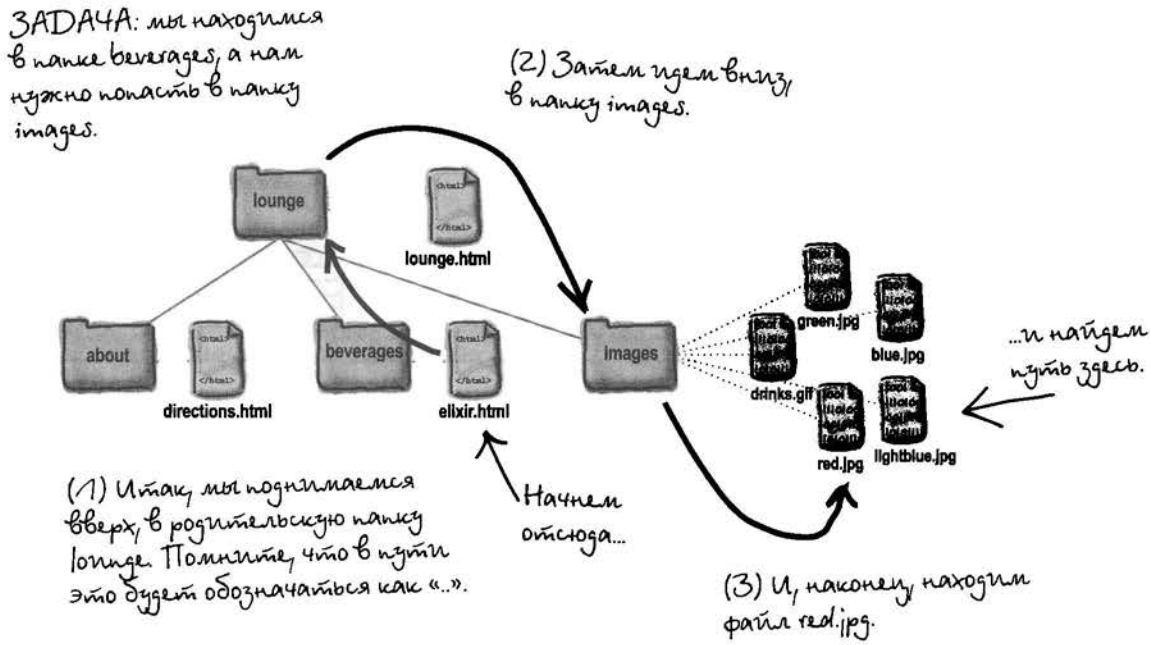


Итак, соединяя (1) и (2) вместе, получим путь `images/drinks.gif`, или:

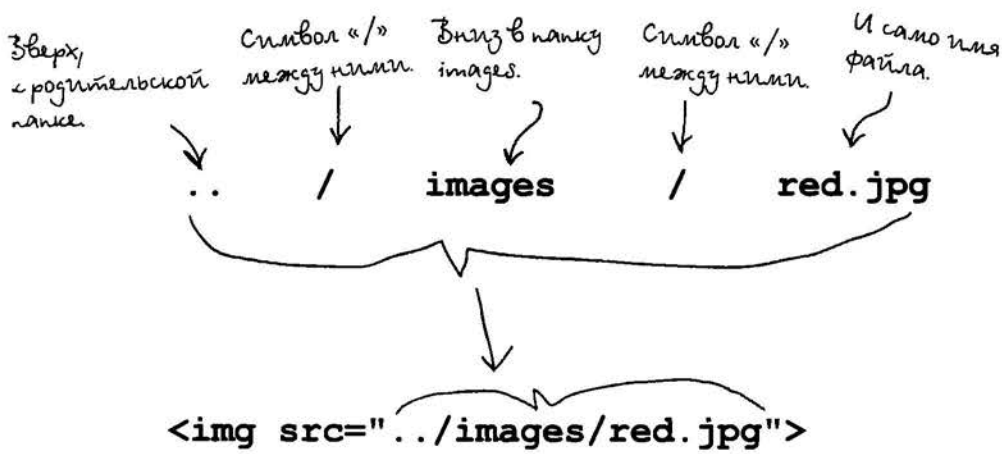
```
<img src="images/drinks.gif">
```

Поиск пути от страницы elixir.html к файлу red.jpg

Страница со списком напитков содержит изображения нескольких из них: `red.jpg`, `green.jpg`, `blue.jpg` и т. д. Давайте определим путь к `red.jpg`, а все остальные изображения будут иметь аналогичные пути, так как они расположены в одной папке.



Итак, соединяя (1), (2) и (3) вместе, получим:





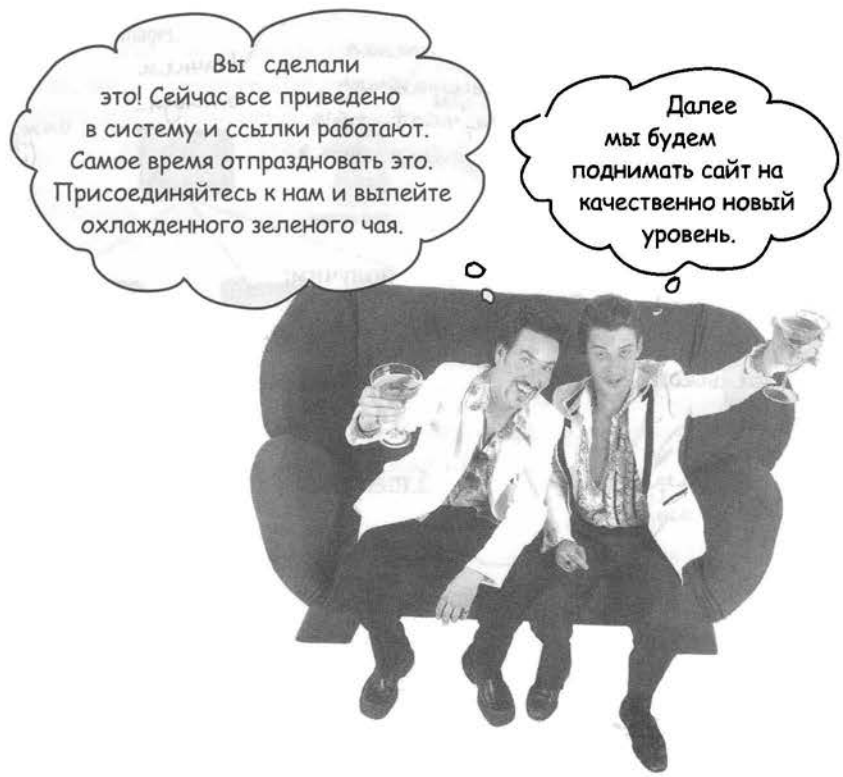
Упражнение

Все остальные ссылки, которые перестали работать, после реорганизации гостевой исправляются аналогичным образом. Вам все-таки необходимо внести изменения в файлы `lounge.html` и `elixir.html`. Рассмотрим, что именно нужно сделать.

1. В файле `lounge.html` измените значение атрибута **`src`** элемента **`<img>`** на `images/drinks.gif`.
2. В файле `elixir.html` исправьте значение атрибута **`src`** так, чтобы перед каждым названием файла с изображением было указано `../images/`.
3. Сохраните оба файла и загрузите страницу `lounge.html` в браузере. Сейчас вы сможете перемещаться по страницам и видеть все рисунки на них.



P. S. Если у вас возникли какие-то проблемы, воспользуйтесь рабочей версией гостевой из папки `chapter2/completelounge`. Проверьте по ней свою работу.





ПОВТОРИМ выученное

- Если вы хотите создать ссылку с одной страницы на другую, используйте элемент **<a>**.
- Атрибут **href** элемента **<a>** указывает на целевую страницу ссылки.
- Содержимое элемента **<a>** — метка ссылки. Метка — это то, что мы видим на веб-странице. По умолчанию она подчеркнута. Это говорит о том, что на ней можно щелкнуть кнопкой мыши.
- В качестве метки ссылки может быть использован текст или изображение.
- Когда вы щелкаете кнопкой мыши на ссылке, браузер загружает веб-страницу, которая указана в значении атрибута **href**.
- Можно ссылаться на файлы в этой же либо в другой папке.
- Относительный путь — это ссылка, указывающая на другие страницы вашего сайта относительно веб-страницы, на которой эта ссылка находится. Точно так же, как на карте место назначения имеет связь со стартовой точкой.
- Используйте символы «..», чтобы сослаться на файл, который находится в родительской папке (по отношению к папке, где находится файл, с которого вы ссылаетесь).
- Символы «..» означают «родительская папка».
- Не забывайте разделять части пути символом «/».
- Если путь к рисунку указан неверно, вы увидите «отсутствующее изображение» на веб-странице.
- Не используйте пробелы в именах, когда даете их файлам и папкам вашего сайта.
- Хорошо, если вы приведете свой сайт в систему с самого начала его создания, тогда вам не придется менять целые группы путей позже, когда он начнет расширяться.
- Существует множество способов приведения сайта в порядок. Вам решать, какой подходит именно вам.

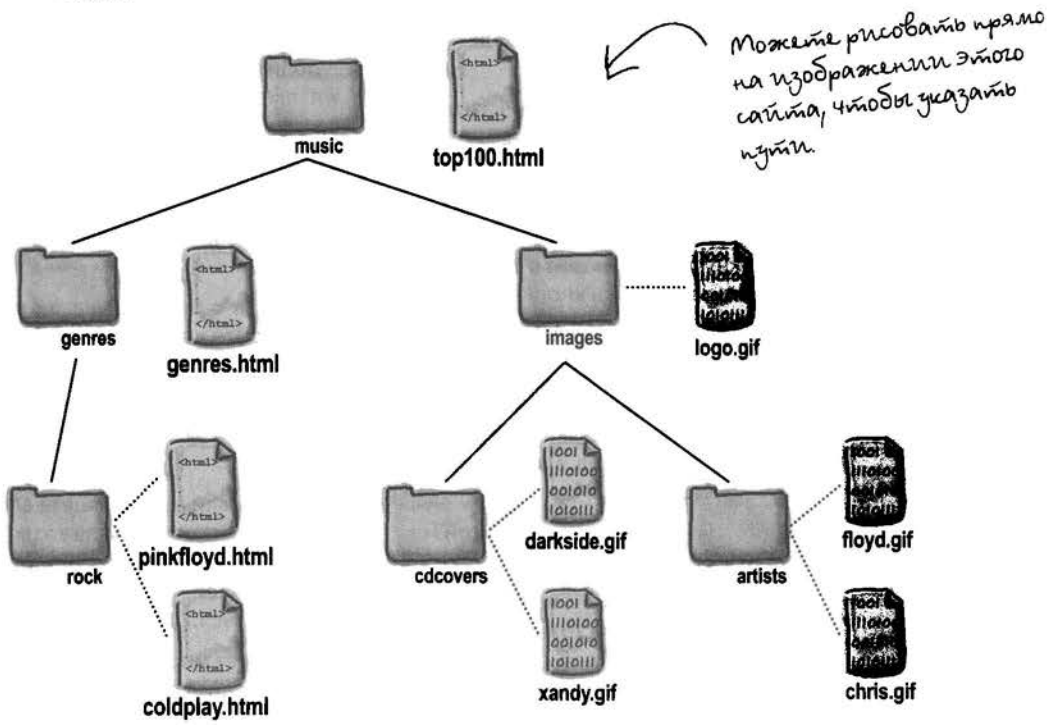


Большая задача по относительным путям

Это ваш шанс проверить свои навыки в области относительных путей. У нас есть сайт для 100 альбомов, расположенных в папке music. В этой папке находятся HTML-файлы, другие папки и рисунки. Ваша задача — определить необходимые относительные ссылки так, чтобы было можно сослаться с данной веб-страницы на другие веб-страницы и файлы.

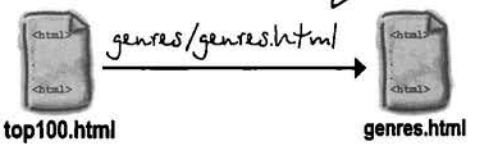
На этой странице вы найдете структуру сайта, а на следующей — задания для проверки ваших навыков. Ваша задача — указать правильный относительный путь для каждого исходного и целевого файла. Если вы справитесь с этим, можете считать себя настоящим чемпионом в области относительных путей.

Удачи!



Пришло время начать конкурс. На старт... Внимание... марш!

ПРИМЕР



Файл top100.html находится в папке music, поэтому, чтобы найти файл genres.html, мы должны опуститься в папку genres.

РАУНД ПЕРВЫЙ



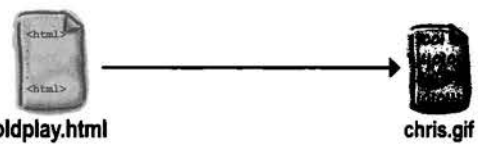
РАУНД ВТОРОЙ



РАУНД ТРЕТИЙ



БОНУСНЫЙ РАУНД





Решение упражнений

```
<html>
  <head>
    <title>Напитки гостевой Head First</title>
  </head>
  <body>
    <h1>Наши напитки</h1>

    <h2>Охлажденный зеленый чай</h2>
    <p>
                <img src="green.jpg">
                Этот напиток содержит огромное ко
                и минералов и приносит пользу для
                составу: зеленый чай, цветки рома
    </p>
    <h2>Охлажденный малиновый сироп</h2>
    <p>
                <img src="lightblue.jpg">
                Сочетая в себе малиновый сок и ли
                цитрусовых и плоды шиповника, это
                напиток освежит и прояснит ваш ра
    </p>
    <h2>Чудо-напиток из голубики</h2>
    <p>
                <img src="blue.jpg">
                Экстракты голубики и вишни, добавленные в травяной чай
                из бузины, сразу же приведут вас в состояние покоя
                и блаженства.
    </p>
    <h2>Клюквенный антиоксидантный взрыв</h2>
    <p>
                <img src="red.jpg">
                Зарядитесь энергией богатого витамином С напитка
                со вкусом клюквы и гибикуса.
    </p>
    <p>
        <a href="lounge.html">Назад в гостевую</a>
    </p>
  </body>
</html>
```



Здесь новый элемент <a> возвращает нас в гостевую.

Мы добавляем ссылку внутрь абзаца, чтобы сохранить аккуратность. В следующей главе мы более подробно поговорим об этом.



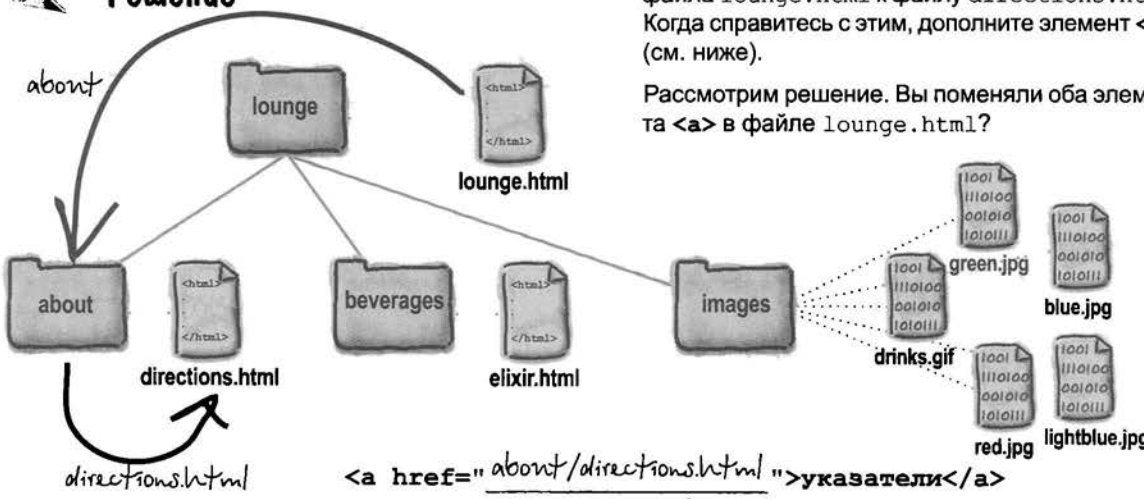
Решение упражнений



Метка	Адресат	Элемент
Горячо или нет?	hot.html	`<a href="hot.html">Горячо или нет?</a>`
Резюме	cv.html	Резюме
Вкусная конфета	candy.html	`<a href="candy.html">Вкусная конфета</a>`
Посмотри на мою Мини	mini-cooper.html	Посмотри на мою Мини
Давайте поиграем	millionaire.html	`<a href="millionaire.html">Давайте поиграем </a>`

Возьми в руку карандаш

Решение



Необходимо прорисовать относительный путь от файла `lounge.html` к файлу `directions.html`. Когда справитесь с этим, дополните элемент `<a>` (см. ниже).

Рассмотрим решение. Вы поменяли оба элемента `<a>` в файле `lounge.html`?

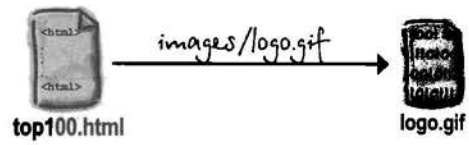
`<a href="about/directions.html">указатели</a>`

ЭТО И ЕСТЬ ОТВЕТ



Решение большой задачи по относительным путям

РАУНД ПЕРВЫЙ



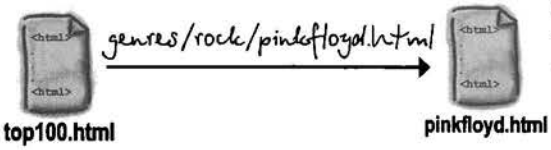
Файл top/100.html находится в папке music, поэтому, чтобы найти файл genres.html, мы должны опуститься в папку genres.

РАУНД ВТОРОЙ



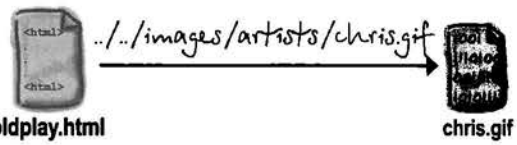
Файл genres.html находится внизу, в папке genres, поэтому, чтобы найти logo.gif, мы должны подняться в папку music, а затем опуститься в папку images.

РАУНД ТРЕТИЙ



От файла top/100.html мы опускаемся в папку genres, затем в rock и находим pinkfloyd.html.

БОНУСНЫЙ РАУНД



Тут была небольшая хитрость. От файла coldplay.html, который находится внизу, в папке rock, поднимаемся на DBE папки вверх и попадаем в папку music, затем опускаемся в папку images и, в конце концов, пытаемся отыскать картинку chris.gif. О, вот она!

Конструирование веб-страниц



Мы говорили вам, что в этой книге вы действительно будете создавать веб-страницы. Конечно, уже многое выучено: теги, элементы, ссылки, пути, однако все это бесполезно, если, используя полученные знания, не попробовать создать парочку потрясающих веб-страниц. В этой главе мы будем расширять строительство веб-страниц: перейдем от их общего осмысления к проектированию, зальем фундамент, построим их и даже выполним кое-какую отделку. Все, что вам нужно, — это усердие, трудолюбие и пояс для рабочих инструментов, так как мы будем добавлять некоторые новые инструменты и давать вам информацию, как пользоваться ими.

От дневника к сайту на скорости 12 миль в час

Для скутера рекомендуется предельная скорость.

Тони был очень занят, пока ездил по Соединенным Штатам на своем скутере. Почему бы вам не помочь ему и не создать для него веб-страницу?

Рассмотрим, что вам нужно сделать.

1. Во-первых, создайте приблизительный набросок дневника, который будет являться основой схемы страницы.
2. Затем используйте основные строительные блоки HTML (**<h1>**, **<h2>**, **<h3>**, **<p>** и др.) для интерпретации этого наброска на примерный план (или проект) HTML-страницы.
3. Когда составите план, преобразуйте его в настоящий HTML-код.
4. И последнее, когда основная страница будет готова, добавьте элементы для ее улучшения. Вы познакомитесь с некоторыми новыми элементами во время работы над сайтом.

СТОП! Выполните это упражнение, перед тем как перевернуть страницу.

Возьми в руку карандаш



Внимательно посмотрите на дневник Тони и подумайте над тем, как представить эту же информацию на веб-странице.

Справа нарисуйте эскиз для этой страницы. Не нужно делать его слишком затейливым, просто создайте черновик. Проследите, чтобы все записи из дневника были на странице.

Здесь будет ваш набросок

Подумайте над этим:

- представляйте себе страницу в виде крупных структурных элементов: заголовков, абзацев, изображений и т. д.;
- есть ли какие-то способы поменять дневник, чтобы его внешний вид был более подходящим для Сети?

Черновик

Дневник Тони очень похож на веб-страницу. Все, что остается сделать, — создать черновик для страницы, чтобы на ней содержались все записи из дневника, и составить план ее общей структуры. Для каждого дня, когда Тони заносил запись в дневник, он указывал заголовок в виде даты, иногда вставлял фотографию и кратко описывал, что произошло в этот день. Давайте посмотрим на черновик...

Тони назвал свой дневник «На скутере по США», так давайте и поместим это в самый верх страницы в качестве названия.

Он также придумал описание для своего дневника. Мы используем это как маленький абзац в верхней части страницы.

Каждый день Тони делает заметки, которые состоят из даты, чаще всего фотографии и описания его приключений в этот день. Итак, это название, изображение и абзац текста.

Иногда он не прилагает фотографию. Так, эта запись состоит только из названия (даты) и описания событий того дня.

Третья запись должна быть такой же, как и первая: заголовок, рисунок и абзац.

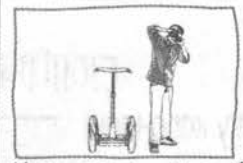
В отличие от бумажного дневника Тони, длина нашей веб-страницы не ограничена, и мы можем поместить много записей из дневника на одну страницу. Его друзья и семья могут просто использовать полосу прокрутки, чтобы просмотреть все записи на странице...

Тем не менее заметьте, что мы изменили порядок записей на странице на обратный: от последней к первой. Так пользователь сможет увидеть свежие записи в самом верху страницы, не используя полосу прокрутки.

На скутере по США

Дневник моих поездок на моем собственном скутере по территории США!

20 августа, 2005



Итак, я уже проехал 1200 миль и побывал в некоторых интересных местах: Вала-Вала, штат Вашингтон, Мэджик-Сити, штат Айдахо, Баунтифул, штат Юта, Лэст Чанс, штат Колорадо, Уай, штат Аризона, Трут-ор-Консекуэнсес, штат Нью-Мексико.

14 июля, 2005

Я видел парочку знаков в стиле Burma Shave на обочине дороги: «Если вы не заметите проезжающие мимо машины, то они могут сбить ва Одно мгновение — и бесконечность...» Я определен но не хотел, чтобы на меня наехала машина!

2 июня, 2005

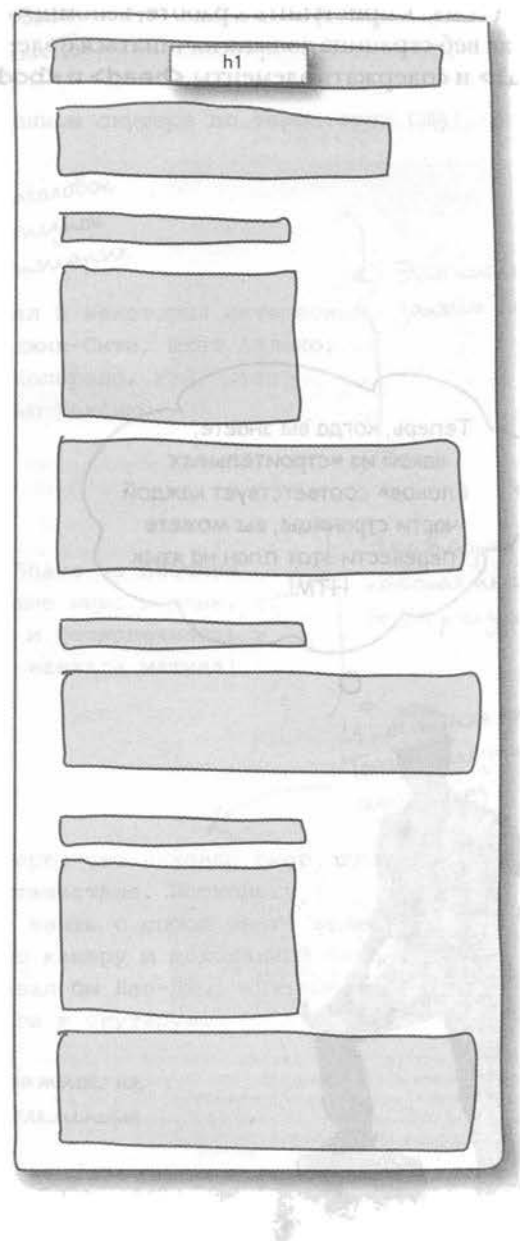


Первый день моего путешествия! Я не верю, что наконец смог отложить все дела в сторону и отправиться в путешествие. Поскольку я собирал ехать на скутере, то не мог взять с собой много вещей: только сотовый телефон, iPod, цифровую камеру и шоколадный батончик. Только все самое необходимое. Как сказал бы Лао-Цзы: «Путешествие на тысячу миль начинается с одного шага к скутеру».

От черновика к плану

Теперь, когда у вас есть черновик страницы, вы можете взять каждую ее часть и нарисовать нечто похожее на план или схему HTML-страницы...

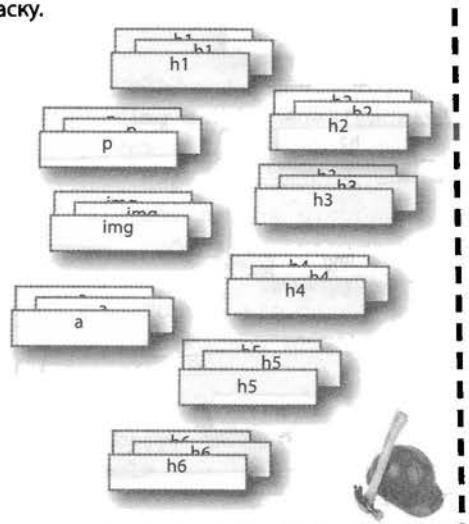
Здесь мы взяли каждый раздел черновика и создали соответствующий блок в схеме.



Все, что вам осталось сделать, — выяснить, какой HTML-элемент соответствует каждому разделу черновика, после чего вы сможете приступить к написанию HTML-кода.

УПРАЖНЕНИЕ: ВЕБ-КОНСТРУИРОВАНИЕ

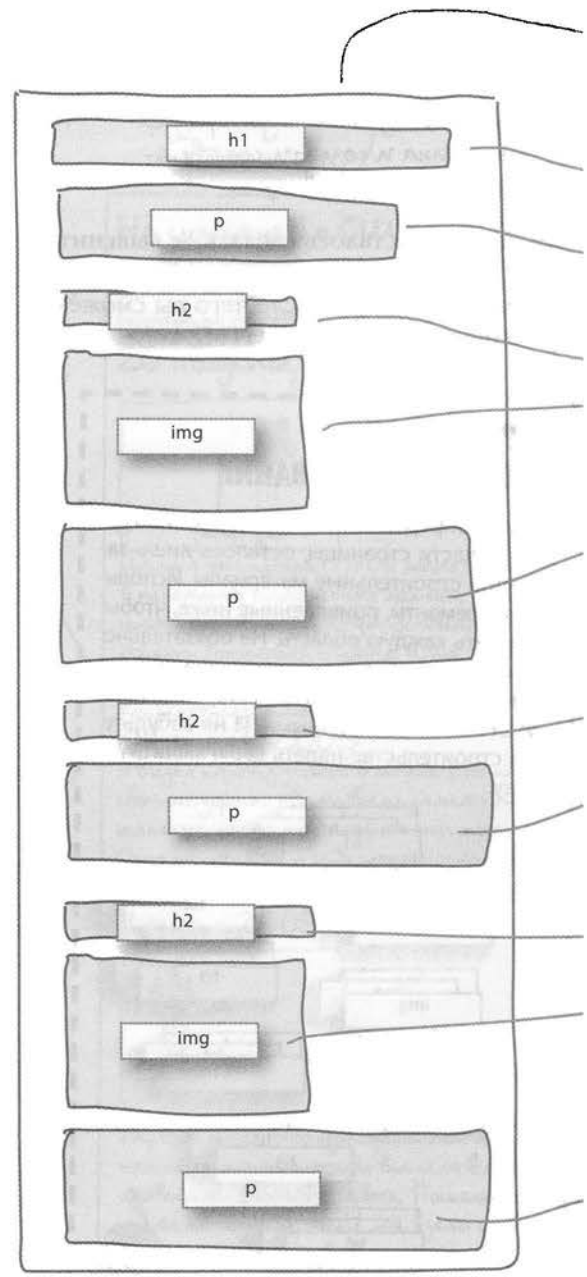
Вы уже определили основные архитектурные области страницы, осталось лишь закрепить строительные материалы. Используйте элементы, приведенные ниже, чтобы пометить каждую область. Не обязательно использовать их все, так что не волнуйтесь, если какие-то строительные материалы останутся неиспользованными. И не забудьте при строительстве надеть свою защитную каску.



От плана к Веб-странице

Вы почти достигли цели. Вы создали план страницы для Тони. Теперь осталось написать соответствующий HTML-код, чтобы сформировать страницу и внести в нее текст из дневника Тони.

Перед тем как приступить к работе, вспомните, что каждая веб-страница должна начинаться с элемента **<html>** и содержать элементы **<head>** и **<body>**.





Не забывайте, что всегда нужны элементы <html>, <head>, <title> и <body>.

Мы используем название дневника в качестве названия веб-страницы

```
<html>
  <head>
    <title>Дневник моих поездок на моем собственном скутере по территории США!</title>
  </head>
  <body>

    <h1>На скутере по США</h1>
    <p>
      Дневник моих поездок на моем собственном скутере по территории США!
    </p>

    <h2>20 августа, 2005</h2>
    <img src="images/segway2.jpg">
    <p>
      Итак, я уже проехал 1200 миль и побывал в некоторых интересных
      местах: Вала-Вала, штат Вашингтон, Мэджик-Сити, штат Айдахо,
      Баунтифул, штат Юта, Лэст Чанс, штат Колорадо, Уай, штат
      Аризона, Трут-ор-Консекуэнсес, штат Нью-Мексико.
    </p>

    <h2>14 июля, 2005</h2>
    <p>
      Я видел парочку знаков в стиле Burma Shave на обочине
      дороги: «Если вы не заметите проезжающие мимо машины, то
      они могут сбить вас. Одно мгновение — и бесконечность…»
      Я определенно не хотел, чтобы на меня наехала машина!
    </p>

    <h2>2 июня, 2005</h2>
    <img src="images/segway1.jpg">
    <p>
      Первый день моего путешествия! Я не верю, что наконец смог отложить
      все дела в сторону и отправиться в путешествие. Поскольку я
      собирался ехать на скутере, то не мог взять с собой много вещей:
      только сотовый телефон, iPod, цифровую камеру и шоколадный батончик.
      Только все самое необходимое. Как сказал бы Лао-Цзы: «Путешествие
      на тысячу миль начинается с одного шага к скутеру».
    </p>

  </body>
</html>
```

Заголовок и описание журнала Тони.

заголовок
рисунок
описание

Это последняя запись Тони.

Это вторая запись, которая не содержит рисунка.

И последняя запись Тони с рисунком segway1.jpg.

И последнее, но немаловажное: не забывайте закрывать элементы <body> и <html>.

Итак, продолжайте работу и наберите этот код в текстовом редакторе. Назовите файл journal.html и сохраните его в папке chapter3/journal. В папке images найдите картинки segway1.jpg и segway2.jpg. Когда справитесь с этим, протестируйте страницу в браузере.

Тестирование страницы Тони





Посмотрите, как хорошо выглядит эта страница. Вы поместили текст и изображения из дневника Тони на хорошо читаемую и структурированную веб-страницу.

Фантастика! Это выглядит великолепно; мне не терпится добавить еще пару записей на страницу.

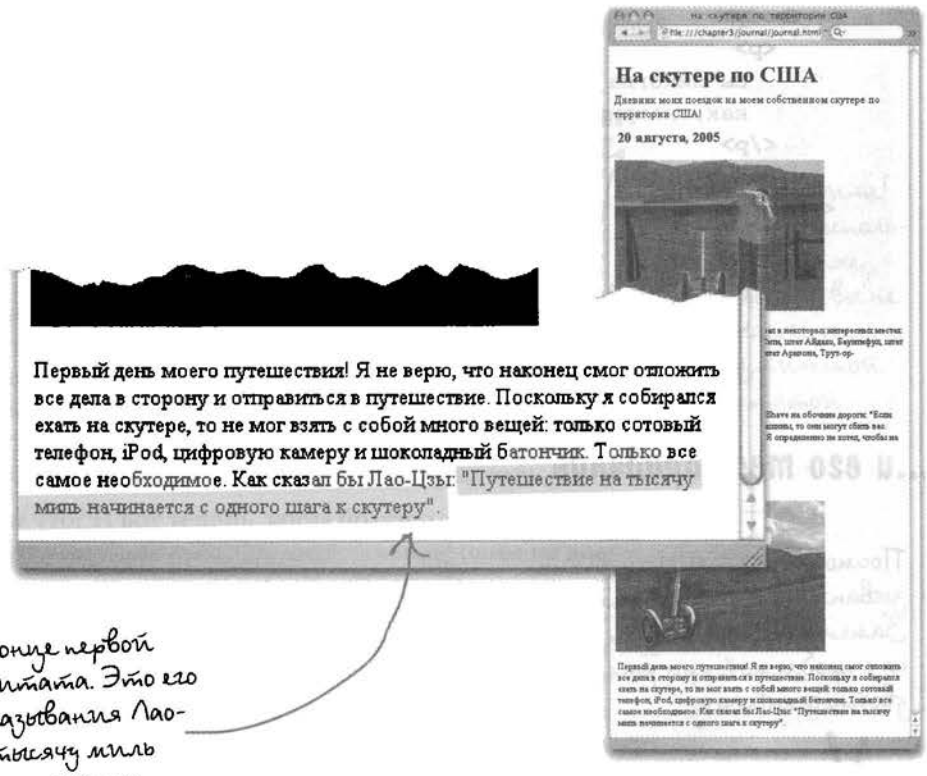


Тони звонит с дороги...

Добавление новых элементов

Вы уже использовали базовые элементы HTML: перешли от рукописного дневника к его онлайн-версии всего за несколько шагов благодаря HTML-элементам **<p>**, **<h1>**, **<h2>** и **<img>**.

Сейчас мы проведем небольшую разминку для вашего мозга и добавим еще немного элементов. Давайте еще раз посмотрим на дневник Тони и выясним, что можно улучшить...



Проверьте это: у Тони в конце первой записи есть небольшая цитата. Это его переделанная версия высказывания Лао-Цзы: «Путешествие на тысячу миль начинается с одного шага к скутеру».

Для таких вещей в HTML предусмотрен элемент <q>. Давайте посмотрим на следующую страницу...

Знакомство с элементом <q>

У вас в тексте есть цитата? Элемент **<q>** — это то, что вам нужно. Приведем небольшой пример HTML-кода, чтобы показать, как работает этот элемент.

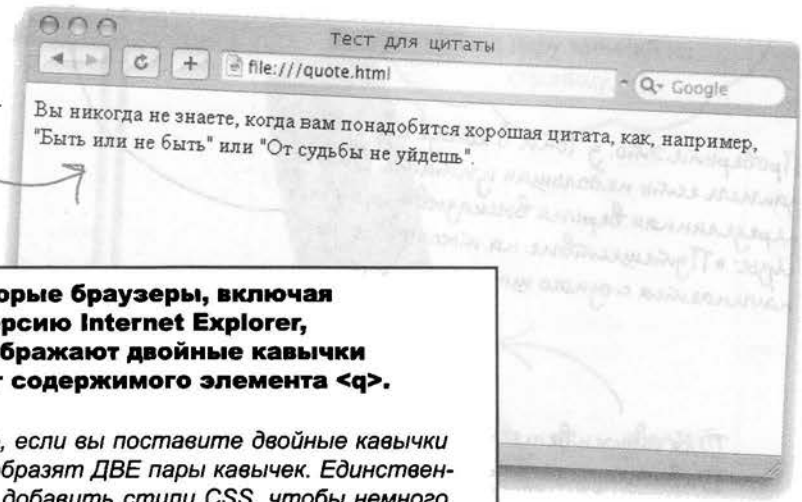
```
<html>
  <head>
    <title>Тест для цитаты</title>
  </head>
  <body>
    <p>
      Вы никогда не знаете, когда вам понадобится хорошая цитата,
      как, например, <q>Быть или не быть</q> или <q>От судьбы не уйдешь</q>.
    </p>
  </body>
</html>
```

В этом HTML есть две цитаты...

Мы окружили каждое высказывание открывающим тегом <q> и закрывающим </q>. Заметьте, что мы не ставим двойные кавычки вокруг высказывания.

...и его тестирование

Посмотрите, как эти высказывания выглядят в браузере. Заметьте, что браузер столкнулся с проблемой при расстановке двойных кавычек.





Будьте осторожны!

Некоторые браузеры, включая 6-ю версию Internet Explorer, не отображают двойные кавычки вокруг содержимого элемента <q>.

Это не очень хорошо, потому что, если вы поставите двойные кавычки сами, то некоторые браузеры отобразят ДВЕ пары кавычек. Единственный способ с этим справиться — добавить стили CSS, чтобы немного изменить ваши цитаты, например использовать курсив. Как выделить цитату курсивом, мы расскажем в главе 9.



Подождите минутку... Вы убрали двойные кавычки и заменили их элементом <q>, который просто используется для их отображения? Я должна удивиться?

Нет. Мы пытаемся сделать все более структурированным и выразительным.

Существует множество случаев, когда люди вводят в тексте двойные кавычки, но когда применяется элемент **<q>**, это означает, что используется *настоящая цитата* (в случае Тони «переделанная» цитата).

Посмотрите! Просто используя двойные кавычки, вы не делаете текст фактической цитатой.

Другими словами, вы должны указать браузеру, что хотите добавить именно цитату. Если вы просто поставите двойные кавычки, то браузер поймет, что есть абзац текста с парой двойных кавычек в нем. А если использовать элемент **<q>**, то браузер *определит*, что часть этого текста – реальная цитата.

Ну и что? И вот теперь, когда браузер знает, что это цитата, он может отобразить ее наилучшим из возможных способов. Некоторые браузеры отобразят двойные кавычки вокруг текста, некоторые не отобразят, а в отдельных случаях могут использоваться и другие методы. Кроме того, не забывайте о мобильных устройствах и речевых браузерах для людей с плохим зрением. Этот элемент полезен и в других ситуациях, например при работе поисковых механизмов, просматривающих Сеть и выбирающих страницы с цитатами. Структура и значение в ваших страницах – очень важные вещи.

Одна из главных причин применения элемента **<q>** (как вы сами убедитесь после того, как мы вернемся к дизайну и использованию CSS в книге чуть позже) – возможность стилизовать цитаты, чтобы они выглядели так, как вам больше нравится. Положим, вы хотите, чтобы цитата отображалась курсивом и зеленым цветом. Используя для ее отображения элемент **<q>**, вы легко сможете это сделать.



Упражнение

Это дневник Тони. Продолжайте работу и оформите его цитату Лао-Цзы с использованием элемента **<q>**. После того как вы сделаете это на бумаге, поменяйте файл `journal.html` и протестируйте его. Решение вы найдете в конце главы.

```
<html>
  <head>
    <title>На скутере по США</title>
  </head>
  <body>

    <h1>На скутере по США</h1>
    <p>
      Дневник моих поездок на моем собственном скутере по территории США!
    </p>

    <h2>20 августа, 2005</h2>
    <img src="images/segway2.jpg">
    <p>
      Итак, я уже проехал 1200 миль и побывал в некоторых интересных
      местах: Вала-Вала, штат Вашингтон, Мэджик-Сити, штат Айдахо,
      Баунтифул, штат Юта, Лэст Чанс, штат Колорадо, Уай, штат Аризона,
      Трут-ор-Консекуэнсес, штат Нью-Мексико.
    </p>

    <h2>14 июля, 2005</h2>
    <p>
      Я видел парочку знаков в стиле Burma Shave на обочине дороги:
      «Если вы не заметите проезжающие мимо машины, то они могут сбить
      вас. Одно мгновение — и бесконечность...» Я определенно не хотел,
      чтобы на меня наехала машина!
    </p>

    <h2>2 июня, 2005</h2>
    <img src="images/segway1.jpg">
    <p>
      Первый день моего путешествия! Я не верю, что наконец смог
      отложить все дела в сторону и отправиться в путешествие. Поскольку
      я собирался ехать на скутере, то не мог взять с собой много вещей:
      только сотовый телефон, iPod, цифровую камеру и шоколадный батончик.
      Только все самое необходимое. Как сказал бы Лао-Цзы: «Путешествие
      на тысячу миль начинается с одного шага к скутеру».
    </p>
  </body>
</html>
```

История двух элементов, разделенных после рождения

Пятиминутная головоломка



Пару лет назад в Webville родились однояйцевые близнецы, и по капризу судьбы, из-за того, что неправильно сработал интернет-маршрутизатор, близнецов разлучили сразу же после рождения. Оба росли, не зная о существовании друг друга, и только в силу некоторых странных обстоятельств позже встретились и выяснили свое родство, однако решили держать это в секрете.

После этого открытия они быстро поняли, что у них удивительно много общего. Оба были женаты на женщинах по имени Сайтейшн. Оба любили кого-нибудь цитировать. Первый – элемент **<q>** – любил короткие содержательные цитаты, в то время как второй – **<blockquote>** – любил длинные цитаты и часто запоминал большие отрывки из книг или стихотворений целиком.

Будучи однояйцевыми близнецами, они внешне были очень похожи друг на друга и поэтому решили реализовать хитрый план, посредством которого они будут заменять друг друга в различных ситуациях. Сначала они протестировали это на своих женах (в детали чего мы не будем углубляться) и прошли это испытание так, что их жены даже не догадались.

Затем они решили опробовать свою способность заменять друг друга на рабочем месте, так как по случайному стечению обстоятельств оба выполняли одну и ту же работу: разметку цитат в HTML-документах. Итак, в один прекрасный день они, никому не говоря, пошли на рабочие места друг друга, чтобы воплотить в жизнь свой зловещий план (в конце концов, если их жены ничего не поняли, то как смогут понять начальники?), и вот тогда начали проявляться печальные последствия их плана. Через 10 минут после начала рабочего дня обоих братьев сразу разоблачили как самозванцев, и власти стандартов были немедленно приведены в состояние боевой готовности.

Как же были пойманы близнецы? Продолжайте читать, чтобы узнать это...

Дли-и-и-инные цитаты

Теперь, когда вы знаете, как делать короткие цитаты, давайте рассмотрим длинные. Тони дал нам длинную цитату с музыкальной заставкой Burma Shave.

В своем дневнике Тони просто поместил цитату прямо внутрь абзаца. Однако не лучше ли вынести ее отдельно в свой собственный блок? Например, так:

Я видел парочку знаков в стиле Burma Shave на обочине дороги:

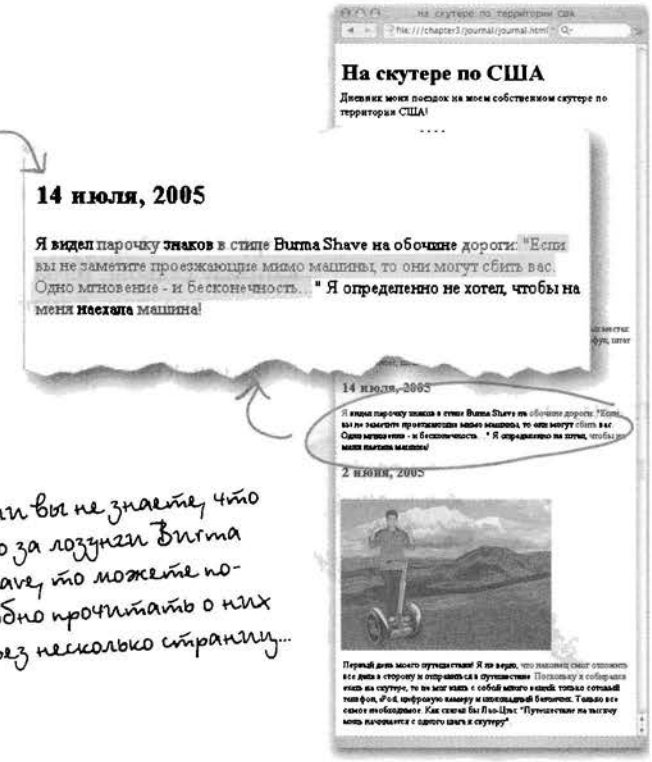
Если вы не заметите проезжающие мимо машины,

то они могут сбить вас.

Одно мгновение –

и бесконечность...

Я определенно не хотел, чтобы на меня наехала машина!



Если вы не знаете, что это за лозунги Burma Shave, то можете подробно прочитать о них через несколько страниц...

И вот тут вступает в действие элемент **<blockquote>**. В отличие от элемента **<q>**, который используется для коротких цитат, являющихся частью текущего абзаца, элемент **<blockquote>** применяется для длинных цитат, которые нужно отобразить отдельно.



Добавление элемента <blockquote>

Давайте добавим элемент **<blockquote>** в онлайн-версию дневника Тони.

1. Откройте файл `journal.html` и найдите запись от 14 июля. Измените абзац, чтобы он выглядел следующим образом:

```
<h2>14 июля, 2005</h2>
<p>
   Я видел парочку знаков в стиле Burma Shave на
   обочине дороги:
</p>
<blockquote>
   Если вы не заметите
   проезжающие мимо машины,
   то они могут сбить вас.
   Одно мгновение –
   и бесконечность…
</blockquote>
<p>
   Я определенно не хотел, чтобы на меня наехала
   машина!
</p>
```

Чтобы вставить элемент <blockquote>, сначала нужно закончить абзац.

Затем мы помещаем девиз Burma Shave внутрь элемента <blockquote>.

Мы также помещаем каждый пункт списка на новой строке, чтобы это больше походило на лозунг.

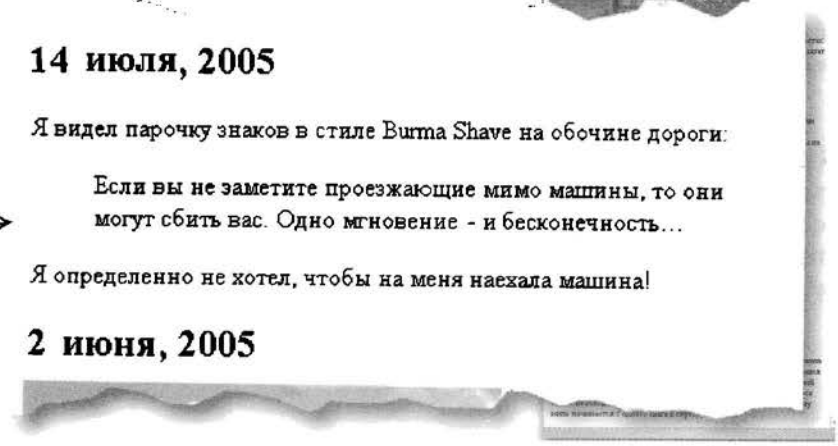
И наконец, мы должны добавить тег <p>, чтобы снова начать абзац после элемента <blockquote>.

2. Пришло время для нового теста. Откройте файл `journal.html` в браузере и посмотрите на результаты работы:



Элемент <blockquote> создает отдельный блок (как и элемент <p>), при этом добавляется небольшой отступ вправо, чтобы текст больше походил на цитату. Это как раз то, чего мы добивались...

Однако наша цитата выглядит не совсем так, как мы хотели, потому что она написана в одну строку. А ведь нам нужно разделить цитату на несколько строк. Хмммм... Вернемся к этому чуть позже.



Часто Задаваемые Вопросы

В: Итак, я правильно понял, что элемент <q> используется, если нужно выделить цитату прямо внутри абзаца, не вынося ее в отдельный блок, а <blockquote> — если необходимо вынести цитату в отдельный блок на веб-странице?

О: Да, все верно. Вы используете <blockquote>, если вам нужно выделить как цитату текст длиной в целый абзац или больше, а <q> — если просто нужно выделить как цитату небольшую часть из текущего текста.

В: Как можно оформить несколько абзацев в одном цитатном блоке?

О: Очень легко. Просто используйте элементы абзаца внутри элемента <blockquote>, для каждого абзаца должен быть свой элемент. *Попробуйте* это самостоятельно.

В: Как мне узнать, как будут выглядеть в моем браузере цитаты (встречающиеся внутри абзаца или выделенные в отдельный блок)? Я так понял, что разные браузеры обрабатывают их по-разному.

О: Да. Добро пожаловать во Всемирную паутину. На самом деле невозможно сказать, как будут выглядеть ваши цитаты, без тестирования в различных браузерах. Одни браузеры используют двойные кавычки, другие — выделение курсивом, а третьи вообще не выделяют цитаты. Единственный способ точно придать желаемый внешний вид цитатам — стилизовать их самостоятельно, и, конечно же, мы займемся этим чуть позже.

В: Я так понял, что элемент <blockquote> выносит цитату в отдельный блок и смещает ее вправо, почему же он не остается внутри абзаца, как элемент <q>?

О: Потому что <blockquote> — это на самом деле *новый* абзац. Это можно сопоставить с набором текста в текстовом редакторе. Когда один абзац заканчивается, вы нажимаете клавишу Enter и начинаете новый абзац. Чтобы напечатать отдельную цитату в блоке, вы сделаете то же самое и сместите цитату вправо. Пока просто запомните, так как это важная информация и мы скоро к ней вернемся.

Помните также, что отступ вправо — это просто способ, которым отдельные браузеры отображают содержимое элемента <blockquote>. И этот способ реализован не во всех браузерах; некоторые просто не будут использовать его в своих новых версиях. Поэтому не стоит полагать, что <blockquote> будет выглядеть одинаково во всех браузерах.

В: Можно ли совмещать цитатные элементы? Например, могу ли я использовать элемент <q> внутри элемента <blockquote>?

О: Конечно. Точно так же, как вы можете поставить элемент <q> внутрь элемента <p>, вы можете поместить <q> внутрь <blockquote>. Это можно сделать, если необходимо процитировать кого-то, кто тоже кого-то цитировал. Но элемент <blockquote> внутри <q> не имеет смысла, не так ли?

В: Вы сказали, что можно стилизовать эти элементы с помощью CSS. Если я, например, хочу выделить текст внутри элемента <q> курсивом или серым цветом, я могу использовать CSS. Но не проще ли задавать элемент для выделения цитаты курсивом?

О: Конечно, можно сделать и так, но это не совсем правильно, потому что вы будете использовать элемент только из-за его эффекта отображения курсивного текста, а не потому, что действительно хотите особо выделить текст. Если человек, которого вы цитируете, акцентирует внимание на каком-то слове или вы сами хотите использовать визуальное выделение, чтобы обратить особое внимание на цитату, то вам придется использовать элемент внутри цитаты. Но не применяйте его просто ради выделения курсивом. Есть более простой и приемлемый способ придать вашему тексту такой вид, какой вы хотите, с использованием CSS.

История двух элементов, разделенных сразу после рождения

Как однояйцевые близнецы были так быстро разоблачены?

Как уже было сказано, **<q>** и **<blockquote>** были разоблачены сразу же, как только приступили к работе и начали разметку документов. Обычно скромные маленькие цитаты элемента **<q>** неожиданно были выделены в отдельные блоки, а большие цитаты **<blockquote>** вдруг затерялись внутри абзацев. В последующих беседах с жертвами этой проделки один текстовый редактор жаловался: «Я потерял целую страницу вложенной цитаты из-за этих сумасшедших». После того как им объявили выговор и усадили обратно на соответствующие рабочие места, **<blockquote>** и **<q>** во всем честно признались своим женам, которые немедленно вместе покинули город. Но это уже совсем другая история (на этом все не закончилось).

Решение пятиминутной головоломки



Полное разоблачение тайны <q> и <blockquote>

Итак, пора прекратить шарады: **<blockquote>** и **<q>** – это совсем разные типы элементов. **<blockquote>** – *блочный* элемент, а **<q>** – элемент, расположенный *внутри* абзаца. В чем разница? Блочные элементы всегда отображаются так, словно перед ними и после них идет разрыв строки, в то время как элементы, находящиеся внутри абзаца, идут в основном тексте без вынесения на отдельную строку.



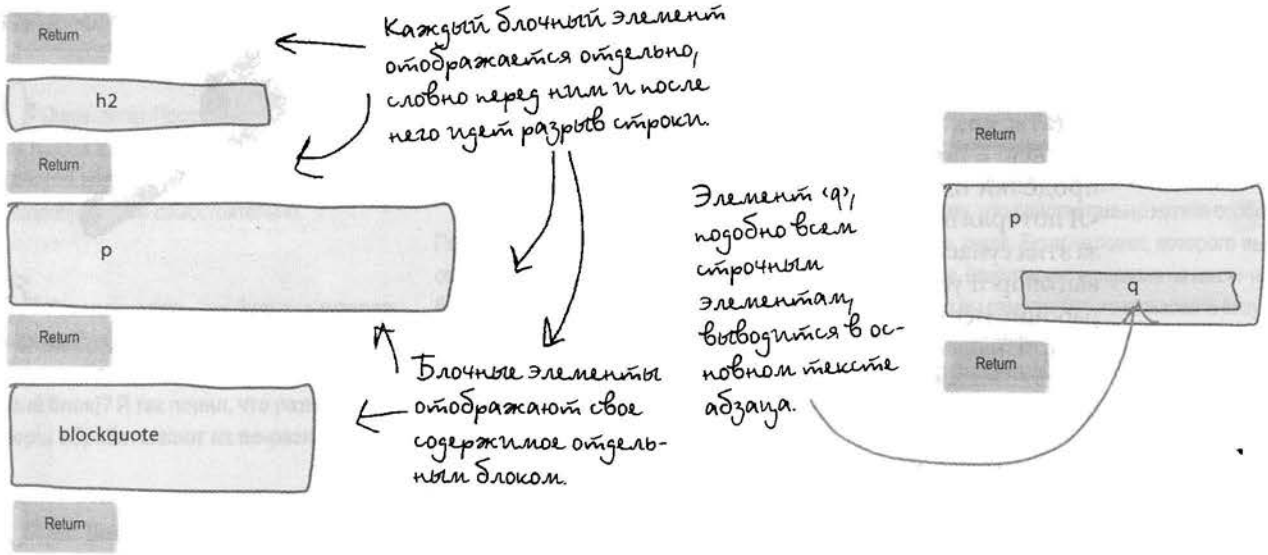
Блочный: отображается отдельным блоком



Строчный: отображается внутри абзаца

‹h1›, ‹h2›, ..., ‹h6›, ‹p› и ‹blockquote› – это блочные элементы

‹q›, ‹a› и ‹em› – строчные элементы



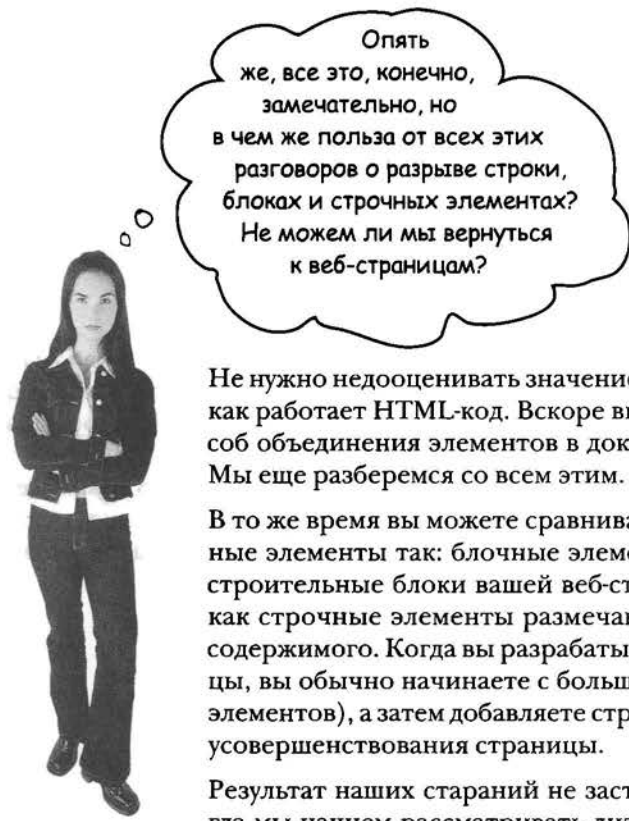
Запомните: блочные элементы отображаются отдельно, а строчные внутри основного текста страницы.

Часто Задаваемые Вопросы



В: Мне казалось, я знаю, что такое разрыв строки; это что-то вроде нажатия клавиши Enter на клавиатуре компьютера. Верно?

О: Почти верно. Буквально разрыв строки — это «разрыв в строке», и он происходит, когда вы нажимаете клавишу Enter. Вы уже знаете, что разрывы строки HTML-документа не отображаются, когда вы загружаете его в браузере. Теперь вы еще знаете, что каждый раз, когда вы применяете блочный элемент, браузер использует разрывы строки для отделения каждого блока.

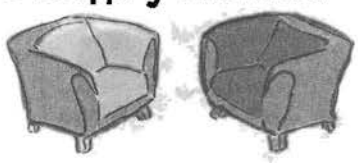


Не нужно недооценивать значение информации о том, как работает HTML-код. Вскоре вы убедитесь, что способ объединения элементов в документе очень важен. Мы еще разберемся со всем этим.

В то же время вы можете сравнивать блочные и строчные элементы так: блочные элементы — это большие строительные блоки вашей веб-страницы, в то время как строчные элементы размечают маленькие части содержимого. Когда вы разрабатываете дизайн страницы, вы обычно начинаете с больших частей (блочных элементов), а затем добавляете строчные элементы для усовершенствования страницы.

Результат наших стараний не заставит себя ждать, когда мы начнем рассматривать дизайн HTML-страниц с применением CSS. Если вы поймете разницу между блочными и строчными элементами, то сможете спокойно пить мартини, в то время как остальные все еще будут заняты корректировкой разметки своих HTML-документов.

Беседа у камина



Вечерний диалог: **строчный и блочный элементы рассуждают о своих различиях.**

Строчный элемент

Привет, Блочный. Я удивлен, что вижу тебя здесь.

Потому что ты достаточно нелюдим. Ты всегда ставишь вокруг себя эти разрывы строки, как будто они твои телохранители, не подпуская к себе никого.

Ты слишком много о себе возомнил. Да, ты действительно играешь важную роль, но кем бы ты был без строчного содержимого? Абзацы, заголовки и все такое — бессмыслица без текста и строчного наполнения, такого как, например, ссылки.

Говорю тебе с полной уверенностью, что **<a>** никуда не собирается переходить. Он был рожден и воспитан как строчный элемент. А если в твоих страницах не будет **<a>**, ****, **<q>** и всех остальных строчных элементов, то они станут совсем неинтересными, даже если их основа будет великолепной.

Блочный элемент

Почему это?

Просто я занятой парень. Блочные элементы — это действительно основные строительные блоки сайтов. Если бы меня не было, то все страницы просто развалились бы.

Я согласен, что **<a>** в самом деле важный элемент, и уже пытался переманить его на свою сторону. Но самое главное для веб-страницы — быть изначально хорошо сконструированной, а это сфера деятельности блочных элементов. Вы ведь не можете просто взять несколько ссылок и создать из них *настоящую* страницу?

Строчный элемент

Ну, многие сначала думали, что элемент **<img>** блочный, но это не так, и он приносит намного больше пользы как строчный элемент. Людям нравится, когда рисунки идут вперемешку с текстом и ссылками.

Потому что люди любят добавлять в основной текст маленькие цитаты. Я же не говорю ничего плохого про **<blockquote>**, так почему ты напираешь на **<q>**? Странно, ты думаешь, что строчные элементы так неважны, но тем не менее пытаешься завербовать многих из них.

О, как выкрутился. Скажи, пожалуйста, как бы выглядели твои страницы без строчных элементов? Я уверен, что в них не было бы никакой пользы. Вот!

Блочный элемент

Возможно, я и не получу **<a>**, но мне говорили, что **<img>** перейдет ко мне навсегда. Он станет замечательным блочным элементом.

Посмотрим. Скажу тебе еще кое-что: **<blockquote>** намного лучше, чем **<q>**. У нас есть замечательный элемент для создания блочных цитат. Зачем нам нужен **<q>**?

Где же ходят эти телохранители — разрывы строки, когда они мне так нужны? Посмотри, как много у меня накопилось работы. Мне нужно срочно возвращаться к построению страниц.



А что, если бы существовал элемент, чьим единственным занятием была бы установка разрывов строк там, где они необходимы?

Разве это не было бы замечательно? Можно было бы обратить внимание браузера на переходы на новую строку и заставить его добавить парочку разрывов строки там, где это нужно.

Оказывается, существует элемент **
**, который используется именно для этих целей. Вот как он работает:

```
<h2>14 Июля, 2005</h2>
<p>
   Я видел парочку знаков в стиле Burma
   Shave на обочине дороги:
</p>
<blockquote>
   Если вы не заметите <br>
   проезжающие мимо машины, <br>
   то они могут сбить вас. <br>
   Одно мгновение — <br>
   и бесконечность... <br>
</blockquote>
<p>
   Я определенно не хотел, чтобы на меня
   наехала машина!
</p>
```

Это запись из дневника Тони за 14 июля.

Добавьте элемент
 в любую строку, где хотите оборвать поток текста и вставить разрыв строки.

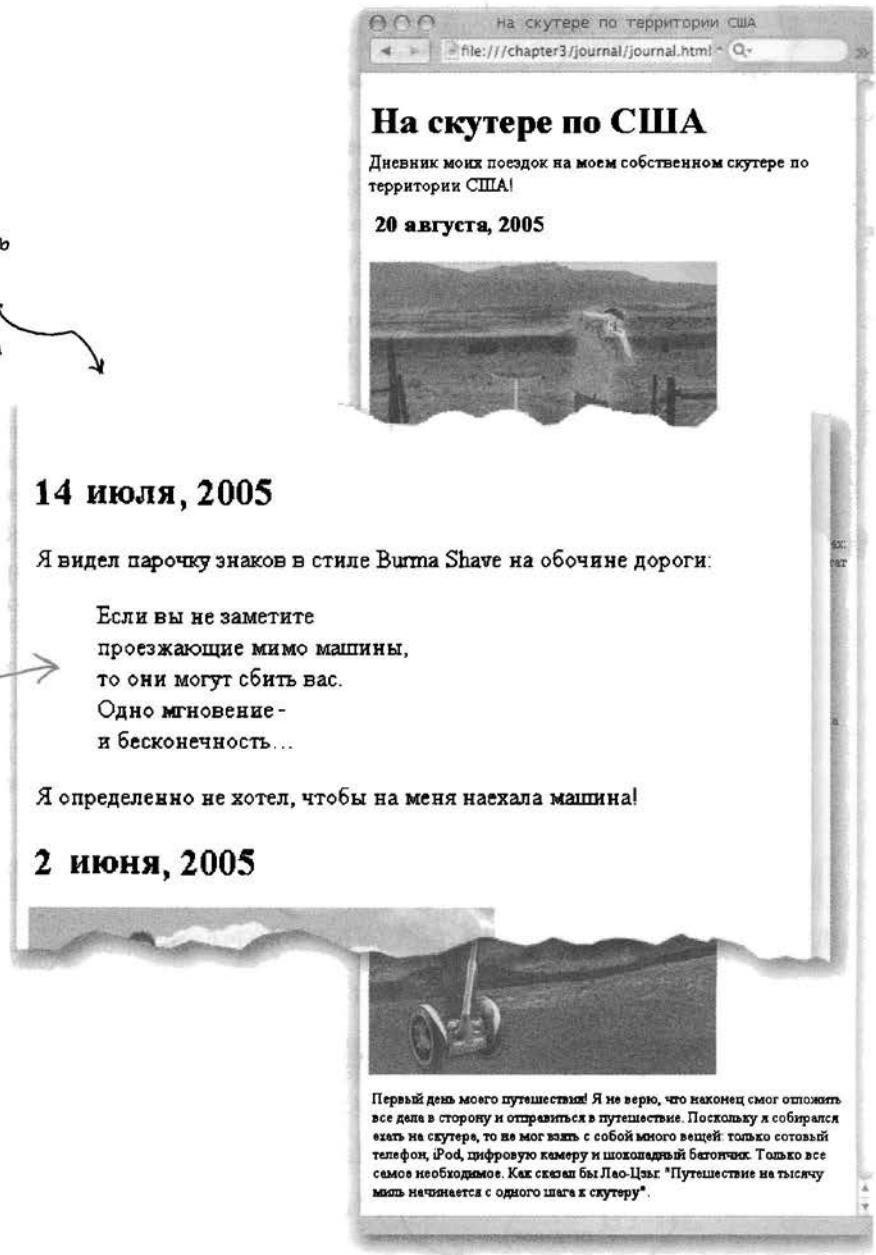


Упражнение

Продолжайте работу и добавьте элементы **
** в дневник Тони. После того как внесете изменения, сохраните файл и протестируйте его в браузере.

Вот как должны выглядеть изменения. Теперь это читается именно так, как должны читаться слоганы Burma Shave!

Сейчас после каждой строки идет разрыв.





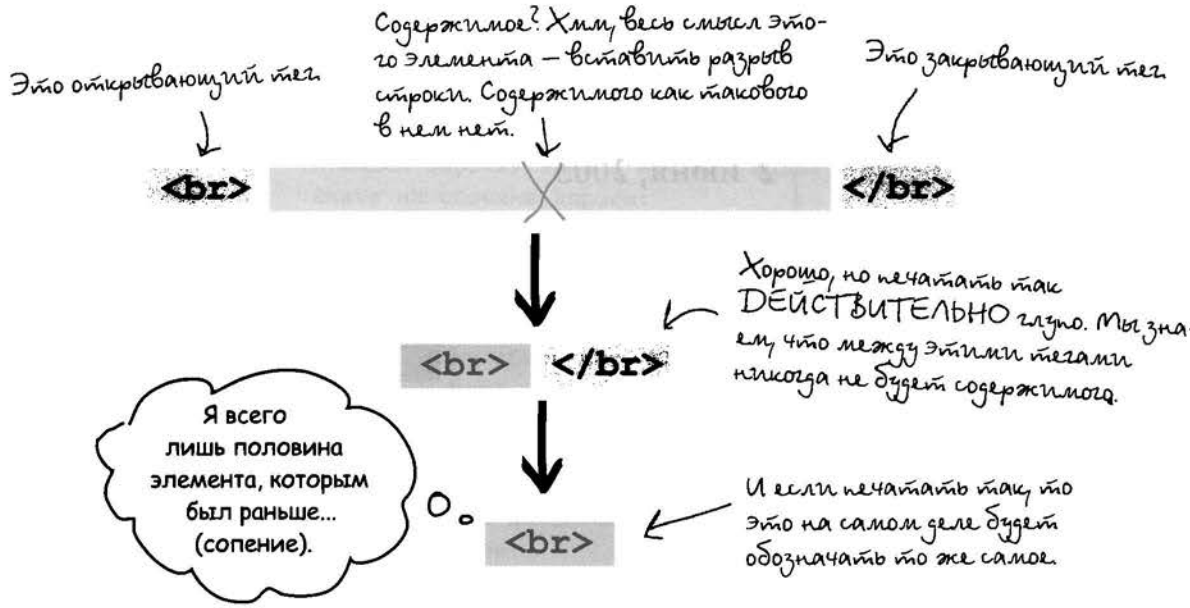
В главе 1 мы говорили, что элемент — это *открывающий тег + содержимое + закрывающий тег*. Как же
 может быть элементом? У него нет ни содержимого, ни даже закрывающего тега!

Именно, у него нет содержимого.

Элемент **
** — это элемент, у которого нет содержимого. Почему? Потому что предполагается, что он означает разрыв строки и ничего больше. Итак, если элемент изначально не имеет никакого содержимого, то мы просто используем краткое описание для представления элемента, и после **
** больше ничего не идет. В конце концов, если бы не было краткого описания, пришлось бы каждый раз для обозначения разрыва строки писать **
</br>**, а разве в этом есть смысл?

**
** — не единственный элемент такого типа; существуют и другие, и они все называются *пустыми элементами*. По сути, мы уже встречали другой пустой элемент — **<img>**. Через несколько глав мы вернемся к нему и разберем более подробно.

Примите во внимание, что краткое описание используется не из-за лени, а для повышения эффективности. Более эффективно представлять пустые элементы именно таким образом (для скорости набора, окончательного количества символов на HTML-странице и т. д.). И в самом деле, спустя некоторое время, проведенное за чтением HTML, вы поймете, что это также легче для ваших глаз.



Часто Задаваемые Вопросы

**В: Итак, единственное назначение элемента
 — вставлять разрывы строки?**

О: Верно. Единственное место, где браузеры сами вставляют разрывы в тексте, — там, где вы начинаете новый блочный элемент (как <p>, <h1> и т. д.). Если вы хотите вставить дополнительный разрыв строки в тексте, используйте элемент <br>.

**В: Почему элемент
 называют пустым?**

О: Потому что он не имеет содержимого, как обычный элемент, который включает в себя **открывающий тег + содержимое + закрывающий тег.**

**В: Я все еще не понял. Объясните, почему элемент
 пустой?**

О: Рассмотрим, например, элемент <h1> (или <p>, или <a>). Основная цель элемента — пометить тегами какое-то содержимое, к примеру:

```
<h1>Не ждите, заказывайте прямо сейчас</h1>
```

Что касается элемента <br>, то его цель — просто вставить разрыв строки в ваш текст. Тут нет содержимого, которое нужно разметить, поэтому он пуст. А поскольку он пуст, нам не нужны все дополнительные скобки и разметка, поэтому мы просто сокращаем этот элемент до наиболее удобной формы.

Если элементу не нужно размечать никакой текст, то, скорее всего, это пустой элемент.

В: Существуют ли еще какие-нибудь пустые элементы? Я думаю, что <img> также является пустым элементом, ведь так?

О: Да, есть еще парочка. Вы уже видели, как мы использовали элемент <img>, и вскоре мы подробно разберемся с ним.

В: Могу ли я сам сделать какой-нибудь элемент пустым? Например, если у меня есть ссылка, но я не хочу задавать для нее содержимое, могу я просто написать <a href="mypage.html">?

О: Нет. В мире существует два типа элементов: стандартные элементы, как <p>, <h1> и <a>, и пустые элементы, как
 и <img>. Вы не можете переключаться между ними. Например, если вы просто напечатали <a href="mypage.html">, то это не пустой элемент, а открывающий тег без содержимого и закрывающего тега.

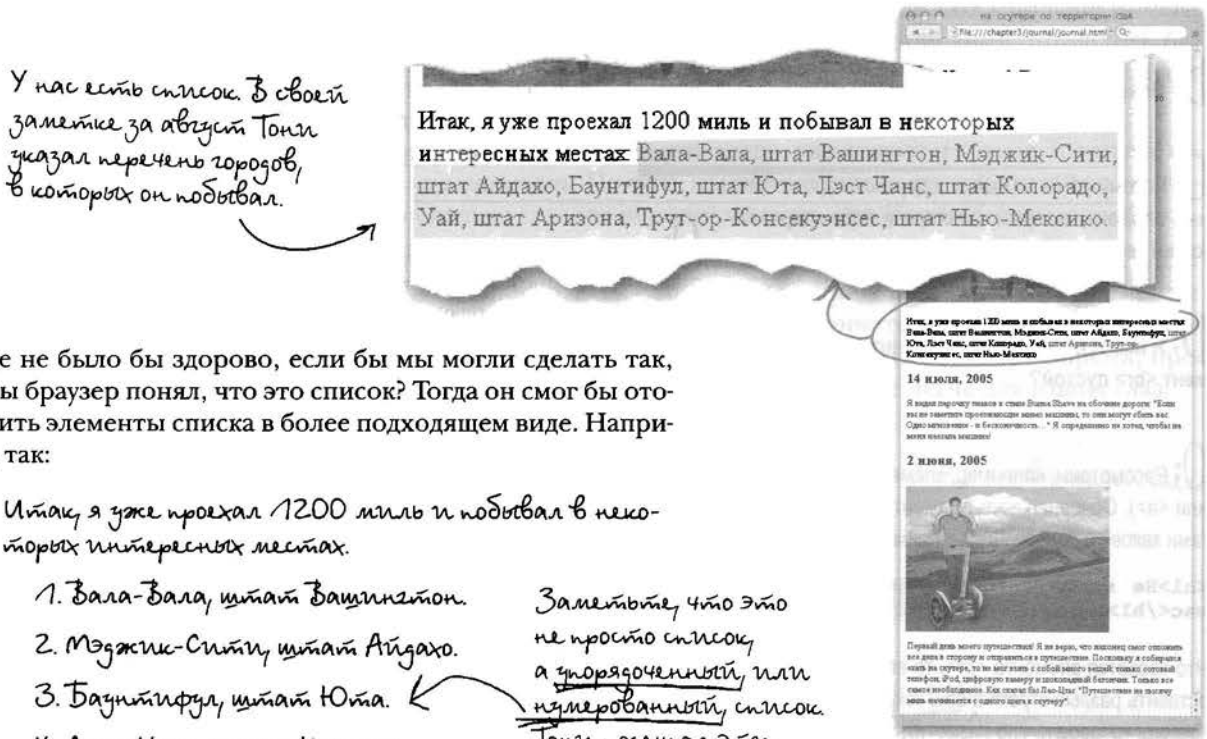
Элементы, изначально не имеющие HTML-содержимого, называются пустыми. Если вам нужно использовать пустой элемент, например,
 или <img>, то просто указывайте открывающий тег. Это удобная краткая форма, уменьшающая количество кода в вашем HTML-документе.

Тем временем вернемся к сайту Тони...

В этой главе вы уже много потрудились: спроектировали и создали сайт для Тони, познакомились с несколькими новыми элементами и узнали об элементах кое-что такое, о чем многие люди, создающие страницы в Сети, даже и не догадываются (например, о наличии блочных и строчных элементов, что действительно пригодится в следующих главах).

Однако вы еще не все сделали. Можно сделать так, чтобы сайт Тони был не просто хорошим, а великолепным. Надо только поискать немного новых возможностей HTML и еще добавить кода.

Например? Как насчет списков? Вот, смотрите.



Разве не было бы здорово, если бы мы могли сделать так, чтобы браузер понял, что это список? Тогда он смог бы отобразить элементы списка в более подходящем виде. Например, так:

Итак, я уже проехал 1200 миль и побывал в некоторых интересных местах.

1. Вала-Вала, штат Вашингтон.
2. Мэджик-Сити, штат Айдахо.
3. Баунтифул, штат Юта.
4. Лэст Чанс, штат Колорадо.
5. Уай, штат Аризона.
6. Трут-ор-Консекуэнсес, штат Нью-Мексико.

Заметьте, что это не просто список, а упорядоченный, или нумерованный, список. Тони посещал эти места в определенном порядке.

Конечно, можно использовать элемент <p> для создания списка

Можно с легкостью создать список, используя элемент **<p>**. Это будет выглядеть примерно так:

```
<p>
1. Красный скутер
</p>
<p>
2. Синий скутер
</p>
```

Два самых популярных цвета для скутера.

Но есть множество причин так не делать

К этому времени вы уже должны понимать основную идею. Всегда нужно стремиться выбрать тот HTML-элемент, который лучше всего отражает структуру вашего текста. Если нужно оформить список, используйте специальный элемент. Делая так, вы обеспечите браузер и себя (как вы убедитесь позже в этой книге) наилучшим способом отображения содержимого, который к тому же максимально производителен и гибок.



МОЗГОВОЙ ШТУРМ

Почему бы не использовать для создания списков элемент **<p>**? (Выберите все подходящие пункты).

- ❑ A. В HTML есть специальный элемент для создания списков. И если вы его используете, то браузер понимает, что текст является списком, и может отобразить его самым лучшим способом.
- ❑ B. Элемент **<p>** на самом деле предназначен для выделения абзацев, а не для построения списков.
- ❑ C. Вероятно, это будет смотреться не как список, а как пронумерованные абзацы.
- ❑ D. Если вы захотите поменять порядок элементов в списке или вставить новый элемент, вам придется перенумеровывать все элементы. Это трудоемко.

Ответы: A, B, C и D.

Разработка HTML-списков в два этапа

Для создания списков в HTML нужны два элемента, которые создают список только при условии использования их обоих. Первый применяется для разметки каждого *пункта списка*. Второй определяет тип создаваемого списка: *упорядоченный (нумерованный)* или *неупорядоченный (маркированный)*.

Давайте пошагово разберем создание списка городов, которые посетил Тони.

Шаг первый

Поместите каждый элемент списка в HTML-элемент <li>.

Чтобы создать список, нужно поместить каждый его пункт в отдельный HTML-элемент **
**. Иными словами, нужно оградить этот пункт списка слева открывающим тегом **
**, а справа – закрывающим **
**. Как и для всех других HTML-элементов, текст внутри тегов может быть любой длины. Кроме того, его можно разбить на любое количество строк.

Здесь показан только отдельный фрагмент HTML-кода из дневника Тони.

Поместите этот HTML-код в файл journal.html и измените его так, как мы предлагаем.

```
<h2>20 августа, 2005</h2>
    <img src="images/segway2.jpg" />
<p>
Итак, я уже проехал 1200 миль и побывал в некоторых
интересных местах:
</p>
```

Во-первых, возьмите элементы списка из абзаца каждый по отдельности. Список должен быть отображен обособленно.

```
<li>Вала-Вала, штат Вашингтон</li>
<li>Мэджик-Сити, штат Айдахо</li>
<li>Баунтифул, штат Юта</li>
<li>Лэст Чанс, штат Колорадо</li>
<li>Уай, штат Аризона</li>
<li>Трут-ор-Консекуэнсес,
    штат Нью-Мексико</li>
```

Затем оградите каждый элемент списка набором тегов <li> и .

Каждый элемент <li> станет пунктом списка.

```
<h2>14 июля, 2005</h2>

<p>
Я видел парочку знаков в стиле Burma Shave на
обочине дороги:
</p>
```

Шаг второй

Заключите все элементы списка либо в элемент <ol>, либо в элемент <ul>.

Если вы используете элемент **<ol>**, то пункты списка будут пронумерованы; если элемент **<ul>**, то список будет отображен как маркированный. Ниже приведен пример того, как правильно заключить пункты списка в элемент **<ol>**.

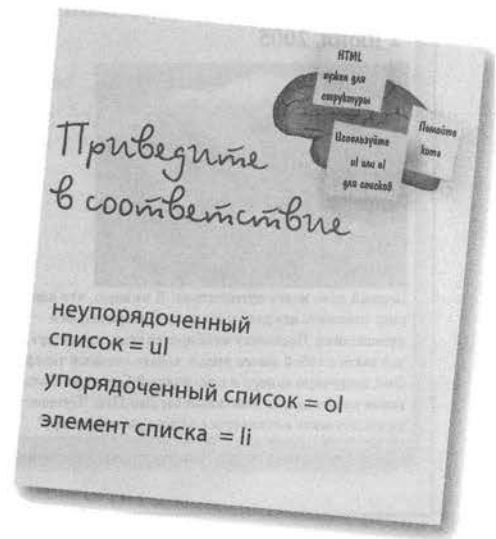
Опять-таки мы здесь приводим только отдельный фрагмент HTML-кода для дневника Тони.

```
<h2>20 августа, 2005</h2>
    <img src="images/segway2.jpg" />
<p>
Итак, я уже проехал 1200 миль и побывал
в некоторых интересных местах:
</p>

<ol>
  <li>Вала Вала, штат Вашингтон</li>
  <li>Мэджик-Сити, штат Айдахо</li>
  <li>Баунтифул, штат Юта</li>
  <li>Лэст Чанс, штат Колорадо</li>
  <li>Уай, штат Аризона</li>
  <li>Трут-ор-Консекуэнсес, штат Нью-Мексико</li>
</ol>

<h2>14 июля, 2005</h2>

<p>
Я видел парочку знаков в стиле Burma Shave на
обочине дороги:
</p>
```

Мы хотим, чтобы это был нумерованный список, потому что Тони посещал все эти города в определенном порядке. Для этого мы используем открывающий тег <ol>.

Все элементы списка помещаются внутрь элемента <ol> и становятся его содержимым.

Здесь мы закрываем элемент <ol>.

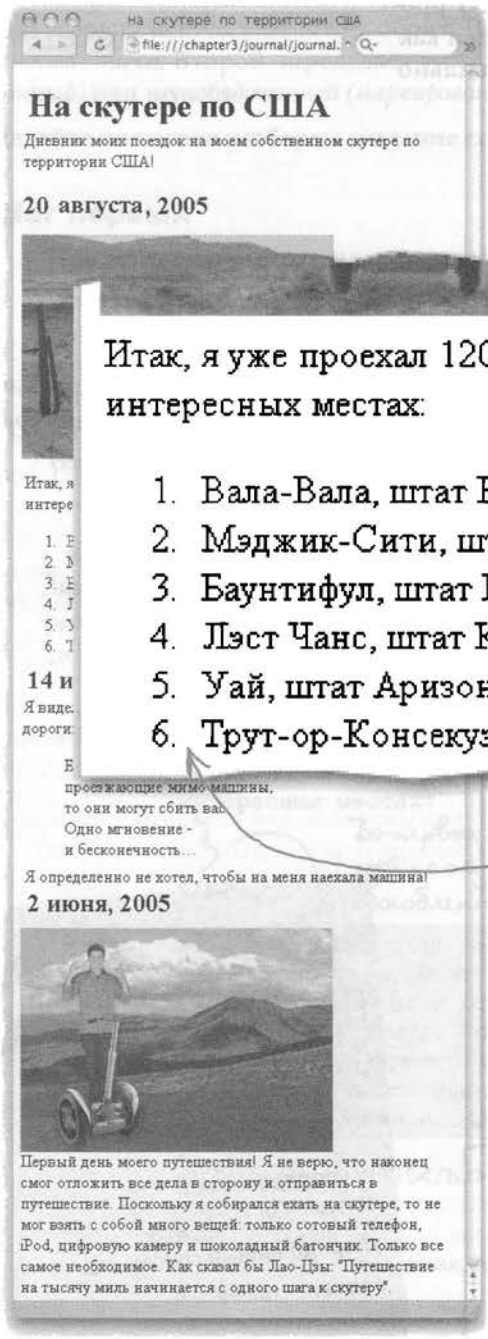


МОЗГОВОЙ ШТУРМ

Элемент **<ol>** – это блочный или строчный элемент? А как насчет **<li>**?

Тестирование списков на примере перечня городов

Убедитесь, что добавили весь HTML-код, необходимый для формирования списка, и обновите файл `journal.html`. В браузере вы увидите что-то вроде этого:



Новый, усовершенствованный список городов.

Перед списком добавлен разрыв строки, поэтому элемент <o>, должно быть, блочный.

Однако разрыв строки есть также после каждого пункта списка, значит <li> тоже блочный элемент!

Обратите внимание, что браузер сам пронумеровал все элементы списка (поэтому вам не нужно заниматься этим самостоятельно).

Возьми в руку карандаш

Оказывается, Тони посетил Аризону *после* Нью-Мексико. Можете ли вы изменить список так, чтобы нумерация стала правильной?



Упражнение

Вот другой список из дневника Тони: сотовый телефон, iPod, цифровая камера и шоколадный батончик. Вы найдете его в записи от 14 июля. Это *маркированный* список элементов.

HTML-код для этой записи дневника приведен ниже. Добавьте код, который бы преобразовал элементы в маркированный список (помните, что для таких списков используется элемент **<ul>**). Мы немного поменяли формат документа, чтобы вам было легче.

Когда справитесь с этим заданием, сверьте свой вариант решения с тем, что приводится в конце главы. Затем внесите эти изменения в свой файл `journal.html` и протестируйте его.

```
<h2>2 июня, 2005</h2>
<img src="segway1.jpg">
<p>
  Первый день моего путешествия! Я не верю, что наконец смог
  отложить все дела в сторону и отправиться в путешествие.
  Поскольку я собирался ехать на скутере, то не мог взять
  с собой много вещей: только

      сотовый телефон
      iPod
      цифровую камеру
      и шоколадный батончик

  Только все самое необходимое. Как сказал бы Лао-Цзы:
  <q>Путешествие на тысячу миль начинается с одного
  шага к скутеру</q>
</p>
```

Часто Задаваемые Вопросы

В: Элементы <ol> и <li> всегда нужно применять вместе?

О: Да, вы всегда должны использовать элементы <ol> и <li> (или <ul> и <li>) вместе. Ни один из этих элементов не имеет смысла без другого. Помните, что список — это группа элементов: элементы <li> используются для определения каждого пункта списка, а элемент <ol> — для их группировки.

В: Можно ли помещать текст или другие элементы внутрь элементов <ol> или ?

О: Нет, элементы <ol> и <ul> предназначены только для работы с элементами <li>.

В: Как насчет маркированных списков? Можно ли изменить внешний вид маркеров?

О: Да. Но не спешите пока. Мы вернемся к этому, когда будем говорить о CSS и дизайне веб-страниц.

В: А что, если я захочу поместить список в другой список? Это возможно?

О: Да, конечно. Используйте <ol> или <ul> в качестве содержимого какого-нибудь из элементов <li>, и вы получите список внутри другого списка (это называется вложенным списком).

Вложенный список

```
<ol>
   <li>Заправить скутер</li>
   <li>Собрать вещи в дорогу
      <ul>
         <li>сотовый телефон</li>
         <li>iPod</li>
         <li>цифровую камеру</li>
         <li>шоколадный батончик</li>
      </ul>
   </li>
   <li>Позвонить маме</li>
</ol>
```

Это элемент <li>. Он содержит вложенный список.

В: Я думаю, что понимаю разницу между блочными и строчными элементами, но я совершенно запутался с тем, какие элементы могут находиться внутри других, или, как вы говорите, что во что может быть вложено.

О: Это один из самых сложных моментов при изучении HTML. Вы будете постигать это на протяжении нескольких глав, а мы разными способами постараемся прояснить этот вопрос. Но все-таки сначала мы немного поговорим о вложенности элементов. И раз уж вы подняли этот вопрос, то это будет следующая тема, которую мы рассмотрим.

В: Итак, в HTML есть упорядоченные и неупорядоченные списки. А нет ли еще каких-нибудь типов списков?

О: На самом деле есть еще один: список определений. Он выглядит следующим образом:

Каждый элемент в этом списке имеет термин — <dt> и описание — <dd>.

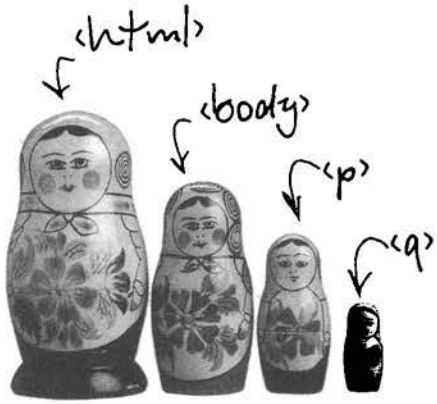
```
<dl>
   <dt>Знаки Burma Shave</dt>
   <dd>Дорожные знаки, распространенные в США
       в 1920-1930-е годы и рекламирующие товары
       для бритья.</dd>
   <dt>Шоссе 66</dt>
   <dd>Самое известное шоссе в сети автомагистралей
       Соединенных Штатов.</dd>
</dl>
```

Напечатайте это и протестируйте в браузере.

В: Burma Shave?

О: Была такая компания, которая в первой половине XX века делала крем для бритья, наносимый без помазка. Она начала рекламировать свою продукцию, используя дорожные знаки, в 1925 году, и эти знаки действительно имели большой успех (хотя и отвлекали внимание водителей).

Эти знаки объединялись в группы по четыре, пять или шесть штук, и на каждом была одна строка из слогана. Были времена, когда этих знаков на обочинах дорог в США можно было насчитать около 7000. Сейчас большинства из них уже нет, хотя парочку еще можно увидеть то тут, то там.



Добавление одного элемента в другой называется вложенностью

Когда мы помещаем один элемент внутрь другого, мы называем это *вложенностью*. Говорят, что «элемент **<p>** вложен в элемент **<body>**». Вы уже видели множество элементов, вложенных в другие. Вы помещали элемент **<body>** в элемент **<html>**, элементы **<p>** – в элемент **<body>**, элемент **<q>** – в элемент **<p>** и т. д. Вы также помещали элемент **<head>** в элемент **<html>**, а элемент **<title>** – в **<head>**. Таким образом и создаются HTML-страницы.

Чем больше вы будете узнавать о HTML, тем более важно четко осознавать, что такое вложенность. Но не волнуйтесь раньше времени, вскоре вы, конечно же, будете думать об элементах примерно так.



Чтобы понять отношения вложенности, изобразим это графически

Графическое изображение вложенности элементов на веб-странице можно сравнить с прорисовкой генеалогического древа. В самом верху расположены бабушки и дедушки, внизу — их дети, внуки и т. д. Вот, например:

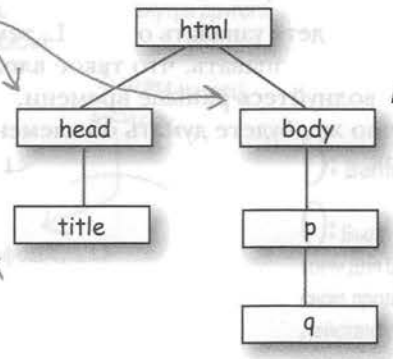
Простая веб-страница.

```
<html>
  <head>
    <title>Размышления</title>
  </head>
  <body>
    <p>
      Как говорил Бакару,
      <q>Единственный
      аргумент в пользу
      времени, это то —
      что все происходит
      неодновременно.</q>
    </p>
  </body>
</html>
```

Преобразуем это в схему, где каждый элемент будет блоком, а каждая линия будет соединять определенный элемент с другим элементом, вложенным в него.

‹html› всегда находится в корне дерева.

‹html› содержит два вложенных элемента: ‹head› и ‹body›. Вы можете называть их «детьми» элемента ‹html›.



‹body› вложен в элемент ‹html›, поэтому мы говорим, что ‹body› — «ребенок» ‹html›.

‹title› вложен в элемент ‹head›.

‹p› — «родитель» ‹q›, ‹body› — «родитель» ‹p›, ‹html› — «родитель» ‹body›.

Используйте вложенность, чтобы убедиться в соответствии тегов

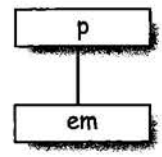


Первый результат понимания того, как вложены друг в друга элементы, — это то, что у вас все теги соответствуют друг другу (вскоре будет еще больше результатов).

Что же означает «несоответствие тегов» и как его можно избежать? Взгляните на этот пример:

```
<p>Я планирую поместить<em>это</em>в блог</p>
```

Вот так выглядит HTML-код, где элемент вложен в элемент <p>.

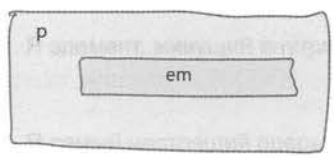


Пока все идет нормально, но ведь очень просто можно перепутать и написать код так:

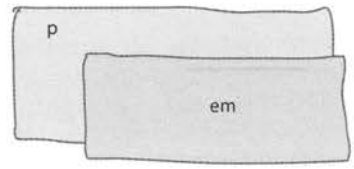
```
<p>Я планирую поместить<em>это</p>в блог</em>
```

ОШИБКА: тег <p> заканчивается перед тегом ! Предполагается, что элемент расположен внутри элемента <p>.

Принимая во внимание уже имеющуюся у вас информацию о вложенности, вы знаете, что элемент **** должен быть полностью вложен в элемент **<p>**.



ХОРОШО: здесь элемент вложен в элемент <p>.



ПЛОХО: здесь элемент вышел за границу элемента <p>. Это означает, что он, по сути, не вложен в него.

Какая разница?

Это нормально, что вы запутались во вложенности. Если вы пишете HTML-код, не соблюдая вложенности элементов, то ваши страницы могут работать в одних браузерах и не работать в других. Но если вы постоянно заботитесь о вложенности, то сможете избежать несоответствия тегов, а ваши страницы будут хорошо отображаться во всех браузерах. Это будет иметь еще большее значение в последующих главах, когда мы основательно займемся «профессиональным» HTML.

Поработайте браузером



Ниже вы найдете HTML-файл, в котором некоторые теги не согласованы. Ваша задача — выполнить работу браузера и найти все ошибки. (Сделав это упражнение, проверьте, все ли ошибки вы нашли (ответ смотрите в конце главы).

```
<html>
<head>
    <title>Топ-100</title>
<body>
<h1>Топ-100
<h2>Dark Side of the Moon</h2>
<h3>Pink Floyd</h3>
<p>
    У Луны нет темной стороны; на самом деле <q>она вся темная.
</p></q>
<ul>
    <li>Speak to Me / Breathe</li>
    <li>On The Run</li>
    <li>Time</li>
    <li>The Great Gig in The Sky</li>
    <li>Money</li>
    <li>Us And Them</em>
    <li>Any Colour You Like</li>
    <li>Brain Damage</li>
    <li>Eclipse</li>
</ul>
</p>
<h2>XandY</h3>
<h3>Coldplay</h2>
<ol>
    <li>Square One
    <li>What If?
    <li>White Shadows
    <li>Fix You
    <li>Talk
    <li>XandY
    <li>Speed of Sound
    <li>A Message
    <li>Low
    <li>Hardest Part
    <li>Swallowed In The Sea
    <li>Twisted Logic
</ul>
</body>
</head>
```

Кто я?



Группа HTML-элементов в маскарадных костюмах играет на вечеринке под названием «Кто я?». Они дадут вам ключ к разгадке, благодаря которому вы попытаетесь угадать, кто они на самом деле. Предполагается, что они все время говорят о себе правду. Опознайте участников и заполните правые столбцы бланка. Кроме того, напишите для каждого участника игры, блочный он или строчный.

Сегодняшние участники:

все очаровательные HTML-элементы, которые вы уже встречали раньше, должны быть разоблачены!

	Имя	Строчный или блочный?
Я заголовок первого уровня.	_____	_____
Я готов сослаться на другую страницу.	_____	_____
С моей помощью вы можете особо выделить текст.	_____	_____
Я список, но я не расставляю свои элементы по порядку.	_____	_____
Я элемент, живущий внутри списка.	_____	_____
Я самый настоящий прерыватель строк.	_____	_____
Я содержу свои элементы в порядке.	_____	_____
Я знаю все об изображениях.	_____	_____
Со мной вы можете выделить цитату внутри абзаца.	_____	_____
Со мной вы можете выделить цитату в отдельный блок.	_____	_____



Вы правы, это может привести к определенным проблемам.

Поскольку браузер использует символы < и > как начало и конец тега, применение их внутри содержимого вашего HTML-кода может привести к проблемам. Но HTML дает вам легкий способ определять эти и другие специальные символы с помощью простых аббревиатур, называемых *ссылками на символы*.

Рассмотрим, как это работает. Для каждого символа, который считается специальным или который вы хотите использовать на своей веб-странице, но который невозможно напечатать в вашем редакторе (например, символ авторского права), вы находите аббревиатуру и печатаете ее в HTML. Например, для символа > аббревиатура `>`, а для символа `<` — `<`.

Допустим, вы хотели напечатать «Элемент **<html>** очень важен» на своей странице. Используя ссылки на символы, вместо этого вы напечатаете:

```
Элемент &lt;html&gt; очень важен.
```

Еще один специальный символ, о котором вам нужно знать, — символ `&`. Если вы хотите, чтобы он был на вашей HTML-странице, используйте ссылку `&` вместо самого символа `&`.

Итак, как же насчет символов авторского права? И всех остальных подобных символов, в том числе иностранных?

Ссылки на основные специальные символы можно посмотреть здесь:

http://www.w3schools.com/tags/ref_entities.asp.

Более полный их список вы найдете здесь:

http://www.unicode.org/charts/.

Часто Задаваемые Вопросы

В: Здорово, я не догадывался, что браузер может отображать так много разных символов. На сайте www.unicode.org можно найти тонны различных символов и языков.

О: Будьте осторожны. Ваш браузер отобразит все эти символы только в том случае, если на вашем персональном компьютере установлены соответствующие шрифты. Поэтому, в то время как вы можете рассчитывать на корректное отображение в браузере основных символов с сайта www.w3schools.com, нет никакой гарантии, что отобразятся остальные символы из полного списка. Однако если вы кое-что знаете о своих пользователях, то должны предполагать, символы каких иностранных языков будут поддерживаться на их машинах.

В: Вы сказали, что & — специальный символ и мне нужно использовать ссылку на этот символ вместо него самого. Но чтобы напечатать эту ссылку, я должен указать символ &. Ведь, скажем, для ссылки на символ > я должен напечатать >?

О: Нет, нет! Причина того, что & — специальный символ, именно в том, что это первый знак любой ссылки на символ. Итак, это совершенно правильно — использовать & в ссылке на символ, только не отдельно. Просто не забывайте добавлять & каждый раз, когда печатаете ссылку на символ, а если вам на самом деле нужен знак &, то используйте вместо него ссылку.

В: Просматривая ссылки на символы на сайте www.w3cschools.com, я заметил, что каждая ссылка также имеет номер. Для чего он нужен?

О: Вы можете использовать либо ссылки на символы, либо числовые ссылки, например &#100 (они делают абсолютно одно и то же). Не для всех символов есть ссылки, в этих случаях единственное, что вы можете сделать, — это использовать их числовой код.



Взлом кода для вычисления месторасположения

Доктор Ивел в своих поисках путей к мировому господству поместил личную веб-страницу в Интернете, чтобы ее могли использовать лишь его сторонники. Вы только что получили перехваченный фрагмент HTML-кода, который, возможно, содержит ключ к месту его пребывания. Как эксперта в HTML, вас попросили взломать код и вычислить месторасположение доктора. Вот небольшой кусочек кода его домашней страницы:

```
В следующем месяце планируется собрание
членов группировки в моем подземном логове
в &#208;&epsilon;&tau;&#114;&ouml;&igrave;&tau;.
Приходите все.
```

Подсказка: посетите сайт http://www.w3schools.com/tags/ref_entities.asp и/или напечатайте HTML-код и посмотрите, что отобразит ваш браузер.

Коктейль из элементов

Каждый раз, когда вы захотите создать ссылку, вам понадобится элемент <a>.

<a>

Используйте этот элемент для коротких цитат, например «Быть или не быть» или «Где бы ты ни был, оставайся самим собой».

<q>

Просто дайте мне абзац, пожалуйста.

<p>

Используйте этот элемент, чтобы обозначить текст, который хотите особо выделить.

<address>

Этот элемент говорит браузеру, что его содержимое — это адрес, например ваша контактная информация.

Нужно отобразить список? Скажем, список ингредиентов рецепта или перечень заданий на сегодня? Используйте элемент <ul>.

Если вам нужен упорядоченный список, используйте элемент <ol>.

Используйте этот элемент, чтобы особым образом выделить текст.

Используйте этот элемент для отформатированного текста, когда хотите, чтобы браузер отобразил его именно так, как вы его напечатали.

<pre>

Пустой элемент, используемый для разрыва строки...

...еще один элемент для вставки горизонтальных линий, например для определения новой логической части текста.

<hr>

Используется для очень длинных цитат. Чего-то такого, что вы хотите пометить как длинную выдержку из книги.

<blockquote>

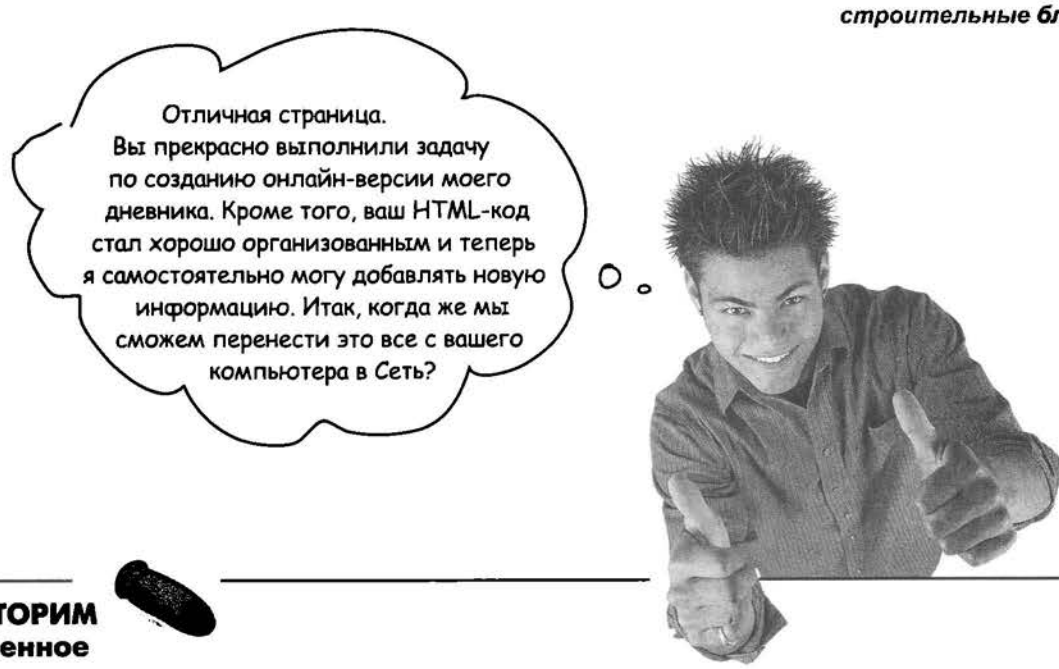
Элемент code для отображения кода из компьютерной программы.

<code>

Для пунктов списка, например: шоколад, горячий шоколад, шоколадный сироп...

Вот несколько элементов, с которыми вы уже знакомы, и парочка новых элементов.

Помните, что половина вашей работы с HTML — это эксперимент! Поэтому создайте парочку своих собственных файлов и попробуйте использовать это.



ПОВТОРИМ выученное

- Перед тем как набирать содержимое, составьте план структуры вашей веб-страницы. Начните с черновика, затем создайте план и только потом пишите HTML-код.
- Планирование страницы начинается с размещения больших блочных элементов, а затем уточняются строчные элементы.
- Помните: по возможности нужно использовать элементы, чтобы сказать браузеру, что означает ваше содержимое.
- Всегда применяйте те элементы, которые наиболее точно соответствуют структуре вашей страницы. Например, никогда не используйте абзац, когда вам нужен список.
- **<p>**, **<blockquote>**, **<ol>**, **<ul>** и **<li>** — это блочные элементы. Они всегда стоят обособленно и изображаются так, что перед их содержимым и после него всегда есть свободное пространство.
- **<q>**, **** и **<a>** — строчные элементы. Содержимое этих элементов сливается с остальным содержимым элемента, который их включает.
- Используйте элемент **
**, если хотите вставить свой собственный разрыв строки.
- **
** — это пустой элемент.
- В пустых элементах нет содержимого.
- Пустой элемент состоит только из одного тега.
- Вложенный элемент — это элемент, полностью находящийся внутри другого элемента. Если все ваши элементы вложены правильно, то все теги будут корректно согласованы.
- Для создания списка в HTML используйте комбинацию двух элементов: для упорядоченных списков применяйте **<ol>** и **<li>**; для неупорядоченных — **<ul>** и **<li>**. Когда браузер выводит упорядоченный список, он сам пронумеровывает его элементы.
- Вы можете задать свой собственный порядок нумерации элементов упорядоченного списка. Для этого используйте атрибут **start**. Чтобы поменять значения отдельных элементов, воспользуйтесь атрибутом **value**.
- Можете создать список, вложенный в другой список, используя элементы **<ol>** или **<ul>** внутри элемента **<li>**.
- Применяйте ссылки на символы для отображения специальных символов в HTML.



Решение упражнений

Здесь приводится исправленная часть HTML-кода с цитатой Лао-Цзы, где применяется элемент **<q>**. Вы тестировали свое решение?

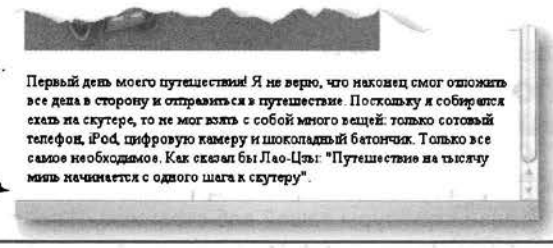
Это часть изменений...

```
<p>
  Первый день моего путешествия! Я не верю, что наконец
  смог отложить все дела в сторону и отправиться
  в путешествие. Поскольку я собирался ехать на скутере,
  то не мог взять с собой много вещей: только сотовый
  телефон, iPod, цифровую камеру и шоколадный батончик.
  Только все самое необходимое. Как сказал бы Лао-Цзы:
  <q>Путешествие на тысячу миль начинается с одного шага
  к скутеру</q>
</p>
```

Мы добавили открывающий тег <q> перед началом цитаты и закрывающий </q> сразу после нее

Заметьте, что мы убрали двойные кавычки.

А вот результаты теста...



Хорошо, особой разницы не видно, но не чувствуете ли вы, что так лучше?



Решение упражнений

Вот другой список из дневника Тони: сотовый телефон, iPod, цифровая камера и шоколадный батончик. Вы найдете его в записи от 14 июля. Это *маркированный* список элементов.

Внесите изменения в файл `journal.html`. Все выглядит так, как вы ожидаете?

```
<h2>2 Июня, 2005</h2>
<img src="segway1.jpg">
<p>
  Первый день моего путешествия! Я не верю, что наконец смог
  отложить все дела в сторону и отправиться в путешествие.
  Поскольку я собирался ехать на скутере, то не мог взять
  с собой много вещей: только
</p>
<ul>
    <li>сотовый телефон</li>
    <li>iPod </li>
    <li>цифровую камеру</li>
    <li>шоколадный батончик</li>
</ul>
<p>
  Только все самое необходимое.
  Как сказал бы Лао-Цзы: <q>Путешествие
  на тысячу миль начинается с одного шага к скутеру</q>
</p>
```

Сначала закончите предыдущий абзац.

Начните маркированный список.

Поместите каждый элемент списка в элемент <li>.

Закончите маркированный список.

Затем нужно начать новый абзац.



Поработайте браузером Решение

```
<html>
<head>
    <title>Топ-100</title>
<body>
<h1>Топ-100
<h2>Dark Side of the Moon</h2>
<h3>Pink Floyd</h3>
<p>
    У Луны нет темной стороны; на самом деле <q>она
вся темная.
</p></q>
<ul>
    <li>Speak to Me / Breathe</li>
    <li>On The Run</li>
    <li>Time</li>
    <li>The Great Gig in The Sky</li>
    <li>Money</li>
    <li>Us And Them</em>
    <li>Any Colour You Like</li>
    <li>Brain Damage</li>
    <li>Eclipse</li>
</ul>
</p>
<h2>XandY</h3>
<h3>Coldplay</h2>
<ol>
    <li>Square One
    <li>What If?
    <li>White Shadows
    <li>Fix You
    <li>Talk
    <li>XandY
    <li>Speed of Sound
    <li>A Message
    <li>Low
    <li>Hardest Part
    <li>Swallowed In The Sea
    <li>Twisted Logic
</ul>
</body>
</head>
```

Пропущен закрывающий тег </head>.

Пропущен закрывающий тег </h1>.

 и <q> неправильно вложены: тег должен идти после тега </q>.

Закрывающий тег стоит вместо закрывающего тега .

Здесь закрывающий тег , для которого нет открывающего тега .

В этом заголовке мы перепутали закрывающие теги </h2> и </h3>.

Мы начали список открывающим тегом <ol>, но поставили ему в соответствие закрывающий тег .

Мы пропустили все закрывающие теги .

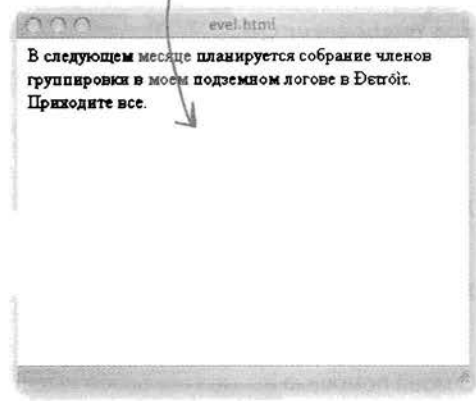
Этот закрывающий тег не соответствует открывающему тегу <ol>, с которого начинается список.

Вот наш пропущенный тег </head>; но нет закрывающего тега </html>.



Взлом кода для Вычисления Месторасположения

Вы можете найти ссылки на все эти символы в справочнике или просто набрать их. В любом случае ответ будет выглядеть так: Detroit!



В следующем месяце планируется собрание членов группировки в моем подземном логове в Ðετröìτ. Приходите все.



Решение упражнений



Группа HTML-элементов в маскарадных костюмах играет на вечеринке под названием «Кто я?». Они дадут вам ключ к разгадке, благодаря которому вы попытаетесь угадать, кто они на самом деле. Предполагается, что они все время говорят о себе правду. Опознайте участников и заполните правые столбцы бланка. Кроме того, напишите для каждого участника игры, блочный он или строчный.

Сегодняшние участники:

все очаровательные HTML-элементы, которые вы уже встречали раньше, должны быть разоблачены!

	Имя	Строчный или блочный?
Я заголовок первого уровня.	h1	Блочный
Я готов сослаться на другую страницу.	a	Строчный
С моей помощью вы можете особо выделить текст.	em	Строчный
Я список, но я не расставляю свои элементы по порядку.	ul	Блочный
Я элемент, живущий внутри списка.	br	
Я самый настоящий прерыватель строк.	li	Блочный
Я содержу свои элементы в порядке.	ol	Блочный
Я знаю все об изображениях.	img	Строчный
Со мной вы можете выделить цитату внутри абзаца.	q	Строчный
Со мной вы можете выделить цитату в отдельный блок.	blockquote	Блочный

Поставлены в тупик? Элемент
 не относится ни к блочным, ни к строчным элементам. Он создает разрыв строки, но не изображается, как все остальные блочные элементы, со свободным пространством перед содержимым и после него.

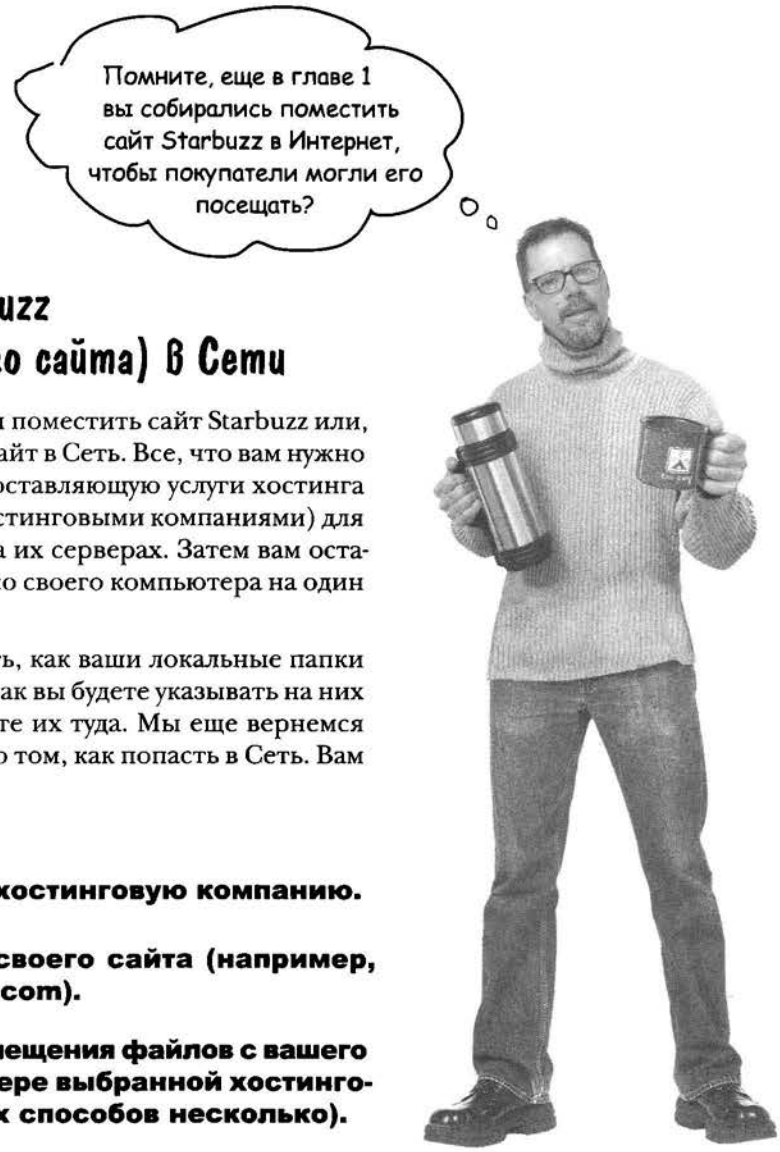
Мы пока подробно об этом не говорили, но элемент <img> действительно строчный. Сейчас просто запомните это, а подробно к данному вопросу мы вернемся в главе 5.

Путешествие в Webville



Веб-страницы предназначены для того, чтобы располагаться и обслуживаться в Интернете. До сих пор вы создавали веб-страницы, которые «жили» только в вашем собственном компьютере. Вы также создавали ссылки только на те страницы, которые хранятся на вашем компьютере. Мы вот-вот изменим это навсегда. В этой главе мы научим вас размещать веб-страницы в Интернете, где все ваши родные, друзья и покупатели действительно смогут их увидеть. Мы также раскроем тайну создания ссылок на другие страницы, взломав код h, t, t, p, :, /, /, w, w, w. Итак, собирайте свои вещи, следующая остановка — Webville.

ВНИМАНИЕ: попав в Webville, вы уже никогда не вернетесь. Пишите нам письма.



Размещение сайта Starbuzz (или вашего собственного сайта) в Сети

Вы уже очень близки к тому, чтобы поместить сайт Starbuzz или, что еще лучше, ваш собственный сайт в Сеть. Все, что вам нужно сделать, – найти компанию, предоставляющую услуги хостинга (условимся называть их просто хостинговыми компаниями) для размещения ваших веб-страниц на их серверах. Затем вам останется скопировать эти страницы со своего компьютера на один из их серверов.

Конечно же, необходимо пояснить, как ваши локальные папки отобразятся в каталоге сервера и как вы будете указывать на них браузеру после того, как поместите их туда. Мы еще вернемся ко всему этому. А пока поговорим о том, как попасть в Сеть. Вам нужно сделать следующее.

1. **Выберите для себя хостинговую компанию.**
2. **Выберите имя для своего сайта (например, www.starbuzzcoffee.com).**
3. **Найдите способ размещения файлов с вашего компьютера на сервере выбранной хостинговой компании (таких способов несколько).**
4. **Расскажите родным, друзьям, покупателям, где найти ваш новый сайт, и наслаждайтесь.**

Мы подробно разберемся с каждым из этих пунктов, и даже если вы не планируете создавать свой сайт *прямо сейчас*, все равно ознакомьтесь с ними, потому что мы расскажем кое-что важное, что так или иначе понадобится вам позже. Итак, приготовьтесь. Сейчас мы ненадолго оставим изучение HTML.



Поиск хостинговой компании

Чтобы поместить страницы в Интернет, вам необходимо найти сервер, который *постоянно* «живет» в Сети. Лучше всего найти хостинговую компанию, которая позаботится о поддержке работы сервера. Не беспокойтесь, найти такую компанию довольно просто и совсем не дорого.

Какую компанию? Ну, мы бы хотели, чтобы вы зарегистрировались в хостинговой компании Head First Hip Web Hosting, Inc., но, к сожалению, таковой не существует. Поэтому вам придется позаботиться об этом самостоятельно. Процесс поиска хостинговой компании совсем не сложный, он сродни выбору компании, предоставляющей услуги кабельного телевидения: следует учесть множество свойств и тарифных планов. Вам нужно подобрать компанию, которая подходит именно вам по цене и качеству своих услуг.

Есть хорошая новость: вы можете приступить к работе, заплатив сначала совсем небольшую сумму. Затем при необходимости вы всегда сможете расширить пакет услуг. Мы не будем советовать вам выбрать какого-то определенного провайдера, но можем рассказать, на что обратить внимание при его выборе



Расслабьтесь

Вам не обязательно помещать свои страницы в Сеть, чтобы продолжить обучение по этой книге.

Хотя намного интереснее, если страницы расположены *в Сети*, вы по-прежнему можете продолжать учить язык HTML по этой книге, располагая все файлы на своем локальном компьютере.

Если же вы хотите разместить сайт в Сети, продолжайте читать дальше эту главу, чтобы выяснить, как это организовать.



Советы по выбору хостинговой компании

Мы не сможем рассказать вам все, что нужно знать о выборе хостинговой компании (в конце концов, эта книга о HTML и CSS), но постараемся указать, в каком направлении двигаться. Вот несколько рекомендаций, которые нужно учитывать при выборе.

- *Техническая поддержка:* имеет ли хостинговая компания хорошую систему по обработке возникающих у вас технических вопросов? В лучших компаниях на ваши вопросы ответят быстро по телефону либо по электронной почте.
- *Передача данных:* существует ограниченное количество страниц и данных, которое хостинговая компания позволит вам отправлять вашим посетителям в течение месяца. Большинство компаний в своих основных планах предлагают приемлемое количество передаваемых данных для небольших сайтов. Если же вы создаете сайт, у которого, как предполагается, будет много посетителей, то вам нужно обратить на это особое внимание.
- *Резервные копии:* регулярно ли хостинговая компания делает резервные копии ваших страниц и данных, которые можно будет восстановить, если на сервере произойдет аппаратный сбой?
- *Доменные имена:* учитывает ли хостинговая компания имя домена при ценообразовании? Читайте подробности на следующей странице.
- *Надежность:* большинство хостинговых компаний заявляют, что они осуществляют поддержку сайтов до 99 % общего времени и более.
- *Дополнительные возможности:* входят ли в ваш пакет какие-нибудь дополнительные возможности, такие как адреса электронной почты, форумы или поддержка сценарных языков (что-то, что в будущем может вам пригодиться).

Привет, мое доменное имя...

Даже если вы никогда не слышали о *доменных именах*, вы видели и использовали огромное их количество. Вы наверняка знаете `google.com`, `yahoo.com`, `amazon.com`, `disney.com` и, может быть, еще парочку, о которых не хотите упоминать.

Итак, что такое доменное имя? Это просто уникальное имя, используемое для определения места для вашего сайта. Рассмотрим пример:

Это часть доменного имени.

www.starbuzzcoffee.com

Эта часть имени — название определенного сервера в домене.

Существуют различные «окончания» доменных имен, которые используются для различных целей: .com, .org, .gov, .edu; а также указывают принадлежность к различным странам: .co.uk, .co.jp и т. д. При выборе домена отдайте предпочтение наиболее подходящему для вас.

Есть несколько причин, по которым вы должны заботиться о доменных именах. Если вам нужно уникальное имя для сайта, то вам понадобится собственное доменное имя. Доменные имена также используются, чтобы вы могли сослаться на свои страницы с других сайтов (мы вернемся к этому через несколько страниц).



И еще кое-что, что вам обязательно нужно знать. Доменные имена контролируются централизованным учреждением (называемым ICAAN). Это позволяет удостовериться, что только один человек в данный момент использует определенное доменное имя. Кроме того (вы догадывались, что так будет), вы ежегодно платите небольшой регистрационный взнос для сохранения за собой выбранного доменного имени.

Как можно получить доменное имя

Самый простой способ — предоставить это своей хостинговой компании. Они часто предлагают регистрацию доменного имени в одном из своих пакетов комплексных услуг.

К сожалению, как и с поиском хостинговой компании, нам придется предоставить вам право самостоятельно выбрать и зарегистрировать доменное имя. После того как вы найдете хостинговую компанию, вам, скорее всего, будет очень легко это сделать.

В: Почему это называется доменным именем, а не именем сайта?

О: Потому что это разные вещи. Если говорить о www.starbuzzcoffee.com, то это имя сайта, но лишь часть starbuzzcoffee.com является доменным именем. Вы также можете создавать другие сайты, использующие это же доменное имя, например corporate.starbuzzcoffee.com или employee.starbuzzcoffee.com. Итак, можно применять одно доменное имя для нескольких сайтов.

В: Если бы мне нужно было получить доменное имя для Starbuzz, разве я не захотел бы выбрать www.starbuzzcoffee.com? Кажется, все используют www в начале имени сайта.

О: Не нужно путать доменное имя с именем сайта: starbuzzcoffee.com — это доменное имя, в то время как www.starbuzzcoffee.com — название сайта. Покупка домена — это как покупка участка земли, скажем, 100mainstreet.com. На этой земле вы можете построить столько участков земли, сколько пожелаете, например: home.100mainstreet.com, toolshed.100mainstreet.com и outhouse.100mainstreet.com.
Итак, www.starbuzzcoffee.com — это всего лишь один сайт в домене starbuzzcoffee.com.

В: Что же все-таки полезного в том, чтобы иметь доменное имя? Действительно ли оно мне необходимо? В моей хостинговой компании сказали, что я могу использовать их имя: www.dirtcheaphosting.com.

О: Если это вас устраивает, то нет ничего плохого в том, чтобы использовать их имя. Но (и это важное НО) в этом есть большое неудобство: если компания когда-нибудь прекратит свою деятельность или вы захотите перейти в другую компанию, то все, кто знал о вашем сайте, больше не смогут быстро его найти. С другой стороны, если у вас есть собственное доменное имя, вы просто можете взять его с собой в новую хостинговую компанию (и пользователи никогда даже не догадаются о том, что вы поменяли компанию).

В: Доменные имена должны быть уникальными, но что, если кто-то уже использует то имя, которое я хочу выбрать. Как я могу это узнать?

О: Хороший вопрос. Большинство компаний, которые предоставляют услугу регистрации доменного имени, позволяют проверить, не используется ли оно кем-то еще (как и при поиске номерных знаков на автомобиль).

Это упражнение, которое вам действительно нужно выполнить. Кроме того, сделать это вам придется самостоятельно. Мы бы, конечно же, были рады помочь каждому из вас, но вопросов было бы слишком много.



Попробуйте сделать это дома

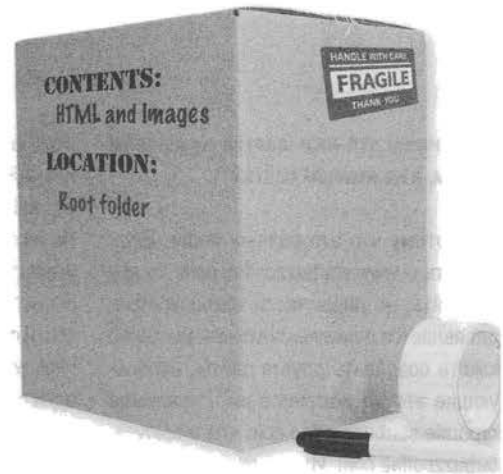
Настало время найти хостинговую компанию и получить доменное имя для вашего сайта. Помните, что вы можете посетить лабораторию Head First для получения нескольких советов и ссылок. Не забывайте также, что вы легко можете продолжить обучение по книге без всего этого (даже несмотря на то, что это действительно полезно знать!).

Моя хостинговая компания: _____

Мое доменное имя: _____

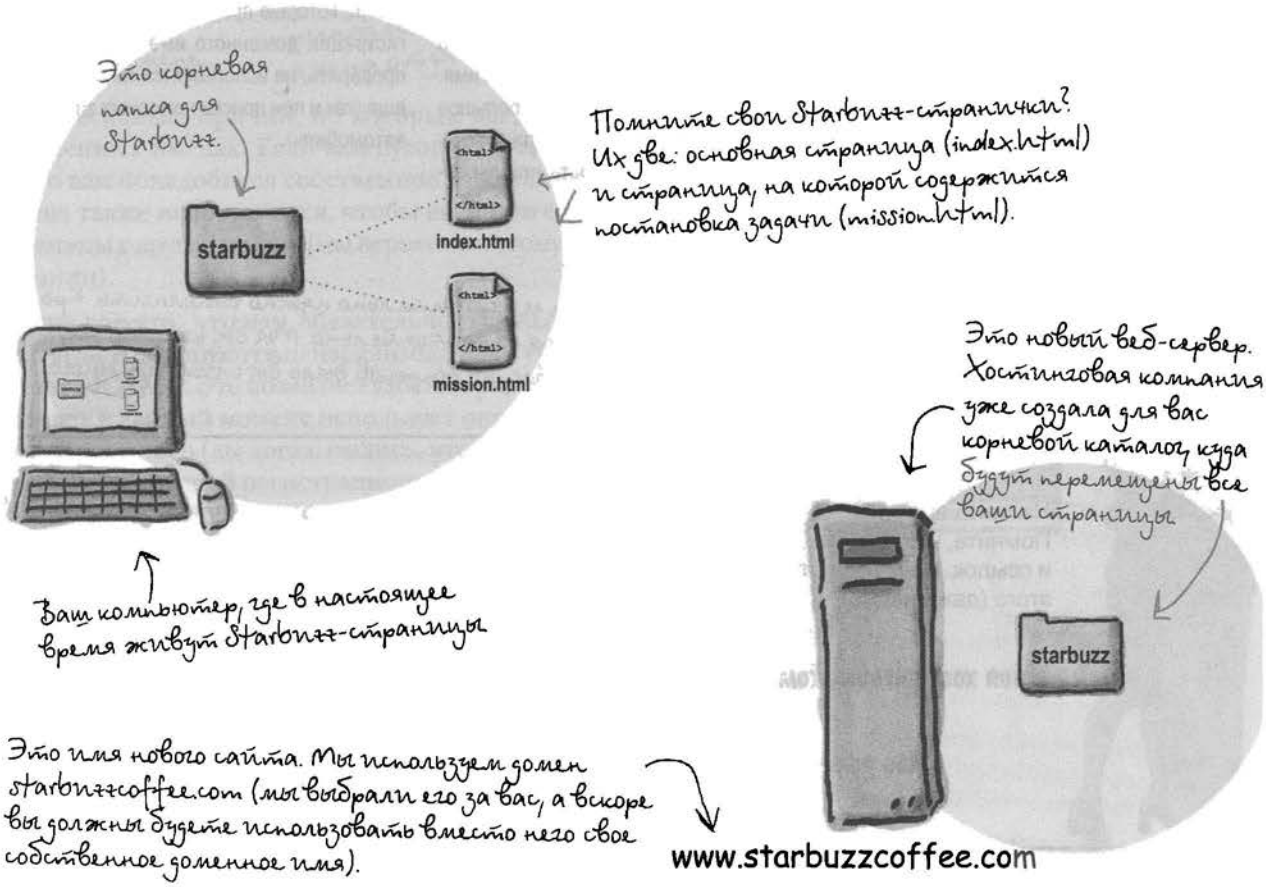


Заселение



Поздравляем! Вы подобрали хостинговую компанию, выбрали доменное имя и получили сервер, готовый разместить ваши веб-страницы (даже если это не так, продолжайте читать, потому что здесь приводится важная информация).

Что дальше? Конечно же, настало время заселяться. Итак, снимайте вывеску «Продается» и собирайте все свои файлы вместе: мы будем заселять их в новый сервер. Как и при любом переезде, основная цель – переместить ваши вещи, например, из кухни в старой квартире на кухню в новой. А в Сети мы заботимся о перемещении информации из вашей собственной корневой папки в корневую папку на веб-сервере. Давайте вернемся к кафе Starbuzz и подробно опишем всю процедуру переезда. Рассмотрим, как все выглядит на данный момент:



Часто Задаваемые Вопросы

В: Минутку, что же все-таки такое «корневая папка»?

О: До сих пор корневой папкой была просто папка наивысшего уровня, в которой хранились файлы для ваших страниц. На веб-сервере значение корневой папки увеличивается, потому что все, что в ней содержится, будет доступно в Сети.

В: Кажется, моя хостинговая компания назвала мою корневую папку mydomain_com. Это плохо?

О: Ничуть. Хостинговые компании могут называть корневые папки как угодно. Главное, что вы знаете, где расположена ваша корневая папка на сервере, и можете скопировать в нее свои файлы (мы вернемся к этому чуть позже).

В: Позвольте мне убедиться, что я правильно все понял. Мы складывали все страницы нашего сайта в одну папку, которую называли корневой. Сейчас мы собираемся скопировать все это в корневую папку сервера?

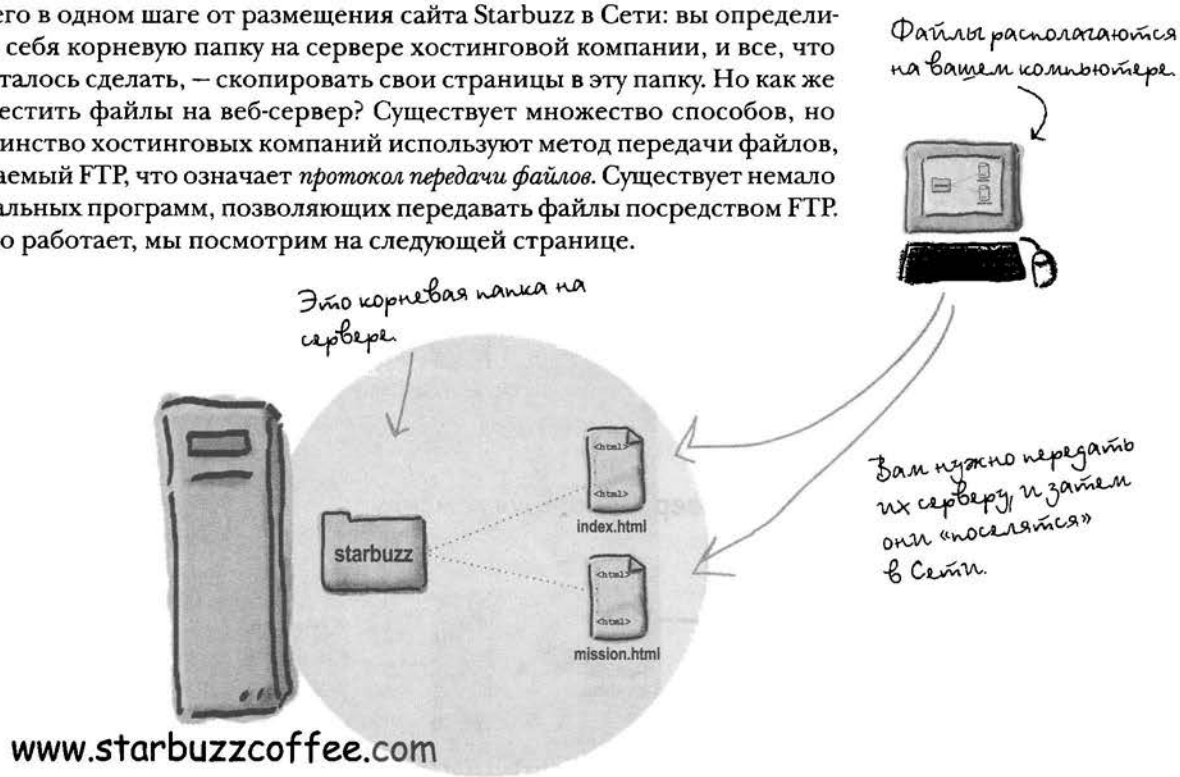
О: Точно. Вам нужно взять все страницы с вашего компьютера и поместить их в корневую папку своего сайта, которая находится на сервере вашей хостинговой компании.

В: А как насчет вложенных папок, таких как папка images? Их тоже нужно копировать?

О: Да, вам нужно скопировать все страницы, файлы и папки из собственной корневой папки в корневую папку на сервере. Итак, если на вашем компьютере была папка images, то вам нужно, чтобы такая же появилась и на сервере.

Перемещение файлов в корневую папку

Вы всего в одном шаге от размещения сайта Starbuzz в Сети: вы определили для себя корневую папку на сервере хостинговой компании, и все, что вам осталось сделать, – скопировать свои страницы в эту папку. Но как же переместить файлы на веб-сервер? Существует множество способов, но большинство хостинговых компаний используют метод передачи файлов, называемый FTP, что означает *протокол передачи файлов*. Существует немало специальных программ, позволяющих передавать файлы посредством FTP. Как это работает, мы посмотрим на следующей странице.



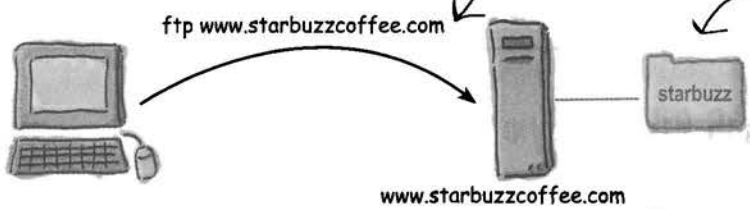
Столько информации об FTP, сколько может поместиться на две страницы

Хотя эта книга о HTML и CSS, мы хотим представить вам небольшой путеводитель по использованию FTP для размещения файлов в Сети. Примите во внимание, что ваша хостинговая компания может дать вам пару советов о том, как наилучшим образом передать файлы на их сервер (а поскольку вы им платите, не пренебрегайте их помощью). Через несколько страниц мы закончим наше отступление, снова вернемся к изучению HTML и CSS и уже не будем ни на что отвлекаться до конца книги (мы обещаем).

Мы предполагаем, что вы уже нашли FTP-клиент. Одни из них – с управлением из командной строки, другие имеют полностью графический интерфейс, а третьи встроены в программы, например в Dreamweaver и GoLive. Все они используют одни и те же команды, но в некоторых программах вам нужно вводить их самостоятельно, в то время как в других вы просто используете графический интерфейс. Рассмотрим по порядку, как работает FTP.

Для подключения вам понадобятся имя пользователя и пароль, установленные вашей хостинговой компанией.

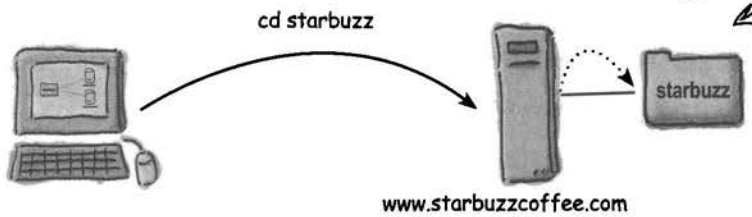
1 Подключитесь к вашему серверу с помощью FTP.



Слова «папка» и «каталог» взаимозаменяемы. Большинство FTP-клиентов используют слово «каталог».

Другими словами, убедитесь, что вы находитесь в папке starbuzz на сервере перед тем, как отправлять туда свои файлы.

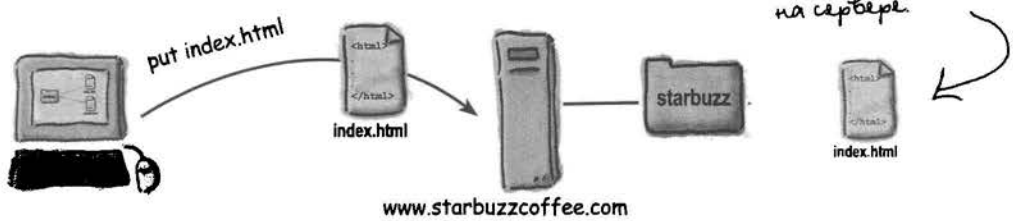
2 Используйте команду cd, чтобы поменять текущий каталог на тот, в который вы хотите переслать файлы.



Устанавливает текущим каталог starbuzz.

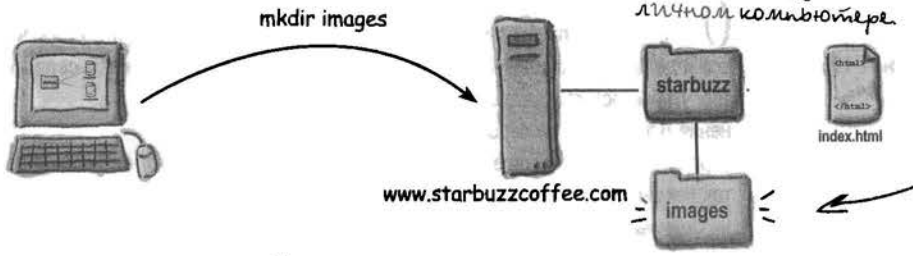
3 Передайте ваши файлы на сервер, используя команду put.

Пересылает копию файла index.html в текущий каталог на сервере.



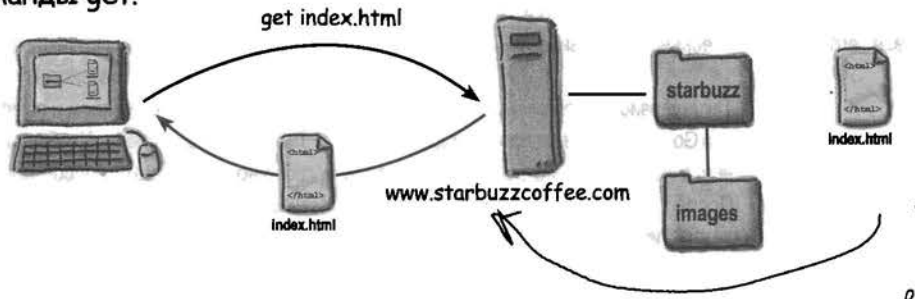
4 Вы также можете создать новый каталог на сервере, используя команду mkdir.

Это то же, что и создание новой папки, только вы создаете ее на сервере, а не на личном компьютере.



Создает новый каталог под названием images, находящийся внутри каталога starbuzz на сервере.

5 Вы можете извлечь файлы с помощью команды get.



Передает копию файла с сервера обратно на ваш компьютер.

Давайте соберем все вместе. Далее приведен пример работы с FTP-клиентом с управлением из командной строки.

В большинстве FTP-клиентов используется намного более удобный графический интерфейс, так что вы спокойно можете не читать это, если работаете с одним из них.

```
File Edit Window Help Jam
%ftp www.starbuzzcoffee.com
Connected to www.starbuzzcoffee.com
Name: headfirst
Password:******
230 User headfirst logged in.
ftp> dir
drwx------  4096 Sep  515:07 starbuzz
ftp> cd starbuzz
CWD command successful
ftp> put index.html
Transfer complete.
ftp> dir
-rw-------  1022 Sep  515:07 index.html
ftp> mkdir images
Directory successfully created
ftp> cd images
CWD command successful
ftp> bye
```

Подключение и идентификация.

Получаем содержимое текущего каталога.

Один каталог называется starbuzz.

Делаем каталог starbuzz текущим.

Пересылаем сюда файл index.html.

Смотрим содержимое каталога и видим в нем файл index.html.

Создаем каталог для изображений, а затем выходим из программы с помощью команды bye.

FTP-команды

Неважно, печатаете вы FTP-команды в программе с управлением из командной строки или используете FTP-клиент с графическим интерфейсом, и там и там абсолютно одинаковые команды и операции, которые вы можете выполнять.

- `dir`: выводит содержимое текущего каталога.
- `cd`: изменяет текущий каталог. Здесь символы «..» также обозначают переход вверх на один каталог.
- `pwd`: показывает, в какой директории вы сейчас находитесь.
- `put <имя_файла>`: пересылает указанный файл на сервер.
- `get <имя_файла>`: извлекает указанный файл из сервера назад на ваш компьютер.

Часто Задаваемые Вопросы

В: Моя хостинговая компания сказала мне использовать SFTP, а не FTP. В чем разница?

О: SFTP, или Secure File Transfer Protocol (протокол безопасной пересылки данных), — это более безопасная версия FTP, которая работает примерно так же. Только перед приобретением SFTP убедитесь, что выбранный FTP-клиент также поддерживает его.

В: Итак, я должен редактировать файлы на своем компьютере, а затем пересылать их на сервер каждый раз, когда хочу обновить свой сайт?

О: Да, для небольших сайтов именно так все и делается. Используйте свой компьютер, чтобы протестировать внесенные изменения, и перед тем, как пересылать файлы на сервер, убедитесь, что все работает именно так, как вы хотите. Для больших сайтов организации часто специально создают тестовый и реальный сайты, чтобы была возможность предварительного просмотра изменений на тестовом сайте, перед тем как они будут внесены на реальный.

Такие сервисные программы, как Dreamweaver или GoLive, позволяют протестировать изменения на вашем собственном компьютере. После того как вы сохраните файлы, они автоматически будут переданы на сайт.

В: Могу ли я редактировать файлы непосредственно на веб-сервере?

О: Обычно это не очень правильно, потому что ваши посетители могут увидеть все изменения и ошибки еще до того, как у вас появится время самому их заметить и исправить.

Однако некоторые хостинговые компании позволят вам войти в систему под своим регистрационным именем и поменять файлы прямо на сервере. Чтобы так сделать, обычно нужно знать способ приглашения на ввод команды MS-DOS или Linux, в зависимости от того, в какой операционной системе работает ваш сервер.



Популярные FTP-клиенты

Вот список наиболее популярных FTP-клиентов для Mac и Windows.

Для Mac OS X:

- Fetch (http://fetchsoftworks.com/) — один из наиболее популярных FTP-клиентов для Mac. $
- Transmit (http://www.panic.com/transmit/) $
- Cyberduck (http://cyberduck.ch/) FREE

Для Windows:

- Smart FTP (http://www.smartftp.com/download/) $
- WS_FTP (http://www.ipswitch.com/products/file-transfer.asp). Основная версия БЕСПЛАТНАЯ, $ для профессиональной версии

Большинство FTP-клиентов имеют пробную версию, которую можно скачать. Таким образом, перед тем, как купить программу, вы можете попробовать с ней поработать.



Попробуйте сделать это дома

Это еще одно домашнее задание (после того как сделаете его, проверьте каждый пункт).

- Убедитесь, что знаете, где расположена ваша корневая папка на сервере вашей хостинговой компании.
- Выберите наилучший способ (и наилучшую сервисную программу) для пересылки файлов с вашего компьютера на сервер.
- Теперь перешлите файлы Starbuzz index.html и mission.html в корневую папку на сервере.

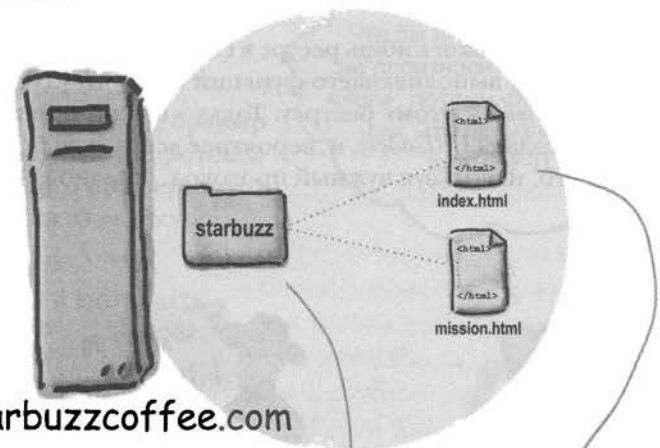


Конец веб-тура

Вернемся к делу...

Вот и закончилась объездная дорога, и мы снова выехали на главную веб-магистраль. К этому моменту у вас должны быть две Starbuzz-страницы: `index.html` и `mission.html`, находящиеся в корневой папке на сервере (или по крайней мере вы уже знаете, как их туда поместить).

Согласитесь, было бы здорово, если бы после всей проделанной работы браузер нашел эти страницы в Интернете и отобразил их в своем окне. Давайте выясним, какой адрес нужно ввести для этого в адресной строке браузера...



Адреса всех веб-страниц начинаются так, верно? Через минутку мы поясним, что означает http.

```
http://  www.starbuzzcoffee.com  /  index.html
```

Это имя сайта.

Для указания корневой папки мы используем символ «/».

Это имя файла для конкретной страницы.

Дальше >

Главная улица, ~~США~~ URL

Вы, скорее всего, слышали выражение «h, t, t, p, двоеточие, слэш, слэш» огромное количество раз. Однако что же оно означает? Во-первых, веб-адреса, которые вы вводите в браузере, называются *URL-адресами*, или унифицированными указателями информационных ресурсов.

Если бы мы решали, то назвали бы их просто веб-адресами, но нас никто не спрашивал, поэтому мы согласимся с предлагаемым названием. Рассмотрим, как расшифровывается URL:

```
http://www.starbuzzcoffee.com/index.html
```

Первая часть URL говорит, какой протокол должен быть использован для нахождения ресурса.

Вторая часть — имя сайта. На данный момент вы уже все об этом знаете.

Третья часть — абсолютный путь доступа к ресурсу из корневой папки.

Чтобы найти какой-нибудь ресурс в Сети, вы должны знать имя сервера, выполняющего функции главного узла, и *абсолютный путь* к этому ресурсу. Только в этом случае вы можете создать *URL-адрес*, и, вероятнее всего, ваш браузер найдет его, используя нужный протокол, обычно HTTP.



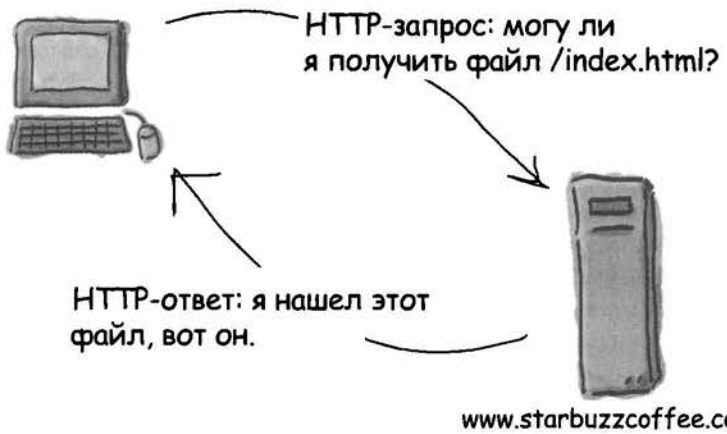
Унифицированный указатель ресурса (URL) — это сетевой адрес, который может быть использован для размещения в Сети какой-либо информации, включая HTML-страницы, аудио, видео и многие другие формы веб-содержимого.

Кроме того, для точного определения месторасположения ресурса URL включает протокол, который нужно использовать, чтобы найти этот ресурс.

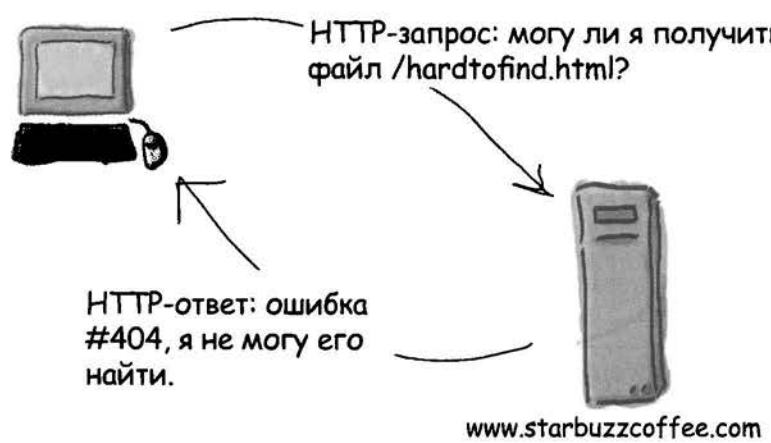
Что такое HTTP-протокол

HTTP известен как *протокол передачи гипертекстовых файлов*. Другими словами, это согласованный метод (протокол) для передачи гипертекстовых документов по Сети. В то время как гипертекстовые документы – это обычно HTML-страницы, протокол может быть использован для передачи изображений или файлов любых других типов, которые могут понадобиться веб-странице.

HTTP – это простой запросно-ответный протокол. Рассмотрим, как он работает.



Итак, каждый раз, когда вы вводите URL в адресной строке, браузер запрашивает у сервера соответствующий ресурс, используя протокол HTTP. Если сервер находит ресурс, он возвращает его браузеру и тот отображает его. Что же случается, если сервер не находит ресурс?



Если ресурс не может быть найден, то вы получите широко известную ошибку 404, которую сервер возвращает вашему браузеру.

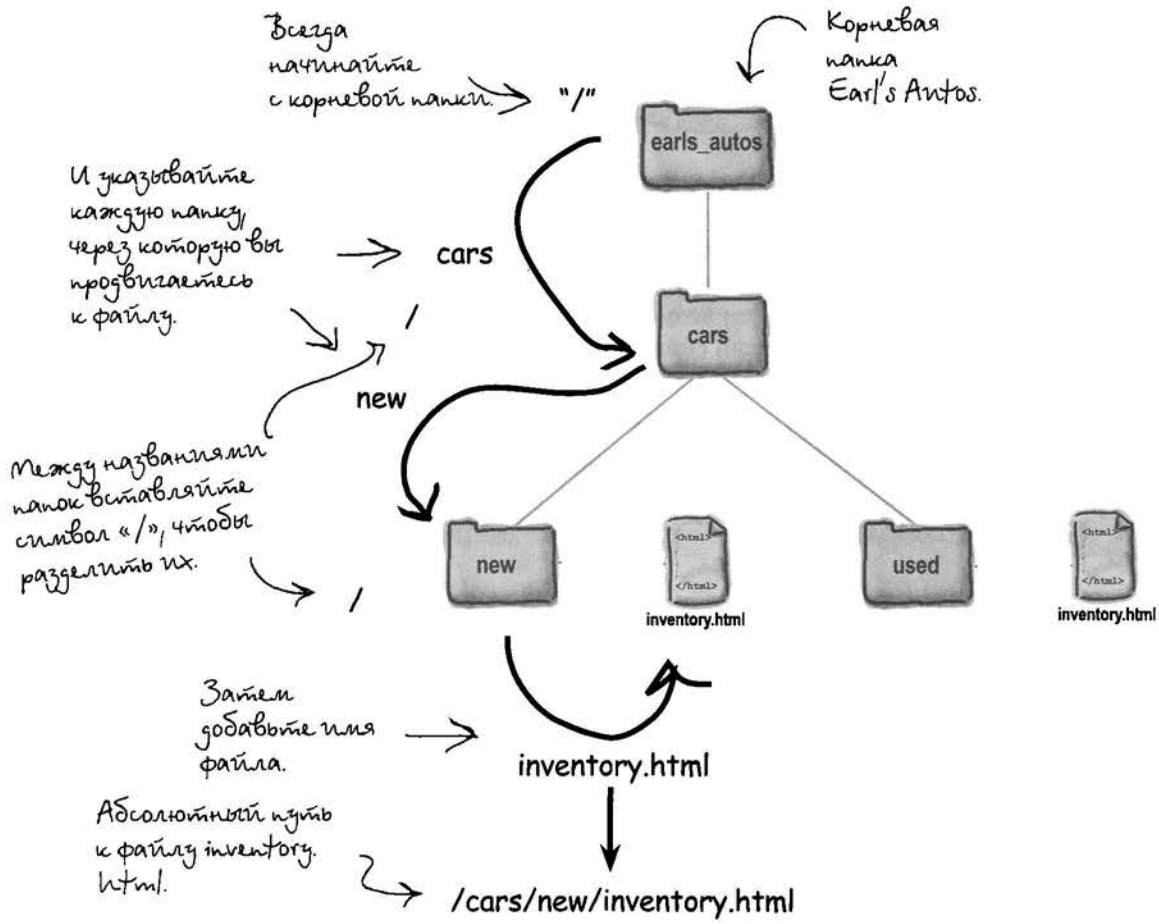


Что такое абсолютный путь

Последний раз о путях мы говорили, когда писали HTML-код для создания ссылок и использовали элемент **<a>**. Пути, с которыми мы хотим разобраться сейчас, называются абсолютными. Это последняя часть URL-адреса, которая идет после протокола (HTTP) и имени сайта (www.starbuzzcoffee.com).

Абсолютный путь говорит серверу, как попасть из вашей корневой папки к конкретной странице или файлу. Возьмем, к примеру, сайт Earl's Autos об автомобилях Эрла. Допустим, вы хотите проверить, не появился ли в каталоге новых товаров Мини Купер. Чтобы это сделать, нужно узнать абсолютный путь к файлу `inventory.html`, который находится в папке new. Все, что для этого следует сделать, — просмотреть все папки, начиная с корневой, чтобы найти папку new, в которой расположен файл `inventory.html`. Путь будет состоять из всех папок, через которые вы спускались, чтобы добраться до new.

Это выглядит так: корневая папка (мы обозначили ее символом «/»), cars, new и, наконец, сам файл `inventory.html`. Посмотрим, как соединить это вместе.



Часто Задаваемые Вопросы

В: Что такого важного в абсолютных путях?

О: Абсолютный путь — это то, что нужно серверу для определения местоположения требуемого файла. Если у сервера нет абсолютного пути, он не будет знать, где искать файл.

В: Мне кажется, что по-отдельности я все понимаю (протоколы, серверы, сайты и абсолютные пути), но мне сложно связать все это воедино.

О: Если вы соедините все эти понятия, то получите URL-адрес и благодаря ему сможете просить браузер извлечь страницу (или другие типы ресурсов) из Сети. Как? Протокол говорит браузеру, какой метод он должен использовать для поиска ресурса (в большинстве случаев это HTTP). Название сайта (которое состоит из названия сервера и доменного имени) говорит браузеру, с какого компьютера в Интернете нужно взять ресурс. А абсолютный путь говорит серверу, какая именно страница нужна.

В: Мы уже учились использовать относительный путь в атрибуте href элемента <a>. Как же сервер может найти страницы по этим ссылкам, если указанный в них путь не абсолютный?

О: О, отличный вопрос. Когда вы щелкаете кнопкой мыши на относительных ссылках, браузер самостоятельно преобразует относительный путь в абсолютный, используя указанный в ссылке путь и относительный путь к странице, на которой эта ссылка расположена. Итак, благодаря вашему браузеру все, что видит веб-сервер, — это абсолютные пути.

В: Если я стану указывать абсолютные пути в своем HTML-коде, облегчит ли это работу браузеру?

О: О, еще один хороший вопрос, но пока оставим его без ответа и вскоре к нему вернемся.

Возьми в руку карандаш

Вы ждали достаточно долго. Пришло время протестировать ваш URL-адрес. Прежде чем это сделать, заполните бланк (см. ниже) и введите свой URL (чего вы еще не делали). Если у вас возникнут какие-либо проблемы, то самое время обратиться в хостинговую компанию, чтобы во всем разобраться. Если же вы так и не воспользовались услугами хостинговой компании, все равно заполните бланк для сайта www.starbuzzcoffee.com и введите URL в своем браузере.

_____ :// _____ _____

протокол **имя сайта** **абсолютный путь**

Я бы хотела, чтобы мои посетители могли просто набирать «http://www.starbuzzcoffee.com», не дописывая «index.html». Есть ли способ это сделать?



Помните, когда мы говорили о веб-серверах или FTP, мы обычно использовали термин «каталог» вместо термина «папка». Но на самом деле это одно и то же.

Да, есть. Один из вопросов, который мы не обсудили, – что будет, если браузер запрашивает у веб-сервера целый каталог вместо отдельного файла. Например, браузер может запросить:

http://www.starbuzzcoffee.com/images/

Каталог images в корневом каталоге.

или

http://www.starbuzzcoffee.com/

Сам корневой каталог.

Когда сервер получает запрос такого типа, он пытается определить файл, выводимый *по умолчанию* в этом каталоге. Обычно такой файл называется `index.html` или `default.htm`. Если сервер находит какой-либо из этих файлов, он возвращает его браузеру для отображения.

Однако вам необходимо выяснить, как ваша хостинговая компания требует, чтобы вы называли свой файл, выдаваемый по умолчанию. Это зависит от того, какой тип серверов она использует.

Итак, чтобы обратиться к файлу в корневом (или любом другом) каталоге, не используя его имени, нужно просто назвать его `index.html` или `default.htm`.

Но я спрашивала про «http://www.starbuzzcoffee.com», что выглядит немного иначе. У него в конце не стоит символ «/».



Уф, вы правы. Когда сервер получает такой запрос, как ваш, без символа «/» в конце, и существует каталог с таким названием, то сервер сам добавит последний слэш.

Итак, если сервер получает запрос на:

http://www.starbuzzcoffee.com

он поменяет его на:

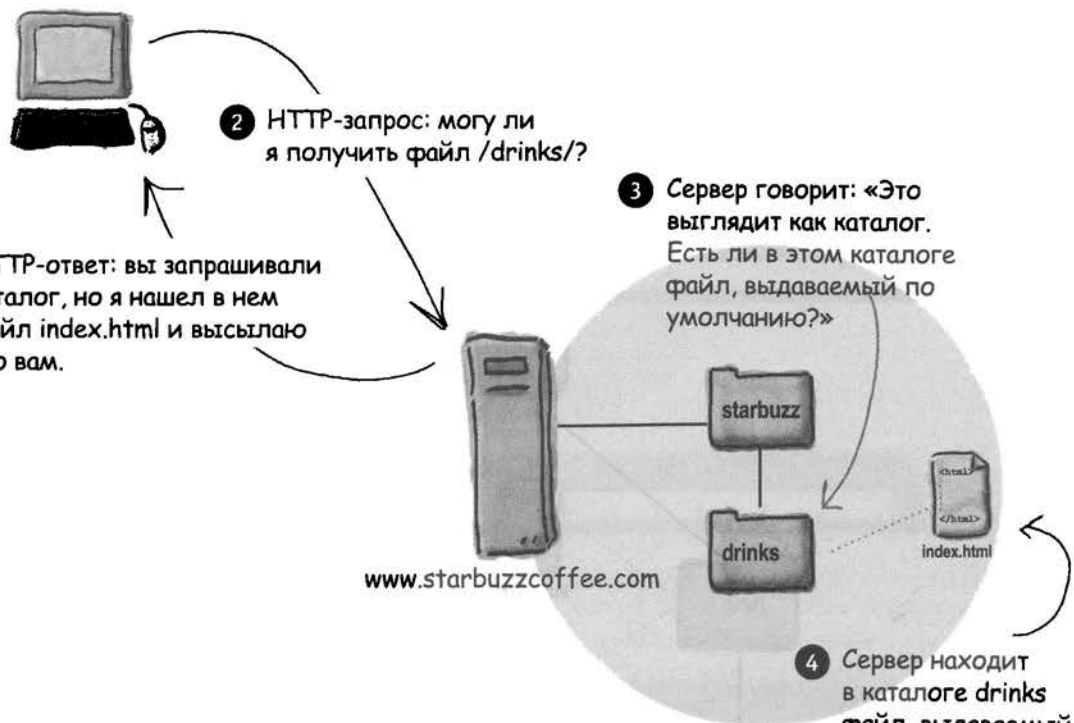
http://www.starbuzzcoffee.com/

После этого сервер начнет искать файл, выдаваемый по умолчанию, и в конце концов вернет файл, как если бы вы изначально ввели:

http://www.starbuzzcoffee.com/index.html

Как работают страницы, выдаваемые по умолчанию





В: Итак, все, кто зайдет на мой сайт с URL http://www.mysite.com, увидят мою страницу index.html?

О: Именно. Или страницу default.htm, в зависимости от того, какой тип веб-сервера использует ваша хостинговая компания. (Обратите внимание, что в имени default.htm в конце обычно не ставится символ «l». Это особенность веб-сервера Microsoft.)

Часто Задаваемые Вопросы

Существуют также другие имена файлов, выдаваемых по умолчанию, например, index.php. С ними вы столкнетесь, если начнете писать сценарии для создания страниц. Это тема не нашей книги, но это не означает, что вы не будете встречаться с таким в будущем.

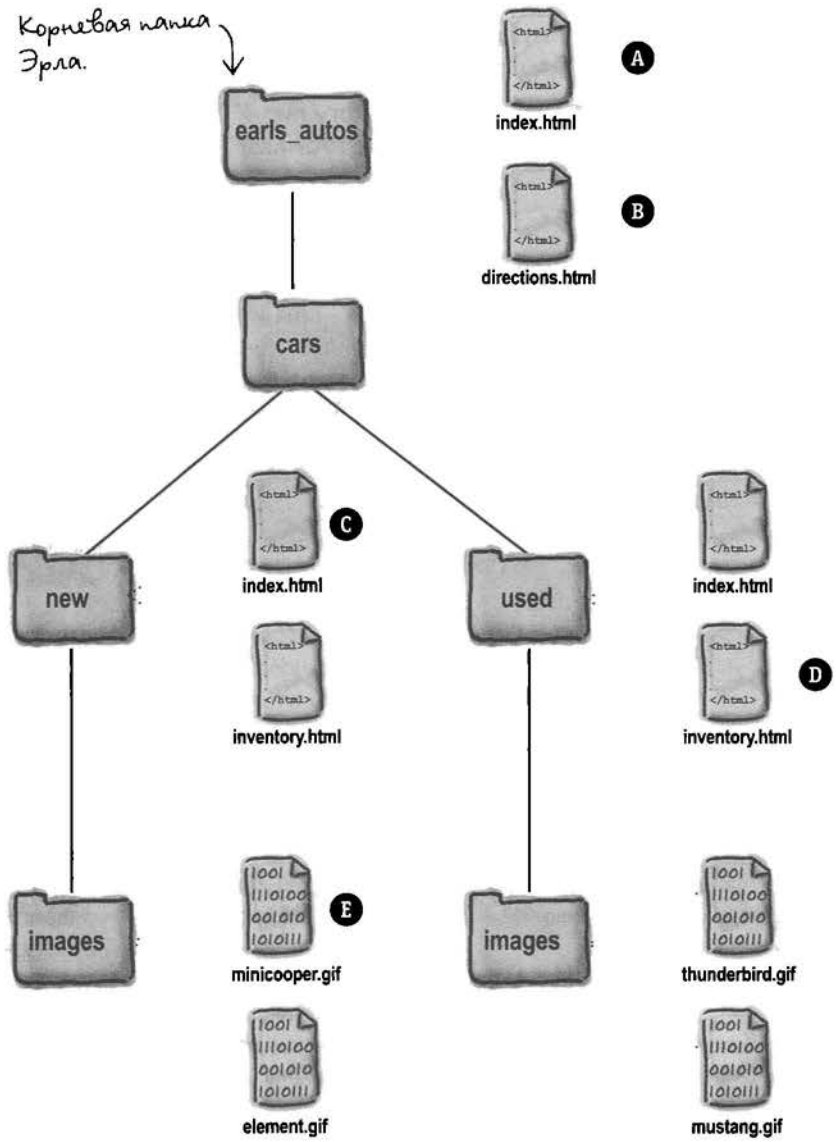
В: Итак, когда я даю кому-то свой URL-адрес, нужно включать в него часть index.html или нет?

О: Нет. Всегда лучше ее опустить. Возможно, вы в будущем поменяете веб-сервер, а он будет использовать другое имя файла, выдаваемого по умолчанию, например default.htm? Или вы начнете писать сценарии и будете использовать имя index.php? Тогда URL, который вы дали сразу, станет недействительным.



Эрлу нужно помочь разобраться с URL-адресами

Эрлу нужно помочь разобраться, какой URL для каждого из приведенных ниже файлов соответствует пометкам A, B, C, D и E. Справа напишите URL-адреса, необходимые для извлечения соответствующих файлов с сайта www.earlsautos.com.



Напишите URL здесь

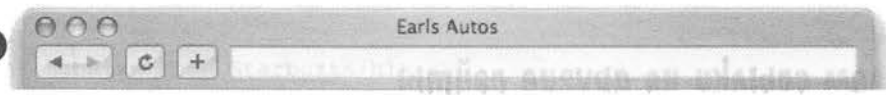
A



B



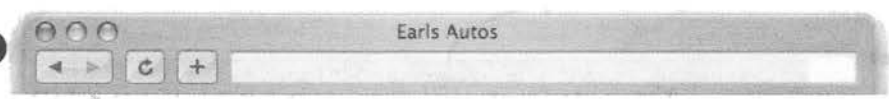
C



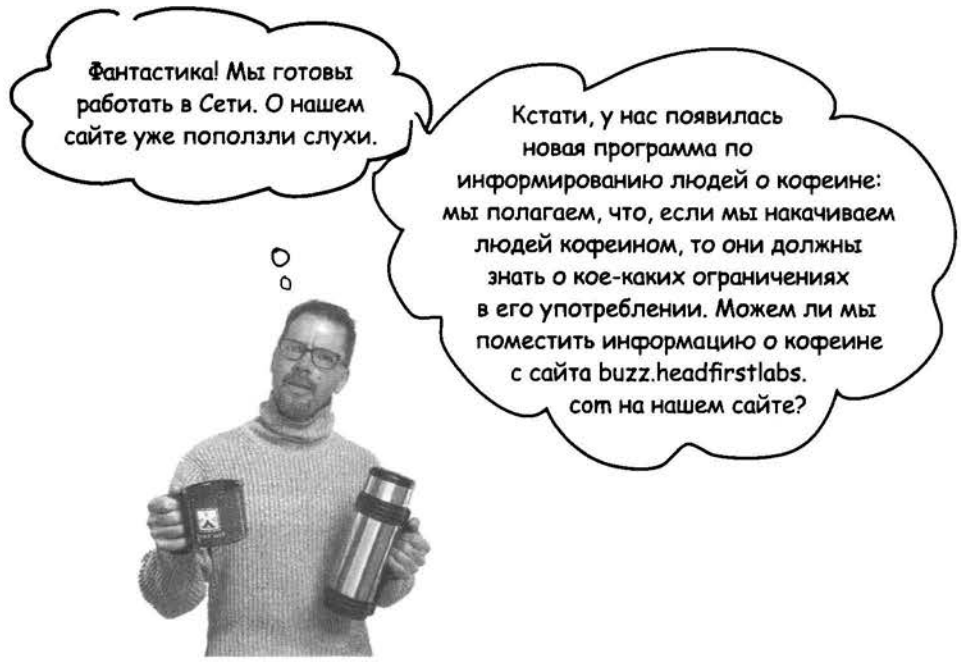
D



E







Как мы создаем ссылки на другие сайты

URL-адреса предназначены не только для того, чтобы набирать их в браузерах. Вы можете использовать их прямо в HTML. И, конечно же, как и следовало ожидать, генеральный директор Starbuzz приготовил для вас новое задание: создать ссылку с главной Starbuzz-страницы на страницу `http://buzz.headfirstlabs.com`, содержащую информацию о кофеине. Как вы, скорее всего, уже догадались, мы вставим этот URL прямо в элемент **<a>**. Вот так:

```
<a href="http://buzz.headfirstlabs.com">Все о кофеине</a>
```

Обычный, привычный элемент <a>.

Мы поместили URL в атрибут href. После щелчка на ссылке «Все о кофеине» откроется страница с сайта buzz.headfirstlabs.com.

Вот, в общем-то, и все. Чтобы сослаться на любой ресурс в Сети, вам понадобится лишь его унифицированный указатель, который нужно поместить в элемент **<a>** в качестве значения атрибута **href**. Давайте будем двигаться в этом направлении дальше и применим полученные знания к Starbuzz-странице `index.html`.

Создание ссылки на страницу о кофеине

Откройте свой файл `index.html` (для Starbuzz) из папки `chapter4/starbuzz` и просмотрите его от начала до конца. Сейчас мы рассмотрим, как добавить две новые ссылки: относительную ссылку на формулировку задачи на `mission.html` и ссылку на страницу «Все о кофеине». Сделайте изменения, указанные ниже, затем сохраните и загрузите свой файл `index.html` в браузере. Щелкните кнопкой мыши на ссылке и получайте удовольствие от новой страницы.

```
<html>
    <head>
        <title>Кафе Starbuzz</title>
        <style type="text/css">
            body {
                background-color: #d2b48c;
                margin-left: 20%;
                margin-right: 20%;
                border: 1px dotted gray;
                padding: 10px 10px 10px 10px;
                font-family: sans-serif;
            }
        </style>
    </head>

    <body>
        <h1>Напитки кафе Starbuzz</h1>
        <h2>Домашняя смесь, $1,49</h2>
        <p>Мягкая, нетерпкая смесь различных сортов кофе из Мексики,
           Боливии и Гватемалы.</p>

        <h2>Кофе мокко, $2,35</h2>
        <p>Эспрессо, кипяченое молоко и шоколадный сироп.</p>

        <h2>Капучино, $1,89</h2>
        <p>Смесь эспрессо и кипяченого молока с добавлением пены.</p>

        <h2>Чай, $1,85</h2>
        <p>Ароматный напиток из черного чая, специй,
           молока и меда.
        </p>
        <p>
            <a href="mission.html">Ознакомьтесь с нашей задачей</a>
            <br>
            Читайте<a href="http://buzz.headfirstlabs.com">Все о кофеине</a> здесь
        </p>
    </body>
</html>
```

Это ссылка на файл mission.html. В ней используется относительный путь, чтобы сослаться на mission.html. Мы использовали элемент <br>, чтобы ссылки были расположены на разных строках.

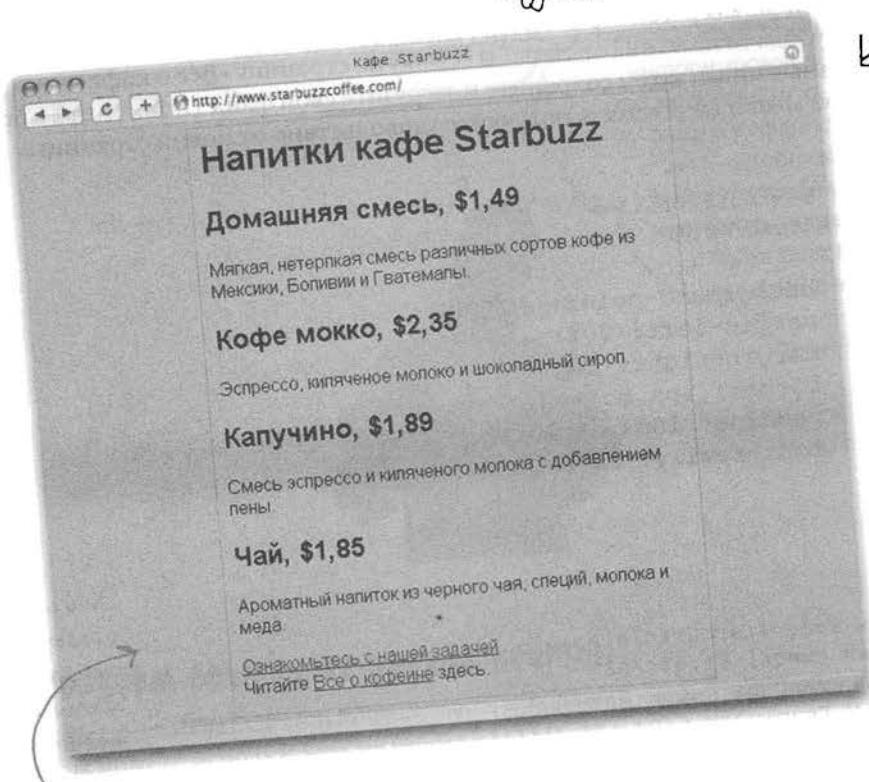
Мы добавили элемент <br>, чтобы разместить ссылки в двух разных строках.

Мы также немного структурировали эту часть, сгруппировав ссылки и текст в один абзац.

Здесь мы добавили ссылку на страницу buzz.headfirstlabs.com.

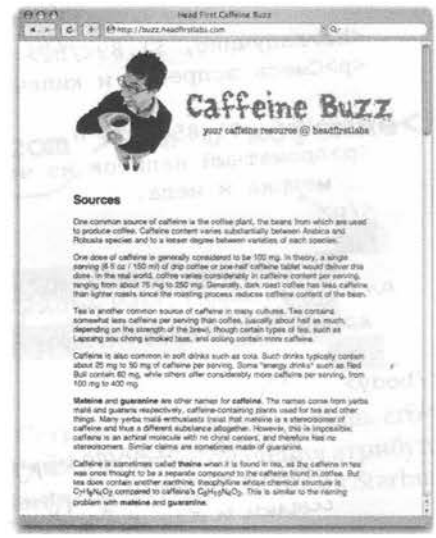
А теперь протестируем...

Это страница со ссылками, которую мы и планировали создать.



Это новая ссылка. Обратите внимание, что в качестве метки мы использовали только «Все о кофеине» и эта ссылка немного отличается от остальных.

Когда вы щелкаете на этой ссылке, браузер формирует HTTP-запрос на buzz.headfirstlabs.com и затем отображает результат.





Часто Задаваемые Вопросы

В: Кажется, теперь существует два способа создать ссылки на другие страницы: относительные пути и URL-адреса.

О: Относительные пути могут быть использованы только для того, чтобы создать ссылку на страницу *с этого же сайта*, в то время как URL-адреса обычно применяются для создания ссылки на другие сайты.

В: Не проще ли использовать URL-адреса для создания ссылок как на мои собственные, так и на внешние страницы? Ведь все будет правильно работать?

О: Конечно, все будет работать, но есть несколько причин, почему не стоит так делать. Одна заключается в том, что с URL-адресами трудно обращаться, если на веб-странице их достаточно много: они длинные, их трудно редактировать и с ними сложнее читать HTML-код (вам как автору страницы).

Кроме того, если на вашем сайте для ссылок на локальные страницы используются только URL-адреса, а вы перемещаете его в другое место или меняете его имя, вам придется поменять все эти URL-адреса, чтобы они соответствовали новому местоположению сайта. Если же вы используете относительные пути, то, пока ваши страницы находятся в неизменном наборе папок, в силу того, что ссылки относительные, вам не придется менять значения атрибутов href в элементах <a>.

Используйте относительные пути, чтобы сослаться на собственные страницы на этом же сайте, и URL-адреса, чтобы создать ссылку на страницы с других сайтов.

В: Разве мы не использовали еще один протокол? Я постоянно видел file://, перед тем как мы начали работать с веб-сервером.

О: Да, хорошее замечание. Этот протокол используется, если браузер считывает файлы прямо с вашего компьютера. URL-адрес файла, например, file:///chapter4/starbuzz/index.html, говорит браузеру, что путь к файлу index.html — /chapter4/starbuzz/. Этот путь может выглядеть по-разному в различных операционных системах.

Одна важная деталь, на которую нужно обратить внимание, если вы пытаетесь напечатать URL файла — это то, что такой URL-адрес содержит три слэша, а не два, как в HTTP. Запомните это так: если взять URL-адрес HTTP и удалить имя сайта, то получится три слэша.

В: А есть ли еще какие-нибудь протоколы?

О: Да, многие браузеры могут искать страницы, используя FTP-протокол. Кроме того, существует протокол пересылки почты, который передает данные посредством электронной почты. HTTP — это протокол, который вы будете использовать чаще всего.

В: Я встречал URL, который выглядел так: http://www.mydomain.com:8000/index.html. Почему в нем используется :8000?

О: 8000 — это дополнительный «порт», который вы можете поместить в URL HTTP. Чтобы понять, что такое порт, используйте следующую ассоциацию: имя сайта — это адрес, а порт — номер почтового ящика (скажем, в многоквартирном доме). Обычно все в Сети доставляется на порт, используемый по умолчанию (это порт 80), но иногда веб-серверы сконфигурированы так, чтобы получать запросы на другой порт (например, 8000). Чаще всего это тестовые серверы. Обычные веб-серверы почти всегда принимают запросы на порт 80. Если вы не укажете другой порт, то по умолчанию будет использоваться порт 80.

Пятиминутная головоломка



История об относительных и абсолютных путях

Перед корпорацией PlanetRobots, Inc. была поставлена задача разработки сайта для каждой из их двух компаний: PlanetRobot Home и PlanetRobot Garden. Было решено заключить договор с двумя разными фирмами. Фирма RadWebDesign, производящая впечатление имеющей большой опыт в такой работе, взялась за сайт для отдела Home и писала его, используя при создании внутренних ссылок только URL-адреса (в конце концов, они выглядят более сложно, а значит, они лучше). Фирма CorrectWebDesign, имеющая меньше опыта в написании сайтов, но с хорошо обученными специалистами, взялась за написание сайта для PlanetRobot Garden и для создания ссылок на все страницы внутри сайта использовала относительные пути.

Когда создание обоих проектов подходило к концу, в обе фирмы позвонили из корпорации PlanetRobots со срочным сообщением: «На нас был подан иск по поводу нарушения права собственности на торговую марку, поэтому мы меняем доменное имя на RobotsRUs. Теперь нашим новым веб-сервером будет `www.robotsrus.com`». В CorrectWebDesign сделали пару небольших изменений, что заняло около пяти минут, и все было готово для презентации сайта в корпорации RobotsRUs. А программисты компании RadWebDesign работали до 4 часов утра, чтобы внести соответствующие изменения в веб-страницы, и, к счастью, успели завершить работу ко времени проведения презентации. Однако во время демонстрации сайтов произошло ужасное: когда участник команды RadWebDesign демонстрировал сайт, он щелкнул на ссылке и появилась ошибка 404 «Страница не найдена». Недовольный генеральный директор корпорации RobotsRUs предложил им подумать над тем, чтобы поменять название своей фирмы с RadWebDesign на BadWebDesign, и спросил представителей фирмы CorrectWebDesign, в состоянии ли они дать консультацию по поводу исправления главного сайта.

Что же произошло? Как в RadWebDesign допустили такой ляп, если нужно было поменять всего лишь имя веб-сервера?

Наивысший уровень качества веб-страниц

Можете ли вы уже говорить о своей «веб-карьере»? Вы, конечно же, успешно справились со всем, о чем вас просил генеральный директор Starbuzz, и, можно сказать, создали хорошо известный всем сайт (и он уже есть в вашем портфолио).

Но вы, конечно же, не собираетесь останавливаться на достигнутом. Вы хотите, чтобы у вашего сайта был тот профессиональный уровень качества, который отличает хорошие сайты от отличных. В оставшейся части этой книги мы подскажем вам множество способов придания сайту особого лоска, но сейчас начнем со способа усовершенствования ссылок.

Улучшение доступности путем добавления названий вашим ссылкам

Разве не было бы здорово, если бы существовал способ получения дополнительной информации о ссылке, на которой вы собираетесь щелкнуть? Это особенно важно для людей со слабым зрением, которые используют экранные дикторы. Такие люди часто не хотят, чтобы им проговаривался весь URL целиком («h, t, t, p, двоеточие, слэш, слэш, w, w, w, точка»), а метка ссылки обычно дает только ограниченное описание, например «Все о кофеине».

У элемента **<a>** есть атрибут `title`, используемый именно для этой цели. Некоторые люди удивляются, услышав об этом атрибуте, потому что существует также *элемент* **<title>**, вложенный в элемент **<head>**. Они называются одинаково, потому что взаимосвязаны: часто предлагается, чтобы значение атрибута `title` совпадало со значением элемента **<title>** веб-страницы, на которую указывает ссылка. Но это необязательное требование, и часто имеет больше смысла дать свое собственное, более подходящее описание в атрибуте `title`.

Рассмотрим, как атрибут `title` добавляется в элемент **<a>**:

```
Читайте <a href="http://buzz.headfirstlabs.com"
           title="Все самое интересное о кофеине">Все о кофеине</a>
```

Значение атрибута title — это текстовое описание страницы, на которую вы ссылаетесь.

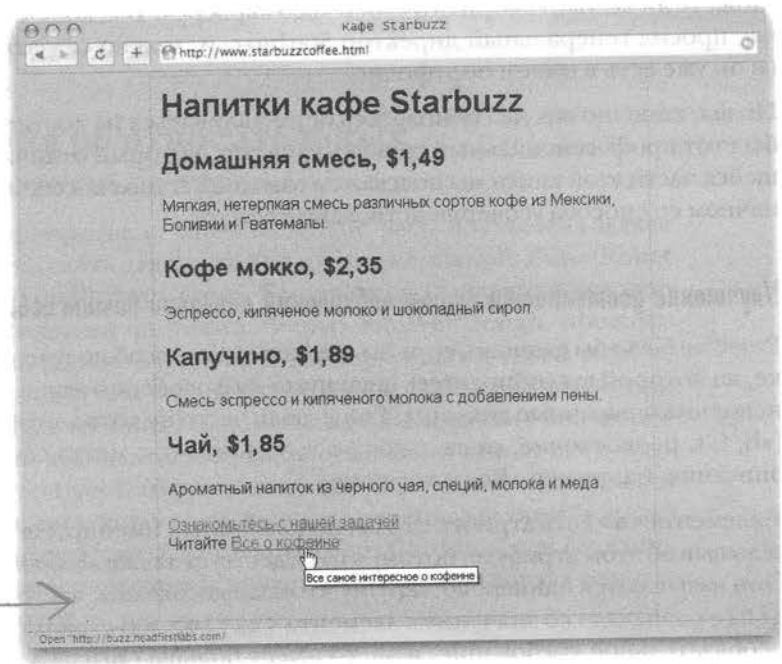


Упражнение

Теперь, когда у нас есть атрибут `title`, давайте посмотрим, как им будут пользоваться ваши посетители. В разных браузерах атрибут `title` применяется по-разному, но в большинстве случаев появляется всплывающая подсказка. Добавьте изменение, описанное выше, в файл `index.html` и обновите страницу, чтобы посмотреть, как это работает в браузере.

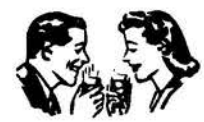
Тестирование атрибута title

Для большинства браузеров значение атрибута `title` отображается во всплывающей подсказке, когда вы наводите указатель мыши на ссылку. Помните, что браузеры для людей с ослабленным зрением могут воспроизводить такой заголовок вслух.



Заголовок отображается во всплывающей подсказке в большинстве браузеров. Просто наведите указатель мыши на ссылку и подождите, пока появится всплывающая подсказка.

Проводник Head First по лучшим ссылкам



Здесь мы приводим пару полезных советов, которые нужно держать в голове, чтобы в дальнейшем усовершенствовать свои ссылки.

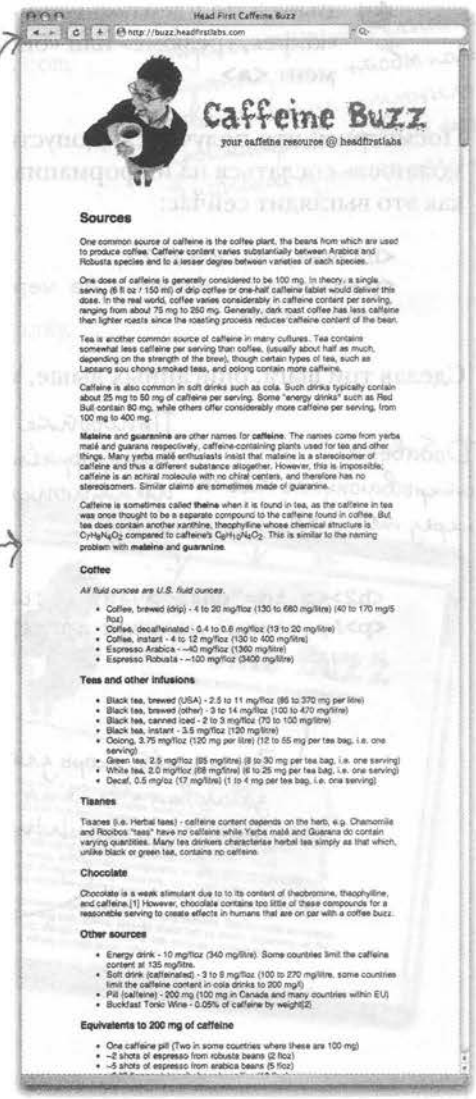
- *Выбирайте короткие метки для ссылок. Не используйте в качестве меток целые предложения или большие абзацы текста. Старайтесь вообще не использовать больше нескольких слов, а дополнительную информацию представляйте с помощью атрибута title.*
- *Выбирайте содержательные метки для ссылок. Никогда не используйте такие метки, как «щелкните здесь» или «эта страница». Пользователи чаще всего сначала бегло просматривают страницы и ищут на них ссылки и только потом читают эти страницы. Итак, содержательные ссылки улучшат работу с вашей страницей. Протестируйте веб-страницу, просто прочитав ссылки на ней. Есть ли в них какая-нибудь логика? Или необходимо прочитать текст около них, чтобы понять, о чем речь?*
- *Избегайте размещения ссылок непосредственно друг за другом. У пользователей возникнут трудности в том, чтобы рассмотреть ссылки, которые расположены одна за другой.*



Упражнение

Откройте файл `index.html` сайта Starbuzz и добавьте атрибут **`title`** для ссылки на `mission.html` с текстом «Узнайте больше о важной задаче кафе Starbuzz». Заметьте, что мы не сделали метку для ссылки максимально короткой. Сократите метку до «Наша задача». Сверьте свои результаты с ответами, приведенными в конце главы.





Создание ссылки внутрь страницы

До сих пор всегда, когда вы переходили по ссылке на другую страницу, она загружалась и браузер отображал ее с самого начала.

Но генеральный директор Starbuzz просит, чтобы вы создали *ссылку на отдельный участок страницы*: раздел о кофе.

Звучит как что-то невыполнимое? Вперед, это ведь Head First – и у нас есть технология для этого. Какая? Ну, мы еще не все вам рассказали об элементе **`<a>`**. Оказывается, он может играть *две роли*. Как вы уже видели, он может служить отправной точкой для путешествия с одной страницы на другую. Кроме того, он может стать *местом приземления* или *пунктом назначения* ссылки.

Использование элемента <a> для указания пункта назначения

Когда элемент **<a>** применяется для указания пункта назначения, можно говорить о якоре. Его достаточно просто создать. Рассмотрим, как это можно сделать за три коротких шага.

1. Найдите то место на странице, которое вы хотите использовать в качестве места приземления. Это может быть любой текст на странице, но чаще применяется короткий фрагмент текста заголовка.
2. Заключите этот текст внутрь элемента **<a>**.
3. Выберите идентификационное имя для фрагмента, например «кофе», «резюме» или «биография», и вставьте атрибут `id` в элемент **<a>**.

Посмотрим, что получится. Допустим, вы хотите предоставить возможность сослаться на информацию о чае на странице Starbuzz. Вот как это выглядит сейчас:

Небольшой фрагмент из файла index.html с заголовком «Чай» и описанием.

```
<h2>Чай, $1.85</h2>
<p>Ароматный напиток из черного чая, специй, молока
и меда.</p>
```

Сделав три шага, описанных выше, мы получим следующее.

Добавьте открывающий тег <a> перед текстом.

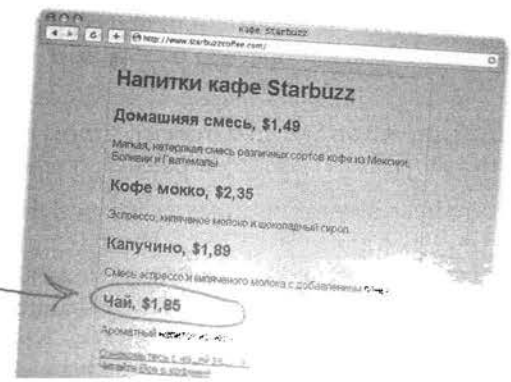
Присвойте этому фрагменту идентификатор chai.

Затем добавьте закрывающий тег элемента.

Убедитесь, что элемент <a> должным образом вложен в элемент <h2>.

```
<h2><a id="chai">Чай, $1.85</a></h2>
<p>Ароматный напиток из черного чая, специй, молока
и меда.</p>
```

Вы создали якорь для заголовка «Чай» на странице index.html.



Как сослаться на якорь

Вы уже знаете, как сослаться на страницы с использованием относительных путей или URL-адресов. А чтобы сослаться на якорь на странице, просто добавьте символ «#» в конец вашей ссылки, за которым укажите имя якоря. Так, если вы хотите сослаться с любой страницы кафе Starbuzz на якорь `chai`, вам нужно написать элемент **<a>** следующим образом:

```
<a href="index.html#chai">Смотри страницу о чае</a>
```

Основная польза от якорей в том, что с их помощью можно ссылаться на отдельные части длинных страниц так, чтобы пользователям не приходилось просматривать весь файл в поисках нужного раздела.

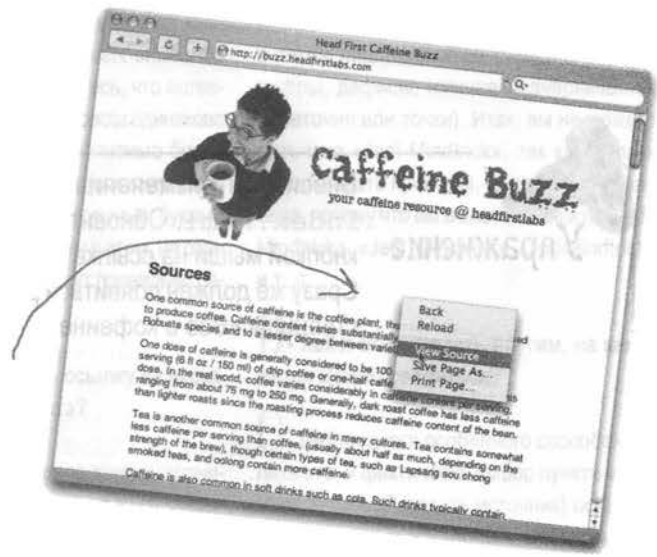
К сожалению, ссылка на информацию о чае с помощью якоря — не очень удачный пример, потому что вся страница достаточно маленькая и легко помещается в окне браузера. Давайте вместо этого создадим ссылку на раздел о кофе страницы `http://buzz.headfirstlabs.com`. Рассмотрим, что вам нужно сделать.

1. Выясните идентификатор якоря.
2. Измените существующий элемент **<a>** в Starbuzz-файле `index.html` так, чтобы указать на якорь.
3. Обновите страницу `index.html` и протестируйте ссылку.

Поиск якоря

Чтобы найти якорь, нужно просмотреть HTML-код страницы `buzz.headfirstlabs.com`. Как? Почти во всех браузерах есть пункт меню View Source (Посмотреть источник). Итак, подождите, пока страница полностью загрузится в окне браузера, и выберите указанный пункт, и вы увидите HTML-код для этой страницы.

В большинстве браузеров можно щелкнуть правой кнопкой мыши, и появится контекстное меню, в котором есть пункт View Source (Посмотреть источник).



Теперь, когда их HTML у вас в руках...

Прокручивайте страницу вниз, пока не увидите раздел о кофе, который выглядит так:

Небольшой фрагмент страницы «Все о кофеине».

```
Это похоже на проблему с названиями
<b>матеин</b> и <b>кофеин</b>.
</p>

<h3><a id="Coffee">Кофе</a></h3>
```

Это раздел «Кофе». Вы видите его название и новый абзац, который начинается после него.

Ах, и здесь есть якорь. Его имя «Coffee».

Исправление ссылки в index.html

Теперь все, что вам остается сделать, — перенаправить ссылку на страницу «Все о кофеине» и добавить имя якоря, как здесь:

Это фрагмент Starbuzz-файла index.html.

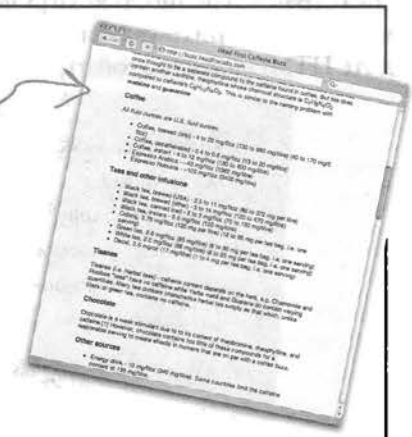
Добавьте символ «#» как идентификатор якоря в ваш атрибут href.

```
Читайте <a href="http://buzz.headfirstlabs.com#Coffee"
         title="Читайте все самое важное о кофеине">Все о кофеине</a>
```



Упражнение

Внесите эти изменения в Starbuzz-файл `index.html`. Обновите его и щелкните кнопкой мыши на ссылке «Все о кофеине». Сразу же должен появиться раздел «Кофе» страницы «Все о кофеине».



Часто Задаваемые Вопросы

В: Если в элементе два атрибута, важна ли их очередность? Например, должен ли атрибут title всегда следовать за href?

О: Порядок следования атрибутов в элементе не имеет значения (в ином случае у нас бы непрерывно болела голова от этого), поэтому используйте любую очередность атрибутов.

В: Обычно, когда я использую элемент <a>, браузер подчеркивает текст, но когда я вместо href использовал атрибут id, то текст не был подчеркнут.

О: Верно. Когда вы добавляете атрибут id, то не применяется никаких визуальных эффектов для текста, который окружается элементом <a>. Помните, предназначение якоря (элемента <a> с атрибутом id) в том, чтобы просто пометить определенную область страницы, а не создать ссылку, поэтому нет никакой необходимости выделять текст.

В: А почему это называется якорем? Что здесь похоже на якорь?

О: На это мы ответим так: «якорь» — это просто плохо подобранное имя, что вводило в заблуждение уже десятки тысяч людей до вас. Мы даже не будем пытаться привести какую-нибудь изящную метафору, чтобы вы поняли, как это на самом деле может быть ассоциировано с якорем. Обычно мы все привязываемся к имени, но сейчас вы знаете, как действительно обстоят дела, и в дальнейшем не будете даже пытаться осмыслить это название.

В: Хорошо, пусть имя выбрано плохо, но почему один и тот же элемент делает совершенно разные вещи? Почему бы не иметь два разных элемента: для ссылок и для указания пункта назначения?

О: Думайте об этом следующим образом: когда необходимо сослаться с чего-то одного на что-то другое, элемент <a> с атрибутом href дает способ описать тот адрес, с которого мы ссылаемся. А то, на что мы ссылаемся в этом случае, — всегда самая «верхушка» веб-страницы. Иными словами, вам не нужно уточнять, на что вы ссылаетесь. А с якорем вы можете сделать это. Итак, все же в этих именах есть кое-какой здравый смысл.

В: Я заметил, что в качестве имен якорей вы использовали «chai» со всеми строчными буквами, а для «Все о кофеине» указали «Coffee» с прописной буквой «С». Это важно?

О: Вы можете использовать любые комбинации строчных и прописных символов в атрибуте id. Просто убедитесь, что остаетесь последовательными и всегда одинаково используете строчные и прописные буквы в атрибутах href и якорях id (поэтому часто лучше использовать все строчные буквы). Если вы непоследовательны в этом, не ожидайте, что ваши ссылки будут правильно работать во всех браузерах.

В: А могу ли я создать ссылку на якорь внутри одного документа?

О: Конечно. На самом деле вверху страницы часто задается якорь «top», а внизу указывается ссылка «На начало страницы». Кроме того, часто в длинных документах используется содержимое всей страницы. Например, чтобы сослаться на якорь «top» на этой же странице, нужно написать На начало страницы.

В: А если в веб-странице не предусмотрено якорей, но мне тем не менее нужно сослаться на определенную часть страницы, как это сделать?

О: Вы не сможете этого сделать. Если нет якоря, то нельзя указать браузеру перейти в определенное место на странице. Вы можете связаться с автором страницы и попросить его добавить якорь (а еще лучше рассказать ему, как это сделать!).

В: Могу ли я использовать такой id, как «Jedi Mindtrick», или id должен состоять из одного слова?

О: Чтобы ваши страницы работали согласованно с большинством браузеров, всегда начинайте id с буквы (A–Z или a–z), а затем пишите любые символы (буквы, цифры, дефисы, нижние подчеркивания, двоеточия или точки). Итак, вы не можете дать имя «Jedi Mindtrick», так как нельзя использовать пробелы, но это небольшая беда, потому что вы можете написать «Jedi-Mindtrick», «Jedi_Mindtrick», «JediMindtrick» и т. д.

В: Как я могу сказать другим, на какие якоря можно сослаться?

О: Нет никакого особенного способа сделать это, и фактически выбор пункта меню View Source (Показать источник) остается самым старым и удобным способом для обнаружения якорей, на которые можно ссылаться.

Пятиминутная головоломка



Решение

История об относительных и абсолютных путях

Итак, как же компания RadWebDesign так провалилась на презентации? Поскольку они использовали для атрибутов `href` URL-адреса вместо относительных ссылок, им пришлось редактировать и менять *каждую отдельную ссылку* с `http://www.planetrobots.com` на `http://www.robotsrus.com`. В чем же они ошиблись? В 3 часа утра кто-то зазевался и случайно напечатал `http://www.robotsru.com` (и по воле случая это оказалась та же ссылка, на которой генеральный директор щелкнул на презентации).

Упс... Кто-то забыл напечатать «s» в конце имени

В CorrectWebDesign, напротив, использовали относительные пути во внутренних ссылках. Например, ссылка со страницы с постановкой задачи компании на страницу с ее продукцией — `<a href="../products.html">` — корректно работала независимо от того, как назывался сайт: PlanetRobots или RobotsRUs. Итак, все, что должны были сделать в CorrectWebDesign, — исправить имя компании на нескольких страницах.

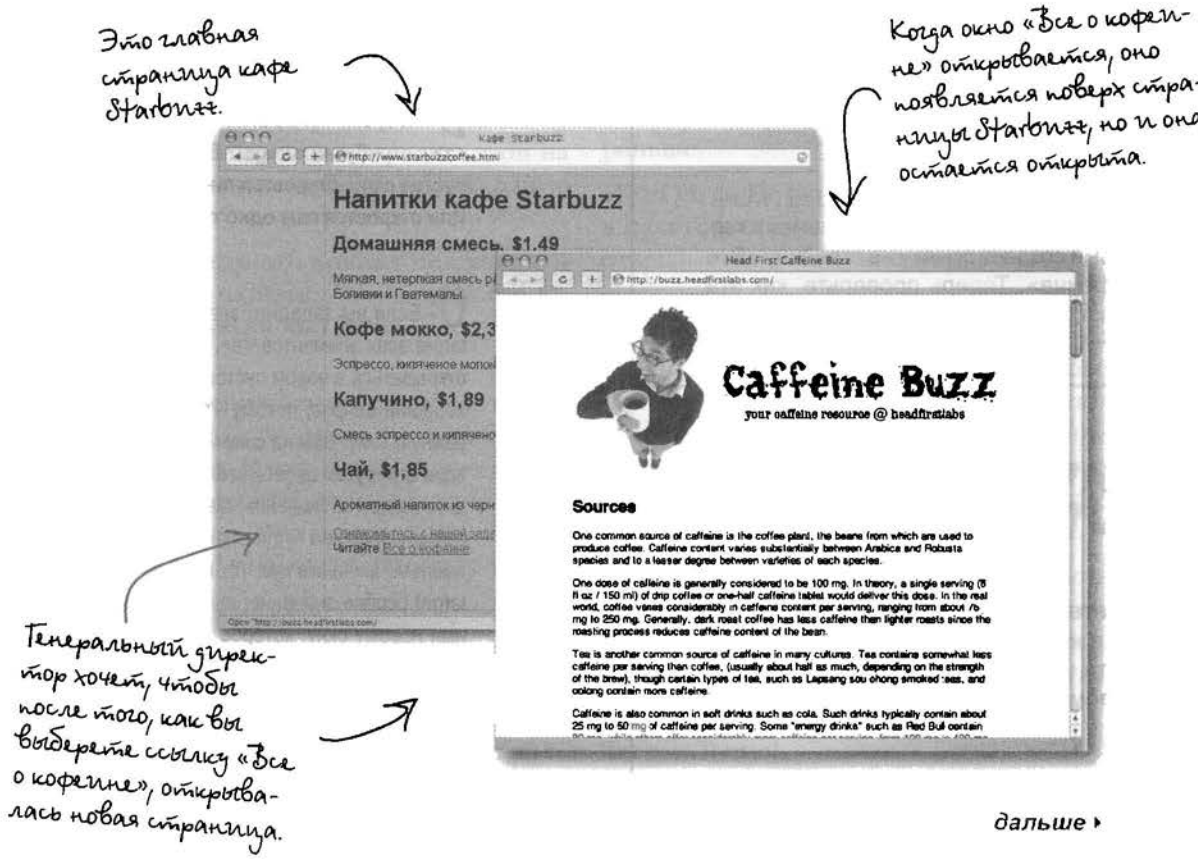
Итак, представители RadWebDesign покинули презентацию сонные и с кислыми выражениями лиц, в то время как представители CorrectWebDesign ушли, получив еще один заказ. Но на этом история не заканчивается. Случилось так, что после этого компании RadWebDesign было отказано еще и небольшим кафе и книжным магазином, и, чтобы не развалиться вовсе, она приобрела книгу по HTML и CSS. Что же случилось потом? Присоединяйтесь к нам через несколько глав и читайте историю противостояния Грубой Силы и Изящества.



Необычное задание по созданию ссылки на сайт о кофе... Я знаю, что постоянно прошу что-то поменять, но это на самом деле в последний раз. Вы можете сделать так, чтобы сайт о кофе появлялся в отдельном окне после того, как я щелкну на ссылке? Я не хочу, чтобы страница Starbuzz исчезала.

Переход по ссылке в новое окно

У нас есть еще одно требование от генерального директора Starbuzz (к сайтам *постоянно* выдвигаются новые требования). Вот что он хочет: когда вы щелкаете на ссылке «Все о кофеине» на странице Starbuzz, эта страница не должна исчезать. Вместо этого должно открываться новое окно со страницей «Все о кофеине» в нем.



дальше ▸

Открытие нового окна с использованием атрибута target

Чтобы открыть страницу в новом окне, вам нужно сообщить браузеру имя окна, в котором нужно открыть страницу. Если вы не говорите браузеру использовать какое-то особенное окно, то он открывает страницу в текущем окне. Если вы хотите сказать браузеру, что нужно использовать *другое* окно, добавьте в элемент **<a>** атрибут `target`. Его значение сообщает браузеру о «целевом окне» для страницы. Если в качестве значения атрибута `target` вы используете `_blank`, то браузер для каждой новой страницы *всегда* будет открывать новое окно. Разберемся в этом более детально:

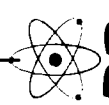
```
<a target="_blank" href="http://buzz.headfirstlabs.com"
   title="Читайте все самое важное о кофеине">Все о кофеине</a>
```

Атрибут target говорит браузеру, где открыть веб-страницу, которая указана в атрибуте href этой ссылки. Если атрибута target нет, то браузер открывает страницу в текущем окне. Если в качестве значения атрибута target используется "_blank", то браузер открывает ссылку в новом окне.



Упражнение

Откройте Starbuzz-файл `index.html`. Добавьте атрибут `target` в элемент **<a>**, который создает ссылку на страницу «Все о кофеине». Теперь проверьте, как эта ссылка работает. Открылось новое окно?



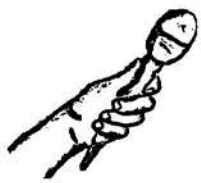
МОЗГОВОЙ ШТУРМ

Можете придумать парочку преимуществ и неудобств использования атрибута `target` для открытия страницы в новом окне?

Часто Задаваемые Вопросы

В: Что, если у меня есть несколько элементов <a> с атрибутами target? Если уже открыто новое пустое окно, откроется ли в нем новая страница? Или откроется еще одно такое окно?

О: Если вы зададите значение _blank атрибутам target всех элементов <a>, то каждая ссылка будет открываться в новом пустом окне. Тем не менее это хороший вопрос, потому что он затрагивает очень важную тему. Вам на самом деле не нужно задавать всем атрибутам target значение _blank. Если вы зададите им другое значение, скажем coffee, то все ссылки с этим значением будут открываться в окне с таким же именем. Причина в том, что, когда вы задаете атрибуту target особое значение, вы на самом деле задаете имя новому окну, которое будет использовано для отображения страницы из ссылки. _blank — это особое значение, с помощью которого браузеру указывается всегда использовать новое окно.



РАЗОБЛАЧЕНИЕ АТРИБУТА TARGET

Интервью, взятое на этой неделе.
Использование target считается плохим тоном?

Head First: Привет, Target, мы так рады, что ты смог с нами пообщаться!

Target: Я тоже рад, что я здесь. Приятно осознавать, что вы все еще интересуетесь мною.

Head First: А почему ты так говоришь?

Target: Ну, если честно, я уже не такой популярный, как раньше.

Head First: А как ты думаешь, почему так случилось?

Target: Думаю, пользователи теперь хотят управлять тем, когда открывать окна. Они не всегда хотят, чтобы в самый неожиданный момент открывалось новое окно.

Head First: Ну, это действительно может создавать неудобство в работе. Нам поступали жалобы от людей, у которых в конце концов появлялось так много окон на экране, что они не могли найти исходную страницу.

Target: Но ведь это совсем не сложно — справиться с окнами... Нужно просто нажать маленькую кнопку Закрыть. Что тут трудного?!

Head First: Верно, но пользователь может запутаться, если не знает, что новое окно было открыто. Иногда новое окно полностью закрывает старое и сложно понять, что произошло. Это может сбить с толку, особенно если у человека слабое зрение.

Target: О, я никогда об этом не думал.

Head First: Хорошо, подумай об этом: если кто-нибудь открыл окно браузера на весь экран, а новое окно появляется поверх старого, то оно закрывает то, что читал пользователь, и это может его очень смутить. Сложно сказать, что происходит на экране, если не видеть его целиком.

Target: Да, я допускаю, что это так. Скорее всего, это также создает трудности при использовании экранных дикторов.

Head First: Точно. В некоторых экранных дикторах при открытии нового окна играет музыка, а в других такой момент просто игнорируется или они немедленно делают новое окно текущим. В любом случае это будет сбивать с толку тех, кто не может видеть, что происходит на экране. И, конечно же, поскольку страница открывается в новом окне, нельзя вернуться к исходной странице, используя кнопку Назад.

Target [вздыхая]: Я начинаю понимать, почему я уже не такой популярный, как раньше.

Head First: Не переживай так сильно. Ведь в некоторых случаях очень здорово, если открывается новое окно, верно?

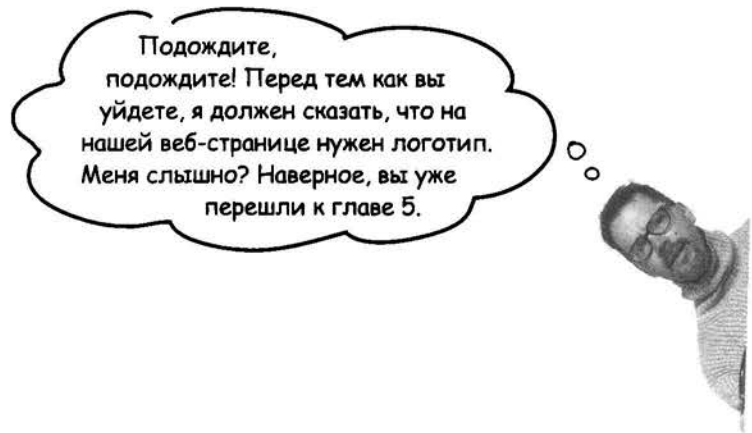
Target: Да, я всегда думал, что эти маленькие окошки с «дополнительной информацией» очень удобны для пользователя, и я особенно горжусь собой тогда, когда люди используют меня, чтобы просмотреть увеличенную версию изображения. Таким образом пользователь может посмотреть большой рисунок, а затем вернуться к основной странице.

Head First: Отлично, временами ты действительно бываешь полезен. Мы просто должны использовать тебя в подходящих ситуациях, не забывать о людях со слабым зрением и не злоупотреблять твоими услугами.

Target: Точно!

ПОВТОРИМ выученное

- Лучший способ разместить что-либо в Сети — найти хостинговую компанию, которая расположит на своем сервере ваши веб-страницы.
- Доменное имя — это уникальное имя, например `amazon.com` или `starbuzzcoffee.com`, используемое для распознавания сайта.
- Хостинговая компания может создать один или несколько веб-серверов в вашем домене. Часто серверы называются www.
- Протокол передачи файлов (FTP) — общее средство пересылки веб-страниц и их содержимого серверу.
- Такие клиенты, как Fetch для Mac или WS_FTP для Windows, могут упростить использование FTP, предоставляя графический интерфейс пользователю.
- URL — это унифицированный указатель ресурса, или веб-адрес, который может быть использован для идентификации какого-либо ресурса в Сети.
- Обычный URL-адрес состоит из протокола, имени сайта и абсолютного пути к ресурсу.
- HTTP — это запросно-ответный протокол, используемый для пересылки веб-страниц между веб-сервером и вашим браузером.
- Протокол `file://` применяется браузером для чтения страниц с вашего компьютера.
- Абсолютный путь — это путь от корневой папки к файлу.
- `index.html` и `default.htm` — примеры страниц, выдаваемых по умолчанию. Если вы указываете каталог без имени файла, то веб-сервер будет искать именно такие страницы, чтобы вернуть их браузеру.
- Чтобы сослаться на другие веб-страницы, можете использовать либо относительные пути, либо URL-адреса в атрибуте `href` элемента **<a>**. Для других страниц с вашего сайта лучше использовать относительные пути, а для внешних сайтов — URL-адреса.
- Применяйте атрибут `id` для создания якоря на странице. Используйте после него символ «#» с именем якоря, чтобы сослаться на эту область вашей страницы.
- Для удобства работы со страницей применяйте атрибут `title`, чтобы предоставить пользователю описание ссылки элемента **<a>**.
- Используйте атрибут `target`, чтобы открыть ссылку в новом окне браузера. Но не забывайте, что этот атрибут может нести с собой некоторые проблемы для пользователей множества различных устройств и альтернативных браузеров.



Возьми в руку карандаш

Решение

http	:// www.starbuzzcoffee.com	/index.html
протокол	имя сайта	абсолютный путь

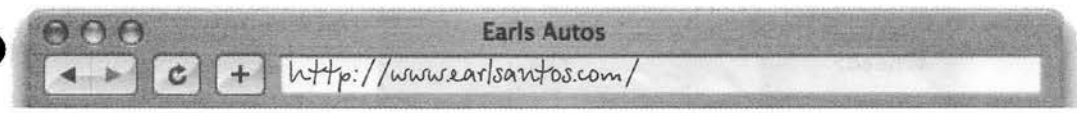
Здесь имя вашего сайта.



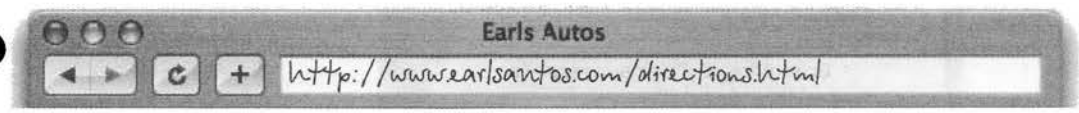
Эрлу нужно помочь разобраться с URL-адресами

Решение

A



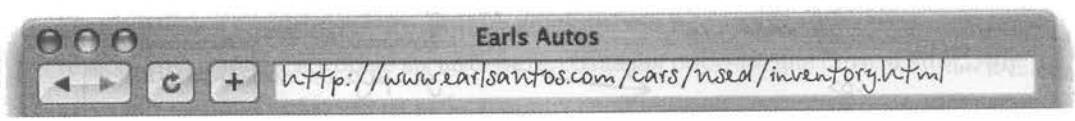
B



C



D



E





Решение упражнений

Добавьте **title** для ссылки `mission.html` с текстом «Узнайте больше о важной задаче Starbuzz Coffee». Заметьте, что мы не сделали метку для ссылки максимально короткой. Сократите метку ссылки до «наша задача». Вот решение; вы тестировали свои изменения?

```
<html>
    <head>
        <title>Кафе Starbuzz</title>
        <style type="text/css">
             body {
                    background-color: #d2b48c;
                    margin-left: 20%;
                    margin-right: 20%;
                    border: 1px dotted gray;
                    padding: 10px 10px 10px 10px;
                    font-family: sans-serif;
                  }
        </style>
    </head>

    <body>
        <h1>Напитки кафе Starbuzz</h1>
        <h2>Домашняя смесь, $1,49</h2>
        <p>Мягкая, нетерпкая смесь различных сортов кофе из Мексики,
           Боливии и Гватемалы.</p>

        <h2>Кофе мокко, $2,35</h2>
        <p>Эспрессо, кипяченое молоко и шоколадный сироп.</p>

        <h2>Капучино, $1,89</h2>
        <p>Смесь эспрессо и кипяченого молока с добавлением пены.</p>

        <h2>Чай, $1,85</h2>
        <p>Ароматный напиток из черного чая, специй, молока
           и меда.</p>

        <p>
            Ознакомьтесь с <a href="mission.html"
title="Ознакомьтесь с важной задачей кафе Starbuzz">нашей задачей</a>
            <br>
            Читайте <a href="http://buzz.headfirstlabs.com"
            title="Читайте все самое важное о кофеине">Все о кофеине</a>здесь.
        </p>
    </body>
</html>
```

Добавьте атрибут title к ссылке на страницу mission.html.

Вынесите «Ознакомьтесь с» за пределы элемента <a>.

Знакомство с медиа



Улыбнитесь и скажите «сыр». Теперь улыбнитесь и скажите «gif», «jpg» или «png» — это те форматы файлов, которые вы выберете, создавая рисунки для Сети. В этой главе вы узнаете все о том, как добавить на веб-страницу свой первый медиафайл — изображение. У вас есть парочка цифровых фотографий, которые вы хотите поместить в Сеть? Никаких проблем. У вас есть логотип, который нужен на веб-странице? И это легко. Или, может быть, сначала вы хотите более близко познакомиться с элементом `<img>`? К концу этой главы вы будете знать все мельчайшие подробности того, как использовать этот элемент и его атрибуты. Вы также узнаете, как этот небольшой элемент побуждает браузер делать такую серьезную работу по поиску и отображению ваших картинок.

Как браузер работает с изображениями

Браузер обращается с элементом **<img>** немного иначе, чем с остальными элементами. Возьмем, к примеру, элемент **<h1>** или **<p>**. Когда браузер видит их на странице, все, что ему нужно, – отобразить их. Достаточно просто. Но когда браузер видит элемент **<img>**, происходит совершенно другое: сначала он должен извлечь картинку и только потом сможет отобразить ее на странице.

Проще всего понять это на примере. Давайте еще раз взглянем на страницу с напитками из гостевой Head First, в которой есть четыре элемента **<img>**:



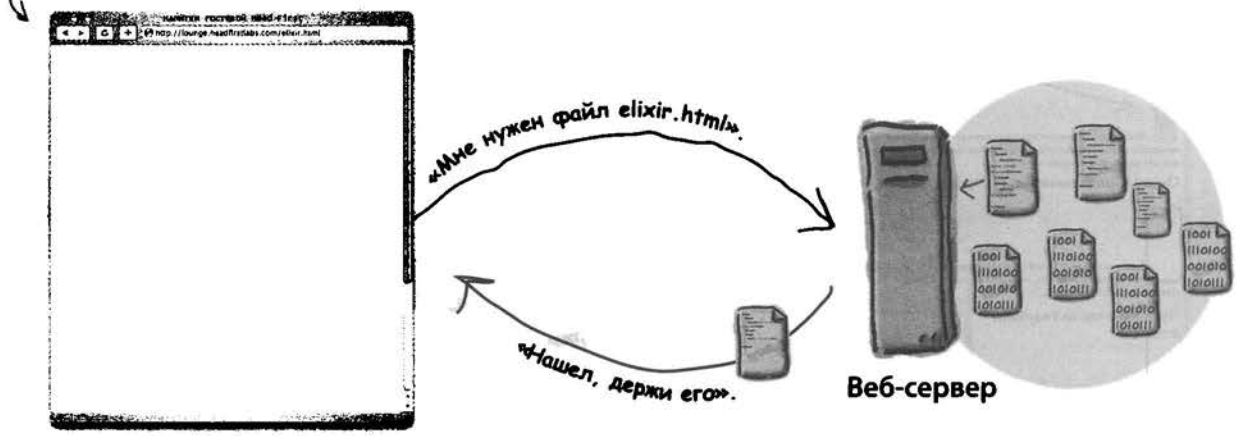

```
<html>
  <head>
    <title>Напитки гостевой Head First</title>
  </head>
  <body>
    <h1>Наши напитки</h1>

    <h2>Охлажденный зеленый чай</h2>
    <p>
            <img src="../images/green.jpg">
            Этот напиток содержит огромное количество витаминов
            и минералов и приносит пользу для здоровья благодаря
            составу: зеленый чай, цветки ромашки и корень имбиря.
    </p>
    <h2>Охлажденный малиновый сироп</h2>
    <p>
            <img src="../images/lightblue.jpg">
            Сочетая в себе малиновый сок и лимонное сорго,
            цедру цитрусовых и плоды шиповника, этот
            прохладительный напиток освежит и прояснит ваш разум.
    </p>
    <h2>Чудо-напиток из голубики</h2>
    <p>
            <img src="../images/blue.jpg">
            Экстракты голубики и вишни, добавленные в травяной
            чай из бузины, сразу же приведут вас в состояние
            покоя и блаженства.
    </p>
    <h2>Клюквенный антиоксидантный взрыв</h2>
    <p>
            <img src="../images/red.jpg">
            Зарядитесь энергией богатого витамином С напитка
            со вкусом клюквы и гибикуса.
    </p>
    <p>
        <a href="../lounge.html">Назад в гостевую</a>
    </p>
  </body>
</html>
```

В этом HTML документе есть четыре изображения.

Давайте теперь подробно рассмотрим этот код и поэтапно проследим, как браузер извлекает и отображает эту страницу, когда она запрашивается с сайта `http://lounge.headfirstlabs.com`.

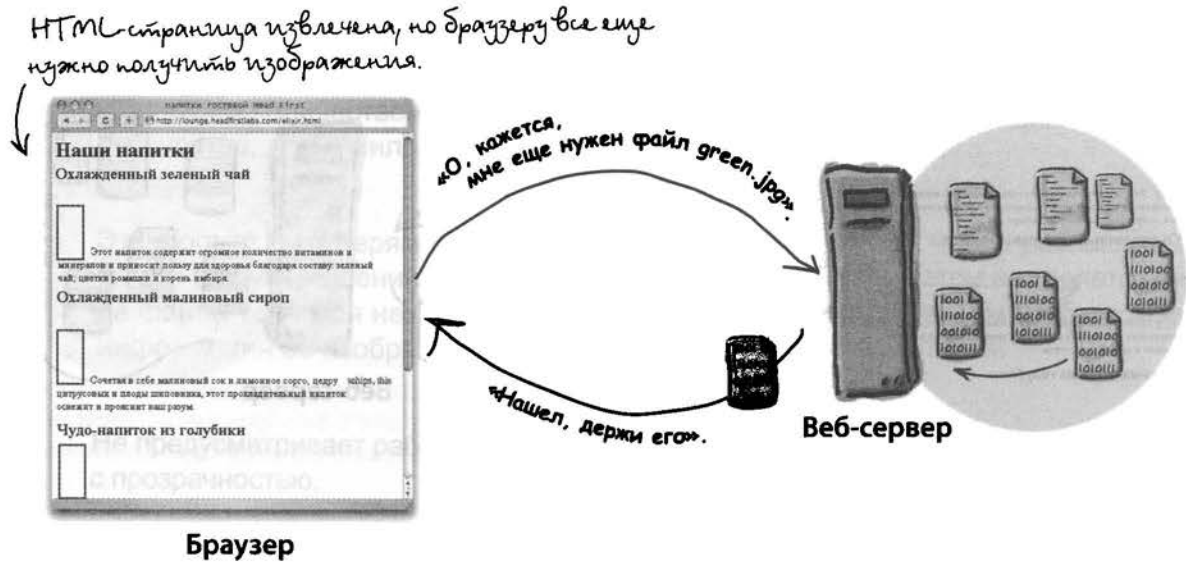


1 Сначала браузер извлекает файл elixir.html с сервера.

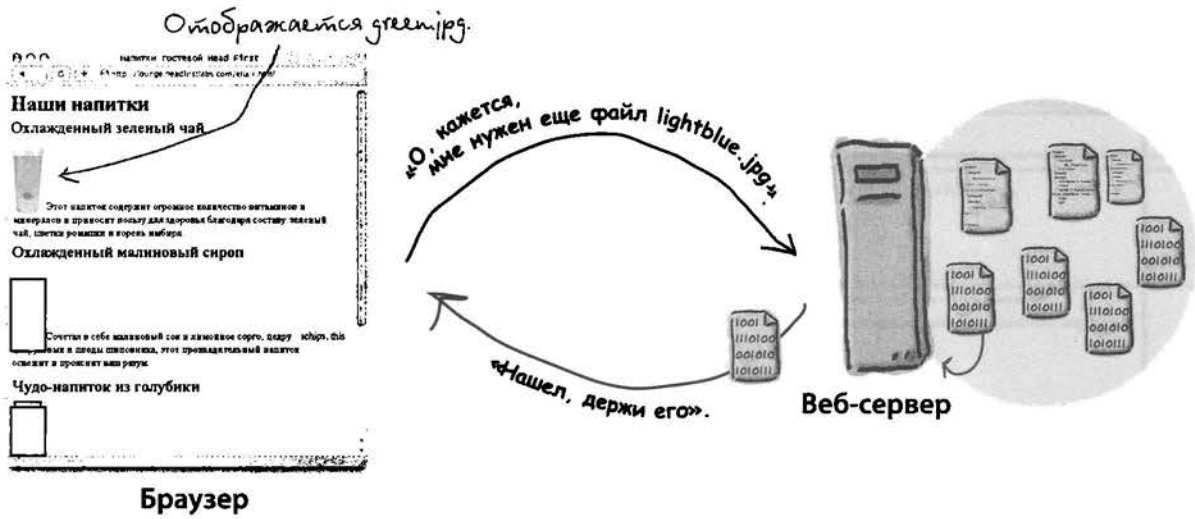


---

2 Затем браузер считывает файл elixir.html, отображает его и видит, что ему еще нужно найти четыре изображения. Итак, необходимо извлечь каждое из них с веб-сервера, начиная с green.jpg.

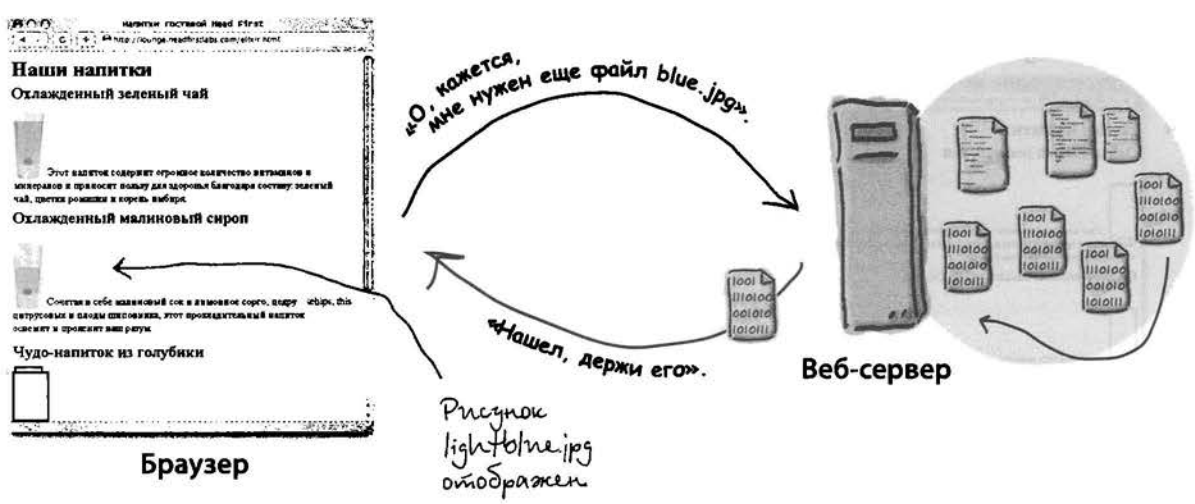


❸ Как только браузер извлекает файл green.jpg, он отображает его и переходит к следующему изображению: lightblue.jpg.



---

❹ Браузер извлек файл lightblue.jpg, выводит эту картинку, после чего переходит к следующей: blue.jpg. Этот процесс повторяется для каждого изображения на странице.



Как работают рисунки

Рисунки — это всего лишь рисунки, верно? На самом деле в мире существует огромное количество форматов изображений, и каждый имеет свои достоинства и недостатки. К счастью, в Сети обычно используются только два формата: JPEG и GIF. Единственная хитрость состоит в том, чтобы выяснить, какой из них и когда лучше использовать.

В чем разница между JPEG и GIF?



Используйте JPEG для фотографий и сложной графики



Используйте GIF для изображений с чистыми цветами, логотипов и геометрических фигур

Подходит для полутоновых нерастровых изображений, таких как фотографии.

Подходит для изображений с несколькими чистыми цветами и таких штриховых изображений, как логотипы, аппликации. Его хорошо применять, если в изображении есть небольшой текст.

Может выводить рисунки с большим количеством различных цветов, до 16 миллионов.

Может выводить рисунки с количеством цветов, не превышающим 256.

Это формат «с потерями», потому что при уменьшении размера файла теряется некоторая информация об изображении.

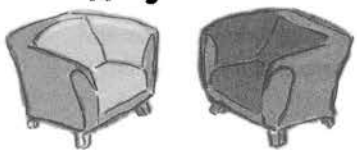
Сжимает файл, чтобы уменьшить его размер, но при этом не теряется никакая информация. Таким образом, это формат «без потерь».

Не предусматривает работу с прозрачностью.

Дает возможность установить «прозрачность» для цвета фона, чтобы сквозь этот фон было видно все, что находится под ним.

дальше ▸

Беседа у камина



Вечерний диалог: **JPEG и GIF сравнивают свои изображения.**

JPEG

Снова здравствуй, GIF. Кажется, я только что видел тебя на веб-странице?

Ха. Как только ты научишься хорошо показывать такие сложные изображения, как фотографии, я уверен, что люди будут счастливы работать только с тобой, но ты до сих пор не знаешь, как отобразить что-нибудь, для чего нужно больше чем несчастные 256 цветов.

Ты хочешь поговорить со мной о качестве? Я позволяю пользователям точно указать, какое качество им нужно.

Это так, но большинство людей это более чем устраивает. Не каждому на его страницах нужны изображения с супервысоким разрешением. С моей помощью пользователи обычно выбирают низкое или среднее качество изображения и остаются вполне довольны этим. А если они используют тебя, то у них получается огромный файл с таким же изображением.

GIF

Да... разве не было бы здорово, если бы все просто работали с GIF? Тогда мне бы не пришлось сталкиваться с тобой так часто.

Эй, нет ничего проще отображения фотографии, если *терять при этом ее качество.* Но для меня качество – самое главное. Если я не могу показать изображение полностью, я не возьмусь за это вообще. Только взгляни на парочку логотипов, которые ты пытался отобразить... Тьфу.

Да, но какой ценой? Согласись, чтобы сжать фотографию до размера, приемлемого для пересылки по Сети, тебе приходится ухудшить качество изображения.

Конечно, конечно, но смотрел ли ты когда-нибудь на штриховые изображения, логотипы, небольшой текст в изображении, чистые цвета? С JPEG они совсем не так хорошо выглядят.

JPEG

Да, конечно, GIF замечательно с ними справляется, но только до тех пор, пока количество цветов совсем невелико. Ты — это как всего лишь уменьшенная версия меня. Я умею делать все то, что умеешь ты.

Что ты сказал, GIF? У нас здесь шоу. Куда ты ушел?

Я думаю, ты придаешь слишком много значения прозрачности. Я могу просто встроить цвет фона в изображение.

Ну, я не очень-то об этом беспокоюсь; существует не так уж много фотографий без фона.

И когда же такое может случиться?

Да, верно. Работай со своими логотипами и рисунками с простым текстом, а я буду работать с фотографиями и сложными изображениями. Все знают, что я лучше подхожу для работы со сложными вещами.

GIF

(GIF исчезает, буквально.)

(GIF появляется снова.)

Без паники. Я просто доказываю свою точку зрения. Если JPEG так велик, то как же так случилось, что ты не можешь делать отдельные части своих изображений прозрачными, как я? С использованием прозрачности можно видеть то, что находится под изображением. Если мои пользователи хотят поместить логотип на своих веб-страницах, а фон страницы цветной, то они выберут меня, потому что знают, что я позволяю видеть фон страницы сквозь нецветные части логотипа.

Конечно, а затем кто-то поменяет цвет веб-страницы. И что получится? Прозрачность — это единственный подходящий выход в подобной ситуации. Тебе придется меня использовать.

А как насчет того, чтобы вырезать на рисунке человека или даже дерево и использовать это на веб-странице без фона?

Ты будешь удивлен, узнав, как часто мне приходится отображать фотографии, только потому, что пользователи хотят видеть прозрачный фон.

Эй, кто-то просит меня установить прозрачность... Я побежал.

КАКОЙ ФОРМАТ ИЗОБРАЖЕНИЯ?

Наши поздравления: вы были назначены лучшим по выбору форматов изображений. Для каждого из приведенных ниже рисунков выберите формат, который позволит лучше отобразить этот рисунок в Сети.

	JPEG	или	GIF
	☐		☐
	☐		☐
	☐		☐
	☐		☐
	☐		☐



Сейчас перейдем к официальному представлению: познакомьтесь с элементом <img>.

Мы достаточно долго откладывали процедуру знакомства. Как вы понимаете, с изображениями намного больше дел, чем с HTML-разметкой страницы. Так или иначе, хватит об этом... Настало время познакомиться с элементом **<img>**.

Начнем с того, что более тщательно рассмотрим этот элемент (хотя вы, вероятно, к данному моменту уже имеете представление, как он работает):



Итак, это все? Не совсем. Есть парочка атрибутов, о которых вы захотите узнать. И, конечно же, вы захотите узнать, как использовать элемент **<img>**, чтобы указать на изображения в Сети, которые находятся вне вашего сайта. Хотя в основном вы уже знаете, как работать с элементом **<img>**.

Сейчас мы расскажем вам о некоторых тонкостях использования элемента **<img>**, а затем вы попытаетесь применить полученные знания на практике.

<img>: теперь не только относительные ссылки

Атрибут `src` может быть использован не только для относительных ссылок; вы также можете вставить в него URL-адрес. Изображения хранятся в Сети вместе с HTML-страницами, поэтому у каждого изображения есть свой собственный URL.

URL-адрес обычно применяется, если вы указываете на изображение, находящееся на *другом* сайте (помните, что для ссылок и изображений, расположенных на *том же* сайте, лучше использовать относительные пути).

Вот как вы ссылаетесь на изображение, используя URL-адрес:

```
<img src="http://www.starbuzzcoffee.com/images/corporate/ceo.jpg">
```

Чтобы, используя URL, добавить на страницу рисунок, просто укажите весь адрес в качестве значения атрибута src.

URL — это путь к рисунку, поэтому имя файла в конце — всегда имя файла с данным рисунком. Нет такого понятия, как изображение, используемое по умолчанию.

Возьми в руку карандаш

Это упражнение на самом деле имеет отношение к карандашам (и к рисункам). В нем поднимается один простой вопрос. *У вас есть обычный и совершенно новый карандаш. Если вы нарисуете сплошную линию, исписав все острие карандаша, то какой длины будет эта линия?*

Какое это имеет отношение к изображениям? Чтобы найти ответ, вам придется написать небольшой HTML-код. Ответ на этот простой вопрос содержится в изображении, которое вы найдете по URL-адресу `http://www.headfirstlabs.com/trivia/pencil.gif`. Ваша задача — добавить изображение в приведенный код и получить ответ:

```
<html>
  <head>
    <title>Возьми в руку карандаш, простой вопрос</title>
  </head>
  <body>
    <p>Линию какой длины можно нарисовать обычным карандашом?</p>
    <p>
      
    </p>
  </body>
</html>
```

Поместите сюда элемент с изображением

Часто Задаваемые Вопросы

В: Итак, использовать элемент <img> достаточно легко. Он просто дает способ указать месторасположение изображения, которое должно быть выведено на странице?

О: Да, но это только краткое описание. Далее мы также поговорим о некоторых атрибутах, которые применяются вместе с этим элементом. Затем вы узнаете, как можно использовать CSS, чтобы поменять стиль изображения.

Но есть также много всего, что нужно знать о самих изображениях. Зачем нужны различные форматы? Когда нужно использовать один формат, а когда другой? Насколько большими должны быть рисунки? Как подготовить изображение к добавлению на веб-страницу?

В: Мы учили, что пустые элементы — это элементы без содержимого. Мы также учили, что элемент <img> — пустой. Но разве он не имеет содержимого (изображение)?

О: Если говорить точнее, то пустой элемент — это элемент, не имеющий содержимого *на HTML-странице*, которое заключено между закрывающим и открывающим тегами. Конечно, изображение — это содержимое, но элемент <img> лишь ссылается на него. Изображение само по себе не является частью HTML-страницы. Оно как бы заменяет элемент <img>, когда браузер отображает страницу. И не забывайте, что HTML-страницы исключительно текстовые, поэтому изображения никогда не смогут быть их непосредственной частью.

В: Насчет примера про загрузку изображений на веб-странице... Когда я загружал веб-страницу, я не заметил, чтобы изображения загружались по очереди. Почему?

О: Браузер часто извлекает все изображения *сразу*. Это значит, что он делает запросы на несколько рисунков одновременно. С учетом скорости компьютеров и сетей все это происходит достаточно быстро, поэтому вы обычно видите страницу сразу со всеми изображениями.

В: Если я вижу на веб-странице изображение, как мне определить его URL-адрес, чтобы можно было на него сослаться?

О: В большинстве браузеров есть возможность щелкнуть правой кнопкой мыши на изображении, чтобы появилось контекстное меню. Оно будет содержать пункт Copy Image Address (Скопировать адрес изображения) или Copy Image Link (Скопировать ссылку на изображение), выбор которых позволит поместить URL-адрес в буфер обмена. Второй способ найти URL — щелкнуть правой кнопкой мыши и выбрать в меню пункт Open Image in New Window (Открыть изображение в новом окне). В результате изображение откроется в окне браузера. Затем вы сможете взять URL рисунка из адресной строки браузера. Последний способ — использовать пункт меню браузера View Source (Посмотреть исходный код) и просмотреть HTML-код. Однако помните, что ссылка на изображение, которую вы найдете, может быть относительной и вам придется «реконструировать» URL, используя доменное имя сайта и путь к изображению.

В: Что делает JPEG-фотографию лучше фотографии GIF или GIF-логотип лучше логотипа JPEG?

О: Понятие «лучше» обычно определяется как комбинация качества изображения и размера файла. JPEG-фотография чаще всего намного меньше, чем фото в формате GIF такого же качества, в то время как логотип GIF обычно выглядит лучше и имеет меньший размер, чем тот же логотип в формате JPEG.

В: Я также слышал о формате изображений PNG. Почему вы не упомянули о нем?

О: PNG — это графический формат, который появился позже остальных. Он также достаточно интересен, так как может поддерживать и JPEG-, и GIF-стили изображений. Кроме того, он предоставляет больше возможностей по работе с прозрачностью, чем GIF. В настоящий момент PNG немного опережает события, потому что не все браузеры его поддерживают. Но его популярность растет очень быстро. Вы можете запросто использовать PNG, только знайте, что он будет работать не во всех браузерах.

Всегда имейте запасной вариант

То, в чем вы всегда можете быть уверены, когда имеете дело с Сетью, – что вы никогда не узнаете заранее, какие браузеры и устройства будут использоваться для просмотра вашей страницы. Всегда есть вероятность, что пользователи будут применять мобильные устройства, экранные дикторы, браузеры, которые работают с очень медленным Интернетом (и могут извлечь только текст сайта, но не изображения), мобильные телефоны, тостеры с выходом в Интернет... Кто знает?

Однако, несмотря на всю эту неопределенность, вы можете быть *ко всему подготовленными*. Даже если браузер не может показать изображения на вашей странице, есть альтернатива. Вы можете сообщить пользователю определенную информацию о том, что представляет собой изображение, благодаря атрибуту `alt` элемента **<img>**. Вот как это работает:

```
<img src="http://www.headfirstlabs.com/trivia/pencil.gif"
  alt="Длина линии, нарисованной карандашом, – 35 миль">
```

Атрибуту alt просто нужен небольшой кусок текста с описанием изображения.

Если изображение не может быть показано, то на его месте выводится этот текст.



Упражнение

В этом упражнении вы увидите, как браузер работает с атрибутом alt, когда появляется «отсутствующее» изображение. Теоретически, если рисунок не может быть найден, вместо него выводится значение атрибута `alt`. Однако не все браузеры делают это, поэтому ваши результаты в разных браузерах могут различаться. Рассмотрим, что вам нужно сделать.

1. Возьмите HTML-код из предыдущего упражнения.
2. Подкорректируйте элемент **<img>** так, чтобы в нем был атрибут `alt` со значением `Длина линии, нарисованной карандашом, – 35 миль`.
3. Поменяйте имя файла с изображением с `pencil.gif` на `broken.gif`. На самом деле такого файла не существует, поэтому вы получите «отсутствующее» изображение.
4. Обновите страницу в браузере.
5. И наконец, загрузите пару других браузеров и протестируйте страницу в них. Получились другие результаты?

Например, вы можете попробовать Firefox (http://www.mozilla.org/) или Opera (http://www.opera.com/).

Сверьте свои результаты с нашими, приведенными в конце главы.

Определение размеров изображений

Остался еще один атрибут элемента **<img>**, о котором вы обязательно должны знать. Точнее, это пара атрибутов: `width` и `height`. Вы можете использовать их, чтобы заранее указать браузеру размер изображения на вашей странице.

Рассмотрим, как используются атрибуты `width` и `height`:

```
<img src="images/drinks.gif" width="48" height="100">
```

Атрибут width указывает браузеру, изображение какой ширины должно появиться на странице.

Атрибут height указывает браузеру, какой длины должно быть изображение.

В обоих атрибутах размер задается в пикселах. Если вы не знаете, что такое пикселы, то сможете прочесть о них немного позже в этой главе. Атрибуты `width` и `height` можно задавать для любого изображения. Если не указать их, то браузер автоматически определит размер изображения перед тем, как вывести его на экран.

Часто Задаваемые Вопросы

В: Зачем вообще нужны эти атрибуты, если браузер все равно может вычислить размер изображений?

О: Если вы укажете ширину и высоту изображения в HTML-коде, то многие браузеры смогут распланировать структуру страницы перед тем, как ее отображать. В ином случае браузеру придется менять планировку страницы после того, как он узнает размеры изображения. Помните, что браузер загружает изображения уже *после* того, как загрузит HTML-файл и начнет отображать страницу. И если вы отдельно не укажете размер изображения, то браузер не будет его знать до тех пор, пока не загрузит рисунок.

Вы также можете указать значения высоты и ширины, которые меньше или больше, чем размеры изображения, и браузер установит пропорции рисунка так, чтобы они соответствовали новым размерам. Многие люди делают так, когда им нужно вывести в браузере существующее изображение, уменьшив или увеличив его реальный размер. Тем не менее, как вы убедитесь чуть позже, есть множество причин не использовать для этого атрибуты width и height.

В: Я должен использовать эти атрибуты только в паре? Могу ли я просто указать ширину или высоту?

О: Можете, но, указывая только один размер, вы оставляете для браузера такой же объем работы, как если бы вообще не указали ни ширины, ни высоты (и не так уж часто бывает полезно указывать лишь высоту или лишь ширину, если только вы не хотите масштабировать изображение до определенной высоты или ширины соответственно).

В: Вы много раз говорили, что HTML предполагается использовать для структуризации, а не для дизайна. А эти атрибуты больше похожи на те, что применяются для оформления веб-страниц. Я не прав?

О: В зависимости от того, как вы используете эти атрибуты. Если вы просто устанавливаете нужные размеры для изображения, то это действительно лишь информация Однако если вы используете width и height, чтобы изменить размеры изображения в браузере, то можно говорить, что вы применяете их для оформления веб-страницы. В этом случае, вероятно, лучше подумать об использовании CSS, что позволит добиться тех же результатов.

Создание сайта для самых больших фанатов: myPod

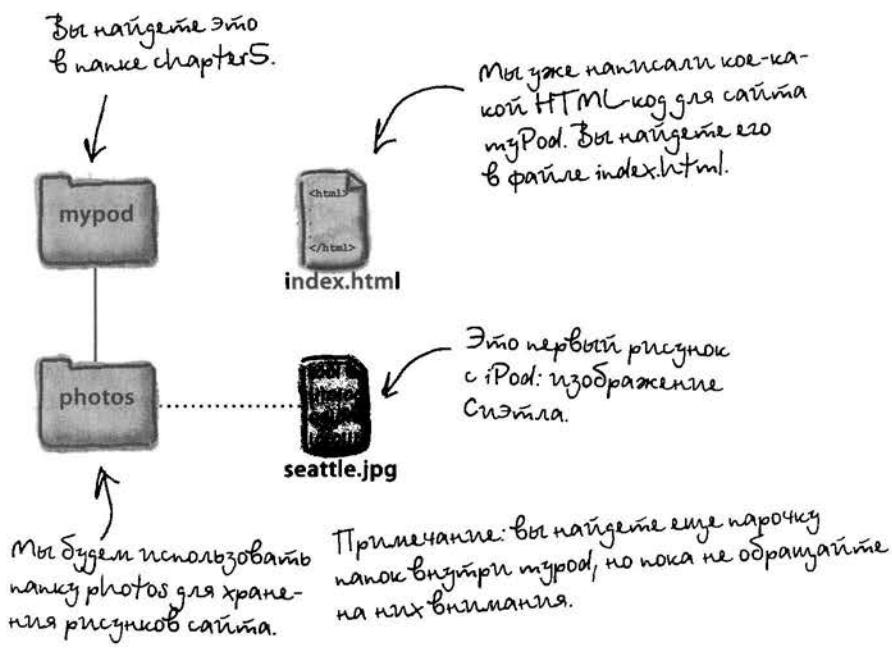
Обладатели iPod очень их любят и берут с собой повсюду. Представьте себе создание нового сайта под названием myPod, который будет содержать фотографии ваших друзей и их iPod в известных местах по всему миру.

Что вам нужно, чтобы приступить к делу? Всего лишь некоторые знания по HTML, парочка изображений и любовь к вашему iPod.

Мы уже написали кое-какой HTML-код для этого сайта, но еще не добавили рисунки. Именно тут вы приступаете к работе. Но перед тем, как вы доберетесь до рисунков, позвольте нам немного вам помочь; найдите папку chapter5 в файлах с примерами для книги. Здесь вы найдете папку mypod, которую нужно открыть. Вот что вы увидите внутри.



Мой iPod в Сиэтле! Здесь видны дождевые облака и башня Спейс Нидл. И не видно 628 кафе



Доработка файла index.html для сайта myPod

Открыв файл `index.html`, вы увидите, что работа над страницей myPod уже начата. Вот HTML-код, который есть на данный момент:

```
<html>
  <head>
    <title>myPod</title>
    <style type="text/css">
      body { background-color: #eaf3da;}
    </style>
  </head>
  <body>

    <h1>Добро пожаловать в myPod</h1>
    <p>
      Добро пожаловать туда, где можно похвастаться своим iPod и рассказать, где
      вы с ним были. Хотите повеселиться вместе с нами? Все, что вам для этого
      нужно, — это любой iPod: от ранней классической версии до самой современной
      нановерсии — самого маленького iPod с фотографиями лучшего качества
      и с цифровой камерой. Просто сфотографируйте свой iPod в любимом месте, и мы
      будем рады поместить этот снимок на сайте myPod. Итак, чего же вы ждете?
    </p>

    <h2>Сиэтл, Вашингтон</h2>
    <p>
      Я и мой iPod в Сиэтле! Здесь видны дождевые облака и башня Спейс Нидл.
      И не видно 628 кафе.
    </p>


  </body>
</html>
```



Мы вставили сюда готовый CSS-код. Пока просто оставьте его таким. Все, что он делает, — придает странице светлый голубой фон. Обещаем, что начнем разбирать CSS через парочку глав!

Этот HTML-код выглядит привычно, так как использованы основные строительные блоки: <h1>, <h2> и <p>.

А вот как это выглядит в браузере. Неплохо, но нам нужны рисунки.



Возьми в руку карандаш

Как вы понимаете, уже написана большая часть HTML-кода, чтобы сайт myPod заработал. Все, что вам осталось сделать, — добавить элемент **<img>** для каждой фотографии, которую вы хотите на нем разместить. Пока есть только одна фотография `seattle.jpg`, так что добавьте нужный элемент, чтобы поместить это изображение внизу страницы. Когда вы с этим справитесь, загрузите страницу в браузере и оцените виды Сиэтла.



Здесь вы должны поместить свою первую фотографию.

```
<html>
  <head>
    <title>myPod</title>
    <style type="text/css">
      body { background-color: #eaf3da;}
    </style>
  </head>
  <body>

    <h1>Добро пожаловать в myPod</h1>
    <p>
      Добро пожаловать туда, где можно похвастаться своим iPod и рассказать,
      где вы с ним были. Хотите повеселиться вместе с нами? Вам для этого нужен
      любой iPod: от ранней классической версии до самой современной нановерсии –
      самого маленького iPod с фотографиями лучшего качества и с цифровой
      камерой. Просто сфотографируйте свой iPod в любимом месте, и мы будем рады
      поместить этот снимок на сайте myPod. Итак, чего же вы ждете?
    </p>

    <h2>Сиэтл, Вашингтон</h2>
    <p>
      Я и мой iPod в Сиэтле! Здесь видны дождевые облака и башня Спейс Нидл.
      И не видно 628 кафе.
    </p>

    <p>

    </p>

  </body>
</html>
```

Ваш элемент <img> должен появиться здесь.

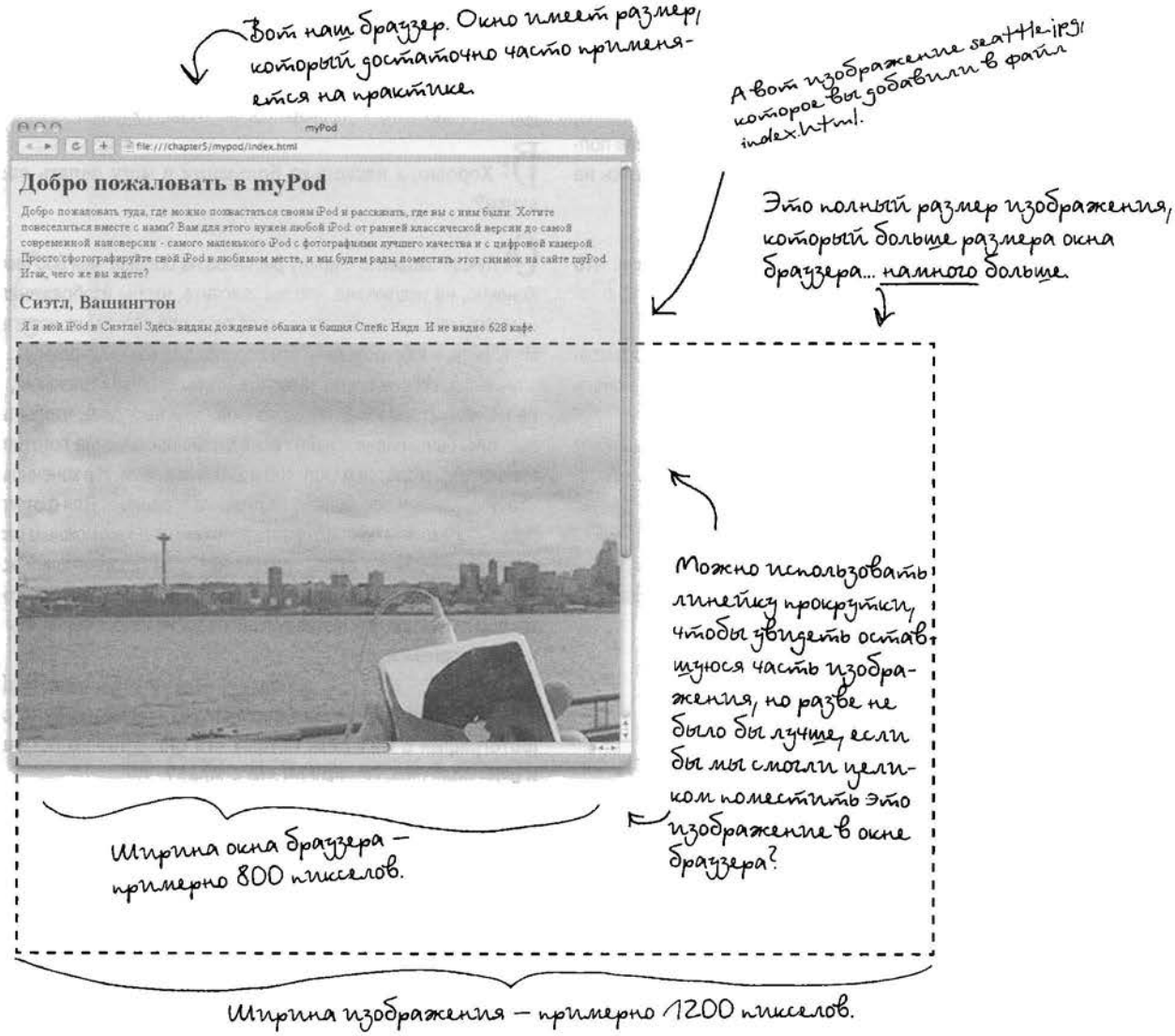
Ого! Рисунок слишком большой

Итак, изображение находится как раз там, где должно, но оно *слишком большое*. И, кстати, почти все фотографии, сделанные современными цифровыми камерами, такие же большие (и даже больше). Может быть, нам оставить изображение таким, какое оно есть, и предположить, что посетители сайта будут использовать линейку прокрутки? Дальше вы увидите, что это плохая идея.



Если изображение отлично вписывается в окно, значит, в браузере включена настройка автоподбора размера изображений. Подробнее об этом через минуту...

Давайте взглянем на изображение и на браузер и поймем, насколько все плохо выглядит.



Часто Задаваемые Вопросы

В: Что плохого в том, чтобы посетитель использовал линейку прокрутки для просмотра всего изображения целиком?

О: Веб-страницами с большими изображениями пользоваться очень неудобно. Дело не только в том, что посетители не могут видеть весь рисунок целиком, — само по себе использование линейки прокрутки обременительно. Кроме того, при работе с большими изображениями сервер и браузер обмениваются большими объемами данных, что занимает много времени и может привести к тому, что ваша страница будет медленно грузиться, особенно у тех, кто использует соединение по телефонной линии и другие медленные подключения.

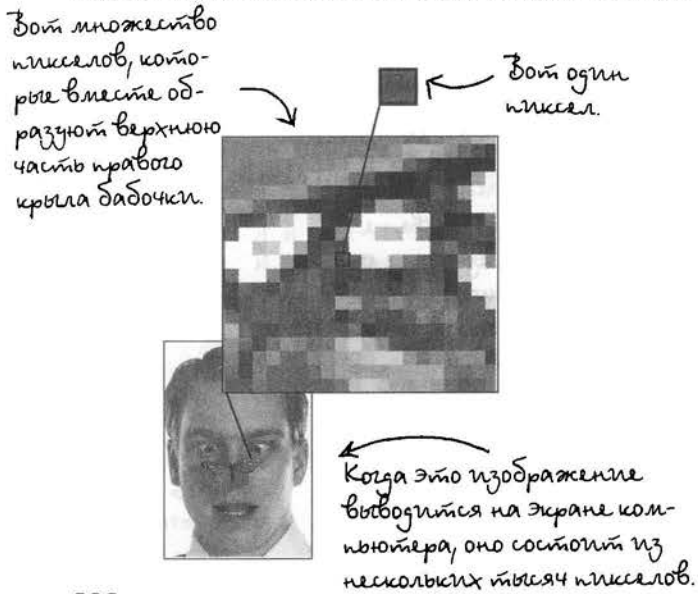
В: Почему я не могу просто использовать атрибуты width и height, чтобы поменять размеры изображений на странице?

О: Потому что браузеру все-таки нужно извлечь изображение полностью перед тем, как уменьшить его, чтобы оно поместилось на вашей странице.

В: Вы сказали, что ширина окна браузера 800 пикселов. Что это означает?

О: Экран монитора состоит из миллионов точек, называемых пикселами. Вы сможете их увидеть, если посмотрите на монитор с очень близкого расстояния.

В то время как размеры экранов мониторов и их разрешения имеют тенденцию различаться (у некоторых людей мониторы большие, у некоторых маленькие), 800 пикселов — это наиболее часто используемая ширина окна браузера, которую люди для себя устанавливают. Итак, 800 пикселов — это та *максимальная* ширина ваших изображений, которой хорошо бы придерживаться на практике (мы вернемся к этому в следующих главах).



В: Какова зависимость размера изображения на экране от количества пикселов?

О: Примерно такова: 72 пиксела на каждый дюйм, хотя, в зависимости от вашего монитора, может быть и 120 пикселов на дюйм. Допустим, разрешение вашего экрана составляет 72 пиксела на каждый дюйм. Если вы хотите, чтобы изображение было приблизительно 3 дюйма в ширину и высоту, то зададите размер, равный 72 (пиксела) × 3 (дюйма) = 246 пикселов, или, округлив, 250 × 250 пикселов.

В: Хорошо, а насколько большими я могу делать свои рисунки?

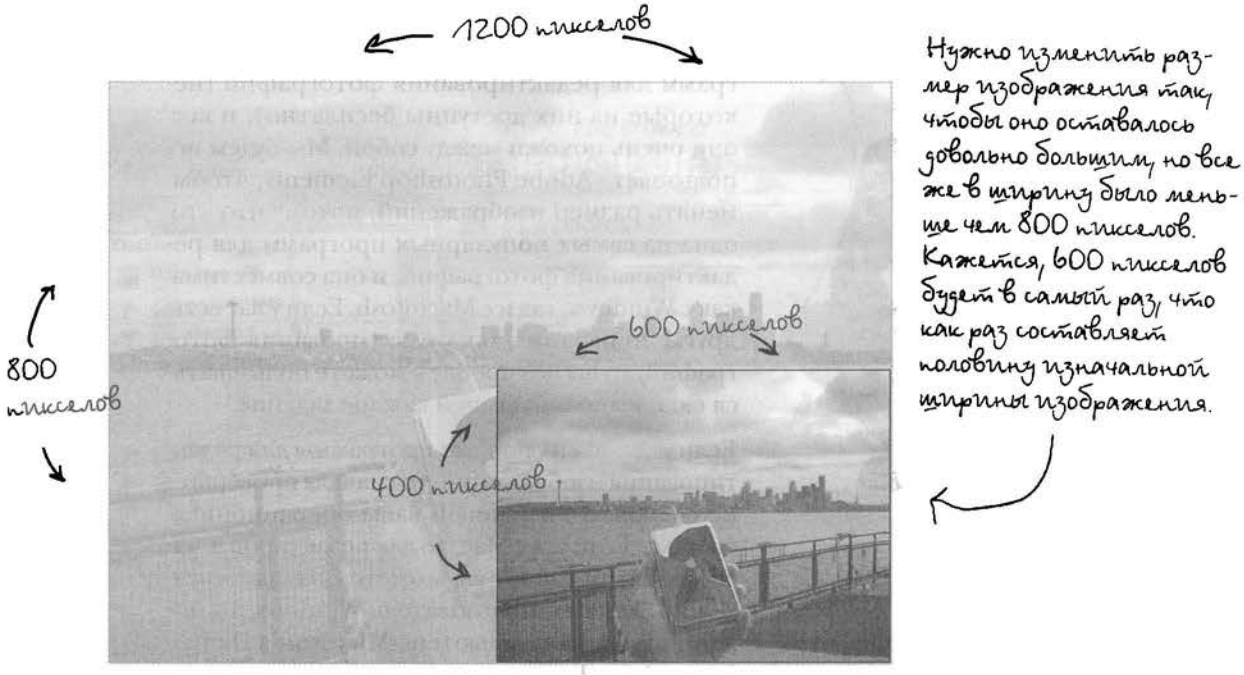
О: Лучше задавать ширину рисунков не больше чем 800 пикселов. Конечно, не исключено, что вы захотите, чтобы изображения были намного меньше, — это зависит от того, для чего вы их используете. Например, изображение — это логотип для вашей страницы. В этом случае вы, скорее всего, захотите, чтобы он был маленький, но тем не менее читаемый. В конце концов, вам не нужно, чтобы ширина логотипа была равна ширине всей страницы. Ширина логотипов, как правило, колеблется между 100 и 200 пикселами. И в конечном счете ответ на ваш вопрос зависит от дизайна страницы. Для фотографий, которые вы хотите просматривать с максимально возможным разрешением, можно создать страницу с миниатюрами изображений, которые быстро загружаются, и дать возможность пользователю щелкнуть на каждом из них для просмотра большой версии изображения. Мы вскоре к этому вернемся.

В: Мне кажется, что браузер автоматически изменил размер фотографии с Сиэтлом, потому что она отлично помещается в окне браузера. Почему он это сделал?

О: У некоторых браузеров есть возможность изменения размера изображения, если его ширина больше ширины окна. Но многие браузеры такого не делают, поэтому не нужно на это полагаться. Если бы даже у каждого браузера было такое свойство, все равно браузер и сервер обменивались бы намного большим количеством данных, чем необходимо, а это замедлило бы загрузку страницы и сделало бы ее менее практичной.

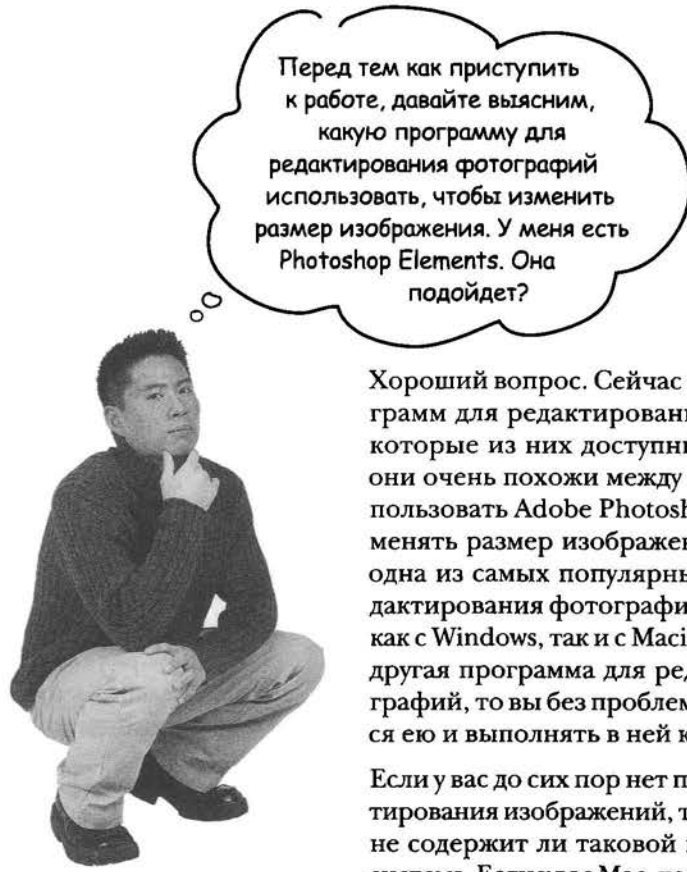
Изменение размера изображения

Давайте изменим размер этого рисунка так, чтобы он помещался в окно браузера. Сейчас размеры такие: 1200 пикселов в ширину и 800 пикселов в высоту (через минутку вы узнаете, как это определить). Поскольку мы хотим, чтобы ширина рисунка была не более 800 пикселов, мы должны точно определить, изображение какой ширины подойдет для нашей веб-страницы myPod наилучшим образом. Весь смысл страницы заключается в просмотре фотографий iPod, поэтому мы, скорее всего, захотим, чтобы изображения были достаточно большими. Если мы уменьшим размеры рисунков до 600 пикселов в ширину и 40 пикселов в высоту, то они все равно будут занимать бóльшую часть ширины окна браузера, но появятся отступы по бокам. Звучит хорошо? Давайте изменим размер этого изображения...



Вот что Вам нужно сделать.

1. **Откройте рисунок в программе для редактирования фотографий.**
2. **Уменьшите размер изображения в два раза (до 600×400 пикселов).**
3. **Сохраните рисунок под названием seattle_med.jpg.**



Хороший вопрос. Сейчас есть множество программ для редактирования фотографий (некоторые из них доступны бесплатно), и все они очень похожи между собой. Мы будем использовать Adobe Photoshop Elements, чтобы менять размер изображений, потому что это одна из самых популярных программ для редактирования фотографий, и она совместима как с Windows, так и с Macintosh. Если у вас есть другая программа для редактирования фотографий, то вы без проблем можете пользоваться ею и выполнять в ней каждое задание.

Если у вас до сих пор нет программы для редактирования изображений, то сначала проверьте, не содержит ли таковой ваша операционная система. Если у вас Mac, то для редактирования своих фотографий вы можете пользоваться iPhoto. Если вы пользователь Windows, то можете найти на компьютере Microsoft's Digital Image Suite. Если же у вас все-таки нет подходящей программы, все равно продолжайте работу и для каждого шага используйте HTML-код и рисунки из папок с примерами.

Если у вас нет Adobe Photoshop Elements, но вы хотите использовать именно эту программу, чтобы продолжить обучаться по этой главе, можете скачать пробную версию и пользоваться ею в течение 30 дней. Вот адрес, где вы сможете ее найти: http://www.adobe.com/products/tryadobe/main.jsp.

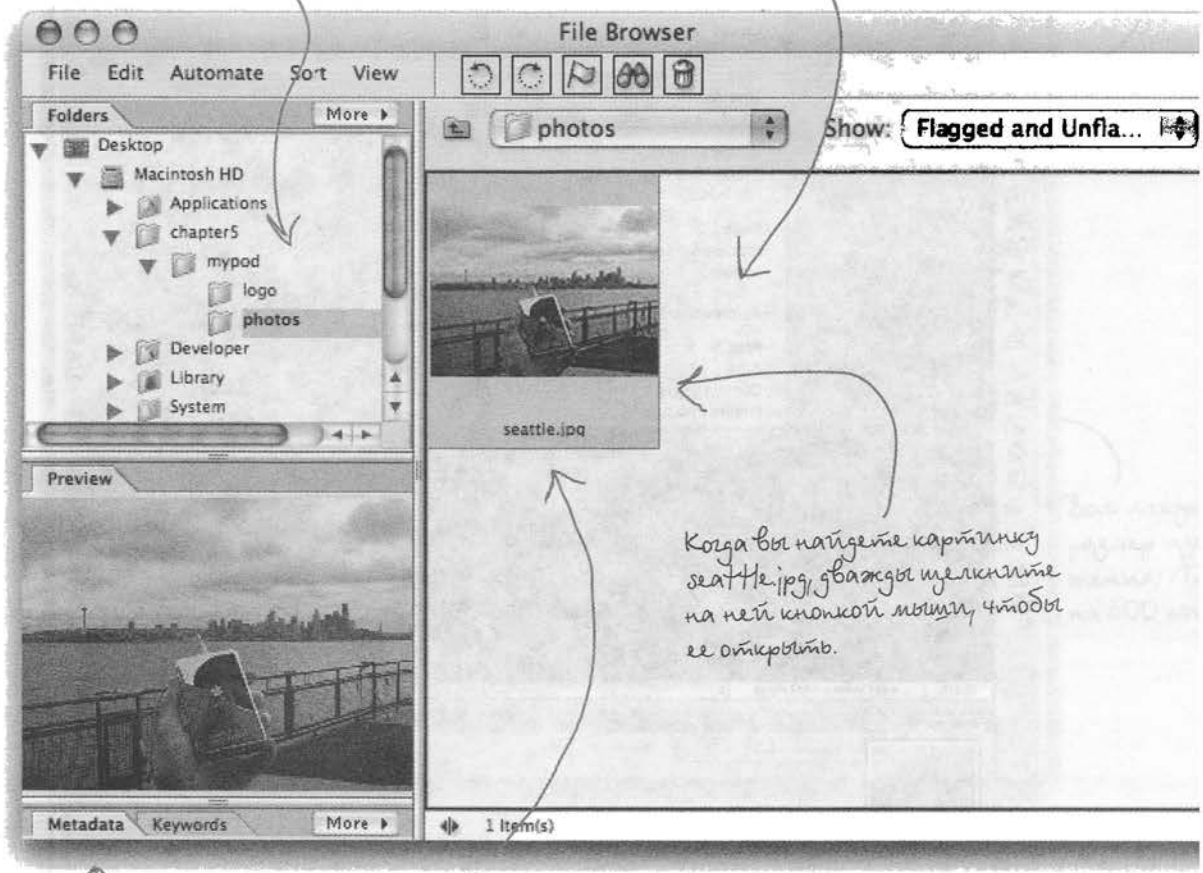
Открытие изображения

Во-первых, запустите программу для редактирования изображений и откройте в ней файл `seattle.jpg`. В Photoshop Elements вам нужно будет выбрать пункт **Browse Folders** (Открыть) в меню **File** (Файл), после чего откроется окно **File Browser** (Открыть). Вы будете его использовать, чтобы найти файл `seattle.jpg` в папке `chapter5/mypod/photos`.

Ах да, если вы работаете в Windows, то сразу используйте меню File>Open (Файл>Открыть), чтобы открыть картинку seattle.jpg.

Вот окно File Browser (Открыть). Используйте его, чтобы найти картинку seattle.jpg.

Перемещаясь по папкам, вы будете видеть в них эскизы изображений.



Когда вы найдете картинку seattle.jpg, дважды щелкните на ней кнопкой мыши, чтобы ее открыть.

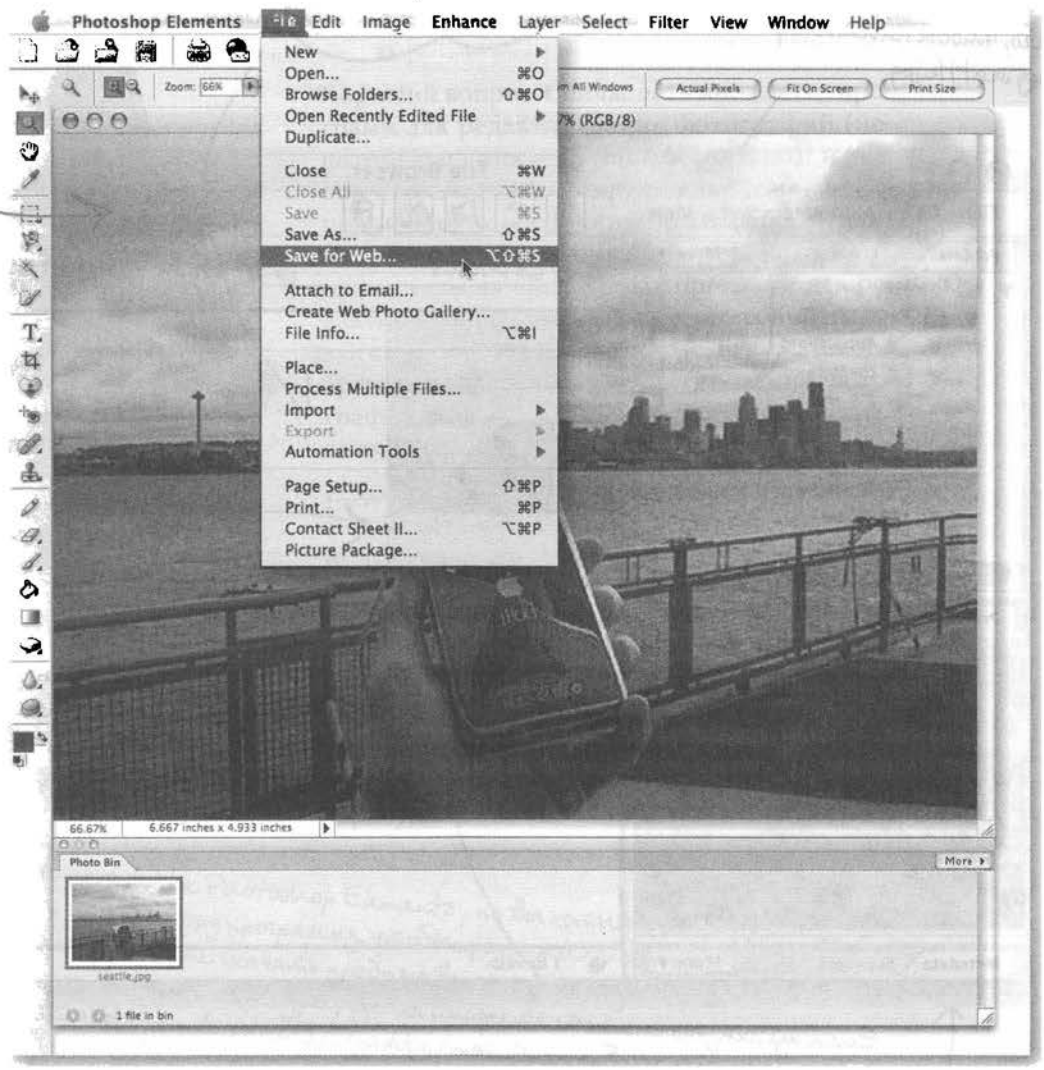
Если щелкнуть кнопкой мыши на эскизе изображения, то появится его большой вариант.

Изменение размера изображения

Теперь, когда файл `seattle.jpg` открыт, для изменения размера рисунка и его сохранения мы будем использовать окно **Save for Web** (Сохранить для Сети). Чтобы открыть это окно, просто выберите пункт **Save for Web** (Сохранить для Сети) в меню **File** (Файл).

Это изображение seattle.jpg в Photoshop Elements.

Чтобы изменить размер изображения, выберите пункт Save for Web (Сохранить для Сети) в меню File (Файл).

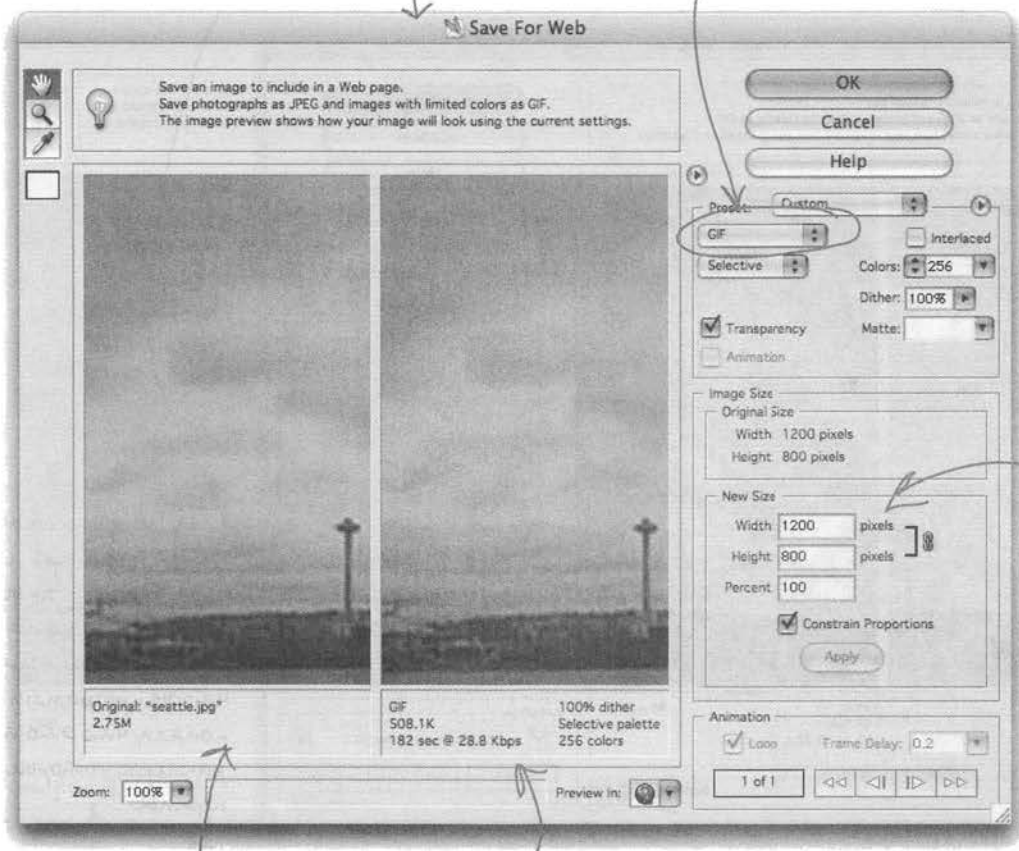


Изменение размера изображения, продолжение...

Выбрав пункт меню Save for Web (Сохранить для Сети), вы увидите одноименное окно; давайте познакомимся с ним перед тем, как начать использовать.

В этом окне вы можете делать множество интересных вещей. На данный момент мы сфокусируемся на том, как его использовать для изменения размера изображения и сохранения в формате JPEG для веб-страницу.

Здесь вы выбираете формат для сохранения файлов. Сейчас выбран GIF, но мы поменяем его на JPEG через пару страницу.

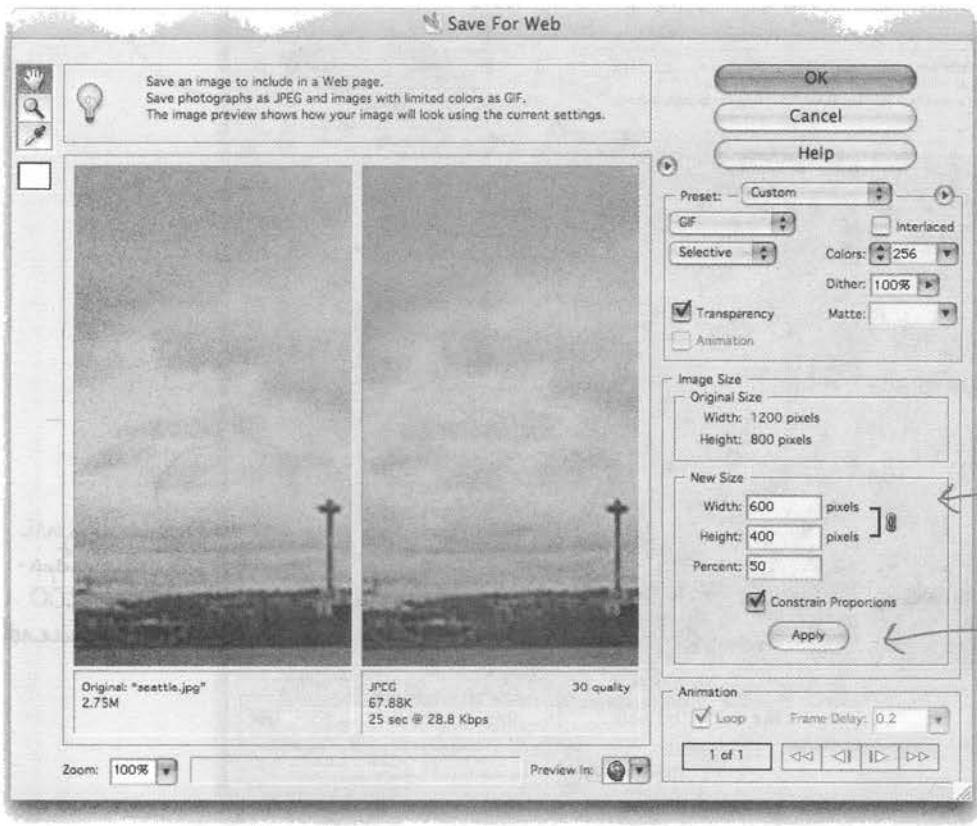


Вот текущий размер изображения: 1200 на 800 пикселов.

В окне слева показывается ваше первоначальное изображение, а справа — изображение в том формате, в котором вы будете сохранять его для Сети. Сейчас в нем показан формат GIF; далее мы поменяем его на JPEG.

Как видите, это окно содержит множество функциональных возможностей. Давайте начнем ими пользоваться. Чтобы изменить размер изображения, нужно изменить ширину на 600 пикселов и высоту на 400 пикселов. Затем следует сохранить изображение в формате JPEG. Начнем с изменения размера изображения...

(1) Измените размер изображения так: ширина – 600 и высота – 400. Если у вас установлен флажок Constrain Proportions (Сохранить пропорции), то все, что нужно сделать, – ввести ширину 600, а программа автоматически поменяет высоту на 400.



(2) После того как высота и ширина корректно заданы, нажмите кнопку Apply (Применить), чтобы программа поняла, что это те размеры, которые вам нужны.

Это повлияет не на первоначальное изображение, а на файл, который вы хотите сохранить.

Вы должны нажать кнопку Apply (Применить), чтобы уменьшить размер изображения; в противном случае оно сохранится с исходными шириной и высотой.

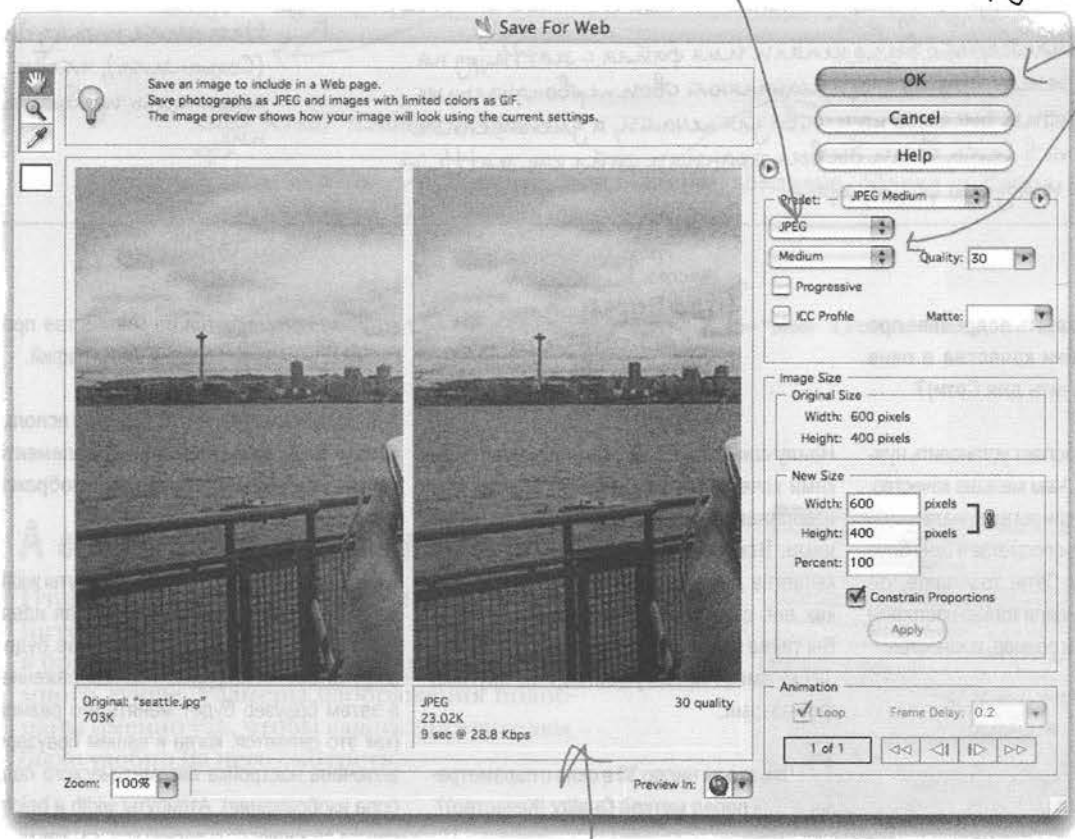
Размеры изменены, теперь сохраняем

Сейчас вам всего лишь нужно сохранить изображение в правильном формате (JPEG). Для этого следует выбрать данный формат и установить качество **Medium** (Среднее). Подробнее о качестве мы поговорим немного позже.

(1) Теперь, когда установлен размер изображения, остается выбрать для него формат. Сейчас выбран GIF; поменяйте его на JPEG, как мы это здесь сделали.

(2) Установите качество Medium (Среднее).

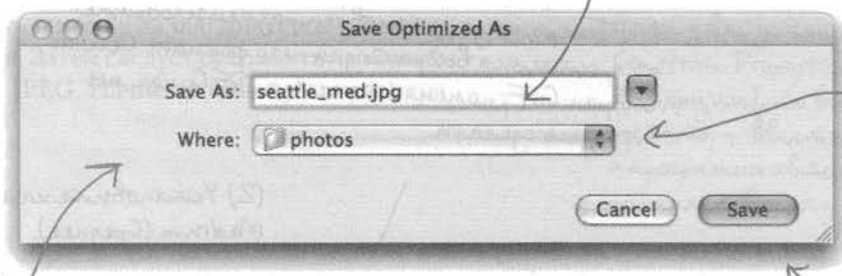
(3) Это все. Нажмите кнопку OK и переходите к следующей странице.



Обратите внимание, что после того, как вы нажали кнопку Apply (Применить) в предыдущем шаге, размер изображения был изменен и оно отобразилось с новыми размерами.

Сохранение изображения

После того как вы нажмете кнопку OK, появится окно для сохранения файла. Сохраните рисунок под именем `seattle_med.jpg`, чтобы не потерять первоначальную фотографию.



Измените имя файла на seattle_med.jpg.

Убедитесь, что вы сохраняете изображение в папку typod, photos.

Нажмите кнопку Save (Сохранить), чтобы сохранить изображение.

Обратите внимание, что вы поменяли имя файла с seattle.jpg на seattle_med.jpg. Зачем? Часто люди сохраняют свои первоначальные большие фотографии высокого качества для печати, а уменьшенные копии помещают в Сеть. Если бы вы сохранили файл как seattle.jpg, то потеряли бы исходную фотографию!

Часто Задаваемые Вопросы

В: Вы можете рассказать подробнее про параметры настройки качества в окне Save for Web (Сохранить для Сети)?

О: Формат JPEG позволяет установить нужный уровень качества. Чем меньше качество, тем меньше размер. Если вы взглянете на панель предварительного просмотра в окне Save for Web (Сохранить для Сети), то увидите, что после того, как вы поменяли только настройки качества, изменились и размер, и качество.



Формат изображения.

Размер изображения (1 Кбайт по размеру эквивалентен 1024 байтам).

Photoshop Elements даже говорит вам, как много времени займет пересылка файла браузеру при соединении по телефонной линии.

Наилучший способ разобраться с настройками качества и различными форматами изображений — поэкспериментировать над ними. Вскоре вы поймете, какие уровни качества для каких изображений и на каких веб-страницах лучше использовать. Вы также узнаете, в каких случаях формат JPEG имеет преимущество над другими форматами.

В: Что это за число 30 в окне с параметрами JPEG перед меткой Quality (Качество)?

О: 30 — это число, которое Photoshop Elements рассматривает как качество Medium (Среднее). На самом деле JPEG использует шкалу от 1 до 100 %, а Low (Низкое), Medium (Среднее), High (Высокое) и т. д. — всего лишь предварительно заданные значения, которые применяются во множестве программ для редактирования фотографий.

В: Нельзя ли вместо этого просто использовать атрибуты width и height элемента <img>, чтобы изменить размер изображения?

О: Вы можете использовать атрибуты width и height, но это не совсем хорошая идея. Почему? Потому что в этом случае будет загружаться полноформатное изображение, а затем браузер будет менять его размер (как это делается, когда в вашем браузере включена настройка автоматического подбора изображения). Атрибуты width и height просто помогают браузеру выяснить, как много места нужно оставить для изображения; и если вы их используете, то они должны соответствовать реальным ширине и высоте изображения.

Исправление HTML-кода для myPod

После того как вы сохранили изображение, можете выйти из программы Photoshop Elements. Теперь все, что вам остается сделать, – внести кое-какие изменения в страницу сайта myPod `index.html`, чтобы она содержала новую версию фотографии: `seattle_med.jpg`.

Рассмотрим фрагмент файла `index.html`, содержащий только те части, которые нужно изменить.

```
<html>
  <head>
    <title>myPod</title>
    <style type="text/css">
      body { background-color: #eaf3da;}
    </style>
  </head>
  <body>
    .
    .
    .
    <h2>Сиэтл, Вашингтон</h2>
    <p>
      Я и мой iPod в Сиэтле! Здесь видны дождевые облака и башня Спейс Нидл.
      И не видно 628 кафе.
    </p>

    <p>
      <img src="photos/seattle_med.jpg" alt="Мой iPod в Сиэтле, Вашингтон">
    </p>

  </body>
</html>
```

Здесь будет остальная часть HTML-кода. Она у вас уже есть в файле index.html.

Все, что вам нужно сделать, – поменять имя файла в элементе <img> на имя файла, который вы только что создали: seattle_med.jpg.

А сейчас протестируем...

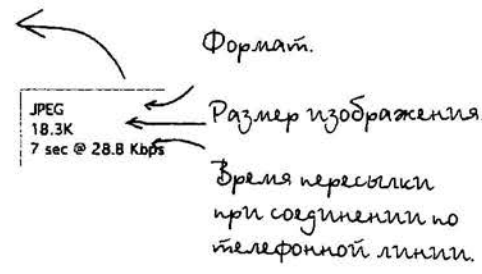
Итак, вперед! Внесите все изменения, сохраните их и обновите страницу `index.html` в браузере. Теперь все должно выглядеть намного лучше. Размеры изображения подобраны именно так, чтобы вашим посетителям было удобно на него смотреть.



Теперь изображение отлично помещается в окне браузера.

КАКОЙ ФОРМАТ ИЗОБРАЖЕНИЯ?

Новое задание для вас: откройте файл `chapter5/testimage/eye.jpg` в Photoshop Elements. Откройте окно Save for Web (Сохранить для Сети) и заполните пропуски (см. ниже), выбирая каждую настройку качества для JPEG (низкое, среднее, высокое и т. д.). Нужную информацию вы найдете в панели предварительного просмотра, которая находится под самим изображением. Когда справитесь с этим, определите, какая настройка лучше всего подходит для данного изображения.



	Формат	Качество	Размер	Время	Победитель
	JPEG	Максимальное			☐
	JPEG	Очень высокое			☐
	JPEG	Высокое			☐
	JPEG	Среднее			☐
	JPEG	Низкое			☐
	GIF	Не применяется			☐

Еще больше фотографий для myPod

Для myPod поступила новая партия фотографий: по две из Сиэтла и от друга из Великобритании. Размеры фотографий уже были уменьшены так, что их ширина не превышает 800 пикселов. Добавьте для них элементы **<img>** (вы найдете фотографии в папке photos).

```
<html>
  <head>
    <title>myPod</title>
    <style type="text/css">
      body { background-color: #eaf3da;}
    </style>
  </head>
  <body>

    <h1>Добро пожаловать в myPod</h1>
    <p>
      Добро пожаловать туда, где можно похвастаться своим iPod и рассказать, где
      вы с ним были. Хотите повеселиться вместе с нами? Вам для этого нужен
      любой iPod: от ранней классической версии до самой современной нановерсии –
      самого маленького iPod с фотографиями лучшего качества и с цифровой камерой.
      Просто сфотографируйте свой iPod в любимом месте, и мы будем рады поместить
      этот снимок на сайте myPod. Итак, чего же вы ждете?
    </p>

    <h2>Сиэтл, Вашингтон</h2>
    <p>
      Я и мой iPod в Сиэтле! Здесь видны дождевые облака и башня
      Спейс Нидл. И не видно 628 кафе.
    </p>

    <p>
      <img src="photos/seattle_med.jpg" alt="Мой iPod в Сиэтле, Вашингтон">
      <img src="photos/seattle_shuffle.jpg" alt="iPod в Сиэтле, Вашингтон">
      <img src="photos/seattle_downtown.jpg" alt="iPod в бизнес-центре Сиэтла,Вашингтон">
    </p>

    <h2>Бирмингем, Англия</h2>
    <p>
      Это несколько фотографий из Бирмингема. Мы повстречали здесь несколько
      людей, страстно влюбленных в свои iPod. Взгляните на классические британские
      красные телефонные будки!
    </p>

    <p>
      <img src="photos/britain.jpg" alt="iPod в Бирмингеме в телефонной будке">
      <img src="photos/applestore.jpg" alt="iPod в Бирмингеме в Apple store">
    </p>
  </body>
</html>
```

Если хотите, можете добавить сюда пару своих фотографий. Только не забудьте сначала изменить их размер.

Давайте будем хранить все фотографии из Сиэтла вместе.

То же самое для фотографий из Бирмингема...

Еще один тест для myPod

Подумайте только, какое значение имеют несколько изображений! Страница начинает совершенно по-другому выглядеть.

Итак, у вас есть много изображений на странице, и вы уже изменили их размеры, но они все равно остаются очень большими. Мало того, что страница будет загружаться все медленнее, потому что вы добавляете больше изображений, но и пользователь должен постоянно прокручивать ее, чтобы увидеть все фотографии. Подумайте, не будет ли лучше, если пользователи смогут сначала видеть маленькое изображение – эскиз – каждой фотографии, а затем щелкать на нем кнопкой мыши, чтобы просмотреть большое изображение?



Это страница со всеми изображениями в их полном размере.

Вот как страница выглядит теперь.



Доработка сайта таким образом, чтобы использовались эскизы

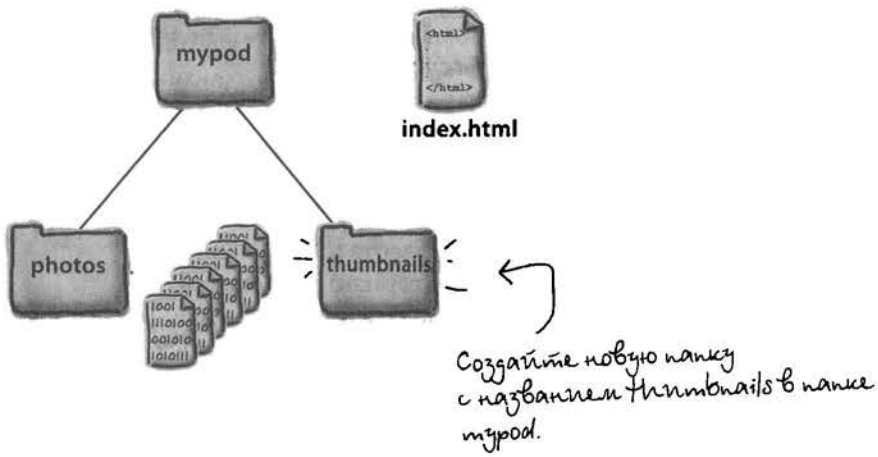
Сейчас вы сделаете вашу страницу более удобной в использовании, заменив имеющиеся фотографии меньшими по размеру (которые мы будем называть *эскизами*). После этого вы создадите ссылки с каждого эскиза на большую фотографию. Рассмотрим, как это сделать.

1. Создайте новый каталог для малых версий изображений.
2. Измените размер каждой фотографии до 150×100 пикселов и сохраните в папку thumbnail.
3. Укажите на эскизы фотографий в атрибутах `src` каждого элемента **`<img>`** файла `index.html`.
4. Добавьте ссылку с каждого эскиза на новую страницу с большой фотографией.

Создание нового каталога для эскизов

Чтобы содержать файлы в порядке, создайте отдельную папку для уменьшенных версий изображений. В противном случае все закончится тем, что у вас будет папка, в которой в одну кучу свалены и большие фотографии, и их уменьшенные копии. В результате, когда будет добавлено значительное количество фотографий, легко можно будет запутаться в них.

Создайте папку с названием thumbnails в папке mypod. Если вы работаете с файлами, приведенными в качестве примеров к книге, то такая папка там уже есть.



Создание эскизов

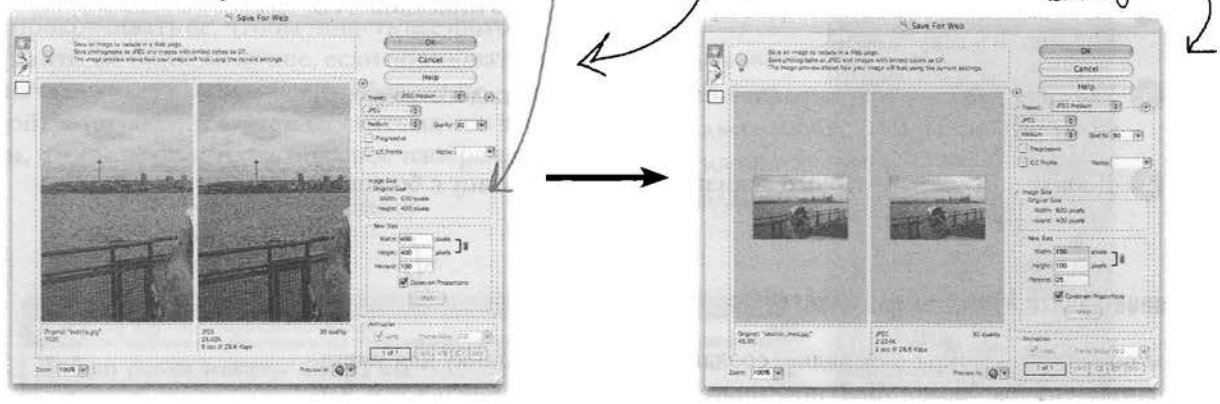
Теперь у вас есть место, куда вы будете складывать эскизы, поэтому можно их создать. Начните с открытия файла `seattle_med.jpg` в программе для редактирования фотографий. Измените размер фотографии до 150×100 пикселов, используя тот же метод, которым уменьшали фотографии до размера 600×400 пикселов.

В Photoshop Elements выберите пункт меню Save for Web (Сохранить для Сети).

Измените ширину на 150, а высоту на 100 и нажмите кнопку Apply (Применить).

Не забудьте поменять формат на JPEG, установив качество Medium (Среднее).

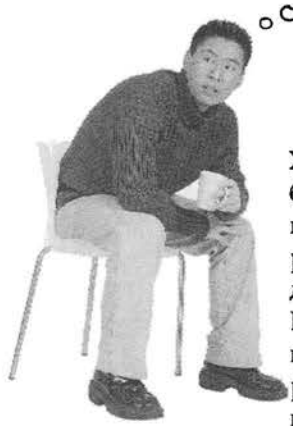
Нажмите кнопку OK.



Изменив размер изображения, нажмите OK и сохраните его с таким же именем, но в папке thumbnail. Будьте осторожны: если вы сохраните этот файл в папке photos, то замените им большую фотографию.

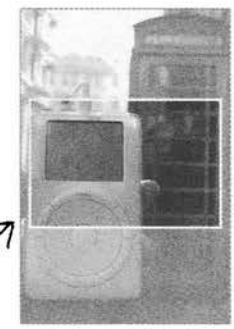
Если вы работаете с файлами, приведенными в качестве примеров к книге, то найдете все эскизы фотографий в папке thumbnails и вам не придется всего этого делать (в конце концов, вы учите HTML, а не занимаетесь обработкой фотографий).

Теперь повторите это для каждой фотографии из папки photos.



Как насчет фотографий из Бирмингема? Их высота больше ширины. Разве логично для них задавать тот же размер — 150×100?

Хорошее замечание. Поскольку высота изображений больше их ширины, у нас есть два варианта: мы можем поменять размеры местами (и сделать рисунки размером 100×150) или обрезать каждый рисунок и сделать для него малую версию размером 150×100 пикселов. Мы выбираем первый вариант. Если хотите выяснить, как это делается в вашей программе для редактирования фотографий, то можете обрезать рисунки и создать эскизы размером 150×100 пикселов.



Доработка HTML-кода таким образом, чтобы использовались эскизы

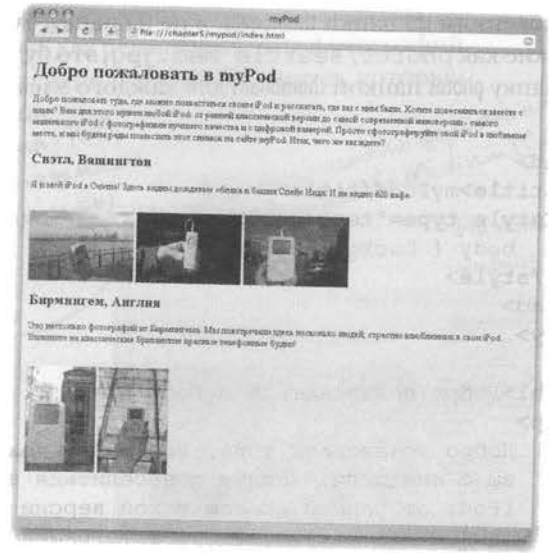
Теперь вам остается лишь поменять HTML-код таким образом, чтобы элементы **<img>** обращались к изображениям из папки thumbnails, а не из папки photos. Поскольку в коде используются относительные пути, такие как `photos/seattle_med.jpg`, это будет нетрудно. Все, что вам нужно сделать, — заменить в пути папку photos папкой thumbnails для каждого элемента **<img>**.

```
<html>
  <head>
    <title>myPod</title>
    <style type="text/css">
      body { background-color: #eaf3da;}
    </style>
  </head>
  <body>

    <h1>Добро пожаловать в myPod</h1>
    <p>
      Добро пожаловать туда, где можно похвастаться своим iPod и рассказать, где
      вы с ним были. Хотите повеселиться вместе с нами? Вам для этого нужен любой
      iPod: от ранней классической версии до самой современной нановерсии –
      самого маленького iPod с фотографиями лучшего качества и с цифровой камерой.
      Просто сфотографируйте свой iPod в любимом месте, и мы будем рады поместить
      этот снимок на сайте myPod. Итак, чего же вы ждете?
    </p>

    <h2>Сиэтл, Вашингтон</h2>
    <p>
      Я и мой iPod в Сиэтле! Здесь видны дождевые облака и башня
      Спейс Нидл. И не видно 628 кафе.
    </p>

    <p>
      <img src="thumbnails/seattle_med.jpg" alt="Мой iPod в Сиэтле, Вашингтон">
      <img src="thumbnails/seattle_shuffle.jpg" alt="iPod в Сиэтле, Вашингтон">
      <img src="thumbnails/seattle_downtown.jpg" alt="iPod в бизнес-центре Сиэтла,
      Вашингтон">
    </p>

    <h2>Бирмингем, Англия</h2>
    <p>
      Это несколько фотографий из Бирмингема. Мы повстречали здесь несколько
      людей, страстно влюбленных в свои iPod. Взгляните на классические британские
      красные телефонные будки!
    </p>

    <p>
      <img src="thumbnails/britain.jpg" alt="iPod в Бирмингеме в телефонной будке">
      <img src="thumbnails/applestore.jpg" alt="iPod в Бирмингеме в Apple store">
    </p>
  </body>
</html>
```

Все, что вам нужно сделать, — поменять название папки с photos на thumbnails.

И снова тест для myPod

Во-о-о-от... намного лучше. Пользователи теперь могут видеть все изображения, имеющиеся в наличии, просто взглянув на страницу. Кроме того, сейчас сразу видно, какая фотография в каком городе была сделана. Теперь нужно найти способ, чтобы создать ссылку с эскизов фотографий на их оригиналы.





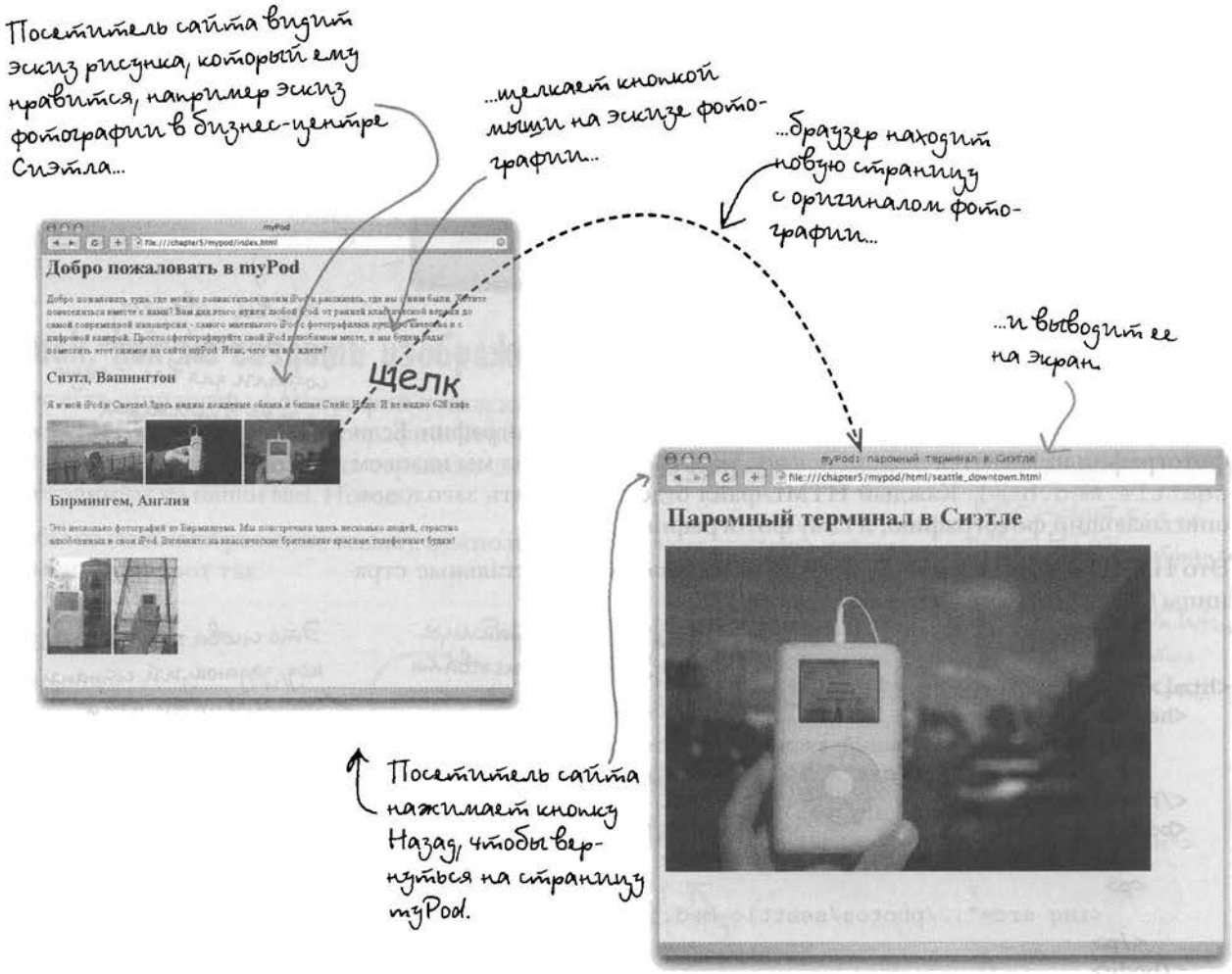
Верно, но элемент <img> — это строчный элемент.

Другими словами, мы никого не пытаемся «одурачить». Поскольку элемент **<img>** строчный, то предполагается, что перед ним и после него нет разрыва строки. Итак, если в вашем HTML-коде идет несколько элементов **<img>** подряд, то браузер будет отображать их рисунки в своем окне один за другим в одной строке, если это позволяет ширина окна браузера.

Причина того, почему большие фотографии не отображались одна за другой, состоит в том, что в окне браузера для этого не хватало места. Вместо этого они отображались одна под другой. Браузер всегда оставляет свободное место перед блочным элементом и после него, а если вы снова посмотрите в окно браузера, то увидите, что фотографии расположены непосредственно одна под другой, без отступов между ними. Это еще один признак того, что **<img>** — строчный элемент.

Превращение эскизов в ссылки

Вы уже почти все сделали. Теперь осталось только создать ссылки с каждого эскиза на его оригинал. Вот как это будет выглядеть.



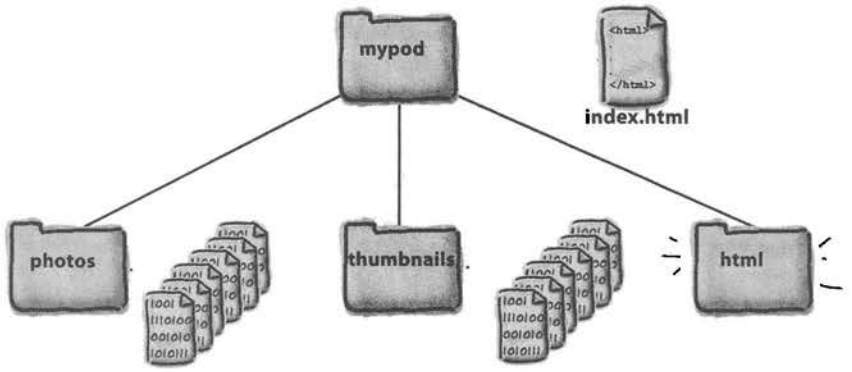
Вам нужно сделать две вещи.

1. **Страницы с изображением каждой фотографии и с названием, которое описывает ее содержание.**
2. **Ссылки с каждого эскиза на странице index.html на соответствующие оригиналы фотографий.**

Сначала создадим страницы, а затем вернемся и сделаем изображения-ссылки.

Создание индивидуальных страниц для фотографий

Сначала создайте новую папку html в папке mypod для хранения этих индивидуальных страниц.



Как вы, наверное, догадались, мы уже создали для вас такую папку в примерах к книге.

Сейчас мы создадим по одному HTML-файлу для каждой фотографии. Если фотография называется `seattle_med.jpg`, то HTML-файл мы назовем `seattle_med.html`. Каждый HTML-файл будет содержать заголовок, описывающий фотографию, и саму фотографию.

Это HTML-код для первой фотографии из Сиэтла. Все остальные страницы будут иметь аналогичную структуру:

Название страницы. Оно будет описывать фотографию.

Это снова готовый CSS-код, задающий странице определенный цвет.

```
<html>
  <head>
    <title>myPod: паромный терминал в Сиэтле</title>
    <style type="text/css"> body { background-color: #eaf3da; } </style>
  </head>
  <body>
    <h1>Паромный терминал в Сиэтле</h1>
    <p>
      <img src="../photos/seattle_med.jpg" alt="iPod в паромном терминале Сиэтла">
    </p>
  </body>
</html>
```

Здесь мы даем странице описательный заголовок.

Это элемент <img>, который указывает на оригинал фотографии seattle-med.jpg. Добавим в элемент <img> описательный атрибут alt.

Обратите внимание, что нам нужно использовать символы «..» в относительном пути, потому что папка html — это «сестра» папки photos, значит, сначала нужно подняться на одну папку вверх, а затем опуститься в папку photos.



Упражнение

Если вы посмотрите в папку html в приложении с примерами файлов к этой главе, то найдете там все индивидуальные страницы для фотографий, кроме страницы для `seattle_downtown.jpg`. Создайте страницу с названием `seattle_downtown.html` в папке html и протестируйте ее. Перед тем как продолжать дальше, проверьте, чтобы она работала. Если у вас будут какие-то проблемы с выполнением этого упражнения, то можете посмотреть правильное решение в конце этой главы.

Итак, как же создать изображения-ссылки?

У вас есть оригиналы фотографий и их эскизы, а также набор индивидуальных HTML-страниц для фотографий. Теперь вам нужно все это соединить и создать ссылки с эскизов на странице `index.html` на соответствующие им страницы из папки html. Но как?

Чтобы создать изображение-ссылку, нужно поместить элемент **<img>** внутрь элемента **<a>**, вот так:

href ссылается на новую HTML-страницу с фотографией seattle-downtown.html, которая находится в директории html.

Это открывающий тег <a> непосредственно перед элементом <img>.

Это элемент <img> для эскиза seattle-downtown.jpg, как он определен в файле index.html.

```
<a href="html/seattle_downtown.html">
  <img src="thumbnails/seattle_downtown.jpg"
       alt="iPod в бизнес-центре Сиэтла, Вашингтон">
</a>
```

Это закрывающий тег .

Элемент <img> полностью вложен в элемент <a>.

Как только вы поместили элемент **<img>** в элемент **<a>**, браузер начинает воспринимать изображение как ссылку, на которой можно щелкнуть. Когда вы щелкаете кнопкой мыши на таком изображении, браузер извлекает страницу, указанную в значении атрибута `href`.

Добавление изображений-ссылок в файл index.html

Это последний шаг. Вам осталось вложить все элементы **<img>** для эскизов в файле `index.html` в элементы **<a>**. Помните, что атрибут `href` каждого элемента **<a>** должен ссылаться на страницу с оригиналом фотографии из папки **html**. Убедитесь, что все ваши ссылки, эскизы и страницы с оригиналами фотографий соответствуют друг другу.

Здесь мы приводим весь файл `index.html`. Все, что вам нужно сделать, — добавить HTML-код, выделенный серым цветом.

```
<html>
  <head>
    <title>myPod</title>
    <style type="text/css">
      body { background-color: #eaf3da;}
    </style>
  </head>
  <body>

    <h1>Добро пожаловать в myPod</h1>
    <p>
      Добро пожаловать туда, где можно похвастаться своим iPod и рассказать, где
      вы с ним были. Хотите повеселиться вместе с нами? Вам для этого нужен любой
      iPod: от ранней классической версии до самой современной нановерсии — самого
      маленького iPod с фотографиями лучшего качества и с цифровой камерой. Просто
      сфотографируйте свой iPod в любимом месте, и мы будем рады поместить этот снимок
      на сайте myPod. Итак, чего же вы ждете?
    </p>

    <h2>Сиэтл, Вашингтон</h2>
    <p>
      Я и мой iPod в Сиэтле! Здесь видны дождевые облака и башня Спейс Нидл.
      И не видно 628 кафе.
    </p>

    <p>
    <a href="html/seattle_med.html">
      <img src="thumbnails/seattle_med.jpg" alt="Мой iPod в Сиэтле, Вашингтон">
    </a>
    <a href="html/seattle_shuffle.html">
      <img src="thumbnails/seattle_shuffle.jpg" alt="iPod в Сиэтле, Вашингтон">
    </a>
    <a href="html/seattle_downtown.html">
      <img src="thumbnails/seattle_downtown.jpg" alt="iPod в бизнес-центре Сиэтла,
       Вашингтон">
    </a>
    </p>

    <h2>Бирмингем, Англия</h2>
    <p>
```

```
      Это несколько фотографий из Бирмингема. Мы повстречали здесь несколько
      людей, страстно влюбленных в свои iPod. Взгляните на классические
      британские красные телефонные будки!
    </p>

    <p>
    <a href="html/britain.html">
      <img src="thumbnails/britain.jpg" alt="iPod в Бирмингеме в телефонной будке">
    </a>
    <a href="html/applestore.html">
      <img src="thumbnails/applestore jpg" alt="iPod в Бирмингеме в Apple store">
    </a>
    </p>
  </body>
</html>
```

Каждый эскиз вложите в элемент <a>. Только будьте аккуратны и используйте правильное значение атрибута href для каждой ссылки!

Добавьте эти элементы <a> в свой файл index.html. Сохраните его, обновите в браузере и проверьте, как работает myPod!

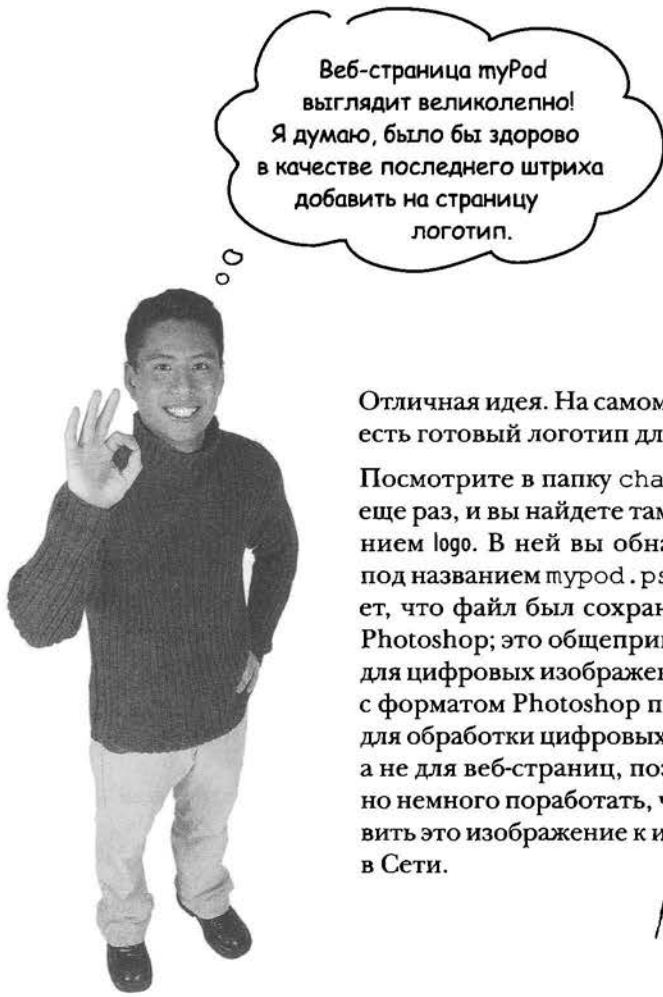
Часто Задаваемые Вопросы

В: Если внутри элемента <a> находится текст, то на странице он отображается подчеркнутым. Почему нет ничего подобного для изображений?

О: На самом деле большинство браузеров выделяют изображение границей вокруг него, чтобы показать, что оно является ссылкой (наш браузер Safari — один из тех, кто этого не делает). Если браузер рисует границу вокруг изображений-ссылок, а вам это не нравится, продолжайте читать книгу, и через несколько глав мы расскажем, как убрать эту границу с помощью CSS. Обратите также внимание, что, когда вы наводите указатель мыши на изображение-ссылку, вид указателя меняется так, чтобы подсказать вам, что на рисунке можно щелкнуть кнопкой мыши. В большинстве случаев пользователям понятно, что изображение является ссылкой, из контекста или по виду указателя мыши, даже если вокруг него нет границы.

В: Нельзя ли создать ссылки непосредственно на JPEG-изображения без всех этих HTML-страниц? Я думал, что браузеры достаточно умны для того, чтобы отображать рисунки просто так.

О: Вы правы, можно создать ссылку непосредственно на изображение: ... . Если вы так сделаете, а потом воспользуетесь такой ссылкой, то браузер сам выведет рисунок в пустом окне. Но вообще считается плохой манерой создавать ссылки непосредственно на изображения, потому что обычно рисунку дают какое-либо описание.



Отличная идея. На самом деле у нас уже есть готовый логотип для myPod.

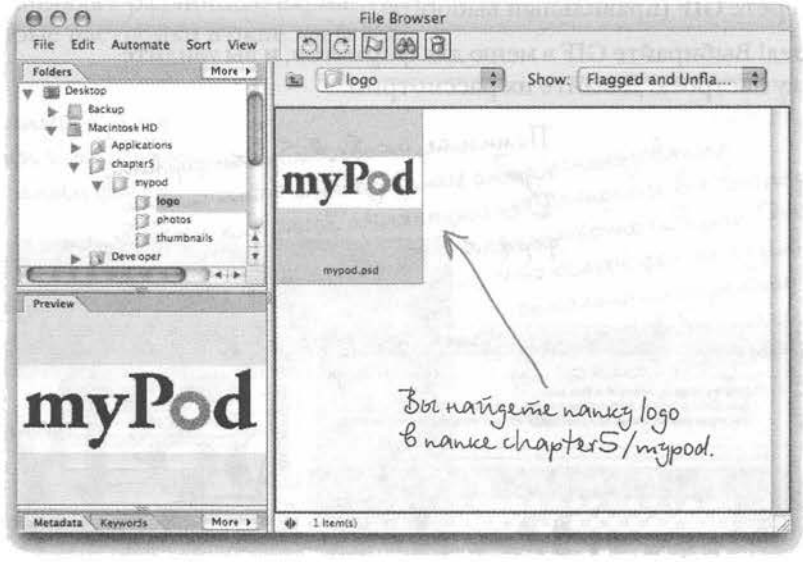
Посмотрите в папку `chapter5/mypod` еще раз, и вы найдете там папку с названием `logo`. В ней вы обнаружите файл под названием `mypod.psd`. PSD означает, что файл был сохранен в формате Photoshop; это общепринятый формат для цифровых изображений. Но файлы с форматом Photoshop предназначены для обработки цифровых изображений, а не для веб-страниц, поэтому нам нужно немного поработать, чтобы подготовить это изображение к использованию в Сети.

Многие программы для редактирования фотографий работают с файлами формата PSD, поэтому, даже если у вас нет Photoshop Elements, продолжайте читать дальше и делайте все аналогично в своем редакторе. Если же в вашей программе нельзя открыть PSD-файл, вы можете воспользоваться готовыми логотипами из папки logo.

Открытие логотипа myPod

Давайте поработаем с логотипом myPod: откройте файл `mypod.psd` из папки `chapter5/mypod/logo` в программе Photoshop Elements.

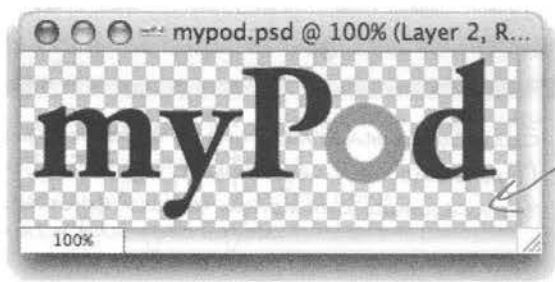
Если в вашей программе для редактирования фотографий файл не откроется, все равно работайте с другим форматом по аналогии.



Подробнее...

Логотип сделан удачно; в нем текст сочетается с двумя кругами: серым и белым (что явно ассоциируется с функцией «click wheel» iPod).

Но что это за узор в клеточку на заднем плане? Это способ, которым большинство программ для редактирования изображений отображают прозрачные области. Вам нужно это помнить, когда будете выбирать графический формат для логотипа...



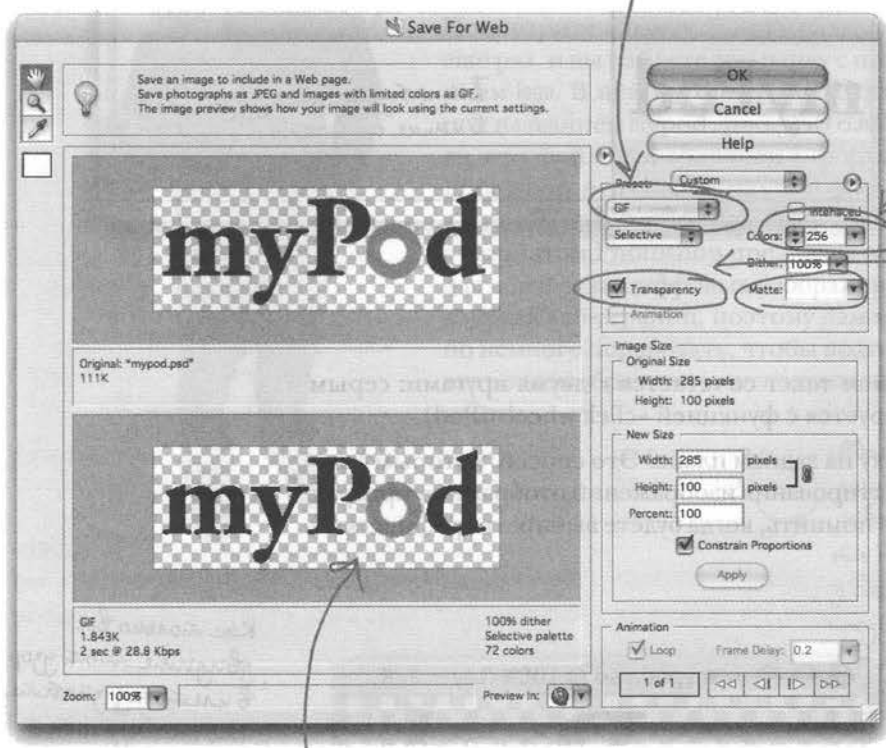
Как только вы увидите этот узор в клеточку, знайте, что он обозначает прозрачные участки на рисунке.

Какой формат использовать?

Вы уже знаете, что есть несколько вариантов сохранения изображения: можно выбрать формат JPEG или GIF. В этом логотипе используется всего три цвета, текст и несколько геометрических фигур. Руководствуясь знаниями, которые у вас уже есть, вы, скорее всего, выберете GIF (правильный выбор!).

Итак, вперед! Выбирайте GIF в меню для форматов, и вы увидите еще парочку настроек. Давайте их рассмотрим...

Помните, чтобы выбрать формат, нужно использовать этот список. Для сохранения логотипа мы выберем формат GIF.



Здесь Photoshop Elements показывает вам количество цветов, использованное при сохранении рисунка в формате GIF. Уже установлено максимальное значение – 256. Мы не будем его менять.

Когда вы выбираете формат GIF, появляется флажок Transparency (Прозрачность). По умолчанию он установлен. Нам нужен прозрачный задний план?

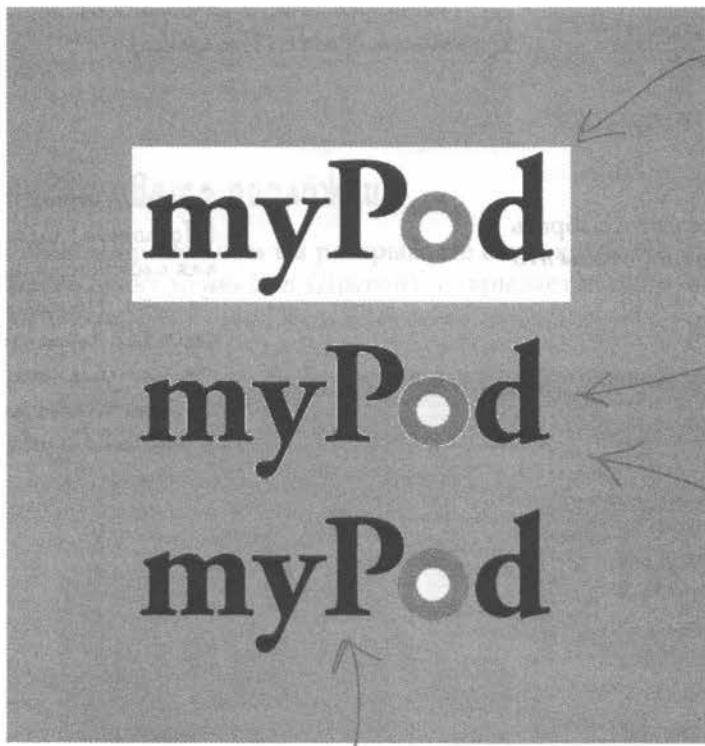
Обратите также внимание на список Matte (Подложка). Он тоже имеет отношение к прозрачности, как вы совсем скоро поймете.

Если вы попытаетесь снять флажок Transparency (Прозрачность), то увидите, что в GIF для предварительного просмотра задний фон стал белым.

Использовать прозрачность или нет?

Логотип myPod будет помещен на светло-зеленый фон. Итак, вам уже нравится идея использовать прозрачность? Хорошо, давайте сравним, как выглядят различные варианты логотипа, используя некоторые настройки в окне Save for Web (Сохранить для Сети).

Это логотипы, сохраненные тремя различными способами и отображенные на веб-странице с зеленым фоном.



Без использования прозрачности все выглядит достаточно сносно. Очевидно, что белый фон не будет нормально смотреться на зеленой веб-странице (он может выглядеть хорошо только на белой веб-странице).

А это то, что мы получаем, если устанавливаем флажок Transparency (Прозрачность) и сохраняем. Уже лучше, но что это за белый «ореол» вокруг букв в логотипе?

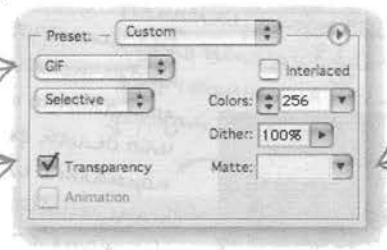
Эффект «ореола» возникает, потому что программа для редактирования фотографий создает подложку, смягчая края текста на выбранном цвете фона. Делая это для нашего логотипа, программа «предполагала», что смягчает края на белом фоне.

О, теперь все выглядит просто здорово. На этот раз мы сказали Photoshop Elements создать подложку вокруг текста с использованием зеленого фона. Как? Покажем вам чуть позже.

Сохранение прозрачного GIF-изображения

Вы знаете, что вам нужна прозрачная версия логотипа в формате GIF, вы также знаете, что нужно использовать подложку, чтобы избежать «ореолов» вокруг текста. Давайте посмотрим на область с настройками GIF в окне Save for Web (Сохранить для Сети).

Вы уже знаете, что нужно выбрать GIF.

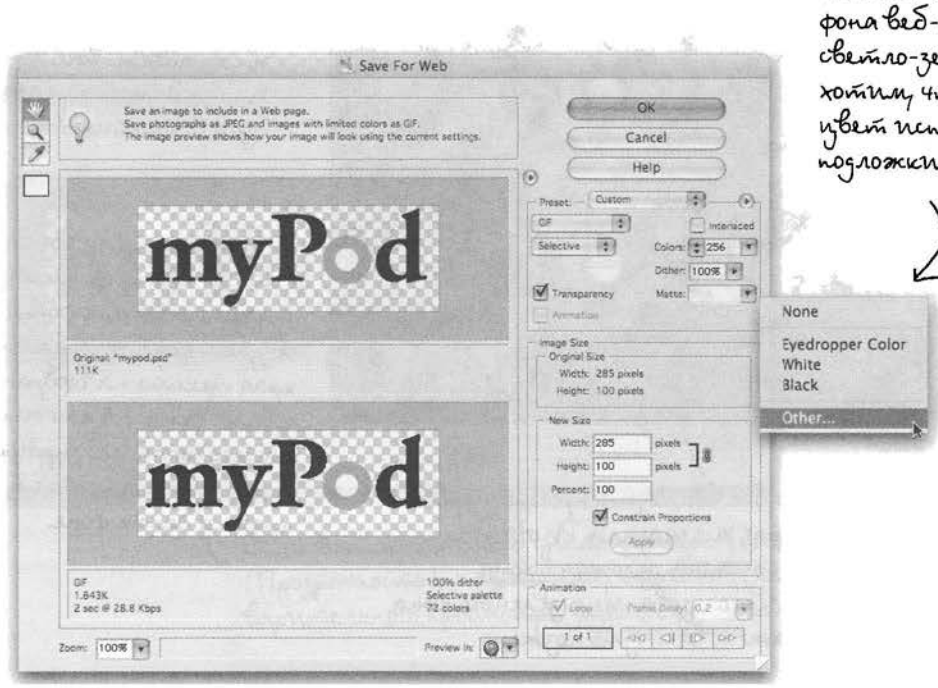


И установить флажок Transparency (Прозрачность).

Сейчас нам нужно разобраться со списком Matte (Подложка).

С помощью списка Matte (Подложка) вы можете выбрать цвет подложки вокруг текста. Желательно, чтобы это был цвет фона веб-страницы.

Список Matte (Подложка) задает цвет для смягчения краев текста. Поскольку цвет фона веб-страницы светло-зеленый, мы хотим, чтобы этот же цвет использовался для подложки.



Выберите пункт Other (Другой), так как нужного цвета в списке нет.

Минуточку, а как узнать цвет фона веб-страницы?

Помните тот готовый CSS-код в файле `index.html`? Он устанавливает светло-зеленый цвет для фона страницы. И вот откуда мы можем взять значение цвета:

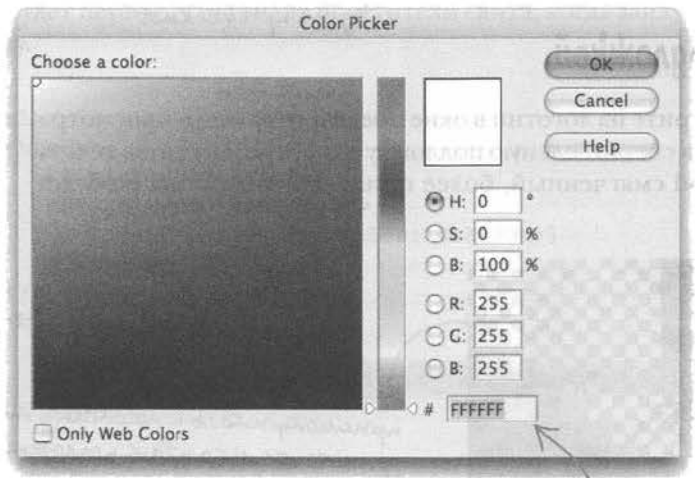
```
<style type="text/css">
   body { background-color: #eaf3da }
</style>
```

Это и есть цвет фона.

Что? Вы бы никогда не сказали, что это светло-зеленый? Пока поверьте нам на слово, а через несколько глав мы к этому вернемся и подробно расскажем вам о цветах.

Установка цвета подложки

Когда в Photoshop Elements вы раскрываете список Matte (Подложка) и выбираете пункт меню Other (Другой), открывается окно Color Picker (Палитра цветов).

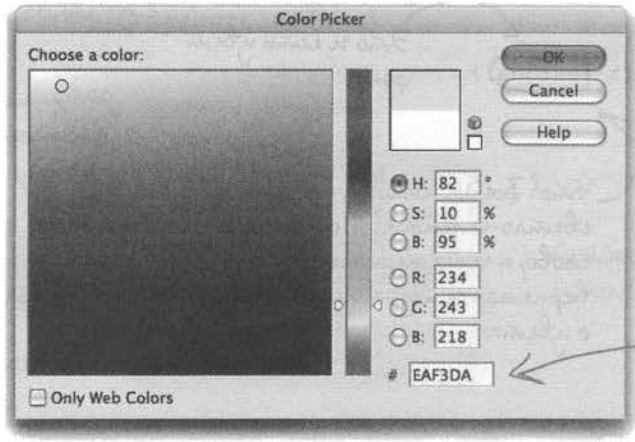


В нем есть множество способов выбрать цвет подложки. Мы просто хотим выбрать цвет фона страницы, уже зная его значение – eaf3da...

... и он должен быть установлен здесь.

Установка цвета подложки, продолжение

Итак, продолжим! Задайте значение цвета `eaf3da` в окне Color Picker (Палитра цветов). Вы увидите, что он стал таким же, как цвет фона страницы myPod.

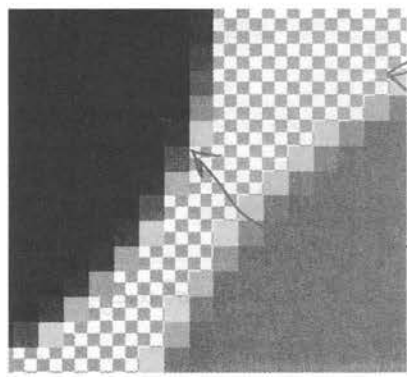


Введите эти буквы прямо сюда. Это поле предназначено специально для указания цвета в веб-формате. Можете вводить либо прописные, либо строчные буквы.

После того как вы ввели цвет в окне Color Picker (Палитра цветов), нажмите кнопку OK, и изменения в логотипе будут применены.

Рассмотрим логотип с подложкой

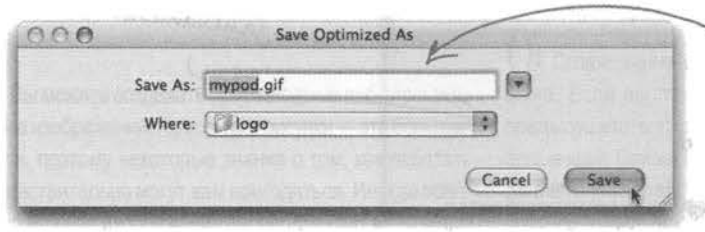
Теперь еще раз внимательно посмотрите на логотип в окне предварительного просмотра. Вы увидите, что программа добавила светло-зеленую подложку вокруг резких углов текста. Это придаст тексту логотипа myPod смягченный, более приятный вид, когда он будет помещен на веб-страницу.



Если сейчас вы присмотритесь к логотипу, то заметите, что цвет подложки совпадает с цветом фона веб-страницы mypod.html.

Сохранение логотипа

Отлично, вы задали все необходимые настройки в окне Save for Web (Сохранить для Сети), теперь нажмите кнопку OK, чтобы сохранить изображение с именем `mypod.gif`.



Программа автоматически поменяет расширение в имени файла на GIF. Сохраните изображение под названием mypod.gif в папку logo.

Добавление логотипа на веб-страницу myPod

Теперь все, что вам осталось сделать, — добавить логотип на веб-страницу myPod. Мы поместим его в самом верху страницы, чтобы он отображался над описанием сайта и фотографиями iPod. В таком случае это будет первое, что увидят посетители сайта, когда зайдут на страницу.

Добавьте изображение логотипа на самый верх веб-страницы myPod. Указывайте правильный относительный путь к логотипу, находящемуся в папке logo, и используйте атрибут alt, который описывает изображение.

```
<html>
   <head>
      <title>myPod</title>
      <style type="text/css">
         body { background-color: #eaf3da;}
      </style>
   </head>
   <body>
      <p>
         <img src="logo/mypod.gif" alt="Логотип myPod">
      </p>

      <h1>Добро пожаловать в myPod</h1>
      .
      .
      .
   </body>
</html>
```

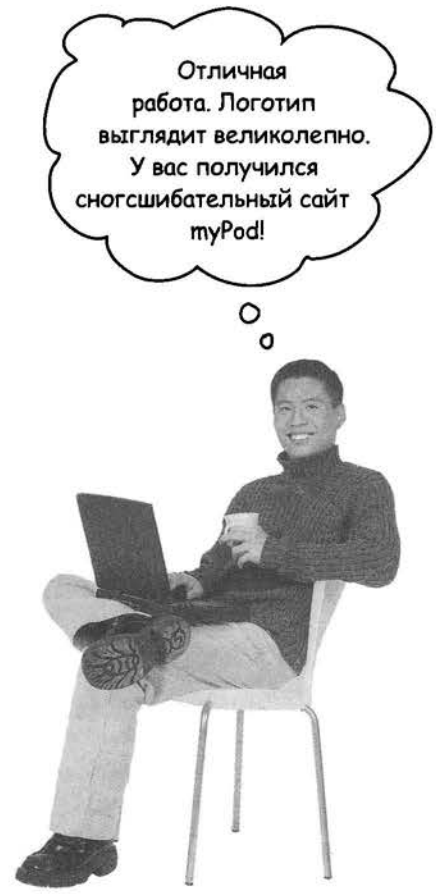
Здесь будет оставшаяся часть HTML кода из файла index.html.

А теперь последний тест

Давайте протестируем эту страницу! Обновите ее в браузере и посмотрите, как выглядит myPod-логотип в формате GIF с использованием прозрачности.



Ваши труды сполна окупились. Теперь на веб-странице myPod есть замечательный логотип.



Часто Задаваемые Вопросы

В: Обязательно ли мне все это знать о форматах изображений, чтобы создавать хорошие веб-страницы?

О: Нет. Вы можете создавать превосходные веб-страницы вообще без изображений. Но все-таки рисунки — это большая часть Сети, поэтому некоторые знания о том, как работать с ними, действительно могут вам пригодиться. Иногда всего лишь одно или два изображения переводят сайт из разряда хороших в отличные. Есть еще много всего, что можно узнать об изображениях, главное, только захотеть.

В: Зачем вообще сглаживать края текста?

О: Сравните два варианта логотипа для myPod:

myPod

myPod

Вы видите, что у логотипа, расположенного выше, очень резкие, зубчатые границы текста, что ухудшает его читаемость. Так текст отображается на экране компьютера по умолчанию. Во второй версии логотипа края улучшены благодаря использованию метода, называемого сглаживанием. На экране компьютера текст, обработанный таким методом, лучше читается и более приятен для глаз.

В: Когда вступает в действие подложка?

О: Сглаживание смягчает границы относительно цвета фона. Если вы поместите нижнюю версию логотипа (из предыдущего вопроса) на цветной фон, то увидите в нем белые края. Список Matte (Подложка) в программе Photoshop Elements позволяет установить цвет фона, на который будет помещен текст, поэтому края текста смягчаются с учетом этого цвета.

В: Этот метод работает только для текста?

О: Нет, он работает для любых линий в вашей графике, на которых могут появляться «зазубрины». К кругу в логотипе myPod тоже был применен этот метод.

В: Почему нельзя просто сделать цвет фона в логотипе чистым и совпадающим с цветом фона веб-страницы?

О: Вы можете сделать и так, но в этом есть один недостаток: если на вашей веб-странице имеются другие элементы, которые будут видны через прозрачные области логотипа, то они не будут отображаться в чистом цвете. Пока вы не видели подобных примеров, но еще увидите, когда будете изучать CSS.

В: Что, если я поменяю цвет фона после того, как создам версию логотипа с использованием подложки?

О: Скорее всего, незначительное изменение цвета фона не будет заметно. Однако, если вы значительно поменяете цвет фона страницы, вам придется исправить GIF, используя новый цвет подложки.

ПОВТОРИМ выученное



- Используйте элемент **<img>**, чтобы поместить изображение на веб-страницу.
- Браузеры обращаются с элементами **<img>** немного иначе, чем с другими элементами HTML. После прочтения HTML-страницы браузер извлекает каждое изображение с веб-сервера и отображает его.
- Если у вас много больших картинок на веб-странице, то вы можете сделать ее более удобной в использовании и уменьшить время ее загрузки. Для этого создайте эскизы изображений, на которых пользователь сможет щелкнуть, чтобы посмотреть оригиналы изображений.
- Элемент **<img>** — строчный элемент. Это означает, что браузер не добавляет разрывы строки перед изображениями и после них.
- С помощью атрибута `src` вы указываете месторасположение файла с рисунком. Вы можете брать изображения со своего сайта, используя относительные пути в атрибуте `src`, или изображения с других сайтов, используя URL-адрес.
- Атрибут `alt` элемента **<img>** позволяет задать содержательное описание изображения. Оно выводится в некоторых браузерах, если само изображение не может быть найдено, а также используется в экранных дикторах, чтобы можно было описать изображение людям со слабым зрением.
- Лучше всего на веб-страницах применять изображения шириной не больше 800 пикселов. Многие фотографии, созданные цифровыми камерами, имеют ширину, большую, чем у веб-страниц, поэтому необходимо менять их размер.
- Photoshop Elements — это одна из множества программ для редактирования фотографий, которую вы можете использовать, чтобы поменять размер изображения.
- Слишком большие изображения для веб-страниц создают неудобства в использовании этих страниц и увеличивают время их загрузки и отображения.
- Пиксел — самая маленькая точка, которая может быть изображена на экране. Каждый рисунок состоит из множества тысяч пикселов. В зависимости от монитора количество пикселов на дюйм колеблется от 72 до 120.
- JPEG и GIF — это два формата изображений, широко используемые в браузерах.
- Формат JPEG лучше всего подходит для фотографий и других сложных изображений.
- Формат GIF больше всего подходит для логотипов и другой простой графики с чистыми цветами, линиями или текстом.
- Изображения в формате JPEG могут быть сжаты до различного уровня качества, и вы можете выбрать наилучшее для себя соотношение размера и качества.
- Формат GIF позволяет создавать изображения с прозрачным фоном. Если вы поместите такое изображение на веб-страницу, то все, что окажется под ним, в том числе и фон самой веб-страницы, будет видно сквозь прозрачные части изображения.
- В Photoshop Elements для выбора нужного цвета при смягчении краев в вашем прозрачном GIF-изображении используйте список Matte (Подложка) из окна Save for Web (Сохранить для Сети).
- Изображения могут быть использованы в качестве ссылок на другие веб-страницы. Чтобы создать изображение-ссылку, используйте элемент **<img>**, вложив его в элемент **<a>**, и поместите ссылку в атрибут `href` элемента **<a>**.



Решение упражнений

КАКОЙ ФОРМАТ ИЗОБРАЖЕНИЯ?

Наши поздравления: вы были назначены лучшим по выбору форматов изображений. Для каждого из приведенных ниже рисунков выберите формат, который позволит лучше отобразить этот рисунок в Сети.

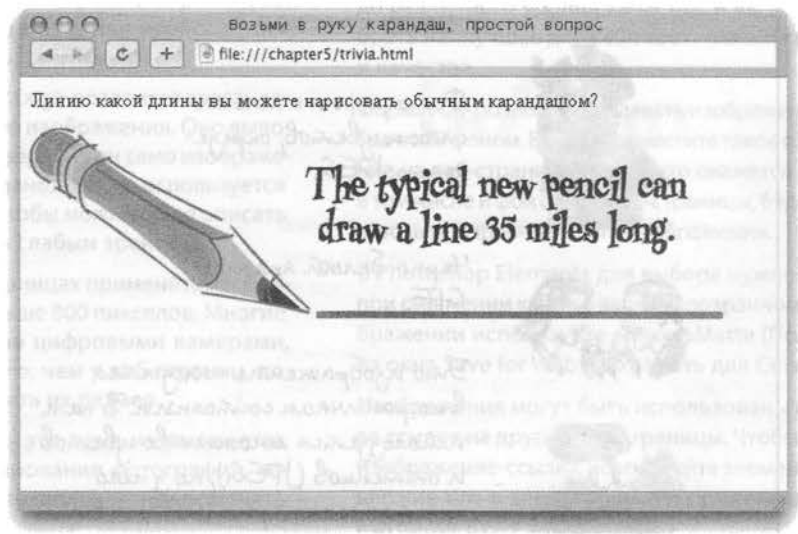
		JPEG	GIF
	Фотография с множеством оттенков серого.	☑	☐
	Всего несколько цветов и небольшой отрывок текста; определенно GIF.	☐	☑
	Фотография с множеством цветов; определенно JPEG.	☑	☐
	Черно-белый ярлык; GIF.	☐	☑
	Это изображение находится в пограничном состоянии. В нем используется множество цветов и оттенков (JPEG), но у него простая геометрия (GIF) и вы, скорее всего, захотите использовать прозрачность (GIF).	☑	☑

Возьми в руку карандаш
Решение

Это упражнение на самом деле имеет отношение к карандашам (и к рисункам). В нем поднимается один простой вопрос. *У вас есть обычный и совершенно новый карандаш. Если вы нарисуете сплошную линию, исписав все острие карандаша, то какой длины будет эта линия?*

Какое это имеет отношение к изображениям? Чтобы найти ответ, вам придется написать небольшой HTML-код. Ответ на этот простой вопрос содержится в изображении, которое вы найдете по URL-адресу `http://www.headfirstlabs.com/trivia/pencil.gif`. Ваша задача — добавить изображение в приведенный код и получить ответ:

```
<html>
   <head>
      <title>Возьми в руку карандаш, простой вопрос</title>
   </head>
   <body>
      <p>Линию какой длины вы можете нарисовать обычным карандашом?</p>
      <p>
         <img src="http://www.headfirstlabs.com/trivia/pencil.gif">
      </p>
   </body>
</html>
```

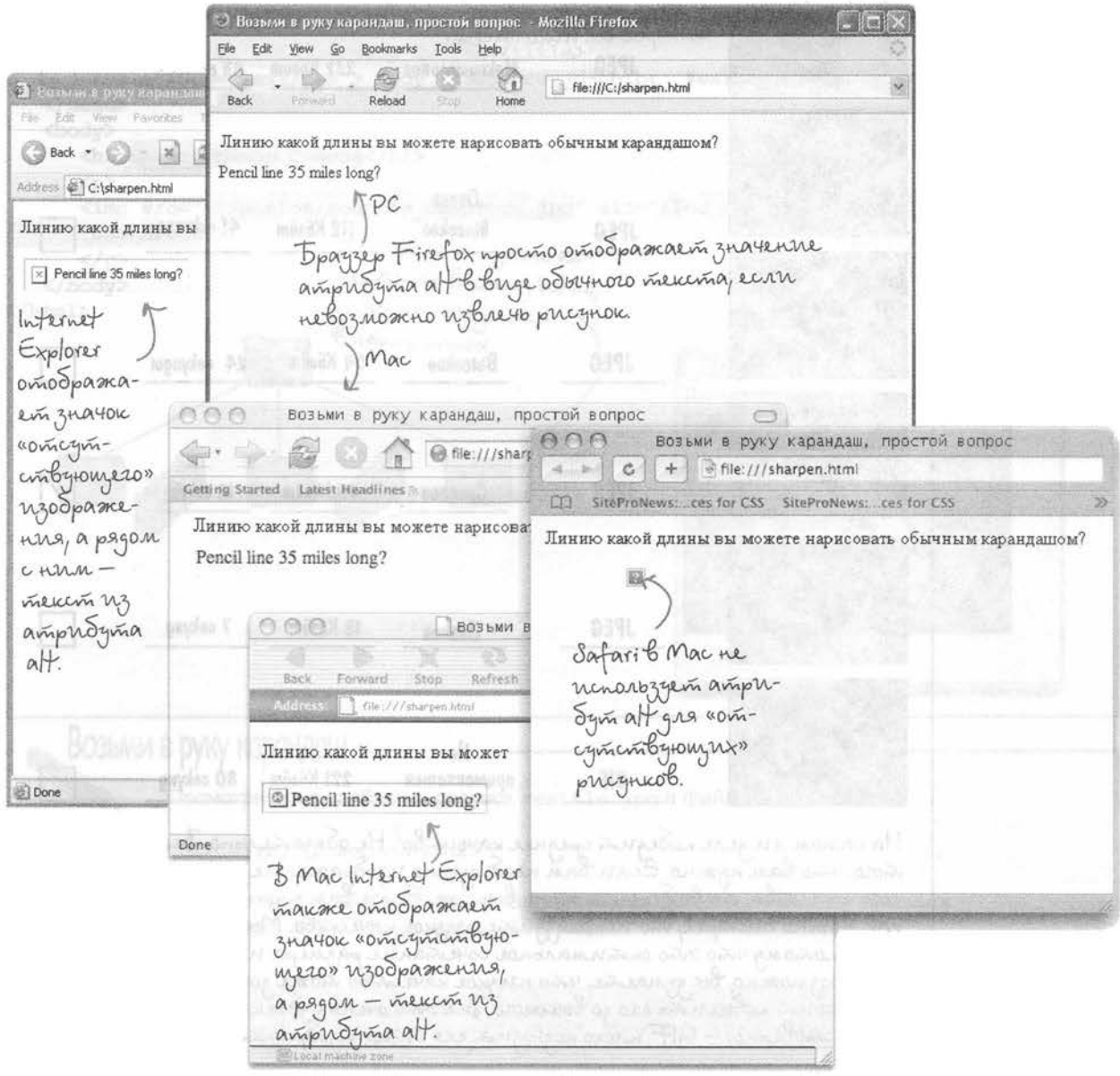


Источник: http://www.papermate.com.



Решение упражнений

Здесь приведены варианты того, как выглядят «отсутствующие» рисунки в различных браузерах. В большинстве случаев браузеры могут использовать дополнительную информацию из атрибута `alt`, чтобы описать то, что показано на рисунке. Почему надо об этом заботиться? В конце концов, это ошибка на веб-странице; и нам просто нужно ее исправить, верно? А в реальном мире все часто неидеально, иногда что-то ломается, прерывается подключение к Интернету, когда загружена только половина страницы, или людям со слабым зрением нужно услышать описание рисунка, потому что они не в состоянии его увидеть.





Решение упражнений

КАКОЙ ФОРМАТ ИЗОБРАЖЕНИЯ?

Вы заметили, как ухудшается качество изображения?

	Формат	Качество	Размер	Время	Победитель
	JPEG	Максимальное	232 Кбайт	83 секунды	☐
	JPEG	Очень высокое	112 Кбайт	41 секунда	☐
	JPEG	Высокое	64 Кбайт	24 секунды	☐
	JPEG	Среднее	30 Кбайт	12 секунд	☑
	JPEG	Низкое	18 Кбайт	7 секунд	☐
	GIF	Не применяется	221 Кбайт	80 секунд	☐

Обратите внимание, что ваши ответы могут немного различаться, в зависимости от версии программного обеспечения, которое вы используете.

На самом ли деле победит среднее качество? Не обязательно. Все зависит от того, что вам нужно. Если вам необходимо изображение действительно высокого качества, то выберите очень высокое. Если вам нужно, чтобы сайт работал как можно быстрее, то используйте низкое качество. Мы выбрали среднее качество, потому что это оптимальное сочетание размера и качества изображения. Возможно, вы думаете, что низкое качество тоже достаточно неплохое или что стоит повысить его до высокого. Все это очень субъективно. Единственное, что достоверно, – GIF плохо подходит для этих изображений (что неудивительно).



Решение упражнений

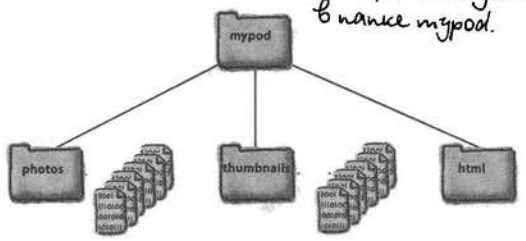
Если вы посмотрите в папку html в примерах к книге, то найдете там все индивидуальные страницы для изображений, кроме одной — `seattle_downtown.jpg`. Создайте страницу `seattle_downtown.html` в папке html и протестируйте ее. Перед тем как продолжить работу, проверьте, чтобы страница работала.

Вот решение:

Этот HTML-код должен размещаться в файле seattle_downtown.html.

```
<html>
  <head>
    <title>myPod: бизнес-центр Сиэтла</title>
    <style type="text/css"> body { background-color: #eaf3da; } </style>
  </head>
  <body>
    <h1>Бизнес-центр Сиэтла</h1>
    <p>
    <img src="../photos/seattle_downtown.jpg" alt="iPod в бизнес-центре Сиэтла,
     Вашингтон">
    </p>
  </body>
</html>
```

Этот файл нужно поместить в папку html, которая находится в папке mypod.



Это результат выполнения кода.



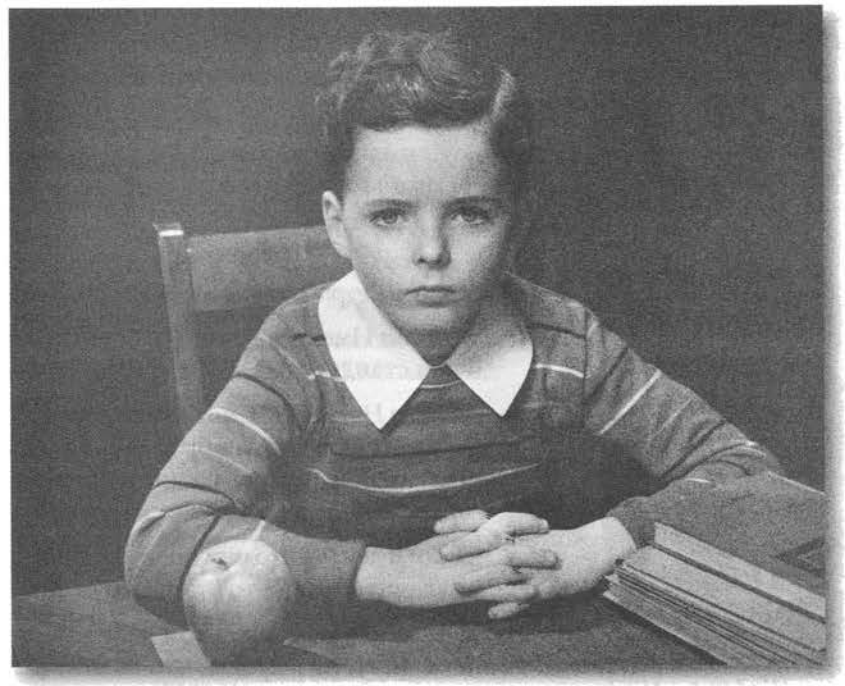
Возьми в руку карандаш



Посмотрите, как добавить рисунок `seattle.jpg` в файл `index.html`.

```
<h2>Сиэтл, Вашингтон</h2>
<p>
    Я и мой iPod в Сиэтле! Здесь видны дождевые облака и башня
    Спейс Нидл. И не видно 628 кафе.
</p>

<p>
    <img src="photos/seattle.jpg" alt="Мой iPod в Сиэтле, Вашингтон">
</p>
```

Серьезный HTML



Что же еще нужно знать об HTML? Вы уже довольно неплохо справляетесь с написанием HTML-страниц. Не настало ли время перейти к CSS и научиться придавать всей этой разметке еще и ошеломительный внешний вид? Перед тем как сделать это, мы должны убедиться, что ваши знания о HTML действительно на должном уровне. Мы планируем это сделать, серьезно подойдя к рассмотрению HTML. Не поймите нас неправильно, вы и так создавали первоклассный HTML, но есть еще несколько моментов, о которых вам нужно знать, чтобы помочь браузеру корректно отображать ваши страницы. Кроме того, вы должны быть уверены, что в вашем коде не будут появляться мелкие ошибки. Что это вам даст? Страницы, которые отображаются в различных браузерах более или менее одинаково (и даже на мобильных устройствах и с помощью экранных дикторов для людей со слабым зрением), страницы, которые быстрее загружаются, и страницы, которые гарантированно хорошо обращаются с CSS. Приготовьтесь, в этой главе вы из любителя превратитесь в профессионала.



Джим: Доработать?

Фрэнк: Имеется в виду – убедиться, что он соответствует всем стандартам HTML.

Джим: Наш HTML-код очень хорош. Посмотрите, как страница отображается в браузере. Она выглядит превосходно. Мы достаточно аккуратны и внимательны. Мы знаем, как правильно выстраивать элементы.

Джо: Я думаю, что они просто в очередной раз пытаются нас озадачить. Какие-то стандарты... Мы знаем, что делаем.

Фрэнк: Ребята, на самом деле мне неприятно признавать этот факт, но я думаю, что на этот раз босс прав.

Джим, Джо: Что?!

Джо: Да брось ты, нам просто дают очередной кусок работы. А нам и так есть что делать.

Фрэнк: Ребята, я пытаюсь сказать, что это поможет нам в будущем делать меньше работы.

Джо: Ха! Это было бы здорово...

Фрэнк: Хорошо, сейчас объясню: браузер читает HTML-код и затем прилагает все усилия, чтобы отобразить его, так? По сути, браузеры очень многое нам прощают... Вы время от времени делаете ошибки или неправильно используете HTML, например случайно помещаете блочный элемент внутрь строчного, и браузер пытается все это исправить сам.

Джим: Ну и?

Фрэнк: Различные браузеры (скажем, Internet Explorer, Firefox и Safari) по-разному обрабатывают погрешности в HTML. Другими словами, если в вашем коде появляются ошибки, совершенно неясно, как ваши страницы будут отображаться в различных браузерах. И только если в коде нет ошибок, большинство браузеров покажут страницы одинаково. Если мы начнем проектировать дизайн с помощью CSS, а наш HTML не будет отвечать определенным требованиям, то различия станут еще более существенными.

Итак, если мы убедимся, что наш HTML, как они говорят, «соответствует стандартам», в будущем у нас будет намного меньше проблем, связанных с некорректным отображением страниц у заказчиков.

Джим: Если это уменьшит количество беспокоящих меня телефонных звонков, раздающихся в 3 часа ночи, то для меня это очень хорошая идея. В конце концов, наши заказчики используют все существующие браузеры.

Джо: Минуточку, я все еще не понял. Разве сейчас мы не придерживаемся стандартов? Что не так с нашим HTML-кодом?

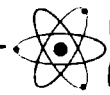
Фрэнк: Может быть, и ничего, но есть парочка вещей, которые нужно сделать, чтобы удостовериться, что все хорошо.

Джо: Например?

Фрэнк: Мы можем начать с того, что немного поможем браузеру, сказав ему, какую именно версию HTML используем.

Джо: Я даже не знаю точно, какую версию мы используем.

Фрэнк: Ха! Значит, точно есть над чем поработать. Ладно, начнем с выяснения того, какую версию HTML мы используем и как сообщить о ней браузеру. Есть еще несколько вещей, которые мы должны сделать, но не волнуйтесь, их не так уж и много. Если мы со всем этим справимся, то это очень облегчит нам жизнь при использовании CSS.



МОЗГОВОЙ ШТУРМ

Браузеры будут одинаково отображать веб-страницы, если HTML-код написан корректно. Однако если вы допускаете ошибки в коде, то часто страницы в различных браузерах выводятся по-разному. Как вы думаете, почему это происходит?

История развития HTML



HTML 1.0–2.0

Это было давно. Вы могли бы уместить все, что на тот момент нужно было знать об HTML, в багажник вашей машины. Внешний вид страниц оставлял желать лучшего, но по крайней мере они были написаны на HTML. Никто особенно не заботился о дизайне — беспокоились только о том, чтобы у всех в Сети была своя собственная «домашняя страница». В те дни даже карандаши, скрепки и стикеры рассматривались как «веб-содержимое» (вы, наверное, думаете, что мы шутим).



HTML 3

Далее начались холодные дни «войн между браузерами». Netscape и Microsoft боролись за господство во всем мире. В конце концов, тот, кто управляет браузером, управляет и всем миром, не так ли?

В эпицентре боевых действий оказались веб-разработчики. Во время этих войн возникла «гонка вооружений», так как каждая компания, поддерживающая свой браузер, продолжала добавлять в него собственные возможности, чтобы быть впереди всех. Сложно было оставаться в курсе всех новинок. Кроме того, в те дни часто приходилось писать сразу по две веб-страницы: одну для браузера Netscape, другую — для Internet Explorer. Мало хорошего.



HTML 4

Ах... Конец «войнам между браузерами» и, к счастью, создание Консорциума Всемирной паутины (или W3C). Его цель — навести в мире порядок, создав ЕДИНЫЙ HTML-«стандарт», чтобы держать все под контролем.

В чем суть их плана? Разделить HTML-структуру и дизайн и использовать для них два разных языка: один для структуры (HTML 4.0) и другой — для дизайна веб-страниц (CSS), а также убедить создателей браузеров, что в их интересах принять эти стандарты.

Но сработал ли их план?

Ох, почти... С некоторыми поправками (смотрите HTML 4.01).

1989 1991 1995 1998

Наша цель в этой главе — подняться на новый уровень и начать использовать HTML 4.01.



HTML 4.01



XHTML 1.0

Начиная с главы 7, нашей целью будет использование XHTML 1.0. Как и все в этом мире, HTML постоянно развивается, и в этой книге мы будем рассказывать об этом.

Ах, жизнь хороша. HTML 4.01 появился на свет в 1999 году, и до сих пор это самая популярная версия. Несмотря на то что все надеялись, что 4.0 будет последней версией, время от времени приходилось вносить в него кое-какие изменения. Они незначительны, и о них не стоит беспокоиться.

По сравнению с теми днями, когда HTML только появлялся (когда всем нам приходилось несладко), сейчас мы получаем удовольствие от написания на HTML 4.01 и спим спокойно по ночам, зная, что почти все браузеры (по крайней мере те, о которых стоит волноваться) хорошо отобразят веб-страницы.

Но, конечно же, в то время, как мы только начинали чувствовать себя комфортно, создавались новые технологии и все менялось. HTML и другой язык для разметки, известный как XML, объединились, и мы не успели оглянуться, как «родился» XHTML 1.0. Он унаследовал особенности обоих родителей: популярность и дружелюбие к браузерам от HTML и способность к расширению и точность от XML. Что это значит? Достаточно скоро вы это выясните, потому что мы хотим, чтобы вы создали пару XHTML-страниц, перед тем как произнесете вслух «XHTML». По крайней мере в следующей главе.

????

А что же случится в будущем? Может быть, мы все будем передвигаться на летающих машинах и есть на ужин питательные пилюли вместо обычной еды? Продолжайте читать, чтобы выяснить это.





УЯЗВИМОСТЬ БРАУЗЕРА

Интервью, взятое на этой неделе.
Почему для тебя так важна версия HTML?

Head First: Мы рады, что ты к нам пришел, Браузер. Как ты знаешь, версии HTML стали достаточно часто обсуждаемым вопросом. С чем это связано? В конце концов, ты браузер. Я даю тебе HTML-код, а ты обрабатываешь его и отображаешь веб-страницу наилучшим способом.

Браузер: В наши дни браузерам приходится нелегко. Существует множество веб-страниц, и многие из них написаны с использованием старых версий HTML или в их разметке есть ошибки. Как вы сказали, моя работа — попытаться отобразить каждую из этих веб-страниц вне зависимости от того, как она написана.

Head First: А в чем сложность? Какая разница, какую версию HTML я использую?

Браузер: Помните войны между браузерами? В HTML было добавлено множество элементов, которые, как предполагается сейчас, мы не должны больше использовать. Однако некоторые люди ждут, что мы — браузеры — должны быть в состоянии воспринимать все элементы, включая устаревшие, а мы не всегда сходимся во взглядах на то, как именно они должны отображаться.

Head First: А почему предполагается, что вы больше не должны поддерживать эти элементы?

Браузер: Перед тем как были изобретены каскадные таблицы стилей — CSS, в HTML были элементы, предназначенные для дизайна страниц, а не для их структуры. Сейчас эти элементы больше не нужны, но все еще осталось немалое количество веб-страниц, в которых они используются.

Head First: Кажется, я начинаю понимать, в чем проблема. Как же ты справляешься с отображением всех этих веб-страниц, написанных на разных версиях HTML? Это достаточно сложная задача.

Браузер: Как я и говорил, браузером быть нелегко. Все закончилось тем, что для нас было создано два набора правил отображения веб-страниц: один — для старого HTML, другой — для более современного, соответствующего стандартам. Когда я использую старые правила, я называю это режимом обратной совместимости, в котором имитируются особенности и ошибки старых версий HTML, потому что при работе с такими страницами может быть много неприятных сюрпризов.

Head First: Мне кажется, что это достаточно хорошее решение.

Браузер: Да, однако при таком подходе у вас могут возникнуть кое-какие проблемы. Если вы пишете на новом HTML, но не говорите мне об этом, я должен *воспринять* это так, будто вы пишете на старом HTML, и применять режим обратной совместимости. А вы этого не хотите.

Head First: Что ты имеешь в виду?

Браузер: Не все браузеры пришли к единому мнению о том, как отображать старый HTML, но мы все примерно одинаково обрабатываем код, основанный на стандартах. Поэтому, если вы используете HTML, соответствующий стандартам, укажите это, и вы получите более или менее одинаковые результаты во всех браузерах.

Head First: Ого, значит, ты можешь начать использовать правила режима обратной совместимости для страниц, написанных на новом HTML?

Браузер: Именно. Если я не знаю о том, что вы пишете на новом HTML, я включаю режим обратной совместимости и делаю все, на что способен. Но вы этого не хотите, потому что из-за этого режима внешний вид страницы очень испортится, в то время как она выглядела бы превосходно, если бы вы заранее сказали, что используете новый HTML.

Head First: Как же со всем этим справиться? Мы хотим, чтобы наши страницы выглядели хорошо.

Браузер: Легко. Заранее предупреждайте меня, какую версию HTML вы используете. Тогда я буду знать, какие правила нужно применять, чтобы отобразить вашу страницу.

Head First: Теперь понятно. Спасибо, Браузер!

Нужно позаботиться о том, чтобы браузеры не использовали режим обратной совместимости для наших страниц!

Для всех было бы лучше, если бы мы заранее говорили браузерам: «Это HTML-страница. Все стандарты соблюдены. Это HTML 4.01, дорогой!»

Если вы так сделаете, то браузер наверняка будет знать, как обрабатывать вашу страницу и (по крайней мере в тех браузерах, которые для вас имеют значение) страница отобразится так, как вы и ожидаете.

Итак, как же сказать это браузеру? Легко, вам просто нужно добавить одну строку в самом верху вашего HTML-файла. Вот как выглядит эта строка:

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

Да, да, мы знаем, что эта строка выглядит пугающе, но не забывайте, что она пишется не для вас, а для браузера. Она называется *определением типа документа*, потому что описывает браузеру *тип документа*. В данном случае документ — это ваша HTML-страница. Давайте просто взглянем на эту строку, чтобы немного понять, какую информацию она несет браузеру. Но опять же, это язык браузера, и вам необязательно все это знать и запоминать. Просто поместите строку в самом верху вашего HTML-документа и можете продолжать работать.

Говорит браузеру, что здесь определяется тип документа для данной страницы

Это означает, что элемент <html> — корневой (первый) на вашей странице

Это просто означает, что стандарт HTML 4.01 является общедоступным

В этой части говорится, что мы используем версию HTML 4.01 и HTML-разметка написана на английском языке

Вы можете напечатать все это в одну строку или, если хотите, разделить на несколько. Главное, убедитесь, что вы не разрываете на несколько строк то, что взято в кавычки.

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

Обратите внимание — это НЕ элемент HTML. После символа «<» сразу идет символ «!», что говорит вам, что это нечто иное

Это значение указывает на файл, который устанавливает особый стандарт.

Чуть позже мы поговорим об этом более подробно.

Напечатать все это вместе со всеми слэшами, кавычками и т. д. без ошибок достаточно сложно. Так что вместо того, чтобы печатать, можете просто скопировать этот текст из файла doctype.txt и вставить туда, куда нужно. Вы найдете этот файл в папке chapter6, если скачаете файлы с примерами к книге с сайта headfirstlabs.com.

Часто Задаваемые Вопросы

В: Что именно вы имеете в виду, когда говорите, что придерживаетесь стандартов или что пишете на стандарте HTML?

О: Стандарт HTML просто обозначает версию HTML, которая по всеобщему согласию была принята за стандарт, и в данный момент это HTML 4.01.
Придерживаться стандартов — это то же самое, что писать страницы на стандарте HTML.

В: Почему я вообще должен заботиться об этих стандартах? Мои страницы и так хорошо выглядят.

О: Вы ведь не хотите при написании веб-страниц и их оформлении с помощью CSS столкнуться с множеством проблем, связанных с тем, что страницы будут несовместимы (что означает «плохо отображены») в некоторых браузерах? Если же вы будете придерживаться стандартов, то ваши страницы гарантированно будут отображаться настолько одинаково в множестве браузеров, насколько это возможно.

В: Как же мне убедиться, что моя страница соответствует стандартам?

О: Для этого вам нужно будет выполнить некоторые действия, которые мы подробно разберем шаг за шагом. Кроме того, мы будем использовать бесплатную сервисную программу, которая проверит ваши страницы на соответствие стандартам.

В: Итак, мы называем HTML 4.01 стандартом?

О: Да, HTML 4.01 — это стандарт HTML, который наиболее широко поддерживается браузерами. Однако Сеть все время развивается, поэтому в следующих главах мы поговорим о том, что нового появляется в мире стандартов.

В: А что будет, когда появится HTML 5?

О: Хороший вопрос. Очень может быть, что HTML 5 вообще не появится, потому что новый стандарт написания веб-страниц — это XHTML. Вы узнаете все об XHTML в следующей главе. Хорошая новость для вас — вы уже достаточно хорошо подготовлены, чтобы писать как на HTML 4.01, так и на XHTML. Неважно, какой стандарт вы выберете, — вам будет легко писать веб-страницы, применяя те знания, которые у вас есть на данный момент.

В: Позвольте мне кое-что для себя прояснить: если я задаю определение типа документа в самом верху моего HTML-файла, то браузер, увидев это, может сделать определенные выводы про мой код, что очень хорошо, не так ли?

О: Да, так. Определив тип документа, вы говорите браузеру: «Я использую HTML 4.01». Когда браузер это видит, он предполагает, что вы знаете, о чем говорите, и действительно пишете на HTML 4.01. Это хорошо, потому что браузер будет работать по правилам верстки и отображения для HTML 4.01 и не будет использовать режим обратной совместимости.

В: Что будет, если я скажу браузеру, что использую HTML 4.01, а на самом деле это будет не так?

О: Браузер поймет, что вы пишете не на HTML 4.01, и вернется к режиму обратной совместимости. И у вас снова появятся проблемы, связанные с тем, что различные браузеры будут по-разному показывать ваши страницы. Единственный способ получить предсказуемый результат — сказать браузеру, что вы используете HTML 4.01 и на самом деле его использовать.

В: Я действительно не должен заботиться о том, что написано в этой строке с определением типа документа? Просто вставить ее в страницу?

О: Да, этого будет вполне достаточно. Однако есть то, на что все же стоит обратить внимание: существует несколько типов документов, о которых вы, возможно, захотите знать, и об одном из них мы поговорим совсем скоро. Но просто для определения типа документа вам достаточно вставить эту строку в самом верху вашего файла. Если вы поместите туда DOCTYPE, вряд ли кто-то будет пытаться разобраться, что именно в ней написано.

В: Меня немного смущает слово transitional (что означает «переходный») в этом типе документа. Я думал, что здесь говорится о стандарте, но слово «переходный» как-то не сочетается с понятием стандарта.

О: Отличное замечание, а у вас хорошая интуиция. Продолжайте читать, и через несколько страниц мы проясним этот вопрос.



Добавление определения типа документа

Хватит разговоров, давайте вставим этот DOCTYPE в HTML. Можете попробовать набрать его сами (надеемся, что глаза вас не подведут) или просто скопировать из файла `doctype.txt` в папке chapter6.

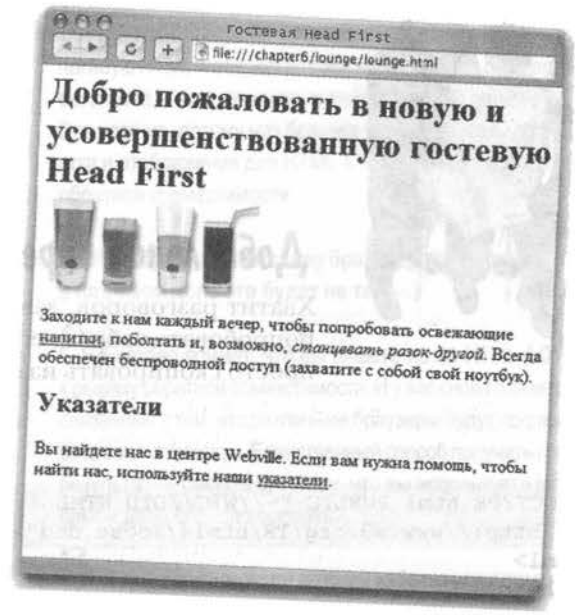
Это строка DOCTYPE. Просто добавьте ее в самое начало файла lounge.html.

Помните, что вы можете печатать все это в одну строку или нажать клавишу Enter между частями, заключенными в кавычки, как мы и сделали здесь.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Гостевая Head First</title>
  </head>
  <body>
    <h1>Добро пожаловать в новую и усовершенствованную гостевую Head First</h1>
       <img src="drinks.gif">
    <p>
       Заходите к нам каждый вечер, чтобы попробовать освежающие
       <a href="beverages/elixir.html">напитки</a>
       поболтать и, возможно,
       <em>станцевать разок-другой</em>.
       Всегда обеспечен беспроводной доступ
       (захватите с собой свой ноутбук).
    </p>
    <h2>Указатели</h2>
    <p>
       Вы найдете нас в центре Webville.
       Если вам нужна помощь, чтобы найти нас, используйте наши
       <a href="directions.html">указатели</a>.
       Присоединяйтесь к нам!
    </p>
  </body>
</html>
```

Тест для DOCTYPE

Внесите изменения в файл `lounge.html` из папки `chapter6/lounge` и загрузите страницу в браузере.



Здорово, ничего не изменилось. Мы на самом деле не ожидали никаких изменений, потому что все, что делает DOCTYPE, – дает браузеру знать, что вы используете HTML 4.01.



Упражнение

Добавьте DOCTYPE в файлы `directions.html` и `elixir.html`. Потом протестируйте эти страницы. Как и в случае с `lounge.html`, ничего удивительного не произойдет (зато по ночам вы будете спать спокойнее).



Джим: Да, действительно просто. Но вот что я до сих пор не уяснил: мы вставляем DOCTYPE в самом верху нашего файла, чтобы сказать браузеру, что страница написана на HTML 4.01, но ведь это не гарантирует того, что она *действительно* на нем написана. Мы ведь можем допускать ошибки. В чем же тогда смысл?

Фрэнк: Ты прав, потому что наше сообщение браузеру, что мы используем HTML 4.01, имеет смысл только в том случае, если мы на самом деле пишем на безупречном HTML 4.01. Это как раз то, о чем я только что собирался рассказать. Мы можем использовать бесплатную сервисную программу, которая определит, соответствует ли наша страница стандартам.

Джим: Правда? А как она работает?

Фрэнк: Работа программы заключается в следующем: сначала определяется тип документа, а затем на предмет корректности и отсутствия ошибок проверяется HTML-код. При этом также проверяется правильность использования названий тегов, их вложенности. Кроме того, программа отслеживает, чтобы ваши строчные элементы находились внутри блочных. Такая программа называется валидатором.

Джим: Класс, и она бесплатная? А кто предоставляет этот сервис?

Фрэнк: Те ребята, которые и предложили эти стандарты. Они называют себя Консорциумом Всемирной паутины, или сокращенно W3C.

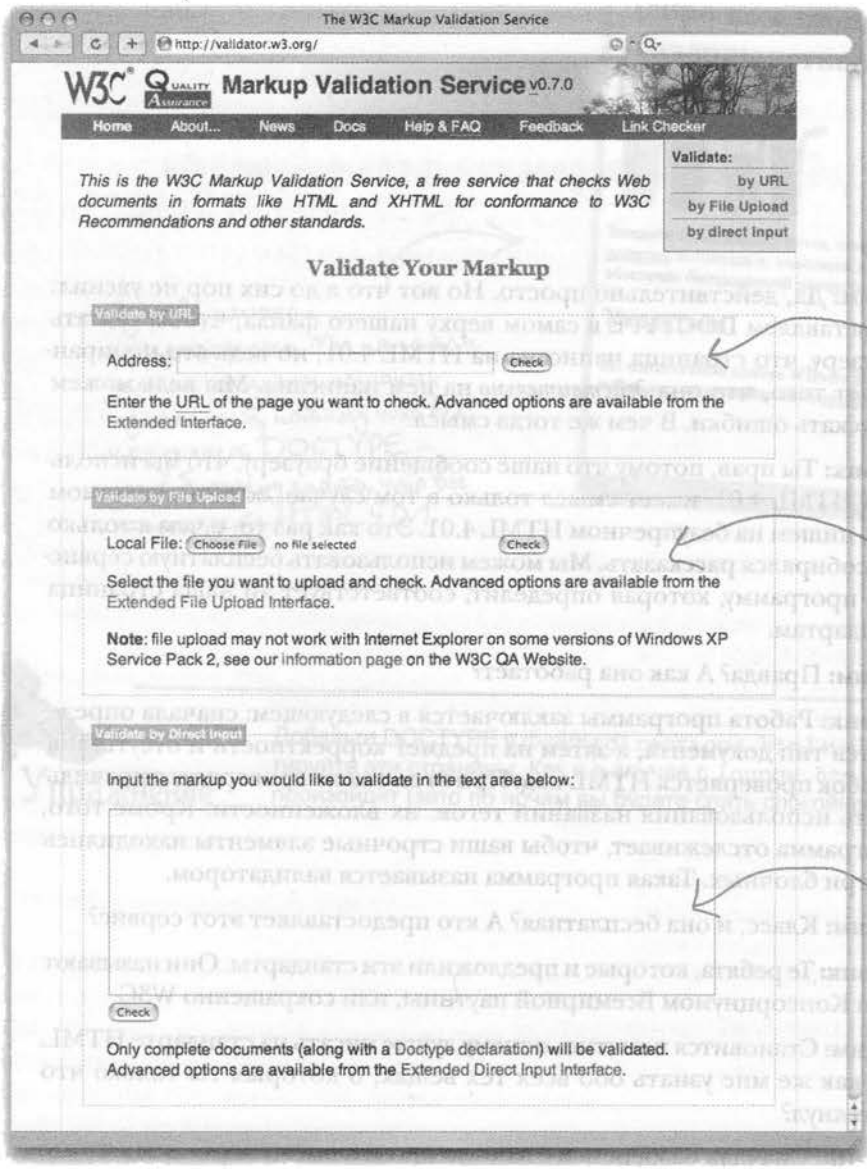
Джим: Становится понятно, почему лучше писать на стандарте HTML. Но как же мне узнать обо всех тех вещах, о которых ты только что упомянул?

Фрэнк: Сначала разберемся с валидатором, а потом вернемся к этому.

Познакомьтесь с W3C-Валидатором

Давайте дадим валидатору поработать и проверим с его помощью файлы гостевой. Чтобы начать, просто откройте ссылку `http://validator.w3.org` в своем браузере.

W3C-валидатор расположен по адресу http://validator.w3.org.



Существуют три способа для проверки вашего HTML.

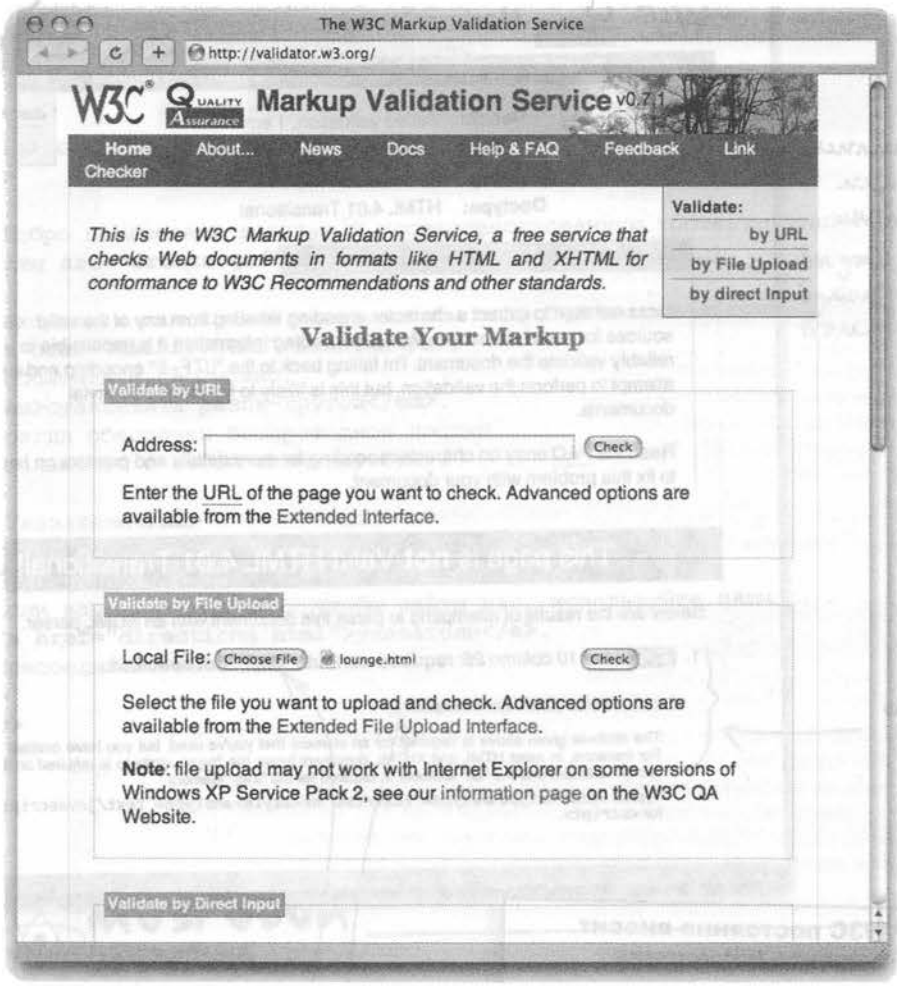
(1) Если страница находится в сети, то вы можете просто напечатать URL-адрес здесь, нажать кнопку Check и сервисная программа сама найдет ваш код и проверит его.

(2) Вы можете нажать кнопку Choose file (или Browse, если используете Windows) и выбрать файл на компьютере. После этого нажмите кнопку Check и браузер загрузит файл с вашего компьютера на удаленный, а сервисная программа проверит его.

(3) Скопируйте свой код и вставьте в эту форму. Затем нажмите кнопку Check, и сервисная программа проверит ваш HTML.

Валидация гостевой Head First

Мы будем использовать способ (3) для валидации файла `lounge.html`. Это значит, что нам нужно скопировать HTML-код из файла `lounge.html` и вставить его в форму, расположенную внизу веб-страницы W3C-валидатора. Попробуйте это сделать самостоятельно...



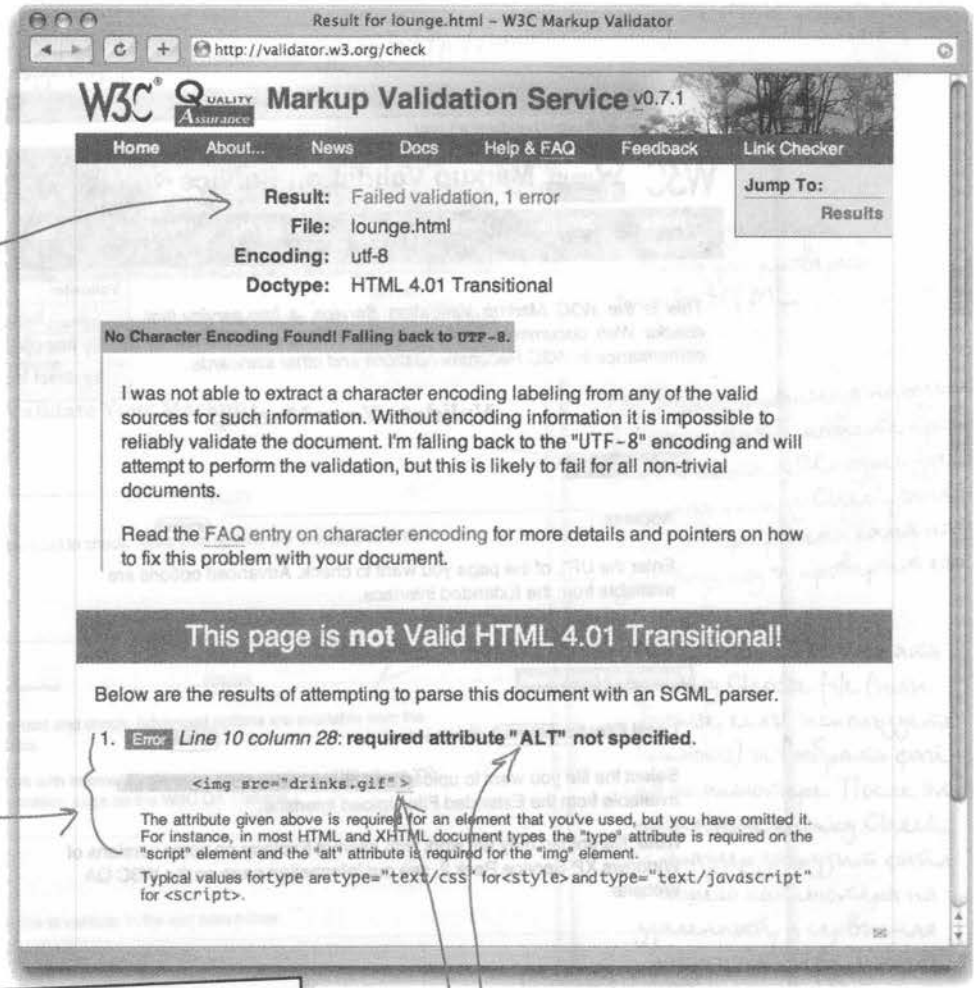
Здесь мы используем способ (2). Мы нажали кнопку Choose File и нашли файл lounge.html, в котором сейчас в самом верху указан DOCTYPE Transitional HTML 4.01. Мы подошли к самому главному... Пройдет ли веб-страница валидацию? Кто-нибудь может сказать наверняка? Нажмите кнопку Check (и переверните страницу), чтобы это выяснить.

Вы можете использовать способ (1) или (3), если вам так удобнее

дальше ▸

Хьюстон, у нас проблема...

Этот красный цвет вряд ли означает что-то хорошее. Выглядит так, будто валидация прошла с ошибкой. Давайте разберемся, в чем дело...



Мы провалились на валидации. Кажется, есть ошибка.

Похоже, это ошибка.

Все не так уж страшно. Кажется, в HTML 4.01 мы должны использовать атрибут alt в элементе <img>.



W3C постоянно вносит какие-то изменения в валидатор.

Поскольку W3C часто вносит какие-нибудь корректировки в валидатор, возможно, ваше сообщение об ошибке будет иным. Не волнуйтесь, просто продолжайте читать, потому что весь материал на нескольких следующих страницах очень важен.

Исправление этой ошибки

Кажется, это достаточно просто исправить. Вам просто нужно добавить атрибут `alt` в элемент **<img>** для версии HTML 4.01. Итак, вперед! Откройте файл `lounge.html`, внесите соответствующие изменения, сохраните, а затем снова попробуйте выполнить валидацию.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Гостевая Head First</title>
  </head>
  <body>
    <h1>Добро пожаловать в новую и усовершенствованную гостевую Head First</h1>
       <img src="drinks.gif" alt="Напитки">
    <p>
       Заходите к нам каждый вечер, чтобы попробовать освежающие
       <a href="elixir.html">напитки</a>,
       поболтать и, возможно,
       <em>станцевать разок-другой</em>.
       Всегда обеспечен беспроводной доступ
       (захватите с собой свой ноутбук).
    </p>
    <h2>Указатели</h2>
    <p>
       Вы найдете нас в центре Webville.
       Если вам нужна помощь, чтобы найти нас, используйте наши
       <a href="directions.html">указатели</a>.
       Присоединяйтесь к нам!
    </p>
  </body>
</html>
```

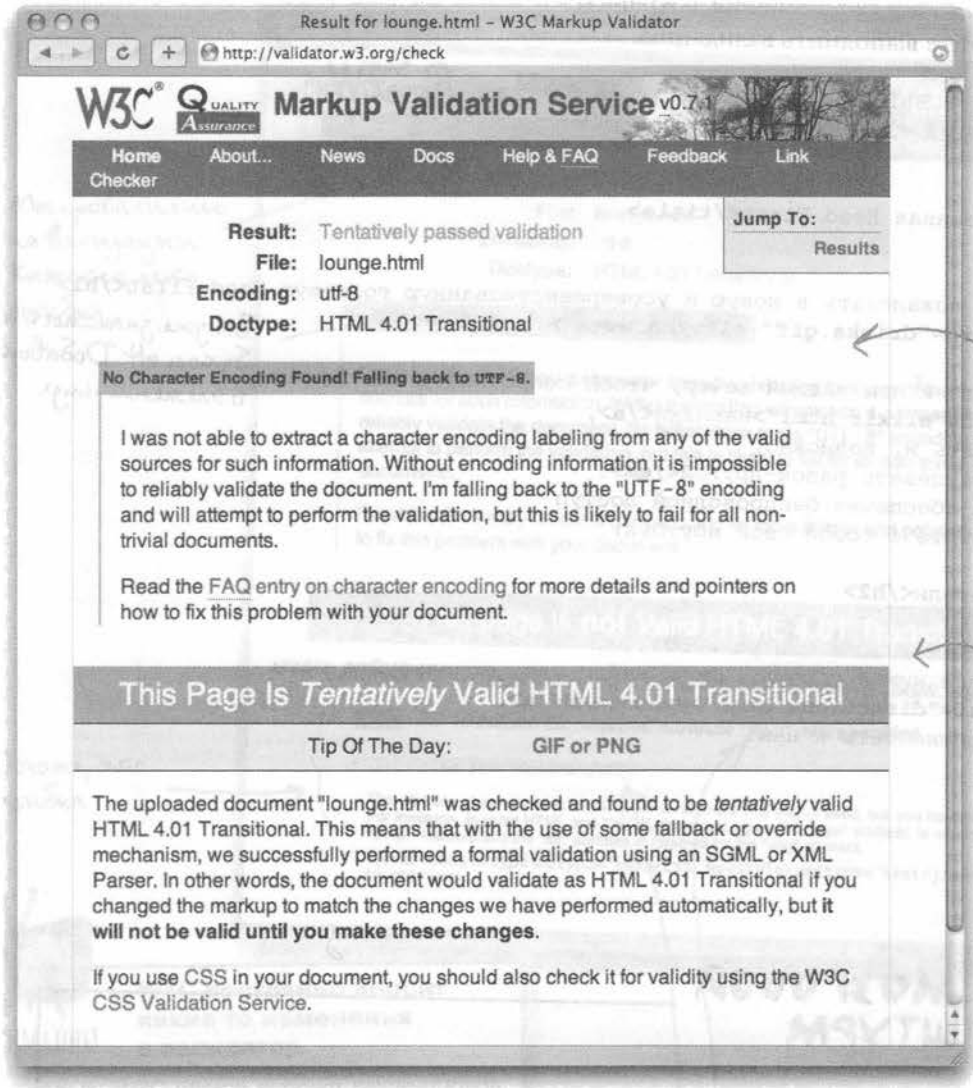
Вы уже знакомы с атрибутом alt. Добавьте его в элемент <img>.



Как вы думаете, почему в HTML 4.01 требуется атрибут `alt`?

Еще не Все...

Кажется, сейчас наш документ определен как tentatively valid HTML 4.01 Transitional, что значит «предварительно валидный». Это звучит как «близко, но не совсем». Давайте посмотрим.



В веб-странице гостевой совершенно точно есть какие-то проблемы, но что все это значит?

Похоже на то, что, если мы решим проблему, описанную выше, наш HTML станет валидным.

Итак, наш HTML-файл абсолютно соответствует требованиям в том смысле, как в нем написан код, но, кажется, мы должны сказать что-то о нашей кодировке символов. Чтобы решить эту проблему, мы сначала должны выяснить, что же это означает.



Фрэнк: Кодировка говорит браузеру, какой набор символов используется на странице. Например, страницы могут быть написаны с использованием кодировок для английского, китайского, арабского и множества других семейств символов.

Джим: А в чем сложность определения кодировки? Если в файле написано «a», то браузер и должен отображать это как «a». Верно?

Фрэнк: А что, если ты, например, используешь китайские символы на своей странице? Это абсолютно другой «алфавит» и в нем намного больше чем 26 символов, как в английском языке.

Джим: Эх, точно подмечено. Но разве браузер не в состоянии сам увидеть разницу? Все эти другие языки совсем не похожи на английский.

Фрэнк: Нет, браузер просто считывает данные. Он, конечно, может предположить, что читает английский алфавит, но что, если это не так? Знание кодировки символов избавляет браузер от необходимости делать какие-то предположения.

Джим: Наш сайт работал так долго, а теперь оказывается, что в нем была проблема. Почему?

Фрэнк: Потому что валидатор говорит: «Эй, вам лучше сказать мне, какие символы вы будете использовать, прежде чем я начну проверять вашу страницу на соответствие веб-стандартам и выявлять ошибки!» И обратите внимание, с этого момента мы всегда будем выполнять такую просьбу браузера. Не переживайте, нам просто нужно добавить еще одну строку в код – это будет тег **<meta>**. Я вскоре к этому вернусь.

Джим: Нет ли еще каких-нибудь сюрпризов для нас? Я действительно думал, что наша веб-страница успешно пройдет валидацию после того, как мы определили тип документа.

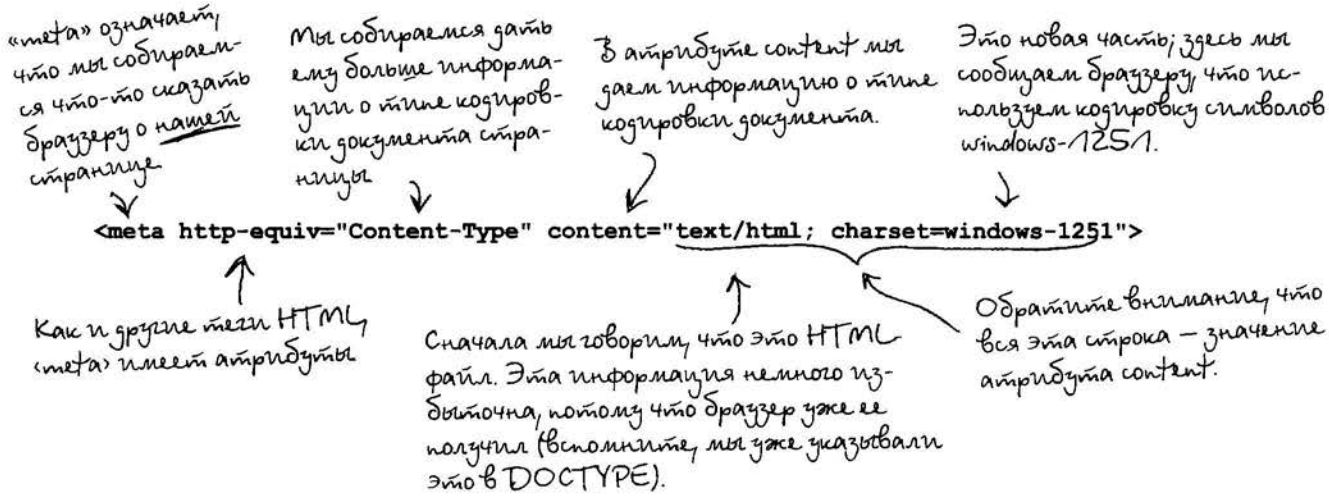
Фрэнк: Конечно, я тоже надеюсь, что больше сюрпризов не будет! Давайте добавим тег **<meta>** в наш файл и выясним это.

Добавление тега <meta> для определения типа кодировки документа

Многие из вас, скорее всего, используют русский язык, поэтому вам придется применять тег **<meta>**, который будет выглядеть так:

```
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
```

Вам нужно будет добавить эту строку в качестве первого элемента внутри **<head>** в своем HTML-коде. Этот тег говорит любому браузеру, какой тип кодировки используется в файле. Давайте рассмотрим тег **<meta>** более подробно.



Часто Задаваемые Вопросы

В: DOCTYPES, теги <meta>... Мне на самом деле нужно все это запомнить, чтобы писать веб-страницы?

О: Указание DOCTYPE и тега <meta> — это плата за то, чтобы ваш код рассматривался браузерами как соответствующий стандартам. Представьте это таким образом: вы уже понимаете 99 % людей, пишущих веб-страницы, что здорово. Но в один прекрасный день все просто поместили в свой HTML-код DOCTYPE и тег <meta> и вы перестали их понимать. Итак, вам тоже нужно указать в своих документах DOCTYPE и тег <meta> и убедиться, что они заданы правильно. Только после этого вы можете перейти к чему-нибудь более увлекательному.

В: Windows-1251?

О: Попытаемся объяснить. Это как WD-40 (товарный знак многофункционального препарата для антикоррозийной защиты и смазки трущихся поверхностей производства одноименной компании). Вам не нужно вникать в то, почему это так называется, просто используйте это.

Windows-1251 — это стандартная 8-битная кодировка для всех русских версий Microsoft Windows. Если вы пишете на каком-нибудь другом языке или пользуетесь другой операционной системой, ищите соответствующую информацию о кодировке символов на сайте http://www.w3.org/International/O-charset.ru.php. Мы же в этой книге будем указывать данную кодировку.

Осчастливим валидатор (и немало браузеров), добавив тег <meta>

Итак, вы знаете, что делать. Вам просто нужно набрать строку с типом кодировки документа (тег **<meta>**) в своем HTML. Давайте сначала добавим ее в файл `lounge html`:

Это тег <meta>. Мы добавили его внутрь элемента <head> перед элементом <title>.

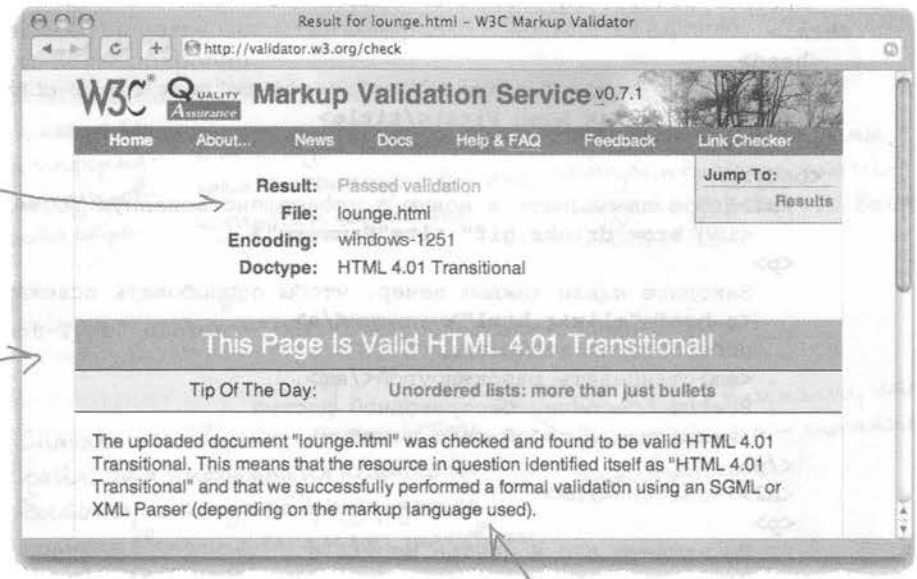
```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
    <title>Гостевая Head First</title>
  </head>
  <body>
    <h1>Добро пожаловать в новую и усовершенствованную гостевую Head First</h1>
       <img src="drinks.gif" alt="Напитки">
    <p>
       Заходите к нам каждый вечер, чтобы попробовать освежающие
       <a href="elixir.html">напитки</a>,
       поболтать и, возможно,
       <em>станцевать разок-другой</em>.
       Всегда обеспечен беспроводной доступ
       (захватите с собой свой ноутбук).
    </p>
    <h2>Указатели</h2>
    <p>
       Вы найдете нас в центре Webville.
       Если вам нужна помощь, чтобы найти нас, используйте наши
       <a href="directions.html">указатели</a>
       Присоединяйтесь к нам!
    </p>
  </body>
</html>
```

Всегда пишите эту строку первой внутри элемента <head>.

Хотите заключить пари? Пройдет ли этот файл процедуру валидации? Во-первых, внесите необходимые изменения в `lounge.html`, сохраните их и обновите страницу в браузере. *Вы* снова не заметите никаких изменений, зато *браузер* заметит. Теперь давайте проверим валидность страницы.

Бог любит троицу?

Как вы уже делали раньше, загрузите свой файл `lounge.html` на веб-страницу валидатора W3C по адресу `http://validator.w3.org`. Можете выполнить проверку, скопировав свой HTML-код в соответствующую форму на этом сайте, или даже переслать свои файлы на сайт и указать валидатору URL-адрес: как вам больше нравится. Когда вы с этим справитесь, нажмите кнопку Check.



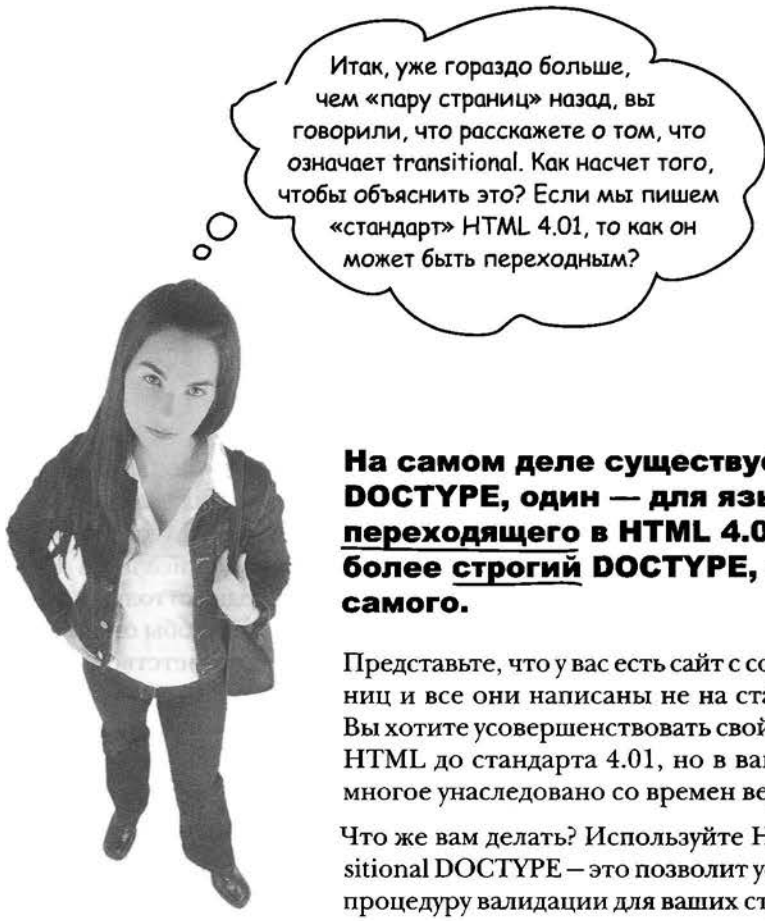
Валидация прошла успешно, вы победили!

Красный цвет обозначал что-то плохое, значит, зеленый должен говорить, что все хорошо.

Как мы и говорили, вы сказали браузеру (и валидатору), что используете HTML 4.01 Transitional, и сервис смог успешно выполнить валидацию.



Только посмотрите! Мы не сделали ничего особенного, но теперь можем сказать, что пишем на валидном HTML 4.01. Я думаю, что босс выпишет нам премию, когда увидит, как быстро мы справляемся с заданиями.

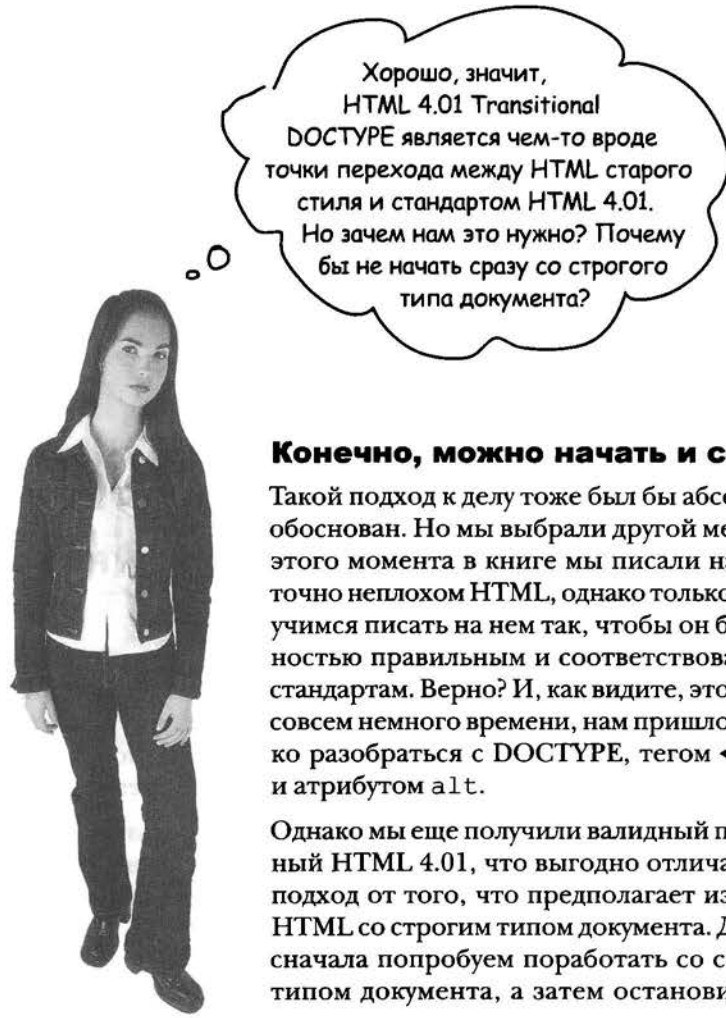


На самом деле существует два DOCTYPE, один — для языка, переходящего в HTML 4.01, и другой, более строгий DOCTYPE, — для него самого.

Представьте, что у вас есть сайт с сотнями веб-страниц и все они написаны не на стандарте HTML. Вы хотите усовершенствовать свой сайт и довести HTML до стандарта 4.01, но в ваших страницах многое унаследовано со времен версий 2.0 и 3.2.

Что же вам делать? Используйте HTML 4.01 Transitional DOCTYPE – это позволит успешно пройти процедуру валидации для ваших страниц, и все-таки разрешит оставить кое-что из «раннего» HTML. Таким образом, вы сможете удостовериться, что в вашей разметке нет явных ошибок (например, опечаток, несоответствующих тегов и т. д.), но вам не придется менять весь свой код, чтобы страница была признана валидной.

Затем, когда вы удалите весь «ранний» HTML, вы подготовите все для того, чтобы тип ваших документов считался «строгим», что гарантирует полное соответствие стандартам вашего сайта.



Конечно, можно начать и с него.

Такой подход к делу тоже был бы абсолютно обоснован. Но мы выбрали другой метод: до этого момента в книге мы писали на достаточно неплохом HTML, однако только сейчас учимся писать на нем так, чтобы он был полностью правильным и соответствовал всем стандартам. Верно? И, как видите, это заняло совсем немного времени, нам пришлось только разобраться с DOCTYPE, тегом **`<meta>`** и атрибутом `alt`.

Однако мы еще получили валидный переходный HTML 4.01, что выгодно отличает наш подход от того, что предполагает изучение HTML со строгим типом документа. Давайте сначала попробуем поработать со строгим типом документа, а затем остановимся на сравнении строгого и переходного типов.



Чтобы перейти от переходного HTML 4.01 к строгому, мы соответственно поменяем версию DOCTYPE. Когда мы это сделаем, валидатор (и браузеры) поймет это так, будто мы играем по более строгим правилам, что не позволяет наличие «раннего» HTML в документах. Мы поговорим об этих правилах через минуту, а сейчас протестируем строгий DOCTYPE. Чтобы это сделать, сначала еще раз взглянем на него:

Эта часть такая же, как и для переходного DOCTYPE.

Слово Transitional исчезло.

Это строгая версия HTML 4.01 DOCTYPE.

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

В целом все выглядит очень похоже.

Помните, этот URL-адрес определяет соответствие строгому HTML.

Разница невелика. Исчезло слово transitional, а мы использовали новый URL, определяющий строгую версию HTML 4.01. Давайте заменим переходный DOCTYPE строгим и попробуем пройти процедуру валидации.

Изменение переходного DOCTYPE на строгий

Откройте файл `lounge.html`. Чтобы поменять переходную версию на строгую, вам нужно внести лишь две правки в DOCTYPE: удалить **`Transitional`** и в URL поменять `loose.dtd` на `strict.dtd`. Или, если хотите, можете удалить старую строку и набрать новую.

Сначала удалите слово Transitional.

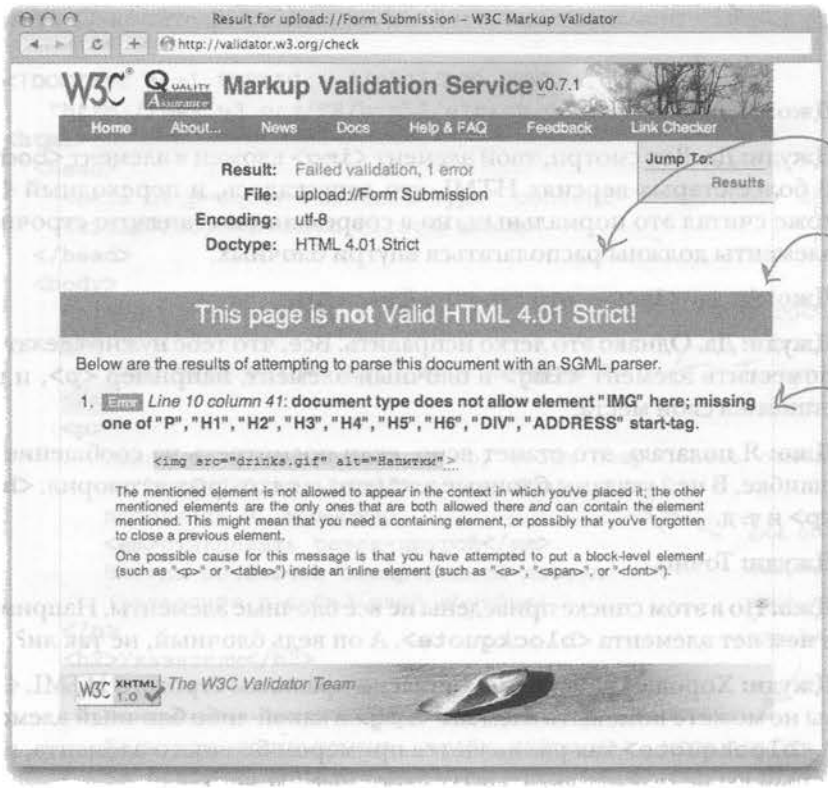
Затем замените loose.dtd на strict.dtd.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
    <title>Гостевая Head First</title>
  </head>
  <body>
    <h1>Добро пожаловать в новую и усовершенствованную гостевую Head First</h1>
      <img src="drinks.gif" alt="Напитки">
    <p>
      Заходите к нам каждый вечер, чтобы попробовать освежающие
      <a href="elixir.html">напитки</a>,
      поболтать и, возможно,
      <em>станцевать разок-другой</em>.
      Всегда обеспечен беспроводной доступ
      (захватите с собой свой ноутбук).
    </p>
    <h2>Указатели</h2>
    <p>
      Вы найдете нас в центре Webville.
      Если вам нужна помощь, чтобы найти нас, используйте наши
      <a href="directions.html">указатели</a>.
      Присоединяйтесь к нам!
    </p>
  </body>
</html>
```

Это все. Только убедитесь, что ваш DOCTYPE выглядит точно так же, как приведенный выше.

Теперь все, что осталось сделать, — попросить валидатор проверить ваш файл на соответствие строгой версии HTML 4.01. После того как убедитесь, что внесли в страницу все изменения, описанные выше, снова используйте валидатор для ее проверки.

Процедура валидации прошла успешно?



Ой-ой-ой, снова красный цвет. Явно что-то не так.

Кажется, наш документ невалидный по строгим правилам, но почему?

Давайте посмотрим на сообщение об ошибке: кажется, строгому HTML не нравится то, где мы поместили элемент <img>. Но переходный на это не ругался. Надо думать, что в строгом HTML изменились правила вложенности?





Джо: Неправильную?

Джуди: Да. Вот смотри, твой элемент **<img>** вложен в элемент **<body>**. В более старых версиях HTML это допускалось, и переходный 4.01 тоже считал это нормальным, но в современном стандарте строчные элементы должны располагаться внутри блочных.

Джо: Ах да, **<img>** – это строчный элемент.

Джуди: Да. Однако это легко исправить. Все, что тебе нужно сделать, – поместить элемент **<img>** в блочный элемент, например **<p>**, и все станет на свои места.

Джо: Я полагаю, это станет ясно, если посмотреть на сообщение об ошибке. В ней указаны блочные элементы, о которых я говорил: **<h1>**, **<p>** и т. д.

Джуди: Точно.

Джо: Но в этом списке приведены не все блочные элементы. Например, в нем нет элемента **<blockquote>**. А он ведь блочный, не так ли?

Джуди: Хорошее замечание. Согласно правилам строгого HTML 4.01, вы не можете помещать элемент **<img>** в какой-либо блочный элемент. А **<blockquote>** как раз является примером блочного элемента, в который вы не можете вкладывать строчные элементы.

Джо: Хорошо, а как можно узнать, что мы правильно используем вложенность, *перед тем как* пройти процедуру валидации? Есть ли где-нибудь список «правил вложенности»?

Джуди: Есть, но большинство из них вы легко запомните, всего лишь один раз взглянув на эти правила.

Фрэнк: Джуди, есть ли еще какие-то места, где мы неправильно применяем вложенность элементов?

Джуди: Я их не вижу. Мне кажется, что все остальное написано верно. Однако именно для этого нам и нужны валидаторы. Они никогда ничего не упускают. А люди могут упустить.

Фрэнк: Ладно, давайте исправим это и снова пройдем для страницы процедуру валидации. Ребята, я уже готов увидеть зеленый цвет, означающий, что страница валидна.

Джуди: Хорошо поработать и удачи. Кажется, вы все поняли.

Исправление ошибки вложенности

Итак, все говорит о том, что строгий HTML 4.01 «предпочитает», чтобы изображения, которые задаются строчными элементами, были вложены внутрь блочных элементов, таких как **<p>** или **<h1>**. Это нетрудно исправить. Откройте файл `lounge.html` и вложите элемент **<img>** в **<p>**.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
    <title>Гостевая Head First</title>
  </head>
  <body>
    <h1>Добро пожаловать в новую и усовершенствованную гостевую Head First</h1>
    <p>
      <img src="drinks.gif" alt="Напитки">
    </p>
    <p>
      Заходите к нам каждый вечер, чтобы попробовать освежающие
      <a href="elixir.html">напитки</a>,
      поболтать и, возможно,
      <em>станцевать разок-другой</em>.
      Всегда обеспечен беспроводной доступ
      (захватите с собой свой ноутбук).
    </p>
    <h2>Указатели</h2>
    <p>
      Вы найдете нас в центре Webville.
      Если вам нужна помощь, чтобы найти нас, используйте наши
      <a href="directions.html">указатели</a>.
      Присоединяйтесь к нам!
    </p>
  </body>
</html>
```

Теперь изображение drinks.gif благополучно вложено внутрь элемента <p>.

Все остальные строчные элементы, такие как <a> и , уже находятся внутри блочных, как эти абзацы.

Когда вы с этим справитесь, сохраните свою страницу и обновите ее в браузере. Вы увидите, что изменения никак не повлияли на внешний вид страницы. Почему? Потому что название над рисунком и абзац под ним — это блочные элементы, перед которыми и после которых добавляется разрыв строки. Поэтому новый элемент **<p>**, в который мы заключили изображение, на самом деле не добавляет новых разрывов строки или пробелов.

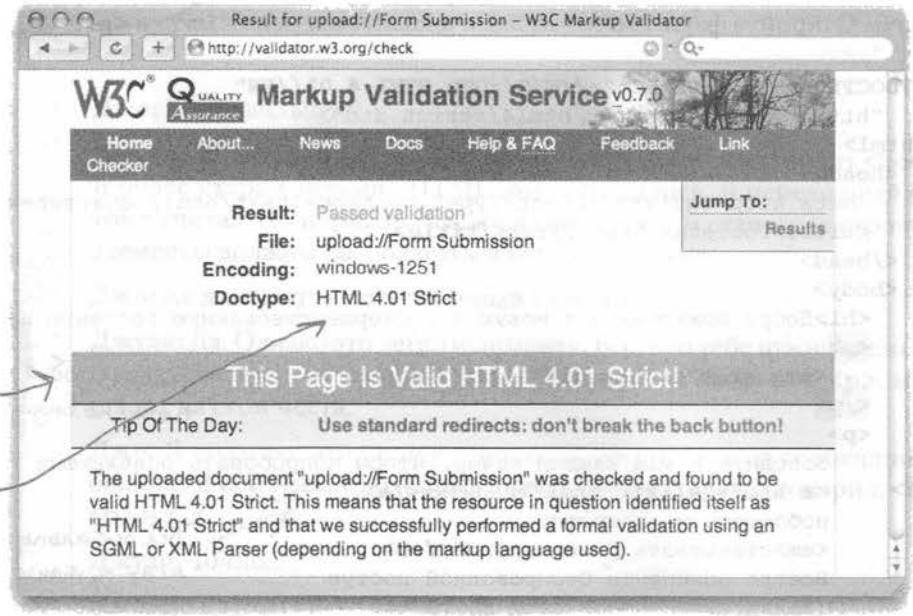


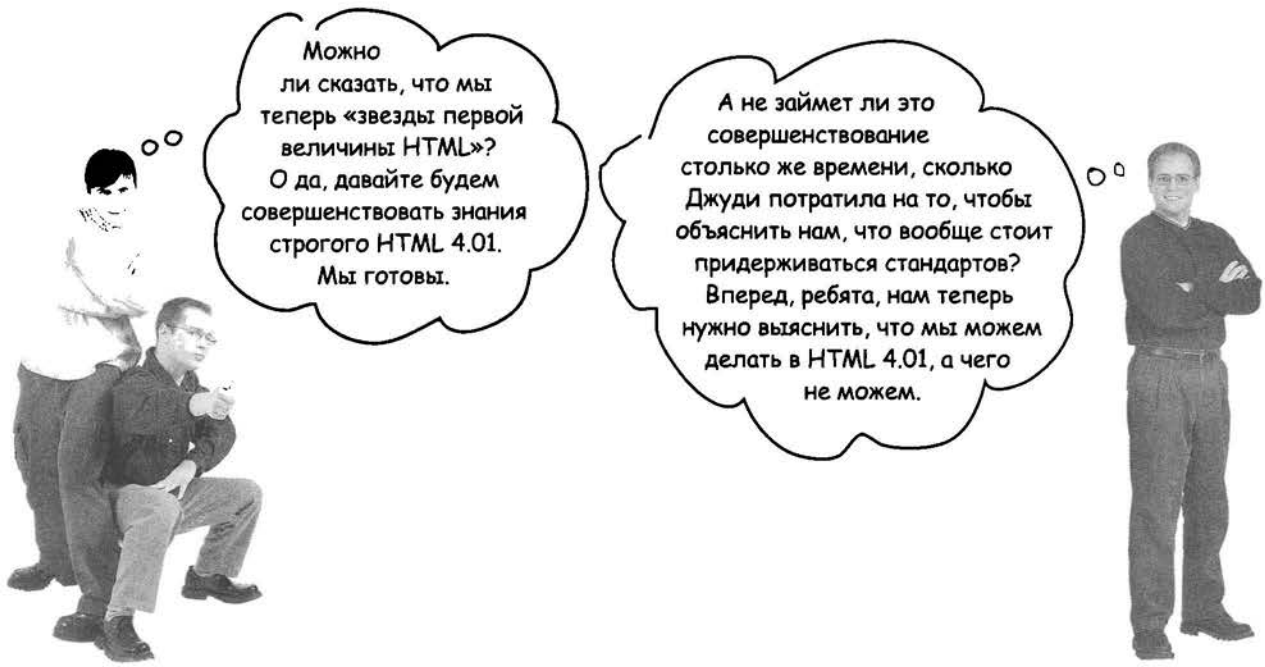
Еще один шанс стать строгим

Вы знаете, что делать. Попросите валидатор еще раз проверить ваш файл `lounge.html`.

Работает! Этот зеленый цвет означает, что проверка прошла успешно.

Теперь мы можем сказать, что файл написан на строгом HTML 4.01.





В: Хорошо, мне кажется, что я все понял. Было забавно давать валидатору проверить мой HTML, но ЗАЧЕМ? Опять же что на самом деле мне дает все это «соответствие стандартам»?

О: Как насчет того, что это радует заказчиков? Если вы знаете, что ваш HTML валиден, то у вас намного больше шансов, что веб-страницы будут одинаково хорошо работать в множестве браузеров, а это оставит лучшее впечатление о вашей веб-странице у пользователей. Есть еще несколько преимуществ: веб-страницы, написанные на соответствующем стандартам HTML, быстрее загружаются и лучше работают на других устройствах, которые в наши дни стали использоваться для работы в Сети (например, телевизоры и телефоны). Они также более доступны для людей со слабым зрением, которые применяют экранные дикторы.

В: Вы можете объяснить суть ошибки более подробно? Я хочу точно понять, с чем она связана.

О: Ошибка возникла из-за того, что элемент <img> не был вложен в блочный элемент. Представьте, что браузер читает ваш HTML и видит элемент <img> там, где ожидает увидеть блочный элемент. В такой ситуации он сразу говорит: «Эй, я ожидал увидеть здесь блочный элемент». Затем он продолжает читать дальше, достигает конца элемента <img> (это происходит, когда браузер встречает символ «>» в конце тега <img> в силу того, что элемент <img> пустой) и говорит: «Эй, вы не можете заканчивать элемент <img> здесь, потому что он вообще не может быть здесь использован».

Кроме того, возможно, вы заметите, что для одной ошибки выдается несколько сообщений. Просто исправляйте все ошибки по очереди, и может получиться так, что после исправления одной ошибки исчезнет сразу несколько сообщений.

В: Все сообщения об ошибках, выдаваемые валидатором, так сложно понять?

О: Как правило, не совсем легко понять, что говорится в сообщении об ошибке. Ведь это программа, а не человек говорит вам, что у вас неправильно. Не забывайте: валидатор не знает, что вы *имели в виду*, когда писали свой HTML-код, и он может только попытаться разгадать это и указать на ошибки, которые вы *сделали*. В большинстве случаев он укажет на строку в коде, где произошла ошибка, что уже является залогом успеха и облегчает задачу. Затем, будем надеяться, вы сами отследите свою ошибку.

Спустя какое-то время вы научитесь понимать многие сообщения об ошибках и часто будете знать, на что они указывают, даже если валидатор не говорит этого прямо.

В: А почему названия всех элементов в сообщениях об ошибках написаны прописными буквами? Я думал, что названия элементов должны писаться строчными буквами.

О: Хороший вопрос. На самом деле HTML позволяет использовать в названиях элементов как прописные, так и строчные буквы или даже прописные и строчные вместе. Вы можете написать <IMG> или даже <ImG>, если захотите. Тем не менее W3C меняет правила, и в будущем названия элементов нужно будет писать строчными буквами. Итак, хотя формально валидатор все еще позволяет писать теги прописными буквами (и отображает их), для строгого HTML 4.01 мы всегда будем использовать строчные буквы, чтобы это вошло у вас в привычку. Это значит, что вам не придется корректировать названия своих тегов в будущем (что уменьшит объем работы). И когда мы говорим «в будущем», на самом деле имеем в виду «в следующей главе».



Упражнение

Ваша очередь. Добавьте строгий DOCTYPE и тег `<meta>` в файлы `directions.html` и `elixir.html`. Попробуйте пройти для них процедуру валидации. Они валидны? Если нет, то исправьте их так, чтобы они стали таковыми.

Строгий HTML 4.01; Вам срочно нужен путеводитель

Вы уже на протяжении нескольких глав путешествуете по Webville. Не кажется ли вам, что настало время выучить местные правила дорожного движения? К счастью, в Webville приготовили удобный путеводитель по использованию строгого HTML 4.01. Он составлен специально для новичков в Webville, не содержит ненужной информации и достаточно хорошо освещает все самые важные и логичные правила. В результате вы всегда сможете узнавать что-то новое, не описанное в этом путеводителе, в процессе своего путешествия по дорогам Webville в следующих главах. Но пока возьмите его совершенно бесплатно.



Webville-путеводитель по строгому HTML 4.01

Путешествие по информационной супермагистрали может быть опасным, если вы не знаете правил дорожного движения. В этом удобном путеводителе мы сократили строгий HTML 4.01 до основного набора правил, начиная с самых главных.



Элемент <html>: не выходите из дома без него.

Всегда начинайте свою страницу с определения DOCTYPE, а сразу за ним указывайте **<html>**, который всегда должен быть корневым элементом вашей веб-страницы. Итак, после DOCTYPE тег **<html>** должен начинать вашу страницу, тег **</html>** — заканчивать, а все остальное должно быть вложено в него.



Всегда используйте <head> и <body>, чтобы HTML-код был лучше.

Только элементы **<head>** и **<body>** могут напрямую быть вложены в элемент **<html>**. Это означает, что все остальные элементы должны располагаться внутри элементов **<head>** или **<body>**. Все без исключения!



Что такое <head> без <title>?

В элементе **<head>** всегда помещайте элемент **<title>**. Это закон. Если вы этого не сделаете, то HTML будет признан не соответствующим стандартам. Элемент **<head>** — единственный, в который вы можете помещать элементы **<title>**, **<meta>** и **<style>**.



Наполняйте элемент <body> только полезными блочными элементами.

Непосредственно внутрь элемента **<body>** вы можете помещать только блочные элементы (**<h1>**, **<h2>** ... **<h6>**, **<p>**, **<blockquote>** и т. д.). Строчные элементы и текст необходимо сначала поместить в другой блочный элемент, и только затем они смогут войти в состав элемента **<body>**.



Никогда не помещайте блочные элементы внутрь строчных.

Внутрь строчных элементов вы можете помещать только другие строчные элементы или текст. Блочные элементы не могут входить внутрь строчных ни при каких обстоятельствах.

Webville-путеводитель по строгому HTML 4.01

Теперь, когда вы усвоили основные правила, давайте рассмотрим некоторые нюансы.



Не помещайте блочные элементы внутрь элемента <p>.

Абзацы предназначены для текста, поэтому не нужно помещать в них блочные элементы. Но, конечно же, любые строчные элементы можно использовать внутри абзацев (, <a>, , <img>, <q> и т. д.).



Списки предназначены для перечисления пунктов.

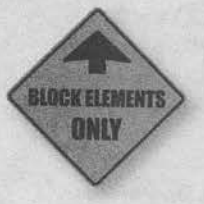
Внутри элементов <ul> и <ol> разрешается использовать только элемент <li>. Но зачем вам помещать в упорядоченный или неупорядоченный список что-то, кроме его пунктов?



Продолжим. Помещайте любые элементы внутрь элементов списка.

Законы Webville, касающиеся элемента <li>, очень либеральны: внутрь элементов списка вы можете помещать любой текст, а также строчные или блочные элементы.

Ой! Мы не придерживались стандарта 4.01, когда в главе 3 создавали элемент <blockquote> для Тони. Этот текст нужно было сначала поместить в элемент <p>.



Кто бы мог подумать? Элемент <blockquote> любит только блочные элементы.

Элемент <blockquote> требует, чтобы внутри его было не менее одного блочного элемента. В то время как мы привыкли видеть текст непосредственно внутри цитаты, это не соответствует правилам Webville. Пожалуйста, всегда помещайте ваш текст и другие строчные элементы в блочные элементы и уже эти блочные элементы добавляйте в <blockquote>.



Будьте внимательны, когда вкладываете строчные элементы друг в друга.

Несмотря на то что вы можете успешно вкладывать некоторые строчные элементы друг в друга, все же есть несколько случаев, в которых это действие нелогично и бессмысленно. Никогда не вкладывайте элемент <a> внутрь другого элемента <a>, потому что это может очень сильно запутать пользователей. И еще, не предусмотрено способа вкладывать строчные элементы в такие пустые элементы, как <img>.

Часто Задаваемые Вопросы

В: Все не так уж плохо. Я ожидал, что придется выучить множество предписаний. Действительно ли я могу писать на строгом HTML 4.01, просто следуя всем этим правилам?

О: Эти правила во многом вам помогут, но не забывайте, что вы еще не выучили все про HTML, так что есть еще несколько нюансов, которые вам нужно будет усвоить. Вообще говоря, все правила запоминать совсем не обязательно. Вашей логики и здравого смысла, а также этого путеводителя вполне достаточно для того, чтобы начать работу. Кроме того, с этого момента вы можете пользоваться справочником по HTML или просто проверять свой код с помощью валидатора (в любом случае лучше проверить!), когда у вас возникают какие-нибудь сложности.

В: А это действительно важно — всегда придерживаться строгих правил?

О: В зависимости от ситуации. Вы просто делаете страницу, которую увидят только три человека? Разве стоит заботиться о стандартах, пока во всех ваших браузерах она хорошо отображается? Но если вы создаете сайт, который будет посещать значительное количество людей, то вам лучше позаботиться о том, чтобы в вашем HTML не было отклонений от стандартов и он был валидным. Должны ли это быть переходные или строгие стандарты? Мир движется в направлении к строгим, так что либо сейчас, либо позже, но в любом случае вам придется перейти к строгим стандартам. Если вы только начинаете учить HTML, то проще сразу начать со строгих стандартов. И если вы будете писать на таком HTML, то вам будет намного проще начать писать на XHTML. Мы как раз собираемся приступить к его изучению в следующей главе, а затем использовать XHTML до конца книги.

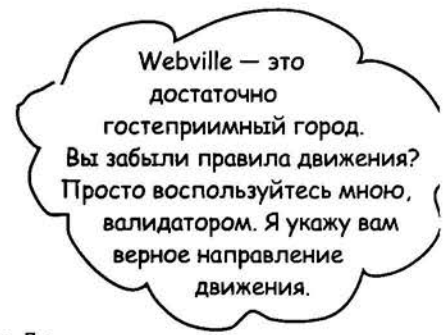
В: Я так понял, что, поместив элемент <a> в другой элемент <a>, мы создадим путаницу для пользователей, и вообще это не будет работать. А могу ли я поместить элемент в другой ? Как насчет такого вложения?

О: В принципе, возможно, кому-то понадобится в уже выделенном тексте выделить какую-то его часть еще больше. Это звучит глупо, но тем не менее никаких проблем при таком вложении не возникает (например, как при вложении элементов <a> друг в друга) и стандарт говорит, что вы можете так делать, если хотите. А как насчет элемента <q> внутри другого <q>? Может ли такое вложение иметь смысл? Возможно, вы будете цитировать кого-то, кто, в свою очередь, цитировал кого-то еще. Итак, можете вкладывать любые строчные элементы друг в друга. При этом что-то будет более логично, а что-то — менее. Единственное исключение: элементы <a> нельзя вкладывать друг в друга. Помните также, что элемент <img> пустой, поэтому вы не сможете в него ничего вложить.

В: Все же почему я не могу поместить текст непосредственно в элемент <blockquote>? А элемент списка может содержать как текст, так и блочный элемент. Мне кажется, что это нелогично.

О: Потому что таковы стандарты. Это просто особый случай. Вы правы, это кажется нелогичным, но для элементов все это имеет какой-то смысл. Возьмем, например, элемент <p>. Он используется для выделения одного абзаца текста, поэтому естественно, что в нем не может быть никаких других блочных элементов. <blockquote>? Он используется для выделения длинных цитат из других источников, которые могут содержать свои заголовки, абзацы и все, что угодно. Это объясняет наличие «цитатных блоков». Элементы списка? Это своеобразные акробаты в мире элементов, они должны уметь включать в себя как небольшие, так и большие куски текста, например, несколько абзацев и даже другие списки, то есть они могут работать со всем.

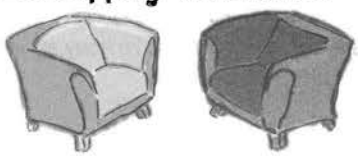
В: Валидатор сказал, что стандарт требует наличия атрибута alt в элементе <img>. А существуют ли еще какие-то обязательные атрибуты?



О: Ого, отличное замечание. Да, действительно, атрибут alt необходим для того, чтобы делать изображения общедоступными, например, с его помощью люди с ослабленным зрением могут узнать, что изображено на рисунке, не видя его. Еще один обязательный атрибут — это атрибут src для изображения, ведь элемент <img> без него вообще не имеет смысла. Есть также несколько атрибутов, которые разрешалось использовать в HTML 3.2, но которые нельзя использовать в строгом HTML 4.01. Почему? Потому что большинство из них влияло на внешний вид страниц, а сейчас предполагается, что для стилизации вы должны использовать CSS (подробнее об этом — через несколько глав).



Беседа у камина



Вечерний диалог: **Переходный и Строгий пытаются завербовать сторонников.**

Переходный

Эй, привет, Строгий. Ты здесь, чтобы поговорить о том, как сильно тебе нравится разочаровывать людей, которые пишут веб-страницы?

О, ты и сам знаешь, что я говорю о тех, кто пишет веб-страницы и потом изо всех сил пытается пройти для них процедуру валидации с твоим строгим DOCTYPE. Ты совсем не такой сговорчивый, как кажется.

Из любви?

О, я тебя умоляю. Далеко не все хотят подчиняться строгим правилам.

Знаешь, далеко не все могут и хотят разом переводить весь свой сайт на строгие стандарты. И в таких случаях я играю очень важную роль.

Как это – создать что-то перспективное, которое всегда будет работать?

Строгий

Что ты имеешь в виду?

Но я делаю это из любви к людям.

Да. Рано или поздно любая более или менее значимая страница все равно должна будет перейти на соответствие строгим стандартам. Сейчас вы можете думать, что я упрямый и несговорчивый, но в конце концов вы меня полюбите.

А? Всеми этими старыми тегами и атрибутами ты поощряешь, чтобы люди отставали от времени. Ты как костыль, который используют, только чтобы устоять и не упасть.

Вот как я это вижу: люди часто стали говорить, что они пишут на «строгом HTML», когда на самом деле они еще не избавились от старых привычек. А я говорю, строгий – значит, без всяких отклонений и старых привычек. Это единственный способ создать перспективный сайт, который будет работать в будущем.

Эй, Переходный, некоторые твои теги уже уходят в прошлое. Если же сейчас придерживаться строгих правил, то в будущем будет гораздо проще внести корректировки, чтобы соответствовать новой версии HTML.

Переходный

Так ты просто собираешься оставить позади все эти миллионы веб-страниц, которые написаны на старых версиях HTML? Полностью их проигнорировать? Я бьюсь об заклад, что ты сам используешь некоторые из этих «нестрогих» веб-страниц. Как насчет того, чтобы я пришел к тебе и проверил твой журнал?

Знаешь, далеко не все заинтересованы в том, чтобы использовать самые современные стандарты. Некоторым людям нравится использовать старые теги. Другие не хотят никуда торопиться, а хотят сначала точно понять, что представляют собой эти новые стандарты, перед тем как все бросить и волей-неволей начать менять свои страницы.

Знаешь, тебе стоило бы быть со мной повежливее, ведь я могу помочь перевести множество страниц на строгие стандарты.

Ты прав, они могут сразу начать со строгого, и тогда я им вообще не понадоблюсь. Так или иначе, мне нужно возвращаться к переводу страниц в строгие стандарты в своей более мягкой и доброй манере. А ты можешь продолжать использовать свой грубый метод.

Да. Я надеюсь, ты не забыл сказать читателям, что к концу этой главы ты тоже уже будешь считаться устаревшим?

Строгий

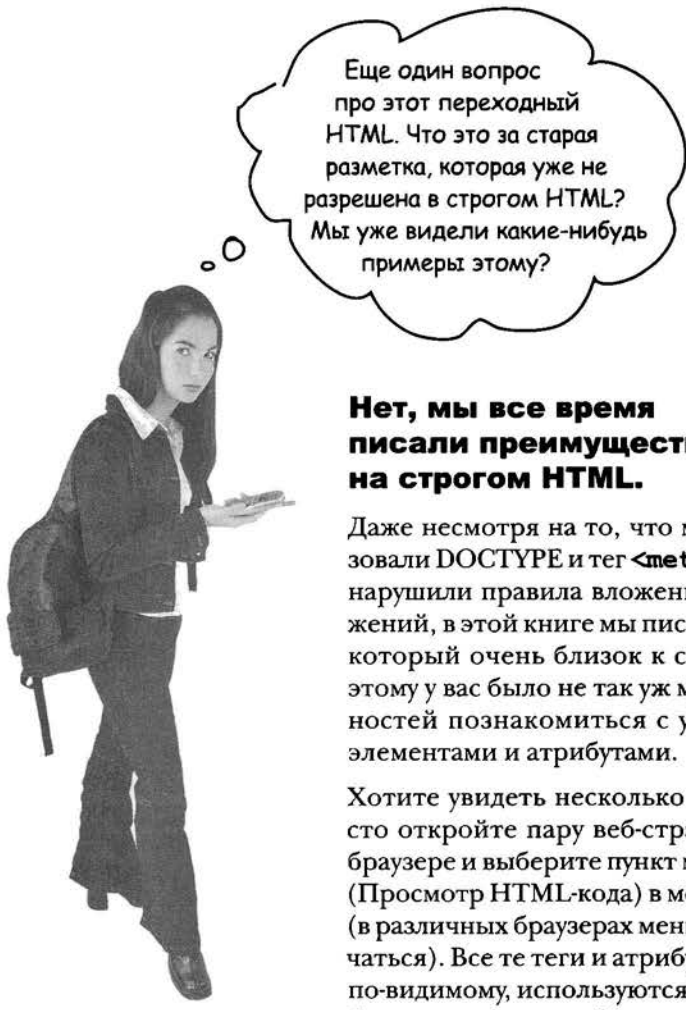
О нет, я не подпущу тебя близко к журналу в моем браузере. Ты прав, действительно есть много полезных страниц, которые нужно обновить. И, возможно, они никогда бы не были обновлены, но мы пытаемся построить новую, улучшенную Сеть. Так что прекрати поощрять людей в отставании от времени.

Волей-неволей? Вряд ли такое можно сказать применительно к 4.01. Он на самом деле проще и понятнее, чем более старые версии HTML. Кроме того, если люди будут правильно писать свои веб-страницы, то эти страницы будут долгое время хорошо работать во всех браузерах.

Ладно, было бы полезно, если бы люди могли помечать свои страницы как переходные, пока не выучат новый материал. Все, что я пытался здесь сказать, – что переходный HTML не должен использоваться как содействие тем, кто не хочет совершенствоваться. И читателям этой книги, являющимся новичками в области HTML и CSS, совсем нет необходимости начинать с переходного HTML.

Эй, поаккуратнее! В конце концов, страницы будут мне благодарны за то, что я всегда держал их в строгости.

Эх...



Нет, мы все время писали преимущественно на строгом HTML.

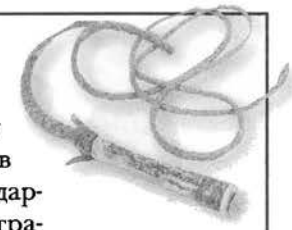
Даже несмотря на то, что мы не использовали DOCTYPE и тег **<meta>** и немного нарушили правила вложенности изображений, в этой книге мы писали на HTML, который очень близок к стандарту. Поэтому у вас было не так уж много возможностей познакомиться с устаревшими элементами и атрибутами.

И это очень хорошо, потому что самое сложное при переходе к каким-нибудь новым стандартам — избавиться от старых привычек.

Хотите увидеть несколько из них? Просто откройте пару веб-страниц в своем браузере и выберите пункт меню View Source (Просмотр HTML-кода) в меню View (Вид) (в различных браузерах меню могут различаться). Все те теги и атрибуты, которые, по-видимому, используются для того, чтобы повлиять на дизайн страницы, вероятнее всего, не применяются в HTML 4.01 (потому что теперь это работа CSS). Конечно же, не повредит узнать кое-что об этих традиционных элементах, так как существует большая вероятность, что вы будете их иногда встречать. Давайте взглянем на них...



Археология HTML



Мы немного покопались в Сети и нашли страницу на HTML 3.2, в которой есть несколько элементов и атрибутов, не являющихся больше частью стандарта. Кроме того, в ней есть несколько распространенных ошибок, которые не допускаются в строгом HTML 4.01.

```
<html>
<head>
   <title>Прогноз погоды Webville</title>
</head>

<body bgcolor="tan" text="black">

   <p>
      Прогноз погоды обещает дождь и сильный ветер в
      <font face="arial">Webville</font> сегодня, так что
      по возможности не выходите на улицу.
   </p>

   <ul>
      <li>Вторник: дождь и 15°C.
      <li>Среда: дождь и 16°C.
   </ul>

   <p align=right>
     Возьмите с собой зонт!

   <center><font size="small">Эта страница предоставлена вам организацией
   Lou Diner, существующей в Webville на протяжении 50 лет.
   </font></center>

</body>
</html>
```

Это атрибуты, отвечающие за дизайн страницы: bgcolor устанавливает цвет фона страницы, а text — цвет самого текста внутри элемента body.

Шрифт был изменен с помощью элемента <font> и его атрибута face.

Вы можете обойтись без некоторых закрывающих тегов, таких как и </p>.

Или даже без двойных кавычек вокруг значений атрибутов.

Размер шрифта установлен элементом <font> в атрибуте size.

Здесь приведены два способа выравнивания текста. Выравнивание по правой стороне абзаца и центрирование отдельного отрывка текста.



Поработай валидатором

Ниже приведен HTML-файл. Ваша задача — представить, что Вы валидатор, и найти все ошибки. Выполнив задание, посмотрите ответы в конце главы и убедитесь, что нашли все ошибки.

Когда справитесь с заданием, используйте валидатор, чтобы проверить свою работу (или, если хотите, используйте подсказки).

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
</head>
<body>
   <img src="chamberofcommerce.gif">
   <h1>Подсказки: как сделать так, чтобы визит в Webville
   доставил вам больше удовольствия</h1>
   <p>
      Это несколько подсказок, которые помогут вам в полной мере насладиться
      пребыванием в Webville.
      <ul>
         <li>Всегда потеплее одевайтесь и защищайте свои head (голову)
             и body (тело) с помощью html.</li>
         <li>Как можно больше отдыхайте, пока здесь находитесь. Сон помогает
             усвоить все эти правила.</li>
         <li>Не упустите возможность посмотреть работы наших местных художников
             в бизнес-центре в галерее CSS.</li>
      </ul>
   </p>
   <p>
      У вас возникли какие-то вопросы? Вы всегда можете найти ответы в
      <a href="http://www.headfirstlabs.com"><em>лаборатории Head First</em></a>.
      Вопросы все еще остались? Расслабьтесь, Webville – это достаточно гостеприимный
      город. Просто попросите кого-нибудь вам помочь. И, как здесь принято говорить:
   </p>
   <blockquote>
      Не волнуйтесь. Все будет в порядке.
   </blockquote>
</body>
</html>
```



ПОВТОРИМ выученное



- HTML 4.01 — это стандарт HTML, наиболее широко поддерживаемый браузерами.
- Консорциум Всемирной паутины (W3C) — это организация, занимающаяся стандартами. Она же определяет, что такое «стандарт HTML».
- У многих браузеров есть два режима отображения HTML: режим обратной совместимости для старого HTML и стандартный режим для HTML 4.01.
- Если не указать, какую версию HTML вы используете, многие браузеры будут применять режим обратной совместимости, что может привести к нестабильному отображению страницы.
- Определение типа документа (DOCTYPE) используется, чтобы сказать браузеру, на какой версии HTML написана ваша веб-страница.
- Строгий DOCTYPE применяется, если вы пишете на HTML 4.01, полностью соответствующем стандартам.
- Используйте переходный DOCTYPE, если в вашем HTML все еще встречаются элементы и атрибуты, отвечающие за дизайн страницы.
- Тег **<meta>** в элементе **<head>** представляет браузеру дополнительную информацию о веб-странице, например о типе документа и его кодировке.
- Кодировка символов говорит браузеру, какой набор символов применяется на странице.
- Валидатор W3C — это бесплатная сервисная программа, которая проверяет страницы на соответствие стандартам.
- Пользуйтесь валидатором, чтобы убедиться, что ваш HTML написан грамотно, а элементы и атрибуты соответствуют стандартам.
- Если вы будете твердо придерживаться стандартов, то ваши страницы будут быстрее отображаться и их внешний вид будет мало различаться при отображении в разных браузерах.



Решение упражнения



Поработай валидатором

Ниже приведен HTML-файл. Ваша задача — представить, что Вы валидатор, и найти все ошибки. Вот решение упражнения.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
</head>
<body>
   <img src="chamberofcommerce.gif">
   <h1>Подсказки: как сделать так, чтобы визит в Webville
   доставил вам больше удовольствия</h1>
   <p>
      Это несколько подсказок, которые помогут вам в полной мере насладиться
      пребыванием в Webville.
      <ul>
         <li>Всегда потеплее одевайтесь и защищайте свои head (голову)
             и body (тело) с помощью html.</li>
         <li>Как можно больше отдыхайте, пока здесь находитесь. Сон помогает
             усвоить все эти правила.</li>
         <li>Не упустите возможность посмотреть работы наших местных художников
             в бизнес-центре в галерее CSS.</li>
      </ul>
   </p>
   <p>
      У вас возникли какие-то вопросы? Вы всегда можете найти ответы в
      <a href="http://www.headfirstlabs.com"><em>лаборатории Head First</em></a>.
      Вопросы все еще остались? Расслабьтесь, Webville – это достаточно гостеприимный
      город. Просто попросите кого-нибудь вам помочь. И, как здесь принято говорить:
   </p>
   <blockquote>
      Не волнуйтесь. Все будет в порядке.
   </blockquote>
</body>
</html>
```

В <head> должен присутствовать элемент <title>.

Строчный элемент напрямую вложен в элемент <body>.

Нет атрибута alt.

Блочный элемент внутри элемента <p>.

<blockquote> может содержать только блочные элементы

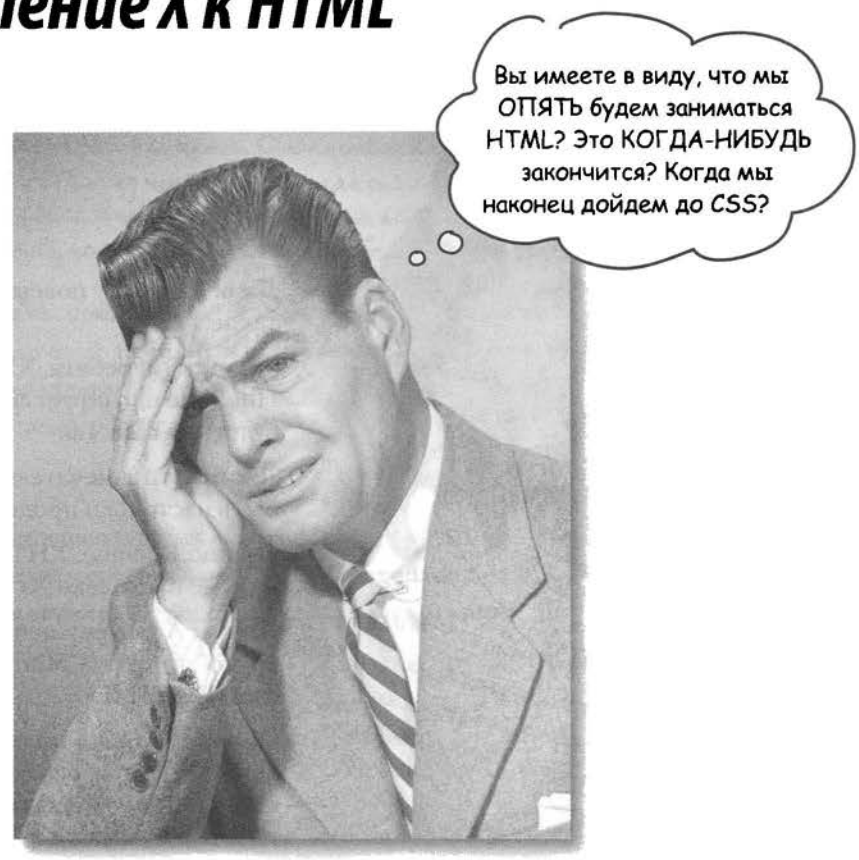


Решение упражнения

Ваша очередь. Добавьте строгий DOCTYPE и тег <meta> в файлы `directions.html` и `elixir.html`. Попробуйте пройти для них процедуру валидации. Они валидны? Если нет, то исправьте их так, чтобы они стали валидными.

Решение: Чтобы успешно пройти процедуру валидации для файла `elixir.html`, нужно добавить атрибут `alt` в каждый элемент <img>.

Добавление X к HTML



Мы вас обманули. Мы знаем, что вы думали, будто покупаете книгу по HTML, но на самом деле это лишь маскировка для книги по XHTML. В действительности все это время мы в основном учили вас XHTML. Вам, наверное, интересно, что это такое? Ладно, познакомьтесь с eXtensible (расширяемым) HTML, также известным как XHTML, следующим этапом в развитии HTML. Он более «скромный» и еще сильнее нацелен на совместимость с браузерами на широком спектре устройств. В этой небольшой главе мы планируем перейти от HTML к XHTML в три простых этапа. Итак, переверните страницу и приступайте... (а после этого мы примемся за CSS).



Джо: Я не могу поверить, что *наш менеджер* знает, что это такое.

Фрэнк: Эй, ребята, XHTML – это новый стандарт HTML. Никто не планирует создавать HTML 5; новым стандартом будет XHTML 1.0.

Джим: Это замечательно, но действительно ли нам нужно быть настолько продвинутыми?

Фрэнк: Вообще, XHTML 1.0 появился в начале XXI века, и он не настолько уж современен, как кажется.

Джим: А почему добавлена буква «X»? Потому что это круто звучит? X-Men, X-Games, X-Files, gen-X, а теперь еще и X-HTML?

Фрэнк: Хорошо подметил, Джим, но это не совсем так. «X» в XHTML обозначает eXtensible (расширяемый) – это еще один способ указать, что этот язык основан на чем-то, называемом XML.

Джо: А не используют ли ребята, занимающиеся программным обеспечением, это для хранения наших данных?

Фрэнк: Да, конечно, используют. XML означает «расширяемый язык разметки».

Джо: Ой-ой, что-то уж очень похоже на «язык гипертекстовой разметки».

Фрэнк: Да, Джо. XML – это язык разметки, как и HTML, но с его помощью вы можете делать намного больше, чем просто «размечать» веб-страницы. Позвольте мне вам это показать.

Что такое XML?

Итак, сейчас мы сделаем большой шаг назад и разберемся с тем, что такое XML (не путайте с XHTML). Начнем...

Давайте для сравнения будем использовать HTML. В HTML у вас есть строго определенный набор элементов, которыми вы можете пользоваться. Итак, если вы просто хотите сами придумать элемент, например **<cool>**, чтобы заключить в него какую-то часть содержимого, то в HTML это невозможно, а в XML – *возможно*. На самом деле с помощью XML вы можете изобрести полностью новый язык разметки. Рассмотрим пример:

Это корневой элемент. Он называется не <html>, а <recipe>, так как этот XML-код предназначен для описания кулинарных рецептов. Обратите внимание, что в нем есть несколько дополнительных атрибутов, которые вы никогда не видели в элементе <html> в HTML.

```
<recipe xmlns="http://www.foodnetwerk.com/recipe" lang="ru" xml:lang="ru">
   <name>Охлажденный зеленый чай гостевой Head First</name>
   <description>Шипучий бодрящий охлажденный чай.
                Мы подаем его весь день.
   </description>
   <ingredients>
      <ingredient measurement="6 стаканов">вода</ingredient>
      <ingredient measurement="2 пакетиков">зеленый чай</ingredient>
      <ingredient measurement="2 пакетиков">графский чай</ingredient>
      <ingredient measurement="6 кубиков">лед</ingredient>
   </ingredients>
   <preparation>
      <time duration="10 минут" />
      <step>Доведите до кипения один стакан воды в специальной
      кастрюле, уберите ее с огня и добавьте чай. Дайте ему настояться
      в течение 5 минут.</step>
      <step>Добавьте в кувшин лед, чай, а затем 5 стаканов
      холодной воды.</step>
      <step>Хорошо это все перемешайте, после чего можно подавать
      к столу. Хорошенько взболтайте чай в шейкере.</step>
   </preparation>
</recipe>
```

Ого, только взгляните на теги. Больше нет тегов <h1> и <p>, а вместо них появились <recipe>, <name>, <description>, <ingredient>, <preparation> и т. д.

Просто взглянув на названия элементов, можно догадаться, что речь идет о рецепте.

Этот пустой элемент выглядит немного странно. Мы вернемся к нему чуть позже.

Несмотря на другие названия элементов, выглядят и используются они точно так же, как в HTML (открывающие теги, закрывающие теги и т. д.).

МОЗГОВОЙ ШТУРМ

Подумайте, как бы вы создали веб-страницу для описания рецепта приготовления чая, используя HTML. Чем бы она отличалась от той, что написана на XML?

Что произойдет с HTML?

Если XML – это язык, который может быть использован для изобретения новых языков разметки, а HTML и есть язык разметки, то можем ли мы использовать XML для *воссоздания* HTML? Конечно, можем. Но перед тем, как мы начнем разбираться, зачем вообще все это нужно, давайте посмотрим, как это выглядит:

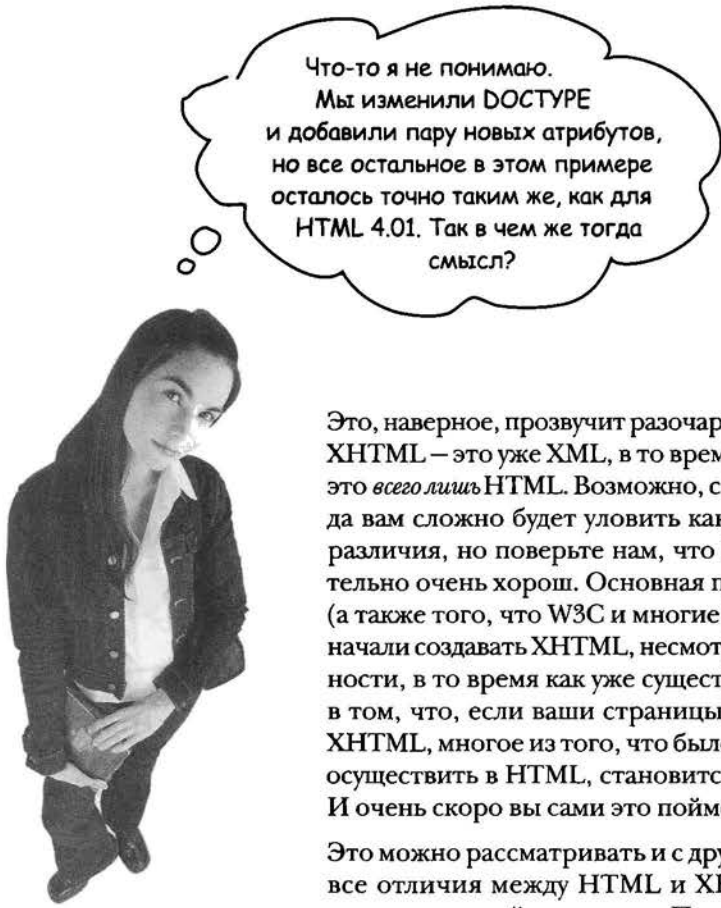
Это DOCTYPE. Вы уже видели его раньше. Обратите внимание, что сейчас мы пишем XHTML 1.0 вместо HTML 4.01.

У элемента <html> сейчас появился атрибут xmlns, а также атрибуты lang и xml:lang.

```
<!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="ru" xml:lang="ru">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=windows-1251"/>
    <title>Гостевая Head First</title>
  </head>
  <body>
    <h1>Добро пожаловать в новую и усовершенствованную гостевую Head First</h1>
    <p><img src="drinks.gif" alt="Напитки" /></p>
    <p>
      Заходите к нам каждый вечер, чтобы попробовать освежающие
      <a href="elixir.html">напитки</a>,
      поболтать и, возможно,
      <em>станцевать разок-другой</em>.
      Всегда обеспечен беспроводной доступ
      (захватите с собой свой ноутбук).
    </p>
    <h2>Указатели</h2>
    <p>
      Вы найдете нас в центре Webville.
      Если вам нужна помощь, чтобы найти нас, используйте наши
      <a href="directions.html">указатели</a>.
      Присоединяйтесь к нам!
    </p>
  </body>
</html>
```

Здесь все выглядит привычно, кроме того, что в пустом элементе на конце используются символы «/».

Однако весь оставшийся HTML выглядит точно так, как и строгий HTML 4.01. Здорово, XHTML очень похож на HTML.



Это, наверное, прозвучит разочаровывающе, но XHTML – это уже XML, в то время как HTML – это *всего лишь* HTML. Возможно, с первого взгляда вам сложно будет уловить какие-то особые различия, но поверьте нам, что XML действительно очень хорош. Основная причина этого (а также того, что W3C и многие другие все же начали создавать XHTML, несмотря на все сложности, в то время как уже существовал HTML) в том, что, если ваши страницы написаны на XHTML, многое из того, что было невозможно осуществить в HTML, становится возможным. И очень скоро вы сами это поймете.

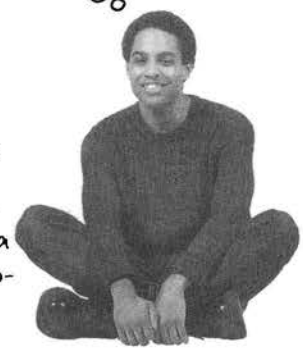
Это можно рассматривать и с другой стороны: все отличия между HTML и XHTML видны на предыдущей странице. Появился новый DOCTYPE, мы внесли несколько небольших изменений в атрибуты, а также по-другому написали пустой элемент. Эти маленькие изменения – все, что необходимо для перевода HTML в XHTML.

Теперь, когда вы знаете, что перейти к XHTML так просто, самое время разобраться, какие преимущества он может вам дать.

Итак, в чем же преимущества использования XHTML?

Используя строгий HTML 4.01, вы уже наверняка увидели некоторые преимущества XHTML. Однако поскольку XHTML — *это XML*, то он имеет некоторые интересные возможности, которых нет у HTML 4.01. Рассмотрим все то, что может дать вам XHTML, глазами людей, которые уже используют его.

Благодаря XHTML я создаю более перспективные веб-страницы и использую преимущества самых последних разработок. Кроме того, повышается вероятность корректного отображения моих страниц на мобильных устройствах и в множестве браузеров.



Человек, хобби которого — поддержка популярного игрового сайта.

Строгий синтаксис XHTML позволяет экранным дикторам более легко представлять веб-содержимое для людей со слабым зрением.



Веб-пользователь с плохим зрением.

Мне нравится быть в курсе всех новых тенденций и технологий и пользоваться ими. XHTML — это будущее, а поскольку он почти не отличается от HTML, почему бы не использовать новую технологию?



Имеет собственный блог

В отличие от HTML, XHTML может быть расширен до использования новой разметки. Например, уже есть расширения, которые добавляют элементы для векторной графики и математических формул.



Математик-исследователь, работает в крупном университете

XHTML становится языком, который выбирается для браузеров, мобильных устройств и сотовых телефонов. В будущем XHTML позволит нам быть более требовательными при выборе конкретного языка для каждого мобильного устройства.



Специалист коммерческого отдела, компания по созданию сотовых телефонов.

Уже имеется достаточно большое количество данных и информации в формате XML, и они намного легче преобразуются в XHTML, чем в HTML. Итак, с помощью XHTML намного проще поместить всю эту информацию в Сеть.



Специалист по базам данных, СМИ.

XHTML может быть прочитан теми существующими программами, которые уже знают, как читать XML.



Младший разработчик программного обеспечения.

XHTML дает нам все преимущества XML (что здорово помогает в хранении больших структурированных наборов документов) наряду с преимуществами HTML, такими как CSS для создания дизайна.



Библиотекарь большой столичной библиотеки.

Вы совсем близки к тому, чтобы начать использовать XHTML

Несмотря на то что HTML и XHTML очень похожи друг на друга, все же есть несколько небольших различий. Здесь мы приводим удобный список тех действий, которые необходимо выполнить для перехода от строгого HTML 4.01 к строгому XHTML 1.0.

Контрольный список XHTML 1.0

Это список тех изменений, которые вам нужно внести, чтобы преобразовать HTML в XHTML.

- ❑ Измените свой DOCTYPE на строгий XHTML. Вы также можете использовать переходный XHTML, если все еще пишете на переходном HTML.
- ❑ Добавьте атрибуты `xmlns`, `lang` и `xml:lang` в открывающий тег **`<html>`**.
- ☑ Открывающий тег **`<html>`** должен быть первым после DOCTYPE, а закрывающий тег **`</html>`** — последним тегом документа.
- ☑ Названия всех элементов должны быть написаны строчными буквами.
- ❑ Для каждого открывающего тега должен быть свой закрывающий. Или, если элемент пустой, тег должен заканчиваться пробелом и затем символами «/>».
- ☑ Значения всех атрибутов должны быть непустыми. Кроме того, их следует взять в двойные кавычки.
- ☑ Не используйте символ «&» в своем HTML, так как он применяется для обозначения начала кода символов. Вместо самого символа «&» используйте его код, собственно, как и для остальных специальных символов.

Здесь приведены те требования, с которыми вы уже хорошо знакомы. Поэтому для перехода к XHTML 1.0 вам осталось сделать не так уж много.

Мы подробнее поговорим об этом чуть позже.

Если вы до сих пор внимательно читали эту книгу и старательно писали на HTML 4.01, то переход к XHTML 1.0 будет для вас быстрым. Вам действительно нужно позаботиться всего о нескольких вещах, о которых мы поговорим чуть позже.

С другой стороны, если у вас есть много документов, написанных на традиционном HTML, а вы хотите преобразовать их в XHTML, то придется выполнить немало работы. Но даже в этом случае есть несколько средств, которые вам помогут. О них мы также поговорим чуть позже.



Верно. Контрольный список предполагает, что до сих пор вы писали на строгом HTML.

Строгий HTML 4.01 и строгий XHTML 1.0 по своей сути одинаковы. Итак, переход от традиционного HTML 4.01 к строгому HTML или XHTML предполагает примерно одинаковое количество работы. Чтобы перейти от традиционного HTML к чему-нибудь строгому, необходимо внести все изменения, о которых говорилось в главе 6: убрать теги, отвечающие за дизайн страницы, и привести в порядок HTML-код.

Существует также традиционная версия XHTML 1.0, которая, по сути, представляет собой почти то же самое, что и традиционная версия HTML 4.01. Они обе допускают использование запрещенных элементов, а также добавление строчных элементов непосредственно в тело вашей страницы. Итак, если вы предпочитаете эту версию, не забудьте указать переходный XHTML 1.0 DOCTYPE вместо строгого DOCTYPE.

Переход от строгого HTML k XHTML 1.0 за три шага

1. **Измените свой DOCTYPE, указав его для строгого XHTML 1.0.**

Вы уже все знаете о DOCTYPE и привыкли к типу документа – HTML 4.01 Strict. Это хорошо, но есть еще тип документа для строгого XHTML 1.0, и вам нужно будет поменять ваш DOCTYPE, чтобы его использовать. Он должен будет выглядеть следующим образом:

DOCTYPE – это общедоступный тип документа.

Это для версии XHTML 1.0 Strict языка XHTML.

```
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Здесь есть URL, указывающий на описание XHTML 1.0 Strict

2. **Добавьте атрибуты xmlns, lang и xml:lang в элемент <html>.**

Помните, что XML может быть использован для создания множества других языков разметки (которые принято называть словарями), кроме XHTML. Чтобы содержать их в порядке, XML должен знать, о *каком именно* словаре вы говорите, когда используете элемент **<html>** (в конце концов, кто-нибудь может придумать с помощью XML свой собственный словарь и назвать его Hippo Tipping Markup Language, что послужит причиной большой неразберихи). Итак, четко все указывайте. Атрибут `xmlns` определяет, *какому* словарю принадлежит элемент **<html>**. А как насчет остальных элементов внутри элемента **<html>**? По умолчанию они наследуют атрибут `xmlns` своего родителя.

Элемент **<html>** также должен содержать атрибуты `lang` и `xml:lang`, которые указывают, какой язык применяется в XML-документе. Рассмотрим, как должен выглядеть открывающий тег **<html>** в XHTML:

```
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
```

Атрибут xmlns используется для того, чтобы указать, какому из словарей XML принадлежит элемент html.

XML использует URL в качестве уникального идентификатора для словаря. Если кто-то напишет Hippo Tipping Markup Language, то ему нужно будет задать http://www.hippotipping.com/html в качестве идентификатора. И неважно, что это за URL. Самого по себе URL-адреса достаточно, чтобы говорить об уникальности словаря.

Осталось указать, что мы используем русский язык. В зависимости от того, каким способом ваш браузер интерпретирует XHTML, вы можете использовать либо оба этих атрибута, либо один из них, но все же, как показывает практика, лучше указывать оба.

3 **Все закрывающие теги должны заканчиваться символами «/», а не одним символом «>».**

Это последний и самый странный шаг преобразования HTML в XHTML 1.0. Однако он не кажется настолько странным, если знать всю историю вопроса.

Ранее мы говорили вам, что XHTML более строгий, чем HTML, но единственное, с чем он строже, – с закрывающими тегами. В HTML вы можете оставить пустой элемент без закрывающего тега. Но если вы не используете закрывающий тег в XHTML, вам придется сообщить об этом браузеру, поставив слэш перед замыкающим символом «>». Возьмем, к примеру, элемент **
**. В HTML мы просто пишем **
**, а в XHTML мы должны писать **
**. Этот малозаметный слэш в конце говорит браузеру, что он не должен ожидать закрывающего тега, потому что **
** – это все, что есть у элемента.

Вы, возможно, заметили, что мы не поставили пробел перед символами «/>». В XHTML этого и не требуется. Тем не менее некоторые старые версии браузеров не могут распознать символы «/>» без пробела перед слэшем, поэтому для совместимости с такими браузерами просто добавляйте пробел.

Рассмотрим несколько примеров преобразования пустых элементов HTML в пустые элементы XHTML.

Строгий HTML 4.01 старой школы

```
<br>
```

```
<img src="drinks.gif" alt="Напитки">
```

Нет закрывающего тега? Не проблема для HTML.

Новый и усовершенствованный XHTML 1.0

```
<br />
```

```
<img src="drinks.gif" alt="Напитки" />
```

В XHTML нужно заявлять о своих намерениях. Если ваш элемент пустой, дайте браузеру об этом знать, указав символ «/» перед закрывающей скобкой «>».

Дайте старым браузерам шанс, вставив символ пробела перед слэшем.



Упражнение

Вам нужно будет взять страницу с дневником Тони (помните его из главы 3?) и преобразовать ее код в XHTML. Мы уже обновили код и преобразовали его в строгий HTML 4.01. Мы вложили элементы `<img>` внутрь элементов `<p>` и добавили атрибуты `alt`, а также поместили слоганы Burma Shave в элементы `<p>` и добавили тег `<meta>`. В папке `chapter7/journal` вы найдете файл `journal.html`, написанный на строгой версии HTML 4.01.

Вам нужно сделать следующее.

1. Поменяйте DOCTYPE с HTML 4.01 Strict на XHTML 1.0 Strict.
2. Добавьте атрибуты `xmlns`, `lang` и `xml:lang` в открывающий тег `<html>`.
3. Измените закрывающую скобку «>» в пустых элементах на сочетание символов «/>».
4. Сохраните страницу и обновите ее в браузере.

Узнать, правильно ли вы выполнили задание, можете из ответов, приведенных в конце главы.

Часто Задаваемые Вопросы

В: Не могли бы вы более подробно рассказать об атрибуте xmlns? Мне кажется, я что-то пропустил.

О: Что ж, неудивительно, потому что это один из самых сложных моментов в XML. Этот атрибут работает так: многие люди могут создавать XML-словари (лично мы используем уже существующие словари, но кто-то может серьезно заниматься созданием своих собственных). Предположим, что два человека назвали свои элементы одинаково. Возьмем, к примеру, название <table>. Для одних людей это HTML-элемент; для других — одна из частей содержимого языка XML. Итак, если вы используете <table> в своем XML, как нам узнать, что именно вы имеете в виду? Вот тут и вступает в действие атрибут xmlns. Он содержит уникальный идентификатор, указывающий, какой язык вы имеете в виду. Для XHTML такой идентификатор — http://www.w3.org/1999/xhtml.

В: Подождите, это ведь URL-адрес, а не идентификатор.

О: Да, эту странность придумали разработчики XML. Для вас это может выглядеть как URL, но вы просто воспринимайте его как что-то такое, что должно быть уникальным. Идея в том, что вы можете посетить этот URL и разузнать все, что нужно, о данном словаре, хотя на самом деле нет требования, чтобы что-то реально существовало по этому адресу.

В: Как так случилось, что в XHTML корневой элемент все же <html>, а не <xhtml>?

О: Предполагается, что XHTML должен быть совместим с более ранними версиями HTML. Если поменять корневой элемент на <xhtml>, то более старые версии браузеров не будут знать, как отображать вашу страницу.

В: Чуть раньше вы упоминали о программных средствах, которые могут помочь преобразовать HTML в XHTML.

О: Да, существует замечательная небольшая сервисная программа Tidy, которая может выполнить большой кусок работы по подготовке ваших HTML-документов к успешной валидации и преобразованию в XHTML. У Tidy есть множество параметров, позволяющих преобразовать невалидный HTML в валидный. Она также может найти HTML-код, унаследованный из старых версий и используемый для дизайна страницы, и заменить его CSS. Скачать Tidy можно по адресу http://tidy.sourceforge.net.

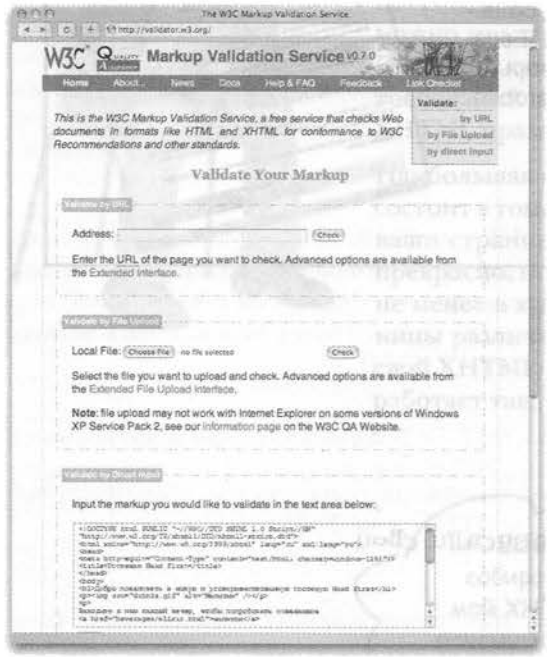
В: Итак, если у меня есть строгий HTML, то действительно ли этого достаточно? Что я должен сделать, чтобы преобразовать его в XHTML?

О: Вообще, достаточно. Но давайте проверим это.

Валидация используется не только для HTML

После изучения главы 6 вас уже можно назвать специалистом по использованию W3C-валидатора, а теперь вы еще узнаете, что он отвечает современным требованиям и готов приступить к валидации вашего XHTML. Эта процедура точно такая же, как и валидация HTML.



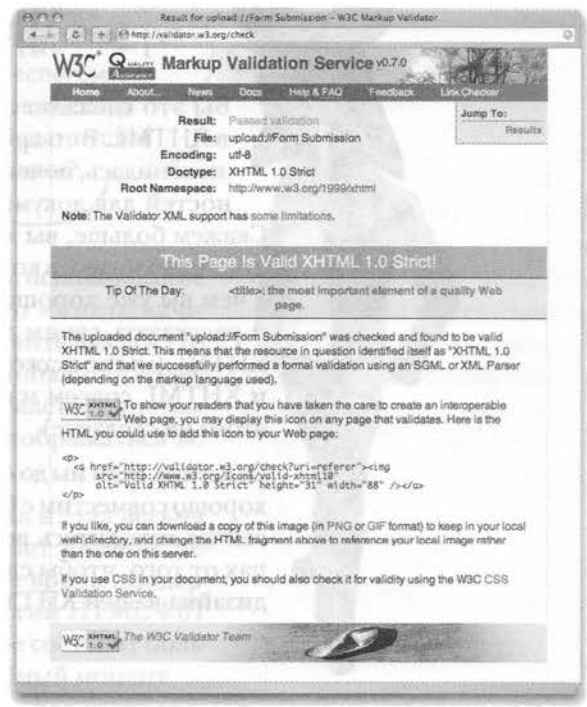


Откройте сайт validator.w3.org и либо вставьте свой XHTML-код, загрузив его на сайт со своего компьютера, либо укажите валидатору свой URL-адрес.

Валидатор проверит ваш XHTML и сообщит, что он валидный, либо представит отчет о найденных ошибках.

Валидатор





Часто Задаваемые Вопросы

В: Как валидатор узнает, для чего я хочу выполнить валидацию: для HTML или XHTML? В конце концов, это ведь та же страница, которую я использовал для HTML.

О: Валидатор смотрит на объявленный вами DOCTYPE, который точно определяет, на чем написан ваш документ: на переходном или строгом XHTML. С учетом этого он выполняет валидацию.



Решение упражнений

Вы же не думаете, что мы оставим вас в покое и позволим обойтись без валидации для XHTML? Проверьте файл `lounge.html` из папки `chapter7/lounge` и файл `journal.html` из папки `chapter7/journal/` (которые вы преобразовывали в XHTML несколько страниц назад) в W3C-валидаторе. Если обнаружатся какие-то ошибки, проверьте, нет ли опечаток, исправьте их и повторно проверьте документ в валидаторе.





Поздравляем, вы только что написали свой первый XHTML!

Вы это сделали: вы преобразовали свой HTML в XHTML. В то время как ваша разметка несильно изменилась, появилась целая серия новых возможностей для документов, написанных на XHTML. Скажем больше, вы теперь можете использовать целую технологию, в которой очень много общего с тем, с чем вы уже хорошо знакомы. Самое время пойти и рассказать своим друзьям, что вы *уже используете* XHTML (мы не скажем им, что различий между HTML и XHTML совсем мало, если, конечно же, вы сами этого не сделаете).

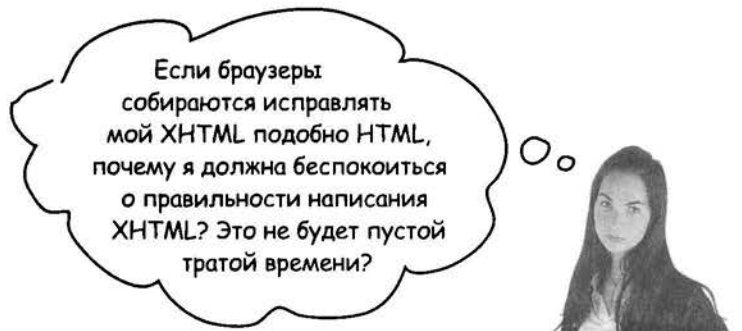
О, кстати, если мы до сих пор этого не сказали, XHTML хорошо совместим с CSS для оформления сайтов и вы сейчас находитесь всего лишь в нескольких страницах от того, чтобы сделать первый шаг для создания дизайна вашей XHTML-страницы.





XHTML действительно хорошая вещь, и перейти от строгого HTML 4.01 можно практически незаметно, но почему только все не делают этого? Перед тем как выполнить этот шаг, вы должны знать, что в XHTML больше требований к коду, чем в HTML. Так что вы можете начать использовать XHTML прямо сегодня, но учитывайте несколько особенностей.

Наибольшая проблема, с которой вы можете столкнуться прямо сейчас, состоит в том, что некоторые браузеры еще будут продолжать исправлять ваши страницы как написанные на HTML. В большинстве случаев это прекрасно, поскольку XHTML разработан как совместимый с HTML. Тем не менее в худшем случае браузер может показывать ваши XHTML-страницы различными способами. Что же делать? Лучше всего тестировать свой XHTML-код в множестве браузеров, чтобы удостовериться, что все работает так, как вы ожидаете.

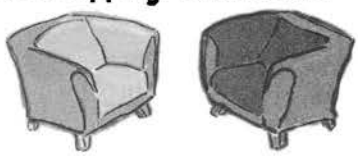




На самом деле лучше сразу решить, выгодно ли для вас использование XHTML. Если ваш ответ будет положительным, то можете начинать работать с XHTML уже сегодня, только сразу учитывая все его требования. Это поможет вам в дальнейшем, когда появятся более строгие браузеры. Если ваши страницы будут оформлены с учетом всех требований языка, они будут превосходно отображаться во всех браузерах.

Язык HTML «прожил» уже довольно длинную жизнь и стал весьма «опытен» во многих вопросах, поэтому, если у вас нет желания переходить к XHTML, можете продолжать некоторое время придерживаться HTML. Если вы будете использовать строгий HTML 4.01 и проверять свои страницы на валидность, то вам не составит большого труда переключиться на XHTML в любой удобный момент.

Беседа у камина



Вечерний диалог: **HTML и XHTML нуждаются в вашей поддержке.**

HTML

Я, HTML 4.01, конечно, очень рад этой возможности убедить вас продолжать со мной работать. Я буду оставаться популярным еще долгое время, можете в этом не сомневаться.

Между нами не так уж много различий, чтобы людей это волновало. Я имею в виду, что 4.01 – это абсолютно то же самое, что и XHTML 1.0.

Но ведь тебе это вообще ничего не дает, ты не получишь за это даже чашку кофе.

В этом-то и проблема: ты думаешь, что всем *нужны* прикладные программы, использующие XHTML, или что все создают сайты для мобильных устройств. Но некоторые люди просто хотят делать хорошие сайты. Зачем ты просишь их прилагать так много напрасных усилий?

XHTML

HTML, признай, что ты – вчерашний день. Ребята, занимающиеся стандартами, уже двигаются дальше. Будущее за мной. Любой нормальный человек будет переходить к использованию XHTML.

Как ты можешь говорить, что мы одинаковы? Ты – HTML, а я – XML.

Эй, минуточку. Количество устройств, которые могут читать XHTML, растет с каждым днем. Кроме того, в мире существует множество прикладных программ, в которых можно использовать XHTML.

На самом деле никаких особых усилий прилагать не нужно. Если вы уже используете HTML 4.01, то до XHTML вам рукой подать. Все, что вам нужно сделать, – изменить DOCTYPE и добавить несколько атрибутов в элемент **<html>**. Итак, что тут трудного? Почему бы не использовать все самое прогрессивное и современное, если для этого нужно потратить всего несколько минут?

HTML

Но ты все же кое о чем забыл. Многие браузеры не умеют хорошо обрабатывать XHTML. Они просто воспринимают его как HTML. Итак, вы выполняете всю эту работу, а потом просто осознаете, что ошиблись, потому что ваш XHTML какой-то не такой, как должен быть.

В чем же тогда смысл? Если твой XHTML просто воспринимается браузерами как HTML, значит, это и есть обычный HTML!

Это все, конечно, здорово, но я продолжаю настаивать на том, что людям совсем не обязательно об этом заботиться. Я достаточно хорош для них. Многим людям совсем не нужен XML.

Допустим, ты прав и за XHTML будущее. Отлично. Но как ты сам и говоришь, XHTML — это язык будущего, поэтому мои пользователи могут просто дождаться этого момента и только затем перейти к работе с XHTML.

Я хочу сказать: «Насильно мил не будешь».

XHTML

Эй, это тоже не упустили из виду. Разработчики XHTML *знали*, что не все браузеры будут поддерживать этот язык, поэтому они сделали его совместимым и с более ранними версиями браузеров. Иными словами, сегодня вы можете преобразовывать все свои XHTML-документы, и они будут успешно работать даже в старых браузерах.

Но ведь ситуация меняется; с каждым днем XHTML поддерживается все шире и шире. Итак, мой вам совет: развивайтесь и меняйте свои документы. Это несложно, а с появлением новых устройств и браузеров вы будете сразу готовы в них работать, не прилагая никаких усилий.

Ты не можешь себе даже представить, насколько широко XHTML будет применяться в будущем. Научившись писать на XHTML сегодня, вы будете ко всему готовы завтра.

Что ты хочешь сказать? «Старого учить — только портить»?

HTML или XHTML? Выбор за вами



Имеют ли для вас значение какие-либо преимущества XHTML? Преобразуете ли вы существующий XML в HTML для Сети? Работаете ли вы над страницами, которые должны хорошо отображаться на мобильных устройствах? Или, может быть, некоторые из технологий XHTML станут необходимы вам в самом ближайшем будущем? О, а может, вы просто хотите идти в ногу со временем? Хорошо, у нас есть для вас хорошие новости: можете начать использовать XHTML уже сегодня. Это будет вам стоить всего лишь добавления нового DOCTYPE и небольших изменений в нескольких тегах. Сегодня не все браузеры одобрят ваш переход к XHTML, но рано или поздно им придется это сделать. Однако и до этого момента ваши страницы будут замечательно отображаться в этих браузерах, потому что они будут восприниматься как HTML (однако не забывайте о том, о чем мы вас уже предупреждали). Итак, доброго пути и наслаждайтесь вашим путешествием в мир XHTML.

Ничего из вышеупомянутого не имеет для вас особого значения? Ваша основная забота — создание отличных веб-страниц? Для вас у нас также есть новости: вы легко можете продолжать пользоваться строгим HTML 4.01 и получать все преимущества от использования языка, который является выбором большинства браузеров на сегодняшний день. Но если все-таки когда-нибудь вам понадобится преобразовать свои страницы в XHTML, то можете воспользоваться программой перехода к нему за три шага, которая приведена в этой главе чуть выше.

Итак, неважно, каков будет ваш выбор, — мы желаем вам успехов в любом случае. Хотя надо сказать, что различия между HTML и XHTML действительно минимальны, так почему же не перейти к XHTML уже сейчас? Мы уже писали на XHTML 1.0 и будем продолжать делать это до конца книги. Но если по какой-либо причине вам нужно будет писать на HTML 4.01, в этом нет ничего плохого. Поскольку эти языки почти одинаковы, у вас не будет никаких проблем при дальнейшем чтении данной книги. Просто убедитесь, что DOCTYPE соответствует той версии, которую вы на самом деле используете.



Решение упражнений

Вам нужно взять страницу с дневником Тони (помните его из главы 3?) и преобразовать ее код в XHTML. Самую последнюю версию дневника вы найдете в папке `chapter7/journal` в файле `journal.html`, написанном на строгой версии HTML 4.01.

Рассмотрим решение:

Изменяем DOCTYPE на XHTML 1.0 Strict.

Добавьте три атрибута в открывающий тег <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="ru" xml:lang="ru">
   <head>
   <meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />
      <title>На скутере по США</title>
   .
   .
   .
   </body>
</html>
```

Не забудьте, что все пустые элементы должны заканчиваться символами «/».

Тег <meta> также должен заканчиваться символом «/».

Начнем работать над дизайном



Раньше говорилось, что в книге будет материал про CSS. До сих пор мы изучали XHTML, применяемый для создания структуры веб-страниц. Но, как видите, манера браузеров оформлять страницы оставляет желать лучшего. Конечно же, можно было позвать полицию моды, но нам это не нужно. Мы отлично справимся с дизайном страниц с помощью CSS, часто даже не меняя XHTML-код. Действительно ли это так легко? Ну, придется выучить новый язык; в конце концов, Webville — это двуязычный город. После прочтения этой главы, посвященной CSS, вы будете в состоянии поддерживать разговор, находясь на *любой* из сторон Мейнстрит.

Помните волшебника из страны Оз? В этой части книги все черно-белое станет цветным.

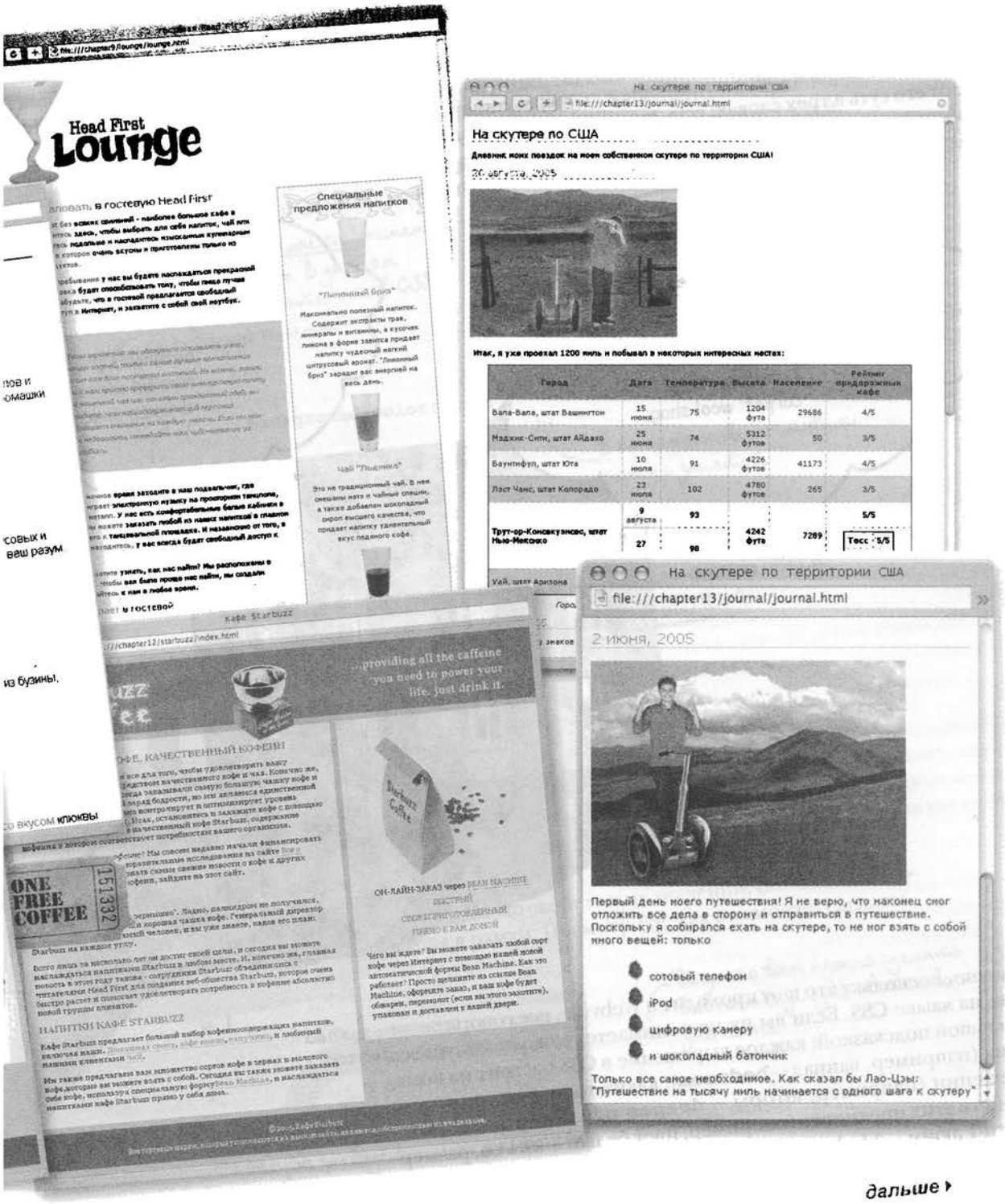
Вы больше не в Канзасе

Вы уже потратили немало сил на изучение разметки, структуры, синтаксиса, вложенности, соответствия стандартам, и сейчас пришло время *развлечься*, создавая дизайн веб-страниц. Однако не беспокойтесь, все те усилия, которые вы затратили на изучение XHTML, не напрасны. На самом деле вы поймете, что хорошее знание XHTML – необходимое условие для изучения (и использования) CSS, а именно изучением каскадных таблиц стилей мы и займемся в следующих нескольких главах.

На этих двух страницах мы разрекламировали дизайн, с которым вы будете иметь дело в оставшейся части книги. Не правда ли, он очень отличается от дизайна тех страниц, которые вы уже успели создать? Итак, что же нужно для того, чтобы приступить к делу? Конечно же, выучить язык CSS.

Давайте начнем...

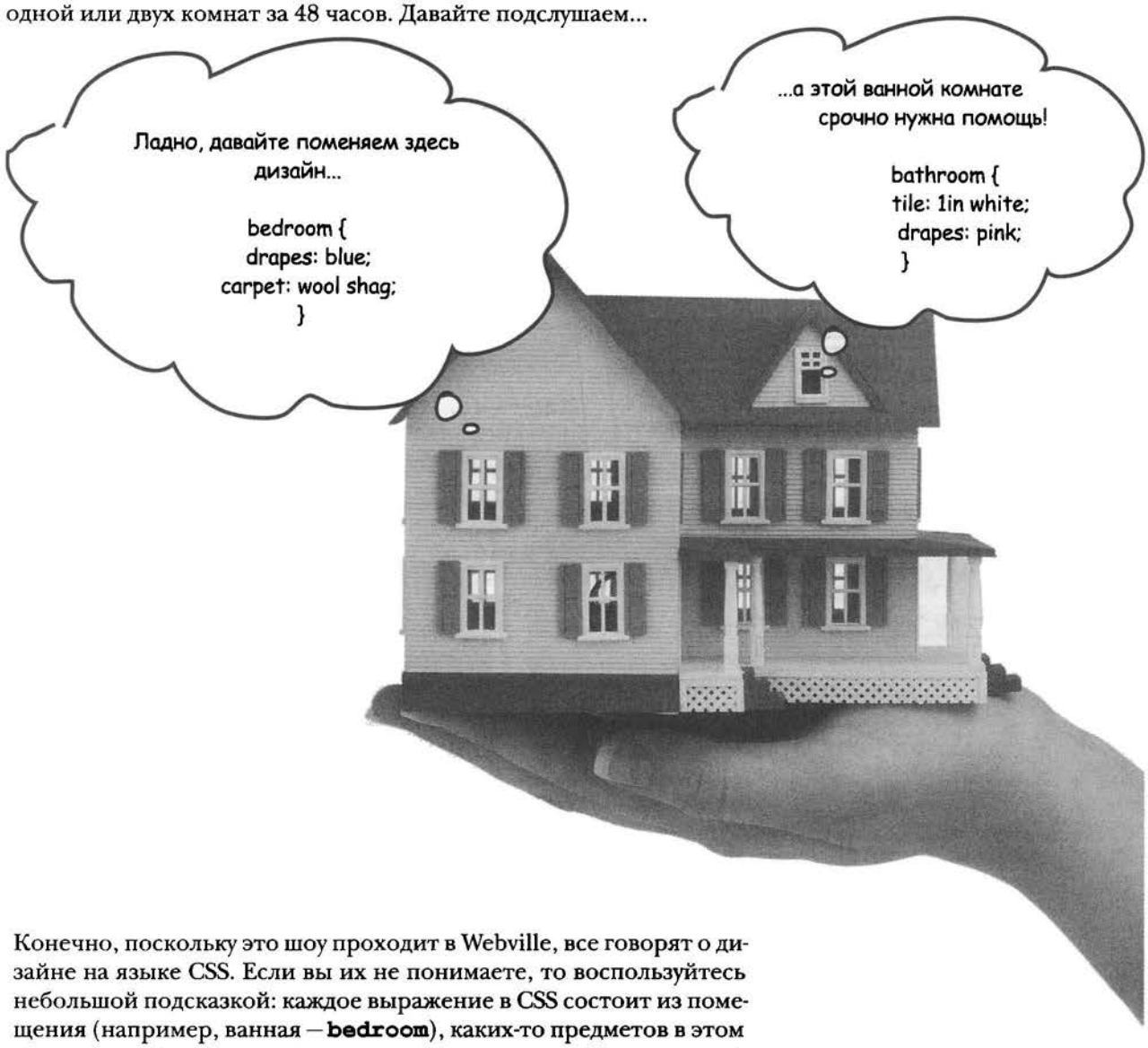



Head First Lounge
Специальные предложения напитков
"Лимонный бриз"
Чай "Пьяница"
На скутере по территории США
file:///chapter13/journal/journal.html
На скутере по США
Дневник моих поездок на моем собственном скутере по территории США!
Итак, я уже проехал 1200 миль и побывал в некоторых интересных местах:
Город
Дата
Температура
Высота
Население
Рейтинг придорожных кафе
Вапа-Вапа, штат Вашингтон
15 июня
75
1204 фута
29686
4/5
Маджик-Сити, штат Айдахо
25 июня
74
5312 футов
50
3/5
Баунтифул, штат Юта
10 июля
91
4226 футов
41173
4/5
Ласт Чанс, штат Колорадо
23 июля
102
4780 футов
265
3/5
Трут-ор-Консекуэнсес, штат Нью-Мексико
9 августа
93
4242 фута
7289
5/5
Кафе Starbuzz
file:///chapter12/starbuzz/index.html
...providing all the caffeine you need to power your life, just drink it.
ONE FREE COFFEE
Starbuzz Coffee
НАПИТКИ КАФЕ STARBUZZ
На скутере по территории США
file:///chapter13/journal/journal.html
2 июня, 2005
Первый день моего путешествия! Я не верю, что наконец смог отложить все дела в сторону и отправиться в путешествие. Поскольку я собирался ехать на скутере, то не мог взять с собой много вещей: только
сотовый телефон
iPod
цифровую камеру
и шоколадный батончик
Только все самое необходимое. Как сказал бы Лао-Цзы:
"Путешествие на тысячу миль начинается с одного шага к скутеру"

дальше ▸

Послушаем, что происходит в реалити-шоу «Квартира соседа» в Webville

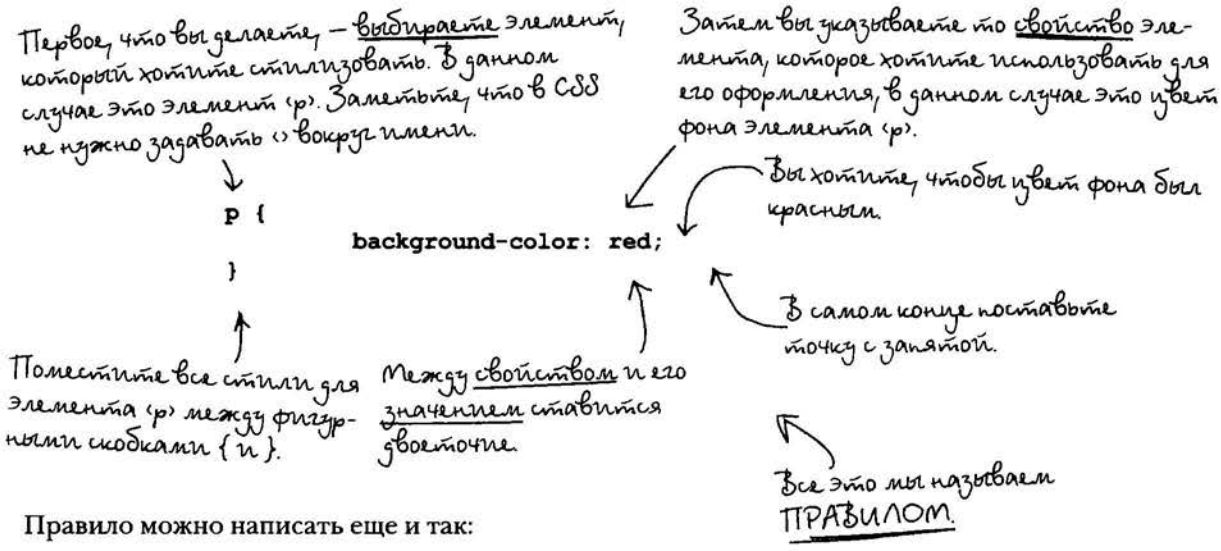
Вы не слышали ничего о последнем телевизионном реалити-шоу? Вот его суть в трех словах: есть два соседа, два дома и $1000. Соседи обмениваются домами и, имея $1000, полностью меняют дизайн одной или двух комнат за 48 часов. Давайте подслушаем...



Конечно, поскольку это шоу проходит в Webville, все говорят о дизайне на языке CSS. Если вы их не понимаете, то воспользуйтесь небольшой подсказкой: каждое выражение в CSS состоит из помещения (например, ванная — **bedroom**), каких-то предметов в этом помещении (например, шторы — **drapes** или ковер — **carpet**) и свойств этих предметов (например, цвет белый — **white**, размер плитки 1 дюйм — **1in**).

Использование CSS вместе с XHTML

Мы, конечно, не сомневаемся, что у CSS блестящее будущее в области дизайна комнат и домов, но давайте вернемся к XHTML. У него нет комнат, но есть элементы, и именно они будут играть роль помещений, которые мы начнем оформлять. Хотите покрасить стены вашего элемента **<p>** в красный цвет? Не проблема; правда, у абзацев нет стен, поэтому придется вместо них довольствоваться свойством, определяющим цвет фона, – **background-color**. Вот как это сделать:



Правило можно написать еще и так:

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

Все, что мы здесь сделали, – убрали разрывы строки. Можете форматировать свой CSS-код так, как хотите. Если правило достаточно длинное, то обычно в него добавляются разрывы строки и отступы для удобства чтения (для вас же самих).

Хотите добавить больше стилей?

Вы можете добавить в каждое CSS-правило столько свойств и их значений, сколько пожелаете. Скажем, вы хотите заключить абзацы в рамки. Посмотрите, как это можно сделать:



Часто Задаваемые Вопросы

В: Все элементы <p> всегда имеют одинаковый стиль? Или, скажем, я могу задать разные цвета тексту различных абзацев?

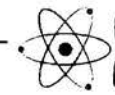
О: Правила стиля CSS, которые мы использовали до сих пор, определяли стиль сразу для *всех* абзацев, но CSS очень многогранен: он может быть использован для задания стилей различными способами, для множества различных элементов и даже для подмножеств элементов. Далее вы еще узнаете, как двум абзацам задать разные цвета.

В: Как мне узнать, какие свойства я могу определить для элемента?

О: Существует *множество* свойств, которые можно задавать для элементов, во всяком случае намного больше, чем вы захотите запомнить и использовать. В следующих нескольких главах вы познакомитесь с наиболее распространенными свойствами. Вам, вероятно, также понадобится хороший справочник по CSS. Сейчас есть множество онлайн-справочников. Кроме того, можете воспользоваться замечательной маленькой книгой издательства O'Reilly's *CSS Pocket Reference*.

В: Напомните мне, почему все эти стили определяются с использованием отдельного языка, а не в самом XHTML. С учетом того, что все элементы написаны на XHTML, разве не было бы легче и стили писать на нем же?

О: В нескольких следующих главах вы узнаете определенные преимущества использования CSS. Однако можем уже сейчас дать краткий ответ на ваш вопрос: CSS больше предназначен для задания стилей, чем XHTML. Используя совсем небольшой кусок CSS-кода, вы можете значительно изменить стиль оформления вашей XHTML-страницы. Далее вы также поймете, что CSS намного лучше подходит для создания дизайна большого количества страниц. Чуть позже в данной главе вы увидите, как это работает.



МОЗГОВОЙ ШТУРМ

Допустим, у вас есть элемент `<em>` внутри абзаца. Как вы думаете, придется ли менять цвет фона для элемента `<em>` таким образом, чтобы он соответствовал цвету фона абзаца, если вы измените цвет фона абзаца?

Добавление CSS в Ваш XHTML

Отлично, вы уже немного познакомились с синтаксисом CSS. Теперь вы знаете, как выбрать элемент и затем написать для него правило стиля со свойствами и их значениями. Но вы все еще не знаете, как связать этот CSS-код с XHTML.

Для начала понадобится какой-нибудь XHTML-код, в который мы и будем добавлять CSS. На протяжении нескольких следующих глав мы посетим кафе Starbuzz и вспомним Тони с его дневником, а также сделаем их страницы более стильными. Как вы думаете, за какой сайт мы возьмемся в первую очередь? Конечно же, за гостевую Head First. Итак, ниже приведен XHTML-код для главной страницы гостевой. Как вы помните, мы внесли кое-какие изменения, чтобы документ соответствовал всем правилам строгого XHTML (вы же не ожидаете от нас меньшего?). Сейчас мы добавим несколько тегов стиля, и это будет самый простой способ оформления вашей страницы.

Нельзя сказать, что это наилучший способ. Чуть позже в данной главе мы рассмотрим другой способ.

```
<!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="ru" xml:lang="ru" >
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />
    <title>Гостевая Head First</title>

    <style type="text/css">

    </style>

  </head>
  <body>
    <h1>Добро пожаловать в новую и усовершенствованную гостевую Head First</h1>
    <p>
       <img src="images/drinks.gif" alt="Напитки" />
    </p>
    <p>
       Заходите к нам каждый вечер, чтобы попробовать освежающие
       <a href="beverages/elixir.html">напитки</a>,
       поболтать и, возможно,
       <em>станцевать разок-другой</em>.
       Всегда обеспечен беспроводной доступ
       (захватите с собой свой ноутбук).
    </p>
    <h2>Указатели</h2>
    <p>
       Вы найдете нас в центре Webville.
       Если вам нужна помощь, чтобы найти нас, используйте наши
       <a href="about/directions.html">указатели</a>.
       Присоединяйтесь к нам!
    </p>
  </body>
</html>
```

Здесь то, что нам нужно: элемент <style>.

Чтобы добавить CSS непосредственно в XHTML-код, добавьте открывающий и закрывающий теги элемента style внутрь элемента <head>.

И тип стиля "text/css".

Ваши правила стилей CSS будут добавлены прямо сюда.



Дальше >

Добавление стиля в гостевую

Теперь, когда у вас уже есть элемент **<style>** в XHTML-коде, вы немного стилизуете гостевую, чтобы понять, как писать на CSS. Возможно, с таким дизайном вы не победите ни на одном конкурсе дизайнеров, но ведь нужно с чего-то начинать.

Первое, что мы сделаем, — поменяем цвет (на такой, который бы хорошо сочетался с красными диванами в гостевой) текста в абзацах. Для этого будем использовать CSS-свойство **color**:

```
<!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="ru"
xml:lang="ru">
  <head>
    <meta http-equiv="Content-Type"
          content="text/html; charset=windows-1251" />
    <title>Гостевая Head First</title>
    <style type="text/css">

      p {
        color: maroon;
      }

    </style>
  </head>
  <body>
    <h1>Добро пожаловать в новую и усовершенствованную
        гостевую Head First</h1>
    <p>
      <img src="images/drinks.gif" alt="Напитки" />
    </p>
    <p>
      Заходите к нам каждый вечер, чтобы попробовать освежающие
      <a href="beverages/elixir.html">напитки</a>,
      поболтать и, возможно,
      <em>станцевать разок-другой</em>.
      Всегда обеспечен беспроводной доступ
      (захватите с собой свой ноутбук).
    </p>
    <h2>Указатели</h2>
    <p>
      Вы найдете нас в центре Webville.
      Если вам нужна помощь, чтобы найти нас, используйте наши
      <a href="about/directions.html">указатели</a>.
      Присоединяйтесь к нам!
    </p>
  </body>
</html>
```

Это правило стиля, которое будет определять цвет текста в абзацах.

Мы выбираем элемент <p>, для которого и будет применяться этот стиль.

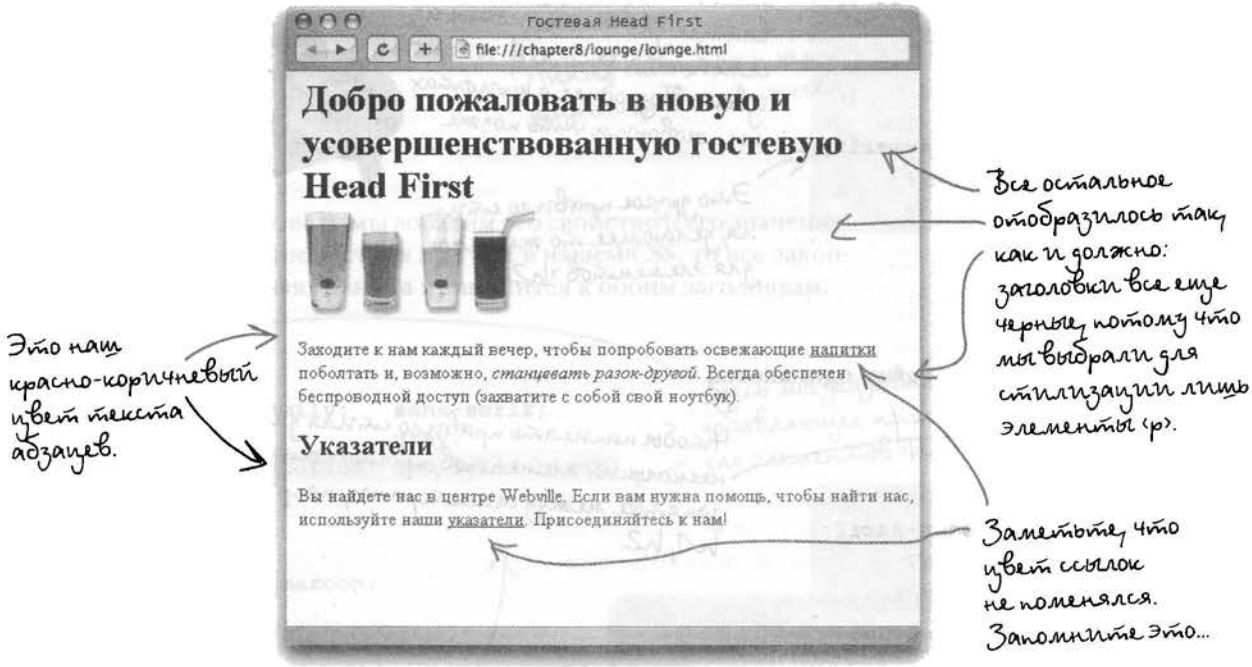
Свойство для изменения цвета шрифта называется color (возможно, вы думали, что оно будет называться font-color или text-color, но это не так).

Мы устанавливаем красивый насыщенный красно-коричневый цвет текста, что будет хорошо сочетаться с диванами в гостевой.

Элемент <p> выбирает все абзацы XHTML.

Тестирование стиля

Итак, продолжим. Внесите все изменения, описанные на предыдущих страницах, в файл `lounge.html` из папки `chapter8/lounge`, сохраните и обновите страницу в браузере. Вы увидите, что цвет текста абзацев поменялся на красно-коричневый.



МОЗГОВОЙ ШТУРМ

Что будет, если вы установите для элементов **<p>** красно-коричневый цвет фона, а не текста? Каким образом изменится способ отображения ваших страниц в браузерах?

Стилизация заголовков

Теперь зададим стиль для наших заголовков. Как насчет того, чтобы немного поменять шрифт? Изменим для заголовков и тип шрифта, и цвет:




```
h1 {
     font-family: sans-serif;
     color:       gray;
}

h2 {
     font-family: sans-serif;
     color:       gray;
}

p {
     color: maroon;
}
```

Это правило стиля для элементов <h1>, устанавливающее для них шрифты из семейства sans-serif и серый цвет. Подробнее о шрифтах мы поговорим чуть позже.

Это другое правило стиля, делающее то же самое для элементов <h2>.

Поскольку эти правила *абсолютно* одинаковы, мы можем их объединить:

```
h1, h2 {
     font-family: sans-serif;
     color:       gray;
}

p {
     color: maroon;
}
```

Чтобы написать правило стиля для нескольких элементов, просто поставьте запятые между селекторами, например h1, h2.

Тест

Добавьте этот новый CSS-код в файл `lounge.html` и обновите страницу в браузере. Вы увидите, что благодаря одному правилу выделили и заголовки **<h1>**, и подзаголовки **<h2>**.

Оба типа заголовков на странице теперь имеют одинаковый стиль: шрифт sans-serif и серый цвет.



Подчеркнем заголовок с приветствием

Улучшим немного внешний вид заголовка «Добро пожаловать в гостевую...». Как насчет того, чтобы подчеркнуть его? Это позволит визуально отделить главный заголовок от всего остального и придать ему изысканный вид. Вот свойство, которое мы будем использовать для этого:

```
border-bottom: 1px solid black;
```

Это свойство контролирует внешний вид нижней границы под элементом.

Линия сплошная и черная, а ее ширина равна 1 пикселу.

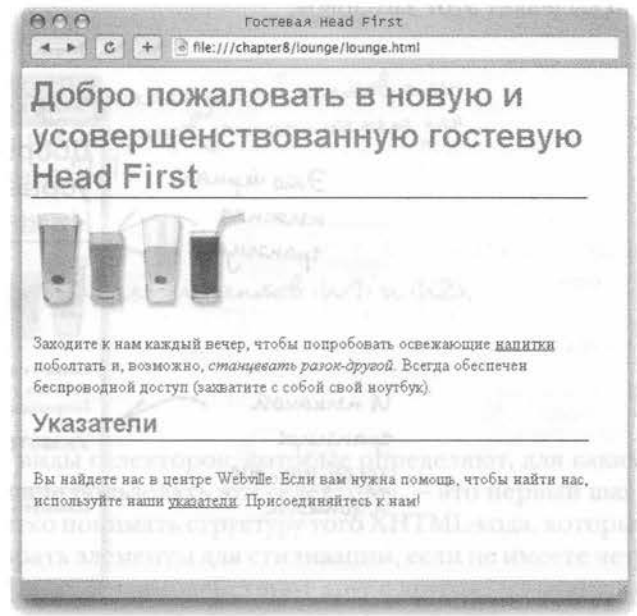
Проблема в том, что, если мы добавим это свойство и его значение в объединенное правило стиля **h1, h2** в нашем CSS, то все закончится тем, что нижняя граница применится к обоим заголовкам:

```
h1, h2 {
     font-family:   sans-serif;
     color:         gray;
     border-bottom: 1px solid black;
}

p {
     color: maroon;
}
```

Здесь мы указываем свойство, добавляющее нижнюю границу для элементов <h1> и <h2>.

Если мы сделаем так...

...то обоим заголовкам будет добавлена нижняя граница. Это не совсем то, что нам нужно.



Итак, как же задать нижнюю границу только для элемента **<h1>**, не затрагивая элемент **<h2>**? Неужели нам придется снова разделить элементы? Переверните страницу, чтобы узнать это...

Существует особая технология: указание второго правила, только для <h1>

Нам не придется разделять правило стиля **h1, h2** на части, нужно будет просто добавить еще одно правило – только для **h1**, а затем установить для него стиль подчеркивания.

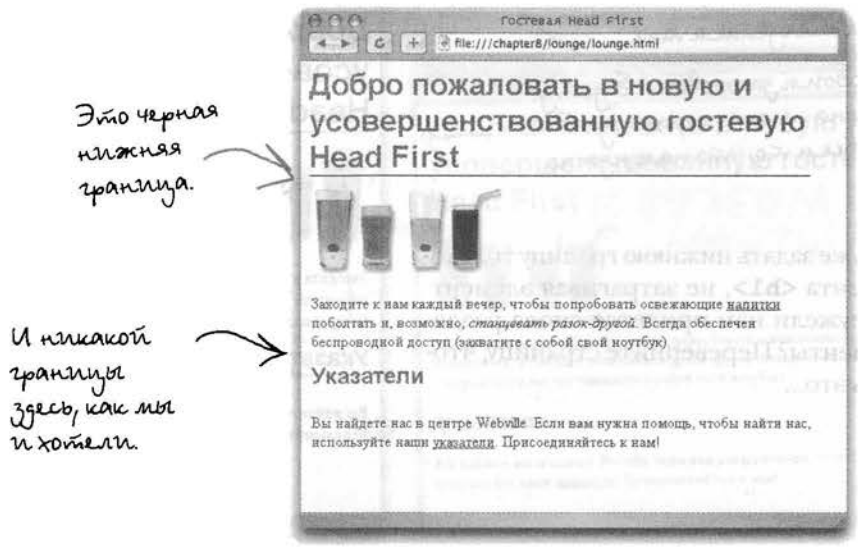
```
h1, h2 {
     font-family:    sans-serif;
     color:          gray;
}

h1 {
     border-bottom: 1px solid black;
}

p {
     color: maroon;
}
```

В первой части ничего не изменилось. Мы все еще используем общее правило для указания семейства шрифтов и цвета элементов <h1> и <h2>.

Но вот мы задаем второе правило, которое добавляет свойство только для элемента <h1>: определение нижней границы.

Снова протестируем

Измените свой CSS-код и обновите страницу. Вы увидите, что новое правило устанавливает черную нижнюю границу для основного заголовка, что выглядит очень изящно и действительно выделяет этот заголовок.



Часто Задаваемые Вопросы

В: Как же все работает, когда для одного элемента задано больше одного правила?

О: Для одного элемента вы можете задавать столько правил стиля, сколько хотите. Каждое новое правило добавляет новую информацию о стиле к тому правилу, которое находится перед ним. Таким образом, вы объединяете все общие правила стилей, как мы сделали для <h1> и <h2>, а затем указываете каждое индивидуальное свойство отдельного элемента в другом правиле, как мы это сделали, подчеркнув основной заголовок.

В: Каковы преимущества такого подхода? Не лучше ли создавать свое правило стиля для каждого отдельного элемента, чтобы наверняка знать, какой он имеет стиль?

О: Не совсем. Если у вас объединены одинаковые стили и какой-то стиль нужно поменять, вам придется изменить только одно правило. А если вы задаете правило для каждого элемента, то вам придется менять множество правил, что может послужить причиной множества ошибок.

В: Почему мы используем нижнюю границу для подчеркивания? Разве не предусмотрен специальный стиль для подчеркивания?

О: Хороший вопрос. Такой стиль действительно предусмотрен, и в данной ситуации можно использовать его. Однако два этих стиля немного по-разному отображаются на странице: если вы задаете нижнюю границу, то линия растягивается на всю ширину страницы, а подчеркивание отображается только под самим текстом. Свойство, применяемое для подчеркивания текста, называется text-decoration, и, чтобы подчеркнуть текст снизу, нужно установить значение underline. Попробуйте применить его, и поймете разницу.

Как же на самом деле все работает

Вы уже видели, как выбрать элемент, чтобы добавить ему стиль:

Мы называем это селектором.

```
h1 {
      color: gray;
}
```

Стиль применяется для элементов, описанных селектором; в данном примере для элемента <h1>.

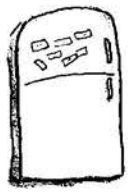
Вы также знаете, как выбрать несколько элементов:

Другой селектор. Стиль применяется для элементов <h1> и <h2>.

```
h1, h2 {
      color: gray;
}
```

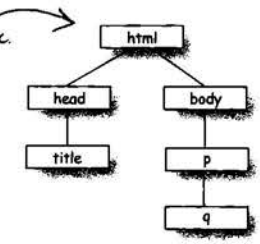
Вы увидите, что CSS позволяет задавать любые виды селекторов, которые определяют, для каких элементов применяется стиль. Понимание того, как использовать эти селекторы, — это первый шаг к освоению CSS. Чтобы овладеть этим, вам нужно четко понимать структуру того XHTML-кода, который вы стилизуете. В конце концов, как вы можете отобрать элементы для стилизации, если не имеете четкого представления об XHTML-элементах и о том, как они взаимодействуют друг с другом?

Итак, составим четкую картину гостевой XHTML в своей голове, а затем вновь вернемся к селекторам.

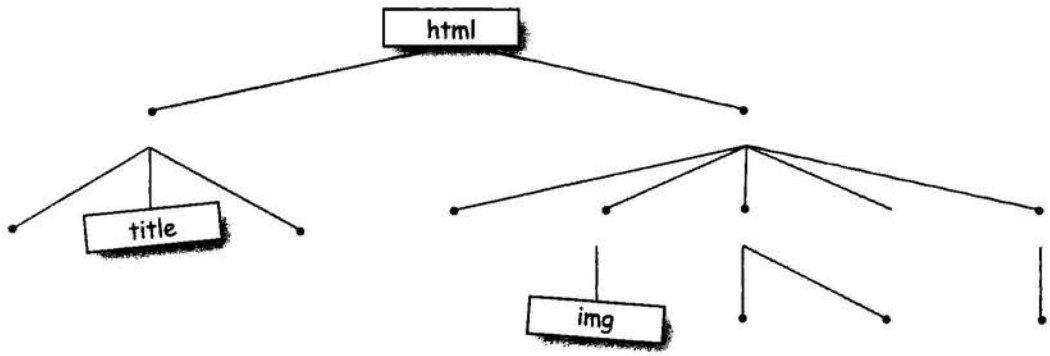


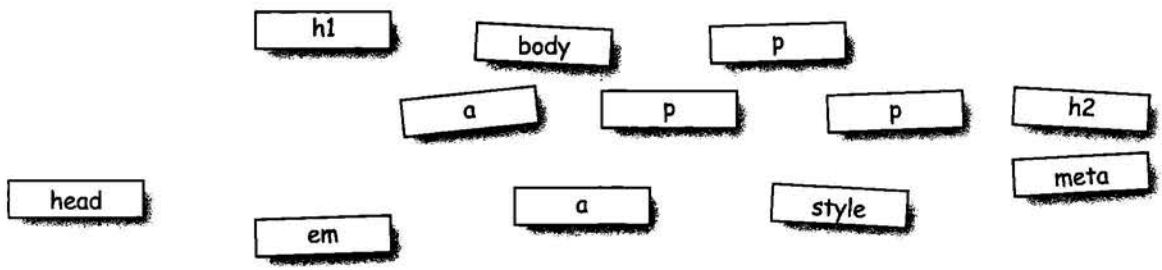
Магниты для разметки

Вот так.



Помните, как мы рисовали диаграмму для HTML-элементов в главе 3? Сейчас вам придется сделать то же самое для главной страницы гостевой. Ниже вы найдете все магниты-элементы, которые понадобятся для составления диаграммы. Используя XHTML-код для гостевой (вы найдете его справа), дополните дерево, нарисованное ниже. Мы уже начали выполнять задание, чтобы вам было легче. Решение упражнения вы найдете в конце главы.





XHTML для гостевой Head First.

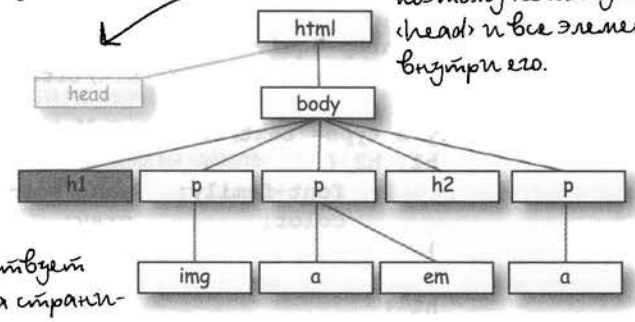
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />
    <title>Гостевая Head First</title>

    <style type="text/css">
       h1, h2 {
             font-family:   sans-serif;
             color:         gray;
       }

       h1 {
             border-bottom: 1px solid black;
       }

       p {
             color: maroon;
       }
    </style>

  </head>
  <body>
    <h1>Добро пожаловать в новую и усовершенствованную гостевую Head First</h1>
    <p>
       <img src="images/drinks.gif" alt="Напитки" />
    </p>
    <p>
       Заходите к нам каждый вечер, чтобы попробовать освежающие
       <a href="beverages/elixir.html">напитки</a>,
       поболтать и, возможно,
       <em>станцевать разок-другой</em>.
       Всегда обеспечен беспроводной доступ
       (захватите с собой свой ноутбук).
    </p>
    <h2>Указатели</h2>
    <p>
       Вы найдете нас в центре Webville.
       Если вам нужна помощь, чтобы найти нас, используйте наши
       <a href="about/directions.html">Указатели</a>.
       Присоединяйтесь к нам!
    </p>
  </body>
</html>
```

Визуальное представление селекторов

Рассмотрим несколько селекторов и определим, где они будут отображаться в дереве, которое мы только что нарисовали. Вот как селектор h1 отобразится на диаграмме:

```
h1 {
    font-family: sans-serif;
}
```

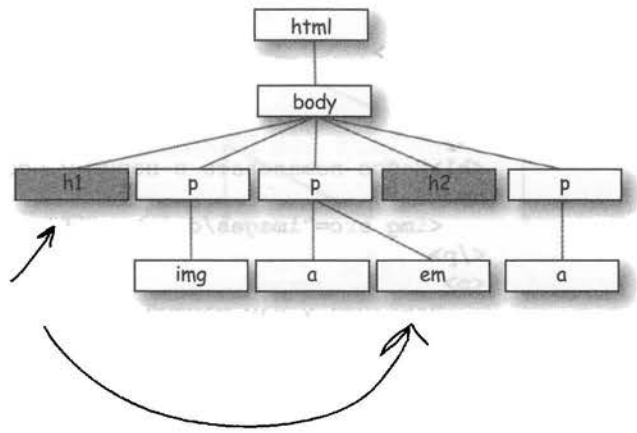
Мы можем добавлять стили только для тех элементов, что находятся внутри body, поэтому не показываем <head> и все элементы внутри его.



Этот селектор соответствует любому элементу <h1> на странице, но у нас он только один.

Здесь посмотрим, как выглядит селектор h1, h2:

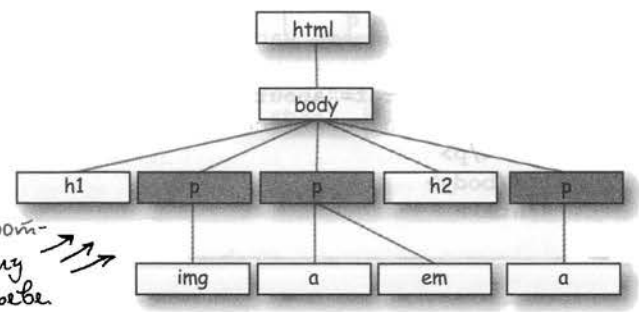
```
h1, h2 {
    font-family: sans-serif;
}
```



Теперь селектор соответствует обоим элементам <h1> и <h2>.

Если мы используем селектор p, то вот как он будет выглядеть:

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

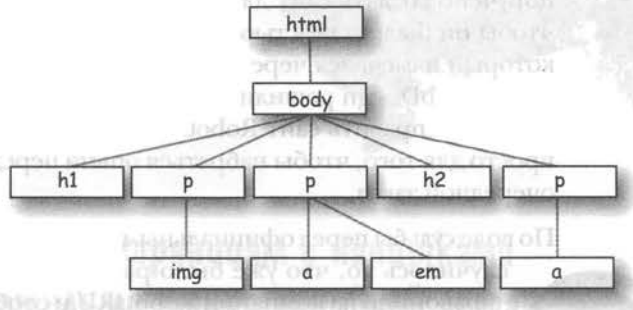


Этот селектор соответствует любому элементу <p> в дереве.

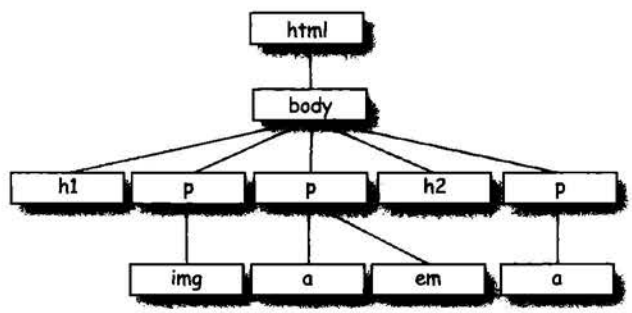
Возьми в руку карандаш

Выделите каким-нибудь цветом элементы, которые **выбраны** следующими селекторами:

```
p, h2 {
    font-family: sans-serif;
}
```




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



История противостояния Грубой Силы и Стиля

Когда мы в последний раз встречались с компанией RadWebDesign в главе 4, она как раз провалилась на презентации и потеряла заказчика RobotsRU. А компании CorrectWebDesign было поручено создать сайт для RobotsRU и позаботиться о том, чтобы он был полностью действующим к моменту выпуска, который намечался через месяц. Как вы помните, сотрудники RadWebDesign решили более тщательно изучить XHTML и CSS и исправить сайт RobotsRU, используя новые знания, просто для того, чтобы набраться опыта перед тем, как взять очередной заказ.

Пятиминутная головоломка



По воле судьбы перед официальным запуском сайта RobotsRU случилось то, что уже было ранее: в CorrectWebDesign позвонили из компании RobotsRU и сообщили ужасную новость: «Мы меняем наш фирменный стиль, и поэтому нужно, чтобы вы изменили цвета отдельных фрагментов текста, фона и шрифты на сайте». На этой стадии в сайте насчитывалось около сотни страниц, поэтому в CorrectWebDesign ответили, что им необходимо несколько дней, чтобы переделать весь сайт. «Но у нас нет этих нескольких дней!» — сказал генеральный директор, после чего, доведенный до отчаяния, он позвонил в RadWebDesign и попросил о содействии. «Вы провалились на презентации в прошлом месяце, но сейчас мы нуждаемся в вашей помощи. Не могли бы вы помочь ребятам из CorrectWebDesign преобразовать сайт так, чтобы он соответствовал новым требованиям?» В RadWebDesign сказали, что они могут сделать даже больше. На самом деле они могли бы всего через час представить им сайт в нужном виде.

Как же специалисты компании RadWebDesign после того позора на презентации превратились в профессиональных веб-разработчиков? Как им удалось изменить дизайн сотни веб-страниц так быстро?



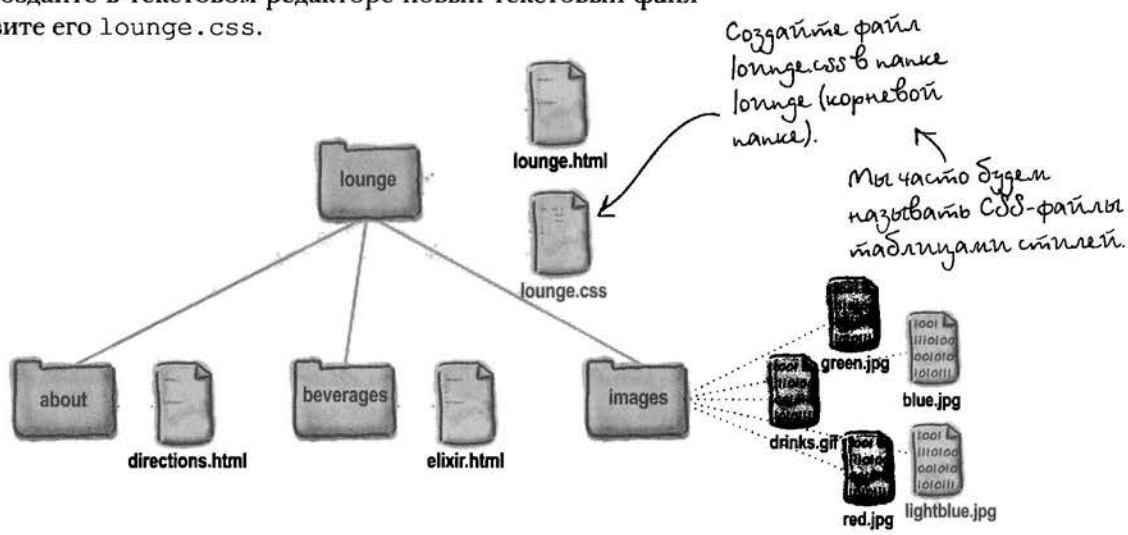
Присвоение страницам с напитками и указателями стиля основной страницы гостевой

Это просто здорово, что нам удалось оформить страницу `lounge.html`, но как насчет `elixir.html` и `directions.html`? Они должны быть оформлены в том же стиле, что и главная страница. Достаточно просто скопировать элемент `style` вместе со всеми правилами в каждый файл, не так ли? Не совсем так. Если вы так сделаете, то при необходимости поменять стиль оформления всего сайта вам придется вносить изменения в *каждый файл*. К счастью, существует способ попроще.

1. Возьмите правила из `lounge.html` и поместите их в файл `lounge.css`.
2. Создайте *внешнюю ссылку* на этот файл из файла `lounge.html`.
3. Создайте такие же внешние ссылки из файлов `elixir.html` и `directions.html`.
4. Хорошенько протестируйте все три файла.

Создание файла lounge.css

Итак, вам нужно создать файл `lounge.css`, который будет содержать правила стиля для всех страниц гостевой Head First. Для этого создайте в текстовом редакторе новый текстовый файл и назовите его `lounge.css`.



Теперь в файле `lounge.css` наберите (или скопируйте из файла `lounge.html`) CSS-правила. Сделав это, удалите правила стиля из файла `lounge.html`.

Обратите внимание, что теги **`<style>`** и **`</style>`** копировать не нужно, потому что файл `lounge.css` содержит только CSS-код и не содержит XHTML.

```
h1, h2 {
   font-family: sans-serif;
   color: gray;
}

h1 {
   border-bottom: 1px solid black;
}

p {
   color: maroon;
}
```

Ваш файл lounge.css должен выглядеть так. Помните: никаких тегов <style>!

Создание ссылки из lounge.html на внешний CSS-файл

Теперь нужно найти способ сказать браузеру, что он должен оформить страницу с помощью внешней таблицы стилей. Можно сделать это, используя XHTML-элемент **<link>**. Рассмотрим, как этот элемент добавляется в XHTML-код:

```
<!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="ru" xml:lang="ru">
  <head>
    <meta http-equiv="Content-Type"
          content="text/html; charset=windows-1251" />
    <title>Гостевая Head First</title>
    <link type="text/css" rel="stylesheet" href="lounge.css" />
    <style type="text/css">
    </style>

  </head>
  <body>
    <h1>Добро пожаловать в новую и усовершенствованную гостевую Head First</h1>
    <p>
       <img src="drinks.gif" alt="Напитки" />
    </p>
    .
    .
    .
    </p>
  </body>
</html>
```

Это XHTML-код, который позволяет сослаться на внешнюю таблицу стилей.

Вам больше не нужен элемент <style> – просто удалите его.

Остальной XHTML-код остается без изменений.



XHTML крупным планом

Давайте подробнее рассмотрим элемент **<link>**, так как раньше мы с ним не встречались:

Используйте элемент link, чтобы «сослаться» на внешнюю информацию.

Тип этой информации – text/css. Другими словами, это таблица стилей CSS.

Месторасположение таблицы стилей указано в атрибуте href (в данном примере мы используем относительную ссылку, но на ее месте может быть и URL-адрес).

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

Атрибут rel устанавливает взаимосвязь между XHTML-файлом и тем, на что вы ссылаетесь. Вы ссылаетесь на таблицу стилей, поэтому используйте значение stylesheet.

<link /> – это пустой элемент.

Создание ссылок на внешние таблицы стилей из файлов elixir.html и directions.html

Сейчас вы создадите ссылки с файлов `elixir.html` и `directions.html` точно так же, как вы делали это для файла `lounge.html`. Единственное, что вам нужно помнить, – файл `elixir.html` находится в папке **beverages**, а `directions.html` – в папке **about**, поэтому в них обоих нужно использовать относительный путь `../lounge.css`.

Итак, все, что вам нужно сделать, – добавить элемент `<link>` в оба файла:

```
<!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="ru" xml:lang="ru">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />
    <title>Напитки гостевой Head First</title>
    <link type="text/css" rel="stylesheet" href="../lounge.css" />
  </head>
  <body>
  .
  .
  .
  </body>
</html>
```

Это файл elixir.html. Просто добавьте элемент <link>.

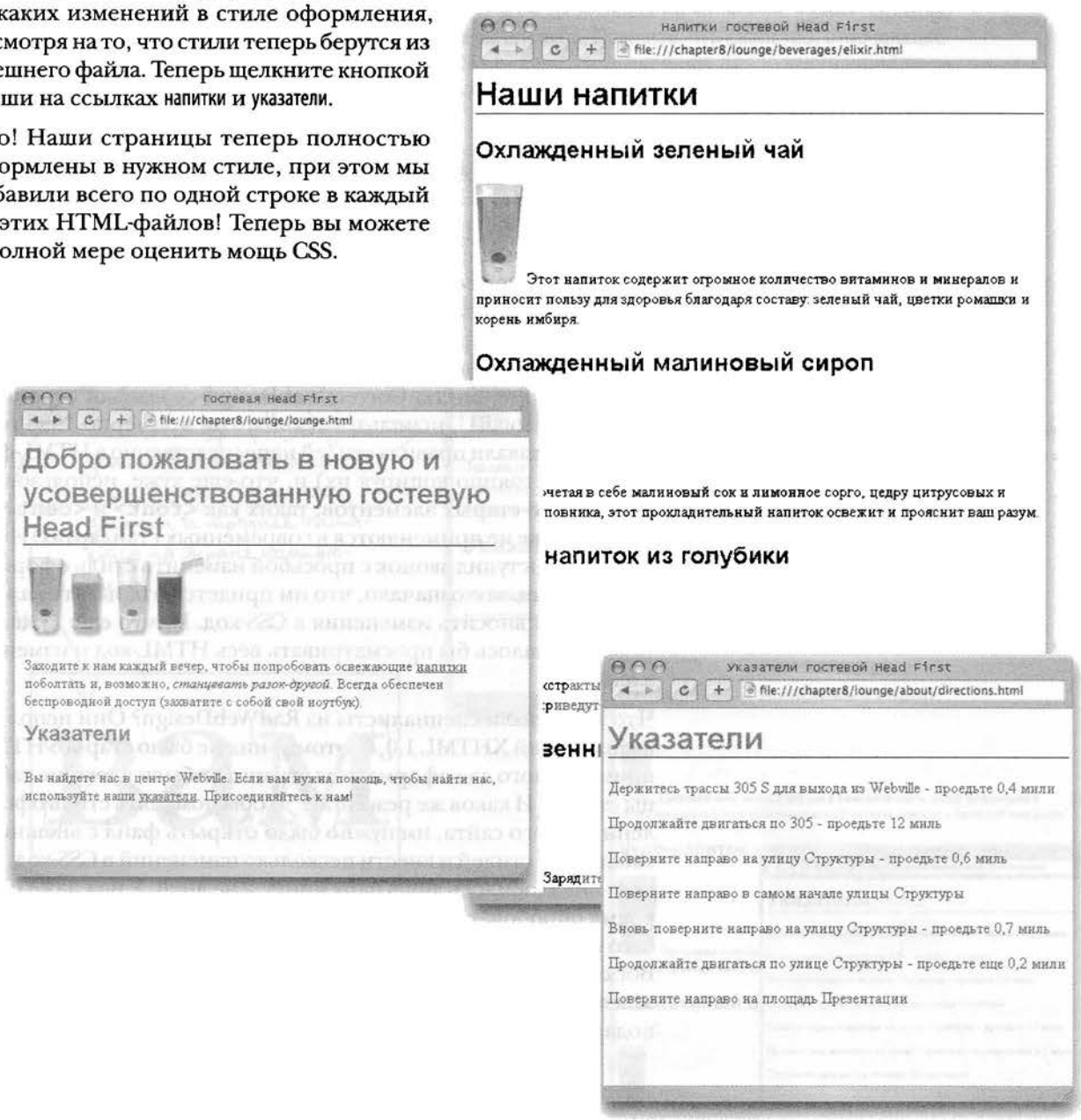
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
                      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />
    <title>Указатели гостевой Head First Lo</title>
    <link type="text/css" rel="stylesheet" href="../lounge.css" />
  </head>
  <body>
  .
  .
  .
  </body>
</html>
```

То же самое для файла directions.html. Добавьте элемент <link> здесь.

Тестирование всего сайта

Сохраните каждый файл и откройте страницу `lounge.html` в браузере. Вы не заметите никаких изменений в стиле оформления, несмотря на то, что стили теперь берутся из внешнего файла. Теперь щелкните кнопкой мыши на ссылках напитки и указатели.

Ого! Наши страницы теперь полностью оформлены в нужном стиле, при этом мы добавили всего по одной строке в каждый из этих HTML-файлов! Теперь вы можете в полной мере оценить мощь CSS.





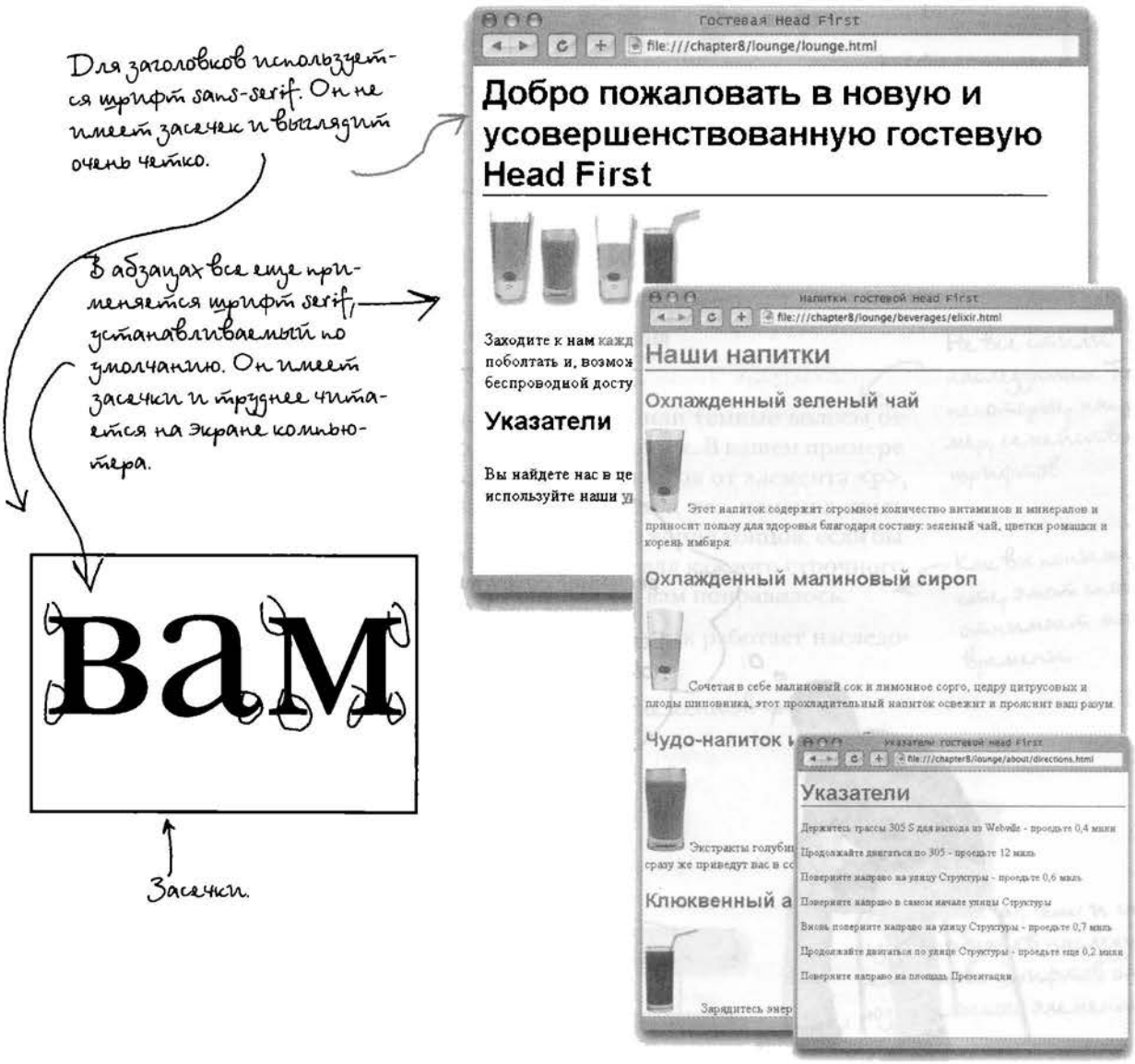
История противостояния Грубой Силы и Стиля

Итак, как же специалистам из компании RadWebDesign удалось стать профессиональными веб-разработчиками? Или все-таки стоит сначала задаться вопросом, как на этот раз «такая правильная» фирма CorrectWebDesign не смогла справиться с поставленной задачей? Суть проблемы заключается в том, что специалисты CorrectWebDesign создавали страницы для RobotsRU, используя технологии примерно 1998 года. Они задавали правила стилей непосредственно в HTML-файлах (постоянно копируя их) и, что еще хуже, использовали множество старых элементов, таких как **<font>** и ** **, которые уже не применяются в современных стандартах. Итак, когда им поступил звонок с просьбой изменить стиль оформления страниц, это означало, что им придется открывать *каждую* страницу и вносить изменения в CSS-код. И, что еще хуже, им также пришлось бы просматривать весь HTML-код и изменять элементы.

Что же сделали специалисты из RadWebDesign? Они использовали строгий XHTML 1.0, поэтому у них не было старого HTML, применяемого для оформления страниц, а были внешние таблицы стилей. И каков же результат? Чтобы поменять стиль оформления целого сайта, им нужно было открыть файл с внешними таблицами стилей и внести несколько изменений в CSS-код, для чего понадобилось лишь пару минут, а не дней. У них даже было время опробовать несколько стилей оформления, в результате чего к моменту выпуска сайта было готово три варианта CSS-кода. Восхищенный генеральный директор RobotsRU не только обязался отдавать новые заказы фирме RadWebDesign, но и обещал подарить им первого робота, который сойдет с конвейера.

Возьми в руку карандаш

Теперь, когда у вас есть внешний файл со стилями (или таблица стилей), используйте его, чтобы поменять шрифт во всех абзацах на sans-serif. Таким образом, он будет совпадать со шрифтом заголовков. Помните, что свойство, отвечающее за стиль шрифта, называется **`font-family`**, а значение для нужного шрифта — **`sans-serif`**. Решение вы найдете на следующей странице.



Возьми в руку карандаш

Решение

```
h1, h2 {
       font-family:  sans-serif;
       color:        gray;
}

h1 {
       border-bottom: 1px solid black;
}

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

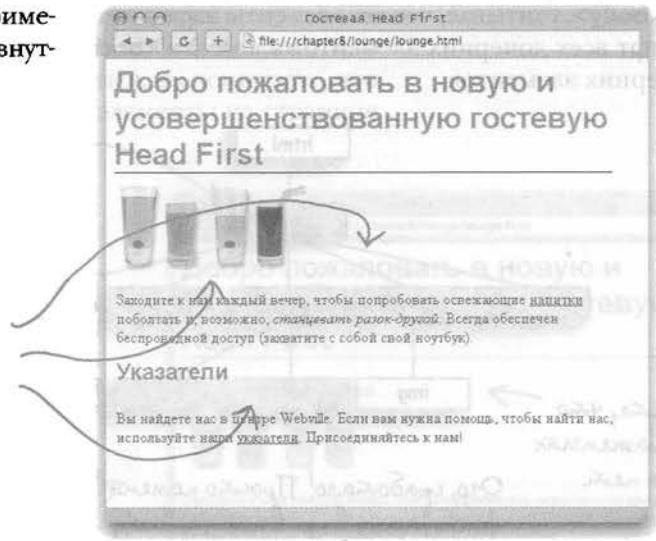
Просто добавьте в файле lounge.css свойство font-family в правило для абзацев.



Интересно, на самом ли деле это лучшее решение? Зачем мы указываем семейство шрифтов для КАЖДОГО элемента? Если, к примеру, кто-то добавит элемент <blockquote>, придется ли нам добавлять правило стиля и для него? Разве нельзя задать шрифт sans-serif сразу для всей страницы?

Пришло время поговорить о наследовании

Если заметили, когда вы добавили свойство **font-family** в селектор **p**, оно также применилось для элементов, которые находятся внутри **<p>**. Рассмотрим это подробнее.



Когда вы добавили свойство font-family в селектор p, семейство шрифтов для элементов <p> изменилось. Кроме того, оно поменялось для ссылок и выделенного текста.

Элементы, находящиеся внутри элемента <p>, наследуют заданное для него семейство шрифтов

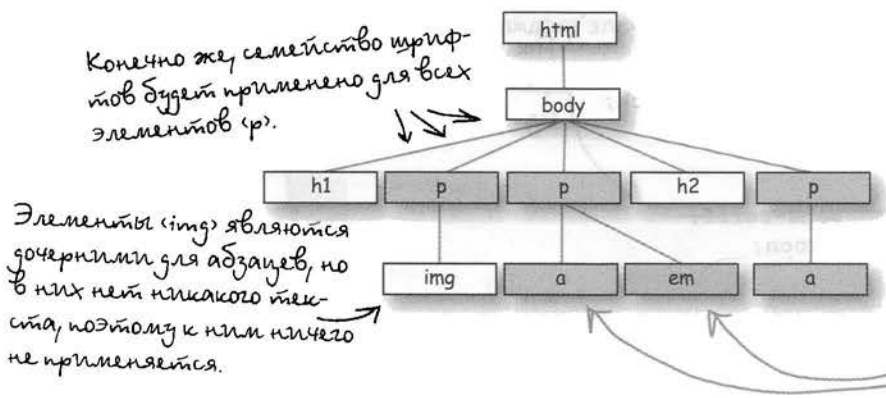
Не все стили наследуются. Только некоторые, например, семейство шрифтов.

Точно так же, как вы можете унаследовать голубые глаза или темные волосы от своих родителей, элементы могут унаследовать стиль от своих. В нашем примере элементы **<a>** и **** унаследовали стиль семейства шрифтов от элемента **<p>**, который является их родительским элементом. Это означает, что, изменив стиль абзацев, вы также поменяете стиль элементов внутри их. В конце концов, если бы это было не так, вам пришлось бы добавлять CSS-правила для каждого строчного элемента в каждом абзаце по всему сайту... что вряд ли бы вам понравилось.

Как вы понимаете, этот способ отнимает много времени.

Давайте взглянем на наше XHTML-дерево, чтобы понять, как работает наследование.

Если мы зададим семейство шрифтов для всех элементов <p>, то это повлияет на стиль этих элементов.



Конечно же, семейство шрифтов будет применено для всех элементов <p>.

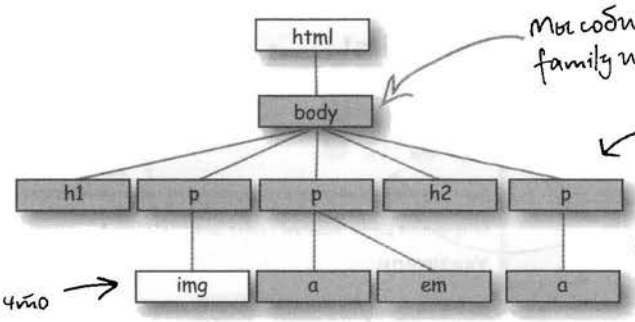
Элементы <img> являются дочерними для абзацев, но в них нет никакого текста, поэтому к ним ничего не применяется.

Элементы <a>, и <a> из двух абзацев наследуют семейство шрифтов от их родительского элемента <p>.

дальше →

Что будет, если мы переместим font вверх по дереву?

Что случится, если мы переместим свойство **font-family** вверх, до элемента **<body>**, учитывая, что все элементы наследуют его? Оно будет влиять на шрифт всех дочерних элементов элемента **<body>** и, в свою очередь, их дочерних элементов.



Мы собираемся переместить свойство font-family из абзацев и заголовков в правило body.

Теперь все эти элементы будут наследовать свойство font-family.

Как и их дочерние элементы

Помните, что в изображениях текста нет.

Ого, сработало! Просто поменяв свойство font-family в правиле для body, мы можем изменить шрифт для всего сайта.

Чего же вы ждете? Давайте протестируем

Откройте свой файл `lounge.css` и добавьте новое правило для элемента **<body>**. Затем удалите свойства **font-family** из правил для заголовков и абзацев, потому что они вам больше не понадобятся.

Вот что вам нужно сделать.

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

h1, h2 {
      font-family: sans-serif;
      color:       gray;
}

h1 {
      border-bottom: 1px solid black;
}

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

Сначала добавьте новое правило для элемента <body>. Затем добавьте свойство font-family со значением sans-serif.

После этого удалите свойство font-family из правила для h1, h2, а также из правила для p.

Протестируем новый CSS-код

Итак, продолжим. Как обычно, внесите соответствующие изменения в таблицу стилей `lounge.css`, сохраните и обновите страницу `lounge.html` в браузере. Не ждите каких-то перемен, так как использован тот же стиль. Оно просто поступает из другого правила. Но, надеемся, вы понимаете, что ваш CSS-код стал лучше, так как теперь все новые элементы на странице автоматически унаследуют шрифт sans-serif.

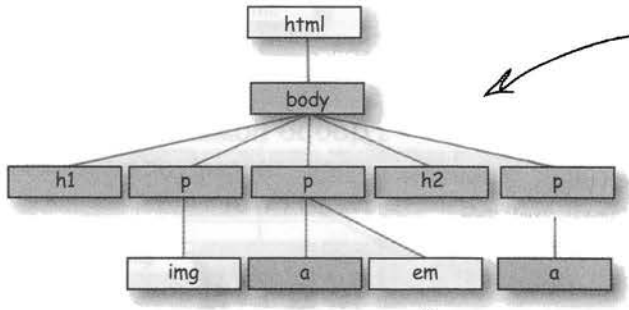
Сюрприз, сюрприз. Никаких изменений не произошло, но это как раз то, чего мы ожидали, не так ли? Все, что мы сделали, — переместили шрифт sans-serif в правило body и позволили всем остальным элементам его унаследовать.





Переопределение наследуемых свойств

Переместив свойство **font-family** в правило **body**, вы установили стиль для всей страницы. Но что делать, если вы не хотите, чтобы шрифт sans-serif был применен ко всем элементам? Например, вы можете решить, что для элементов **** лучше использовать шрифт serif.



Свойство font-family установлено в правиле body, поэтому каждый элемент, находящийся внутри body, наследует семейство шрифтов sans-serif.

Вы хотите, чтобы вместо него к элементу было применено семейство шрифтов serif. Нужно переопределить наследование новым CSS-правилом.

Отлично, значит, можно переопределить наследуемые правила, применив индивидуальное правило для элемента ****. Рассмотрим, как добавляется элемент **** для перекрытия свойства **font-family**, указанного в **body**:

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

h1, h2 {
      color:        gray;
}

h1 {
      border-bottom: 1px solid black;
}

p {
      color:        maroon;
}

em {
      font-family: serif;
}
```

Чтобы переопределить свойство font-family, наследуемое от body, добавьте новое правило и установите значение serif данного свойства для элемента .

Тест

Добавьте в свой CSS-код правило стиля для элемента **** со значением **serif** свойства **font-family** и обновите страницу `lounge.html`.

Обратите внимание, что для текста «станцевать разок-другой», который находится в элементе , сейчас используется шрифт serif.



Вообще, такое изменение шрифта для отдельного фрагмента текста внутри абзаца считается плохим тоном, поэтому, когда закончите тестирование, верните все в первоначальное состояние (без правила стиля для).

Часто Задаваемые Вопросы

В: Как браузер узнает, какое правило нужно применять к элементу , если я переопределяю значение наследуемого правила стиля?

О: В CSS всегда используется наиболее приоритетное правило. Итак, если у вас есть правило для <body> и более приоритетное правило, заданное только для элементов , то будет использовано более приоритетное правило. Чуть позже мы подробнее рассмотрим, какие правила стиля считаются более приоритетными.

В: Как узнать, какие CSS-правила являются наследуемыми, а какие нет?

О: Здесь пригодятся хорошие справочники, например *CSS Pocket Reference* издательства O'Reilly. Вообще, все стили, влияющие на внешний вид текста и определяющие его цвет, семейство шрифтов, с которым мы только что работали, и другие связанные со шрифтом свойства, например устанавливающие его размер, плотность (для полужирного шрифта) и начертание (для выделения курсивом), являются наследуемыми. Другие свойства, например граница, не наследуются, что логично, не так ли? Ведь если вы хотите, чтобы ваш элемент <body> был взят в рамку, это совсем не значит, что и все его внутренние элементы тоже должны быть взяты в рамку. В большинстве случаев вы можете сами догадаться, какие элементы наследуются, а какие — нет (или попытаться догадаться и затем проверить себя). Когда вы больше поработаете с различными свойствами и поймете, для чего они используются, у вас вообще не будет с этим проблем.

В: Всегда ли я могу переопределить свойство, унаследованное от родительского элемента, если мне нужно задать для него другое значение?

О: Да. Вы всегда можете использовать более приоритетный селектор, чтобы переопределить свойство, унаследованное от родительского элемента.

В: Все это кажется достаточно сложным. Существует ли какой-нибудь способ добавлять комментарии, чтобы напоминать самому себе, для чего используются эти правила?

О: Да. Чтобы добавить комментарий в CSS-код, просто поместите его между символами /* и */. Например:

```
/* Это правило применяется для
всех абзацев и устанавливает синий цвет текста */
```

Обратите внимание, что комментарий может быть расположен на нескольких строках. Вы также можете окружить CSS-код комментариями, и браузер проигнорирует его, вот так:

```
/* это правило ни к чему не применяется, потому что это комментарий

p { color: blue; }  */
```

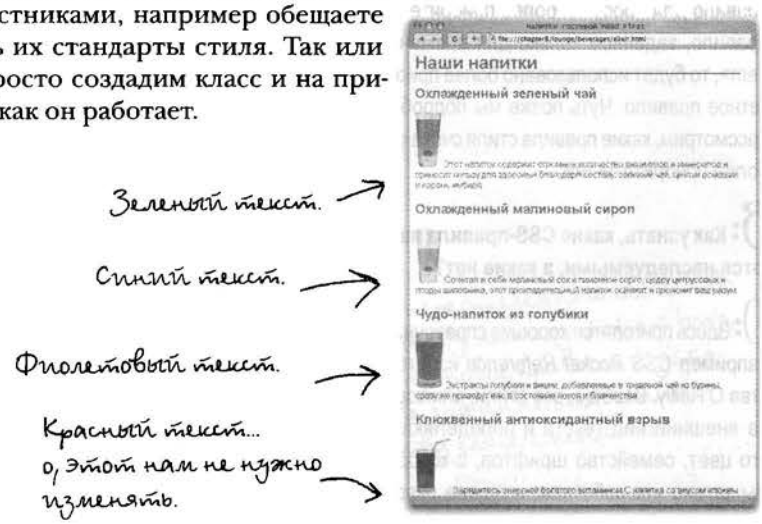
дальше >



Мы не уверены, что это будет действительно красиво смотреться, но, в конце концов, вы ведь заказчик.

Можно ли оформить каждый абзац отдельно, чтобы цвет текста сочетался с напитком? Проблема в том, что правило стиля с селектором **p** применяется для *всех* элементов **<p>**. Как же можно создать отдельное правило для каждого абзаца?

Вот тут-то и начинают работать *классы*. Используя XHTML и CSS вместе, можно задать классы элементов, а затем применить стиль к каждому элементу, принадлежащему этому классу. Чтобы понять, что такое класс, можете ассоциировать его с клубом «Зеленый чай». Вступая в этот клуб, вы принимаете все права и обязанности наравне с остальными участниками, например обещаете строго соблюдать их стандарты стиля. Так или иначе, давайте просто создадим класс и на примере посмотрим, как он работает.



Создание класса в файле elixir.html

Откройте файл `elixir.html` и найдите абзац «Охлажденный зеленый чай». Мы хотим поменять цвет этого текста на зеленый. Все, что нужно будет сделать, — добавить элемент **<p>** в класс под названием **greentea**. Вот как это выполнить:

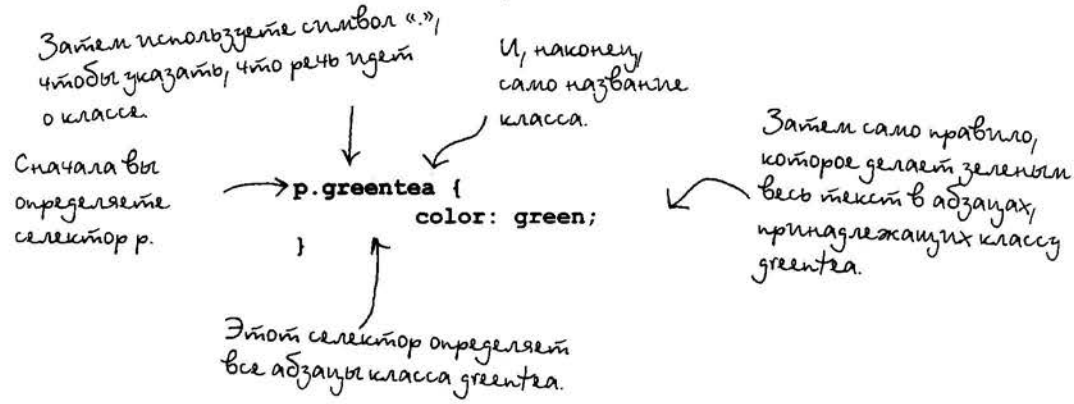
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />
    <title>Напитки гостевой Head First</title>
    <link type="text/css" rel="stylesheet" href="../lounge.css" />
  </head>
  <body>
    <h1>Наши напитки</h1>
    <h2>Охлажденный зеленый чай</h2>
    <p class="greentea">
            <img src="../images/green.jpg" />
            Этот напиток содержит огромное количество витаминов и минералов
            и приносит пользу для здоровья благодаря составу: зеленый чай,
            цветки ромашки и корень имбиря.
    </p>
    <h2>Охлажденный малиновый сироп</h2>
    <p>
            <img src="../images/lightblue.jpg" />
            Сочетая в себе малиновый сок и лимонное сорго, цедру
            цитрусовых и плоды шиповника, этот прохладительный напиток
            освежит и прояснит ваш разум.
    </p>
    <h2>Чудо-напиток из голубики</h2>
    <p>
            <img src="../images/blue.jpg" />
            Экстракты голубики и вишни, добавленные в травяной чай
            из бузины, сразу же приведут вас в состояние покоя
            и блаженства.
    </p>
    <h2>Клюквенный антиоксидантный взрыв</h2>
    <p>
            <img src="../images/red.jpg" />
            Зарядитесь энергией богатого витамином С напитка
            со вкусом клюквы и гибискуса.
    </p>
  </body>
</html>
```

Чтобы добавить элемент в класс, просто используйте в нем атрибут class со значением, соответствующим названию класса, например greentea.

Теперь, когда абзац с описанием охлажденного зеленого чая принадлежит классу **greentea**, остается лишь подготовить несколько правил для оформления этого класса элементов.

Создание селектора для класса

Чтобы описать класс, вы задаете селектор:



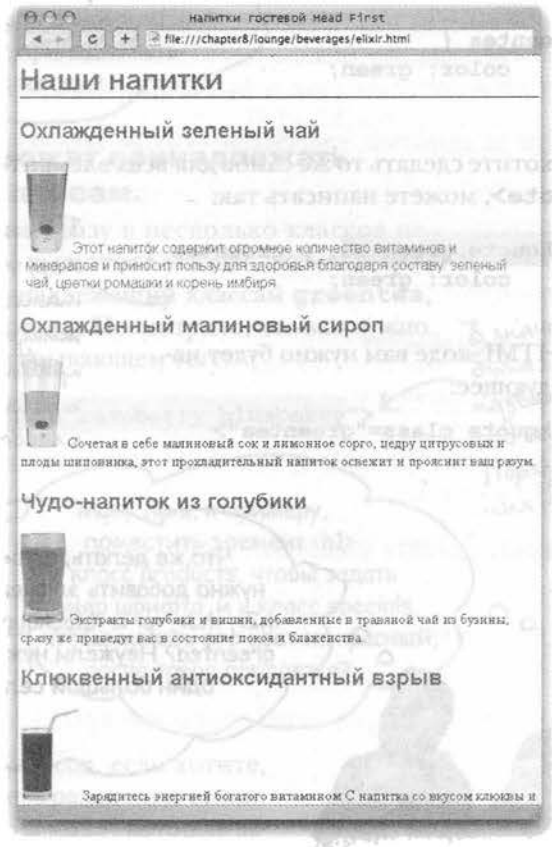
Итак, теперь вы знаете способ отобрать элементы **<p>**, принадлежащие конкретному классу. Все, что вам для этого нужно, — добавить атрибут **class** во все элементы **<p>**, содержимое которых вы хотите отображать на странице зеленым цветом. В результате такое правило будет применено к ним. Проверьте это сами: откройте файл **lounge.css** и добавьте в него селектор класса **p.greentea**.

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

h1, h2 {
     color: gray;
}

h1 {
     border-bottom: 1px solid black;
}

p {
     color: maroon;
}

p.greentea {
     color: green;
}
```

Тестирование класса greentea

Сохраните все изменения и обновите страницы, чтобы протестировать новый класс.

Новый класс greentea применен к абзацу. Теперь в нем текст зеленого цвета, что хорошо смотрится рядом с рисунком для охлажденного зеленого чая. Может быть, все-таки этот дизайн не был такой уж плохой идеей.



Возьми в руку карандаш

Ваша очередь: добавьте классы **raspberry** и **blueberry**, чтобы изменить оформление соответствующих абзацев на странице `elixir.html`, а затем напишите правила стилей для этих классов, чтобы придать тексту голубой и фиолетовый цвета соответственно. Значение свойства для `raspberry` — **blue**, а для `blueberry` — **purple**. Напишите их в самом низу CSS-файла, под правилом для класса **greentea**: сперва для **raspberry**, затем для **blueberry**.

Вы наверняка удивляетесь, как малина может быть голубой. Ну, если малиновый сироп на нашей страничке голубой, то для нас этого достаточно. И на самом деле, если вы присмотритесь к голубике, то заметите, что она больше фиолетовая, чем голубая. Просто доверьтесь нам и следуйте приведенным здесь инструкциям.

Поработаем с классами еще

Вы уже написали одно правило для класса **greentea**, которое меняет цвет текста во всех абзацах класса на зеленый:

```
p.greentea {
        color: green;
}
```

Если вы захотите сделать то же самое для всех элементов **<blockquote>**, можете написать так:

```
blockquote.greentea, p.greentea {
        color: green;
}
```

Просто добавьте еще один селектор, чтобы управлять элементами <blockquote>, принадлежащими классу greentea. Такое правило будет применяться к элементам <p> и <blockquote> класса greentea.

В своем XHTML-коде вам нужно будет написать следующее:

```
<blockquote class="greentea">
```



Нет, существует способ проще. Если вы хотите, чтобы правило стиля применилось ко всем элементам класса **greentea**, можете написать его таким образом:

```
.greentea {
        color: green;
}
```

Если вы опустите название всех элементов, добавьте точку (.) и сразу за ней — название класса, то правило будет применено ко всем членам класса.



Здорово! Это действительно работает. И еще один вопрос... Вы сравнили принадлежность к какому-либо классу с членством в клубе. Но я ведь могу стать членом множества клубов одновременно. А может ли элемент принадлежать нескольким классам?

Да, элемент может принадлежать нескольким классам.

Поместить элемент сразу в несколько классов нетрудно. Например, вы хотите указать элемент **<p>**, одновременно принадлежащий классам **greentea**, **raspberry** и **blueberry**. Посмотрите, что вам нужно будет написать в открывающем теге:

```
<p class="greentea raspberry blueberry">
```

В значении атрибута class укажите названия всех классов через пробелы. Порядок их следования неважен.

Могу ли я, к примеру, поместить элемент <h1> в класс products, чтобы задать размер шрифта, и в класс specials, чтобы менять его цвет на красный, когда товар продается?



Конечно. Используйте несколько классов, если хотите, чтобы к вашему элементу применились различные стили. В данном примере все ваши элементы **<h1>**, которые ассоциируются с товарами, будут иметь свой определенный стиль, но ведь не все ваши товары одновременно выставлены на продажу. Создав отдельный класс **specials** для выделения элементов красным цветом, вы можете просто добавить в него те элементы, которые ассоциируются с товарами, выставленными на продажу в данный момент.

Сейчас вас, скорее всего, интересует следующий вопрос: что произойдет, если элемент принадлежит нескольким классам, каждый из которых задает *одно и то же* свойство (как правило для элемента **<p>**, описанное чуть выше)? Как узнать, какой стиль будет применен? Вы знаете, что все эти классы задают свойство **color**. Итак, каким же будет цвет текста в абзаце: зеленым, голубым или фиолетовым?

Мы детально разберем это чуть позже, после того как вы чуть больше поработаете с CSS, но на следующей странице вы найдете краткое руководство, информации из которого вам хватит на данный момент.

Самое краткое в мире руководство по применению классов

Элементы, правила стиля, классы, наследование... Во всем этом можно запутаться. Как разобраться со всем этим и понимать, какие правила стиля к каким элементам должны применяться? Как мы уже сказали, для этого вам необходимо больше поработать с CSS, чем мы и займемся в следующих нескольких главах. Но перед этим давайте все-таки ознакомимся с основными принципами применения правил стилей.

Существует ли селектор, соответствующий элементу?

Представьте, что вам нужно узнать значение свойства **font-family** для элемента. Первое, что вы должны проверить: есть ли в вашем CSS-файле селектор для этого элемента. Если он там есть и при этом содержит свойство **font-family** с конкретным значением, то это значение и будет применяться для вашего элемента.

Как насчет наследования?

Если для вашего элемента селектора не нашлось, то придется полагаться на наследование. Итак, посмотрите на родительские элементы, их родительские элементы и т. д., пока не найдете определенное в них свойство, которое вам нужно. Если вы его найдете, то это и будет значение данного свойства для вашего элемента.

Снова не нашли нужного свойства? Тогда используйте установленное по умолчанию

Если ваш элемент не наследует значение ни от одного из своих предков, то используется значение свойства, установленное браузером по умолчанию. На самом деле эта схема немного сложнее, чем мы здесь описали, но подробно мы ее рассмотрим чуть позже.

Сразу несколько селекторов соответствуют одному элементу?

О, это как раз тот случай, который был у нас с абзацем, принадлежащим всем трем классам:

```
<p class="greentea raspberry blueberry">
```

Здесь есть множество селекторов, соответствующих этому элементу, и они определяют одно и то же свойство **color**. Мы называем это «конфликтом». Как же он разрешается? Итак, побеждает правило с наивысшим *приоритетом*. Но как же этот приоритет определяется? В следующих главах мы вернемся к этому вопросу и *подробно* объясним, как определять приоритет селектора, а пока посмотрим лишь на основные правила, чтобы получить кое-какое представление о приоритетности:

```
p { color: black;}

.greentea { color: green; }

p.greentea { color: green; }

p.raspberry { color: blue; }

p.blueberry { color: purple; }
```

Это правило для любого элемента <p>.

Это правило для любого члена класса greentea. Оно более приоритетно.

Это правило только для абзацев из класса greentea, поэтому оно еще более приоритетно.

Эти правила тоже лишь для абзацев из конкретного класса. Так что у них с правилом p.greentea одинаковый приоритет.

У нас до сих пор нет явного победителя?

Итак, если бы ваш элемент принадлежал только классу `greentea`, то у вас был бы явный победитель: селектор **`p.greentea`** имеет наибольший приоритет, поэтому текст станет зеленым. Но наш элемент принадлежит *сразу трем* классам: **`greentea`**, **`raspberry`** и **`blueberry`**. Итак, селекторы **`p.greentea`**, **`p.raspberry`** и **`p.blueberry`** соответствуют элементу **`p`** и имеют одинаковый приоритет. Что же теперь делать? Выбирать *последнее* правило в CSS-файле. Если нельзя разрешить конфликт, поскольку два селектора имеют одинаковый приоритет, то используется порядок следования правил в файле с таблицами стилей. Это значит, что применяется последнее правило в CSS-файле (которое ближе всех к концу файла). И в рассматриваемом примере это будет правило **`p.blueberry`**.



Упражнение

В файле `lounge.html` измените элемент абзаца для описания охлажденного зеленого чая так, чтобы он включал в себя все три класса:

```
<p class="greentea raspberry blueberry">
```

Сохраните файл и обновите страницу. Какого цвета стал текст в абзаце с описанием чая? _____

Далее поменяйте порядок следования файлов в вашем XHTML-коде:

```
<p class="raspberry blueberry greentea">
```

Сохраните файл и обновите страницу. Какого цвета теперь стал текст? _____

Откройте свой CSS-файл и переместите правило `p.greentea` в самый конец.

Сохраните файл и обновите страницу. Какого цвета стал текст в абзаце с описанием чая? _____

И наконец, переместите правило `p.raspberry` в самый конец файла.

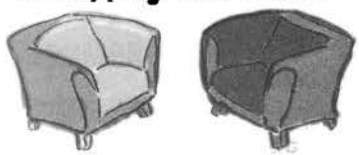
Сохраните файл и обновите страницу. Какого цвета теперь стал текст? _____

Когда вы со всем этим справитесь, верните элемент, описывающий абзац, к первоначальному виду:

```
<p class="greentea">
```

Сохраните файл и обновите страницу. Какого цвета теперь стал текст в абзаце с описанием чая? _____

Беседа у камина



Вечерний диалог: **сравнение CSS и XHTML.**

CSS

Ты это видишь? Я словно Гудини! Я вырвался из твоего элемента **<style>** и убежал в свой собственный файл. А в главе 1 ты говорил, что я никогда не смогу сбежать от тебя.

Приходится ссылаться? Да брось; ведь ты знаешь, что без моего стиля твои страницы ничего не стоят.

Если ты внимательно читал эту главу, то должен был заметить, что я очень хорошо знаю свое дело.

Что ж, это уже намного лучше. Мне нравится твоя новая позиция.

XHTML

Не преувеличивай; мне все еще приходится ссылаться на тебя, чтобы ты был хоть чем-нибудь полезен.

Опять ты за свое... В то время, пока я и мои элементы пытаемся создать стройную структуру страницы, ты говоришь о краске для волос и лаке для ногтей.

Ладно, ладно, с этим я соглашусь; использование CSS явно облегчает мою работу. Все эти старые (и уже не соответствующие стандартам) элементы, которые отвечали за оформление страниц, доставляли мне немало хлопот. И мне нравится то, что мои элементы могут быть стилизованы без вставки специального куска кода в XHTML, за исключением редкого атрибута `class`.

Но я все еще не могу забыть, как ты посмеивался над моим синтаксисом... <помнишь>?

CSS

Ты должен согласиться, что все еще довольно неуклюж и унаследовал это со времен, когда использовались старые технологии 1990-х годов.

Ты шутишь? Я достаточно точен и выразителен. Я могу выбрать любой элемент и подробно описать, как хочу его оформить. И ты видел только немногое из того, что я умею.

Да-да; просто наберись терпения, и ты в этом сам убедишься. Я могу очень разнообразно и красиво оформлять текст и фон страницы. Я даже могу контролировать то, как каждый элемент влияет на внешний вид части страницы, прилегающей к нему.

Ха-ха-ха. А ты думал, что контролировал меня, заключив в свои теги **<style>**? Ты увидишь, что, если я захочу, смогу вывернуть твои элементы наизнанку.

XHTML

Но такой синтаксис прошел испытание временем. А ты думаешь, что сам очень изящен? Я имею в виду, что ты – всего лишь набор правил. Как вообще это можно назвать языком?

Неужели?

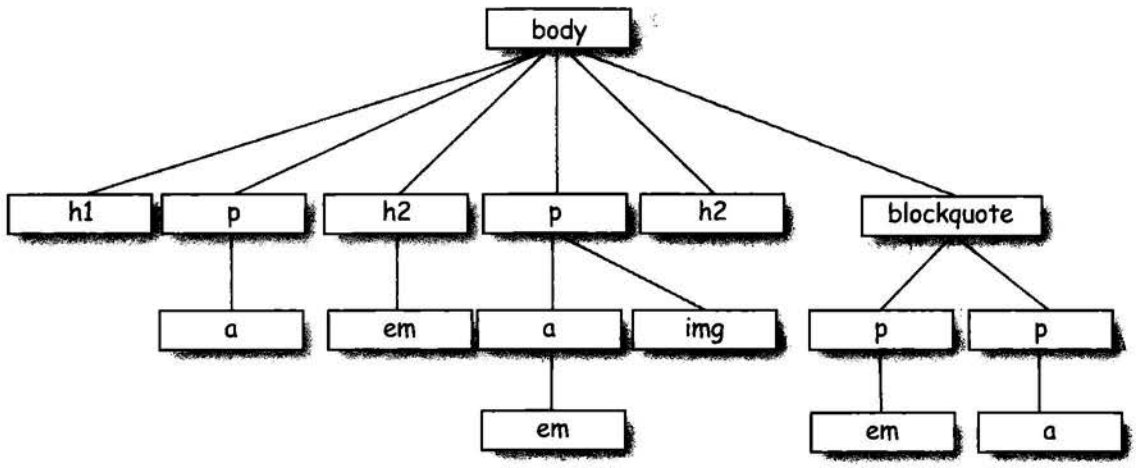
Хмммм... это звучит так, будто у тебя больше возможностей, чем должно быть. И мне это не очень нравится. В конце концов, мои элементы хотят быть сами себе хозяевами.

Эй, охрана... охрана?!



Кто унаследует признак?

Ох, элемент **<body>** ушел из нашего мира, но оставил после себя множество потомков и наследство в виде зеленого цвета. Ниже вы найдете его фамильное дерево. Отметьте всех потомков, которые наследуют зеленый цвет элемента **<body>**. Но сначала взгляните на CSS-код, приведенный ниже.




```
body {
       color: green;
}

p {
       color: black;
}
```

Это CSS-код. Используйте его, чтобы определить, кому из перечисленных выше элементов крупно повезло, и они унаследовали зеленый цвет.

Поработайте браузером

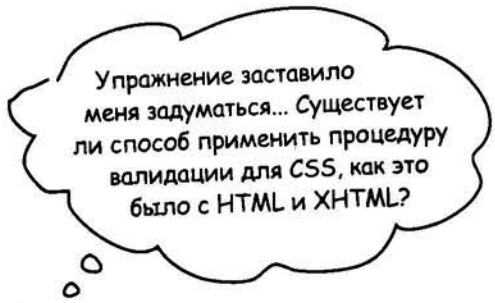
Если в вашем CSS-коде есть ошибку, то все правила, описанные ниже этой ошибки, просто игнорируются. Так что сразу привыкайте искать ошибки заранее. В этом вам поможет следующее упражнение.



Ниже вы найдете CSS-файл, в котором есть несколько ошибок. Ваша задача — поставить себя на место браузера и найти эти ошибки. После того как сделаете это, можете свериться с нашим решением в конце главы.

Файл style.css.

```
<style>

body {
    background-color: white

h1, {
    gray;
    font-family: sans-serif;
}

h2, p {
    color:
}

<em> {
    font-style: italic;
}

</style>
```





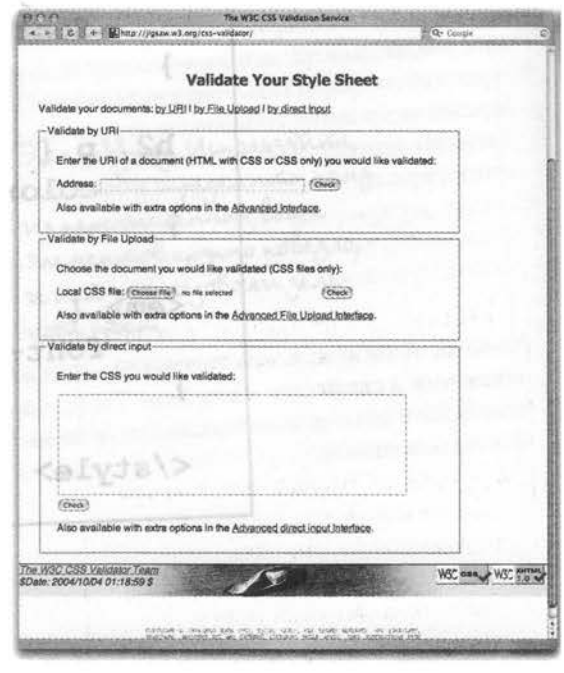
Конечно, да!

W3C-разработчики не просто просиживают свои штаны, а усердно работают. Их CSS-валидатор вы найдете по адресу:

`http://jigsaw.w3.org/css-validator/`

Введя этот URL в адресную строку браузера, вы наверняка почувствуете себя на этом сайте как дома. Вы найдете здесь валидатор, который работает точно так же, как валидатор для HTML и XHTML. Просто укажите ему URL-адрес CSS-файла, загрузите файл с кодом на этот сайт или просто скопируйте его в соответствующую форму и запустите проверку.

Вам не нужно указывать DOCTYPE или кодировку символов для CSS. Итак, приступайте! Попробуйте использовать этот валидатор (так или иначе, на следующей странице вам уже придется пользоваться им).



Убедимся, что CSS-код для гостевой валидный

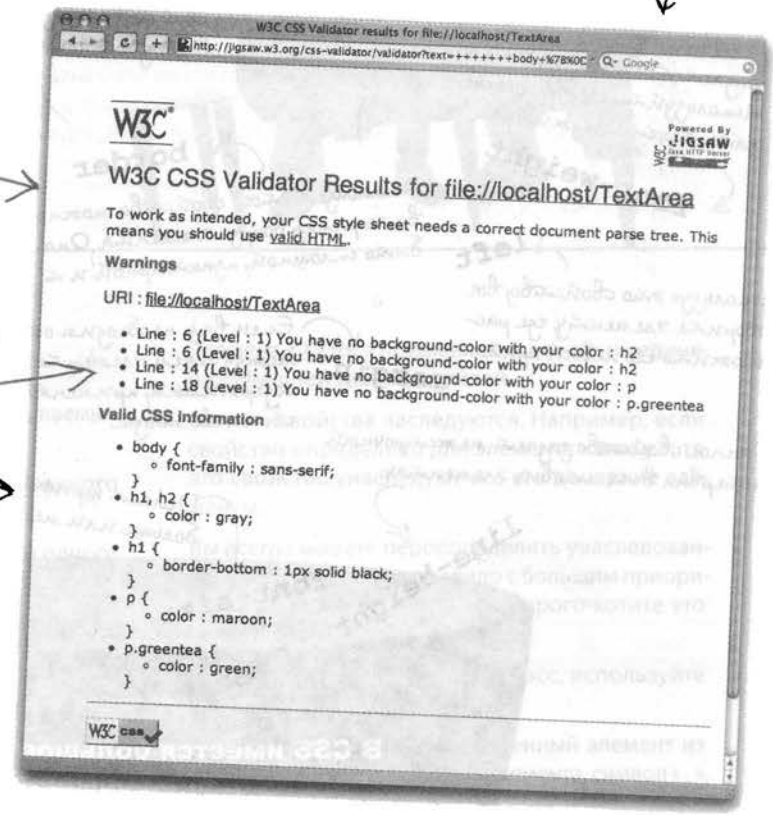
Давайте убедимся, что весь CSS-код для гостевой Head First соответствует стандартам, чтобы спокойно приступать к изучению следующей темы. Используйте любой способ, чтобы «показать» код W3C. Если ваш CSS находится на сервере, просто введите его URL-адрес в соответствующую форму. В ином случае либо загрузите CSS-файл на сайт, либо скопируйте код в соответствующую форму (если вы используете загрузку, убедитесь, что указываете на CSS-, а не на XHTML-файл). Когда справитесь с этим, нажмите кнопку Check (Проверить).

Если ваш код не валидный, сверьте его с тем, что приводился пару страниц назад, и найдите ошибки. После этого перепроверьте код в валидаторе.

Здесь просто говорится, что для вашего CSS необходим корректный XHTML-код, поэтому удостоверьтесь, что ваш XHTML (или HTML) валидный.

Это предупреждающие сообщения для CSS, которые правильнее будет назвать советами. Например, во всех этих сообщениях предлагается установить цвет фона для заголовков и абзацев.

Это CSS-код, который признан валидным. Как видите, это ВЕСЬ ваш код. Это значит, что ваш CSS успешно прошел процедуру валидации.



Часто Задаваемые Вопросы

В: Стоит ли мне обращать внимание на эти предупреждающие сообщения? О чем они говорят?

О: Хорошо бы их просматривать, хотя некоторые из них можно отнести в категорию советов, а не правил, «обязательных для выполнения». Однако иногда валидатор бывает чересчур дотошным, просто помните об этом.

Коктейль из свойств

top

Определяет позицию верхнего края элемента.

text-align

Это свойство используется для выравнивания текста по левому краю, по центру или по правому краю.

letter-spacing

Позволяет задавать интервал между символами в пределах элемента.

Используйте свойство color для задания цвета текста элемента.

color

background-color

Это свойство определяет фоновый цвет элемента.

Задает толщину шрифта. Используйте его, чтобы сделать текст полужирным.

font-weight

Это свойство применяется для выделения текста курсивом.

font-style

border

С помощью этого свойства можно нарисовать рамку вокруг элемента. Она может быть сплошной, пунктирной и т. д.

left

Используя это свойство, вы говорите элементу, где расположить его левый край.

Это свойство позволяет контролировать то, как будут выглядеть пункты списка.

list-style

margin

Если вам необходим отступ между краем элемента и его содержимым, применяйте свойство margin.

Данное свойство задает межстрочный интервал в текстовых элементах.

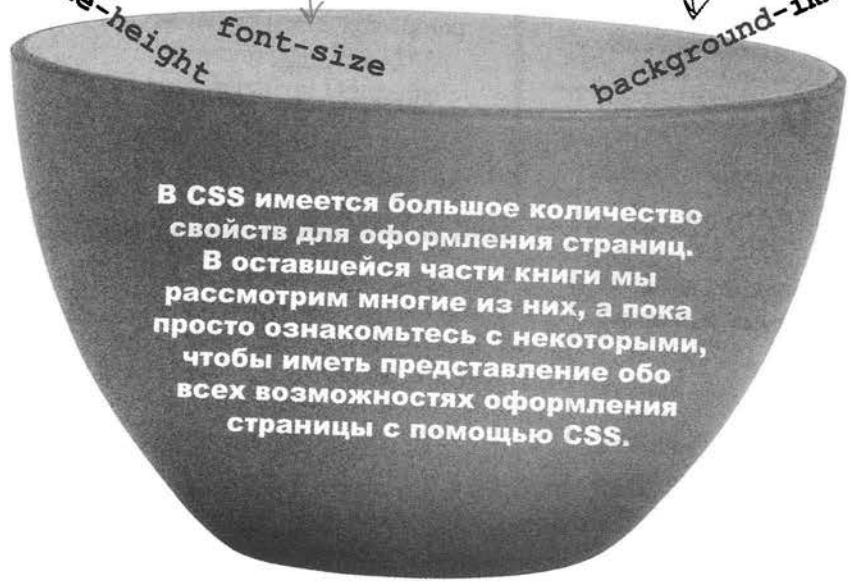
line-height

Делает шрифт больше или меньше

font-size

Используйте это свойство, чтобы поместить изображение под элементом.

background-image



В CSS имеется большое количество свойств для оформления страниц. В оставшейся части книги мы рассмотрим многие из них, а пока просто ознакомьтесь с некоторыми, чтобы иметь представление обо всех возможностях оформления страницы с помощью CSS.



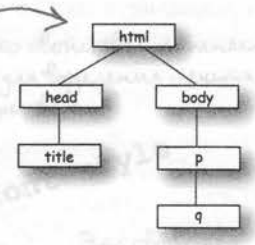
ПОВТОРИМ выученное

- CSS состоит из простых операторов, называемых правилами.
- Каждое правило задает стиль для выбранного XHTML-элемента.
- Обычно правило состоит из селектора и одного или нескольких свойств и их значений.
- Селектор указывает, к какому элементу применяется данное правило.
- В конце описания каждого свойства ставится точка с запятой.
- Все свойства и их значения для каждого правила заключаются в фигурные скобки.
- Вы можете выбрать любой элемент, используя в качестве селектора его название.
- Можно выбрать сразу несколько элементов, разделив их названия запятыми.
- Один из самых простых способов стилизовать HTML — использовать тег **<style>**.
- При использовании XHTML-кода и при создании сайтов с достаточно большим уровнем сложности лучше ссылаться на внешние таблицы стилей.
- Элемент **<link>** применяется для присоединения внешней таблицы стилей.
- Многие свойства наследуются. Например, если свойство определено для элемента **<body>**, то это свойство унаследуют все его дочерние элементы.
- Вы всегда можете переопределить унаследованное свойство, создав правило с большим приоритетом для того элемента, у которого хотите это свойство поменять.
- Чтобы добавить элементы в класс, используйте атрибут **class**.
- Если хотите выбрать определенный элемент из класса, указывайте название элемента, символ «.», а затем название класса.
- Чтобы выбрать все элементы класса, сразу пишите **.название_класса**.
- Элемент может принадлежать нескольким классам. Для этого в значении атрибута **class** укажите нужные классы через пробел.
- Вы можете проверить свой CSS-код на валидность, используя валидатор W3C, который находится по адресу **http://jigsaw.w3.org/css-validator**.

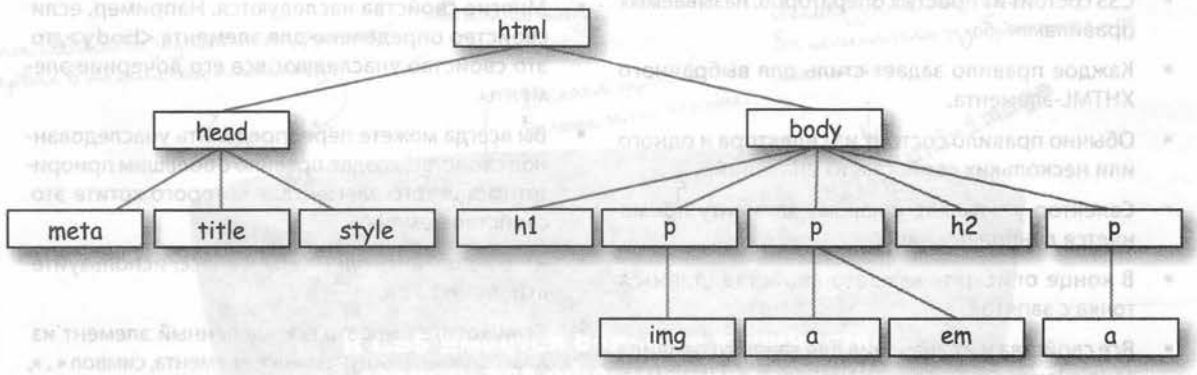


Магниты для разметки

Вот так.



Помните, как вы рисовали диаграмму из элементов XHTML в главе 3? Теперь вы должны были это сделать для главной страницы гостевой. Вот решение этого упражнения:

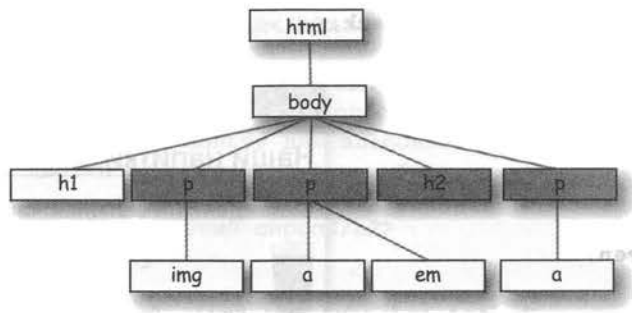


Возьми в руку карандаш

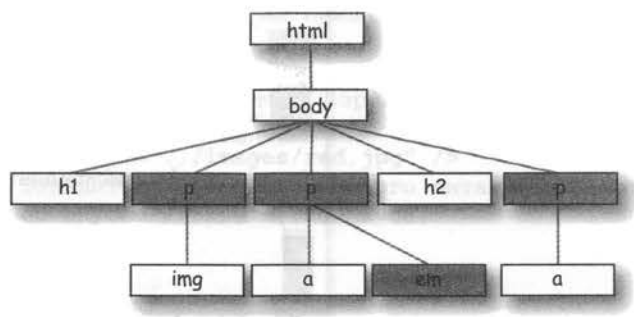
Решение

Выбранные элементы выделены цветом:

```
p, h2 {
    font-family: sans-serif;
}
```




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

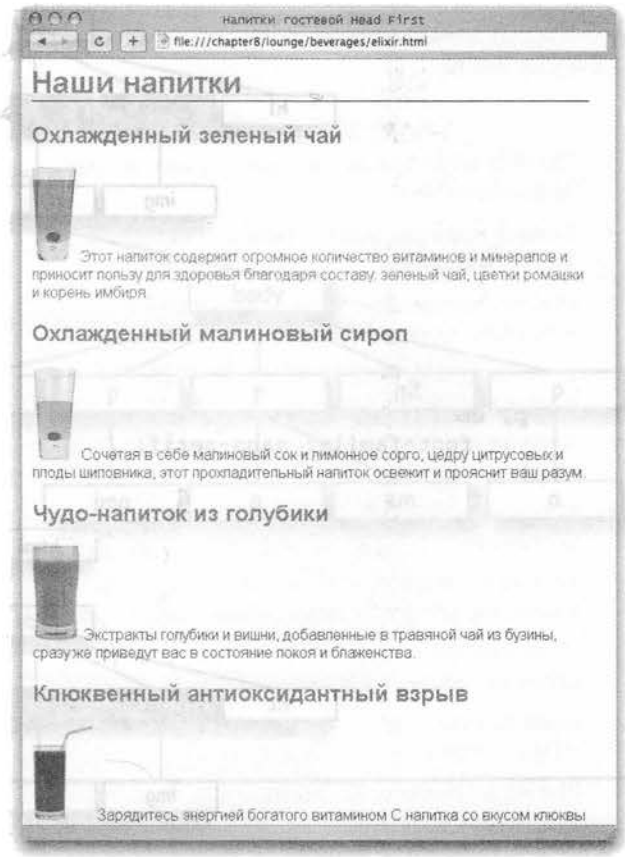


Возьми в руку карандаш

Решение

Ваша очередь: добавьте классы **raspberry** и **blueberry**, чтобы изменить оформление соответствующих абзацев на странице `elixir.html`, а затем напишите правила стилей для этих классов, чтобы придать тексту голубой и фиолетовый цвета соответственно. Значение свойства для **raspberry** — **blue**, а для **blueberry** — **purple**.

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

h1, h2 {
        color: gray;
}

h1 {
        border-bottom: 1px solid black;
}

p {
        color: maroon;
}

p.greentea {
        color: green;
}

p.raspberry {
        color: blue;
}

p.blueberry {
        color: purple;
}
```



Возьми в руку карандаш

Решение

```
<!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="ru" xml:lang="ru">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />
    <title>Напитки гостевой Head First</title>
    <link type="text/css" rel="stylesheet" href="../lounge.css" />
  </head>
  <body>
    <h1>Наши напитки</h1>
    <h2>Охлажденный зеленый чай</h2>
    <p class="greentea">
            <img src="../images/green.jpg" />
            Этот напиток содержит огромное количество витаминов и минералов
            и приносит пользу для здоровья благодаря составу: зеленый чай,
            цветки ромашки и корень имбиря.
    </p>
    <h2>Охлажденный малиновый сироп</h2>
    <p class="raspberry">
    <p>
            <img src="../images/lightblue.jpg" />
            Сочетая в себе малиновый сок и лимонное сорго, цедру
            цитрусовых и плоды шиповника, этот прохладительный напиток
            освежит и прояснит ваш разум.
    </p>
    <h2>Чудо-напиток из голубики</h2>
    <p class="blueberry">
    <p>
            <img src="../images/blue.jpg" />
            Экстракты голубики и вишни, добавленные в травяной чай
            из бузины, сразу же приведут вас в состояние покоя
            и блаженства.
    </p>
    <h2>Клюквенный антиоксидантный взрыв</h2>
    <p>
            <img src="../images/red.jpg" />
            Зарядитесь энергией богатого витамином С напитка
            со вкусом клюквы и гибискуса.
    </p>
  </body>
</html>
```



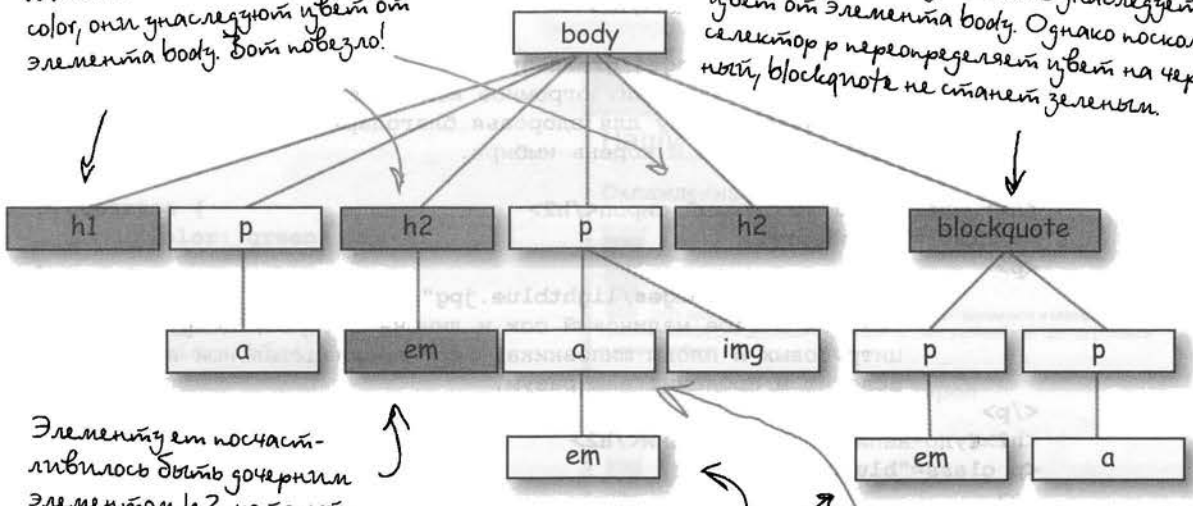
Решение упражнения

Кто унаследует признак?

```
body {
        color: green;
}

p {
        color: black;
}
```

Поскольку у элементов h1 и h2 нет своего свойства color, они унаследуют цвет от элемента body. Вот повезло!

Для элемента blockquote нет CSS-правила, поэтому он тоже унаследует цвет от элемента body. Однако поскольку селектор p переопределяет цвет на черный, blockquote не станет зеленым.



Элементу em посчастливилось быть дочерним элементом h2, который, в свою очередь, унаследовал цвет от body. А поскольку нет отдельного правила для em, которое перекрывало бы заданный цвет своим собственным, то em наследует цвет от body.

Эти «бедные» элементы a также являются дочерними для элемента p, поэтому они не унаследовали цвет от body.

К несчастью для этих элементов em, в их родительских элементах p переопределяется цвет, унаследованный от body. Таким образом, они не получают цвета в наследство от body.

img — дочерний элемент для p, поэтому он не наследует цвет от body. Он ни в каком случае не получит цвет в наследство.



Решение упражнения



Поработайте браузером

Ниже вы найдете CSS-файл, в котором есть несколько ошибок. Ваша задача — поставить себя на место браузера и найти эти ошибки. Вы нашли их все?

```
<style>

body {
    background-color: white

h1,
    gray;
    font-family: sans-serif;
}

h2, p {
    color:
}

<em>
    font-style: italic;
}

</style>
```

В вашем CSS-коде не должно быть никакого XHTML! Тег <style> — это XHTML, и он не работает в таблицах стиля CSS.

Не хватает точки с запятой.

Вот здесь нужна скобка }.

Лишняя запятая.

Не указано название свойства.

Не хватает значения свойства и точки с запятой.

Вместо названия элемента использован тег XHTML. Должно быть написано просто em.

В таблицах CSS тег </style> не используется.



Решение упражнения

В файле `lounge.html` измените элемент абзаца для описания охлажденного зеленого чая так, чтобы он включал в себя все три класса:

```
<p class="greentea raspberry
blueberry">
```

Сохраните файл и обновите страницу. Какого цвета стал текст в абзаце с описанием чая?

Фиолетовый.

Используется фиолетовый цвет, потому что правило для класса blueberry указано последним в CSS-файле.

Далее поменяйте порядок следования файлов в вашем XHTML-коде:

```
<p class="raspberry blueberry
greentea">
```

Сохраните файл и обновите страницу. Какого цвета теперь стал текст?

Фиолетовый.

Все еще используется фиолетовый цвет, потому что порядок следования классов в атрибуте class не имеет никакого значения.

Откройте свой CSS-файл и переместите правило `p.greentea` в самый конец.

Сохраните файл и обновите страницу. Какого цвета стал текст в абзаце с описанием чая?

Зеленый.

На этот раз используется зеленый цвет, так как теперь правило для класса greentea указано в CSS-файле последним.

И наконец, переместите правило `p.raspberry` в самый конец файла.

Сохраните файл и обновите страницу. Какого цвета теперь стал текст?

Голубой.

Сейчас используется голубой цвет, так как теперь в CSS-файле последним стоит правило для класса raspberry.

Когда вы со всем этим справитесь, верните элемент, описывающий абзац, к первоначальному виду:

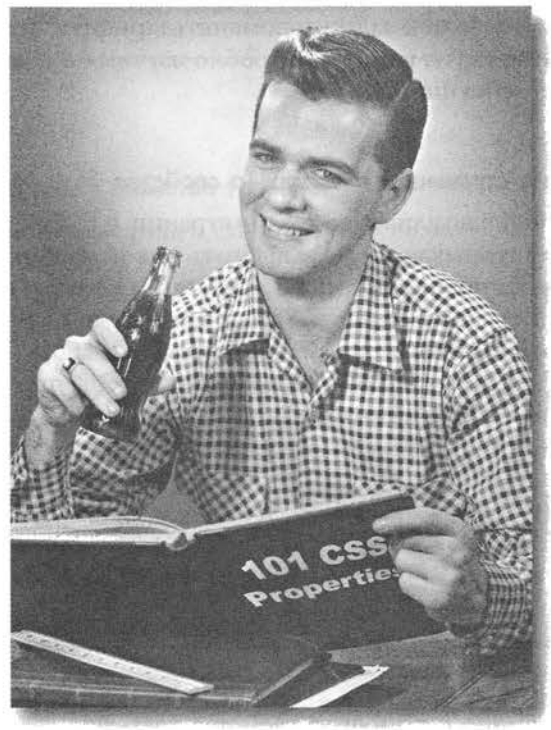
```
<p class="greentea">
```

Сохраните файл и обновите страницу. Какого цвета теперь стал текст в абзаце с описанием чая?

Зеленый.

Отлично, теперь элемент <p> принадлежит только одному классу и используется правило с наибольшим приоритетом — правило p.greentea.

Увеличиваем словарный запас



Ваше изучение языка CSS проходит успешно. Вы уже ознакомились с основами CSS и знаете, как создавать правила, выбирать элементы и определять для них стили. Теперь настало время увеличить ваш словарный запас, а это означает, что вам нужно познакомиться с некоторыми новыми свойствами и узнать, что они могут делать. В настоящей главе мы поработаем с несколькими наиболее используемыми свойствами, которые влияют на оформление текста. Для этого вам придется кое-что узнать о цветах и шрифтах. Вы поймете, что совершенно не обязательно устанавливать те шрифты, которые применяются повсеместно, или те размеры и стили, что по умолчанию используются браузерами для абзацев и заголовков. Вы также узнаете, что существует намного больше цветов, чем может различить ваш глаз.

Самое главное о тексте и шрифтах

Множество CSS-свойств созданы для того, чтобы помочь вам оформить текст. С помощью CSS вы можете контролировать гарнитуру шрифта, его стиль, цвет и даже декоративные элементы, встроенные в текст, — обо всем этом мы расскажем в данной главе. Начнем с изучения существующих шрифтов, которые используются для отображения текста на ваших страницах. Вы уже видели свойство **`font-family`**, а в этой главе узнаете больше о назначении различных шрифтов.

Перед тем как приступить, рассмотрим основные свойства, которые вы можете использовать, чтобы задать и поменять шрифты. Затем мы рассмотрим базовые шрифты и очень подробно изучим особенности использования каждого из них.

Задавайте шрифты на страницах с помощью свойства `font-family`.

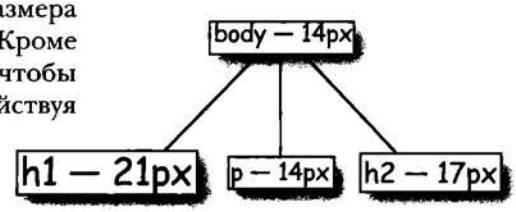
Шрифты могут очень сильно влиять на дизайн страниц. В CSS они разделены на семейства, в которых вы можете выбрать, какой шрифт лучше подходит для определенного элемента на странице. На большинстве компьютеров обычно установлен только строго определенный набор шрифтов, так что вам нужно быть осторожными при их выборе. В этой главе мы расскажем все, что вам необходимо знать, чтобы правильно задавать шрифты и извлекать из этого максимум пользы.

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

Andale Mono
Arial
Arial Black
Comic Sans
Courier New
Georgia
Impact
Times New Roman
Trebuchet MS
Verdana

Задавайте размер шрифта с помощью свойства `font-size`.

Размер шрифта очень влияет на дизайн веб-страницы и читабельность ее текста. В CSS есть несколько способов задания размера шрифта, и в этой главе мы рассмотрим каждый из них. Кроме того, вы узнаете, как задавать шрифт таким способом, чтобы ваши пользователи могли увеличивать его размер, воздействуя при этом на дизайн страницы.

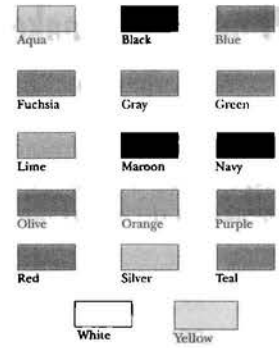



```
body {
      font-size: 14px;
}
```

Определяйте цвет текста с помощью свойства color.

Благодаря этому свойству можно задать любой цвет текста. На данном этапе очень полезно знать кое-что о «безопасных» (web-safe) цветах, и мы совсем скоро расскажем вам о них очень детально, рассмотрев также их загадочные шестнадцатеричные коды.

```
body {
      color: silver;
}
```



Назначайте начертание шрифтов с помощью свойства font-weight.

Какой смысл довольствоваться скучными шрифтами стандартного вида, если можно, например, увеличить их жирность там, где это необходимо? Ваш шрифт выглядит слишком громоздко? Уменьшите его жирность до стандартной. Все это легко сделать, используя свойство **font-weight**.

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

lighter

normal

bold

bolder

Оформляйте свой текст еще лучше с помощью свойства text-decoration.

Применяя свойство **text-decoration**, вы можете сделать текст надчеркнутым, подчеркнутым или зачеркнутым. Кроме того, разработчики CSS добавили этому свойству значение, позволяющее сделать текст мерцающим (хотя от браузеров не требуется поддержка этого значения).

```
body {
      text-decoration: underline;
}
```

none

underline

overline

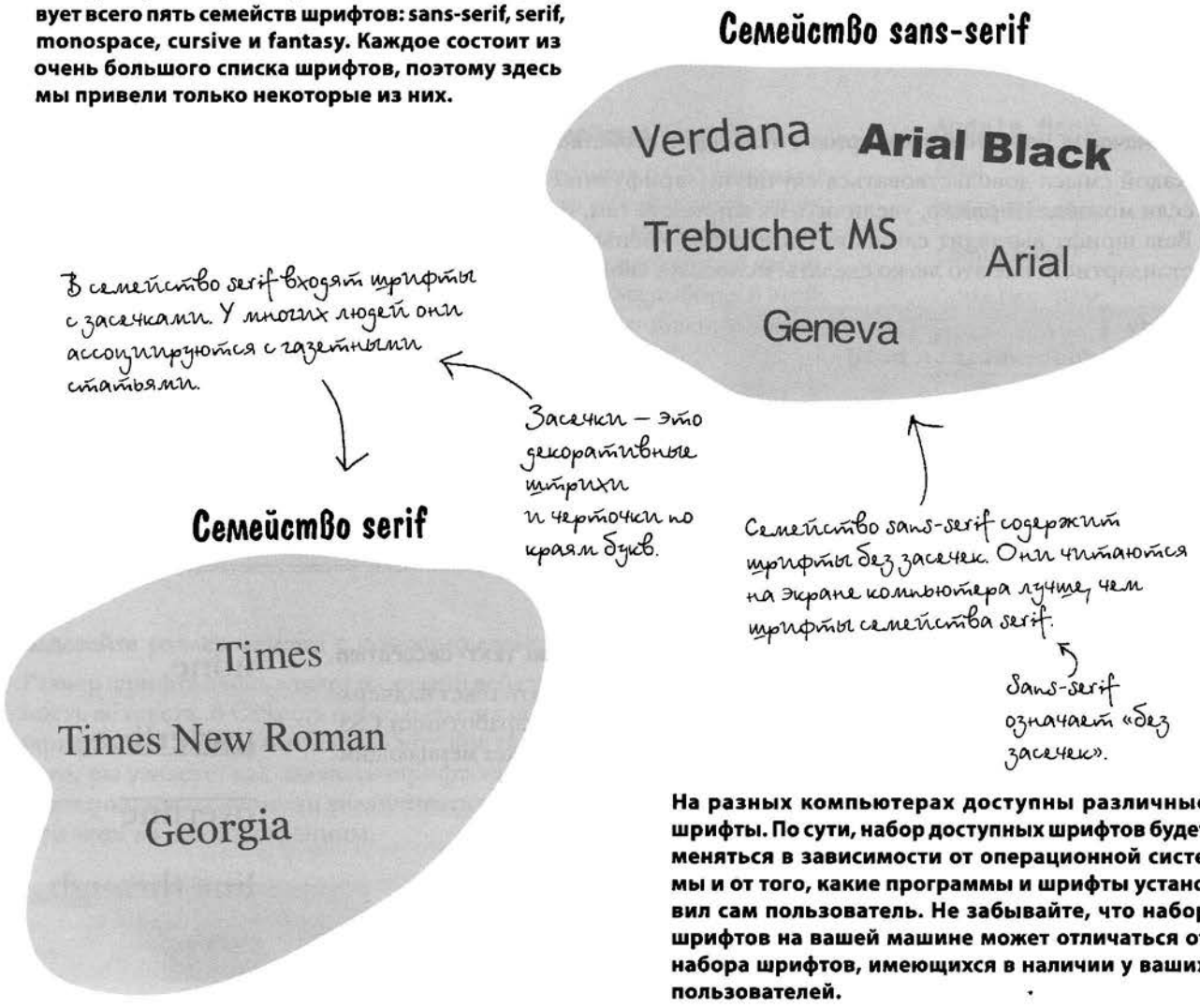
~~line-through~~

blink

Итак, что такое семейство шрифтов?

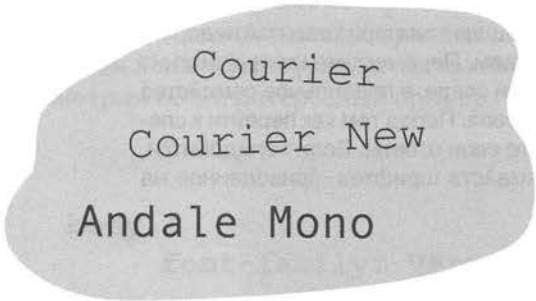
Вы уже встречались со свойством **font-family** и даже задавали ему значение **sans-serif**. Данное свойство предоставляет большие возможности по оформлению страниц, однако сначала рассмотрим, что это вообще такое — семейство шрифтов. Здесь приводятся краткие сведения по этому вопросу...

Каждое семейство состоит из набора шрифтов, обладающих общими характеристиками. Существует всего пять семейств шрифтов: sans-serif, serif, monospace, cursive и fantasy. Каждое состоит из очень большого списка шрифтов, поэтому здесь мы привели только некоторые из них.



На разных компьютерах доступны различные шрифты. По сути, набор доступных шрифтов будет меняться в зависимости от операционной системы и от того, какие программы и шрифты установил сам пользователь. Не забывайте, что набор шрифтов на вашей машине может отличаться от набора шрифтов, имеющихся в наличии у ваших пользователей.

Семейство monospace

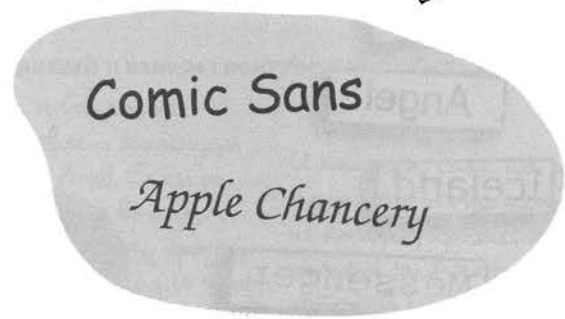


Семейство monospace состоит из шрифтов, символы которых имеют одинаковую фиксированную ширину. Например, символ «i» будет иметь такую же ширину, как символ «m». Эти шрифты используются в основном для отображения примеров кода программ.

Еще раз внимательно посмотрите на семейства шрифтов: текст, напечатанный шрифтом из семейства serif, выглядит элегантно и традиционно, в то время как текст, напечатанный шрифтом из семейства sans-serif, очень четкий и хорошо читается. Тексты, напечатанные шрифтом из семейства monospace, выглядят так, будто были напечатаны на пишущей машинке. Шрифты из семейств cursive и fantasy применяются для художественного оформления или в декоративных целях.

Семейство cursive состоит из шрифтов, буквы в которых подобны рукописным. Иногда вы будете встречать такие шрифты в заголовках.

Семейство cursive



Семейство fantasy



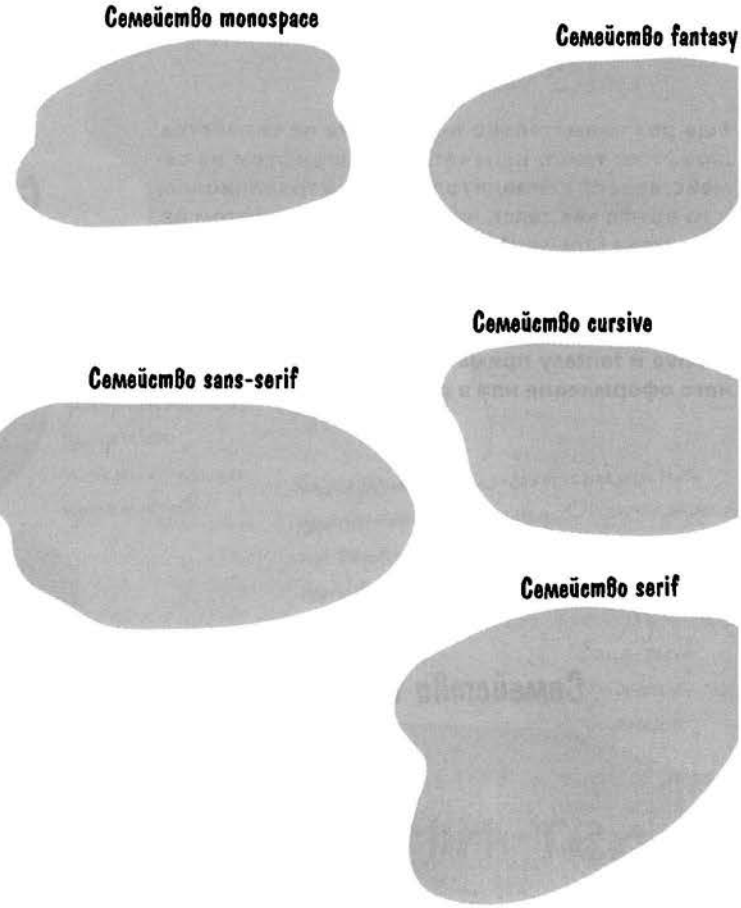
И последнее семейство шрифтов — fantasy. Оно состоит из художественных и декоративных шрифтов. Эти шрифты не очень широко распространены, доступны не на всех компьютерах и редко используются для серьезного веб-дизайна.



Магниты для шрифтов

Ваша задача — помочь вымышленным шрифтам найти дорогу домой, к их родным семействам. Переместите каждый магнит для разметки, расположенный слева, в правильное семейство шрифтов, расположенное справа. Перед тем как перейти к следующему разделу, проверьте свои ответы. Если понадобится, пересмотрите описание семейств шрифтов, приведенное на предыдущих страницах.





Определение семейств шрифтов в CSS

Итак, существует множество хороших шрифтов, принадлежащих нескольким семействам. Но как же использовать их на страницах? В прошлой главе вы уже немного поработали со свойством **font-family**, когда на странице гостевой задавали ему значение **sans-serif**. Рассмотрим более интересный пример.

Обычно характеристика семейства шрифтов представляет собой список принадлежащих ему взаимозаменяемых шрифтов.

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

В свойстве font-family можно определить несколько шрифтов. Просто укажите их названия через запятую.

Пишите названия шрифтов правильно, с учетом регистра букв.

В конце списка всегда указывайте название семейства, например serif, sans-serif, cursive или monospace. Для чего это нужно, вы узнаете через минуту.

Как работает свойство font-family

Посмотрите, как браузер понимает список шрифтов, заданный в вашем свойстве **font-family**:

Проверяет, установлен ли шрифт Verdana на компьютере, и если да, использует его в качестве шрифта для этого элемента (в данном случае для элемента <body>).

Если Verdana не установлен, то пишет шрифт Geneva. В случае успешного поиска использует его для <body>.

Если и Geneva не установлен, то пишет шрифт Arial. Если он имеется на вашем компьютере, то браузер применяет его для <body>.

И наконец, если ни один из указанных шрифтов не найден, применяется шрифт sans-serif, который считается браузером используемым по умолчанию.

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

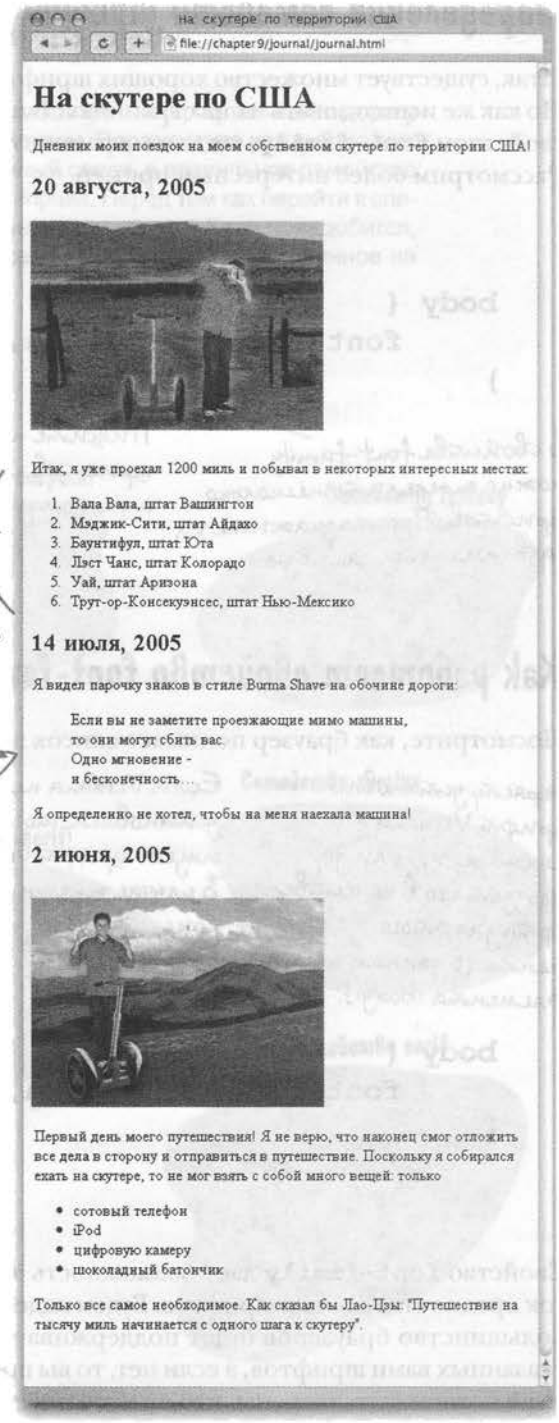
Вам не обязательно указывать четыре взаимозаменяемых шрифта. Можете указать два, три и т. д. В предыдущей главе мы использовали только один, применяемый по умолчанию шрифт sans-serif. Надо отметить, что так делать нежелательно, потому что это не дает возможности в полной мере контролировать шрифты, которые будут использованы.

Свойство **font-family** дает возможность задать список предпочтительных шрифтов. Будем надеяться, что большинство браузеров будет поддерживать один из указанных вами шрифтов, а если нет, то вы по крайней мере можете быть уверены, что он использует типовой шрифт из этого же семейства.

А теперь поменяем несколько шрифтов на ваших страницах...

Вновь поработаем с дневником Тони

Теперь, когда вы знаете, как задавать шрифты, давайте еще раз посмотрим на страницу о путешествии Тони по США и немного преобразуем ее. Мы внесем пару небольших, малозаметных изменений в стиль текста, и в то время как ни одно из них по отдельности не повлияет на внешний вид страницы, мы все же думаем, что к концу главы вы согласитесь с нами, что сайт выглядит совсем по-другому. Сначала подумаем над тем, что было бы неплохо улучшить, а затем добавим на страницу новое семейство шрифтов.



Вспомните, что мы не задавали никаких стилей для сайта Тони, поэтому в нем используется семейство шрифтов serif.

Размер шрифта, используемый по умолчанию для заголовков страницы, слишком большой, что не очень украшает ее.

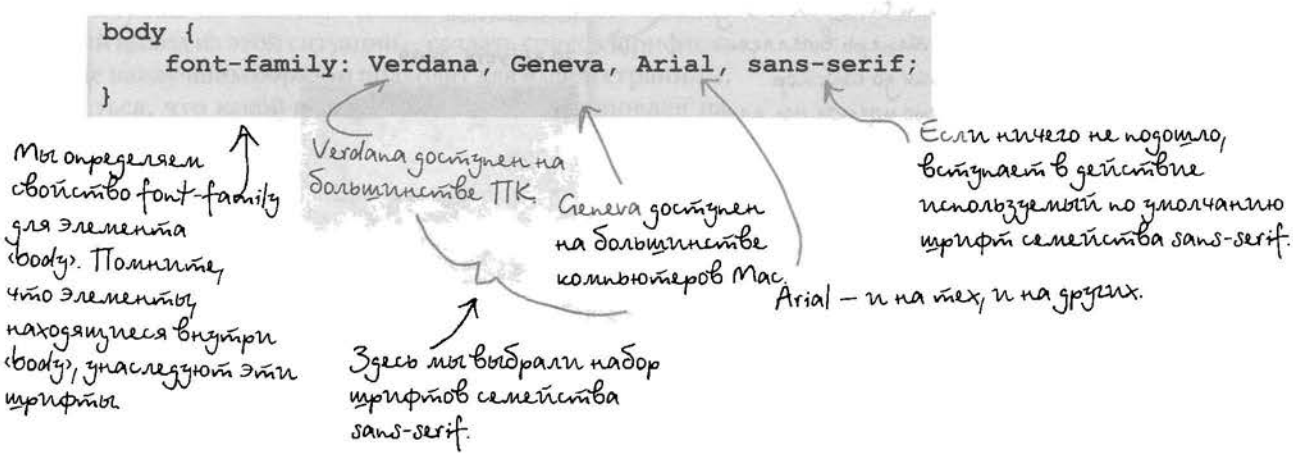
Цитата выделена просто отступом вправо. Было бы здорово немного улучшить ее внешний вид, добавив стиль шрифтам.

Если не учитывать фотографии, эта страница одноцветная, поэтому не помешает добавить шрифтам цвета.

Задаем новое свойство font-family

Давайте поможем Тони с использованием свойства **font-family**. Начнем с некоторых популярных шрифтов семейства sans-serif. Создайте новый файл `journal.css` в папке `chapter9/journal` и добавьте в него это правило:

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



Теперь необходимо создать ссылку с дневника Тони на новый файл с таблицей стилей. Для этого откройте файл `journal.html`, находящийся в папке `chapter9/journal`. Вам нужно добавить в него элемент **<link>**, что позволит сослаться на стили в файле `journal.css`, как мы сделали это ниже.

Помните, в главе 7 вы перевели дневник Тони на строгий XHTML?

```
<!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="ru" xml:lang="ru">
   <head>
      <meta http-equiv="Content-Type" content="text/html, charset=windows-1251" />
      <link type="text/css" rel="stylesheet" href="journal.css" />
      <title>На скутере по США</title>
   </head>
   <body>
     .
     .
     .
   </body>
</html>
```

Здесь мы создаем ссылку на новый файл journal.css.

После того как вы внесете это изменение, сохраните файл и откройте страницу в браузере.

Тестирование новых шрифтов страницы Тони

Откройте в браузере страницу с новым CSS-кодом. Теперь у нас есть достаточно неплохой набор шрифтов семейства sans-serif. Продолжим работу со шрифтами для страницы Тони...

Шрифт определенно меняет внешний вид веб-страницы. Заголовки теперь выглядят более четко, без засечек, хотя до сих пор они казались достаточно громоздкими для этой страницы.

Текст в абзаце также четкий и хорошо читаемый.

Поскольку font-family — наследуемое свойство, теперь шрифт sans-serif используется для всех элементов на странице, даже для этих пунктов списка...

...и для blockquote.



Часто Задаваемые Вопросы

В: Как мне задать шрифт, название которого состоит из нескольких слов, например, Courier New?

О: Объявляя шрифт, заключите такое название в двойные кавычки, например: font-family: "Courier New", Courier;

В: Итак, свойство font-family представляет из себя набор взаимозаменяемых шрифтов?

О: Да. По сути, это список шрифтов, указанных в порядке первоочередности. Первым идет тот шрифт, который вы хотите видеть на странице, за ним — его основной заместитель, затем следующий заместитель и т. д. Что касается последнего шрифта, то здесь нужно указать универсальный шрифт, характерный для конкретного вида: sans-serif или serif, который должен принадлежать тому же семейству, что и остальные шрифты списка.

В: Разве serif и sans-serif — это шрифты?

О: Serif и sans-serif — это не названия реально существующих шрифтов. Тем не менее ваш браузер использует действительно существующий шрифт вместо serif или sans-serif, если все предшествующие шрифты, указанные в font-family, не будут найдены. Вместо них браузер будет применять тот шрифт, который определен им как используемый по умолчанию для указанного семейства шрифтов.

В: Как узнать, какой шрифт лучше использовать? Serif или sans-serif?

О: Строгих правил для этого нет. Тем не менее многие полагают, что для текста на экране компьютера лучше подходит sans-serif. Существует множество дизайнов с использованием шрифтов serif для основного текста или шрифтов serif и sans-serif вперемешку. Итак, дизайн вашей страницы зависит только от вашей фантазии и вкуса.

Как быть, если у разных пользователей установлены различные шрифты?

Стоит упомянуть об одном неприятном моменте, касающемся шрифтов. Вы не можете предусмотреть, какие шрифты будут доступны на машинах ваших пользователей. Самый хороший выход из этой ситуации — создать список шрифтов, которые наилучшим образом подходят для вашей страницы, и надеяться, что какой-нибудь из них будет установлен на машине пользователя. В конце концов, вы сможете рассчитывать на то, что браузер поддержит типовой шрифт из этого семейства.

Это основная стратегия, позволяющая быть уверенным, что на странице отображаются подходящие шрифты. Но, оказывается, учитывая, что в различных операционных системах используются различные шрифты (особенно велика разница между Windows и Mac OS), вам придется еще очень постараться, чтобы действительно хорошо сделать эту работу. Вам нужно будет убедиться, что в списке шрифтов, задаваемом в свойстве **font-family**, с наибольшей степенью вероятности будут указаны шрифты, поддерживаемые как в Windows, так и в Mac (а также в некоторых других системах, с которыми могут работать ваши пользователи, например Linux).

Приводим здесь небольшой путеводитель по самым популярным шрифтам для каждой операционной системы, но советуем вам более подробно изучить этот вопрос, если придется вплотную заниматься шрифтами на ваших страницах.

Andale Mono
Arial
Arial Black
Comic Sans
Courier New
Georgia
Impact
Times New Roman
Trebuchet MS
Verdana

Эти шрифты, скорее всего, будут установлены и в Windows, и в Mac OS.

Geneva
Courier
Helvetica
Times

Эти шрифты будут найдены на компьютерах Macintosh

Давайте еще раз взглянем на определение шрифтов для страницы Тони.

(1) Мы хотим, чтобы был использован шрифт Verdana, но...

(3) В этом нет ничего страшного, так как мы можем рассчитывать, что Arial будет установлен и в Windows, и в Mac, но если уж и он не будет найден...

```
font-family: Verdana, Geneva, Arial, sans-serif;
```

(2) Если он не будет найден, то нас устроит и Geneva, но это возможно, скорее всего, только для Mac OS. Но если и он не будет найден...

(4) Это все равно под контролем, так как мы просто предоставим браузеру самостоятельно выбрать шрифт из семейства sans-serif.

Размеры шрифта

Теперь, когда на странице Тони используется новый набор шрифтов, необходимо поработать с их размерами, так как чаще всего размеры заголовков, устанавливаемые по умолчанию, немного великоваты. Как же задаются размеры шрифтов? Для этого предусмотрено несколько способов. Рассмотрим некоторые из них и затем поговорим о том, какой *больше* подходит для того, чтобы размеры шрифтов были последовательными и удобными для пользователя.

Если вы все сделаете правильно, то каждый пользователь сможет самостоятельно увеличивать размеры шрифта на вашей веб-странице, чтобы ему было комфортнее читать. Совсем скоро вы узнаете, как это сделать.

px

Вы можете задать размер шрифта в пикселах, как мы это делали в главе 5. Когда вы указываете размер шрифта таким образом, вы говорите браузеру, какой высоты должны быть символы шрифта.

```
font-size: 14px;
```

В CSS указывается количество пикселов и сразу за ним символы px. В данном случае мы задаем размер шрифта 14 пикселов в высоту.

Символы px должны следовать сразу же после числа. Не ставьте здесь пробел.

Так определяется свойство font-size для правила body.

```
body {
    font-size: 14px;
}
```

hip } 14 пикселов

Такая высота шрифта означает, что от нижнего до верхнего края символов будет расстояние 14 пикселов.

%

Размер шрифта, заданный в процентах, определяет высоту шрифта *относительно* высоты других шрифтов. Итак,

```
font-size: 150%;
```

означает, что данный шрифт в высоту будет составлять 150 % от другого шрифта. Но *какого* другого шрифта? Учитывая, что **font-size** — это наследуемое свойство, при задании размера таким способом проценты будут вычисляться относительно размера шрифта родительского элемента. Посмотрим, как это работает...

Здесь мы задали размер шрифта для элемента <body> в пикселах, а для заголовков первого уровня установили размер 150%.

```
body {
    font-size: 14px;
}
h1 {
    font-size: 150%;
}
```

em

Вы также можете определить размер шрифта, используя относительную величину em. С помощью em вы задаете масштабный коэффициент. Рассмотрим, как используется эта величина:

Не путайте с элементом «em»!

```
font-size: 1.2em;
```

Это означает, что размер шрифта будет масштабирован с коэффициентом 1,2.

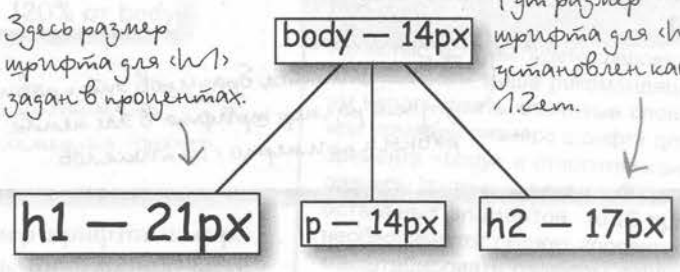
Допустим, вы таким образом задаете размер заголовков **<h2>**. Тогда размер ваших заголовков **<h2>** будет составлять 1,2 размера шрифта родительского элемента, что в нашем случае означает в 1,2 раза больше, чем **14px**, и примерно равняется **17px**.

На самом деле размер будет равняться 16,8, но большинство браузеров округлит его до 17.

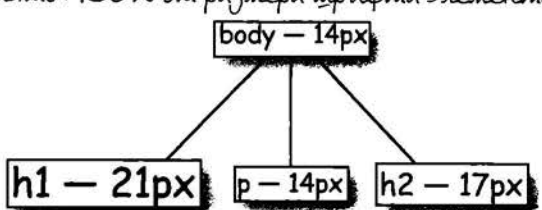
```
body {
      font-size: 14px;
}
h1 {
      font-size: 150%;
}
h2 {
      font-size: 1.2em;
}
```

Здесь размер шрифта для <h1> задан в процентах.

Тут размер шрифта для <h2> установлен как 1.2em.



Если нарисовать небольшое дерево для документа, то станет видно, что <h1> наследует свойства от <body>, поэтому размер его шрифта будет составлять 150 % от размера шрифта элемента <body>.



Размер шрифта заголовков <h1> составляет 150 % от размера шрифта элемента <body>, что равняется 21px.

Поскольку мы не задавали размер шрифта для элемента <p>, то он наследует размер шрифта элемента <body>, равный 14px.

Ключевые слова

Существует еще один способ задавать размеры шрифта: использовать ключевые слова. Вы можете определить размер шрифта как **xx-small**, **x-small**, **small**, **medium**, **large**, **x-large** или **xx-large**, и браузер преобразует эти слова в значения в пикселах.

Обычно размеры шрифтов, которые задаются различными ключевыми словами, соотносятся друг с другом таким образом. Каждый следующий размер примерно на 20% больше предыдущего, а small обычно равен 12 пикселов в высоту. Тем не менее не забывайте, что ключевые слова не всегда одинаково определяются в различных браузерах и пользователи при желании могут их переопределить.

```
body {
      font-size: small;
}
```

В большинстве браузеров это правило задает размер шрифта в элементе body равным примерно 12 пикселов.

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

Итак, как же задается размер шрифта?

Как вы уже поняли, существует большое количество способов задания размера шрифта. Какой же использовать? Придерживайтесь следующих рекомендаций при определении размера шрифта, и это позволит вам быть уверенными, что страница правильно отобразится во всех браузерах.

1. Выберите ключевое слово (мы рекомендуем `small` или `medium`) и укажите его в качестве размера шрифта в правиле для **body**. Таким образом вы зададите размер шрифта, используемый на вашей странице по умолчанию.

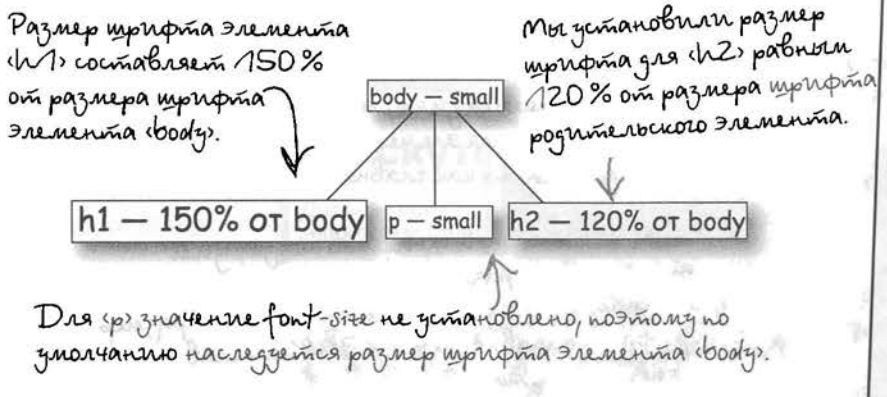
2. Задайте размеры шрифтов для остальных элементов относительно размера шрифта элемента **body**, используя либо проценты, либо em (выбор остается за вами, так как, по сути, это два разных способа выполнения одного и того же действия).

Это неплохие требования, но все же что в них хорошего? Задав размеры шрифтов относительно размера шрифта элемента **body**, вы действительно легко сможете поменять их на всей веб-странице, просто изменив высоту шрифта для **body**. Хотите изменить страницу так, чтобы на ней использовались большие размеры шрифтов? Если значение размера шрифта для **body** равняется **small**, просто поменяйте его на **medium**, и размеры шрифтов для всех остальных элементов автоматически пропорционально увеличатся, потому что они заданы относительно размера шрифта элемента **body**. Кроме того, в таком случае ваши пользователи и сами смогут изменить размер шрифтов на странице.

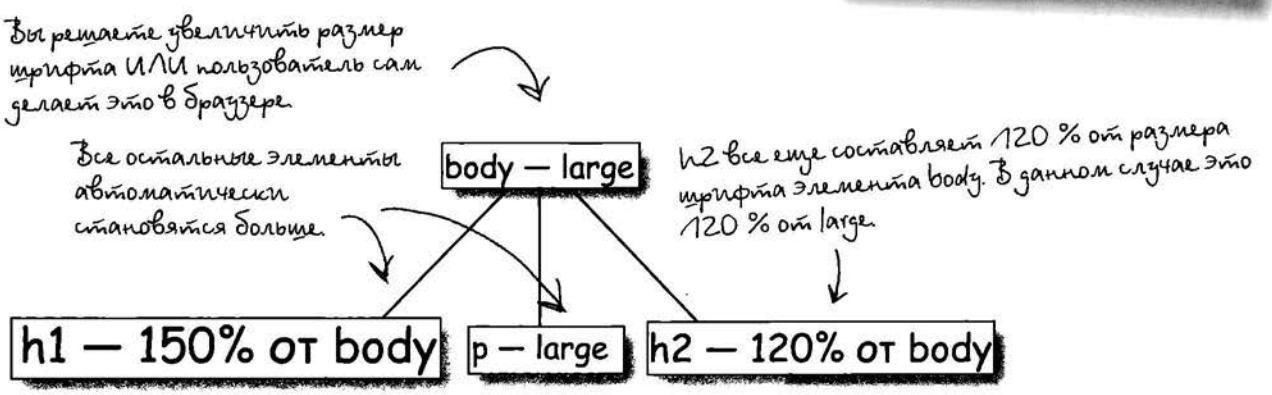
Давайте посмотрим, как это все работает. Во-первых, вы устанавливаете размер шрифта для элемента **<body>**. Затем вы задаете все остальные размеры относительно его, вот так:

```
body { font-size: small; }
h1 { font-size: 150%; }
h2 { font-size: 120%; }
```

Таким образом, вы получаете следующее дерево документа:



Допустим, теперь вы хотите увеличить размер шрифтов на странице или, возможно, это делает пользователь. Тогда вы получаете дерево, подобное этому:



Теперь размер шрифта элемента **body** стал крупнее, и все остальные размеры изменились соответственно. Это великолепно, потому что вам не пришлось менять вручную остальные размеры шрифта – достаточно было изменить размер шрифта элемента **body**. Если же вы пользователь, то вы вообще не должны знать, как это происходит. Когда вы увеличили размер шрифта, весь текст стал еще больше, поскольку все размеры элементов заданы относительно друг друга. В результате страница стала выглядеть еще лучше с увеличенным размером шрифта.



ВНИМАНИЕ! **Internet Explorer НЕ поддерживает масштабирование текста, если размер шрифта задан в пикселах.**

К сожалению, пользователи Internet Explorer не могут менять размеры шрифтов, если те заданы в пикселах. Это одна из причин, по которой не стоит применять такой способ задания размеров шрифтов. Если вы все же используете пикселы, то многие пользователи не смогут работать с вашими страницами в полную силу.

К счастью, если вы будете следовать приведенным выше рекомендациям использовать ключевые слова при указании размера шрифта для элемента <body> и относительные размеры (заданные в em или %) для остальных элементов, то IE при необходимости сможет корректно масштабировать размеры ваших шрифтов.

Поменяем размеры шрифтов для веб-страницы Тони

Настало врямя поработать над размерами шрифтов на веб-странице Тони. Добавьте новые свойства в файл `journal.css`, находящийся в папке `chapter9/journal`. Когда внесете все необходимые изменения, обновите страницу в браузере и посмотрите, как поменялись размеры шрифтов. Если вы не заметите разницы, то ищите ошибки в своем CSS-коде.

```
body {
    font-family: Verdana, Geneva, Arial,
sans-serif;
    font-size: small;
}
h1 {
    font-size: 170%;
}
h2 {
    font-size: 130%;
}
```

Следуя рекомендациям, мы задаем размер шрифта равным small для элемента <body>. Он будет использоваться как главный.

Размеры остальных шрифтов мы зададим относительно размера шрифта элемента <body>. В данном случае для <h1> мы применим размер шрифта, составляющий 170% от главного размера шрифта.

Размер шрифта для элемента <h2> сделаем немного меньшим, чем для элемента <h1>: 130% от размера шрифта элемента <body>.

Возьми в руку карандаш

Каковы будут значения размеров шрифтов, если вы задавали их для **<h1>** и **<h2>** с помощью em, а не процентов?

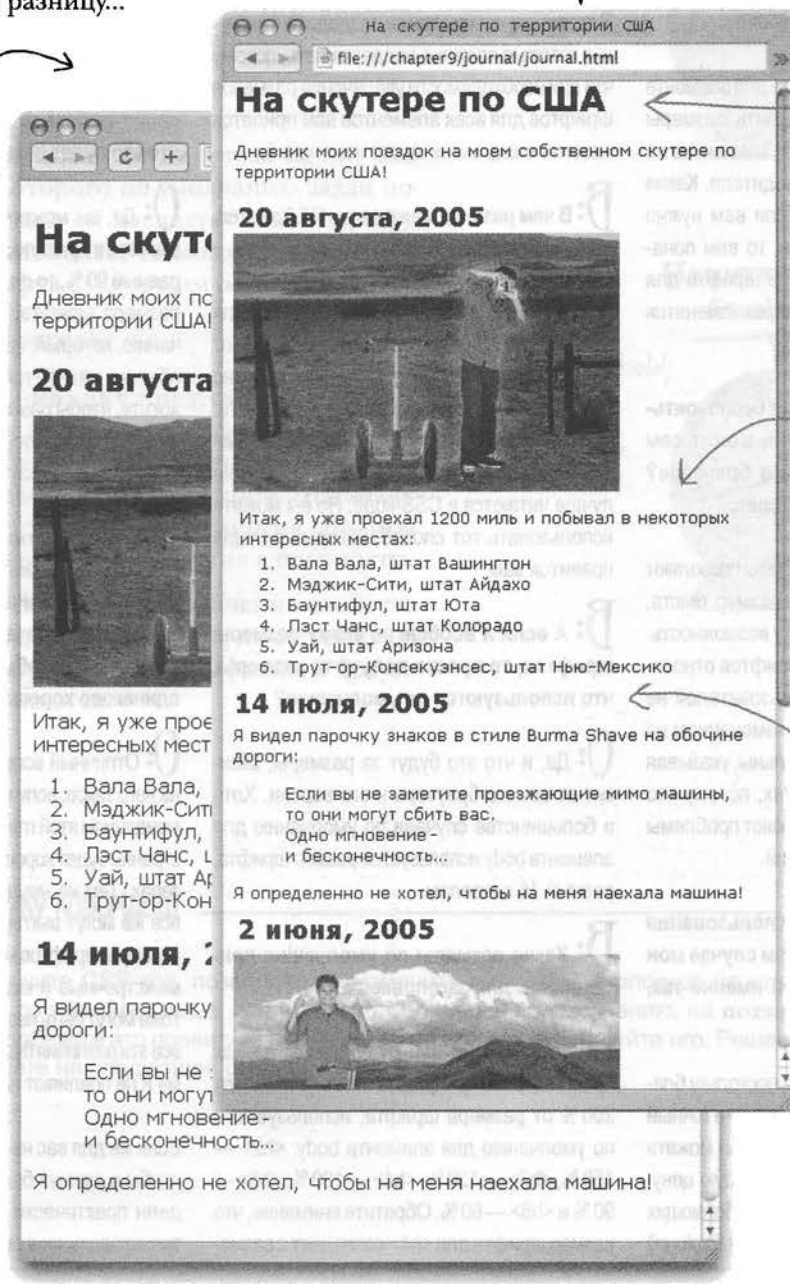
Ответ: **<h1>** будет равняться **1.7em**, а **<h2>** — **1.3em**.

Тестирование страницы с новыми размерами шрифтов

Это улучшенный дневник, в котором теперь используются новые шрифты, меньшие по размеру. Оцените разницу...

Это новая версия, в которой размер шрифтов уже уменьшен. Дизайн страницы становится более приятным для глаз!

Это предыдущая версия (до того как мы поменяли размеры шрифтов).



Теперь заголовок <h1> выглядит намного лучше. Он больше не подавляет своим размером основной текст на странице.

Основной текст стал немного меньше. По умолчанию для него используется размер шрифта, равный 16px, но обычно это зависит от браузера. Однако в размере small (который, скорее всего, равняется 12px) он все еще хорошо читается.

Заголовки <h2> также немного уменьшились, и их размер хорошо сочетается с размером заголовка <h1>.

Часто Задаваемые Вопросы

В: Итак, задавая размер шрифта для элемента <body>, я каким-то образом определяю и все остальные размеры на странице. Как же это работает?

О: Установив размер шрифта для элемента <body>, вы сможете определить размеры шрифтов для всех остальных элементов на странице относительно их родителя. Какие преимущества это дает? Если вам нужно изменить размеры шрифтов, то вам понадобится лишь указать размер шрифта для body, а все остальные размеры изменятся пропорционально.

В: Действительно ли стоит беспокоиться о том, что пользователь может сам поменять размер шрифта в браузере? Лично я никогда этого не делал.

О: Да, стоит. Почти все браузеры позволяют увеличивать или уменьшать размер текста, и многие люди используют эту возможность. Если вы задаете размеры шрифтов относительным способом, то у пользователей не будет никаких проблем с их изменением на странице. Но будьте внимательны, указывая размеры шрифтов в пикселах, потому что у некоторых браузеров возникают проблемы при изменении таких значений.

В: Мне нравится идея использования пикселов, потому что в таком случае мои страницы будут выглядеть именно так, как я хочу.

О: В этом есть доля правды, поскольку благодаря этому способу вы определяете точный размер для каждого элемента. Но вы можете заплатить за это достаточно большую цену, лишив некоторых пользователей (работающих с определенными версиями Internet Explorer) возможности самостоятельно подбирать подходящие для них размеры шрифтов.

В таком случае вы также создадите страницы, которые сложнее поддерживать, потому что при необходимости увеличения размеров шрифтов для всех элементов вам придется вносить в код очень много изменений.

В: В чем разница между em и %? Кажется, они делают одно и то же.

О: По сути, это два разных способа для достижения одной и той же цели. Оба дают возможность задать размер относительно размера шрифта родительского элемента. Многие люди полагают, что с процентами разобраться проще, чем с em, и что они лучше читаются в CSS-коде. Но вы можете использовать тот способ, который больше нравится вам.

В: А если я вообще не укажу размеры шрифтов, то просто получу те размеры, что используются по умолчанию?

О: Да, и что это будут за размеры, зависит от самого браузера и его версии. Хотя в большинстве случаев по умолчанию для элемента body используется размер шрифта, равный 16 пикселам.

В: Какие размеры по умолчанию применяются для заголовков?

О: Это тоже зависит от браузера, но, как правило, размер шрифта <h1> составляет 200 % от размера шрифта, используемого по умолчанию для элемента body, <h2> — 150 %, <h3> — 120 %, <h4> — 100 %, <h5> — 90 % и <h6> — 60 %. Обратите внимание, что размер шрифта для <h4> совпадает с размером, используемым по умолчанию для body, а для <h5> и <h6> он даже меньше.

В: Итак, могу ли я при задании размеров в правиле для body вместо ключевых слов использовать em или %? Если я укажу свойство font-size для body равным 90%, что это будет означать? Это 90 % от чего?

О: Да, вы можете сделать и так. Если вы укажете в правиле для body размер шрифта, равный 90 %, то он будет составлять 90 % от размера шрифта, используемого по умолчанию, который, как мы только что сказали, обычно равняется 16 пикселам. Если вы хотите, чтобы размер шрифта немного отличался от того, который задается ключевыми словами, используйте % или em.

В: Кажется, что между браузерами так много различий: семейства шрифтов, размеры шрифтов, различные настройки, используемые по умолчанию и т. д. Возможно ли сделать так, чтобы мой дизайн смотрелся одинаково хорошо во всех браузерах?

О: Отличный вопрос. Самый простой ответ на него таков: если вы будете следовать рекомендациям этой главы, то большинство ваших страниц будет хорошо выглядеть во всех браузерах. Тем не менее нужно помнить, что они все же могут выглядеть немного по-разному: размеры шрифтов могут немного различаться, межстрочные и межсимвольные интервалы тоже могут быть неодинаковыми и т. д. Однако все эти различия будут совсем незначительными и не повлияют на читаемость страниц.

Если же для вас на самом деле очень важно, чтобы в разных браузерах страницы выглядели практически одинаково, то придется тестировать их в множестве браузеров.

Настройка насыщенности шрифтов

Свойство **font-weight** позволяет управлять тем, насколько жирным выглядит текст. Как вы знаете, жирный текст выглядит темнее и немного толще, чем обычный. Вы можете использовать полужирный текст для любого элемента, задав свойству **font-weight** значение **bold**:

```
font-weight: bold;
```

Однако можете пойти и другим путем. Если у вас есть элемент, для которого по умолчанию задан полужирный шрифт, или он наследует такой шрифт от родительского элемента, то можете уменьшить насыщенность шрифта следующим образом:

```
font-weight: normal;
```

Для свойства **font-weight** предусмотрены также два относительных значения: **bolder** и **lighter**. Они сделают ваш текст более или менее плотным по сравнению с унаследованным значением. Эти значения используются редко, так как немногие шрифты поддерживают столь слабые различия в плотности.

Вы также можете устанавливать значение свойства **font-weight** числами от 100 до 900, но это тоже не очень хорошо поддерживается шрифтами и браузерами и редко применяется.





Возьми в руку карандаш

Напишите CSS-код, позволяющий изменить жирность заголовков на странице Тони со значения **bold**, которое использовалось по умолчанию, на **normal**. Затем добавьте это правило в нужный CSS-файл и протестируйте его. Решение вы найдете на следующей странице.

Заголовки с плотностью normal. Тестирование страницы

Посмотрите, как должен выглядеть CSS-код после того, как вы внесли в него несколько изменений, чтобы задать свойству **font-weight** для заголовков **<h1>** и **<h2>** значение **normal**:

```
body {
      font-family: Verdana, Geneva, Arial, sans-serif;
      font-size: small;
}
h1, h2 {
      font-weight: normal;
}
h1 {
      font-size: 170%;
}
h2 {
      font-size: 130%;
}
```

Здесь мы меняем значение font-weight на normal для обоих заголовков в одном CSS-правиле. Это хорошая идея — объединять одинаковые свойства в одно правило.

А вот результаты. Сейчас заголовки стали тоньше.



Стилизация шрифтов

Вы уже знакомы с *курсивом*, верно? Курсивные символы пишутся под наклоном и иногда имеют дополнительные засечки. Например, сравните эти два стиля:

не курсивный
курсивный

Курсивные символы наклонены вправо и имеют дополнительные засечки.

В CSS можно задать курсивный стиль тексту, используя свойство **font-style**:

```
font-style: italic;
```

Существует распространенная ошибка, когда вместо italic пишут italics. Если вы так сделаете, то ваш текст не станет курсивным, поэтому не забывайте проверять свой код.

Однако не во всех шрифтах предусмотрен курсив, и вместо него можно увидеть лишь *наклонные символы*. Такие символы также наклонены вправо, но при этом браузер не использует специально разработанный набор символов, а просто наклоняет стандартный шрифт. Сравните следующие стили:

не наклонный
наклонный

В наклонном стиле стандартные символы просто наклонены вправо.

Чтобы получить наклонный текст, вы также можете использовать свойство **font-style**, задав для него такое значение:

```
font-style: oblique;
```

Дальше вы поймете, что на практике в зависимости от выбранного шрифта и браузера два стиля могут иногда выглядеть одинаково, а иногда — нет. Итак, если разница между наклонными и курсивными символами для вас не очень велика, выберите один из них и продолжайте работать. Если же она имеет для вас значение, придется протестировать различные сочетания шрифтов и браузеров для достижения лучшего результата.

Курсивный и наклонный — это два стиля, которые устанавливают для шрифта наклон вправо.

Поскольку вы не можете управлять теми шрифтами и браузерами, что используют ваши посетители, вы будете в одних случаях получать курсивный шрифт, а в других — наклонный, независимо от того, какой стиль вы указали.

Таким образом, можете просто задавать курсивный стиль и не обращать внимания на различия (скорее всего, вы все равно не сможете это контролировать).

Стилизация цитаты курсивом на странице Тони

Теперь мы будем использовать свойство **font-style**, чтобы сделать цитаты на странице Тони более привлекательными. Помните слоган Burma Shave, заданный элементом **<blockquote>**? Давайте сделаем этот слоган курсивным, чтобы выделить его на фоне остального текста. Для этого нужно добавить элементу **<blockquote>** свойство **font-style** со значением **italic**:

```
blockquote {
      font-style: italic;
}
```

Добавьте это новое CSS-правило в файл `journal.css`, сохраните его и протестируйте страницу. Вы увидите, что слоган Burma Shave стал курсивным.



Это новый стиль для слогана Burma Shave на странице Тони. Как и было нужно, получился наклонный текст.

Часто Задаваемые Вопросы

В: Текст элемента <blockquote> на самом деле находится внутри элемента <p>, который вложен в элемент <blockquote>. Каким же образом текст абзаца становится курсивным?

О: Вспомните, что большинство элементов по умолчанию наследуют стиль шрифта от родительских элементов, а родительским для <p> является <blockquote>. Поэтому <p>, вложенный в <blockquote>, наследует курсивное начертание.

В: Почему бы просто не поместить текст внутри элемента <blockquote> в элемент ? Разве мы не сделаем таким образом то же самое и текст внутри <blockquote> не станет курсивным?

О: Помните, что применяется для структуризации и говорит о том, что слова должны быть выделены особым образом. Наша же задача — стилизовать <blockquote>, а не указать, что текст внутри <blockquote> должен быть особо выделен. Хотя в чем-то вы правы, так как в большинстве браузеров содержимое становится курсивным, это все же не лучший способ стилизации текста внутри <blockquote>. Кроме того, стиль элемента может быть разным, поэтому вы не сможете рассчитывать на то, что его содержимое всегда будет отображаться курсивом.

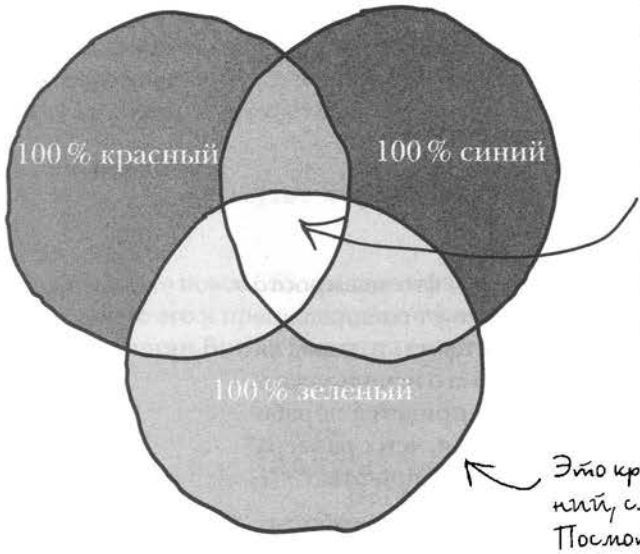


Вы, наверное, думаете, что мы просто скажем вам, что существует специальное свойство для задания цвета, и дадим задание самостоятельно его использовать. Однако с цветами вам придется поработать значительно больше, чем с размерами, плотностью и стилем шрифта.

Итак, далее вы начнете активно работать с цветами и узнаете все, что необходимо для их задания на страницах. В частности, вы узнаете, как цвета отображаются на экране, что это за таинственные шестнадцатеричные коды, стоит ли заботиться о «безопасных» цветах и какой самый простой способ найти и определить цвет, а также ознакомитесь с различными способами задания цветов в CSS.

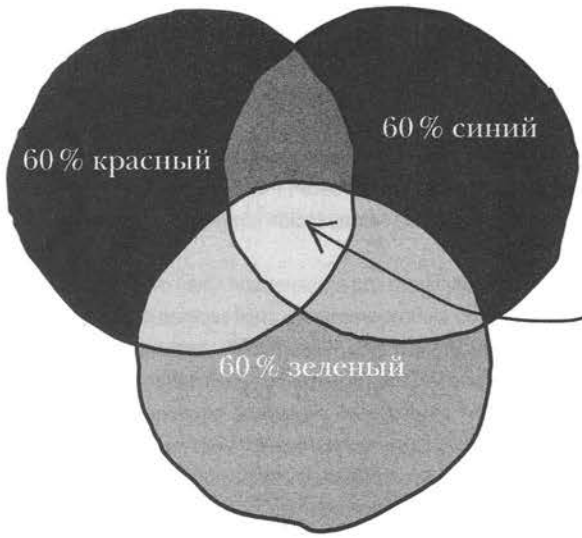
Как работают «безопасные» цвета?

Как вы уже понимаете, на ваших страницах существует множество объектов, для которых можно определить цвет: фон, границы, шрифты. Но как на самом деле работают цвета на компьютере? Давайте посмотрим.



«Безопасный» цвет определяется в зависимости от того, в какой пропорции в нем смешаны красный, зеленый и синий цвета. Вы указываете количество каждого цвета в процентах от 0 до 100 и затем соединяете их все, чтобы получить итоговый цвет. Например, если вы смешаете 100% красного, 100% зеленого и 100% синего вместе, то получите белый. Обратите внимание, что на компьютере при смешивании различных цветов получается более светлый цвет.

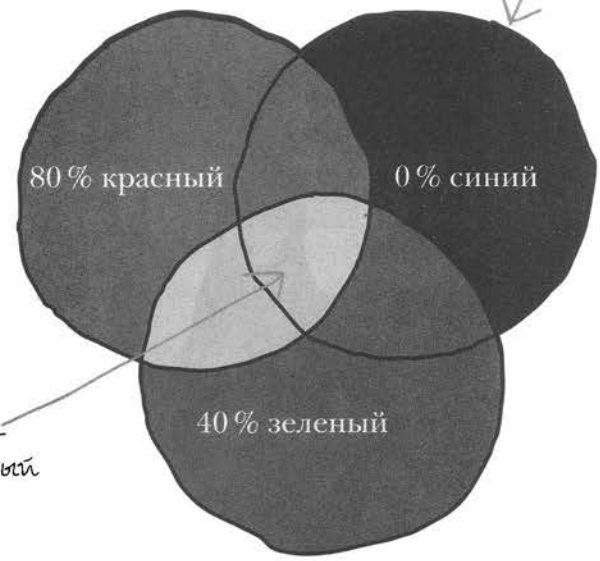
Это красный, зеленый и синий, смешанные друг с другом. Посмотрите в центр, и вы увидите, что получится, если их перемешать.



Если вы добавите, например, только по 60% каждого компонента, то чего ожидать в этом случае? Меньше света, верно? Другими словами, вы получите серый цвет, потому что мы смешиваем три цвета в тех же пропорциях, но с меньшей насыщенностью.

Синий не добавляется в результирующий цвет на экране компьютера, так как задано 0% синего.

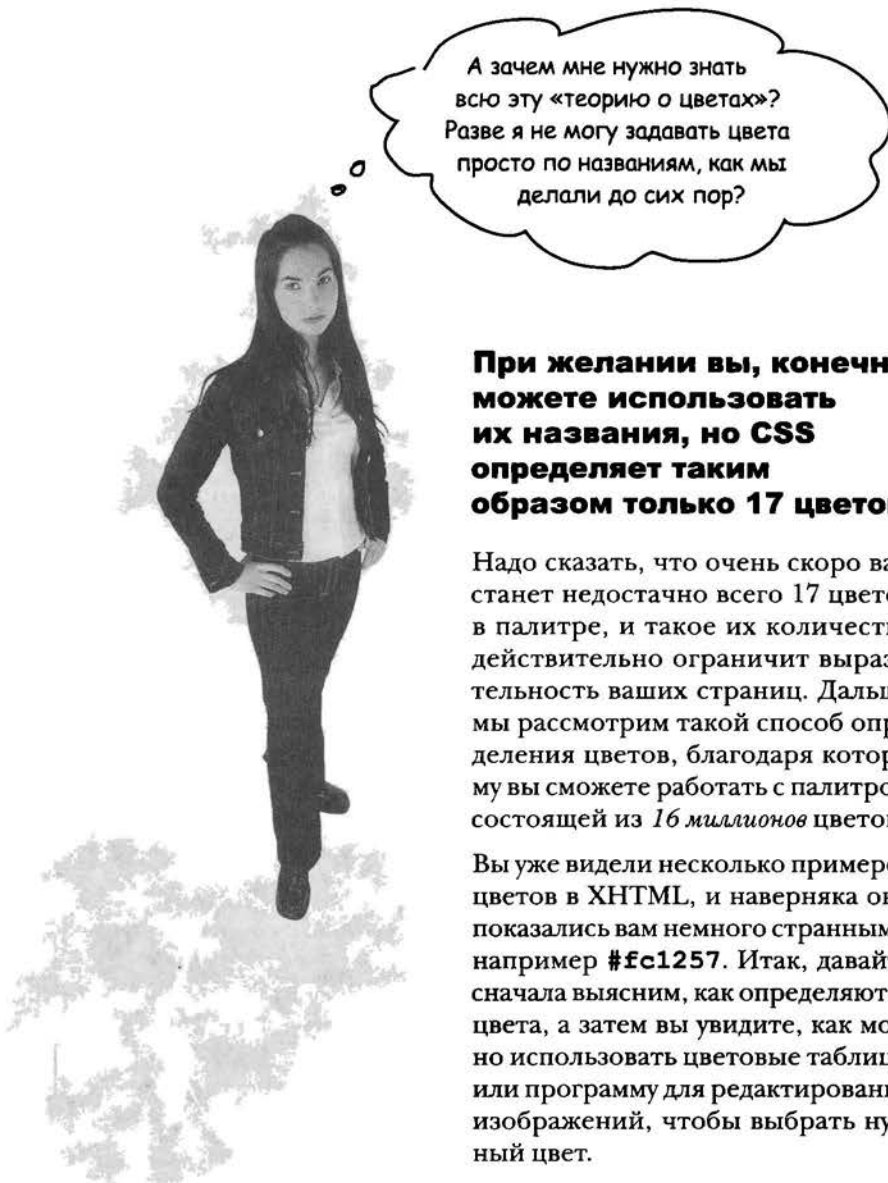
Или, допустим, вы смешиваете 80% красного и 40% зеленого, ожидая увидеть оранжевый цвет. И это именно то, что вы получите. Обратите внимание, что если насыщенность одного цвета равняется нулю, то он не повлияет на два других цвета.



Смешивая 80% красного и 40% зеленого, мы получим приятный оранжевый цвет.

А что получится, если смешать по 0% красного, зеленого и синего? Это означает, что вы вообще не посылаете никакого цвета на экран компьютера, поэтому получите черный.





При желании вы, конечно, можете использовать их названия, но CSS определяет таким образом только 17 цветов.

Надо сказать, что очень скоро вам станет недостаточно всего 17 цветов в палитре, и такое их количество действительно ограничит выразительность ваших страниц. Дальше мы рассмотрим такой способ определения цветов, благодаря которому вы сможете работать с палитрой, состоящей из *16 миллионов* цветов.

Вы уже видели несколько примеров цветов в XHTML, и наверняка они показались вам немного странными, например `#fc1257`. Итак, давайте сначала выясним, как определяются цвета, а затем вы увидите, как можно использовать цветовые таблицы или программу для редактирования изображений, чтобы выбрать нужный цвет.

Как задаются «безопасные» цвета? Рассмотрим разные способы...

В CSS существует несколько способов задания цвета. Вы можете указать его *название*, задать цвет посредством *процентного соотношения красной*, зеленой и синей составляющих или с использованием *шестнадцатеричного кода*.

Возможно, вы думаете, что Сеть должна использовать какой-то один из этих форматов, но они все активно применяются, поэтому будет полезно знать о каждом. Хотя надо отметить, что чаще всего «безопасные» цвета определяют с помощью шестнадцатеричного кода.

Помните, все эти три способа в конечном счете просто позволяют указать браузеру количество красного, зеленого и синего цветов, входящих в определяемый цвет. Рассмотрим каждый метод по очереди.

Определение цвета через его название

Наиболее простой способ описать цвет в CSS — просто использовать его название. Но, как вы уже знаете, существует только 17 цветов, которые могут задаваться таким способом. Предположим, вы хотите задать серебристый цвет для фона элемента **body**. Для этого вам нужно будет написать в CSS следующее:

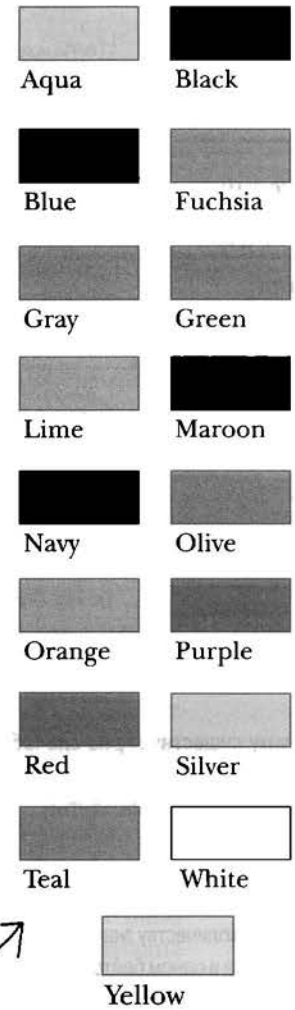
```
body {
    background-color: silver;
}
```

Это правило для body.

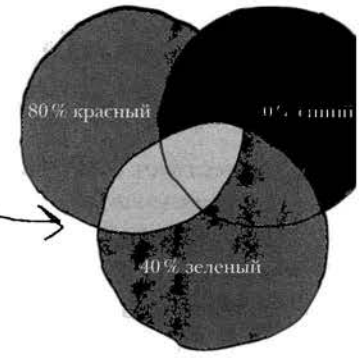
Свойство background-color.

Цвет, заданный названием.

Итак, чтобы задать цвет таким образом, просто укажите его название в качестве значения свойства. При этом не имеет значения, пишете вы названия строчными или прописными буквами, поэтому можете напечатать `silver`, `Silver` или `SILVER`, и все это сработает. Здесь 17 цветов, определенных в CSS. Помните, что это всего лишь названия заданных сочетаний красного, зеленого и синего цветов в определенных пропорциях.



Определение цвета через значение, задаваемое сочетанием красного, зеленого и синего



Можно определить цвет, указав сочетания красного, зеленого и синего в определенной пропорции. Допустим, вам нужно задать оранжевый цвет, который мы рассматривали пару страниц назад. Как вы помните, он состоит из 80 % красного, 40 % зеленого и 0 % синего. Вот как это сделать:

```
body {
    background-color: rgb(80%, 40%, 0%);
}
```

Начинайте с RGB — аббревиатуры от первых букв слов red, green и blue.

Затем в круглых скобках задавайте пропорции красного, зеленого и синего цветов, не забывая ставить символ % после каждого значения.

Можете также задавать значение красного, зеленого и синего числами от 0 до 255. Например, вместо 80 % красного, 40 % зеленого и 0 % синего можно писать 204 красного, 102 зеленого и 0 синего.

Откуда берутся эти числа?
80 % от 255 — это 204,
40 % от 255 — это 102
и 0 % от 255 — это 0.

Посмотрите, как для задания цветов используются обычные числовые значения:

```
body {
    background-color: rgb(204, 102, 0);
}
```

Мы все еще начинаем с RGB.

Чтобы указать значения в числах, а не в процентах, просто напечатайте их без символа % на конце.

Часто Задаваемые Вопросы

В: Почему существует два способа задания RGB-значений? Разве не лучше просто использовать проценты?

О: Иногда лучше, но все же есть логика и в задании чисел от 0 до 255. Такое число соответствует количеству значений, которые могут храниться в одном байте информации. По некоторым историческим и техническим причинам 255 часто принимается за единицу измерения при определении красной, зеленой и синей составляющих цвета. Вы уже, наверное, заметили, что в программах для редактирования изображений часто предлагается определять значения цветов числом от 0 до 255 (если нет, то вскоре вы увидите, как это делается).

В: Я никогда не видел, чтобы кто-нибудь использовал RGB или реальные названия цветов в CSS. Кажется, все используют тип кода для цвета, например #00fc9.

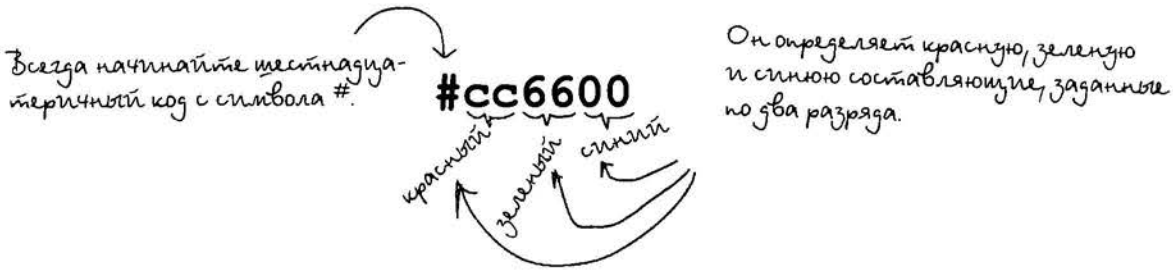
О: Использование RGB-процентов или числовых значений становится все более распространенным методом, но вы правы — шестнадцатеричные коды применяются чаще всего, потому что люди считают их самым удобным способом определения цвета.

В: Важно ли мне как разработчику, посмотрев на запись типа rgb(100, 50, 200), сразу определять, что это за цвет?

О: Не очень. Самый лучший способ узнать, что означает запись rgb(100, 50, 200), — проверить код в браузере или использовать программу для редактирования изображений.

Определение цвета через шестнадцатеричный код

Теперь перейдем к шестнадцатеричным кодам. Откроем вам небольшой секрет: каждый набор двух цифр такого кода представляет красную, зеленую и синюю составляющие цвета. Так, первые две цифры представляют красный цвет, следующие две — зеленый и последние две — синий. Вот так:





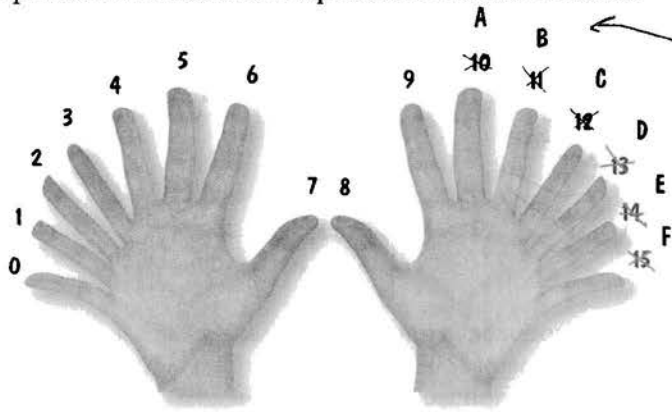
Хотите верьте, хотите нет, но они являются цифрами, просто написаны с использованием буквенных символов.

Хорошо, откроем вам второй секрет к чтению шестнадцатеричных кодов: каждый набор двух цифр представляет номер из 0 до 255. Проблема в том, что если бы мы использовали числа, то могли бы указывать их только до 99, верно? Чтобы не ограничиваться таким количеством цифр, разработчики решили, что могли бы представлять все 255 значений с помощью некоторых букв (от A до F).

Давайте быстро рассмотрим, как все на самом деле работает, и тогда вы поймете, как получить нужный шестнадцатеричный код цвета из таблицы цветов или в приложении для редактирования фотографий.

Двухминутное руководство по использованию шестнадцатеричных кодов

Самое главное, что вы должны знать о шестнадцатеричных кодах, — они основываются не на десяти (от 0 до 9), а на шестнадцати цифрах (от 0 до F). Рассмотрим основные моменты работы с такими числами.



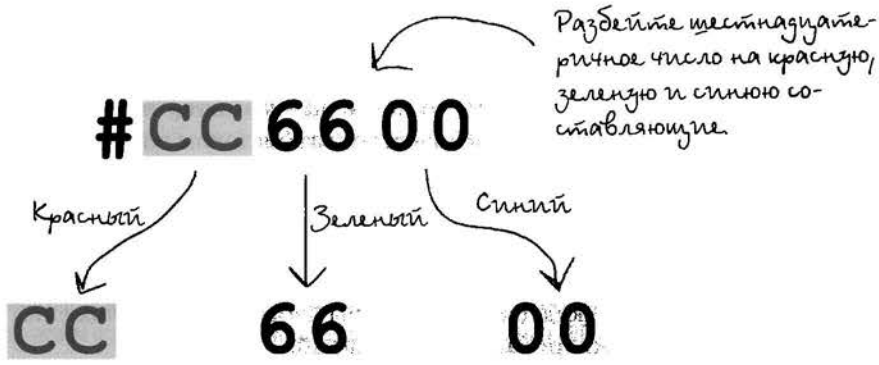
При использовании шестнадцатеричной системы счисления вам достаточно однозначных чисел для счета от 0 до 9. Когда вы достигаете 9, начинайте использовать буквы.

Если вы, к примеру, видите шестнадцатеричное число B, знайте, что имеется в виду 11. Что же обозначают числа BB, E1 или FF? Разберем шестнадцатеричный цвет, чтобы понять, из чего он на самом деле состоит.

Шаг первый

Разделите шестнадцатеричный цвет на три компонента.

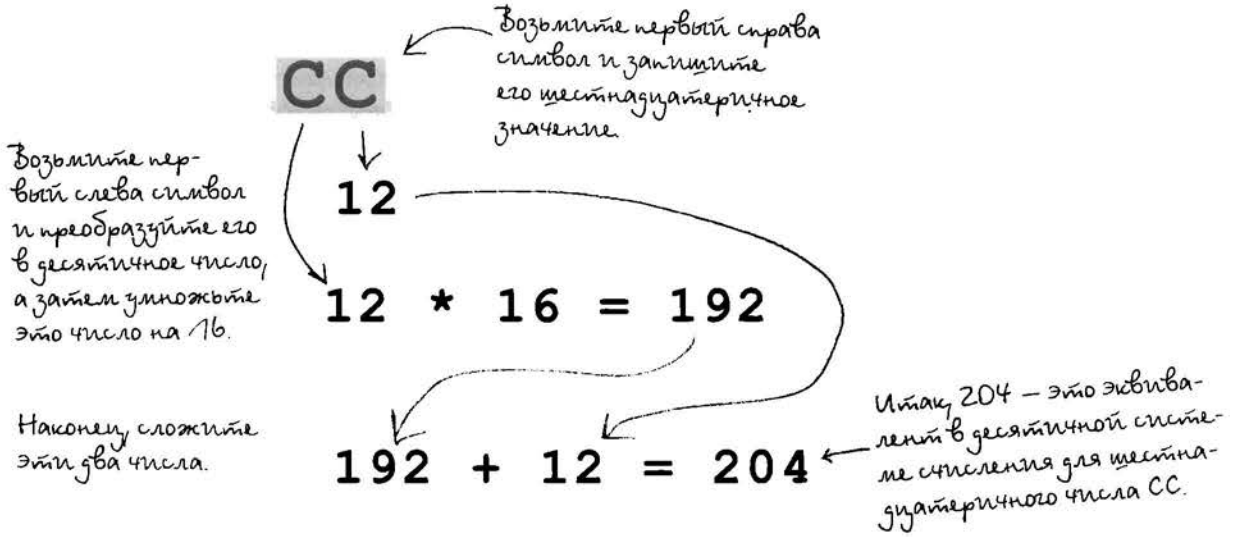
Помните, что каждый шестнадцатеричный цвет состоит из красной, зеленой и синей составляющих. Первым делом вам нужно разделить их.



Шаг второй

Преобразуйте каждое шестнадцатеричное число в десятичное.

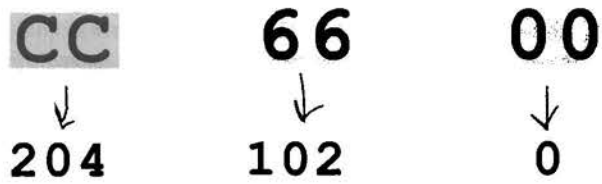
Теперь, когда составляющие разделены, можете преобразовать каждую в десятичное число от 0 до 255. Начнем с красной составляющей:



Шаг третий

Теперь сделайте то же самое для остальных значений.

Повторите такие же действия для двух оставшихся значений. Вот что вы должны получить:



Чтобы преобразовать 66, нужно выполнить такие вычисления: (6 * 16) + 6 = 102.

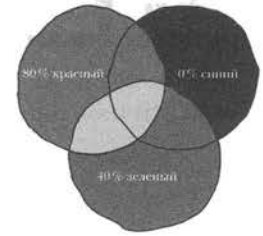
Чтобы преобразовать 00, вы должны выполнить такие вычисления: (0 * 16) + 0 = 0.

Шаг четвертый

А никакого четвертого шага нет, вы уже все сделали!

Ну вот, кажется, и все. Теперь у вас есть числа для каждой составляющей и вы точно знаете, сколько красного, зеленого и синего входит в ваш цвет. Можете таким же способом разобрать любое шестнадцатеричное число.

Объединим все вместе



Теперь вы знаете несколько способов определения цветов. Возьмем, к примеру, все тот же оранжевый цвет, который состоит из 80 % красного, 40 % зеленого и 0 % синего. В CSS можно задать его любым из следующих способов:

```
body {
      background-color: rgb(80%, 40%, 0%);
}

body {
      background-color: rgb(204, 102, 0);
}

body {
      background-color: #cc6600;
}
```

Определение через процентное соотношение красного, зеленого и синего.

Определение через значение, задающее количество красного, зеленого и синего цветов по шкале от 0 до 255.

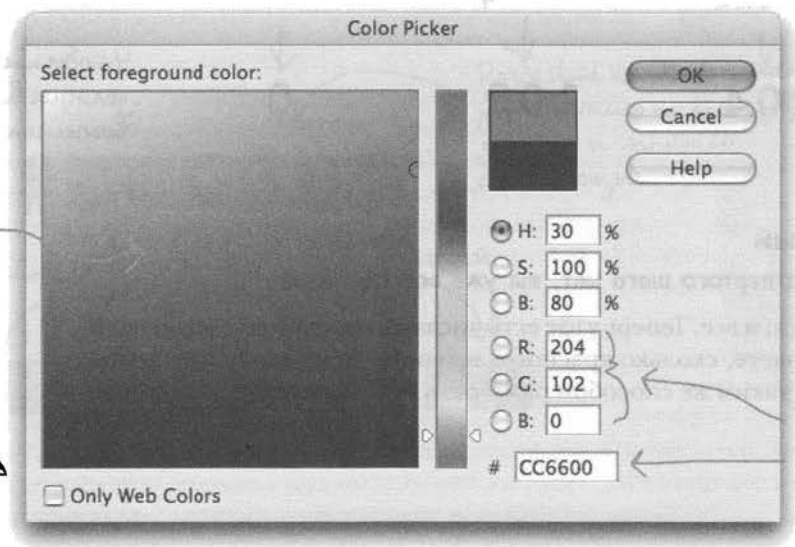
Определение через компактный шестнадцатеричный код.

Где найти «безопасные» цвета

Два самых распространенных способа для поиска «безопасных» цветов — использование таблицы цветов или такой программы, как Photoshop Elements. Кроме того, существует несколько веб-страниц, позволяющих выбрать «безопасный» цвет и преобразовать его RGB в шестнадцатеричный код, и наоборот. Давайте воспользуемся программой Photoshop Elements (большинство программ для редактирования изображений предлагают такую же возможность).

Большинство программ для редактирования изображений содержат цветовую палитру, в которой можно на глаз выбрать любой цвет, используя один или несколько цветовых диапазонов.

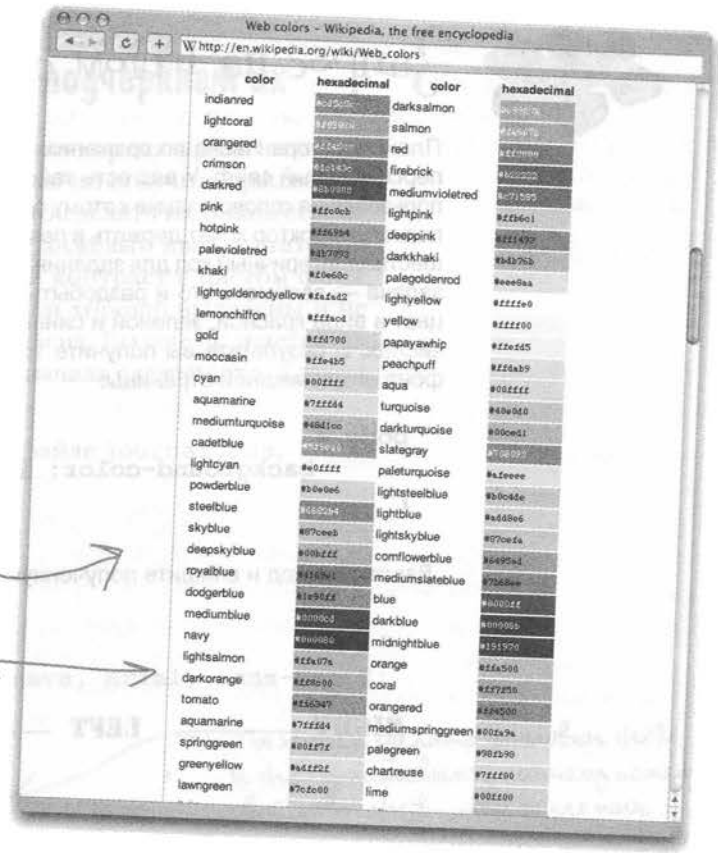
Кроме того, цветовая палитра позволяет выбирать только безопасные цвета. Мы поговорим об этом через минуту.



Как только вы выберете нужный цвет, программа покажет вам его значение в RGB-компонентах, а также его шестнадцатеричный код.

Использование онлайн-таблицы цветов

Вы можете найти несколько полезных таблиц цветов в Сети. Эти таблицы обычно отображают веб-цвета, которые упорядочены по шестнадцатеричному коду. Использовать такие таблицы очень легко, так как вы просто выбираете цвета, которые хотите видеть на своей странице, и копируете их шестнадцатеричный код в свой CSS.



Эта таблица со страницы http://en.wikipedia.org/wiki/Web_colors (русский аналог — http://ru.wikipedia.org/wiki/%D0%A6%D0%B2%D0%B5%D1%82%D0%BE_%D0%B2_Web), представленной в «Википедии». Набрав в поисковике «HTML таблица цветов», вы найдете еще множество других таблиц.

Убедитесь, что для своих цветов используете правильные шестнадцатеричные коды, а не просто эффектные названия, которые, скорее всего, не будут поддерживаться браузерами.

Часто Задаваемые Вопросы

В: Я слышал, что, если не использовать «безопасные» цвета, страницы никогда не будут хорошо отображаться в браузерах. Почему мы более подробно не поговорили о «безопасных» цветах?

О: Давным-давно, когда браузеры только появлялись, мало у кого были компьютеры с экранами, которые могли отображать большое количество тонов, поэтому палитра «безопасных» цветов была создана, чтобы разработчики были уверены, что страницы отображаются одинаково на большинстве экранов.

Сегодня ситуация кардинально изменилась и экраны компьютеров большинства пользователей поддерживают миллионы цветов. Итак, можете быть уверены в том, что «безопасные» цвета — это прошлое, если, конечно, у вас нет знакомых пользователей, у которых экраны отображают ограниченное количество цветов.

В: Теперь я знаю, как определяются цвета. Как же мне выбрать их для шрифтов, чтобы они хорошо смотрелись вместе?

О: Чтобы серьезно ответить на этот вопрос, нужно написать целую книгу. Мы же дадим вам несколько основных рекомендаций по выбору цвета шрифта. Самое главное — использовать контрастные цвета для текста и его фона. Например, черный текст на белом фоне имеет самый высокий контраст. Вам не обязательно все время использовать черный и белый цвета — можете попробовать добавить темный тон какого-либо цвета для самого текста и его светлый оттенок для фона. Некоторые цвета, если использовать их вместе, могут создать странный визуальный эффект (например, синий и оранжевый или красный и зеленый), поэтому сначала тестируйте свои сочетания цветов на друзьях и только потом выставляйте их на всеобщее обозрение.

В: Я видел шестнадцатеричный код #cb0. Что он означает?

О: Если каждая пара цифр состоит из одинаковых символов, вы можете использовать такие сокращения. К примеру, #ccbb00 может быть укорочен до #cb0, а #11eeaa — до #1ea. Однако если ваш шестнадцатеричный код пишется, например, так: #ccbb10, то вы не можете сокращать его запись.



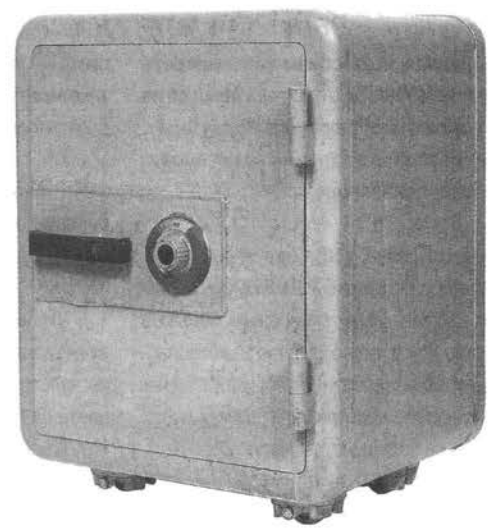
Запрос на взлом сейфа

Планы доктора Ивела по сражению за мировое господство были спрятаны в его персональный сейф. У вас есть тайно полученная информация о том, что он использовал на кодовом замке к этому сейфу шестнадцатеричный код. На самом деле, поскольку доктор хочет держать в памяти это сочетание символов, он использовал шестнадцатеричный код для задания цвета фона своей домашней страницы. Ваша задача — взломать его и раздобыть код к сейфу. Для этого просто представьте цвет в виде красной, зеленой и синей составляющих, выраженных в десятичных числах. В результате вы получите три числа для нужной комбинации. Вот цвет фона его домашней страницы:

```
body {
      background-color: #b817e0;
}
```

Взломайте код и впишите полученную комбинацию сюда:

RIGHT _____ **LEFT** _____ **RIGHT** _____



Вернемся к странице Тони... Сделаем заголовки оранжевыми и подчеркнем их

Теперь, когда вы уже знаете о цветах все, настало время применить эти знания на странице Тони. Он хотел и до сих пор хочет получить оранжевый цвет заголовков. Но вместо того, чтобы просто сделать их оранжевыми, мы добавим к цвету некоторые украшения. Оранжевый цвет достаточно темный и не может обеспечить нам хороший контраст с цветом фона, поэтому, чтобы убедиться в том, что заголовки хорошо выделены и не сливаются с записями дневника, мы подчеркнем их. Вы еще не знаете, как добавить подчеркивание текста, но давайте сначала сделаем это, а затем поговорим о декорировании текста в целом.

Вот все изменения в CSS. Внесите их в своем файле `journal.css`.

```
body {
      font-family: Verdana, Geneva, Arial, sans-serif;
      font-size: small;
}

h1, h2 {
      font-weight: normal;
      color: #cc6600;
      text-decoration: underline;
}

h1 {
      font-size: 170%;
}

h2 {
      font-size: 130%;
}

blockquote {
      font-style: italic;
}
```

Мы хотим сделать заголовки <h1> и <h2> оранжевыми, поэтому помещаем свойство color в правило для них.

Это шестнадцатеричный код для оранжевого цвета, известного еще как rgb(80%, 40%, 0%).

А это способ, которым мы устанавливаем подчеркивание текста. Мы используем свойство text-decoration и задаем ему значение underline

Тестирование оранжевых заголовков на странице Тони

Когда вы внесете соответствующие изменения в файл `journal.css`, добавив свойство **`color`** в правило **`h1, h2`**, обновите веб-страницу и оцените результаты.

Теперь заголовки <h1> и <h2> оранжевые. Это хорошо сочетается с оранжевой темой и рубашкой Тони.



В добавок к этому заголовки подчеркнуты. Хмм... мы думали, что это хороший способ выделения, но наши заголовки стали похожи на ссылки, на которых можно щелкнуть, а люди привыкли думать, что можно щелкнуть на всем, что подчеркнуто в Сети.

Итак, использовать подчеркивание было плохой идеей. Давайте взглянем на другие виды декорирования текста, а затем пересмотрим эти подчеркивания на веб-странице.

Возьми в руку карандаш

Что общего имеют все эти цвета? Попробуйте задать каждый из них на веб-странице, например, в качестве цвета фона. Можете также использовать палитру цветов из вашей программы для редактирования изображений, вводя шестнадцатеричный код непосредственно в окно палитры.

#111111	**#444444**	**#777777**	**#aaaaaa**	**#dddddd**
#222222	**#555555**	**#888888**	**#bbbbbb**	**#eeeeee**
#333333	**#666666**	**#999999**	**#cccccc**	

Все, что вы хотели знать о декорировании текста

Декорирование позволяет добавлять такие элементы оформления текста, как подчеркивание, надчеркивание, зачеркивание (также известное как перечеркивание) и – в некоторых браузерах – мерцающий текст. Чтобы декорировать текст, просто установите для элемента свойство **text-decoration**, вот так:

```
em {
      text-decoration: line-through;
}
```

В результате применения этого правила посередине текста элемента будет проведена линия.

Вы можете задавать сразу несколько элементов декорирования. Допустим, вы хотите одновременно применить и подчеркивание, и надчеркивание, тогда задайте оформление текста следующим образом:

```
em {
      text-decoration: underline overline;
}
```

В результате применения этого правила к тексту элемента будет применено и подчеркивание, и надчеркивание.

Если же ваш текст наследует какой-либо вид оформления, а вам это не нужно, просто используйте значение **none**:

```
em {
      text-decoration: none;
}
```

В результате применения этого правила все виды декорирования будут удалены из текста элемента .

Часто Задаваемые Вопросы

В: Итак, если у меня задано два разных правила для элемента и одно из них устанавливает подчеркивание, а другое — надчеркивание, будут ли они объединены так, что я получу оба эффекта?

О: Нет. Вам нужно объединить два этих значения в одно правило, чтобы получить оба вида оформления. Для декорирования текста нужно только одно правило, а эффекты оформления, заданные в разных правилах, не объединяются. Только то правило, которое предназначено для декорирования текста, определит используемые виды оформления, поэтому единственный способ получить сразу два эффекта оформления — задать их в одном правиле.

В: Я давно хотел спросить, почему свойство color не называется text-color?

О: Это свойство на самом деле задает цвет переднего плана элемента, то есть цвет текста и его границы, хотя вы можете определить для границы отдельный цвет, используя свойство border-color.

В: Мне нравится эффект оформления, который называется «перечеркивание». Могу ли я использовать его для редактируемого текста, чтобы обозначить то, что должно быть удалено?

О: Да, можете, но существует способ лучше. В XHTML есть элемент, который называется <del>. Он помечает то содержимое вашего XHTML-документа, которое должно быть удалено. Есть еще похожий элемент <ins>, помечающий те части содержимого, которые должны быть добавлены. Обычно браузеры стилизуют эти элементы перечеркиванием и подчеркиванием соответственно. Используя же CSS, вы можете стилизовать их так, как захотите. Благодаря <del> и <ins> вместе со стилем элемента вы зададите его смысловое значение.

В: А когда используется такой эффект, как мерцание?

О: Мерцание пришло к нам со времен старого стиля Netscape. Браузеры не всегда его поддерживают, а большинство разработчиков и вовсе полагают, что использование мерцания — признак плохого вкуса.

Удаление подчеркивания

Давайте избавимся от этого сбивающего с толку подчеркивания, а вместо него добавим изящную нижнюю границу, как мы сделали в странице гостевой. Для этого откройте файл `journal.css` и внесите нижеприведенные изменения в совместное правило **`h1, h2`**:

```
h1, h2 {
      font-weight: normal;
      color: #cc6600;
      border-bottom: thin dotted #888888;
      text-decoration: underline;
}
```

Добавьте нижнюю границу для элементов ‹h1› и ‹h2›. Выражение thin dotted переводится как «тонкий пунктирный». В данном случае мы добавляем тонкую пунктирную линию цветом #888888 по нижней границе элемента.

В следующей главе мы подробно рассмотрим границы. Подождите немного, это будет уже совсем скоро!

Уберите оформление текста.

Вот как выглядит наше новое «подчеркивание». Оно более стильное и меньше сбивает с толку, чем эффект оформления «подчеркивание текста».



Теперь у нас под элементами ‹h1› и ‹h2› задана нижняя граница, а не подчеркивание.

Обратите внимание, что границы отображаются во всю ширину страницы, а не только под самим текстом. Почему? Вы узнаете это в следующей главе.



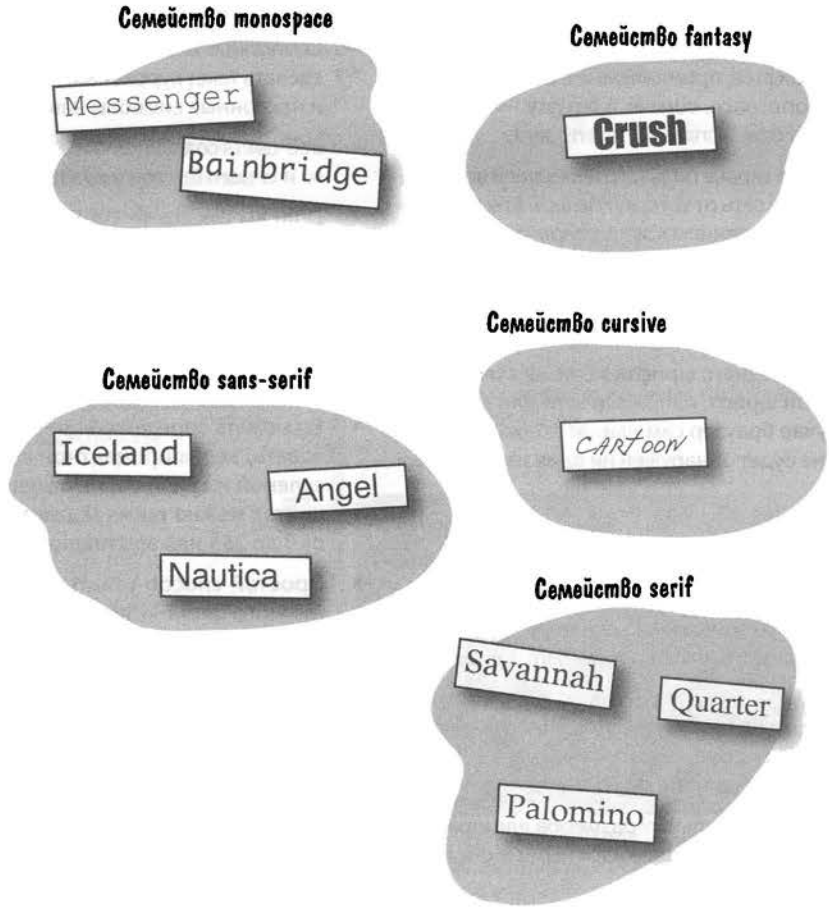
ПОВТОРИМ выученное

- CSS предоставляет множество способов контролировать вид задаваемых шрифтов, используя такие свойства, как **font-family**, **font-weight**, **font-size** и **font-style**.
- Семейство шрифтов — это набор шрифтов с общими характеристиками.
- Семейства шрифтов, применяемые в Сети, — это serif, sans-serif, monospace, cursive и fantasy. Чаще всего используются семейства serif и sans-serif.
- Тип шрифтов на экране пользователя вашей веб-страницы будет зависеть от того, какие шрифты установлены на его собственном компьютере.
- В CSS-свойстве **font-family** полезно указывать несколько шрифтов. Это пригодится, если у ваших пользователей не установлен шрифт, который вы задали.
- В качестве последнего шрифта в списке всегда задавайте типовой шрифт, например serif или sans-serif. В таком случае браузер сам найдет подходящую замену, если не будет обнаружен ни один из указанных шрифтов.
- Размеры шрифтов обычно задаются через пикселы, em, % или ключевые слова.
- Если для определения размера шрифта вы используете пикселы, то тем самым вы говорите браузеру, сколько пикселов в высоту должны быть буквы этого шрифта.
- Em и % — это относительные способы задания размеров шрифтов, то есть размер букв будет зависеть от размера шрифта родительского элемента.
- Применение относительных размеров для шрифтов делает ваши страницы более удобными в обслуживании.
- Используйте ключевые слова, чтобы задать основной размер шрифта в правиле для **body**. В таком случае все браузеры смогут масштабировать размеры текста, если пользователи захотят, чтобы шрифт был больше или меньше.
- Вы можете сделать текст полужирным, используя CSS-свойство **font-weight**.
- Свойство **font-style** применяется для того, чтобы сделать текст курсивным или наклонным. Курсивные и наклонные символы наклонены вправо.
- Веб-цвета создаются смешиванием красного, зеленого и синего цветов в различных количествах.
- Если вы смешаете 100% красного, 100% зеленого и 100% синего, то получите белый цвет.
- Смешав 0% красного, 0% зеленого и 0% синего, вы получите черный цвет.
- В CSS существует 17 предопределенных цветов, включая черный, белый, красный, синий и зеленый.
- Вы можете определить цвет, который хотите использовать, задав процентное соотношение красной, зеленой и синей составляющих в нем. Вместо процентов можно также задавать численные значения от 0 до 255 или шестнадцатеричные коды.
- Простой способ узнать шестнадцатеричный код нужного цвета — использовать цветовую палитру из программы для редактирования изображений или одну из множества таблиц цветов в Интернете.
- Шестнадцатеричные коды состоят из шести цифр, и каждая цифра берется из интервала 0–F. Первые две цифры задают количество красного, вторые — зеленого, а последние — синего цвета.
- Чтобы подчеркнуть текст, можете использовать свойство **underline**. Подчеркнутый текст пользователи часто путают со ссылками, поэтому применяйте его осторожно.



Магниты для разметки

Ваша задача — помочь вымышленным шрифтам найти дорогу домой, к их родным семействам. Переместите каждый магнит для разметки, расположенный слева, в правильное семейство шрифтов, расположенное справа. Перед тем как перейти к следующему разделу, проверьте свои ответы. А вот и решение.





Взлом сейфа. Решение

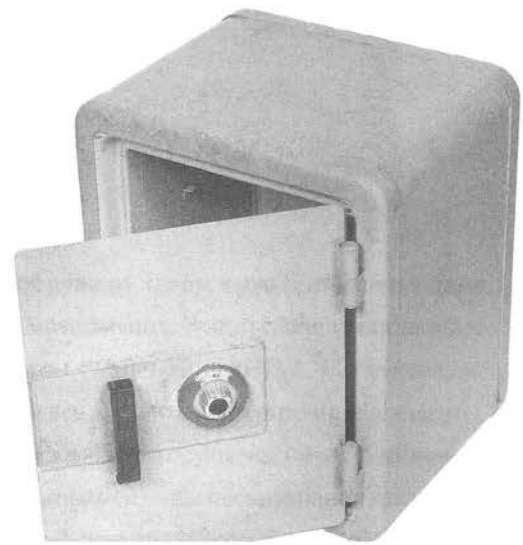
Планы доктора Ивела по завоеванию мирового господства были спрятаны в его персональный сейф. У вас есть тайно полученная информация о том, что он использовал на кодовом замке к этому сейфу шестнадцатеричный код. На самом деле, поскольку доктор хочет держать в памяти это сочетание символов, он использовал шестнадцатеричный код для задания цвета фона своей домашней страницы. Ваша задача — взломать его и раздобыть код к сейфу. Для этого просто представьте цвет в виде красной, зеленой и синей составляющих, выраженных в десятичных числах. В результате вы получите три числа для нужной комбинации. Вот цвет фона его домашней страницы:

```
body {
      background-color: #b817e0;
}
```

Взломайте код и впишите полученную комбинацию сюда:

(11 * 16) + 8 = (1 * 16) + 7 = (14 * 16) + 0 =

RIGHT 184 **LEFT** 23 **RIGHT** 224



Возьми в руку карандаш

Решение

Что общего имеют все эти цвета? Попробуйте задать каждый из них на веб-странице, например, в качестве цвета фона. Можете также использовать палитру цветов из вашей программы для редактирования изображений, вводя шестнадцатеричный код непосредственно в окно палитры.

#111111
#222222
#333333
#444444
#555555
#666666
#777777
#888888
#999999
#aaaaaa
#bbbbbb
#cccccc
#dddddd
#eeeeee



Все цвета, в шестнадцатеричных кодах которых используется только одна цифра, — это оттенки серого: от очень темного (почти черного) до очень светлого (почти белого).

Познакомимся с элементами поближе



Чтобы создавать современные веб-сооружения, вам нужно на самом деле хорошо разбираться в строительных материалах. В этой главе мы подробно рассмотрим наши строительные материалы — элементы XHTML. Мы буквально под микроскопом изучим, из чего сделаны все эти блочные и строчные элементы. Вы узнаете, как можно управлять практически всеми возможностями оформления элементов в CSS. Однако на этом мы не остановимся — вы также узнаете, как можно присваивать элементам уникальные идентификаторы. И если этого будет недостаточно, вы выучите, в каком случае и как использовать несколько таблиц стилей. Итак, переворачивайте страницу и познакомьтесь с элементами поближе.

Модернизация гостевой

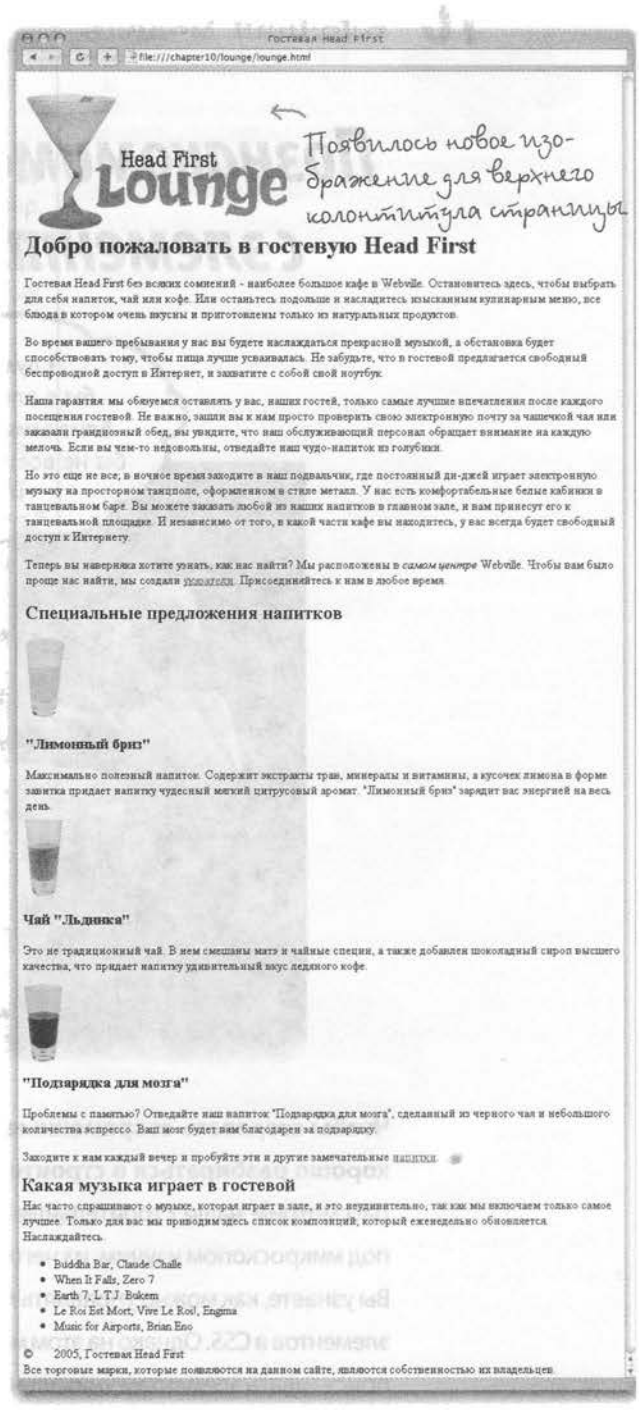
Вы прошли долгий путь из девяти глав и уже создали прекрасную гостевую Head First. В двух следующих главах мы полностью модернизируем ее, добавив абсолютно новое содержимое на главную страницу, и заново стилизуем ее. Чтобы уже сейчас немного вас заинтересовать, мы вкратце расскажем о том, что получится в результате. Итак, посмотрите — на этой странице вы найдете новую нестилизованную гостевую страницу с новым содержимым на ней, а на следующей странице — ее же стилизованную версию, которую мы создадим к концу этой главы.

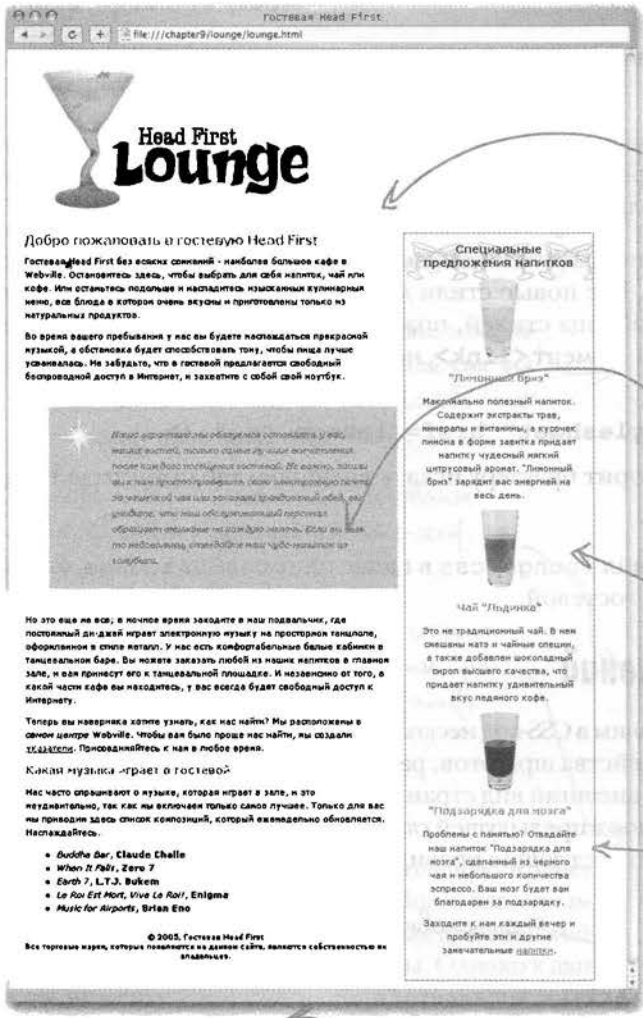
Добавлено много нового текста для описания самой гостевой и того, что в ней предлагается.

Включена информация о специальных напитках.

Кроме того, посетители могут прослушивать кое-какую музыку, которая играет в гостевой каждую неделю.

Наконец, у страницы появился особый стиль, охраняемый авторским правом, что указано в нижнем колонтитуле страницы.





Наши заголовки стали по цвету сочетаться с основной темой сайта — аквамариновой. Кроме того, используются хорошо читаемые шрифты семейства sans-serif.

Этот абзац очень хорошо стилизован, что помогает выгодно выделить его на фоне остального текста и придает странице привлекательный внешний вид. Кажется, что тут еще используется шрифт serif, отличающийся от шрифта остального текста.

Значительно изменен стиль изображения напитков, и сейчас рисунки стали вызывать непреодолимое желание попробовать эти напитки.

Кроме того, напитки были отодвинуты в сторону. Как это произошло?

Стилизован и отдел со списком музыкальных композиций.

Кроме того, нижний колонтитул выровнен по центру, а в нем используется очень мелкий шрифт.

Новая и суперстильная гостевая

Все не так уж и страшно. Теперь, возможно, вам кажется, что гостевая стала «суперстильной», но не забывайте, что это гостевая и она должна быть такой. Вам наверняка еще кажется, будто дизайн становится слишком замысловатым, но только подумайте, что можно сделать с вашими страницами, используя одни и те же технологии. После прочтения этой и следующей глав вам будет очень просто создавать подобные дизайны.

Подготовка к работе с новой гостевой

Перед тем как начать вносить глобальные изменения, ознакомимся с новой гостевой. Вам нужно будет сделать следующее.

1. Откройте папку `chapter10/lounge` и найдите в ней файл `lounge.html` с новым содержимым. Откройте этот файл в текстовом редакторе и просмотрите его. Все должно быть вам понятно: основная часть, абзацы, несколько изображений, блочная цитата и список.

2. В этой главе большую часть времени вы потратите на добавление стиля в XHTML-код, поэтому вам понадобится место для CSS-кода. Все новые стили для гостевой вы будете создавать в файле `lounge.css`, где содержится таблица стилей, поэтому вы увидите, что в элементе **<head>** файла `lounge.html` все еще есть элемент **<link>**, но больше нет предыдущей версии таблицы стилей `lounge.css`.

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

   Помните, что этот элемент **<link>** говорит браузеру искать внешнюю таблицу стилей под названием `lounge.css`.

3. Затем вам нужно будет создать новый файл `lounge.css` в папке `chapter10/lounge`. В нем будет храниться весь новый CSS-код для гостевой.

Начнем с нескольких простых изменений

Теперь вы готовы оформлять гостевую. Сначала добавим в CSS-код несколько правил, чтобы определить все самое основное: семейства шрифтов, размеры и кое-какие цвета – все то, что сразу же улучшит внешний вид страницы (к тому же это хороший шанс повторить все изученное в предыдущей главе). Итак, откройте файл `lounge.css` и добавьте в него следующие правила.

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

h1, h2 {
      color: #007e7e;
}

h1 {
      font-size: 150%;
}

h2 {
      font-size: 130%;
}
```

Это основной размер шрифта страницы

Мы будем придерживаться в гостевой семейства sans-serif. Мы выбрали несколько альтернативных шрифтов, а в конце указали типовой шрифт sans-serif.

Для заголовков <h1> и <h2> зададим аквамариновый цвет, чтобы они сочетались с докалом на логотипе.

Теперь определим подходящие размеры для заголовков <h1> и <h2>. Поскольку мы задаем для них два разных размера, то нужно разделить правила.

Очень простой тест

Давайте быстро протестируем страницу, чтобы посмотреть, как все эти стили на ней отобразились. Убедитесь, что внесли все изменения, затем сохраните файл и тестируйте.

Сейчас заголовки написаны шрифтом sans-serif и имеют цвет, подходящий к логотипу, что создает особую цветовую тему страницы.

Для текста в абзацах тоже используется шрифт sans-serif, так как все элементы наследуют свойство font-family от элемента <body>.

Заголовки <h2> также стилизованы новым цветом и шрифтом sans-serif, но они немного меньше.

Мы не задали никаких правил стиля для элементов <h3>, поэтому они просто наследуют свойство font-family от элемента <body>.



Голубой цвет ссылки, установленный по умолчанию, выглядит немного неуместно. Нужно будет исправить это, что мы сделаем чуть позже.

Еще одна правка

Внесем в гостевую еще одну небольшую правку, прежде чем перейти к некоторым более значимым изменениям. Эта правка затрагивает новое свойство, с которым вы пока незнакомы. Однако к данному моменту у вас уже накопилось достаточно опыта, чтобы мы могли не объяснять слишком тщательно каждое новое свойство. Итак, просто берите и тестируйте его.

Отрегулируем высоту строки текста на всей странице, чтобы между каждыми двумя строками был интервал побольше. Для этого в правило **body** добавим свойство **line-height**:

Увеличение межстрочного интервала может улучшить читаемость текста. Кроме того, это позволяет обеспечить визуальное разделение текста на различные части (как это работает, вы увидите очень скоро).

Здесь мы меняем межстрочный интервал на 1.6em, то есть он будет в 1,6 раза больше высоты шрифта.

```
body {
      font-size:   small;
      font-family: Verdana, Helvetica, Arial, sans-serif;
      line-height: 1.6em;
```

Поработаем с межстрочными интервалами

Как вы, скорее всего, уже догадались, свойство **line-height** позволяет задать межстрочный интервал для вашего текста. Как и в других свойствах для стилизации шрифтов, в нем вы можете задавать интервал в пикселах или через значение, выраженное в em или процентах по отношению к размеру самого шрифта.

Посмотрим, как свойство **line-height** повлияло на внешний вид гостевой. Убедитесь, что добавили это свойство в свой CSS-файл, и сохраните изменения. После обновления страницы в браузере вы увидите, что межстрочный интервал в тексте увеличился.



С помощью свойства line-height мы увеличили межстрочные интервалы во всем тексте с того, который был установлен по умолчанию, до 1.6em.

Do

Во время вашего пребывания у нас вы будете наслаждаться прекрасной музыкой, а обстановка будет способствовать тому, чтобы пища лучше усваивалась. Не забудьте, что в гостевой предлагается свободный беспроводной доступ в Интернет, и захватите с собой свой ноутбук.

Во время вашего пребывания у нас вы будете наслаждаться прекрасной музыкой, а обстановка будет способствовать тому, чтобы пища лучше усваивалась. Не забудьте, что в гостевой предлагается свободный беспроводной доступ в Интернет, и захватите с собой свой ноутбук.

После

В издательском деле межстрочный интервал еще называют «интерлинья-жем».

Свойство line-height наследуется, поэтому, задав его для элемента body, вы «дадите команду» всем элементам страницы тоже использовать межстрочный интервал 1.6em.



Упражнение

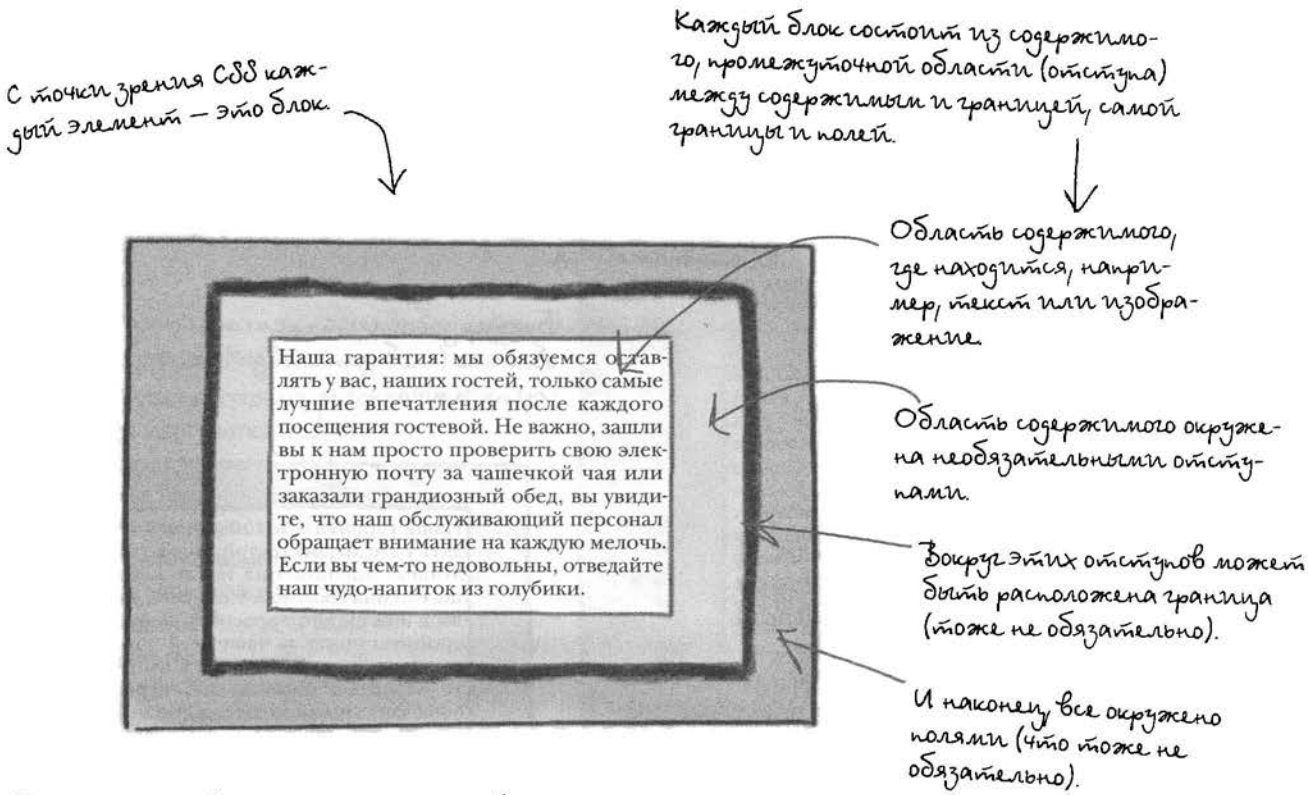
Попробуйте задать различные значения для свойства **line-height**, например `200%`, `.5em` и `20px`, и посмотрите, как это повлияет на межстрочный интервал. Что выглядит лучше, а что хуже? Что лучше всего читается? Когда закончите с этим, убедитесь, что вернули свойству **line-height** значение `1.6em`.

Подготовка к главной реконструкции

Спустя всего четыре страницы этой главы текст гостевой уже достаточно хорошо стилизован. Поздравляем!

Сейчас все станет действительно интересно, так как мы перейдем от изменения таких простых свойств элементов, как размер, цвет и декоративные эффекты, к настоящему художественному и техническому оформлению, затрагивающему основные характеристики отображения элементов. Именно на этом этапе вы переходите в главную лигу.

Однако прежде, чем это случится, вам необходимо узнать, что такое *блочная модель*. Если говорить просто, то блочная модель – это образ вашей страницы со всеми элементами, как ее «видит» CSS. В данном случае каждый отдельный элемент воспринимается как блок. Посмотрим, что это означает.



Все элементы воспринимаются как блоки: абзацы, заголовки, блочные цитаты, списки, элементы списков и т. д. Даже строчные элементы, например и <link>, воспринимаются CSS как блоки.

Рассмотрим блочную модель более подробно

Вскоре вы научитесь задавать с помощью CSS каждую составляющую блока: размер промежуточной области вокруг содержимого, наличие границы у элемента (а также ее величину и стиль), величину отступов до соседних элементов. Сейчас мы разберемся с каждой частью блока и ее назначением.

Что такое область содержимого?

У каждого элемента есть определенное содержимое, например текст или изображение, которое расположено внутри блока, имеющего достаточный размер, чтобы вместить его. Обратите внимание, что между содержимым и границей блока, в котором оно находится, нет промежутка.

Область содержимого обычно имеет такой размер, чтобы в ней помещалось лишь основное содержимое элемента.

Наша гарантия: мы обязуемся оставлять у вас, наших гостей, только самые лучшие впечатления после каждого посещения гостевой. Не важно, зашли вы к нам просто проверить свою электронную почту за чашечкой чая или заказали грандиозный обед, вы увидите, что наш обслуживающий персонал обращает внимание на каждую мелочь. Если вы чем-то недовольны, отведайте наш чудо-напиток из голубики.

Мы нарисовали контур вокруг области содержимого лишь для того, чтобы вы знали ее размеры. Однако в браузере этот контур никогда не отображается.

Что такое отступ?

В любом блоке между содержимым и границей может быть промежуточная область. Вам не обязательно ее задавать, но можете ее использовать, чтобы создать свободное пространство между содержимым и границей элемента. Эта область прозрачна и не может иметь цвет или другие эффекты оформления.

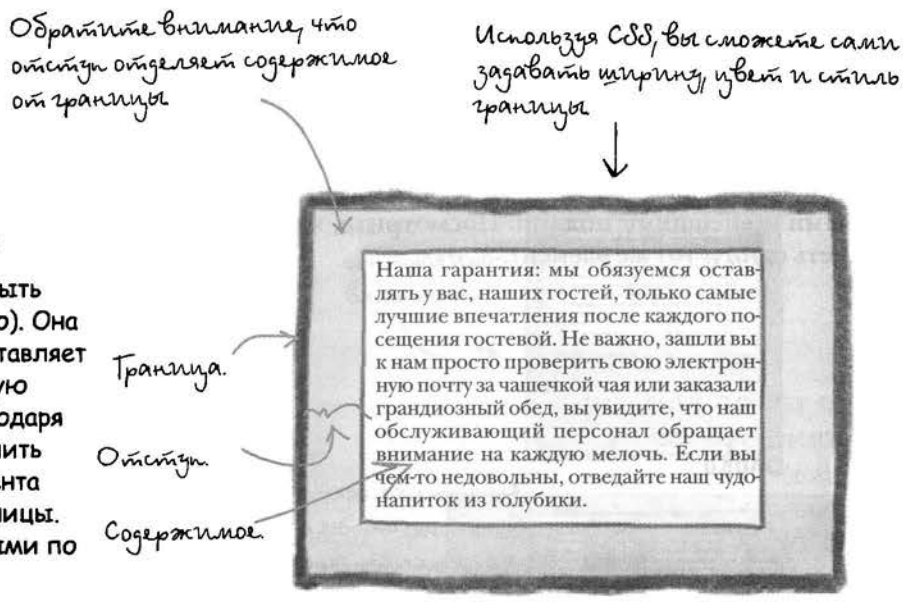
Браузер добавляет промежуточную область (необязательную) вокруг содержимого.

Наша гарантия: мы обязуемся оставлять у вас, наших гостей, только самые лучшие впечатления после каждого посещения гостевой. Не важно, зашли вы к нам просто проверить свою электронную почту за чашечкой чая или заказали грандиозный обед, вы увидите, что наш обслуживающий персонал обращает внимание на каждую мелочь. Если вы чем-то недовольны, отведайте наш чудо-напиток из голубики.

Используя CSS, вы сможете сами задавать размер этого отступа и даже указывать его отдельные размеры с каждой стороны (сверху, справа, снизу или слева).

Что такое граница?

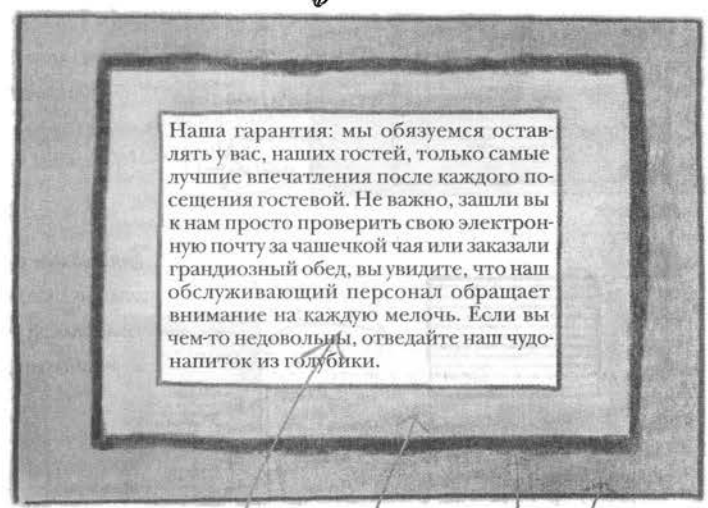
Вокруг элементов может быть граница (но не обязательно). Она окружает отступы и представляет собой линию, нарисованную вокруг содержимого. Благодаря ей можно визуально отделить содержимое вашего элемента от других элементов страницы. Границы могут быть разными по ширине, цвету и стилю.



Используя CSS, вы сможете сами задавать ширину полей со всех сторон или отдельно с каждой стороны (сверху, справа, снизу или слева).

Что такое поле?

Поля окружают границу и также являются необязательными для использования. С их помощью вы можете отделить свой элемент от остальных элементов на странице. Если два блока находятся рядом, то поля применяются для задания расстояния между ними. Как и отступы, поля прозрачны и не могут иметь цвета и других эффектов оформления.

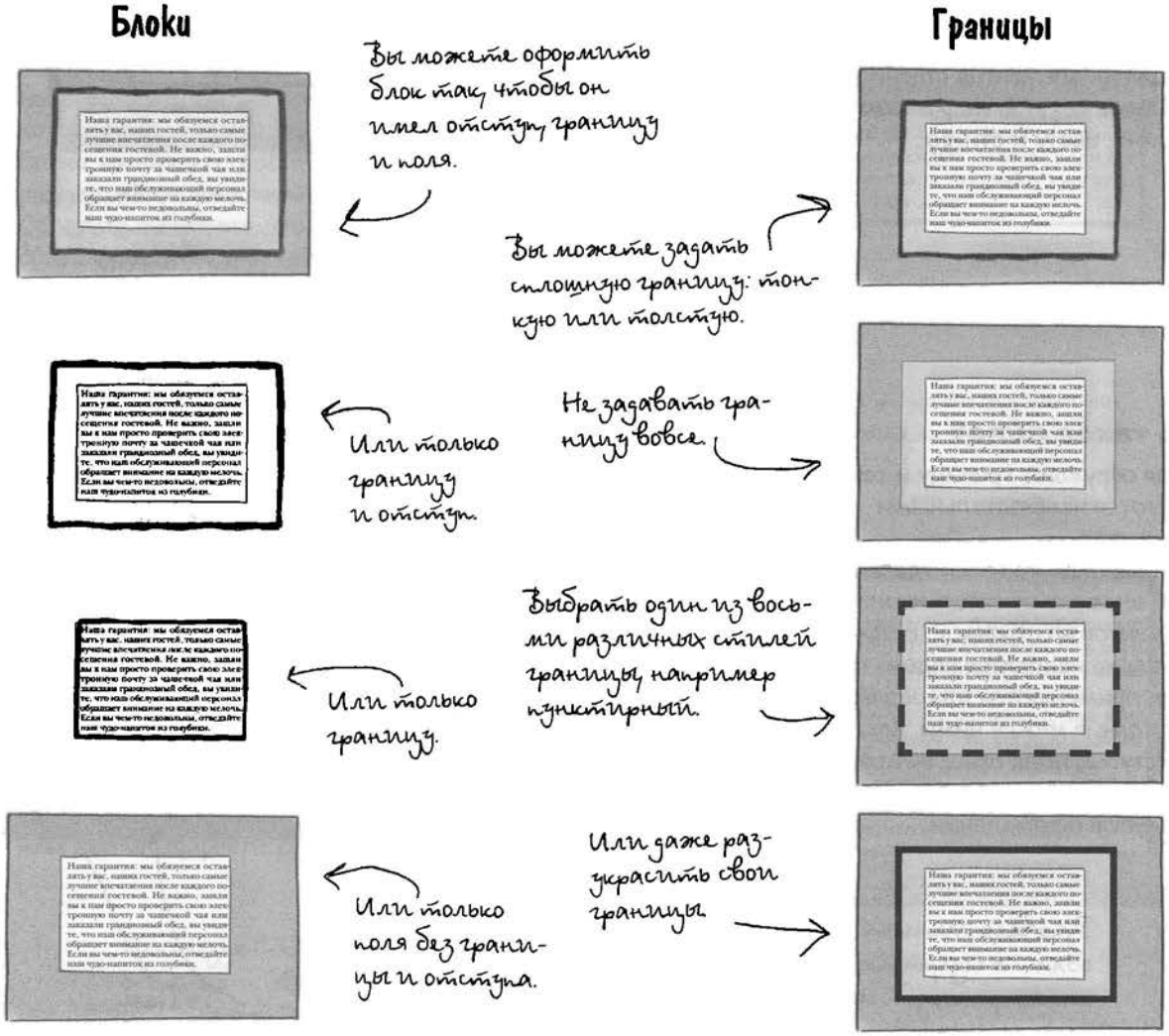


Это весь элемент целиком. Здесь есть область содержимого, окруженная отступами, границей и полями (все эти части необязательны).

Содержимое.
Отступ.
Граница.
Поле.

Что можно делать с блоками

Блочная модель выглядит достаточно простой: содержимое, отступ, граница и поля. Но если объединить все эти составляющие вместе, то можно получить бесконечное множество способов планировки схемы элемента с его внутренними отступами и внешними полями. Посмотрите, как может выглядеть один и тот же элемент.



Отступ

С помощью CSS можно задавать различные промежутки между содержимым и границей со всех сторон от области содержимого. Здесь мы задали большие левый и правый отступы.



А здесь — верхний и нижний отступы.



Поля

Вы точно так же можете контролировать поля.



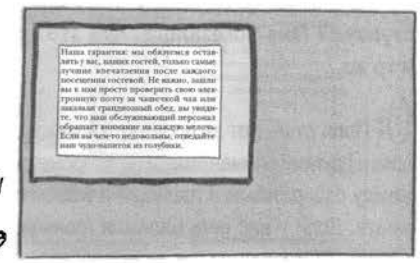
Здесь большие левое и правое поля.



Здесь содержимое смещено к нижнему правому углу.

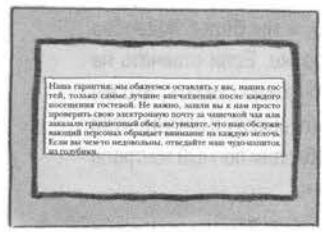


Как и для отступов, вы можете задавать параметры для каждой стороны отдельно, чтобы создавать такие поля, как, например, эти.



Область содержимого

Вы даже можете различными способами задавать ширину и высоту. Здесь мы расширили область содержимого.





Здесь область содержимого растянута вверх, но сужена.

Часто Задаваемые Вопросы

В: Мне кажется, что все эти знания о блочной модели пригодились бы мне, если бы я создавал программное обеспечение для браузеров, но как они могут помочь мне улучшить мои веб-страницы?

О: Чтобы создавать не просто веб-страницы, использующие стандартную разметку, предлагаемую браузерами, а пойти намного дальше, вам необходимо уметь контролировать то, как элементы располагаются на самой странице и относительно других элементов на ней. Для этого придется менять размеры отступов и полей для каждого элемента. Итак, для создания интересных дизайнов веб-страниц вам определенно нужно кое-что знать о блочных моделях.

В: В чем разница между полями и отступами? Мне показалось, что это одно и то же...

О: Поля отвечают за промежутки между вашим и другими элементами, а не за промежутки между содержимым и границей в вашем элементе. Если у вас есть видимая граница, то отступы, естественно, находятся внутри этой границы, а поля — снаружи. Считайте отступы частью элемента, а поля — тем, что окружает ваш элемент и отделяет от всего, что находится снаружи.

В: Я знаю, что все эти свойства необязательны, но нужно ли задавать отступы, если я хочу задать границу или поля?

О: Нет, все эти свойства не зависят друг от друга, так что вы можете задать границу, не устанавливая отступы, или определить поля, не задавая границу, и т. д.

В: Я не совсем понял, как элементы располагаются на странице и как на это влияют поля.

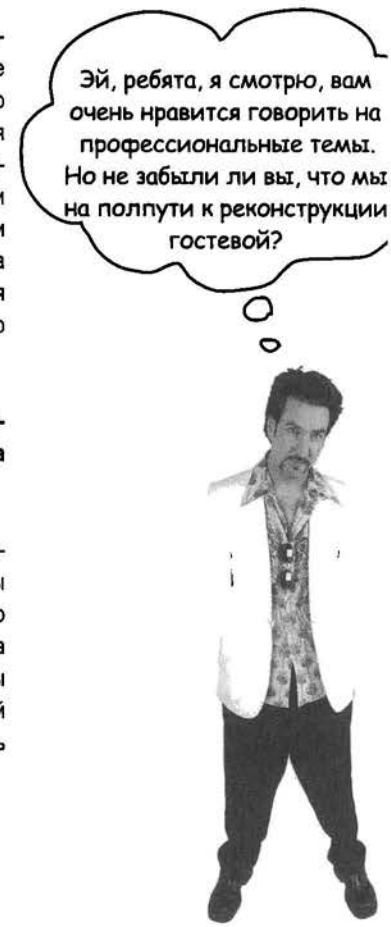
О: Пока подождите с этим. В этой главе вы еще узнаете кое-что о взаимодействии полей с другими элементами на странице, но подробно эту тему мы будем разбирать в главе 11, когда начнем говорить о позиционировании.

В: Вся стилизация полей и отступов заключается в том, что для них можно задать размер?

О: Так и есть. И поля, и отступы предназначены лишь для того, чтобы определять пустое пространство на странице, и нельзя напрямую задать цвет или другой эффект оформления этим свойствам. Однако, поскольку они прозрачные, они будут принимать цвет фона или фоновых рисунков. Различие между отступами и полями состоит в том, что цвет фона элемента (или фоновый рисунок) будет растягиваться под первым и не будет — под вторым. Как это работает, вы увидите очень скоро.

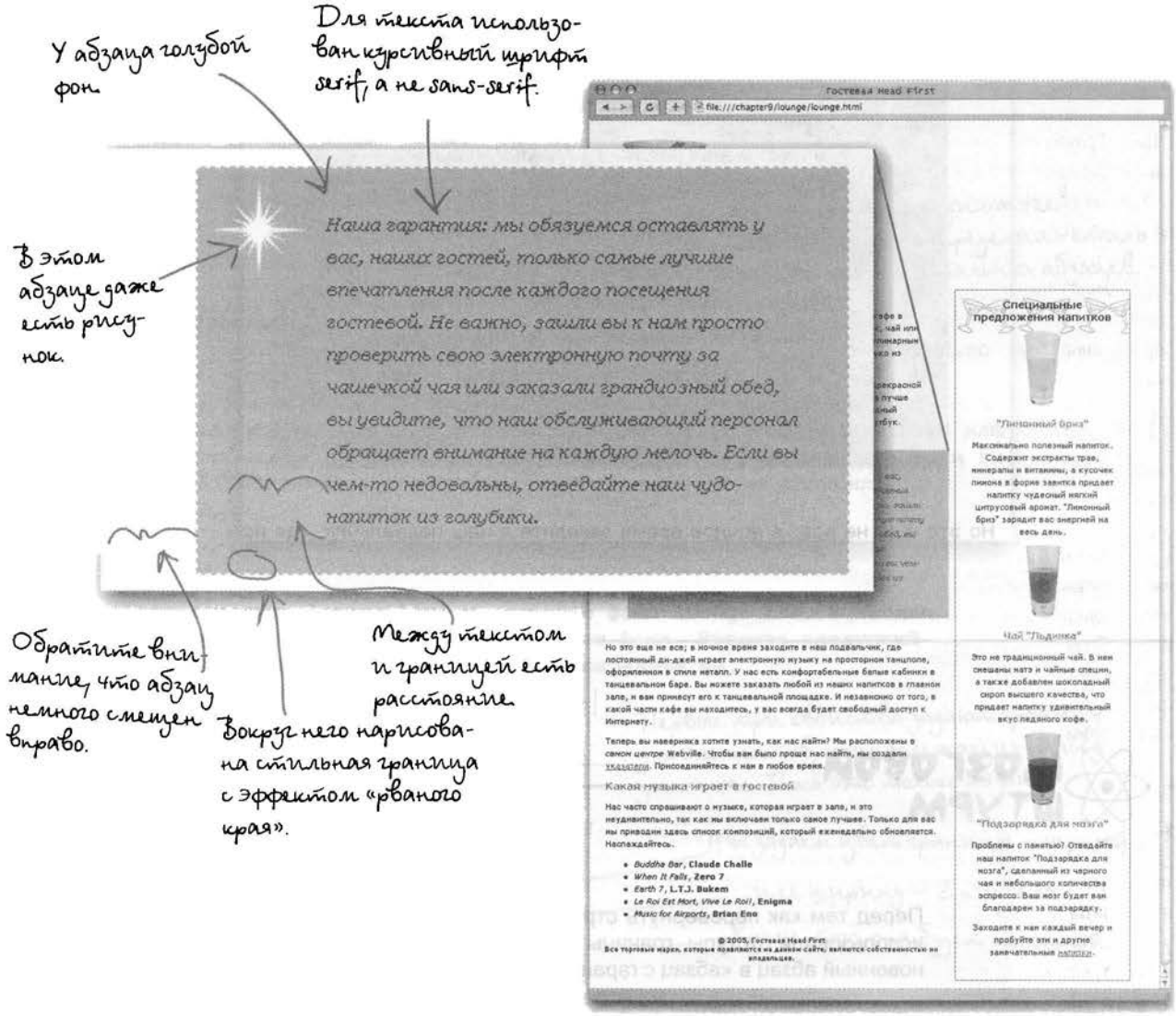
В: Размер области содержимого определяется автоматически исходя из размера самого содержимого?

О: Браузеры используют множество различных правил для определения высоты и ширины области содержимого, и мы более подробно рассмотрим их чуть позже. Если отвечать на ваш вопрос кратко, то можно сказать, что вы сами можете установить высоту и ширину этой области, если вам необходим полный контроль над размером элемента.



Тем временем вернемся к гостевой

Нам действительно нужно продолжать работу над гостевой страницей, поэтому вернемся к ней. Вы обратили внимание на абзац голубого цвета с различными стилями, когда смотрели на финальную версию страницы в начале этой главы? В этом абзаце был текст с гарантией, которая дается руководством гостевой своим клиентам. Совершенно очевидно, что они действительно хотели особо выделить свое обещание. Рассмотрим этот абзац очень подробно, а затем создадим его.



Возьми в руку карандаш

Проверьте, сможете ли вы верно распознать отступ, границу и поля этого абзаца. Обозначьте все отступы и поля (левые, правые, верхние и нижние).

свободный беспроводной доступ в Интернет, и захватите с собой свой

Наша гарантия: мы обязуемся оставлять у вас, наших гостей, только самые лучшие впечатления после каждого посещения гостевой. Не важно, зашли вы к нам просто проверить свою электронную почту за чашечкой чая или заказали грандиозный обед, вы увидите, что наш обслуживающий персонал обращает внимание на каждую мелочь. Если вы чем-то недовольны, отведайте наш чудо-напиток из голубики.

Но это еще не все; в ночное время заходите в наш подвальчик, где по

МОЗГОВОЙ ШТУРМ

Перед тем как перевернуть страницу, подумайте о том, как бы вы использовали отступы, границы и поля, чтобы преобразовать обыкновенный абзац в «абзац с гарантией».

Создание стиля оформления для «абзаца с гарантией»

Начнем с того, что внесем несколько небольших изменений в стиль оформления абзаца просто ради того, чтобы прочувствовать, как настраивается блок для этого абзаца. Вам нужно будет добавить данный абзац в класс **guarantee**, чтобы иметь возможность задать несколько специальных стилей отдельно для этого абзаца. Затем вы нарисуете границу вокруг него и определите фоновый цвет для области, находящейся внутри этой границы, что позволит вам точно понять, как абзац может быть блоком. Затем придется поработать над остальной частью стиля оформления. Рассмотрим, что вам нужно сделать.

1 **Откройте файл lounge.html и найдите абзац, который начинается словами «Наша гарантия». Укажите, что элемент <p> для этого абзаца принадлежит классу guarantee:**

Добавьте атрибут class со значением "guarantee". Помните, что благодаря классу можно определить стиль для этого абзаца, не влияя на стили остальных абзацев.

```
<p class="guarantee">
      Наша гарантия: мы обязуемся оставлять у вас, наших гостей, только
      самые лучшие впечатления после каждого посещения гостевой. Не
      важно, зашли вы к нам просто проверить свою электронную почту
      за чашечкой чая или заказали грандиозный обед, вы увидите, что наш
      обслуживающий персонал обращает внимание на каждую мелочь. Если вы
      чем-то недовольны, отведайте наш чудо-напиток из голубики.
</p>
```

2 **Сохраните страницу lounge.html и откройте файл lounge.css. В «абзац с гарантией» нужно добавить границу и цвет фона. Введите следующий CSS-код в конце вашей таблицы стилей и сохраните ее.**

Первые три свойства задают границу для элементов, принадлежащих классу guarantee. Пока это только абзац.

```
.guarantee {
      border-color:     black;
      border-width:     1px;
      border-style:     solid;
      background-color: #a7cece;
}
```

Мы задаем цвет границы – черный...

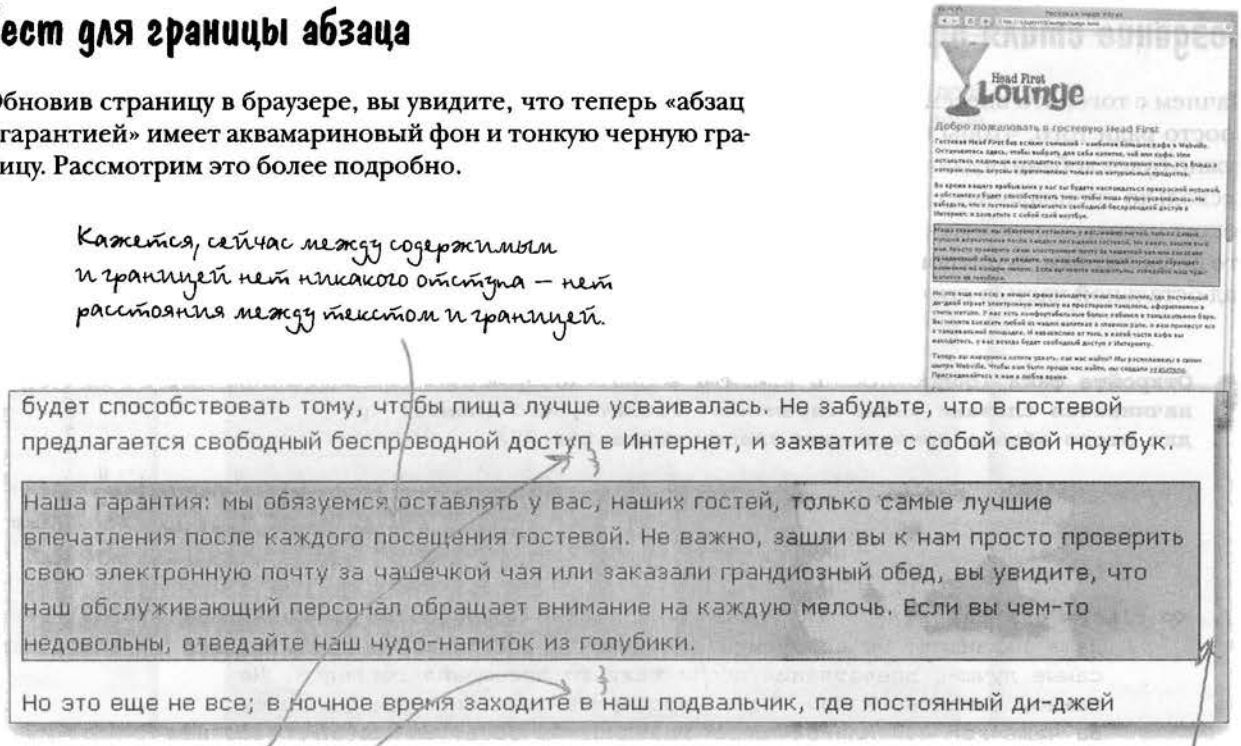
...и ее ширину – 2 пиксела.

Кроме того, она будет сплошной.

Мы задаем фоновый цвет для элемента, что поможет понять разницу между отступами и полями, а также улучшит внешний вид абзаца.

Тест для границы абзаца

Обновив страницу в браузере, вы увидите, что теперь «абзац с гарантией» имеет аквамариновый фон и тонкую черную границу. Рассмотрим это более подробно.

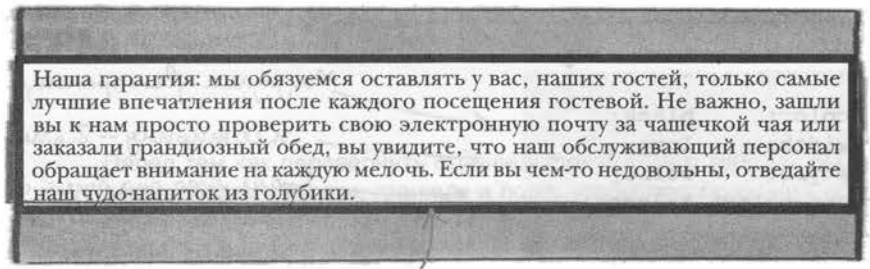


Кажется, сейчас между содержимым и границей нет никакого отступа — нет расстояния между текстом и границей.

Однако, кажется, внизу и вверху этого абзаца есть поля.

Полей между боковыми сторонами абзаца и краями окна браузера не видно.

Вот как будет выглядеть наш абзац, если мы нарисуем его блочную модель.



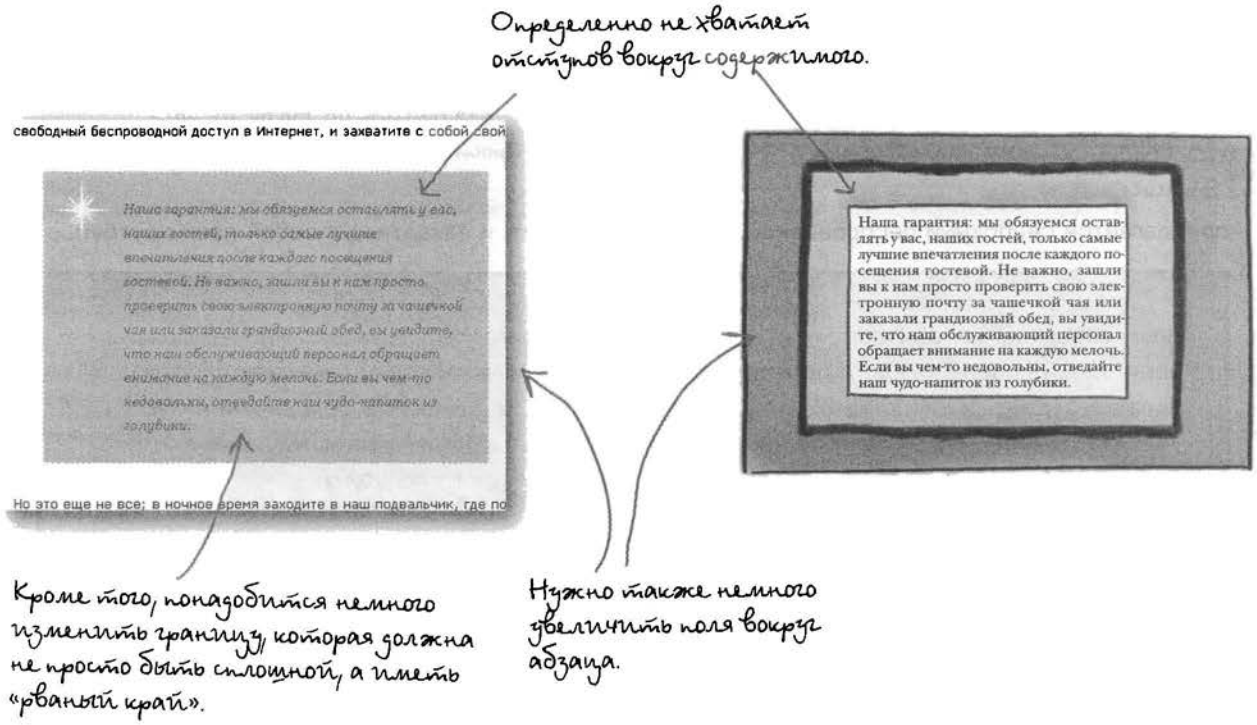
У нас есть отчетливые верхнее и нижнее поля.

Но левое и правое поля очень маленькие.

Кроме того, есть граница, которая расположена слишком близко к содержимому. Это означает, что отступ очень мал или его нет вообще.

Отступ, граница и поля «абзаца с гарантией»

Теперь, когда вы знаете, какие отступ, граница и поля заданы для «абзаца с гарантией», задумайтесь о том, какими бы вы их действительно хотели видеть.



Добавление отступов

Начнем с добавления промежутков между содержимым и границей. В CSS есть свойство **padding**, которое вы можете использовать, чтобы задать отступы со всех четырех сторон содержимого. Его можно задавать либо в пикселах, либо в процентах относительно размеров области, находящейся внутри границы. Мы будем использовать пикселы и установим отступы шириной по 25 пикселов с каждой стороны.

```
.guarantee {
      border-color:     black;
      border-width:     1px;
      border-style:     solid;
      background-color: #a7cece;
      padding:          25px;
}
```

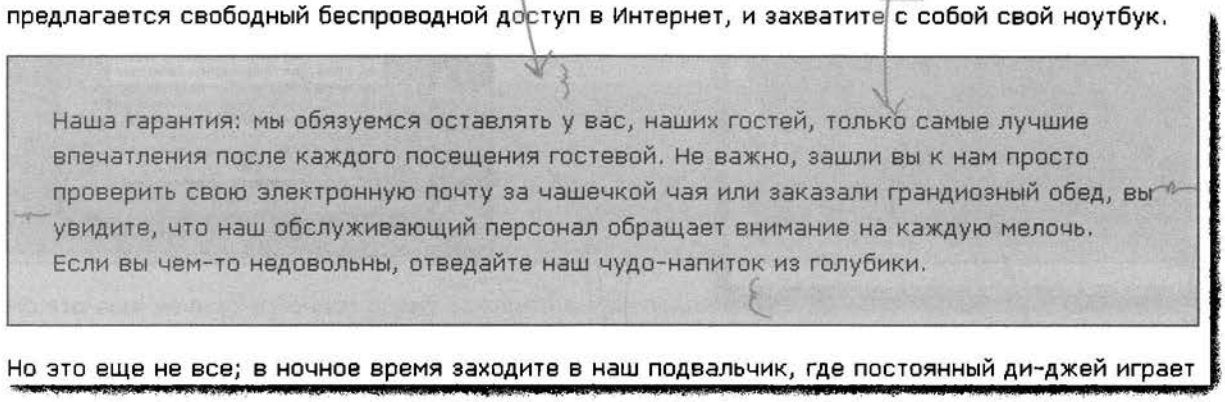
Мы добавляем отступы шириной 25 пикселов с каждой стороны содержимого (сверху, справа, снизу и слева).

Тест для отступов

Обновив страницу в браузере, вы увидите, что теперь текст «абзаца с гарантией» не так плотно зажат в рамку. Между содержимым и границей появилось расстояние, и текст стало намного проще читать.

Теперь между краем текстового содержимого и границей вы можете видеть свободное пространство шириной 25 пикселов.

Обратите внимание, что цвет фона применяется и для самого содержимого, и для отступов. Но он не распространяется на поля.



Теперь добавим поля

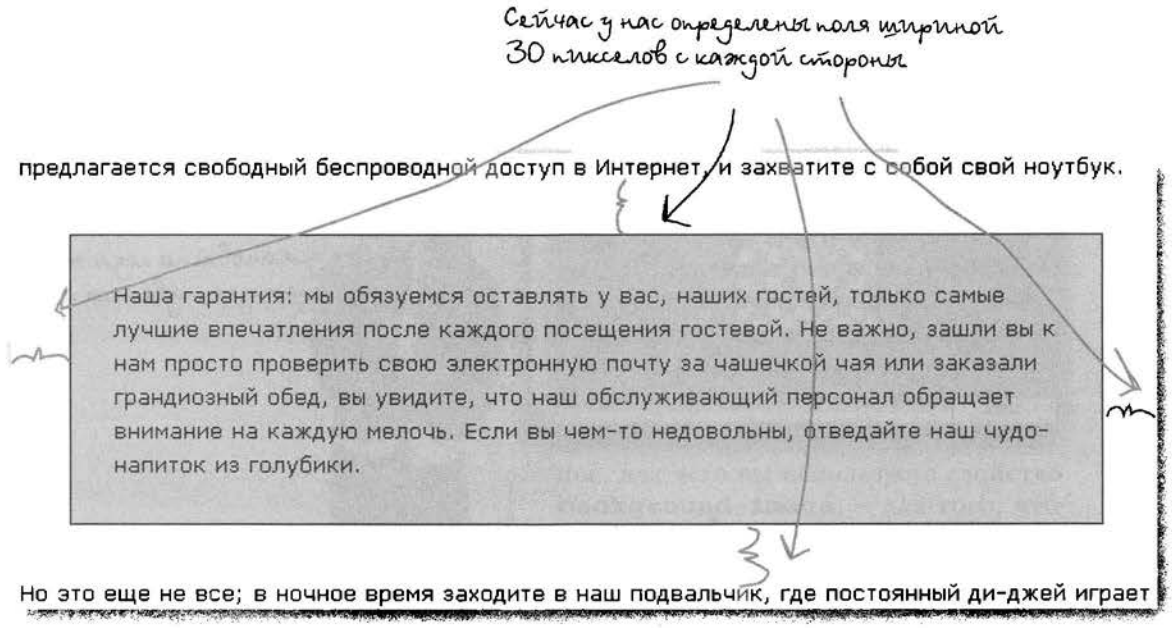
С помощью CSS очень просто добавить поля. Как и для отступов, вы можете задать размер полей в процентах или пикселах. Добавьте поля шириной 30 пикселов вокруг всего «абзаца с гарантией». Посмотрите, как это сделать:

```
.guarantee {
      border-color:     black;
      border-width:     1px;
      border-style:     solid;
      background-color: #a7cece;
      padding:          25px;
      margin:           30px;

}
```

Мы добавляем поля шириной 30 пикселов со всех сторон содержимого (сверху, справа, снизу и слева).

Тест для полей

Обновив гостевую страницу, вы увидите, что с появлением полей абзац действительно стал отделяться от остального текста. В сочетании с фоновым цветом это позволяет привлечь к абзацу особое внимание пользователя.





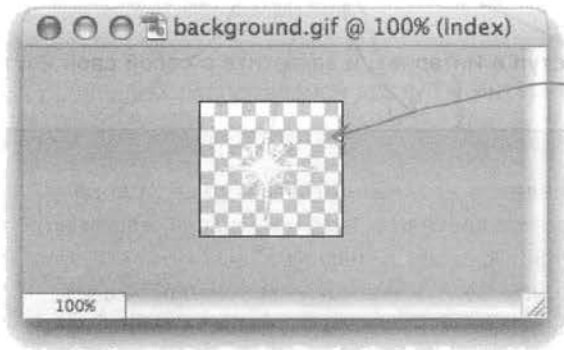
Упражнение

Если вы посмотрите на «абзац с гарантией» в его окончательной версии, то увидите, что он должен иметь курсивный шрифт serif, а межстрочные интервалы здесь должны быть больше, чем на остальной странице. Кроме того, присмотревшись внимательно, вы заметите, что установлен серый цвет текста. Напишите CSS-код для задания межстрочных интервалов `1.9em`, курсивного начертания шрифта, цвета шрифта `#444444` и семейства шрифтов Georgia, Times New Roman, Times, serif. Сверьте свой код с приведенным в конце главы, а затем протестируйте его.

Добавление фонового рисунка

Вы почти закончили работу. Что осталось сделать? Все еще нужно поместить в абзац белый рисунок в виде звездочки и поработать над границей, которая пока представляет собой сплошную черную линию. Начнем работу с изображения.

Если вы откроете папку `chapter10/lounge/images`, то найдете там изображение `background.gif`, которое выглядит следующим образом.



Это изображение представляет собой простую белую звездочку на прозрачном фоне. Обратите внимание, что вокруг нее также установлена подложка, по цвету совпадающая с цветом фона.

Теперь вам просто нужно поместить это изображение в элемент **<p>**. Для этого вы будете использовать элемент **<img>**, верно? *Не спешите.* Если хотите использовать изображение в качестве фона какого-нибудь элемента, то для этого есть другой способ. В CSS можно использовать фоновый рисунок для какого-нибудь элемента, определяя свойство **background-image**. Попробуем это сделать и посмотрим, как оно работает:

Это свойства, которые вы добавили, выполняя упражнение из предыдущего раздела.

```
.guarantee {
      line-height:      1.9em;
      font-style:       italic;
      font-family:      Georgia, "Times New Roman", Times, serif;
      color:            #444444;
      border-color:     black;
      border-width:     1px;
      border-style:     solid;
      background-color: #a7cece;
      padding:          25px;
      margin:           30px;
      background-image: url(images/background.gif);
}
```

Добавьте это в свой CSS-код, сохраните файл и обновите страницу.



Нет, свойство **background-image** имеет строго определенное назначение – устанавливать фоновые рисунки для элементов. Оно не используется просто для добавления рисунков на страницы – для этого вы однозначно должны применять элемент **<img>**.

Можете запомнить это так: единственное, для чего вы используете свойство **background-image**, – для того, чтобы ваше изображение смотрелось более привлекательно. Цель использования на страницах элемента **<img>** более существенная – размещение на странице таких изображений, как фотографии и логотипы.

Итак, мы могли бы просто поместить изображение внутрь абзаца и, скорее всего, получили бы такой же результат. Однако эта звездочка – чисто декоративный элемент, она не несет никакой смысловой нагрузки и используется только для того, чтобы украсить другой элемент. Так что имеет смысл использовать свойство **background-image**.

Тест для фонового рисунка

Этот тест очень интересный: мы получили фоновое изображение, но рисунок дублируется под абзацем несколько раз. Давайте детально разберемся с фоновыми рисунками в CSS, и после этого вы сами сможете исправить код так, чтобы звездочка отображалась только один раз.



Это изображение звездочки установлено в качестве фона «абзаца с гарантией». Обратите внимание, что оно расположено над фоновым цветом и имеет прозрачный фон.

Обратите также внимание на то, что фоновые рисунки, как и фоновые цвета, отображаются только под областью содержимого и отступами и не отображаются под полями.



CSS крупным планом

Свойство `background-image` помещает изображение на фон элемента. Рассмотрим еще два свойства, которые предназначены для того, чтобы регулировать фоновые изображения. Это `background-position` и `background-repeat`.

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

Свойство background-image задает адрес, который может быть либо относительным путем, либо полным URL-адресом (http://...).

Обратите внимание, что вокруг URL не нужно ставить никаких двойных кавычек.

Закрепление фонового изображения

По умолчанию фоновые рисунки повторяются по горизонтали и вертикали. Но, к счастью, существует значение **no-repeat** свойства **background-repeat**, которое управляет тем, повторяется ли изображение, и если повторяется, то как. Кроме того, по умолчанию браузеры размещают изображения в верхнем левом углу элемента, как раз там, где мы хотим его видеть. Однако мы все же добавим свойство **background-position**, чтобы посмотреть, как оно работает.

```
.guarantee {
      line-height:         1.9em;
      font-style:          italic;
      font-family:         Georgia, "Times New Roman", Times, serif;
      color:               #444444;
      border-color:        black;
      border-width:        1px;
      border-style:        solid;
      background-color:    #a7cece;
      padding:             25px;
      margin:              30px;
      background-image     url(images/background.gif);
      background-repeat:   no-repeat;
      background-position: top left;
}
```

Необходимо добавить два новых свойства.

Мы хотим, чтобы фоновое изображение не повторялось.

Нам также нужно, чтобы оно располагалось в верхнем левом углу.

Свойство background-position устанавливает месторасположение изображения и может быть задано в пикселах, процентах или с помощью таких ключевых слов, как top, left, right, bottom и center.

Помещает изображение в верхнем левом углу элемента.

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

Есть множество различных способов определения месторасположения элемента в CSS. Подробнее мы поговорим об этом через две главы.

По умолчанию фоновое изображение «замощает» фон элемента или повторяется по горизонтали и вертикали снова и снова, пока не заполнит все пространство. Свойство background-repeat контролирует, как именно происходит это повторение.

Это другие значения background-repeat, которые вы можете использовать.

```
background-repeat: repeat;
```

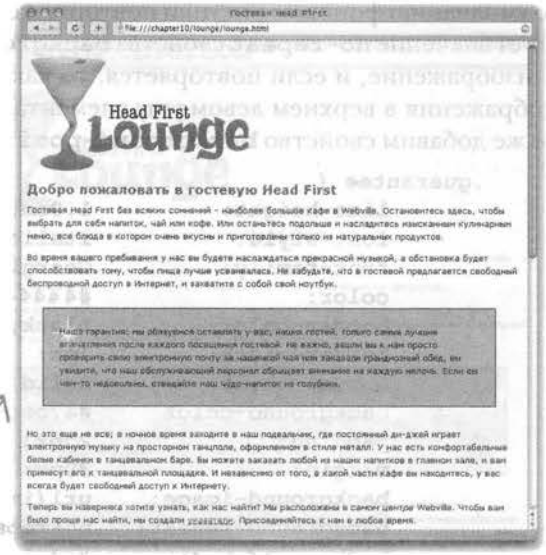
Указывает, что изображение должно повторяться и по вертикали, и по горизонтали. Такое поведение устанавливается по умолчанию.

no-repeat	Отображает картинку только один раз, то есть не повторяет ее.
repeat-x	Повторяет изображение только по горизонтали.
repeat-y	Повторяет изображение только по вертикали.
inherit	Делает то же самое, что и родительский элемент.

Дальше »

Еще один тест для фонового изображения

Итак, снова протестируем. На этот раз, кажется, мы намного ближе к тому, чего хотим. Однако, поскольку это фоновое изображение, текст располагается прямо поверх него. Как нам это исправить? Именно для этого применяется отступ! Он позволяет добавить промежутки вокруг содержимого. Увеличим отступ с левой стороны и посмотрим, можем ли мы избавиться от этой проблемы раз и навсегда.



Так намного лучше. Теперь изображение не повторяется.

Но мы еще хотим, чтобы текст не накладывался на рисунок.

Как увеличить отступ только с левой стороны?

Для отступов, полей и даже границ в CSS включены свойства, определяющие направление: **top**, **right**, **bottom** и **left**. Чтобы увеличить промежуток между содержимым и границей только с левой стороны, используйте свойство **padding-left**:

```
.guarantee {
      line-height:        1.9em;
      font-style:         italic;
      font-family:        Georgia, "Times New Roman", Times, serif;
      color:              #444444;
      border-color:       black;
      border-width:       1px;
      border-style:       solid;
      background-color:   #a7cece;
      padding:            25px;
      padding-left:       80px;
      margin:             30px;
      background-image:   url(images/background.gif);
      background-repeat:  no-repeat;
      background-position: top left;
}
```

Мы используем свойство padding-left, чтобы увеличить отступ с левой стороны.

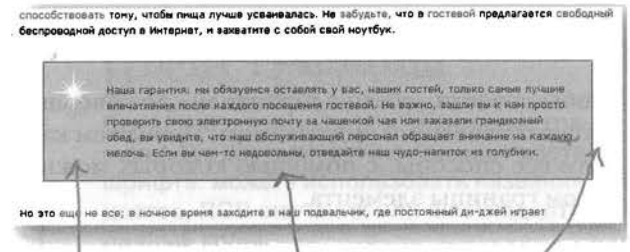
Обратите внимание, что сначала мы задали отступы шириной 25 пикселов с каждой стороны содержимого, а затем использовали свойство только для левой стороны.

Здесь очередность свойств имеет значение. Если вы измените ее, то сначала зададите отступ с левой стороны, а затем общее свойство padding установит ширину 25 пикселов для всех отступов, включая и тот, что находится слева!

Мы все еще здесь?

Убедитесь, что сохранили все изменения, и обновите страницу. Вы увидите, что отступ с левой стороны увеличился и текст теперь не наползает на изображение звездочки. Это очень хороший пример того, когда лучше использовать отступы, а не поля. Если вам нужно больше свободного пространства вокруг самой области содержимого, используйте отступы, а если вам нужно пространство между разными элементами или между каким-то элементом и краем окна браузера, то используйте поля.

На самом деле нам стоило бы немного увеличить поле с правой стороны, чтобы наш абзац еще больше выделялся на фоне остального текста. Давайте сделаем это, и затем останется лишь подкорректировать границу.



Отступы выглядят великолепно. Теперь текст хорошо расположен по отношению к изображению и не наползает на него.

Нам все еще нужно поработать над границей.

Сейчас мы можем увеличить поле справа, чтобы абзац еще больше выделялся на странице.

Как увеличить размер поля только с правой стороны?

Это делается точно так же, как и для отступа: добавляется еще одно свойство, **margin-right**, позволяющее увеличить размер правого поля.

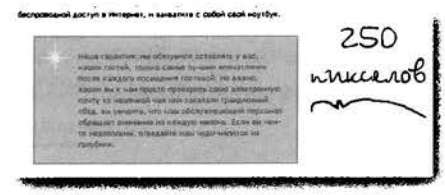
Видите закономерность? Существуют свойства, применяемые всем сторонам сразу, и свойства для каждой стороны, если нужно задать индивидуальные расстояния.

```
.guarantee {
      line-height:         1.9em;
      font-style:          italic;
      font-family:         Georgia, "Times New Roman", Times, serif;
      color:               #444444;
      border-color:        black;
      border-width:        1px;
      border-style:        solid;
      background-color:    #a7cece;
      padding:             25px;
      padding-left:        80px;
      margin:              30px;
      margin-right:        250px;
      background-image:    url(images/background.gif);
      background-repeat:   no-repeat;
      background-position: top left;
}
```

Помните, мы уже задали ширину полей по 30 пикселов с каждой стороны.

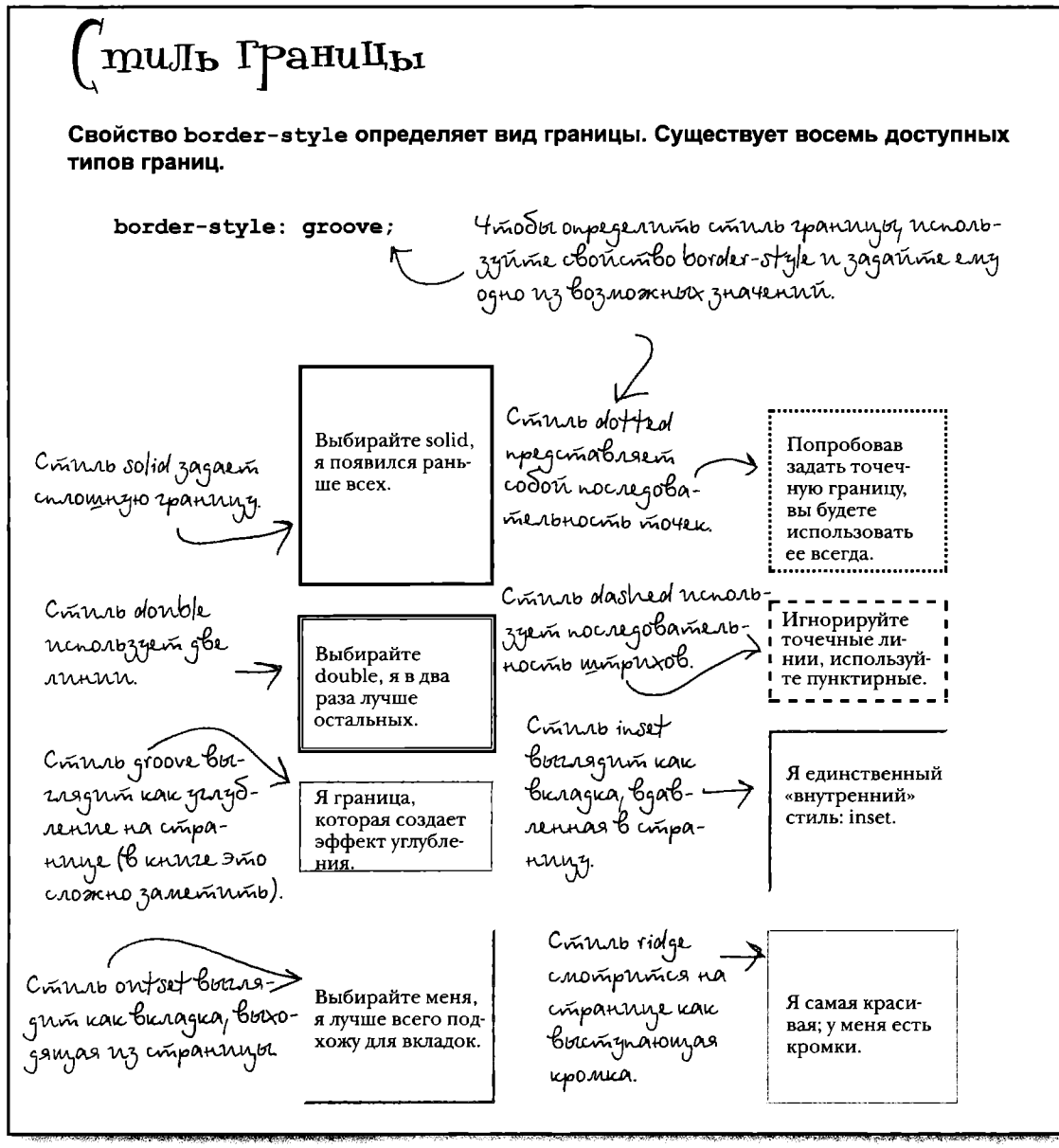
Теперь мы хотим переопределить это значение для правой стороны и установить для нее ширину поля 250 пикселов.

Добавьте новое свойство **margin-right** и обновите страницу. Теперь у абзаца с правой стороны будет поле шириной 250 пикселов.



Двухминутное руководство по границам

Чтобы довести «абзац с гарантией» до совершенства, осталось лишь улучшить внешний вид границы. Перед тем как это сделать, рассмотрите все способы, с помощью которых можно управлять внешним видом границы элемента.



Ширина границы

Свойство `border-width` задает ширину границы. Чтобы устанавливать ширину, можете использовать ключевые слова или пикселы.

```
border-width: thin;
border-width: 5px;
```

Можете определить ширину с помощью ключевых слов thin, medium или thick. Кроме того, можно задать количество пикселов.

thin
medium
thick

1px
2px
3px
4px
5px
6px

Цвет границы

Свойство `border-color` задает цвет границы. Тут все аналогично заданию цвета шрифта: можете использовать названия цветов, RGB-значения или шестнадцатеричные коды.

```
border-color: red;
border-color: rgb(100%, 0%, 0%);
border-color: #ff0000;
```

Используйте свойство border-color, чтобы задать цвет границы. Вы можете определить цвет любым из привычных способов.



Описание сторон границы

border-top-color
border-top-style
border-top-width

border-right-color
border-right-style
border-right-width

border-bottom-color
border-bottom-style
border-bottom-width

border-left-color
border-left-style
border-left-width

Как для полей и отступов, вы можете задать свой собственный стиль, ширину или цвет каждой отдельной части границы (верхней, нижней, левой или правой):

```
border-top-color: black;
border-top-style: dashed;
border-top-width: thick;
```

Эти свойства заданы только для верхней части границы. Вы можете описывать каждую часть отдельно.

Доведение границы до совершенства

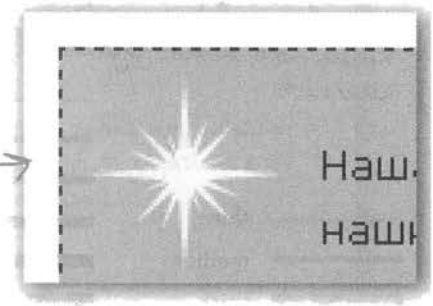
Настало время довести «абзац с гарантией» до совершенства. Нам осталось придать границе эффект «рваного края». Но что это за стиль? Имеющиеся в наличии значения — **solid**, **double**, **dotted**, **dashed**, **groove**, **ridge**, **inset** и **outset**. Как же мы можем придать границе нужный эффект? На самом деле это просто небольшая хитрость: мы используем пунктирную границу белого цвета (цвет фона страницы тоже белый).

Начните с того, что задайте для границы стиль **dashed**. Найдите в файле `lounge.css` свойство **border-style** и измените его следующим образом:

```
border-style: dashed;
```

Здесь мы поменяли стиль границы с solid на dashed.

Далее сохраните файл и обновите страницу. Вы увидите такую границу:



Теперь, чтобы придать границе эффект «рваного края», поменяйте ее цвет на белый. Это создаст впечатление того, что граница сливается с фоном. Итак, попробуйте это сделать: найдите свойство **border-color** и поменяйте его значение на **white**.

```
border-color: white;
```

Здесь мы поменяли цвет границы с черного на белый.

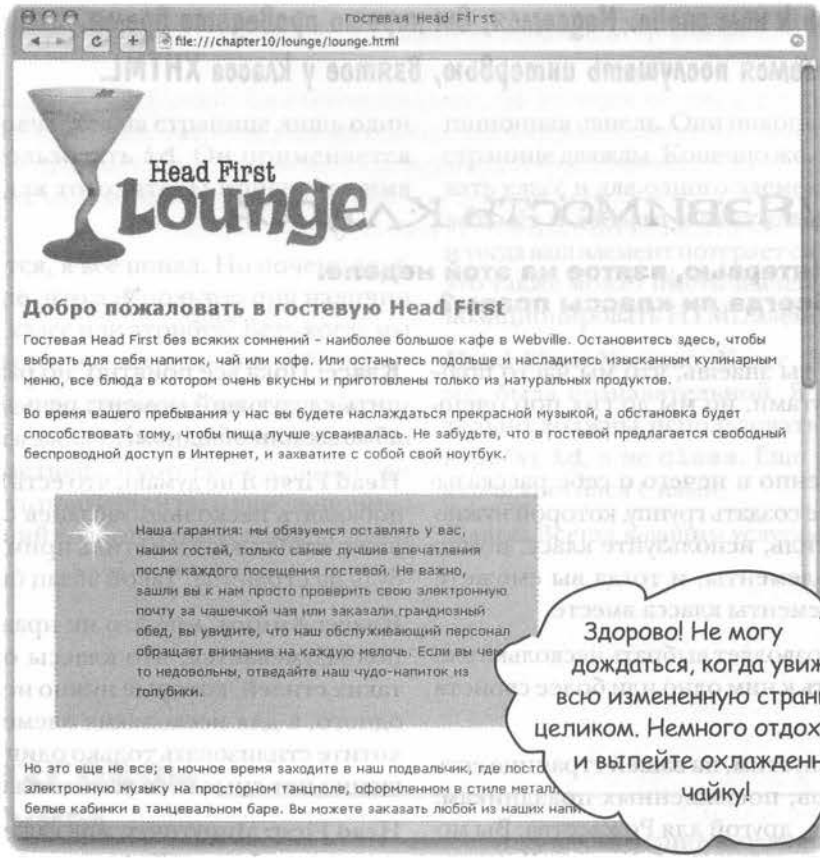
Снова сохраните файл и обновите страницу. На этот раз вы увидите границу с эффектом «рваного края».





Размеры, задаваемые ключевыми словами thin, medium и thick, по умолчанию в разных браузерах могут различаться.

Если размер границы действительно очень важен для вас, лучше задавайте ее ширину в пикселах.



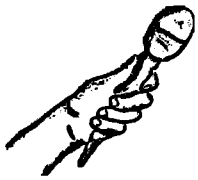


Поздравляем!

Браво! Вы взяли обычный HTML-абзац и преобразовали его во что-то намного более привлекательное и стильное, использовав лишь 15 строк CSS-кода.

Вы неплохо потрудились, поэтому теперь мы предлагаем вам немного отдохнуть. Возьмите себе чашечку охлажденного чая и расслабьтесь. Вам понадобится некоторое время, чтобы переварить всю информацию. Когда вы вернетесь, мы расскажем вам еще о нескольких нюансах работы с CSS.

Добро пожаловать к нам снова. Надеемся, вы хорошо проведете время. Мы как раз собираемся послушать интервью, взятое у класса XHTML.



Уязвимость класса

Интервью, взятое на этой неделе.
Всегда ли классы правы?

Head First: Класс, ты знаешь, что мы часто пользуемся твоими услугами, но мы до сих пор очень мало знаем о тебе.

Класс: А мне особенно и нечего о себе рассказывать. Если вы хотите создать группу, которой нужно придать особый стиль, используйте класс, поместите в него свои элементы, и тогда вы сможете стилизовать все элементы класса вместе.

Head First: Класс позволяет выбрать несколько элементов и применить к ним одно или более свойств стиля?

Класс: Именно. Допустим, на вашей странице есть несколько разделов, посвященных праздникам: один для Хэллоуина, другой для Рождества. Вы можете добавить все элементы из раздела о Хэллоуине в класс **`halloween`**, а из раздела о Рождестве — в класс **`christmas`**. Тогда вы сможете оформлять наборы элементов из различных классов независимо друг от друга, например использовать оранжевый цвет для элементов, относящихся к Хэллоуину, и красный — для элементов, относящихся к Рождеству. Для этого вам нужно будет написать правила стиля для каждого класса.

Head First: Использовать классы достаточно удобно. Мы совсем недавно видели отличный пример этому, не так ли?

Класс: Не могу сказать наверняка, я не работал в это время. Вам придется описать мне, что это был за пример.

Head First: Хорошо, мы работали над гостевой страницей Head First, где есть «абзац с гарантией» от ее владельцев. Они хотели, чтобы этот абзац выделялся на фоне остального текста.

Класс: Пока все понятно, но разрешите мне уточнить следующий момент: речь идет о нескольких абзацах или об одном?

Head First: Я не думаю, что есть какие-то причины добавлять несколько «абзацев с гарантией», и не вижу, чтобы этот же стиль применялся еще где-нибудь на странице. Такой абзац был только один.

Класс: Хмммм, мне это не нравится. Понимаете, подразумевается, что классы определяются для таких стилей, которые нужно использовать не для одного, а для нескольких элементов. Если же вы хотите стилизовать только один элемент, то совершенно нет смысла создавать класс.

Head First: Минуточку, мне кажется, что наш класс отлично работает. Что же тогда не так?

Класс: Тише-тише, не надо нервничать. Все, что вам нужно сделать, — использовать вместо атрибута **`class`** атрибут **`id`**. Это займет не больше минуты.

Head First: Атрибут **`id`**? Я думал, он используется для якорей, как сказано в главе 4.

Класс: Атрибут **`id`** используется в разных случаях. На самом деле он представляет собой уникальный идентификатор для элемента.

Head First: Можешь ли ты рассказать нам про атрибут **`id`** более подробно? Я ничего не знал об этом раньше. Оказывается, на протяжении всей этой главы я неправильно использовал класс!

Класс: Без паники; это достаточно распространенная ошибка. По сути, вам нужно знать, что класс используется тогда, когда одинаковый стиль оформления планируется применять к нескольким элементам. Но если элемент, который вам нужно

стилизовать, встречается на странице лишь один раз, следует использовать **id**. Он применяется исключительно для того, чтобы присвоить имя одному элементу.

Head First: Кажется, я все понял. Но почему вообще имеет значение, что я использую при наличии одного элемента: класс или атрибут? Ведь когда мы использовали класс, у нас все отлично работало.

Класс: Потому что на странице иногда действительно встречаются элементы, единственные в своем роде. «Абзац с гарантией», о котором вы упомянули, один из них; но есть примеры и получше, например верхний или нижний колонтитул на странице, навигационная панель. Они никогда не используются на странице дважды. Конечно же, вы можете использовать класс и для одного элемента, но кто-то другой возьмет и добавит в этот класс еще один элемент, и тогда ваш элемент потеряет свою уникальность. Все это также может иметь значение, когда вы начнете позиционировать HTML-элементы.

Head First: Хорошо, Класс. Эта беседа, конечно же, была познавательной. Я думаю, мы действительно должны использовать для нашего абзаца атрибут **id**, а не **class**. Еще раз благодарю тебя, что встретился с нами.

Класс: Всегда к вашим услугам, Head First!



Выберите, какой атрибут лучше использовать для следующих элементов (**class** или **id**).

id	*class*	
❑	❑	Абзац, используемый в качестве нижнего колонтитула страницы.
❑	❑	Совокупность заголовков и абзацев, где приводятся биографии компаний.
❑	❑	Элемент **<img>** с «картинкой дня».

id	*class*	
❑	❑	Набор элементов **<p>**, в которых содержатся рецензии на фильмы.
❑	❑	Элемент ** ** со списком заданий, которые вам нужно выполнить.
❑	❑	Элементы **<q>**, содержащие цитаты Баккару Банзая.

Ответ: Нижний колонтитул, «картинка дня», список заданий — главные кандидаты на использование атрибута **id**.

Атрибут id

Поскольку вы уже использовали атрибут **id** для элементов **<a>** и знаете, как применяется атрибут **class**, вам не придется тратить много времени, чтобы узнать, как еще можно работать с атрибутом **id**. Допустим, на вашей странице есть нижний колонтитул. Поскольку на странице обычно только один нижний колонтитул, в данном случае явно лучше использовать атрибут **id**. Присвойте идентификатор **footer** абзацу с текстом колонтитула таким образом:

Просто добавьте атрибут id и укажите уникальное имя идентификатора.

В отличие от атрибута class, на странице можно указать всего один элемент с идентификатором footer.

```
<p id="footer">Пожалуйста, не крадите эту страницу, хотя она не защищена
авторским правом</p>
```

Любой элемент может иметь не более одного идентификатора.

Имена идентификаторов должны начинаться с буквы и состоять только из букв и цифр. Пробелы и специальные символы использовать нельзя.

Присвоение идентификатора имени имеет много общего с добавлением элемента в класс. Различия состоят в том, что атрибут называется **id**, а не **class**, один элемент не может иметь несколько идентификаторов, а вы не можете задавать на странице несколько элементов с одинаковым идентификатором.

Часто Задаваемые Вопросы

В: Какой же в этом смысл? Зачем мне применять id только для того, чтобы указать, что такой элемент на странице единственный? Я ведь могу с тем же успехом использовать и class.

О: Вы всегда можете использовать класс вместо уникального идентификатора, но есть множество причин, по которым этого лучше не делать. Допустим, вы работаете над веб-проектом в команде людей. Один из членов вашей команды, увидев класс, может подумать, что его можно применять для других элементов. Однако если он увидит уникальный идентификатор, то поймет, что тот предназначен для одного-единственного элемента. Есть еще несколько причин того, почему важно использовать id, и вы узнаете их через несколько глав. Например, при позиционировании элементов на странице вам понадобится, чтобы каждый элемент имел собственный идентификатор.

В: Может ли один элемент одновременно иметь идентификатор и принадлежать какому-либо классу?

О: Да, может. Запомните это так: id — всего лишь уникальный идентификатор элемента, но он не запрещает элементу принадлежать одному или нескольким классам (точно так же, как уникальное имя не запрещает вам стать членом одного или нескольких клубов).

Как же селектор идентификатора применяется в CSS

Элемент с идентификатором выбирается почти так же, как элемент, принадлежащий классу. Вспомним еще раз: если у вас есть класс **specials**, то существует несколько способов для выбора его элементов. Вы можете выбрать конкретные элементы класса следующим образом:

```
p.specials {
        color: red;
}
```

Так выбираются только абзацы, принадлежащие классу specials.

Вы также можете выбрать все элементы, принадлежащие классу **specials**:

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

Так выбираются все элементы класса specials.

Селектор идентификатора используется почти таким же образом. Чтобы выбрать элемент по его значению, укажите перед идентификационным именем символ «#» (сравните: работая с классом, вы указываете перед его названием символ «.»). Допустим, вы хотите выбрать элемент с идентификатором **footer**:

```
#footer {
        color: red;
}
```

Так выбирается любой элемент с идентификатором footer.

Кроме того, вы можете выбрать только элемент **<p>** с идентификатором **footer**. Это делается так:

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

Так выбирается элемент <p>, имеющий идентификационное имя footer.

Единственное различие между селекторами состоит в том, что селектор идентификатора должен соответствовать только одному элементу на странице.

Использование идентификатора для гостевой

Итак, наш «абзац с гарантией» должен иметь идентификатор, так как подразумевается, что он будет использоваться на странице только один раз. Вам будет не сложно внести соответствующие изменения даже сейчас.

Шаг первый: замените атрибут class атрибутом id в файле lounge.html.

Просто замените атрибут class атрибутом id.

```
<p id="guarantee">
      Наша гарантия: мы обязуемся оставлять у вас, наших гостей, только
      самые лучшие впечатления после каждого посещения гостевой. Не
      важно, зашли вы к нам просто проверить свою электронную почту
      за чашечкой чая или заказали грандиозный обед, вы увидите, что наш
      обслуживающий персонал обращает внимание на каждую мелочь. Если вы
      чем-то недовольны, отведайте наш чудо-напиток из голубики.
</p>
```

Шаг второй: замените селектор класса .guarantee селектором идентификатора в lounge.css.

Просто замените в селекторе символ «.» символом «#».

```
#guarantee {
      line-height:         1.9em;
      font-style:          italic;
      font-family:         Georgia, "Times New Roman", Times, serif;
      color:               #444444;
      border-color:        white;
      border-width:        1px;
      border-style:        dashed;
      background-color:    #a7cece;
      padding:             25px;
      padding-left:        80px;
      margin:              30px;
      margin-right:        250px;
      background-image:    url(images/background.gif);
      background-repeat:   no-repeat;
      background-position: top left;
}
```

Шаг третий: сохраните изменения и обновите страницу.

Итак, все должно выглядеть ТОЧНО так, как и прежде. Но разве вам не приятнее осознавать, что теперь все сделано именно так, как должно быть?



Часто Задаваемые Вопросы

В: Почему вы использовали селектор #guarantee, а не p#guarantee?

О: Можно применять любой из этих селекторов, и они сделают одно и то же. Мы знаем, что на нашей странице идентификатор всегда ссылается на абзац, так что нет разницы, какой селектор использовать (к тому же #guarantee проще). Однако при работе с более сложным набором страниц у вас может возникнуть ситуация, когда на различных страницах уникальный идентификатор ссылается на разные элементы: например, на одной — на абзац, на другой — на список или блочную цитату. Вам может понадобиться несколько правил для одного идентификатора, например p#someid и blockquote#someid, что зависит от того, какой элемент применяется на странице.

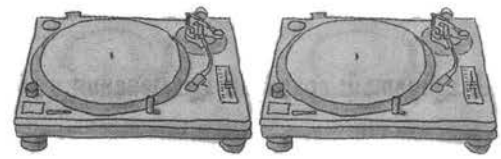
В: Нужно начинать с использования атрибута class, а затем, если становится известно, что элемент будет встречаться на странице только один раз, заменять его атрибутом id?

О: Нет. В большинстве случаев при разработке веб-страниц вы заранее знаете, будет элемент уникальным или нет. А в этой главе мы сделали все именно в такой последовательности только потому, что в начале работы вы ничего не знали про селектор идентификатора.

В: В главе 4 мы использовали атрибут id для элемента <a>, чтобы создать якорь. Если я применяю атрибут id для других типов элементов, будет ли он так же использоваться в качестве пункта назначения?

О: В большинстве современных браузеров это поддерживается, однако в более старых версиях — нет.

Смешивание нескольких таблиц стилей



Перед тем как мы закончим с этой главой, давайте попробуем смешать несколько таблиц стилей. До сих пор вы использовали только одну таблицу. Это хорошо, но разве мы когда-то говорили, что нельзя использовать несколько? Вы можете определить целый набор таблиц стилей для какого-нибудь XHTML-документа. Вы спросите, когда это может понадобиться? Есть несколько таких случаев. Рассмотрим один из них.

Представьте, что дела гостевой Head First быстро пошли в гору, ее владельцы стали франшизодателями, разместили свои акции на рынке и т. д. (конечно же, все благодаря вам и вашей отличной работе). В таком случае, конечно же, появится корпоративный сайт, состоящий из сотни страниц, и очевидно, что вы захотите оформить его с помощью внешних таблиц стилей. В компании появится множество отделов, каждый из которых будет с индивидуальным стилем. Однако наверняка при этом франшизодатели гостевой захотят контролировать общий стиль. Вот как это будет выглядеть:



Использование нескольких таблиц стилей

Итак, как же сначала определить общий корпоративный стиль, а потом позволить различным отделам и франшизополучателям гостевой внести в него индивидуальные изменения? Вы используете несколько таблиц стилей, например, так:

```
<!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="ru" xml:lang="ru" >
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />
    <title>Гостевая Head First</title>
      <link type="text/css" href="corporate.css" rel="stylesheet" />
      <link type="text/css" href="beverage-division.css" rel="stylesheet" />
      <link type="text/css" href="lounge-seattle.css" rel="stylesheet"  />
  </head>
  <body>
  .
  .
  .
  </body>
</html>
```

В своем XHTML вы можете задать несколько таблиц стилей. В данном случае мы указали три.

Одна таблица стилей — для всей корпорации.

Гостевая в Сиэтле имеет свою особенность.

Порядок имеет значение! Правила каждой следующей таблицы стилей могут переопределять правила предыдущей.

Отдел напитков может внести немного индивидуальных особенностей в общий корпоративный стиль или даже переопределить отдельные правила стиля корпорации.

Часто Задаваемые Вопросы

В: Итак, порядок следования таблиц стилей имеет значение?

О: Да, каждая последующая таблица имеет преимущество над предыдущей. Например, если у вас свойство font-family для элемента <body> определено как в таблице стилей для всей корпорации, так и в таблице стилей для отдела, то преимущество будет иметь то, что задано для отдела, так как ссылка на него из XHTML-кода расположена ниже.

В: Понадобится ли мне все это когда-нибудь для простого сайта?

О: Вы удивитесь, но иногда таблицы стилей применяются в качестве основы и вместо того, чтобы менять такую таблицу, можно просто сослаться на нее, а затем добавить собственную таблицу стилей, в которой указать, что необходимо поменять.

В: Вы можете подробнее рассказать о том, как выбирается стиль для определенного элемента?

О: Мы уже кое-что рассказывали об этом в главе 8. Сейчас к тем знаниям просто добавьте еще немного — порядок следования ссылок на таблицы стилей имеет значение. В следующей главе, когда вы узнаете еще несколько подробностей о CSS, мы детально разберем то, как браузер узнает, какой стиль и для какого элемента использовать.

Таблицы стилей — теперь не только для представления в окнах браузеров



Если вам захочется больше узнать о стилях для страниц, предназначенных для распечатывания или для отображения на мобильных устройствах, то ищите подробности в приложении к книге.

Перейдем ко второй причине использования нескольких таблиц стилей. Допустим, вы хотите, чтобы стили оформления страницы на экранах персональных и карманных компьютеров, на мобильных устройствах и в печатной версии имели небольшие различия. Существует дополнительный атрибут `media` для элемента **`<link>`**. Можете его использовать, чтобы указать тип устройства, для которого предназначен файл с таблицами стилей.

С помощью атрибута media можно указать тип устройства, для которого предназначена данная таблица стилей.

```
<link type="text/css" rel="stylesheet" href="lounge-screen.css" media="screen" />
```

Здесь мы указали, что таблица стилей подходит для экранов компьютеров.

Вы можете задать несколько элементов **`<link>`** с различными типами устройств, например так:

Если вы не задаете тип устройства, то таблицы стилей будут применяться для отображения документа на всех устройствах.

```
<link type="text/css" href="lounge.css" rel="stylesheet" />
<link type="text/css" href="lounge-print.css" rel="stylesheet" media="print" />
<link type="text/css" href="lounge-mobile.css" rel="stylesheet" media="handheld" />
```

Теперь мы задали еще два элемента <link>: один используется для печати страницы, другой — для ее отображения на небольших устройствах с маленьким экраном и ограниченной скоростью передачи данных.

Часто Задаваемые Вопросы

В: Все это здорово. Получается, что я могу задавать различные таблицы стилей для разных устройств?

О: Да, вы можете создать несколько таблиц стилей и затем ссылаться на них из XHTML-кода. Работа браузера — выбрать нужную таблицу стилей, основываясь на типе устройства.

В: Допустим, имеется ссылка для небольших устройств с маленьким экраном и ограниченной скоростью передачи данных и ссылка для любых браузеров. Какая из них будет применена?

О: Браузер небольшого устройства с маленьким экраном и ограниченной скоростью передачи данных выберет обе ссылки. Но, учитывая то, что ссылка для таких устройств указана позже ссылки для всех устройств, преимущество будет за ней, как и в случае с таблицами стилей для корпорации и ее отделов, о которых мы только что говорили.

В: Итак, существуют типы устройст screen (для компьютеров), print (для печа ти) и handheld (для мобильных устройст и сотовых телефонов). А какие еще бы вают типы устройств?

О: Есть еще aural (для экранных диктс ров), braille (для людей со слабым зрением использующих устройства чтения азбук Брайля), projection (для презентаций с по мощью проекторов), tty (для телетайпо и терминалов) и tv (для телевидения).



Упражнение

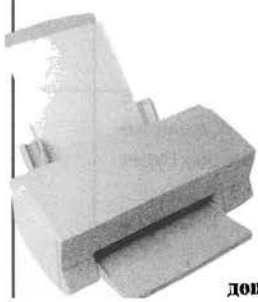
В папке `chapter10/lounge/print` найдите файл `print.css`. Откройте файл `lounge.html`, который находится в папке `chapter10/lounge`, и создайте новую ссылку на эту таблицу стилей для типа устройства **`print`**. Убедитесь, что вы добавили атрибут **`media="screen"`** в элемент **`<link>`**, который ссылается на `lounge.css`. Таким образом, у вас будет одна таблица стилей для экрана компьютера, другая — для принтера. Затем сохраните файл, обновите страницу и выберите в браузере пункт меню Print (Печать). Запустите принтер, чтобы увидеть результат!

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

Это новая ссылка, которую вам нужно добавить в файл lounge.html.

Это печатная версия. С помощью CSS вы совершенно изменили внешний вид страницы в ее печатной версии.

К сожалению, не все браузеры поддерживают атрибут media, поэтому если вы не добились такого результата, попробуйте сделать то же самое в другом браузере.





ДОПОЛНИТЕЛЬНО НЕОБХОДИМ ПРИНТЕР, ОН В КНИГУ НЕ ВКЛЮЧЕН.

ПОВТОРИМ выученное

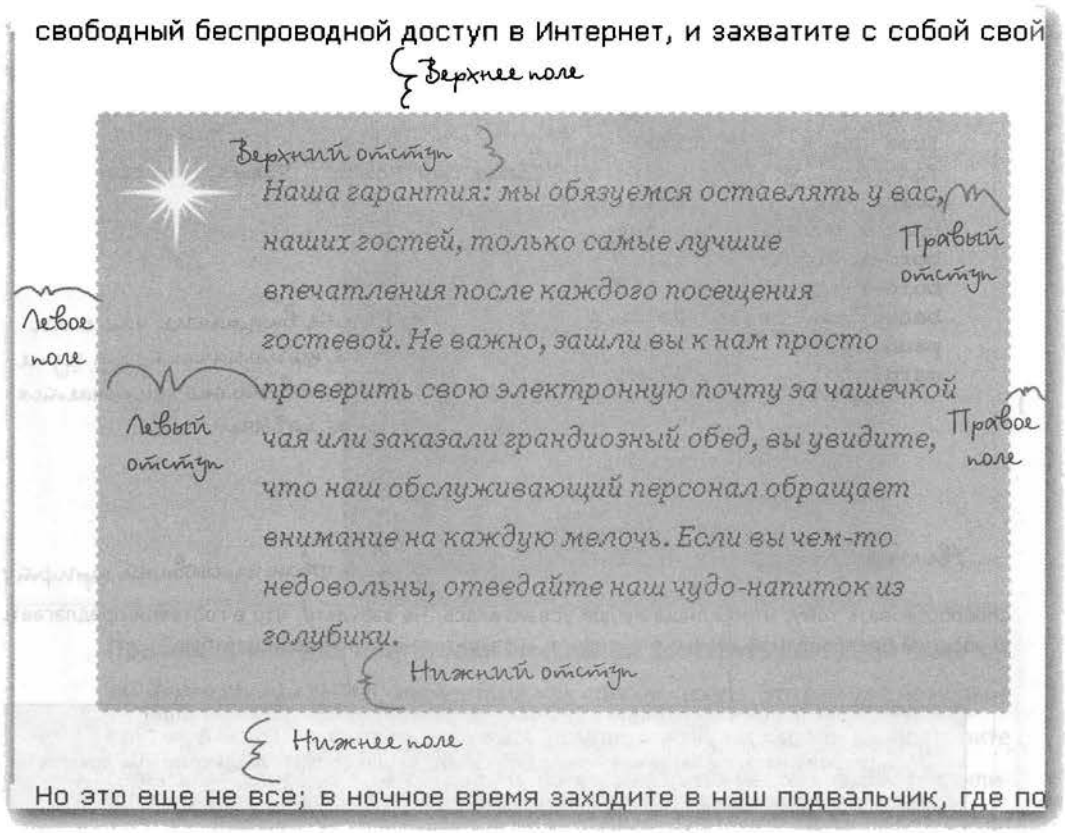


- Для управления способом отображения элементов в CSS используется блочная модель.
- Блоки состоят из области содержимого, отступа, границы и полей.
- В области содержимого находится само содержимое элемента.
- Отступы используются для того, чтобы создать свободное пространство вокруг области содержимого.
- Граница окружает область содержимого и это свободное пространство вокруг него.
- Поля окружают границу, отступы и область содержимого и позволяют добавлять свободное пространство между разными элементами.
- Отступ, граница и поля — необязательные части блока.
- Фон элемента виден под областью содержимого и отступами и не виден под полями.
- Размер отступов и полей может задаваться в пикселах либо в процентах.
- Ширина границы может задаваться в пикселах или с помощью ключевых слов **thin**, **medium** и **thick**.
- Существует восемь различных стилей границ, среди которых: **solid**, **dashed**, **dotted** и **ridge**.
- В CSS имеются свойства для определения полей, отступов и границ со всех сторон сразу (сверху, справа, снизу, слева) или с каждой стороны отдельно.
- Применяйте свойство **line-height**, чтобы определять межстрочный интервал.
- Используя свойство **background-image**, вы можете поместить в фоне элемента изображение.
- Применяйте свойства **background-position** и **background-repeat**, чтобы задать месторасположение фонового рисунка и определить, будет ли он повторяться по горизонтали и (или) по вертикали.
- Используйте атрибут **class** для элементов, которые хотите оформить в одинаковом стиле.
- Применяйте атрибут **id**, чтобы дать элементу уникальное имя. Вы можете использовать его, чтобы задать определенный стиль для конкретного элемента.
- На странице может быть только один элемент с данным идентификационным именем.
- Вы можете выбрать элемент по его идентификационному имени, используя запись типа **id # селектор**; например **#мойлюбимыйid**.
- У элемента не может быть несколько идентификационных имен, но он может принадлежать нескольким классам.
- В своем XHTML-коде вы можете использовать несколько таблиц стилей.
- Если одно и то же свойство по-разному определено в двух таблицах стилей, то преимущество будет иметь та таблица, ссылка на которую в XHTML-файле расположена позже.
- Используя атрибут **media**, например **print** или **handheld**, в элементе **<link>**, вы можете указывать тип устройств, на которых будет отображена страница.

Возьми в руку карандаш

Решение

Проверьте, сможете ли вы верно распознать отступ, границу и поля этого абзаца. Обозначьте все отступы и поля (левые, правые, верхние и нижние).





Решение упражнения

Если вы посмотрите на «абзац с гарантией» в его окончательной версии, то увидите, что он должен иметь курсивный шрифт serif, а межстрочные интервалы здесь должны быть больше, чем на остальной странице. Кроме того, присмотревшись внимательно, вы заметите, что установлен серый цвет текста. Напишите CSS-код для задания межстрочных интервалов `1.9em`, курсивного начертания шрифта, цвета шрифта `#444444` и семейства шрифтов Georgia, Times New Roman, Times, serif. Ниже приведено решение. Вы тестировали это?

Вы можете добавлять новые свойства в любом месте правила. Мы добавили их вверху.

```
.guarantee {
      line-height:      1.9em;
      font-style:       italic;
      font-family:      Georgia, "Times New Roman", Times, serif;
      color:            #444444;
      border-color:     black;
      border-width:     1px;
      border-style:     solid;
      background-color: #a7cece;
      padding:          25px;
      margin:           30px;
}
```

Обратите внимание, что если название шрифта состоит из нескольких слов, то оно заключается в двойные кавычки.

Увеличен межстрочный интервал.

Использован курсивный шрифт serif.

способствовать тому, чтобы пища лучше усваивалась. Не забудьте, что в гостевой предлагается свободный беспроводной доступ в Интернет, и захватите с собой свой ноутбук.

Наша гарантия: мы обязуемся оставлять у вас, наших гостей, только самые лучшие впечатления после каждого посещения гостевой. Не важно, зашли вы к нам просто проверить свою электронную почту за чашечкой чая или заказали грандиозный обед, вы увидите, что наш обслуживающий персонал обращает внимание на каждую мелочь. Если вы чем-то недовольны, отведайте наш чудо-напиток из голубики.

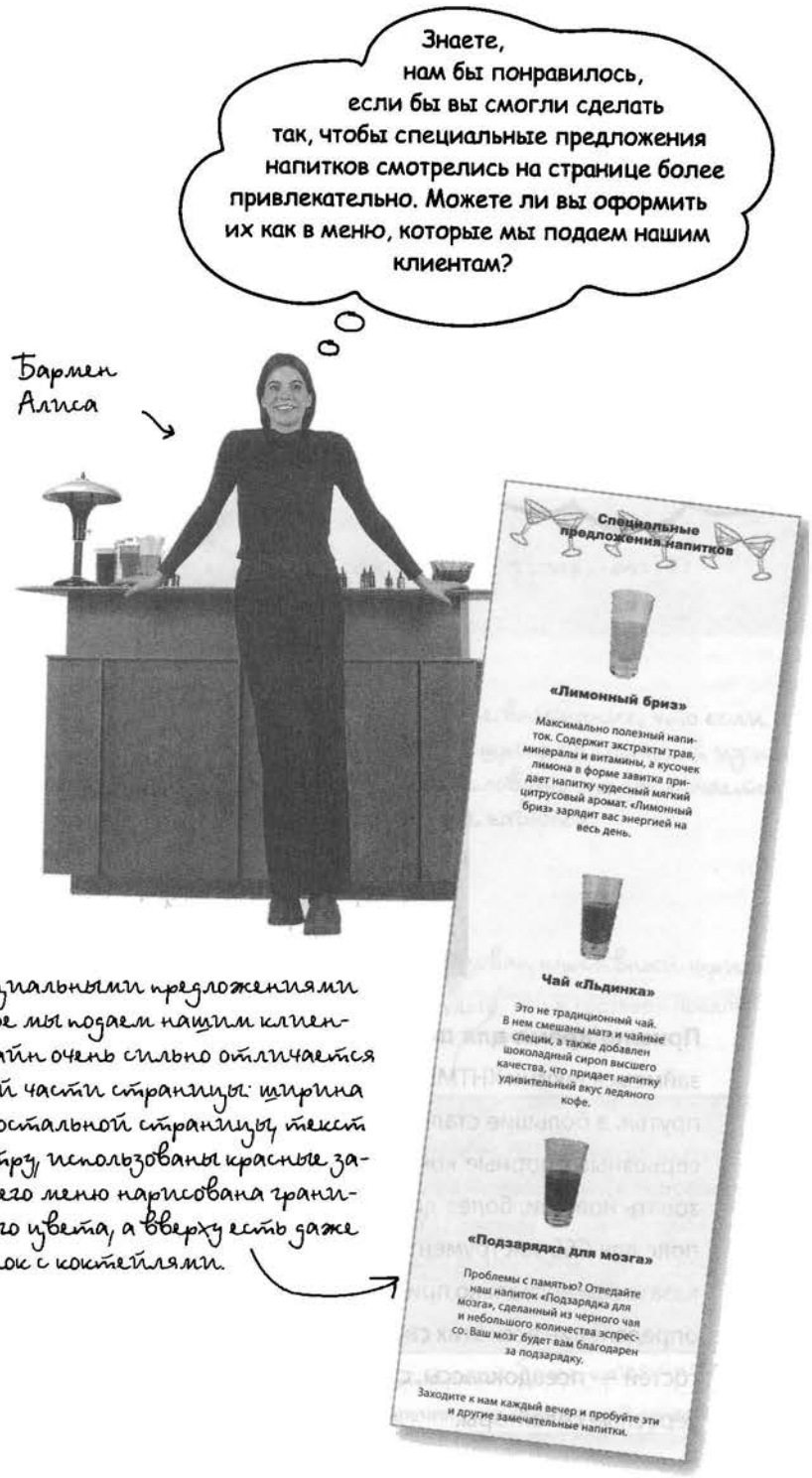
Серый цвет делает текст более приятным для глаз.

Но это еще не все; в ночное время заходите в наш подвальчик, где постоянный ди-джей играет

Современная веб-конструкция



Пришло время для подготовки массивной конструкции. В этой главе мы займемся такими XHTML-элементами, как и . Это вам уже не мелкие прутья, а большие стальные балки. С помощью и вы построите серьезные опорные конструкции и, расставив их по местам, сможете стилизовать новыми, более действенными методами. Обратите внимание, что ваш пояс для CSS-инструментов практически заполнен, так что настало время показать вам несколько приемов для быстрого доступа к ним, что очень облегчит определение всех этих свойств. В эту главу мы также пригласили специальных гостей — псевдоклассы, с помощью которых вы сможете создавать очень интересные селекторы.



Это меню со специальными предложениями напитков, которое мы подаем нашим клиентам. Ого, его дизайн очень сильно отличается от дизайна другой части страницы: ширина меньше ширины остальной страницы, текст выровнен по центру, использованы красные заголовки, вокруг всего меню нарисована граница аквамаринового цвета, а вверху есть даже несколько картинок с коктейлями.

Рассмотрим XHTML с описанием напитков

Конечно, Алиса задала сложную задачу. Она хочет, чтобы мы взяли имеющийся XHTML-код для гостевой и сделали так, чтобы страница выглядела как меню, которое они подают своим клиентам. Хммм... немного страшновато, но ведь у нас есть CSS, так что давайте попробуем.

Перед тем как с головой погрузиться в создание дизайна, давайте пересмотрим имеющийся у нас код. Это фрагмент XHTML-документа со специальными предложениями напитков. Вы найдете его в файле `lounge.html` из папки `chapter11/lounge`:

Раздел со специальными предложениями начинается с заголовка <h2>.

У нас есть три напитка, имеющих одну структуру.

В элементе <p> для каждого напитка есть изображение...

...для названия используется заголовок <h3>...

...и описание, также в отдельном абзаце.

Эта структура повторяется для каждого напитка.

И наконец, в самом низу есть еще один абзац с кое-каким текстом и ссылкой на саму страницу.

```
<h2>Специальные предложения напитков</h2>

<p>
      <img src="images/yellow.gif" alt="Напиток «Лимонный бриз»" />
</p>
<h3>«Лимонный бриз»</h3>
<p>
      Максимально полезный напиток. Содержит экстракты трав,
      минералы и витамины, а кусочек лимона в форме завитка
      придает напитку чудесный мягкий цитрусовый аромат.
      «Лимонный бриз» зарядит вас энергией на весь день.
</p>

<p>
      <img src="images/chai.gif" alt="Напиток Чай «Льдинка»" />
</p>
<h3>Чай «Льдинка»</h3>
<p>
      Это не традиционный чай. В нем смешаны матэ и чайные
      специи, а также добавлен шоколадный сироп высшего качества,
      что придает напитку удивительный вкус ледяного кофе.
</p>

<p>
      <img src="images/black.gif" alt="Напиток «Подзарядка для мозга»" />
</p>
<h3>«Подзарядка для мозга»</h3>
<p>
      Проблемы с памятью? Отведайте наш напиток «Подзарядка для
      мозга», сделанный из черного чая и небольшого количества
      эспрессо. Ваш мозг будет вам благодарен за подзарядку.
</p>

<p>
      Заходите к нам каждый вечер и пробуйте эти и другие
      замечательные
      <a href="beverages/elixir.html"
         title="Напитки гостевой Head First">напитки</a>.
</p>
```



Джим: Да брось ты, Фрэнк, мы можем просто создать один или два класса и стилизовать все элементы меню отдельно от остальной страницы.

Фрэнк: Да, возможно, все не так уж страшно. Я уверен, что существует простое свойство, с помощью которого можно выровнять текст по центру. И мы также знаем, как задавать цвет для текста.

Джим: Минуточку, а как насчет границы, которая нарисована вокруг всего меню?

Фрэнк: Очень просто. Мы только что узнали, как создаются границы. Вспомни: граница может быть нарисована вокруг любого элемента.

Джо: Хмм, я так не думаю. Если ты посмотришь на XHTML-код, то увидишь, что он представляет собой набор элементов **<h2>**, **<h3>** и **<p>**. Если мы зададим границы для каждого отдельного элемента, то они и будут выглядеть как отдельные блоки.

Фрэнк: Ты прав, Джо. Нам нужен элемент, в который мы вложим все наши элементы, а потом задим границу только для него одного. Тогда у нас будет одна граница вокруг всего раздела со специальными предложениями напитков.

Джим: Теперь я понимаю, почему тебе платят большие деньги, Фрэнк. Можем ли мы вложить все наши элементы в элемент **<p>** или **<blockquote>**?

Фрэнк: Нельзя использовать **<p>** потому что он не может включать в себя другие блочные элементы, а заголовки и абзацы таковыми как раз и являются. И я не думаю, что мы бы стали так делать, ведь абзацы предназначены только для текста.

Джо: И **<blockquote>** — это тоже не то, что нам нужно, потому что это меню с напитками, а не цитата.

Фрэнк: Все-таки мне кажется, что мы на верном пути. Я читаю одну книгу по HTML и CSS и как раз дошел до раздела, в котором описывается новый элемент ** **. Я думаю это именно то, что нам нужно.

Джо: ** ** — что это? Звучит так будто это какой-то математический термин.

Фрэнк: Надо сказать, что ты не так уж и далек от истины, потому что с помощью ** ** страницу можно *разбить* на логические разделы.

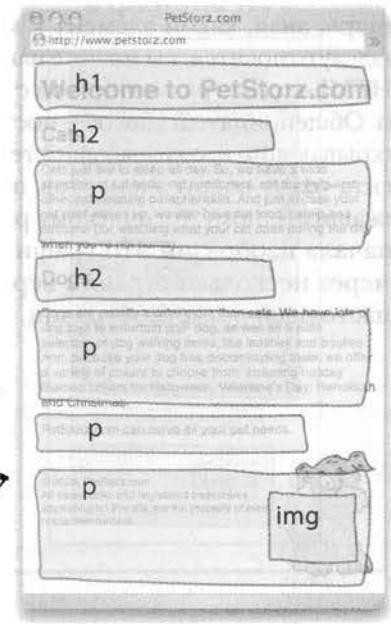
Джим: Кажется, это как раз то, что нужно!

Фрэнк: Точно. Ребята, давайте я сначала покажу вам, как можно разбить страницу на логические разделы, а потом расскажу, что я знаю про ** **...

Выясним, как можно разбить страницу на логические разделы

Посмотрите на веб-страницу, расположенную справа: это страница для PetStorz.com. Сейчас мы рассмотрим, как можно добавить ей дополнительную структуру, определив несколько логических разделов и затем заключив каждый в элемент ** **.

Мы нарисовали схему страницы PetStorz.

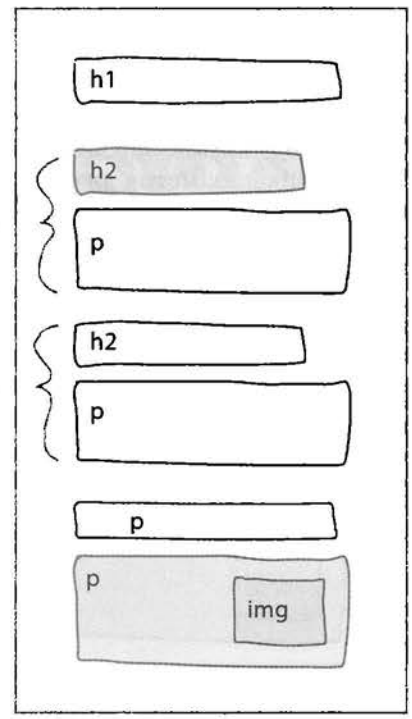


Эта страница выглядит привлекательно: множество заголовков, абзацев и рисунок.

Однако, сконцентрировавшись лишь на структуре, вы не сможете много сказать о странице в целом. Какие элементы входят в верхний колонтитул? Есть ли на странице нижний колонтитул? Каковы области содержимого?

Определение логических разделов

Итак, определим логические разделы этой страницы. Что такое логический раздел? Это просто группа взаимосвязанных элементов. Например, на странице PetStorz.com есть несколько элементов, имеющих отношение к котам, и несколько — к собакам. Давайте посмотрим.

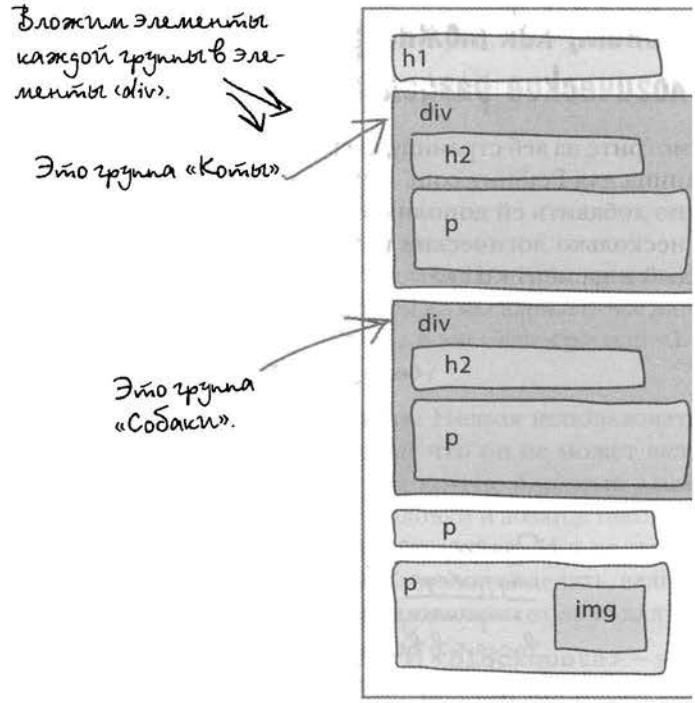


Страница PetStorz имеет две основные области содержимого, одна — для котов, другая — для собак. В ней есть также несколько других областей, но мы вернемся к этому позже.

В данном случае оба раздела состоят из двух элементов: заголовка и абзаца. Но очень часто такие группы содержат намного больше элементов.

Использование для обозначения разделов

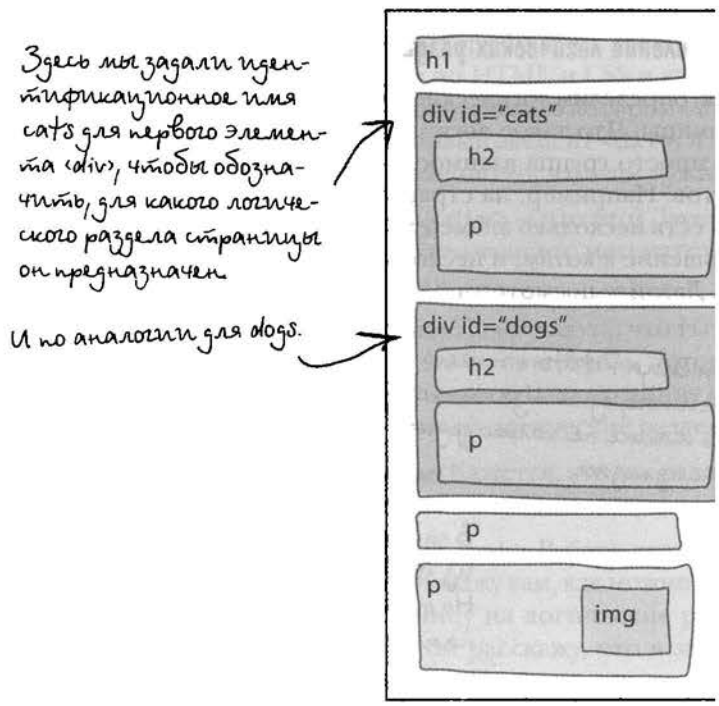
Теперь, зная, какой элемент к какому разделу относится, вы можете написать XHTML-код для разметки этой структуры. Общепринятый способ – поставить открывающий и закрывающий теги элемента ** ** вокруг элементов, принадлежащих одному логическому разделу. Сначала изобразим это графически, а через несколько страниц вернемся к настоящей разметке страниц.



Присваивание меток элементам

Просто вложив несколько элементов в ** **, вы говорите, что они принадлежат одной группе. Но вы не использовали никаких меток для обозначения того, что это за группы, верно?

Хороший способ это сделать – задать атрибут **id**. Это позволит обеспечить элемент ** ** уникальным идентификатором. Например, зададим элементу ** **, содержащему информацию про котов, идентификатор **cats**, а элементу ** ** с информацией про собак – идентификатор **dogs**.

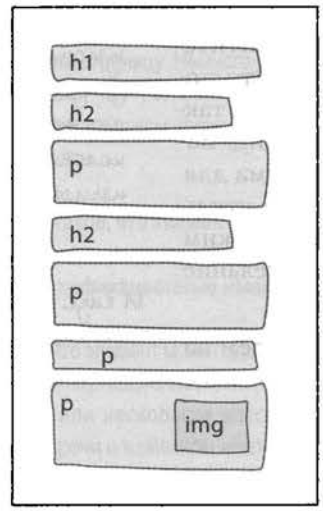


МОЗГОВОЙ ШТУРМ

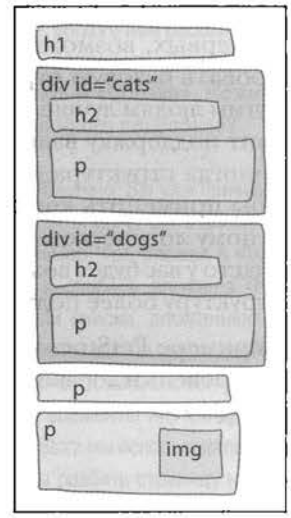
По рекомендации генерального директора Starbuzz вас попросили дать консультацию по изменению стиля оформления главной страницы PetStorz. Как быстро вы поймете структуру, если вам покажут только страницу № 1?

Как насчет страницы № 2?

Страница № 1



Страница № 2

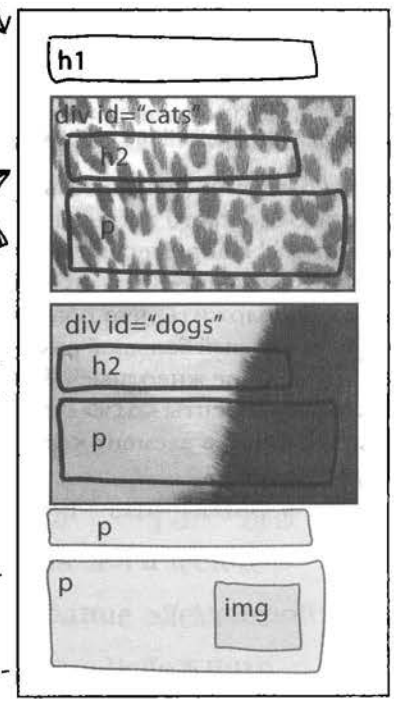


Немного поработаем над стилем

Итак, вы задали кое-какую логическую структуру страницы PetStorz и разметили ее, присвоив уникальный идентификатор каждому элементу ** **. Этого достаточно, чтобы начать работать над стилем группы элементов, вложенных в ** **.

Теперь элементы имеют свой стиль оформления.

Фон, заданный для , также используется для всех его элементов.



Элементы, вложенные в , как и любые дочерние элементы, унаследуют и другие свойства от (например, размер шрифта, цвет и т. д.).

Здесь у нас есть два правила: по одному для каждого элемента . Каждый выбирается через селектор id.

```
#cats {
   background-image: url(leopard.jpg);
}

#dogs {
   background-image: url(mutt.jpg);
}
```

В каждом правиле задается свойство background-image. Для группы «Коты» используется рисунок с изображением леопарда, а для группы «Собаки» — с изображением маленькой собачки.

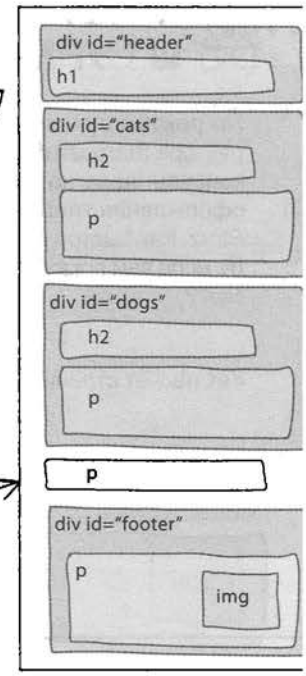
Усовершенствование и детализация структуры

Есть несколько причин, по которым вы захотите еще немного усовершенствовать структуру своих страниц с помощью элементов ** **. Во-первых, возможно, вы захотите детализировать базовую структуру, что поможет другим людям лучше ее понять, а также облегчит поддержку ваших страниц. Во-вторых, иногда структура необходима для того, чтобы применить конкретный стиль к конкретному логическому разделу. Таким образом, часто у вас будет возникать желание сделать структуру более подробной.

Итак, на примере PetStorz мы перейдем на следующий уровень и добавим еще несколько элементов ** **.

На этот раз мы добавили элемент с идентификационным именем, указывающим на то, что это верхний колонтитул страницы

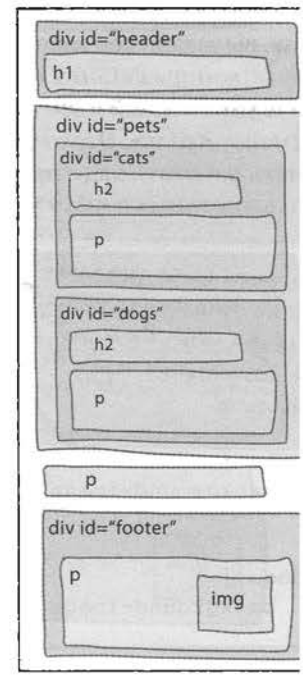
И еще один — для нижнего колонтитула страницы



Структуризация страницы с помощью элементов может даже помочь продумать ее дизайн. Например, действительно ли этот одинокий элемент <p> должен быть здесь?

Добавление вложенной структуры

Однако не останавливайтесь на достигнутом. Достаточно часто встречаются вложенные структуры. Например, на странице PetStorz есть два раздела: «Коты» и «Собаки». Логично предположить, что оба раздела вместе образуют новый большой раздел под названием «Домашние животные». Итак, мы можем вложить элементы ** ** с идентификаторами **cat** и **dog** в элемент ** ** с идентификатором **pets**.



Теперь мы разметили XHTML-код таким образом, что на странице появился логический раздел pets, содержащий информацию о домашних животных. Этот раздел включает в себя два логических подраздела: cats — о котах и dogs — о собаках.

Часто Задаваемые Вопросы

В: Итак, элемент выступает в роли контейнера, в который вы можете сложить элементы, чтобы хранить их в одном месте?

О: Да, конечно. Мы часто описываем элементы как контейнеры. Хотя нельзя сказать, что они выступают в роли только логических контейнеров, которые можно использовать для хранения набора взаимосвязанных элементов (как, например, элементы логического раздела cat). Когда в следующей главе мы начнем задавать элементам стиль и использовать их для позиционирования, вы увидите, что они могут выступать и в роли графических контейнеров.

В: Если я уже разметил страницу на абзацы, заголовки и т. п., нужно ли после этого добавлять верхний уровень структуры с помощью элементов ?

О: И да, и нет. Структуру необходимо создавать для конкретных целей, а не ради структуры самой по себе. Если хотите грамотно создавать страницы, всегда следите, чтобы ваша структура была настолько проста, насколько это возможно. Например, если создание общего раздела «Домашние животные», содержащего подразделы «Коты» и «Собаки», несет какую-то пользу, то, бесспорно, его нужно создавать. Однако если никакой пользы вы от этого не получаете, то просто усложняете страницу. Немного поработав с элементами , вы сами начнете чувствовать, когда и в каком количестве их нужно использовать.

В: Бывает ли такое, что элементы помещают в класс, вместо того чтобы давать им идентификационные имена?

О: Вспомните, что элемент может одновременно иметь идентификационное имя и быть членом одного или нескольких классов. Таким образом, речи о взаимном исключении в данном случае не идет. Конечно, бывают ситуации, когда вы создаете элементы и помещаете их в классы. Допустим, у вас на странице есть несколько разделов с музыкальными альбомами. Можете поместить все элементы, составляющие один альбом, в элемент (и так для каждого альбома), а затем поместить все эти в класс под названием albums. Таким образом, вы определите, где находятся альбомы, а затем сможете оформить их в одном стиле, используя тот факт, что они составляют класс. В то же время вы можете дать каждому альбому идентификационное имя, чтобы он мог иметь собственный стиль, который применяется только для него.

В: Мне было не все понятно об элементе , когда о нем рассказывалось на примере разделов «Домашние животные», «Коты» и «Собаки». Можете объяснить это немного подробнее?

О: Конечно. Вы уже привыкли к тому, что элементы могут быть вложенными, верно? Например, <p> вложен в <body>, который, в свою очередь, вложен в <html>. Вы даже видели списки, вложенные друг в друга. Элемент на самом деле ничем не отличается от остальных: вы просто вкладываете одни элементы внутрь других. В примере с PetStorz мы использовали элементы , чтобы разбить страницу на большие части (подразделы «Коты» и «Собаки» вложены в раздел «Домашние животные»). Вы также можете использовать элементы , чтобы вложить подраздел «Пиво» в раздел «Напитки», который, в свою очередь, вложен в раздел «Меню».

Но самый лучший способ понять, зачем один элемент вкладывать в другой, — опробовать все самим. Просто не забывайте эту информацию, и очень скоро вы увидите пример ситуации, когда вам понадобится такое вложение.

Используйте на страницах элементы , но не злоупотребляйте ими. Задавайте дополнительную структуру там, где это поможет вам разбить страницу на логические разделы и сделать ее более понятной. Использование элементов только для создания детальной структуры усложнит ваши страницы и не принесет никакой пользы.

Тем временем вернемся в гостевую

Отлично, хватит изучать теорию по элементам ** **, давайте используем их для гостевой. Вспомните: нам нужно сгруппировать все элементы, имеющие отношение к напиткам, а затем стилизовать их так, чтобы они выглядели как меню для посетителей. Итак, откройте файл `lounge.html` из папки `chapter11/lounge`, найдите элементы, имеющие отношение к напиткам, и окружите их открывающими и закрывающими тегами элемента ** **.

Это открывающий тег и мы присваиваем ему идентификатор elixirs.

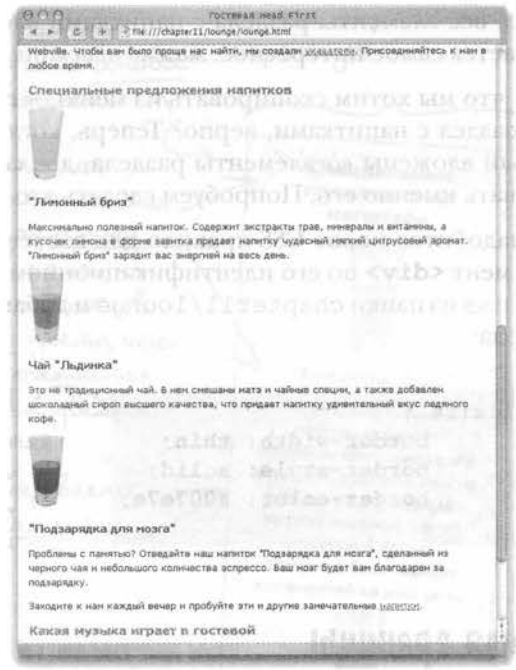
Не забывайте, мы показываем только фрагмент XHTML-кода. Открыв файл lounge.html, вы увидите всю разметку страницы.

```
<div id="elixirs">
    <h2>Специальные предложения напитков</h2>

    <p>
        <img src="images/yellow.gif" alt="Напиток «Лимонный бриз»" />
    </p>
    <h3>«Лимонный бриз»</h3>
    <p>
        Максимально полезный напиток. Содержит экстракты трав, минералы и витамины,
        а кусочек лимона в форме завитка придает напитку чудесный мягкий цитрусовый
        аромат. «Лимонный бриз» зарядит вас энергией на весь день.
    </p>

    <p>
        <img src="images/chai.gif" alt="Напиток Чай «Льдинка»" />
    </p>

    <h3>Чай «Льдинка»</h3>
    <p>
        Это не традиционный чай. В нем смешаны матэ и чайные специи,
        а также добавлен шоколадный сироп высшего качества, что
        придает напитку удивительный вкус ледяного кофе.
    </p>

    <p>
        <img src="images/black.gif" alt="Напиток «Подзарядка для мозга»" />
    </p>

    <h3>«Подзарядка для мозга»</h3>
    <p>
        Проблемы с памятью? Отведайте наш напиток «Подзарядка для
        мозга», сделанный из черного чая и небольшого количества
        эспрессо. Ваш мозг будет вам благодарен за подзарядку.
    </p>

    <p>
        Заходите к нам каждый вечер и пробуйте эти
        и другие замечательные
        <a href="beverages/elixir.html"
           title="Напитки гостевой Head First">напитки</a>.
    </p>
</div>
```

А это закрывающий тег.

Тест для

Это было просто, не так ли? Теперь, когда у нас есть более структурированная страница, откроем ее в браузере и посмотрим, как она отобразится.



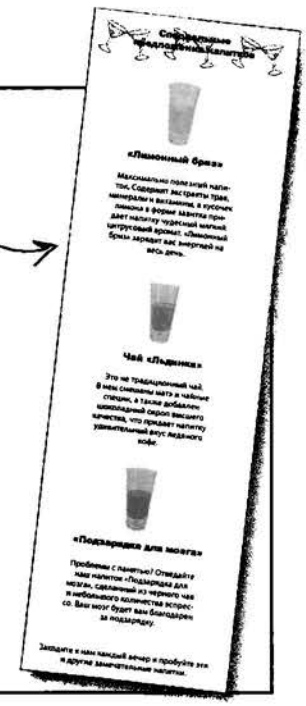
Хммм... вообще ничего не изменилось! Но это хорошо: — это чистая структура, и он не имеет никакого «внешнего вида» на странице.

Надо сказать, что — блочный элемент и вы можете применять к нему любые стили. Итак, если вы знаете, как определять стиль для блочных элементов (а вы это знаете), то сможете стилизовать и элемент .

МОЗГОВОЙ ШТУРМ

Напоминаем: нужно изменить оформление раздела со специальными предложениями напитков таким образом, чтобы он выглядел как меню, которое подают клиентам.

До того как мы отвлеклись на теорию об элементе ** **, мы пытались выяснить, как же нарисовать границу вокруг всех элементов, имеющих отношение к напиткам. Теперь, когда у нас в файле `lounge.html` уже есть элемент ** **, что нужно сделать, чтобы нарисовать эту границу?



Добавление границы

Отлично, все элементы раздела с напитками вложены в элемент ** **, и начинается самое интересное: *можно приступить к стилизации.*

Первое, что мы хотим скопировать из меню, — граница, которая окружает весь раздел с напитками, верно? Теперь, когда создан элемент ** **, в который вложены *все* элементы раздела, для создания границы можно стилизовать именно его. Попробуем сделать это прямо сейчас.

Вам понадобится новое CSS-правило для гостевой, которое будет выбирать элемент ** ** по его идентификационному имени. Откройте файл `lounge.css` из папки `chapter11/lounge` и добавьте это правило в самый конец кода:

```
#elixirs {
        border-width: thin;
        border-style: solid;
        border-color: #007e7e;
}
```

Добавьте это в самый конец вашего CSS-кода. Правило позволяет выбрать элемент по его идентификатору elixirs и добавить тонкую сплошную границу нашего любимого аквамаринового цвета.

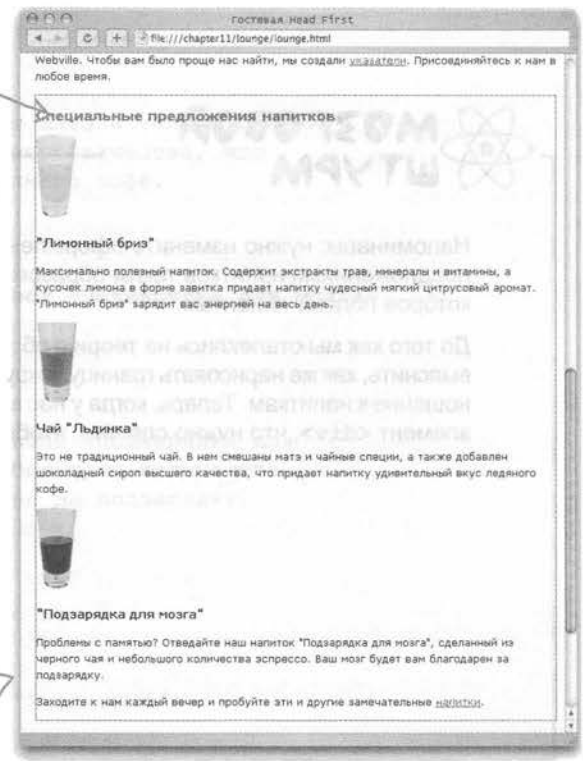
Тест для границы

После того как вы добавили новое правило, сохраните файл и обновите страницу `lounge.html`.

Это граница, которую вы только что добавили для элемента с идентификатором elixirs.

Вы задали видимую границу для элемента , но у него все еще нет отступов и полей. Их тоже нужно добавить.

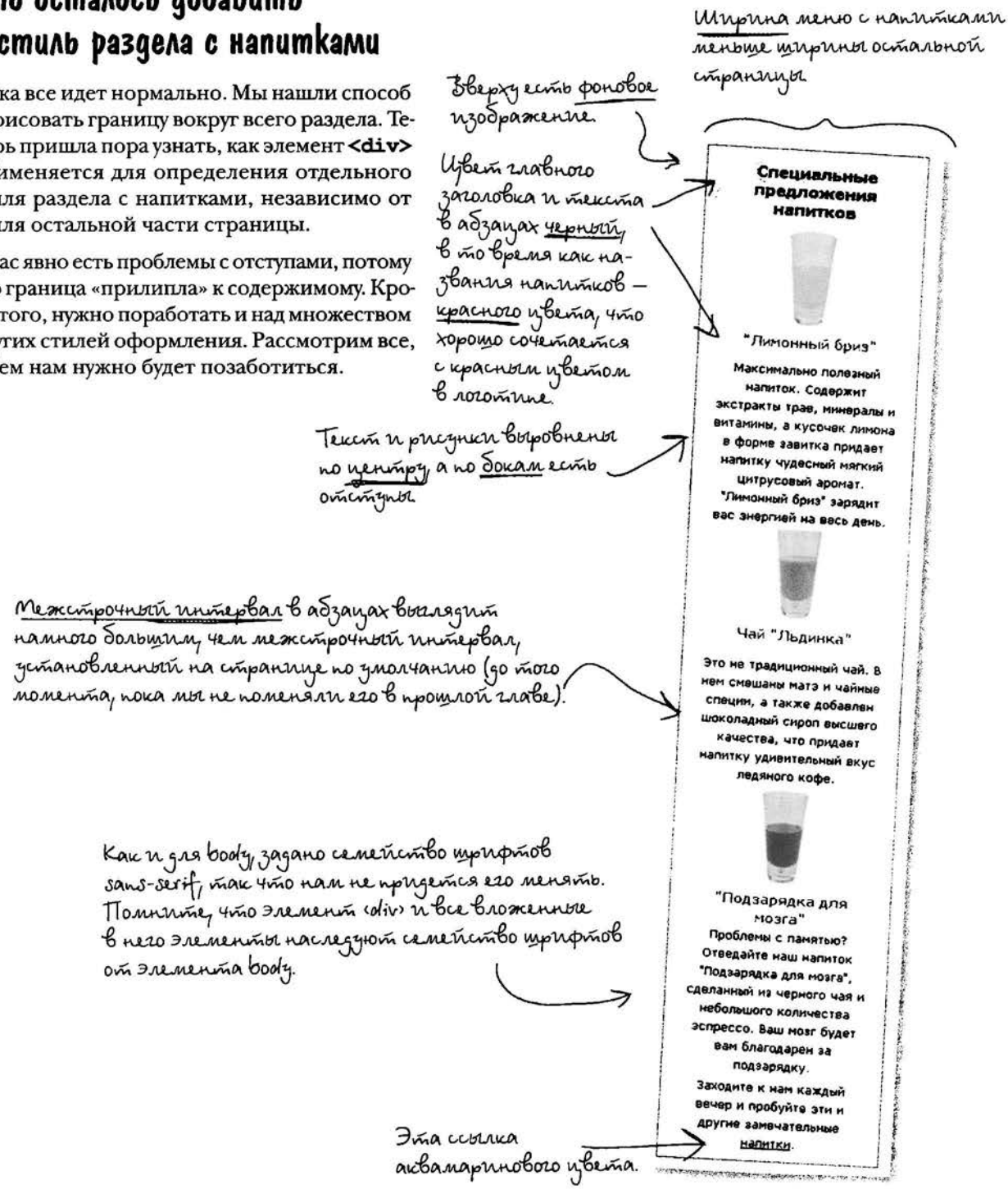
Обратите внимание, что граница нарисована вокруг всех элементов, вложенных в . Как и любой другой элемент, его можно представить в виде блока, поэтому, когда вы добавляете границу, она рисуется вокруг содержимого, каковым являются все элементы внутри .



Что осталось добавить в стиль раздела с напитками

Пока все идет нормально. Мы нашли способ нарисовать границу вокруг всего раздела. Теперь пришла пора узнать, как элемент ** ** применяется для определения отдельного стиля раздела с напитками, независимо от стиля остальной части страницы.

У нас явно есть проблемы с отступами, потому что граница «прилипла» к содержимому. Кроме того, нужно поработать и над множеством других стилей оформления. Рассмотрим все, о чем нам нужно будет позаботиться.



План действий

В стиле оформления придется изменить многое, так что давайте сначала составим план действий. Вот что нам нужно будет сделать.

- ❑ Во-первых, мы изменим ширину элемента ** ** с идентификатором **elixirs**, сделав ее меньше.
- ❑ Затем мы исправим те стили, с которыми уже хорошо знакомы, такие как отступ и фоновое изображение. Мы также поэкспериментируем с выравниванием текста, с которым раньше не работали.
- ❑ И наконец, нам останется изменить межстрочные интервалы и цвета заголовков. Для этого придется еще кое-что узнать о селекторах CSS.

Предстоит много работы, так что начнем прямо сейчас.

Поработаем над шириной раздела с напитками

Нам нужно, чтобы раздел с напитками был достаточно узким, как меню, которое подают в гостевой, и занимал примерно 1/4 ширины обычного окна браузера. Итак, если большинство людей устанавливают ширину окна браузера 800 пикселов, то нам нужно где-то 200 пикселов. Вы уже устанавливали ширину отступов, границ и полей, но еще никогда не задавали ширину самого содержимого. Для этого используется свойство **width**:

```
#elixirs {
        border-width: thin;
        border-style: solid;
        border-color: #007e7e;
        width: 200px;
}
```

С помощью свойства width можно задавать ширину области содержимого элемента. В данном примере мы задаем ширину 200 пикселов.

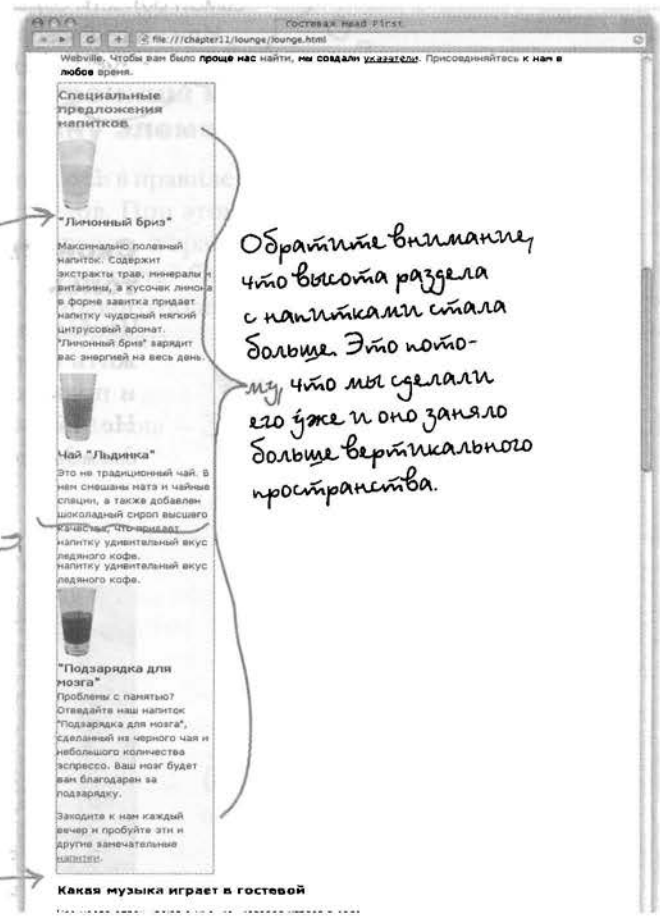
Мы задаем ширину для элемента с идентификатором elixirs. Раздел с напитками будет иметь ширину 200 пикселов, и правила разметки браузера сработают так, чтобы поместить все элементы, вложенные в , в столбец такой ширины.

Попробуйте это сделать. Откройте файл `lounge.css` и добавьте это правило в самый конец.

Протестируем ширину раздела с напитками

Сохраните CSS-файл и обновите страницу `lounge.html`. Вы увидите, что раздел с напитками стал намного у́же благодаря ширине, которую вы задали для него. Ширина содержимого элемента ** ** сейчас составляет 200 пикселов. Однако здесь есть еще кое-что интересное, на что нужно обратить внимание.

Теперь все содержимое элемента с идентификатором elixirs умещается в столбце шириной 200 пикселов. Это значение не изменится, даже если вы сделаете ширину окна браузера больше или меньше. Проверьте сами!



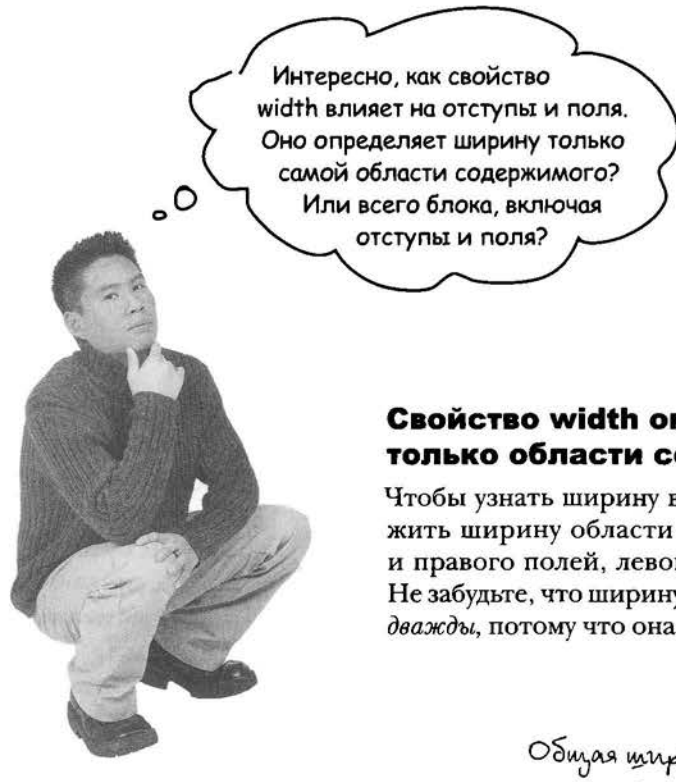
200 пикселов

Обратите внимание, что высота раздела с напитками стала больше. Это потому, что мы сделали его у́же и оно заняло больше вертикального пространства.

Сравните поведение с поведением других элементов при изменении ширины окна браузера. Абзацы автоматически меняют свой размер, чтобы соответствовать новой ширине окна браузера. Чуть позже мы поговорим об этом подробнее.

МОЗГОВОЙ ШТУРМ

Вы можете изменить размер окна браузера таким образом, чтобы его ширина стала меньше, чем ширина раздела с напитками? Одни браузеры не позволят вам этого сделать, другие — позволят. Если вы можете сделать окно браузера у́же раздела с напитками, то сравните текст внутри этого раздела с остальным текстом страницы. Абзацы изменяют свою ширину независимо от того, насколько узким становится окно браузера, но ширина элемента ** ** с идентификатором **elixirs** никогда не станет меньше 200 пикселов.



Свойство width определяет размер только области содержимого.

Чтобы узнать ширину всего блока, нужно сложить ширину области содержимого, левого и правого полей, левого и правого отступов. Не забудьте, что ширину границы нужно учесть *дважды*, потому что она есть и слева, и справа.

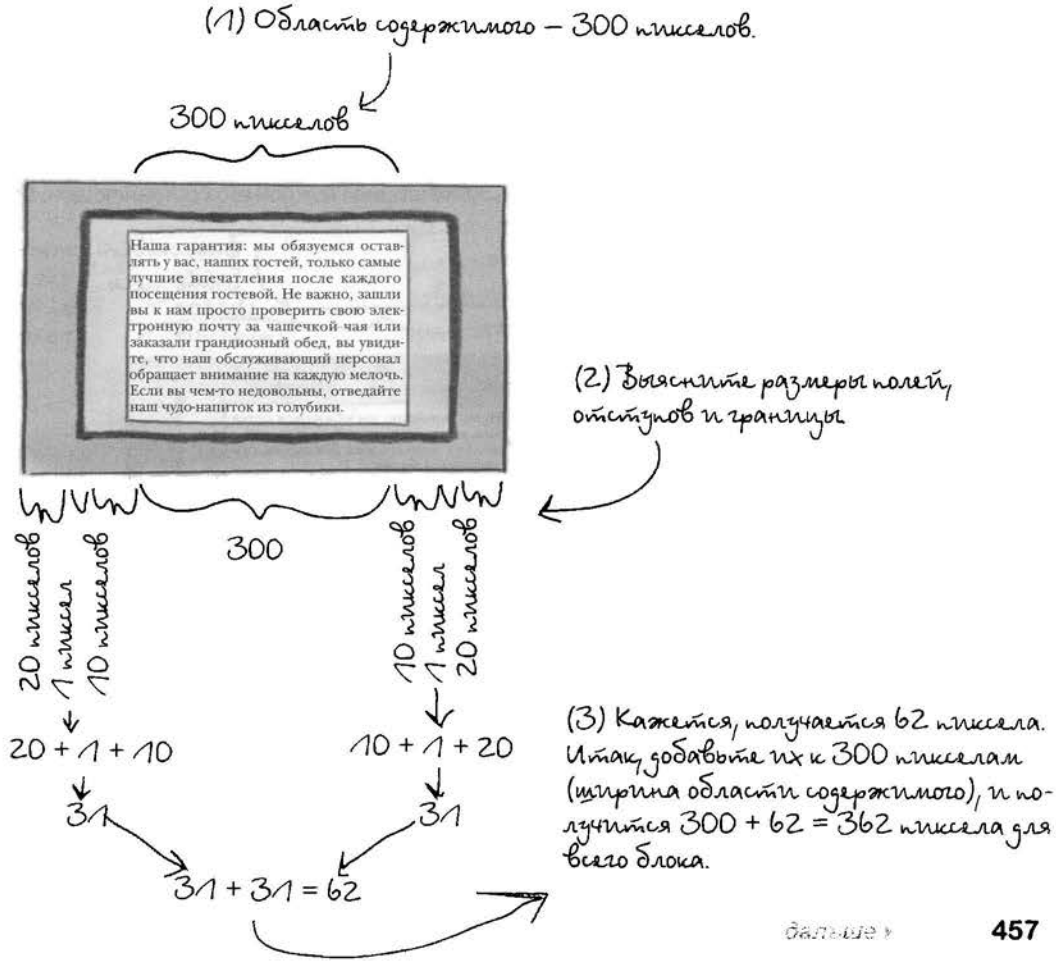




Хорошо, а как же задать ширину для всего элемента сразу?

Никак. Вы определяете ширину области содержимого, отступа, границы и полей. Все это вместе и составит ширину элемента.

Допустим, используя свойство **width** в правиле CSS, вы установили ширину области содержимого 300 пикселов. При этом ширина полей составляет 20 пикселов, отступов — 10 пикселов, а границы — 1 пиксел. Какова же ширина всего блока для этого элемента? Она равняется сумме ширины области содержимого, левого и правого полей, левого и правого отступов и границы. Посмотрим, как ее посчитать.



Часто Задаваемые Вопросы

В: Если я сам не укажу ширину элемента, откуда она возьмется?

О: По умолчанию для блочных элементов используется автоширина. Это означает, что элемент будет растянут ровно настолько, сколько есть свободного места. Если вы представите любую из созданных веб-страниц, вам покажется, что каждый блочный элемент растянут на всю ширину окна браузера, и это именно так. Просто запомните, что благодаря автоширине элемент (учитывая отступы, границы и поля) занимает все свободное пространство. Более подробно мы поговорим об этом в следующей главе.

В: А если у меня нет отступов, границ и полей?

О: Тогда ширина содержимого будет *равна* ширине блока. Если ширина области содержимого равна 300 пикселам и у вас нет отступов, границ и полей, то ширина всего блока тоже будет равняться 300 пикселам.

В: Какие есть способы задания ширины?

О: Вы можете указать реальный размер в пикселах или процентах. Если вы используете второй вариант, то ширина будет определена как процент от ширины того элемента, в котором содержится ваш (это может быть элемент <body>, и т. д.).

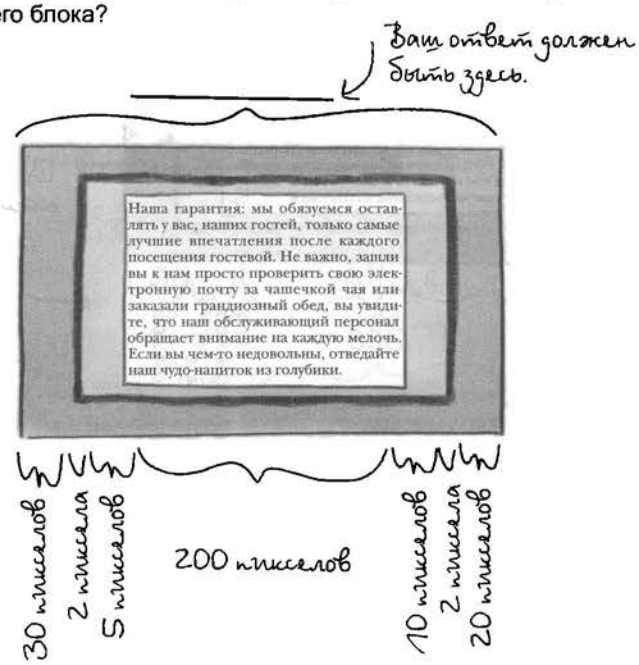
В: А как насчет высоты?

О: Высота элементов по умолчанию устанавливается автоматически, и браузер растягивает область содержимого в вертикальном направлении так, чтобы отобразилось все содержимое. Взгляните на размер раздела с напитками после того, как мы задали ему ширину 200 пикселов, и вы увидите, что стал намного выше.

Вы, конечно, можете и сами задать высоту, но тогда вы рискуете тем, что нижняя часть содержимого будет обрезана, если высота элемента недостаточно велика, чтобы уместить его полностью. Вообще, лучше не задавать высоту элементов, и пусть по умолчанию применяется автоширина. Об этом мы тоже более подробно поговорим в следующей главе.

Возьми в руку карандаш

Это блок, в котором помечена ширина каждой его составляющей. Какова ширина всего блока?



Стилизация раздела с напитками

Мы закончили с шириной раздела. Что нам осталось сделать?

- [x] Во-первых, мы изменим ширину элемента ** ** с идентификатором **elixirs**, сделав ее меньше.
- [ ] Затем мы исправим те стили, с которыми уже хорошо знакомы, такие как отступ и фоновое изображение. Мы также поэкспериментируем с выравниванием текста, с которым раньше не работали.
- [ ] И наконец, нам останется изменить межстрочные интервалы и цвета заголовков. Для этого придется еще кое-что узнать о селекторах CSS.

Переходим к следующему этапу.

Теперь мы сконцентрируемся на базовых стилях: изменим отступ, выравнивание текста, а также применим для элемента ** ** с идентификатором **elixirs** фоновый рисунок с изображением бокалов с коктейлями. Вы уже знаете, как работает большинство из этих стилей, так что давайте бегло просмотрим CSS-код.

Мы хотим применить это правило стиля к элементу с идентификатором elixirs, чтобы оно повлияло только на сам и элементы внутри его, но не на всю страницу.

```
#elixirs {
   border-width:      thin;
   border-style:      solid;
   border-color:      #007e7e;
   width:             200px;

   padding-right:     20px;
   padding-bottom:    20px;
   padding-left:      20px;

   margin-left:       20px;

   text-align:        center;

   background-image:  url(images/cocktail.gif);
   background-repeat: repeat-x;
}
```

По умолчанию для используется отступ шириной 0 пикселов. Увеличим его, чтобы создать немного свободного пространства вокруг области содержимого. Обратите внимание, что мы не добавляем отступ сверху, потому что там и так достаточно свободного места благодаря размеру поля, который используется по умолчанию, и заголовка <h2> (вернитесь к предыдущему тестированию страницы и вы увидите, что над <h2> достаточно свободного места). Однако нам необходимо добавить отступы справа, снизу и слева.

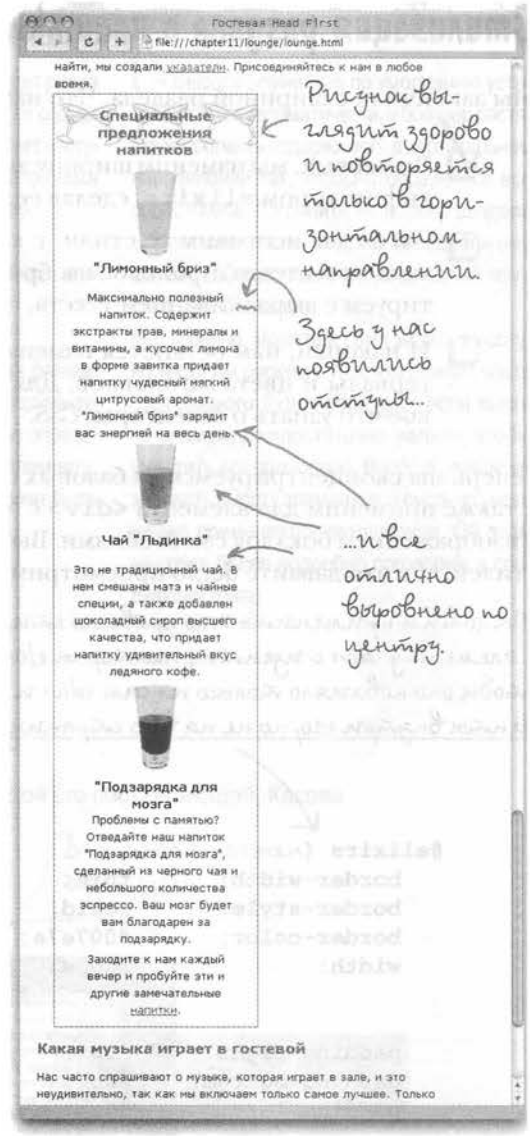
Мы расширили поле слева, чтобы немного сместить раздел с напитками вправо. Нам это пригодится позже...

Используйте свойство text-align для блочных элементов, чтобы выровнять содержащийся в них текст. В данном случае мы выравниваем текст по центру.

И наконец, мы указываем, какое изображение будет использовано в качестве фона, в данном случае это рисунок с коктейлями. Мы устанавливаем значение repeat-x свойства background-repeat, в результате чего изображение будет повторяться только по горизонтали.

Тест для новых стилей

Теперь пришло время добавить новые свойства в файл `lounge.css` и обновить страницу. Посмотрим, что изменилось: заголовки, рисунки и текст в элементе ** ** теперь выровнены по центру и вокруг них появилось чуть больше свободного пространства благодаря отступам. Мы немного украсили верхнюю часть раздела повторяющимся рисунком с коктейлями.





Хорошее замечание. Но суть в том, что свойство **text-align** выравнивает *все строчное содержимое* блочного элемента. Итак, в данном примере мы задаем это свойство для блочного элемента ** **, и в результате все его строчное содержимое великолепным образом выравнивается. Просто помните, что **text-align** действует вопреки своему имени и работает со всеми строчными элементами. И еще запомните следующее: свойство **text-align** можно задавать только для блочных элементов. Оно не сработает, если применить его напрямую к строчному элементу (например к **<img>**).



Хорошо подмечено. Весь текст внутри элемента ** ** вложен в другие блочные элементы, но он все же выровнен. Это происходит потому, что блочные элементы *наследуют* свойство **text-align** от элемента ** **. Вместо того чтобы элемент ** ** сам выравнивал текст в абзацах и заголовках (чего он сделать не может, так как это блочные элементы), абзацы и заголовки наследуют значение **center** свойства **text-align** и затем выравнивают *свое содержимое* по центру.

Что это дает? Вы можете заключить какой-то раздел страницы в элемент ** ** и стилизовать уже его, вместо того чтобы возиться с каждым отдельным элементом. Конечно, помните, что не все свойства по умолчанию являются наследуемыми, так что этот способ не будет работать для всех свойств.

Возьми в руку карандаш

Итак, теперь, когда вы умеете хорошо работать с шириной блока, вычислим общую ширину раздела с напитками. Мы знаем, что ширина области содержимого равняется 200 пикселам. Кроме того, заданы левый и правый отступы, а также граница, ширину которой мы определили как `thin`. Просто предположите, что толщина такой границы равна 1 пикселу (для большинства браузеров). Как насчет полей? Мы задали значение для ширины левого поля, но не задали для правого, то есть по умолчанию она равняется 0 пикселов.

Вот все свойства, имеющие отношение к ширине. Ваша задача — вычислить общую ширину элемента ** **.

```
border-width:   thin;

width:          200px;

padding-right:  20px;
padding-bottom: 20px;
padding-left:   20px;

margin-left:    20px;
```



Мы почти закончили...

Мы уже практически закончили работу над разделом с напитками. Что осталось сделать?

- [x] Во-первых, мы изменим ширину элемента ** ** с идентификатором **elixirs**, сделав ее меньше.
- [x] Затем мы исправим те стили, с которыми уже хорошо знакомы, такие как отступ и фоновое изображение. Мы также поэкспериментируем с выравниванием текста, с которым раньше не работали.
- [ ] И наконец, нам останется изменить межстрочные интервалы и цвета заголовков. Для этого придется еще кое-что узнать о селекторах CSS.

← Переходим к последнему шагу.

Кажется, все очень просто, так? В конце концов, вы уже делали это раньше. По сути, опираясь на имеющиеся знания, вы можете просто задать стили для элемента ** **, и они будут унаследованы. С этим вы справитесь достаточно быстро.



Фрэнк: Да, это интересно. Главный заголовок **<h2>** раздела с напитками имеет аквамариновый цвет, потому что для **<h2>** в CSS уже есть правило. Но нам нужно, чтобы он был черным. Кроме того, в разделе с напитками есть заголовки **<h3>**, которые должны быть красными.

Джим: Нет проблем, нам нужно просто добавить несколько новых правил.

Фрэнк: Не торопись. Если мы поменяем правило для **<h2>** или добавим правило для **<h3>**, то изменится цвет заголовков на всей странице. А нам нужно поменять их только в разделе с напитками.

Джим: О, хорошее замечание. Хммм... Ну, мы можем использовать два класса.

Фрэнк: Такой вариант сработает, хотя нельзя назвать его наилучшим. Каждый раз, когда вы будете добавлять новый заголовок в элемент ** ** с идентификатором **elixirs**, придется указывать, что он является членом класса.

Джим: Такова жизнь.

Фрэнк: На самом деле, Джим, перед тем как использовать классы, лучше рассмотреть вариант использования селекторов потомков. Я думаю, они в данном случае будут более уместны.

Джим: Селекторы потомков?

Фрэнк: Именно. Они предоставляют способ выбирать элементы, например, следующим образом: выбрать элемент **<h2>**, но только если он находится внутри элемента ** ** с идентификатором **elixirs**.

Джо: Не совсем понятно.

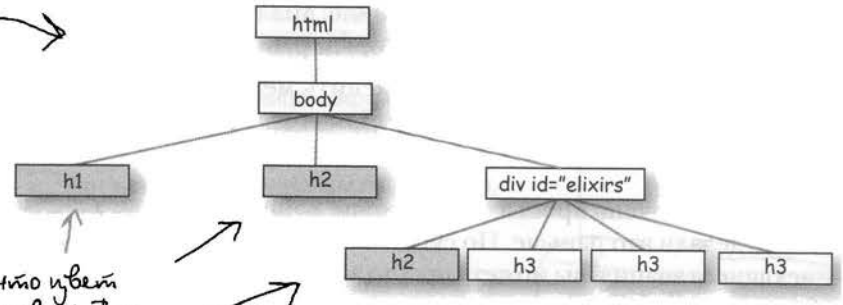
Фрэнк: Хорошо, разберемся в этом пошагово.

Что мы пытаемся сделать

Посмотрим, что нужно сделать с цветами заголовков.

Что мы имеем на данный момент

Это элементы для главных заголовков в XHTML-коде для гостиевой.



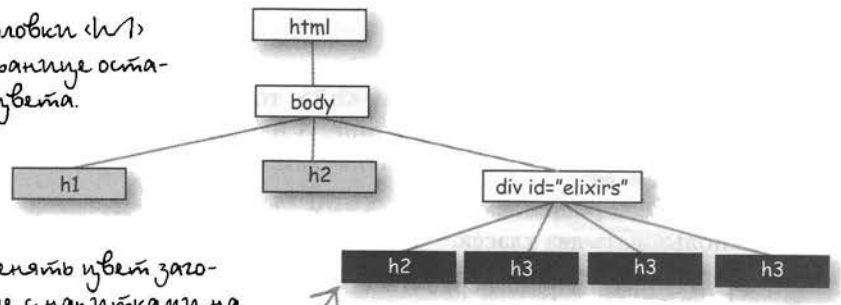
На данный момент CSS говорит, что цвет элементов <h1> и <h2> аквамариновый. Все элементы <h1> и <h2>, даже находящиеся внутри элемента с идентификатором elixirs, имеют этот цвет.

```
h1, h2 {
        color: #007e7e;
}
```

Это правило, определяющее цвет заголовков <h1> и <h2> в файле lounge.css.

Что мы хотим получить

Мы хотим, чтобы заголовки <h1> и <h2> на главной странице остались аквамаринового цвета.



Кроме того, мы хотим поменять цвет заголовков <h2> и <h3> в разделе с напитками на черный и красный соответственно.

Но если мы изменим существующее правило для <h2>, то это повлияет на цвет шрифта всех заголовков <h2> главной страницы. И если мы добавим новое правило для <h3>, то все заголовки такого уровня, которые будут добавлены на главную страницу позже, будут красного цвета, а это не то, что нам нужно. Сейчас мы могли бы указать класс, как предлагает Джим, но все же сначала воспользуемся идеей Фрэнка.

```
h1, h2 {
        color: #007e7e;
}
```

?

Нам нужен способ выбрать потомков

На данный момент мы не рассмотрели способ сказать CSS, что хотим выбрать только те элементы, которые являются *потомками* определенных элементов. Это то же самое, как если бы вы сказали, что хотите, чтобы ваше наследство перешло детям одного сына или дочери. Рассмотрим, как пишутся селекторы потомков.

Поставьте пробел между именем родительского элемента и именем потомка.

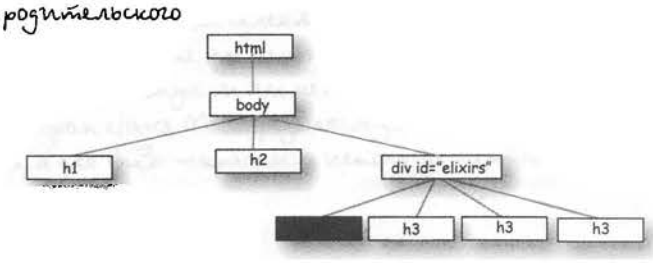
Это родительский элемент h2.

Это потомок.

```
div h2 {
    color: black;
}
```

Оставшаяся часть правила пишется как обычно.

Это правило говорит выбрать любой элемент <h2>, являющийся потомком элемента .



Вот элемент, который это правило выбирает в гостевой.

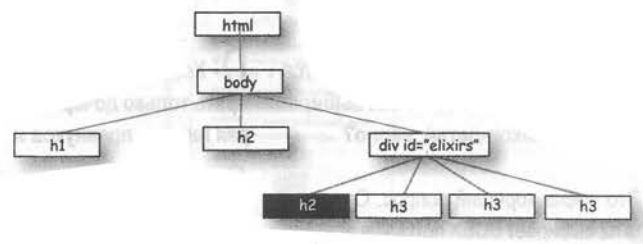
Теперь единственная проблема с этим правилом заключается в том, что кто-нибудь может создать еще один элемент ** ** в файле `lounge.html`, тогда он унаследует заголовки **<h2>** черного цвета, даже если этого не хочет. Но у нашего элемента ** ** есть идентификатор **elixirs**, так что используем его, чтобы конкретно указать, какие потомки нам нужны.

Теперь родительский элемент — это элемент с идентификатором elixirs.

Это его потомок.

```
#elixirs h2 {
    color: black;
}
```

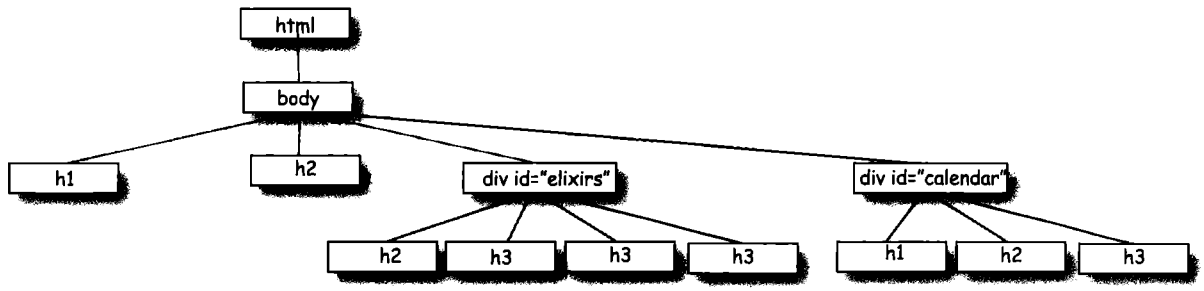
Правило говорит выбрать любой элемент <h2>, являющийся потомком элемента с идентификатором elixirs.



Правило выбирает тот же элемент, но оно более конкретное. Таким образом, все будет нормально, если мы добавим на страницу еще один с элементами <h2> внутри, потому что это правило выбирает только те элементы <h2>, которые находятся в с идентификатором elixirs.

Возьми в руку карандаш

Ваша очередь. Напишите селектор, который будет выбирать только элементы **<h3>** внутри элемента ** **, относящегося к напиткам. В правиле задайте значение **#d12c47** свойства **color**. Кроме того, в приведенной ниже схеме расставьте метки над элементами, которые выбраны.



Часто Задаваемые Вопросы

В: Под потомками обычно понимаются дети, внуки, правнуки. Здесь мы выбираем только потомков-детей, верно?

О: Это очень хороший вопрос. Селектор #elixirs h2 выбирает ВСЕХ потомков элемента с идентификатором elixirs, так что <h2> может являться непосредственно дочерним элементом элемента или может быть вложен в <blockquote> или другой (что сделает его внуком) и т. д. Итак, селектор потомка выбирает и элементы <h2>, вложенные в другие элементы, независимо от того, как глубоко они вложены.

В: Хорошо, а существует ли способ выбрать только дочерний элемент (не выбирая внуков, правнуков и т. д.)?

О: Да. Например, вы можете написать #elixirs > h2, чтобы выбрать только те <h2>, которые являются непосредственными детьми элемента с идентификатором elixirs.

В: А что, если мне нужно что-то более сложное, например <h2>, который является дочерним для элемента <blockquote>, вложенного в элемент с идентификатором elixirs?

О: Все делается аналогично. Просто указывайте больше потомков, например, так:

```
#elixirs blockquote h2 {
    color: blue;
}
```

Таким образом будут выбраны элементы <h2>, являющиеся потомками элементов <blockquote>, которые, в свою очередь, являются потомками элемента с идентификатором elixirs.

Изменение цвета для заголовков раздела с напитками

Теперь, когда вы знаете все о селекторах потомков, определим черный цвет для заголовков **<h2>**, находящихся внутри раздела с напитками, и красный — для **<h3>**.

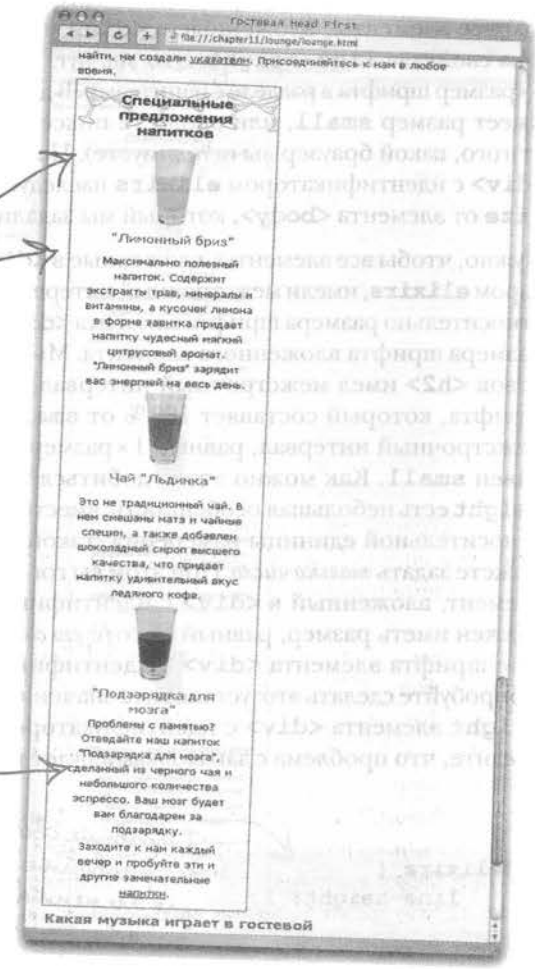
```
#elixirs h2 {
    color: black;
}

#elixirs h3 {
    color: #d12c47;
}
```

Здесь мы используем селекторы потомков, чтобы выбрать только те элементы ‹h2› и ‹h3›, которые находятся внутри элемента ‹div› с идентификатором elixirs. Мы определяем для ‹h2› черный цвет, а для ‹h3› — красный, используя шестнадцатеричный код.

Быстрый тест

Вперед! Добавьте эти новые свойства в самый конец файла `lounge.css`, сохраните его и обновите страницу `lounge.html`.

Мы получили черные и красные заголовки в разделе с напитками и никак не повлияли на аквамариновый цвет заголовков ‹h2›, используемый на главной странице.



Теперь нам осталось изменить межстрочный интервал.

далее ›

Решение проблемы с межстрочными интервалами



Вспомните, что в прошлой главе мы сделали межстрочный интервал для текста гостевой немного больше обычного. Это выглядит замечательно, но нужно, чтобы в разделе с напитками межстрочный интервал был таким же, как и в меню, которое подается клиентам. Кажется, это достаточно просто, не так ли? Просто определите свойство **line-height** для элемента ** **, и все будет замечательно, потому как это свойство наследуемое. Единственная проблема заключается в том, что заголовки тоже унаследуют свойство **line-height**, и все закончится примерно так:

```
#elixirs {
    line-height: 1em;
}
```

Если вы зададите свойство line-height для всего элемента , то оно будет унаследовано всеми элементами внутри , включая заголовки. Обратите внимание, что межстрочный интервал в заголовке слишком мал, строки почти сливаются.

Причина маленьких межстрочных интервалов в заголовках в том, что все элементы, вложенные в ** **, наследуют значение **1em** свойства **line-height**. Это значит, что интервал равен 1×размер шрифта в разделе с напитками. В данном случае шрифт имеет размер **small**, или около 12 пикселов (в зависимости от того, какой браузер вы используете). Помните, что элемент ** ** с идентификатором **elixirs** наследует свойство **font-size** от элемента **<body>**, который мы задали как **small**.

Вот размеры шрифтов для элементов. Для элемента body мы задали значение small, которое наследуется элементом с идентификатором elixirs.



Значение свойства line-height для <h2> задано равным размеру высоты шрифта элемента с идентификатором elixirs, то есть small, или около 12 пикселов.

Нужно, чтобы все элементы, вложенные в ** ** с идентификатором **elixirs**, имели межстрочный интервал, вычисляемый не относительно размера шрифта элемента ** **, а относительно размера шрифта вложенного элемента. Мы хотим, чтобы заголовок **<h2>** имел межстрочный интервал, равный 1×размер шрифта, который составляет 120% от **small**, а абзацы **<p>** — межстрочный интервал, равный 1×размер шрифта, который равен **small**. Как можно этого добиться? У свойства **line-height** есть небольшая особенность: вместо числа с указанием относительной единицы измерения, такой как em или %, вы можете задать *только число*. При этом вы говорите, что каждый элемент, вложенный в ** ** с идентификатором **elixirs**, должен иметь размер, равный высоте *его собственного* шрифта, а не шрифта элемента ** ** с идентификатором **elixirs**. Попробуйте сделать это: установите значение **1** свойства **line-height** элемента ** ** с идентификатором **elixirs**, и вы увидите, что проблема с заголовками будет решена.

Размер шрифта для элемента p равен small (p наследует свойство font-size от элемента с идентификатором elixirs), так что он будет иметь межстрочный интервал 12 пикселов. Это как раз то, что нам нужно.




```
#elixirs {
    line-height: 1;
}
```

Добавьте свойство line-height и присвойте ему значение 1 для элемента с идентификатором elixirs, чтобы изменить межстрочный интервал каждого элемента.

Мы хотим, чтобы межстрочный интервал для элемента <h2> равнялся высоте его собственного шрифта, что составляет 14 пикселов (120% от small).

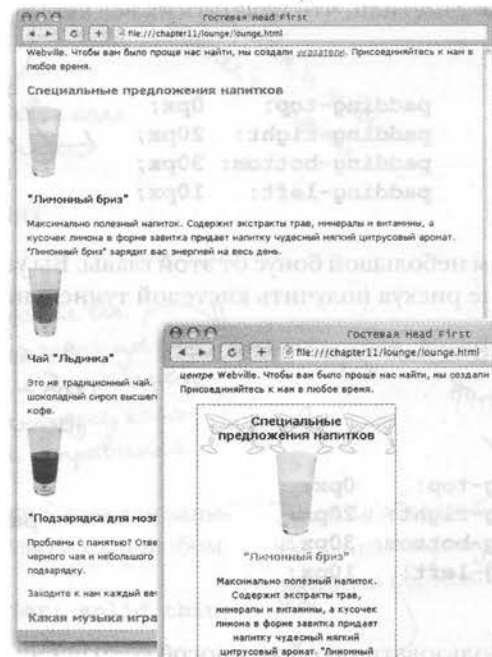
Посмотрите, что получилось

Взгляните на раздел с напитками. Вы его полностью изменили, и теперь он выглядит как меню, которое подают посетителям гостевой. Вы смогли сделать это, всего лишь добавив в свой XHTML-код пару CSS-правил и свойств, элемент ** ** и атрибут **id**.

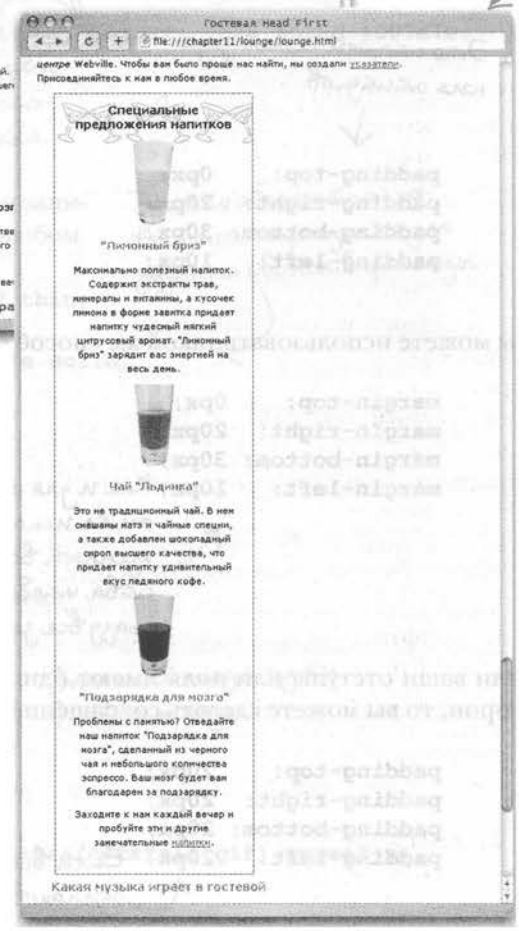
К настоящему времени вы уже, скорее всего, осознаете, насколько силен CSS и как легко можно менять веб-страницы, если их структура (XHTML) и стилизация (CSS) разделены. Вы можете придать XHTML-коду полностью новый вид, просто изменив CSS-код.



Вспомните, как выглядел раздел с напитками в самом начале...



...а вот как он выглядит сейчас.



Пришло время воспользоваться сокращениями



Вы, скорее всего, заметили, что есть немало CSS-свойств, которые часто применяются вместе, например **padding-left**, **padding-right**, **padding-bottom** и **padding-top**. То же самое можно сказать о свойствах для задания полей. А как насчет **background-image**, **background-color** и **background-repeat**? Все они определяют различные особенности фона элемента. Вы, наверное, также заметили, что печатать их все немного утомительно.

```
padding-top:    0px;
padding-right:  20px;
padding-bottom: 30px;
padding-left:   10px;
```

Приходится много печатать, чтобы задать четыре числа.

Хорошо, вот вам небольшой бонус от этой главы. Вы узнаете, как задать все эти значения, не рискуя получить кистевой туннельный синдром.

Это старый способ задания отступов.

```
padding-top:    0px;
padding-right:  20px;
padding-bottom: 30px;
padding-left:   10px;
```

А это новый и усовершенствованный способ — нужно написать сокращенные названия свойств.

```
padding: 0px 20px 30px 10px;
```

верхнее, правое, нижнее, левое

Вы можете использовать такой же способ сокращения для полей:

```
margin-top:    0px;
margin-right:  20px;
margin-bottom: 30px;
margin-left:   10px;
```

```
margin: 0px 20px 30px 10px;
```

верхнее, правое, нижнее, левое

Как и для отступов, вы можете использовать сокращенный вариант свойства, чтобы определить сразу все значения полей.

Если ваши отступы или поля имеют одинаковые размеры со всех сторон, то вы можете сделать сокращение *на самом деле* коротким:

```
padding-top:    20px;
padding-right:  20px;
padding-bottom: 20px;
padding-left:   20px;
```

```
padding: 20px;
```

Это значит, что с каждой стороны блока отступы будут по 20 пикселов.

Если все отступы имеют одинаковый размер, это можно записать таким образом.

Но и это еще не все...



Рассмотрим еще один распространенный способ сокращать записи для задания полей (или отступов).

```
margin-top:    0px;
margin-right:  20px;
margin-bottom: 0px;
margin-left:   20px;
```

Верхнее и нижнее одинаковы

Правое и левое одинаковы

```
margin: 0px 20px;
```

верхнее и нижнее — правое и левое

Если верхнее и нижнее, а также правое и левое поля одинаковы, то можете использовать сокращение.

Как насчет свойств, описывающих границы? Для них вы также можете использовать сокращения.

```
border-width:  thin;
border-style:  solid;
border-color:  #007e7e;
```

Перепишите все свойства границы как одно. Они могут идти в любом порядке, который вам понравится.

```
border: thin solid #007e7e;
```

Сокращения для свойств границы еще более гибкие, чем для записи полей и отступов, потому что вы можете задавать их в любом порядке.

Все это абсолютно правильные сокращения для задания свойств границы

```
border: solid thin #007e7e;
border: #007e7e solid thin;
border: solid thin;
border: #007e7e solid;
border: solid;
```

...и не забудьте сокращения для задания фона элемента

Добавляя свойства для определения фона элемента, вы также можете использовать сокращения:

```
background-color:  white;
background-image:  url(images/cocktail.gif);
background-repeat: repeat-x;
```

Значения могут идти в любом порядке.

```
background: white url(images/cocktail.gif) repeat-x;
```

Еще больше сокращений



Описание сокращенных вариантов свойств будет неполным без упоминания сокращений для шрифтов. Рассмотрим свойства, которые можно задать для шрифтов: **font-family**, **font-style**, **font-weight**, **font-size**, **font-variant** и **line-height**. Итак, существует сокращение, которое охватывает все эти свойства:

Это те свойства, которые перечислены в сокращении. В отличие от предыдущих случаев, их порядок имеет значение...

И наконец, вам нужно указать семейство шрифтов. Необходимо определить по крайней мере один шрифт, хотя дополнительные также желательно указать.

Вы должны определить размер шрифта.

```
font: font-style font-variant font-weight font-size/line-height font-family
```

Все эти значения необязательны. Вы можете задавать любую их комбинацию, но все они должны стоять перед значением свойства font-size.

Свойство line-height тоже необязательное. Если вы хотите его задать, просто добавьте символ «/» сразу после значения свойства font-size и задайте межстрочный интервал.

Между названиями семейств шрифтов ставьте запятые.

Итак, попробуем применить это на практике. Вот свойства шрифта для элемента **body** гостевой:

```
font-size: small;
font-family: Verdana, Helvetica, Arial, sans-serif;
line-height: 1.6em;
```

Преобразуем их в сокращение:

Мы не используем эти свойства, но это не страшно, так как все они являются необязательными.

```
font: font-style font-variant font-weight font-size/line-height font-family
```

Напишем это сокращение:

```
font: small/1.6em Verdana, Helvetica, Arial, sans-serif;
```

Ого, достаточно кратко, да? Теперь вы сможете гораздо больше времени уделить своему хобби.

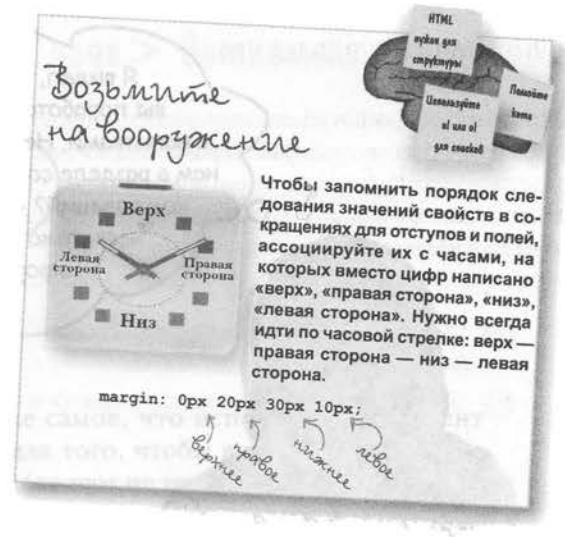
Часто Задаваемые Вопросы

В: Обязательно ли использовать сокращения?

О: Нет, не обязательно. Некоторые полагают, что длинная запись лучше читается. Преимущество сокращений состоит в том, что они делают CSS-файлы меньше по объему и, конечно же, быстрее вводятся с клавиатуры. Однако если возникают проблемы, то их немного сложнее устранить, если использованы некорректные значения или неверный порядок. Итак, применяйте любую форму записи, которая более удобна для вас, так как они обе абсолютно допустимы.

В: Сокращения достаточно сложны, потому что приходится запоминать порядок следования свойств и то, какие из них являются обязательными, а какие — нет. Как это все запомнить?

О: Вы сами не заметите, как быстро сумеете все запомнить. Однако надо сказать, что разработчики, использующие сокращения, предпочитают иметь под рукой справочное руководство. Просто купите одно из таких руководств и, если вам понадобится быстро найти название какого-нибудь свойства или его синтаксис, возьмите этот справочник и найдите то, что вам нужно. Лично мы предпочитаем книгу *CSS Pocket Reference*, написанную Эриком Мейером (Eric Meyer). Она совсем маленькая, но содержит огромное количество справочной информации.





Упражнение

Пришло время применить все новые знания на практике. Вы помните, что в нижней части гостевой страницы есть небольшой раздел с информацией об авторском праве, который является нижним колонтитулом страницы. Добавьте элемент ** **, чтобы создать для него собственный логический раздел. После этого стилизуйте этот раздел, используя следующие свойства:

```
font-size: 50%;
text-align: center;
line-height: normal;
margin-top: 30px;
```

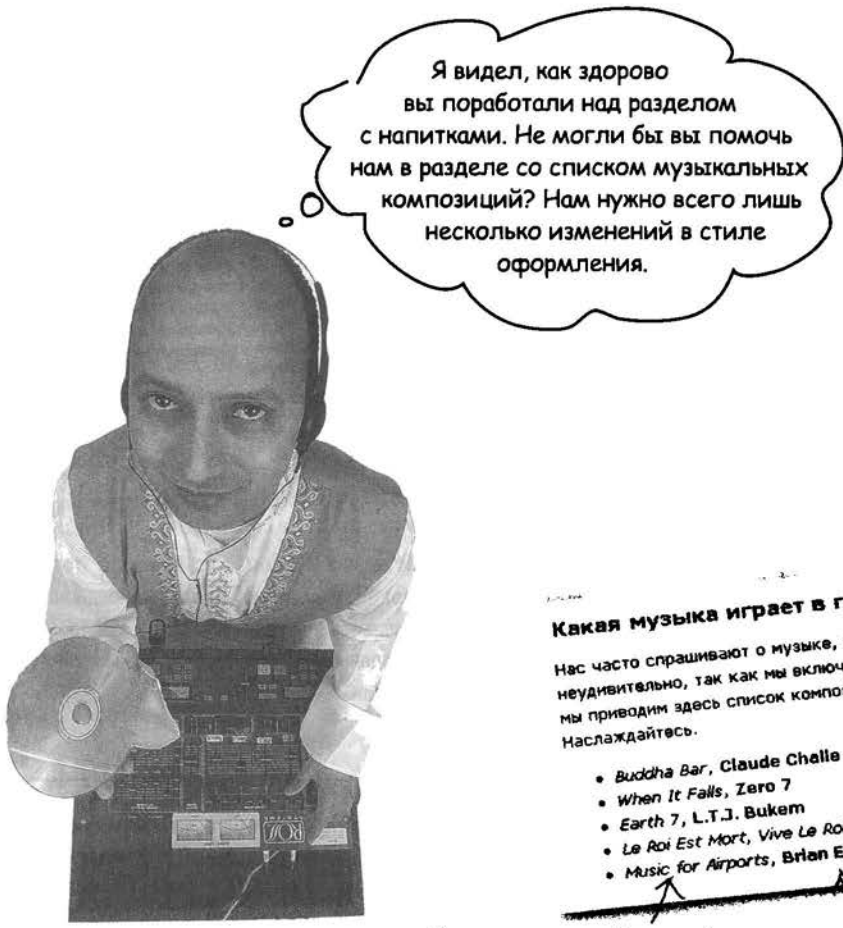
Сделаем текст действительно маленьким, ведь это ВАЖНАЯ ИНФОРМАЦИЯ, К КОТОРОЙ ОБЫЧНО СТАРАЮТСЯ НЕ ПРИВЛЕКАТЬ ВНИМАНИЕ.

Выровняем текст по центру.

Кроме того, установим для межстрочного интервала значение normal (ключевое слово, с которым вы раньше не встречались). Это значение позволяет браузеру выбрать подходящий размер для свойства line-height, которое обычно основывается на параметрах шрифта.

Давайте также зададим верхнее поле, чтобы отделить нижний колонтитул от остальной информации на странице.

Пока вы не перешли к чему-то другому, просмотрите файл `lounge.css`. Есть ли еще места, где лучше использовать сокращения? Если да, то внесите соответствующие изменения.



Постоянный диджей гостевой.

Какая музыка играет в гостевой

Нас часто спрашивают о музыке, которая играет в зале, и это неудивительно, так как мы включаем только самое лучшее. Только для вас мы приводим здесь список композиций, который еженедельно обновляется. Наслаждайтесь.

- *Buddha Bar*, **Claude Challe**
- *When It Falls*, **Zero 7**
- *Earth 7*, **L.T.J. Bukem**
- *Le Roi Est Mort, Vive Le Roi!*, **Enigma**
- *Music for Airports*, **Brian Eno**

Выделить названия всех компакт-дисков курсивным шрифтом.

Выделить имена всех исполнителей полужирным шрифтом.

МОЗГОВОЙ ШТУРМ

Как вы думаете, какой способ оформления компакт-дисков и имен исполнителей в разделе «Какая музыка играет в гостевой» лучший?



Фрэнк: Да, но это то же самое, что использовать элемент **<blockquote>** только для того, чтобы выделить текст. Мне кажется, что на самом деле нам не нужно придавать особый смысл названиям компакт-дисков и именам исполнителей. Нам просто нужно выделить их курсивным и полужирным шрифтом соответственно. Кроме того, а если кто-то переопределит стиль элементов **** и ****? Это также повлияет на наши названия.

Джим: Хорошо, я действительно думал об этом. Я понимаю, что перед нами всего лишь текст, представляющий собой пункт списка, и его лучше выделить иным способом. Мне кажется, я знаю такой способ.

Фрэнк: Что ты решил сделать?

Джим: Мы можем стилизовать элементы, выбрав только часть текста, например Music for Airports, Brian Eno. Нам придется указать элементы вокруг каждой части текста, чтобы по-особому стилизовать ее.

Фрэнк: О, и правда. Я понимаю, что ты задумал.

Джим: Я предполагаю, мы должны написать что-то вроде этого:

```
<div class="cd">Music for Airports</div>
<div class="artist">Brian Eno</div>.
```

Но это блочный элемент, поэтому нужно добавлять разрывы строки.

Фрэнк: Ах, я думаю, это прекрасно, Джим. Есть такой элемент, как ** **, только предназначенный для строчных элементов. Он называется ****. Если его добавить, твоя идея может сработать.

Джим: Я готов. Как он работает?

Фрэнк: Элемент **** позволяет группировать строчные символы и элементы. Давай попробуем добавить его...

Добавление элементов за три простых шага

Элементы **** дают возможность логически выделить строчное содержимое аналогично тому, как элементы ** ** позволяют создать логическое выделение содержимого блочного уровня. Чтобы увидеть, как это работает, стилизуем раздел со списком музыкальных композиций, заключив в теги **** названия компакт-дисков и имена исполнителей, а затем написав два CSS-правила для стилизации этих элементов. Рассмотрим, что нужно сделать.

1. Вложите каждое название компакт-диска и имя исполнителя в отдельный элемент ****.
2. Добавьте одни элементы **** в класс **cd**, а другие — в класс **artist**.
3. Создайте правило для стилизации класса **cd** курсивным шрифтом, а класса **artist** — полужирным.

Шаг первый и второй: добавление элементов

Откройте файл `lounge.html` и найдите заголовок «Какая музыка играет в гостевой». Ниже приведен маркированный список композиций.

Каждый пункт списка состоит из названия компакт-диска, затем и имени исполнителя.

```
<ul>
<li>Buddha Bar, Claude Challe</li>
<li>When It Falls, Zero 7</li>
<li>Earth 7, L.T.J. Bukem</li>
<li>Le Roi Est Mort, Vive Le Roi!, Enigma</li>
<li>Music for Airports, Brian Eno</li>
</ul>
```

Добавим элементы **** первой паре «диск — исполнитель»:

Просто добавьте открывающий тег с атрибутом class и его значением cd.

Затем добавьте закрывающий тег после названия компакт-диска.

Сделайте то же самое для имени исполнителя. Вложите его в элемент , только на этот раз поместите в класс artist.

```
<ul>
<li><span class="cd">Buddha Bar</span>, <span class="artist">Claude Challe</span></li>
<li>When It Falls, Zero 7</li>
<li>Earth 7, L.T.J. Bukem</li>
<li>Le Roi Est Mort, Vive Le Roi!, Enigma</li>
<li>Music for Airports, Brian Eno</li>
</ul>
```

Шаг третий: стилизация элемента

Перед тем как пойти дальше, сохраните файл и обновите страницу в браузере. Как и элемент ** **, по умолчанию **** не имеет никакого стиля, так что вы не увидите изменений.

Теперь привнесем немного стиля. Добавьте два следующих правила в самый конец файла `lounge.css`:

Мы добавим по одному классу для каждого нового правила: cd и artist.

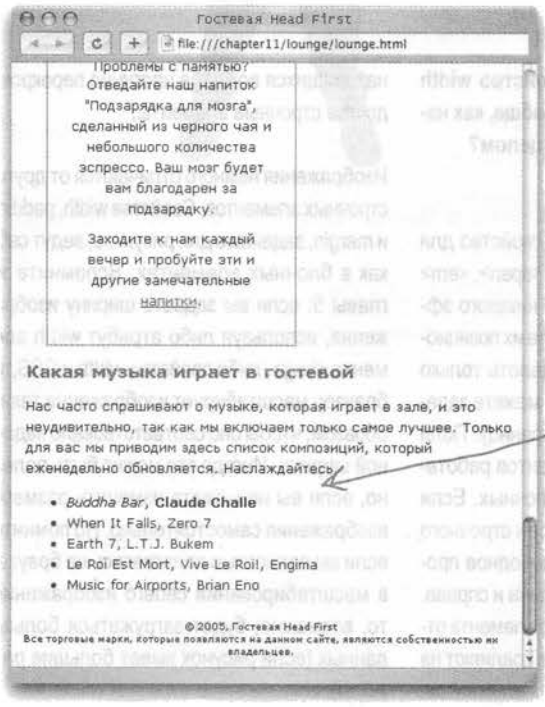
Для компакт-дисков зададим курсивный стиль шрифта.

```
.cd {
        font-style: italic;
}

.artist {
        font-weight: bold;
}
```

Установим для свойства font-weight значение bold, чтобы выделить имена исполнителей.

Тестирование элементов

Итак, сохраните изменения и обновите страницу. Вы увидите следующее:



Теперь первая музыкальная композиция оформлена правильно.



Возьми в руку карандаш



Вам необходимо закончить работу. Добавьте элементы **** для оставшихся музыкальных композиций и протестируйте страницу. Решение можно посмотреть в конце главы.

```
<ul>
<li><span class="cd">Buddha Bar</span>, <span class="artist">Claude Challe</span></li>
<li>When It Falls, Zero 7</li>
<li>Earth 7, L.T.J. Bukem</li>
<li>Le Roi Est Mort, Vive Le Roi!, Enigma</li>
<li>Music for Airports, Brian Eno</li>
</ul>
```

Часто Задаваемые Вопросы

В: В каких случаях лучше использовать элемент вместо таких строчных элементов, как или ?

О: Как обычно, размечать содержимое лучше с помощью тех элементов, которые лучше всего соответствуют его значению. Итак, если вы хотите выделить слова логически, то используйте элемент ; если вы пытаетесь подчеркнуть важность информации, используйте . Но если вы просто хотите изменить стиль оформления отдельных слов, например названий музыкальных альбомов или исполнителей, то лучше использовать элементы и распределить их по определенным классам, чтобы сгруппировать и затем иметь возможность стилизовать все вместе.

В: Могу ли я задавать свойство width для элементов ? А вообще, как насчет строчных элементов в целом?

О: Вы можете задавать это свойство для таких строчных элементов, как , и , но вы не заметите никакого эффекта до тех пор, пока не станете их позиционировать (это вы научитесь делать только в следующей главе). Вы также можете задавать для них отступы, поля и границу. Поля и отступы для строчных элементов работают немного не так, как для блочных. Если вы добавите поля со всех сторон строчного элемента, то заметите, что свободное пространство появилось только слева и справа. Вы можете добавить для такого элемента отступы сверху и снизу, но они не повлияют на расстояние до других строчных элементов, находящихся вокруг, а частично перекроют другие строчные элементы.

Изображения немного отличаются от других строчных элементов. Свойства width, padding и margin, заданные для рисунков, ведут себя как в блочных элементах. Вспомните из главы 5: если вы задаете ширину изображения, используя либо атрибут width элемента <img>, либо свойство width в CSS, то браузер масштабирует изображение таким образом, чтобы оно соответствовало заданной ширине. Иногда это может быть полезно, если вы не можете изменить размеры изображения самостоятельно. Но помните: если вы полностью полагаетесь на браузер в масштабировании своего изображения, то, возможно, будет загружаться больше данных (если рисунок имеет большие размеры).

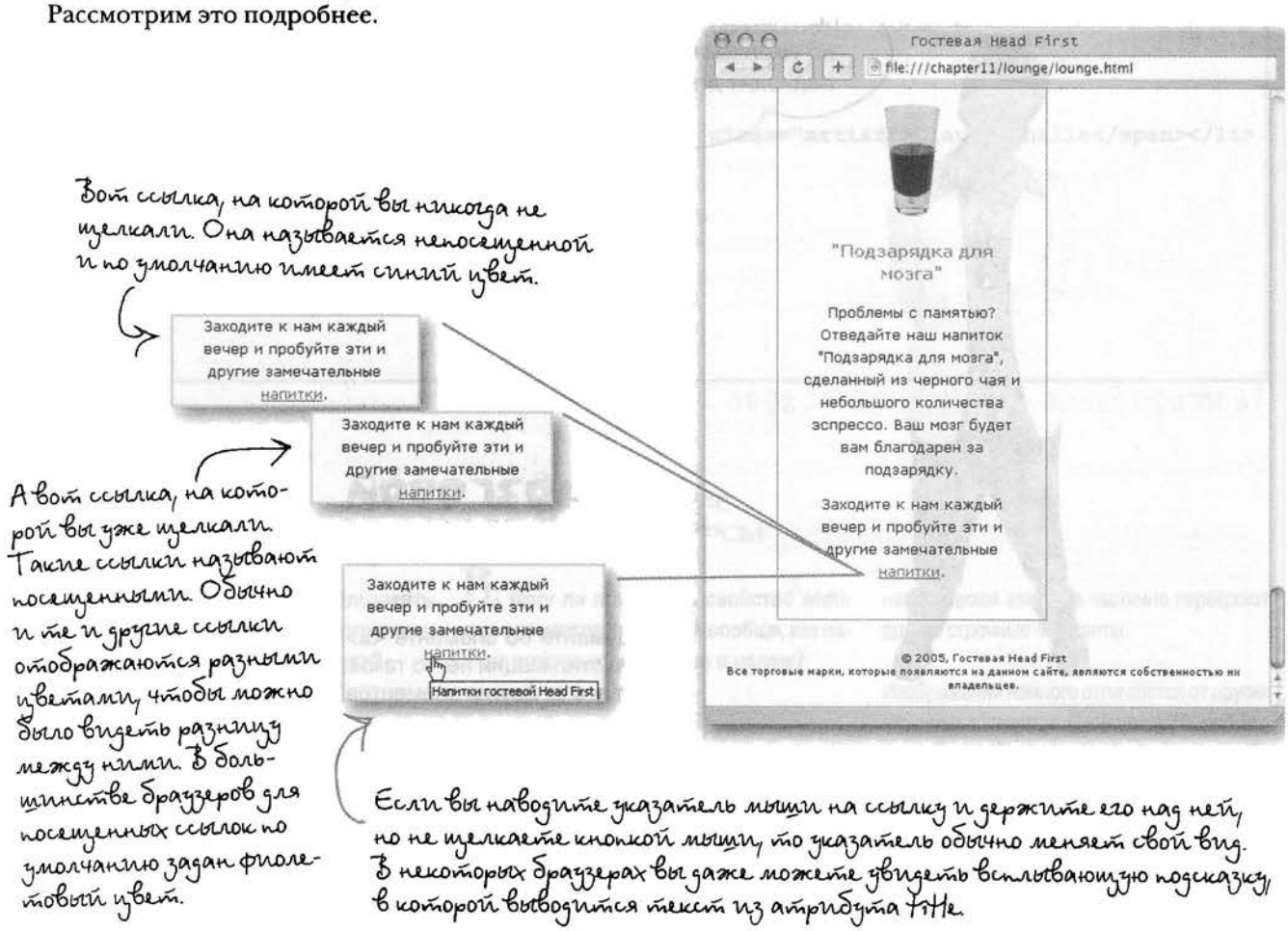


МОЗГОВОЙ ШТУРМ

Подумайте об элементе **<a>**. Есть ли в его стилизации нечто такое, что отличает его от других элементов?

Элемент <a> и его разносторонняя личность

Заметили ли вы, что когда дело доходит до стилизации, ссылки немного отличаются от остальных элементов? Ссылки – это хамелеоны мира элементов. В зависимости от обстоятельств они могут мгновенно менять свой стиль. Рассмотрим это подробнее.



Стиль элемента **<a>** меняется в зависимости от его *состояния*. Если на ссылке еще ни разу не щелкали, то она будет иметь один стиль, а если щелкали – другой. Если же вы просто навели указатель мыши на ссылку, то она может иметь третий стиль. Возможно, у элемента **<a>** даже больше стилей оформления, чем можно заметить.

Как можно по-разному стилизовать элементы с учетом их состояния?

Ссылка может находиться в нескольких состояниях: она может быть непосещенной, посещенной или ненажатой (над которой находится указатель мыши). Кроме того, есть еще несколько других состояний. Итак, как же можно извлечь выгоду из этих состояний? Например, было бы хорошо задать цвета для посещенных и непосещенных ссылок. Или, может быть, выделить ссылку подсветкой, когда пользователь наводит на нее указатель мыши. Если только существует способ это сделать...

Конечно же, существует, но если мы скажем вам, что для этого нужно использовать *псевдоклассы*, вы, скорее всего, просто решите, что и так достаточно узнали сегодня, и закроете книгу. Верно? Но подождите минутку! Сделаем вид, что мы никогда не произносили слово *«псевдоклассы»*, и просто посмотрим, как можно стилизовать ссылки.

Обратите внимание, что вслед за символом «a» идет символ «:», а затем задается состояние, которое мы хотим описать.

```
a:link {
    color: green;
}

a:visited {
    color: red;
}

a:hover {
    color: yellow;
}
```

Этот селектор применяется к непосещенным ссылкам.

Данный селектор будет применен для ссылок, которые уже были посещены.

А этот селектор применяется, когда пользователь наводит на ссылку указатель мыши.



Упражнение

Добавьте эти правила в самый конец файла `lounge.css`, сохраните его и обновите страницу `lounge.html`. Поэкспериментируйте со ссылками, чтобы увидеть все их состояния. Обратите внимание на то, что вам придется почистить журнал в браузере, чтобы увидеть цвет непосещенных ссылок (зеленый). Проделав все это, убедитесь, что удалили новые правила из файла `lounge.css`, а затем можете продолжить работу.

Часто Задаваемые Вопросы

В: Что будет, если я стилизую элемент <a> как обычный элемент? Например:

```
a { color: red; }
```

О: Вы, конечно, можете так сделать, но тогда ваши ссылки будут выглядеть одинаково во всех состояниях, что сделает их менее удобными для использования, потому что не будет видно, какие из ссылок уже посещались, а какие — нет.

В: Какие еще есть состояния ссылок?

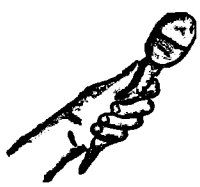
О: Есть еще два состояния: в фокусе и активное. Первое возникает, когда браузер фокусируется на вашей ссылке. Что это означает? Некоторые браузеры позволяют пользователю пройтись по всем ссылкам на странице, если он будет нажимать клавишу табуляции. При этом выбранная ссылка получает признак фокусировки. Активное состояние появляется, когда пользователь щелкает на ссылке впервые. Единственное, на что стоит обратить внимание, работая с этими состояниями, что они пока не очень хорошо поддерживаются всеми браузерами. Таким образом, если это все-таки для вас важно, тестируйте свои ссылки.

В: Могут ли мои ссылки находиться одновременно в нескольких состояниях? Например, ссылка может быть посещенной и при этом пользователь будет активно щелкать на ней?

О: Конечно, могут. Вы определяете, какой стиль будет применен, устанавливая порядок правил. Итак, правильный порядок обычно такой: link, visited, focus, hover и затем active. Если вы будете задавать состояния ссылок в таком порядке, то получите те результаты, которых ожидаете.

В: Хорошо, я понял. Что же такое псевдокласс?

О: Это всего лишь одно из тех слов в языке CSS, которое сбивает людей с толку. Но, как вы уже заметили, стилизация ссылок достаточно проста и понятна. Итак, поговорим о псевдоклассах...



УЯЗВИМОСТЬ ПСЕВДОКЛАССА

Интервью, взятое на этой неделе.
Знакомство с псевдоклассом.

Head First: Добро пожаловать, Псевдокласс. Очень приятно, что ты пришел к нам. Должен признаться, что когда меня попросили взять это интервью, я немного смутился. Что такое псевдокласс? Единственное, что приходило на ум, — песня Фила Коллинза 1980-х годов.

Псевдокласс: Ах, наверное, это песня *Sussudio*. Меня зовут *Псевдо*.

Head First: Ой. Это чистое недоразумение. Может быть, с этого и начнем? Не мог бы ты рассказать о том, откуда появилось твое имя?

Псевдокласс: Под «псевдо» обычно понимают что-то, кажущееся настоящим, но таковым не являющееся.

Head First: А фамилия? Класс?

Псевдокласс: Все вы знаете, что такое CSS-классы. Это группы для размещения в них элементов, позволяющие стилизовать всю группу вместе. Объедините «псевдо» и «класс», и получится «псевдокласс»: он работает как класс, но на самом деле им не является.

Head First: А что же в нем ненастоящего, если он работает так же, как настоящий класс?

Псевдокласс: Хорошо, откройте XHTML-файл и найдите в нем класс `:visited`, `:link` или `:hover`. Когда найдете, дайте мне знать.

Head First: Я не могу найти ни одного из них.

Псевдокласс: И тем не менее псевдоклассы `a:link`, `a:visited` и даже `a:hover` позволяют определить стиль так, будто являются классами. Однако это псевдоклассы. Иными словами, псевдоклассы можно стилизовать, но никто никогда не печатает их в своем XHTML-коде.

Head First: Хорошо, как же они работают?

Псевдокласс: Браузер внимательно исследует ваш код и распределяет элементы `<a>` по псевдоклассам. Если ссылка уже посещена, то — никаких проблем — она будет помещена в класс `visited`. Пользователь навел указатель мыши на ссылку и задумался? Не проблема, браузер добавит ее в класс `hover`. О, пользователь уже убрал указатель со ссылки? Браузер «достает» ссылку из класса `hover`.

Head First: Ого, никогда об этом не слышал. Значит, браузер сам добавляет элементы в эти классы и удаляет их оттуда?

Псевдокласс: Да, именно так, и об этом очень важно знать, иначе как вы сможете стилизовать свои ссылки, находящиеся в различных состояниях?

Head First: Итак, Псевдо, ты работаешь только со ссылками?

Псевдокласс: Нет, я работаю и с другими элементами. Некоторые браузеры уже поддерживают такие псевдоклассы, как `active` и `hover`, для разных элементов. Кроме того, существует еще несколько других псевдоклассов. Так, псевдокласс `:first-child` предназначен для определения дочерних элементов, например первого абзаца в элементе `<blockquote>`. Но будьте осторожны, используя псевдоклассы, потому что браузеры пока не полностью их поддерживают.

Head First: Отлично, мы, конечно же, вынесем из этого интервью кое-что полезное для себя. Кто знал, что песня на самом деле называется Sussudio?! Спасибо, что был с нами, Псевдокласс.

Применение псевдоклассов на практике

Итак, будем честны. Вы только что ознакомились с одной из самых важных тем в этой книге. Она важна не потому, что псевдоклассы позволяют стилизовать элементы, основываясь на различных классах, например **:link** или **:first-child**, к которым, как решает ваш браузер, они принадлежат. И не потому, что они дают вам отличный способ стилизовать элементы, основываясь на том, что происходит, пока пользователи работают с вашей страницей. Тема важна для вас потому, что, когда вы в следующий раз будете на собрании дизайнеров и начнете со знанием дела говорить о псевдоклассах, вы, скорее всего, станете *лидером команды*. Вы получите бонусы, продвижение по службе и как минимум благоговение и уважение ваших коллег.

Попробуем использовать эти псевдоклассы на практике. Вы уже добавили несколько новых правил в файл `lounge.css` и понаблюдали за их влиянием на внешний вид ссылок, однако эти правила не совсем подходят для гостевой. Давайте немного изменим их стиль.

Итак, большие перемены. Используем селектор потомков в сочетании с псевдоклассами. Первый селектор говорит выбрать любой непосещенный элемент <a>, который вложен в элемент с идентификатором elixirs. Зададим стиль ТОЛЬКО для ссылок внутри раздела с напитками.

```
#elixirs a:link {
    color: #007e7e;
}

#elixirs a:visited {
    color: #333333;
}

#elixirs a:hover {
    background: #f88396;
    color: #0d5353;
}
```

В этих двух правилах мы определяем цвет. Для непосещенных ссылок это аквамариновый...

...а для посещенных — темно-серый.

Теперь рассмотрим действительно интересное правило. Когда пользователь просто наводит указатель мыши на ссылку, цвет фона меняется на красный. Благодаря этому кажется, что ссылка подсвечивается красным цветом, когда вы наводите на нее указатель мыши. Попробуйте это сами!

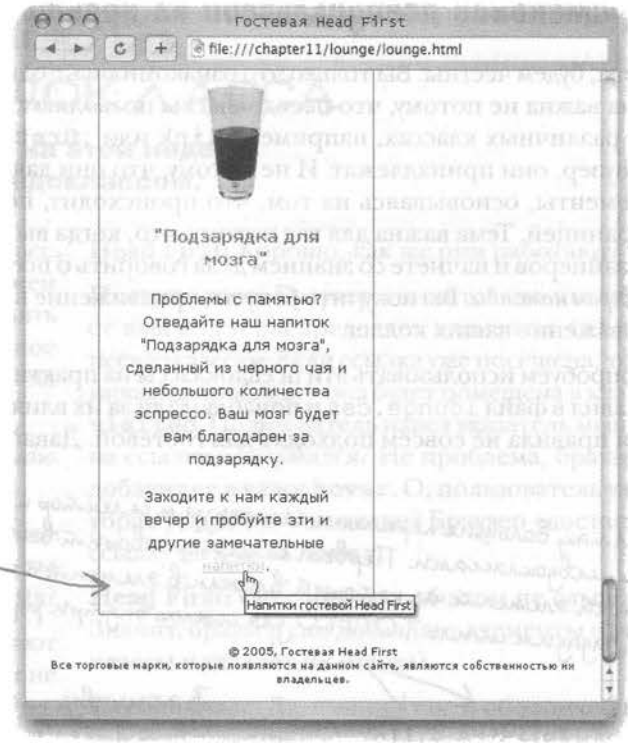


Упражнение

Откройте файл `lounge.css` и измените правила **a:link**, **a:visited** и **a:hover** так, чтобы в них использовались селекторы потомков и новые определения стиля. Сохраните изменения и переворачивайте страницу.

Тест для ссылок

Обновив страницу, вы увидите, что в разделе с напитками немного изменился стиль оформления ссылок. Помните: чтобы увидеть непосещенные ссылки, вам придется почистить журнал, иначе браузер будет знать, что вы уже посещали эти ссылки.



Теперь непосещенные ссылки отображаются аквамариновым цветом, а посещенные — серым. Если же вы наводите указатель мыши на ссылку, то появляется красная подсветка.

Возьми в руку карандаш

Ваша задача — определить для ссылки указатели в гостевой такой же стиль, как и для ссылок в разделе с напитками. Иными словами, все непосещенные ссылки должны быть аквамаринового цвета, а посещенные — серого. Однако нам не нужно, чтобы для остальных ссылок гостевой при наведении на них указателя мыши применялся какой-либо стиль. Это особенность ссылок в разделе с напитками. Как же это сделать? Заполните пропуски для стилей ссылки указатели и другой ссылки, которую вы, возможно, еще добавите в гостевую. Сверьте ваш ответ с приведенным в конце главы. Затем можете внести соответствующие изменения в файл `lounge.css`.

```
_____ { _____: #007e7e; }
_____ { _____: #333333; }
```

Не пора ли поговорить о каскадности?

Мы уже достаточно далеко зашли (если быть точными, мы на 485-й странице), а до сих пор ничего не говорили вам о том, что это за каскады в *каскадных* таблицах стилей. На самом деле, чтобы полностью понимать каскадность, нужно очень многое знать о CSS. Но нам кажется, что вы уже достаточно знаете.

Здесь мы приведем последнюю информацию, которая понадобится вам для понимания каскадности. Вы уже знаете, как работать с несколькими таблицами стилей, которые используются, чтобы лучше организовать стили или чтобы ваши страницы хорошо отображались на различных типах устройств. Существует еще один тип таблиц стилей, которые появляются, когда пользователи посещают вашу страницу. Давайте посмотрим.



Автор (это вы!)

Во-первых, есть все те таблицы стилей, которые вы сами написали для страницы.

Обратите внимание, что существует способ, позволяющий пользователю переопределить все ваши стили. Для этого в конце описания свойства задается выражение !important.

Некоторые браузеры позволяют пользователям создавать собственные стили для XHTML-элементов. Если в вашей таблице не задан какой-то стиль, то применяется таблица стилей пользователя.



Читатели (ваши пользователи)

И наконец, вы уже знаете, что браузер сам содержит набор стилей, который используется по умолчанию, если вы не определили какой-то стиль для элемента. Кроме того, стили по умолчанию используются, если нет ни авторских, ни пользовательских таблиц стилей.



Браузер

Когда браузеру необходимо определить, какой стиль применить к элементу, он использует все эти таблицы стилей. Приоритет имеют авторские (то есть ваши) стили, затем идут пользовательские и, наконец, те, что заданы в браузере по умолчанию.



На самом деле в этом и заключается смысл каскадности. Это способ, с помощью которого браузер выбирает стиль элемента, при условии, что задан целый набор стилей из нескольких таблиц. Чтобы ответить на этот вопрос, нужно свести вместе все имеющиеся в наличии таблицы стилей, правила и объявления отдельных свойств в этих правилах.

На следующих двух страницах мы подробно разберем, как все это работает. В первую очередь вы научитесь определять, какие правила элемента являются наиболее приоритетными. Как результат, вы сможете разбираться в том, как применяются стили в различных, даже самых запутанных, ситуациях, и даже будете понимать принципы каскадности лучше, чем 99 % веб-разработчиков (и это не шутка).

Каскадность



Для выполнения этого упражнения вам придется поработать браузером. Допустим, у вас на странице есть элемент **<h1>**, и вы хотите узнать для него значение свойства **font-size**. Вот как это сделать.

Шаг первый

Соберите вместе все таблицы стилей.

На этом шаге вам понадобятся ВСЕ таблицы стилей: таблицы, которые написал автор веб-страницы, таблицы, созданные пользователем, и стандартные стили браузера. Помните, сейчас вы играете роль браузера, так что имеете доступ ко всем этим таблицам!

Шаг второй

Найдите все соответствующие объявления свойства.

Нужно именно свойство **font-size**, так что найдите все его обьявления с таким селектором, который, вероятно, может выбирать элемент **<h1>**. Просмотрите все таблицы стилей и выберите из них все правила, соответствующие элементу **<h1>** и имеющие свойство **font-size**.

Мы уже говорили, что пользователи могут помечать свои CSS-правила выражением !important. В таком случае эти правила станут первыми в нашей сортировке.

Шаг третий

Возьмите все, что нашли, и отсортируйте.

Теперь, когда вы собрали все подходящие правила вместе, отсортируйте их в таком порядке: авторские, пользовательские, правила браузера. Иными словами, правила, написанные автором страницы, важнее правил, написанных пользователем, а пользовательские стили важнее стандартных стилей браузера.

Шаг четвертый

Теперь отсортируйте все обьявления по приоритетам.

Вспомните, мы уже немного говорили об этом в главе 8. Вы можете интуитивно предположить, что правило имеет больший приоритет, если оно более точно выбирает элемент. Например, селектор потомка **blockquote h1** имеет больший приоритет, чем селектор **h1**, потому как выбирает только те элементы **<h1>**, которые находятся внутри элементов **<blockquote>**. Но существует специальный способ, с помощью которого можно вычислить точный приоритет элемента, и мы рассмотрим его на следующей странице.

Шаг пятый

Наконец, отсортируйте все конфликтные правила в том порядке, в котором они появляются в отдельных таблицах стилей.

Теперь осталось взять полученный список и отсортировать конфликтующие правила в порядке их появления в соответствующей таблице стилей (от нижнего к верхнему). Таким образом, если вы помещаете новое правило в таблицу стилей, оно может переопределить все правила, расположенные выше.

Вот и все! Первое правило в новом списке станет главным, и будет применено именно его свойство **font-size**. Теперь посмотрим, как можно точно определить приоритет элемента.

Поиграем в игру «Каков мой приоритет?»

Давным-давно использовались четыре числа, но это было до появления XHTML... Разве вы не рады, что сейчас все изменилось?

Чтобы посчитать приоритет, нужно рассмотреть набор из трех цифр, например, так:

0 0 0

Затем следует просто подсчитать различные элементы в селекторе, например, так:

Имеются ли в селекторе названия каких-нибудь элементов? По одному баллу за каждое.

Имеются ли в селекторе какие-либо классы или псевдоклассы? По одному баллу за каждый.

Имеются ли в селекторе какие-нибудь идентификаторы? По одному баллу за каждый.

0 0 0

Например, в селекторе **h1** есть один элемент, так что получается:

Селекторы h1 и h1. b/ne содержат по одному элементу, так что для них обоих число в правом столбце будет равняться 1.

Читайте это как число один

0 0 1

И другой пример: в селекторе **h1.blue** есть один элемент и один класс, так что получается:

В селекторе h1.b/ne также содержится один класс, так что в среднем столбце для него будет стоять число 1

Читайте это как число одиннадцать.

0 1 1

Идентификаторов нет ни в одном селекторе, так что для обоих в левом столбце будет стоять число 0

После того как вы подсчитали количество всех идентификаторов, элементов и классов, можете сделать следующий вывод: чем больше число, тем больший приоритет имеет правило. Итак, поскольку **h1.blue** имеет число приоритета 11, то оно более приоритетно, чем **h1**, которое имеет число приоритета 1.

Возьми в руку карандаш

Проверьте свои силы в вычислении приоритета для правил, селекторы которых приведены ниже:

h1.greentea	_____	**ol li p**	_____	**em**	_____
p img	_____	**.green**	_____	**span.cd**	_____
a:link	_____	**#elixirs h1**	_____	**#sidebar**	_____

Часто Задаваемые Вопросы

В: Что делает одно число приоритета больше другого?

О: Просто читайте их как обыкновенные числа: 100 (сто) больше чем 010 (десять), что, в свою очередь, больше, чем 001 (один), и т. д.

В: Как насчет правила h1, h2? Каков его приоритет?

О: Рассматривайте это правило как два отдельных: правило h1, имеющее число приоритета 001, и правило h2 с числом приоритета 001.

В: Можете подробнее рассказать про выражение !important?

О: Пользователь может переопределить стиль, указав !important в самом конце объявления свойства, например, так:

```
h1 {
  font-size: 200%
  !important;
}
```

Это переопределит все авторские правила.

В: Я не могу получить пользовательские таблицы стилей. Как же можно выяснить, каким образом сработает каскадность?

О: Никак. Если пользователь переопределяет ваши стили, то вы не можете это контролировать. Поэтому просто делайте так, чтобы на ваших страницах применялись те стили, которые вам нравятся. Если же пользователь захочет их переопределить, то он получит то, что желает (возможно, это будет выглядеть лучше, а возможно, хуже).

Соберем Все это Вместе

Ура! Настало время для примера. Допустим, вам нужно узнать значение свойства **color** для элемента **<h1>**:

```
<h1 class="blueberry">Чудо-напиток из голубики</h1>
```

Давайте заставим пройти его через весь каскад.

Помните, вы — браузер, потому что пытаетесь выяснить, как отображать элемент <h1>.

Шаг первый

Соберите все таблицы стилей вместе.

```
h1 {
   color: black;
}
```



Браузер

Это вы (на данный момент).

```
h1 {
   color: #efefef;
}

h1.blueberry {
   color: blue;
}
```

Обычно в роли автора (человека, который пишет CSS) выступаете вы. Но сейчас вы играете роль браузера.



Автор

```
body h1 {
   color: #cccccc;
}
```



Пользователь — человек, использующий браузер.

дальше »

Шаг второй

Найдите все подходящие обьявления свойства.

Вот все правила, которые возможно, могут соответствовать элементу <h1> и содержать свойство color.

Пользователь
```
body h1 {
   color: #cccccc;
}
```

Браузер
```
h1 {
   color: black;
}
```

Автор
```
h1 {
   color: #efefef;
}

h1.blueberry {
   color: blue;
}
```

Шаг третий

Теперь возьмите все найденные правила и отсортируйте их в следующем порядке: авторские, пользовательские, правила браузера.

Автор
```
h1 {
   color: #efefef;
}

h1.blueberry {
   color: blue;
}
```

Пользователь
```
body h1 {
   color: #cccccc;
}
```

Браузер
```
h1 {
   color: black;
}
```

Здесь мы просто изменили очередность правил и записали их в следующем порядке: авторские, пользовательские, правила браузера.

Шаг четвертый

Отсортируйте правила по приоритетам. Для этого нужно сначала вычислить число приоритета для каждого правила, а затем изменить порядок следования этих правил.

```
h1 {
   color: #efefef;
}                          0 0 1

h1.blueberry {
   color: blue;
}                          0 1 1

body h1 {
   color: #cccccc;
}                          0 0 2

h1 {
   color: black;
}                          0 0 1
```

Правило с классом blueberry перемещается в самый верх, так как имеет максимальное число приоритета.

```
h1.blueberry {
   color: blue;
}                          0 1 1

h1 {
   color: #efefef;
}                          0 0 1

body h1 {
   color: #cccccc;
}                          0 0 2

h1 {
   color: black;
}                          0 0 1
```

Обратите внимание, что мы сортируем правила только внутри категорий правил: авторские, пользовательские, правила браузера. Мы не сортируем весь список целиком, потому что иначе правило body h1 находилось бы выше правила h1, которое задано автором.

Шаг пятый

Наконец, отсортируйте все конфликтующие правила в порядке их появления в отдельных таблицах стиля.

На данном этапе нет проблем, так как у нас вообще не оказалось конфликтующих правил. Правило с классом blueberry, имеющее число приоритета 111 — явный лидер. Если бы у нас оказалось сразу два правила с числом приоритета 011, то выиграло бы то правило, которое располагалось бы ниже в таблице стилей.



```
h1.blueberry {
   color: blue;
}

h1 {
   color: #efefef;
}
```

Автор

```
body h1 {
   color: #cccccc;
}
```

Пользователь

```
h1 {
   color: black;
}
```

Браузер

У нас есть победитель...

Пройдя первичный отбор элементов, первую и вторую сортировку, а затем выиграв в соревновании по приоритетам, правило h1.blueberry поднялось на самую вершину. Итак, для элемента <h1> будет применено значение blue свойства color.

Часто Задаваемые Вопросы

В: Правило, расположенное в CSS-файле ниже, имеет приоритет над тем, что расположено выше, но что будет, если в XHTML есть ссылки сразу на несколько таблиц стилей?

О: Всегда идите по порядку: сверху вниз, независимо от того, один у вас CSS-файл или несколько. Просто представьте, что вы разместили все CSS-таблицы в том порядке, в котором расположены ссылки на них в вашем XHTML-коде. В результате станет ясно, как расположены правила одно относительно другого.

В: Получается, что, когда происходит сортировка по приоритету, порядок меняется не для всего списка целиком?

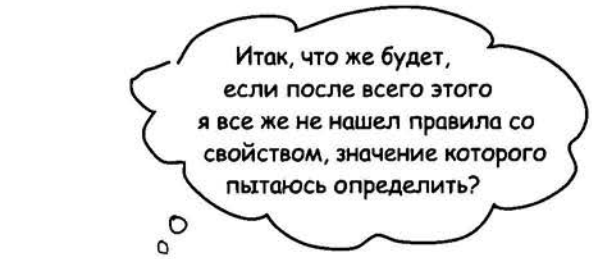
О: Каждая последующая сортировка происходит с учетом предыдущей. Итак, сначала правила сортируются по таким категориям: авторские, пользовательские, правила браузера. Затем внутри каждой категории вы сортируете их по приоритету. И наконец, вы сортируете только те правила, числа приоритета у которых оказались равными, учитывая их порядок следования в таблицах стилей.

В: Неужели пользователи на самом деле создают собственные таблицы стилей?

О: Вообще-то, нет. Но все же такие случаи бывают. Например, люди с ослабленным зрением могут создавать свои таблицы стилей и, конечно же, всегда найдется любопытный человек, который захочет поэкспериментировать с оформлением страницы. Но поскольку каждый пользователь может управлять лишь своим вариантом страницы, это никак не повлияет на ваш дизайн.

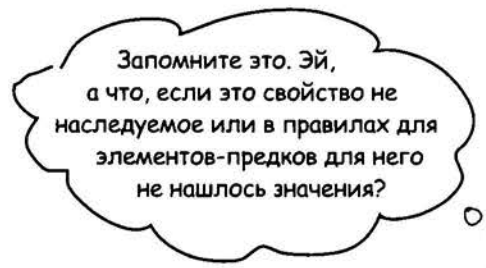
В: Что из этого всего действительно стоит запоминать?

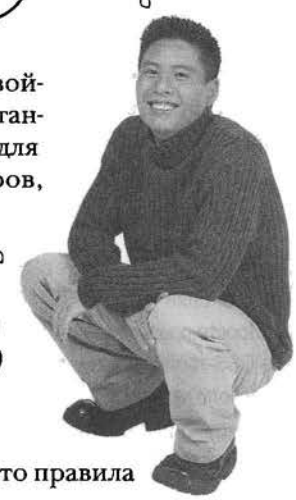
О: Вы будете работать с таблицами стилей на интуитивном уровне, и изо дня в день интуиция будет подводить вас все реже и реже. Время от времени вы будете видеть, как на ваших страницах появляются стили, которые вы совсем не ожидали увидеть, и именно в эти моменты вы будете возвращаться к изучению теории. Но если вы будете понимать, как работает каскадность, то всегда будете точно знать, что может произойти на ваших страницах.



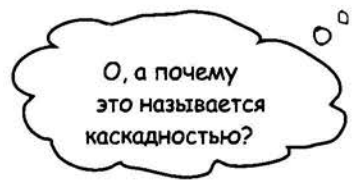


О, хороший вопрос. Мы уже немного говорили об этом в главе 8. Если вы не нашли соответствующего свойства во всем каскаде, то попытайтесь использовать наследование. Помните, что не все свойства являются наследуемыми: например, свойство **`border`** не наследуется. Но что касается *наследуемых* свойств (**`color`**, **`font-family`**, **`line-height`** и т. д.), браузер ищет элемент-предка, начиная с родительского элемента, и пытается найти значение нужного свойства. В случае успеха вы получите искомое значение.



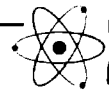


Тогда остается перейти к значению свойства, которое браузер берет из своих стандартных таблиц стилей. Такие таблицы для каждого элемента есть у всех браузеров, и они используются по умолчанию.



Такое название было выбрано потому, что правила из всех таблиц стилей как бы «льются каскадом» на страницу, при этом применяется самое приоритетное правило стиля для каждого элемента. Если вам все еще непонятно, почему это назвали каскадностью, не расстраивайтесь. Просто продолжайте двигаться вперед.

СТОП! Сделайте это упражнение, прежде чем перейти к следующей главе!



МОЗГОВОЙ ШТУРМ

ПОВЫШЕННАЯ СЛОЖНОСТЬ

Это особенное упражнение. Оно настолько необычное, что мы дадим вам время подумать над ним, прежде чем вы перейдете к следующей главе. Рассмотрим, что нужно сделать.

1. Откройте файл `lounge.html` и найдите элемент **`<div>`** с идентификатором **`elixirs`**.
2. Переместите весь раздел **`<div>`** с индентификатором **`elixirs`** в самый верх файла, чтобы над ним был только абзац с логотипом гостевой.
3. Сохраните файл и обновите страницу. Что изменилось?
4. Откройте файл `lounge.css`.
5. Найдите правило **`#elixirs`**.
6. Добавьте объявление этого свойства в самый конец правила:

```
float: right;
```

7. Сохраните файл и обновите страницу в браузере.

Что изменилось? Как вы думаете, что делает это свойство?

ПОВТОРИМ выученное



- Элементы ** ** применяются, когда нужно объединить группу взаимосвязанных элементов в логический раздел.
- Создание логических разделов может помочь вам выделить основные области содержимого, верхний и нижний колонтитулы страницы.
- Вы можете применять элементы ** **, чтобы сгруппировать элементы, для которых нужно задать общий стиль.
- Используйте вложенные элементы ** **, чтобы еще более детально структурировать свои файлы. Но не добавляйте структуру, пока она на самом деле вам не понадобится.
- После того как вы создали логический раздел, сгруппировав различные элементы с помощью элемента ** **, можете стилизовать этот ** ** точно так же, как стилизовали бы любой другой блочный элемент. Например, можно добавить границу вокруг группы элементов, используя свойство **border** для элемента ** **, в который они вложены.
- Свойство **width** определяет ширину области содержимого элемента.
- Общая ширина элемента состоит из ширины области содержимого, ширины отступов, самой границы и полей, которые вы добавили.
- Как только вы зададите элементу ширину, он больше не будет растягиваться, чтобы соответствовать ширине окна браузера.
- **Text-align** — свойство для блочных элементов, которое выравнивает по центру строчное содержимое блочного элемента. Оно наследуется всеми вложенными блочными элементами.
- Вы можете использовать селекторы потомков, чтобы выбрать элементы, вложенные в другие элементы. Например, селектор потомка

```
div h2 { ... }
```

выберет все элементы **<h2>**, вложенные в элементы ** ** (включая дочерние элементы, дочерние дочерних и т. д.).

- Для родственных свойств можно указывать сокращения. Например, **padding-top**, **padding-right**, **padding-bottom** и **padding-left** — все задают отступы и могут быть определены одним сокращенным правилом **padding**.
- В сокращенной форме могут быть определены свойства **padding**, **margin**, **border**, **background** и **font**.
- Строчный элемент **** похож на элемент ** **: он используется для группировки взаимосвязанных строчных элементов и текста.
- Вы можете добавлять элементы **** в классы (или давать элементам **** уникальные идентификаторы), чтобы стилизовать их.
- Элемент **<a>** может иметь несколько состояний. Основные состояния элемента **<a>** — это **unvisited**, **visited** и **hover**.
- Используя псевдоклассы, вы можете стилизовать элемент для каждого состояния отдельно. Псевдоклассы, которые чаще всего используются для элемента **<a>**, — это **:link** (для непосещенных ссылок), **:visited** (для посещенных ссылок) и **:hover** (для статуса «наведение указателя мыши»).
- Отдельные элементы предусматривают работу с псевдоклассом **:hover**, а некоторые браузеры также поддерживают псевдоклассы **:first-child**, **:active** и **:focus** для других элементов.

Возьми в руку карандаш

Решение

Это блок, в котором помечена ширина каждой его составляющей. Какова ширина всего блока? Вот решение.

$$30 + 2 + 5 + 200 + 10 + 2 + 20 = 269$$

Возьми в руку карандаш

Решение

Итак, теперь, когда вы умеете хорошо работать с шириной блока, вычислим общую ширину раздела с напитками. Мы знаем, что ширина области содержимого равняется 200 пикселам. Кроме того, заданы левый и правый отступы, а также граница, ширину которой мы определили как `thin`. Просто предположите, что толщина такой границы равна 1 пикселу (для большинства браузеров). Как насчет полей? Мы задали значение для ширины левого поля, но не задали для правого, то есть по умолчанию она равняется 0 пикселов.

Вот все свойства, имеющие отношение к ширине. Ваша задача — вычислить общую ширину элемента `<div>`.

$$20 + 20 + 200 + 1 + 1 + 0 + 20 = 262$$

Левый отступ + Правый отступ + Область содержимого + Левая граница + Правая граница + Правое поле + Левое поле

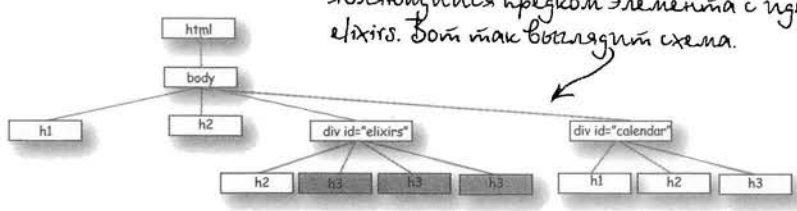
Возьми в руку карандаш

Решение

Ваша очередь. Напишите селектор, который будет выбирать только элементы `<h3>` внутри элемента `<div>`, относящегося к напиткам. В правиле задайте значение `#d12c47` свойства `color`. Кроме того, в приведенной ниже схеме расставьте метки над элементами, которые выбраны. Вот решение.

```
#elixirs h3 {
    color: #d12c47;
}
```

Это правило. Мы выбираем любой элемент <h3>, являющийся предком элемента с идентификатором elixirs. Вот так выглядит схема.





Решение упражнения

Пришло время применить все новые знания на практике. Вы помните, что в нижней части гостевой страницы есть небольшой раздел с информацией об авторском праве, который является нижним колонтитулом страницы. Добавьте элемент ** **, чтобы создать для него собственный логический раздел. После этого стилизуйте этот раздел, используя следующие свойства:

Сделаем текст действительно маленьким, ведь это ВАЖНАЯ ИНФОРМАЦИЯ, К КОТОРОЙ ОБЫЧНО СТАРАЮТСЯ НЕ ПРИВЛЕКАТЬ ВНИМАНИЕ.

```
font-size: 50%;
text-align: center;
line-height: normal;
margin-top: 30px;
```

Выровняем текст по центру.

Кроме того, установим для межстрочного интервала значение normal.

Давайте также зададим верхнее поле, чтобы отделить нижний колонтитул от остальной информации на странице.

Окружите раздел с информацией об авторском праве тегами .

Дайте ему идентификационное имя footer.

```
<div id="footer">
  <p>
    &copy; 2005, Гостевая Head First<br />
    Все торговые марки, которые появляются на данном сайте,
    являются собственностью их владельцев.
  </p>
</div>
```

Это CSS-код для нижнего колонтитула.

```
#footer {
   font-size: 50%;
   text-align: center;
   line-height: normal;
   margin-top: 30px;
}
```



Возьми в руку карандаш

Решение

Вам нужно было добавить элементы **** для оставшихся музыкальных композиций и протестировать страницу. Вот решение:

```
<ul>
<li><span class="cd">Buddha Bar</span>,
    <span class="artist">Claude Challe</span></li>
<li><span class="cd">When It Falls</span>,
    <span class="artist">Zero 7</span></li>
<li><span class="cd">Earth 7</span>,
    <span class="artist">L.T.J. Bukem</span></li>
<li><span class="cd">Le Roi Est Mort, Vive Le Roi!</span>,
    <span class="artist">Enigma</span></li>
<li><span class="cd">Music for Airports</span>,
    <span class="artist">Brian Eno</span></li>
</ul>
```

Какая музыка играет в гостевой

Нас часто спрашивают о музыке, которая играет в зале, и это неудивительно, так как мы включаем только самое лучшее. Только для вас мы приводим здесь список композиций, который еженедельно обновляется. Наслаждайтесь.

- *Buddha Bar*, **Claude Challe**
- *When It Falls*, **Zero 7**
- *Earth 7*, **L.T.J. Bukem**
- *Le Roi Est Mort, Vive Le Roi!*, **Enigma**
- *Music for Airports*, **Brian Eno**

Возьми в руку карандаш



Решение

Ваша задача — определить для ссылки указатели в гостевой такой же стиль, как и для ссылок в разделе с напитками. Иными словами, все непосещенные ссылки должны быть аквамаринового цвета, а посещенные — серого. Однако нам не нужно, чтобы для остальных ссылок гостевой при наведении на них указателя мыши применялся какой-либо стиль. Это особенность ссылок в разделе с напитками. Как же это сделать? Заполните пропуски для стилей ссылки указатели и другой ссылки, которую вы, возможно, еще добавите в гостевую. Вот решение упражнения:

```
a:link      { color   : #007e7e; }
a:visited   { color   : #333333; }
```

Возьми в руку карандаш



Решение

Проверьте свои силы в вычислении приоритета для правил, селекторы которых приведены ниже. Решение таково.

h1.greentea	011	**ol li p**	003	**em**	001
p img	002	**.green**	010	**span.cd**	011
a:link	011	**#elixirs h1**	101	**#sidebar**	100

Расставим элементы по местам



Пришло время обучить XHTML-элементы новым трюкам. Пора разбудить их и заставить помочь нам создавать страницы с реальными схемами размещения элементов. Каким образом? Ну, вы хорошо разобрались со структурными элементами и и теперь знаете, как работает блочная модель. Пришла пора применить эти знания и создать несколько дизайнов. Мы говорим не просто о цвете фона и шрифта, а о полностью профессиональном дизайне, в котором используется многоколоночная разметка.

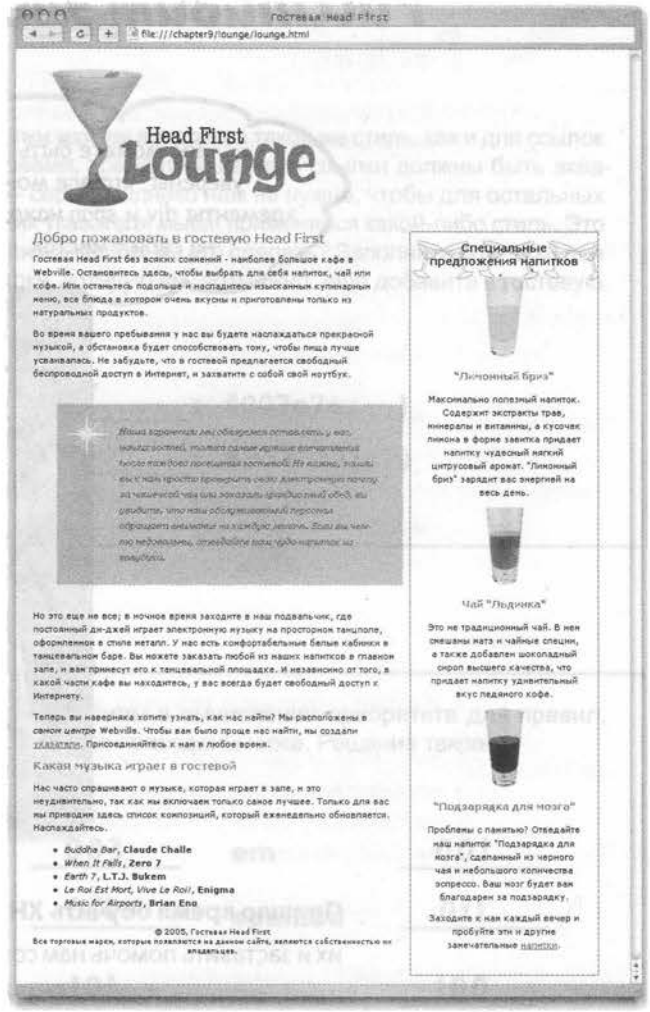
Вы сделали упражнение «Мозговой штурм» повышенной сложности?

Если вы не сделали упражнение «Мозговой штурм» повышенной сложности, приведенное в конце предыдущей главы, то сейчас же вернитесь назад и сделайте его. Это необходимое условие.

Итак, вы немного заинтригованы этим странным заданием? Мы просили вас переместить элемент ** ** с идентификатором **elixirs** вверх и расположить его сразу под логотипом, а затем добавить небольшое свойство в правило для элемента с идентификатором **elixirs**:

```
float: right;
```

Посмотрите, какие изменения может внести одно маленькое свойство! Неожиданно страница преобразилась из милой, но достаточно обыкновенной в поистине потрясающую и разбитую на две колонки. Она сразу же стала лучше читаемой и более приятной для глаз.



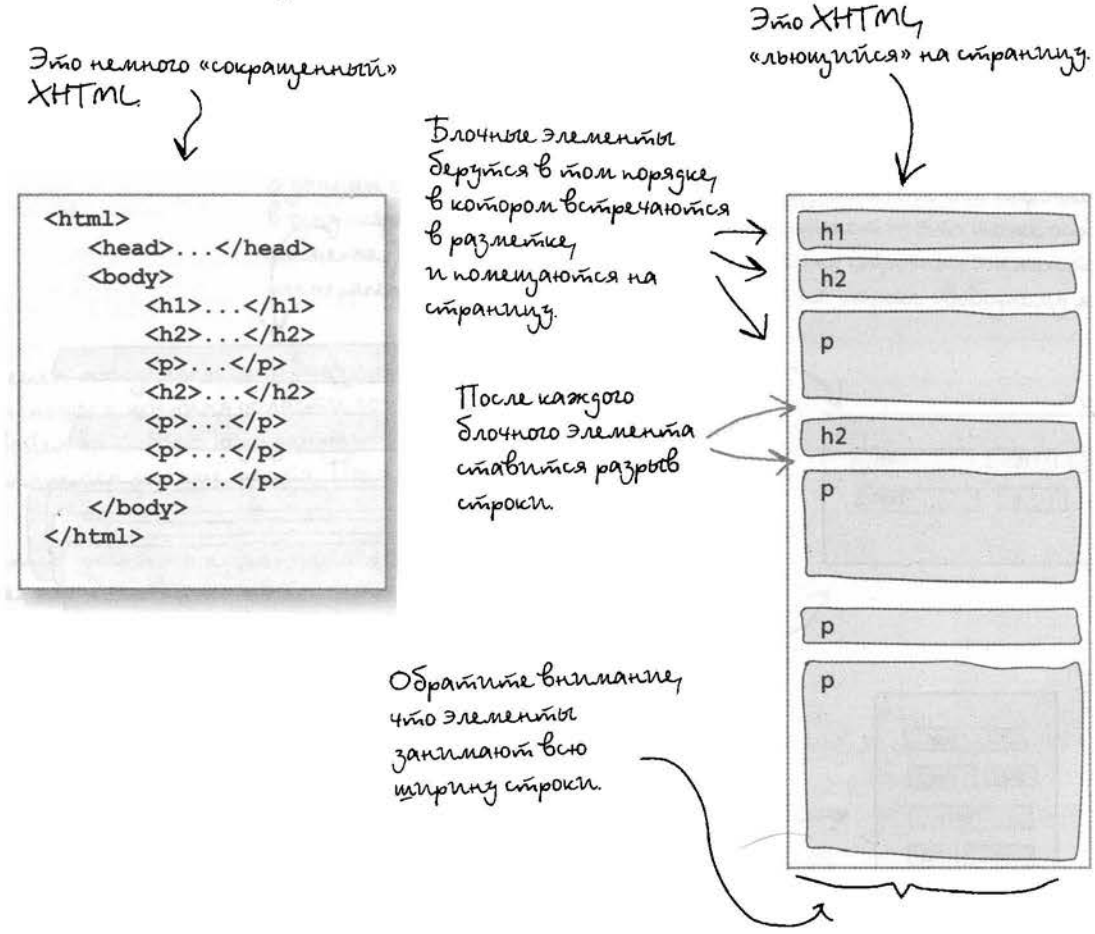
В чем же заключается волшебство? Как обычное свойство могло произвести такой потрясающий эффект? Можно ли использовать его, чтобы делать еще более интересные вещи на наших страницах? Конечно, можно. Но сначала вам нужно будет узнать, как браузер размещает элементы на странице.

Вы уже все знаете о блочных и строчных элементах и даже полностью разобрались с блочной моделью. Теперь вам остается узнать лишь то, как именно браузер берет все эти элементы со страницы и решает, куда их поместить.

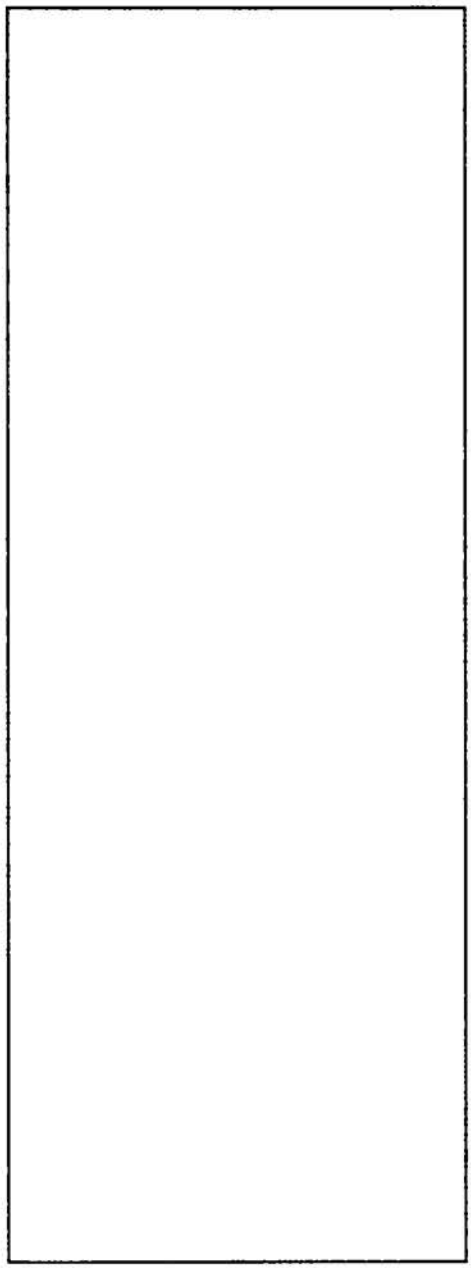
Используй поток, Люк

Поток – это то, что дает власть специалисту в области CSS. Это энергетическое поле, созданное всем живым. Оно окружает нас и проходит сквозь все. Оно связывает всю галактику... Ой, просим прощения.

Поток – это то, что браузер использует для разметки страниц с XHTML-элементами. Браузер начинает с самой вершины любого XHTML-файла, проходит сквозь весь поток элементов, отображая каждый встреченный элемент на странице один под другим. Если рассматривать только блочные элементы, то он ставит между ними разрывы строки. Итак, первый элемент файла отображается первым, затем идет разрыв строки, затем второй элемент, разрыв строки и т. д., с самого начала до самого конца вашего файла. Это поток.



Вот ваша страница. Залейте блочные элементы на страницу lounge.html.

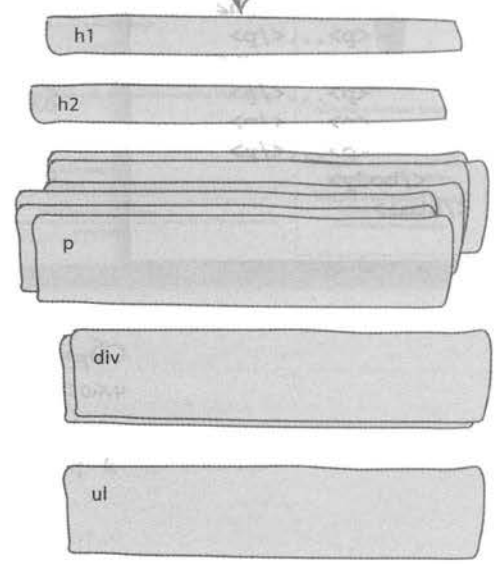




Поработайте браузером

Откройте файл lounge.html и найдите все блочные элементы. Залейте каждый из них на страницу слева. Сосредоточьтесь на блочных элементах, вложенных непосредственно в элемент body. Вы можете проигнорировать свойство float в своем CSS-коде, так как пока не знаете, что оно делает. Проверьте свой ответ, прежде чем двигаться дальше.

Это все блочные элементы, которые вам понадобятся для выполнения задания.



Как насчет строчных элементов?

Итак, вы знаете, что блочные элементы заливаются на страницу потоком сверху вниз, с разрывом строки после каждого. Достаточно просто. А как насчет строчных элементов?

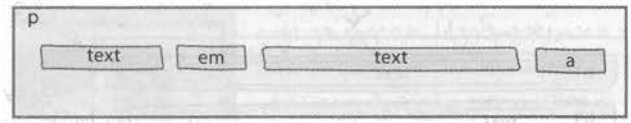
Строчные элементы заливаются один за другим: сверху вниз, слева направо. Вот как это работает.

Это еще один небольшой фрагмент XHTML-кода.

```
<p>
Заходите к нам <em>каждый вечер</em>,
чтобы попробовать освежающие <a
href="beverages/elixir.html" title="Напитки
гостевой Head First">напитки</a>.
</p>
```

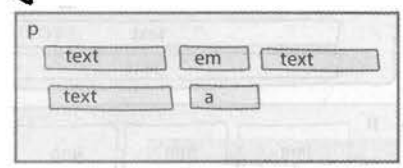
Если мы захотим залить на страницу строчное содержимое этого элемента <p>, то начнем с левого верхнего угла.

Строчные элементы размещаются один за другим в горизонтальном направлении, пока не встретится правая граница страницы.



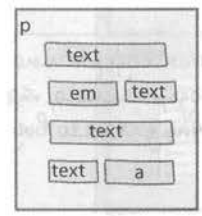
В данном случае места хватает, чтобы разместить все строчные элементы в одну строку. Обратите внимание, что текст — это тоже особый строчный элемент. Браузер разбивает его на несколько строчных элементов, имеющих размер, соответствующий оставшемуся на строке свободному месту.

А если мы сделаем окно браузера немного уже или уменьшим ширину области содержимого с помощью свойства width? Тогда для размещения строчных элементов останется меньше места. Посмотрим, как это работает.



Теперь содержимое заливается на страницу слева направо до тех пор, пока в строке есть место, а затем осуществляется переход на следующую строку. Обратите внимание, что браузеру приходится немного по-другому разбить текст на части, чтобы он хорошо помещался в окне.

А что случится, если мы сделаем область содержимого еще меньше? Браузер использует столько строк, сколько нужно, чтобы залить на страницу все содержимое.



Как все это работает

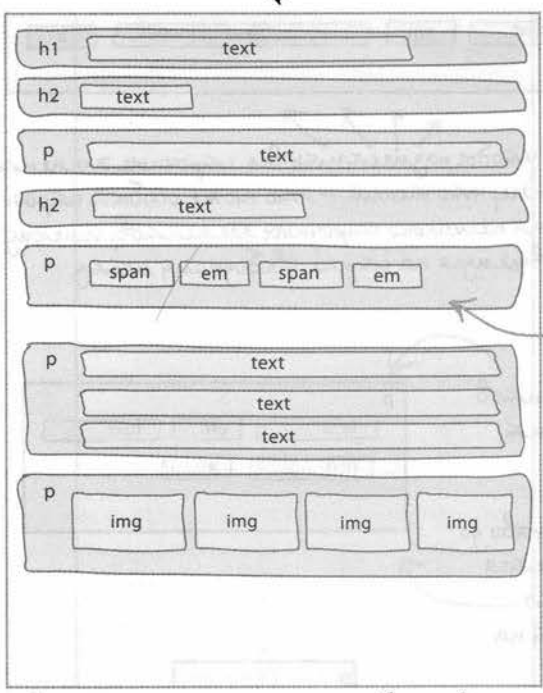
Теперь, когда вы знаете, как заливаются на страницу строчные и блочные элементы, давайте объединим их вместе. Мы будем использовать обычную страницу с заголовками, абзацами и несколькими строчными элементами, такими как пару элементов **span** и **em**, и даже изображениями. Не будем забывать и о текущем тексте.

Начинаем с окна браузера, размер которого должен быть изменен так, чтобы его ширина наилучшим образом нам подходила.

Блочные элементы заливаются на страницу сверху вниз, и между ними вставляется разрыв строки.

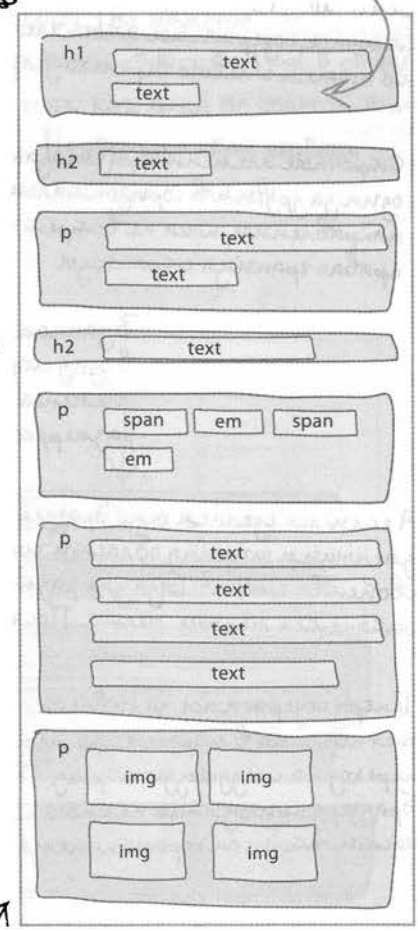
Строчные элементы заливаются слева направо, с переходом на новую строку в том месте, где достигается правая граница области содержимого.

Если строчное содержимое блока вмещается по ширине в область содержимого, то оно там и отображается. В ином случае для него выделяется следующая строка.



Здесь мы изменили размер окна браузера, чтобы все содержимое занимало меньше места в горизонтальном направлении.

Элементы заливаются на страницу аналогичным образом, хотя в некоторых местах строчные элементы занимают больше строк.



Теперь блочные элементы занимают больше места по вертикали, потому что приходится размещать строчное содержимое в более узком окне.

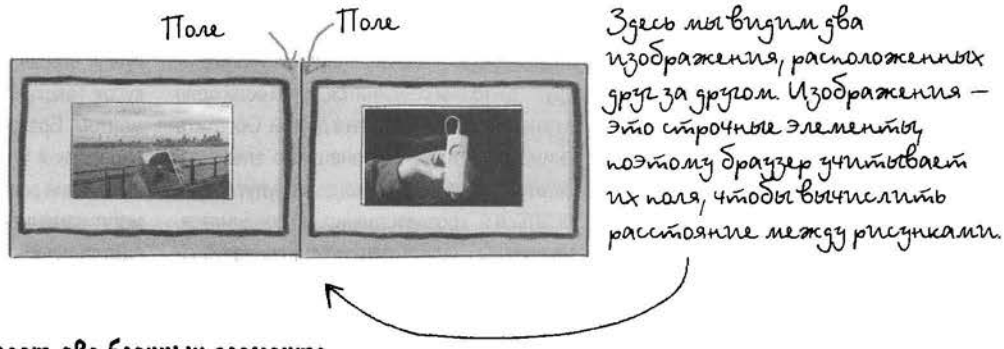
Еще один момент, который вы должны понять



Давайте немного углубимся в детали и рассмотрим еще один нюанс того, как браузер размещает на странице блочные и строчные элементы. Оказывается, он по-разному обращается с полями в зависимости от того, какой тип элемента отображается на странице.

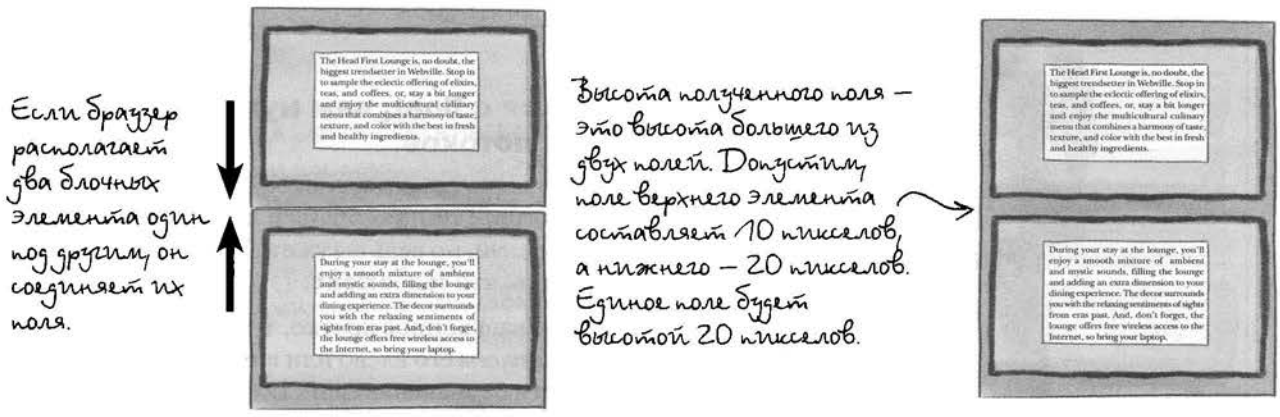
Если браузер располагает два строчных элемента один за другим...

Если перед браузером стоит задача разместить один за другим два строчных элемента и у этих элементов есть поля, то браузер поступит так, как вы того и ожидаете. Он оставит достаточно свободного пространства между элементами, чтобы отобразить поля обоих. Итак, если поле левого элемента составляет 10 пикселов, а правый элемент имеет поле шириной 20 пикселов, то между этими элементами останется промежуток 30 пикселов.



Если браузер располагает два блочных элемента один под другим...

В данном случае все намного интереснее. Если браузер располагает два блочных элемента один под другим, он превращает их поля в одно. Высота полученного поля равняется высоте большего из двух полей.



Часто Задаваемые Вопросы

В: Итак, если у меня есть блочный элемент без полей (0 пикселов), а под ним расположен блочный элемент с полем высотой 20 пикселов, то поле между двумя этими элементами будет высотой 20 пикселов?

О: Верно. Если одно поле больше другого, то для общего поля используется размер большего поля, даже если второе равняется 0 пикселов. Но если поля имеют одинаковый размер, скажем, по 10 пикселов, то они просто прячутся одно под другое и получается общее поле высотой все те же 10 пикселов.

В: На самом ли деле строчный элемент может иметь поля?

О: Конечно, может, хотя вы скоро заметите, что поля для строчных элементов задаются очень редко. Единственное исключение составляют изображения. Для них достаточно часто задаются не только поля, но и границы с отступами. И хотя в этой главе мы не будем задавать поля ни для каких строчных элементов, чуть позже рассмотрим, как задавать границу для одного из них.

В: А если у меня один элемент вложен в другой и оба имеют поля? Они тоже могут соединяться?

О: Да, такое может случиться. Если у вас есть два соприкасающихся друг с другом поля (верхнее — одного элемента, нижнее — другого), то они соединятся, даже если один из этих элементов вложен в другой. Обратите внимание, что если у внешнего элемента есть граница, то поля никогда не будут соприкасаться и, соответственно, не соединятся. Но если вы удалите границу, они превратятся в одно поле. Столкнувшись с этим в первый раз, постарайтесь не пугаться. Теперь же отложите эту информацию в дальний уголок сознания до тех пор, пока подобное не случится.

В: Как именно текст рассматривается в виде элемента, хотя на самом деле является содержимым?

О: Даже несмотря на то, что текст — это содержимое, браузеру необходимо залить его на страницу, верно? Сначала браузер выясняет, сколько текста поместится на текущей строке, а затем рассматривает этот кусок текста так, будто он является элементом. Браузер даже создает вокруг него небольшой блок. Как вы уже видели, при изменении размера страницы все эти блоки могут измениться, так как текст перестраивается, чтобы соответствовать размерам новой области содержимого.



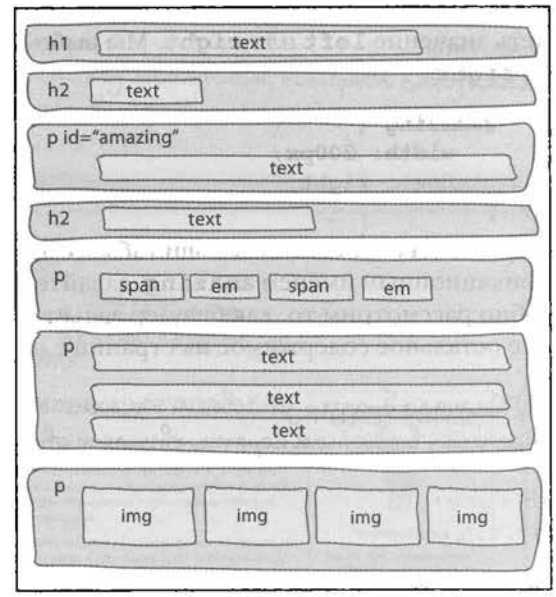
Чтобы разобраться с float, вам нужно хорошо понимать работу потоков.

Может быть, это всего лишь одно маленькое свойство, но то, как оно работает, тесно связано с тем, как браузер заливает на страницу элементы и содержимое. Эй, но ведь вы уже это знаете, так что мы можем начать объяснять, что такое `float`.

Итак, если ответить на ваш вопрос кратко, то свойство `float` сначала берет элемент и *смещает* его влево или вправо, насколько это возможно (в зависимости от значения свойства). Затем все содержимое, находящееся под элементом, обтекает этот элемент. Конечно же, есть еще кое-какие детали, которые мы сейчас рассмотрим...

Как создать плавающий элемент

Рассмотрим пример того, как сделать элемент плавающим и как он в результате отобразится на странице.



Дайте ему идентификационное имя

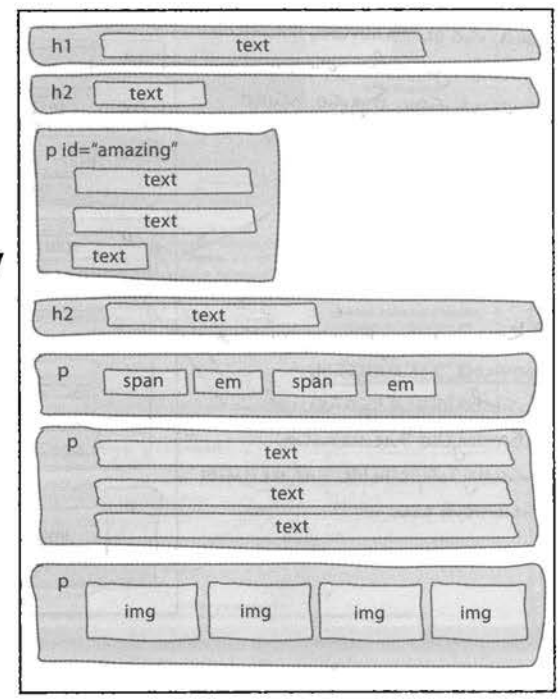
Возьмем один из этих абзацев и дадим ему идентификационное имя. Мы бы хотели назвать его amazing floating paragraph, но для краткости будем называть его просто amazing.

Как задать ширину

Требование для плавающих элементов — наличие заданной ширины. Определим ширину этого абзаца равной 200 пикселам.

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

Теперь ширина абзаца равна 200 пикселам, а его строчное содержимое подогнано под этот размер. Помните, абзац — это блочный элемент, поэтому никакие элементы не будут расположены рядом с ним, так как перед блочными элементами и после них добавляется разрыв строки.

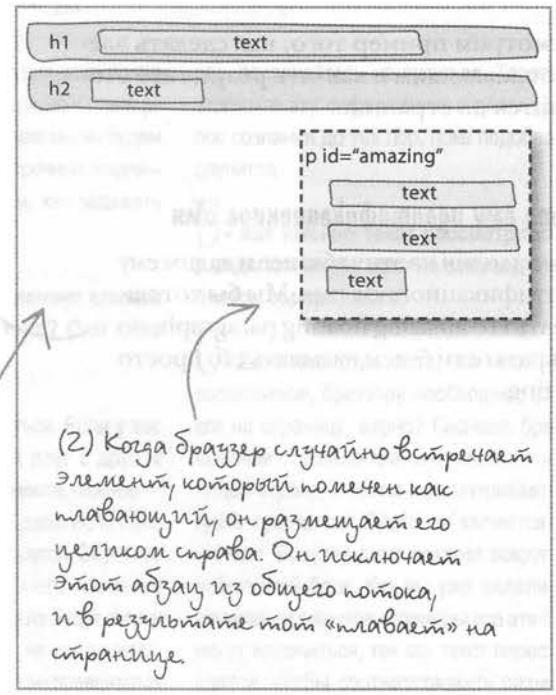


Сделайте его плавающим

Теперь добавим свойство **float**. Оно может иметь значение **left** или **right**. Мы выберем **right**:

```
#amazing {
   width: 200px;
   float: right;
}
```

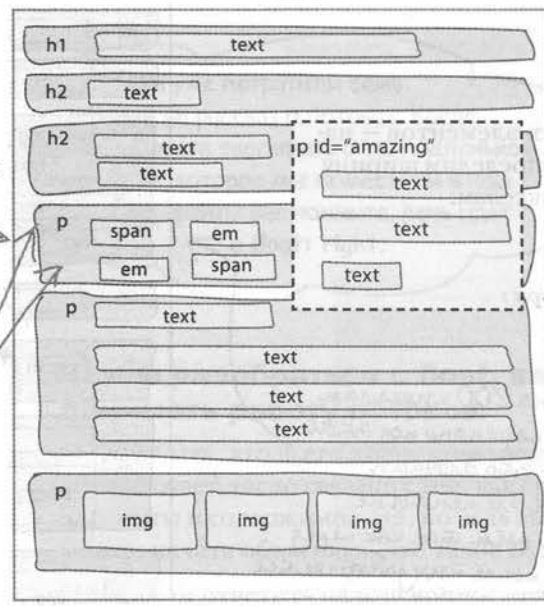
Теперь, когда у нас есть плавающий абзац с идентификационным именем **amazing**, давайте подробно рассмотрим то, как браузер зальет его и все остальное содержимое на страницу.



(1) Сначала браузер заливает элементы на страницу в обычном порядке, двигаясь сверху вниз.

(2) Когда браузер случайно встречает элемент, который помечен как плавающий, он размещает его целиком справа. Он исключает этот абзац из общего потока, и в результате тот «плавает» на странице.

(3) Поскольку плавающий абзац был исключен из общего потока, то остальные блочные элементы заливаются на страницу так, будто этого абзаца там вовсе не было.



Обратите внимание, что блочные элементы располагаются под плавающим элементом. Это происходит потому, что плавающий элемент больше не является частью общего потока.

(4) Когда позиционируются строчные элементы учитываются границы плавающего элемента, поэтому строчные элементы обтекают его.

Однако если строчные элементы заливаются на страницу вместе с блочными, они обтекают границы плавающего элемента.

Поэкспериментируем над плавающим элементом в гостевой

Теперь вы знаете, что такое поток и как плавающие элементы размещаются на странице. Вернемся в гостевую и посмотрим, как это совмещается.

Мы задали для элемента с идентификатором elixirs свойство float со значением right. Кроме того, мы переместили вверх и расположили его прямо под логотипом, который находится на самом верху страницы.

Перемещение позволило нам расположить его справа, а все остальные элементы страницы обтекают его слева. Если бы мы оставили с идентификатором elixirs под разделом с музыкальными композициями, то раздел с напитками сместился бы вправо после того, как все содержимое страницы уже было бы отображено выше в обычном режиме.

Все эти элементы следуют за элементом с идентификатором elixirs, так что они его обтекают.

Помните, что с идентификатором elixirs заливается на самый верх страницы. Все остальные элементы идут за ним, но, когда строчное содержимое заливается на страницу, учитываются границы раздела с напитками.

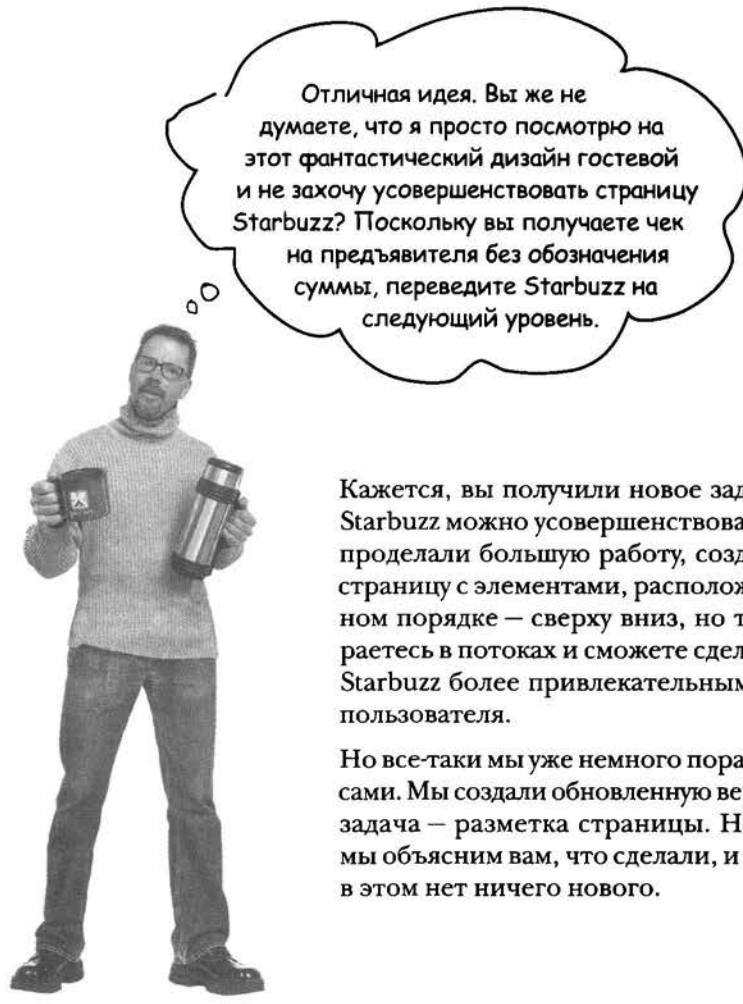
Обратите также внимание, что текст обтекает раздел с напитками и снизу, потому что он находится в блочных элементах, ширина которых равна ширине страницы. Если у вас текст не обтекает раздел с напитками снизу, попробуйте сузить окно браузера.





Упражнение

Переместите элемент `<div>` на его изначальное место — под списком музыкальных композиций, затем сохраните файл и обновите страницу. Где теперь плавает элемент? Проверьте свой ответ в конце главы, а затем верните элемент `<div>` с идентификатором `elixirs` под верхний колонтитул страницы.



Кажется, вы получили новое задание. Страницу Starbuzz можно усовершенствовать. Конечно, вы проделали большую работу, создав стандартную страницу с элементами, расположенными в обычном порядке — сверху вниз, но теперь вы разбираетесь в потоках и сможете сделать дизайн кафе Starbuzz более привлекательным и удобным для пользователя.

Но все-таки мы уже немного поработали над этим сами. Мы создали обновленную версию кафе. Ваша задача — разметка страницы. Не беспокойтесь, мы объясним вам, что сделали, и вы поймете, что в этом нет ничего нового.

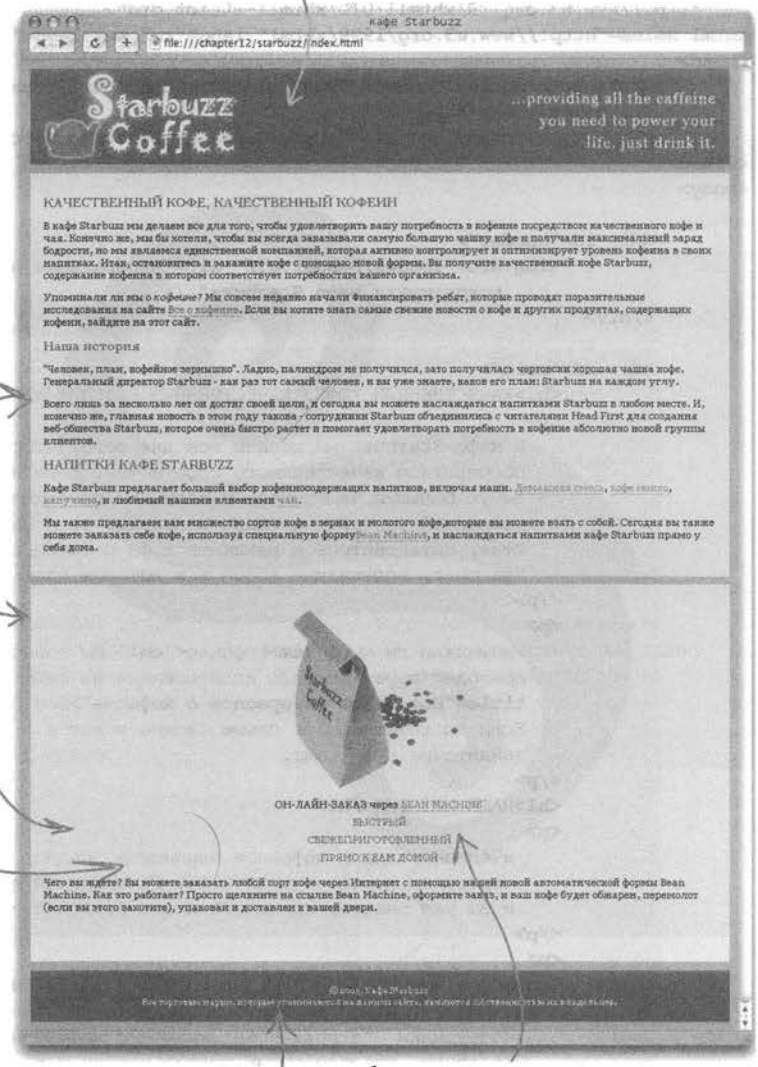
Новый сайт для Starbuzz

Посмотрим, что у нас есть на данный момент, и начнем с того, как сейчас выглядит страница. Затем мы рассмотрим разметку и CSS-код, который ее описывает.

У нас есть верхний колонтитул с замечательным логотипом Starbuzz и основной задачей компании. Это изображение в формате GIF.

Есть четыре раздела: верхний колонтитул, раздел с основным содержимым, раздел с рекламой какой-то новинки, называемой Bean Machine, и нижний колонтитул.

Каждый раздел — это элемент , который может быть оформлен независимо от остальных элементов.



Кажется, для всей страницы используется один и тот же фоновый цвет. Кроме того, у каждого элемента есть собственный фоновый рисунок.

Это раздел Bean Machine. Это ссылки на новый раздел кафе Starbuzz, где вы можете заказать себе напиток в режиме онлайн. Они пока не работают, потому что вы будете создавать новый раздел в следующей главе.

Это нижний колонтитул. У него нет фонового изображения, а есть лишь фоновый цвет.

Обратите внимание, что мы оформили ссылки необычным способом: подчеркнули их точечными линиями.

дальше ▸

Посмотрим на разметку

Теперь посмотрим на новую разметку страницы Starbuzz. Каждый логический раздел мы поместили в элемент ** **, которому присвоили соответствующий идентификатор. Кроме элементов ** ** и ****, вы не увидите здесь ничего такого, чего не видели ранее в главе 5. Итак, взгляните на код и ознакомьтесь со структурой, а затем переверните страницу, чтобы посмотреть на CSS-стили.

Это стандартное определение типа документа XHTML.

```
<!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="ru" xml:lang="ru" >
<head>
        <meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />
        <title>Starbuzz Кaффе</title>
        <link type="text/css" rel="stylesheet" href="starbuzz.css" />
</head>
<body>

        <div id="header">
                <img src="images/header.gif" alt="Изображение для верхнего
                     колонтитула кафе Starbuzz" />
        </div>

        <div id="main">
                <h1>КАЧЕСТВЕННЫЙ КОФЕ, КАЧЕСТВЕННЫЙ КОФЕИН</h1>
                <p>
                   В кафе Starbuzz мы делаем все для того, чтобы удовлетворить вашу потребность в кофеине
                   посредством качественного кофе и чая. Конечно же, мы бы хотели, чтобы вы всегда заказывали
                   самую большую чашку кофе и получали максимальный заряд бодрости, но мы являемся единственной
                   компанией, которая активно контролирует и оптимизирует уровень кофеина в своих напитках.
                   Итак, остановитесь и закажите кофе с помощью новой формы. Вы получите качественный кофе
                   Starbuzz, содержание кофеина в котором соответствует потребностям вашего организма.
                </p>
                <p>
                   Упоминали ли мы о <em>кофеине</em>? Мы совсем недавно начали Финансировать ребят, которые
                   проводят поразительные исследования на сайте <a href="http://buzz.headfirstlabs.com"
                   title="Все самое интересное о кофеине">Все о кофеине</a>.
                   Если вы хотите знать самые свежие новости о кофе и других продуктах, содержащих кофеин,
                   зайдите на этот сайт.
                </p>
                <h1>НАША ИСТОРИЯ</h1>
                <p>
                    «Человек, план, кофейное зернышко». Ладно, палиндром не получился, зато получилась
                    чертовски хорошая чашка кофе. Генеральный директор Starbuzz — как раз тот самый человек,
                    и вы уже знаете, каков его план: Starbuzz на каждом углу.
                </p>
                <p>
                    Всего лишь за несколько лет он достиг своей цели, и сегодня вы можете наслаждаться
                    напитками Starbuzz в любом месте. И, конечно же, главная новость в этом году такова —
                    сотрудники Starbuzz объединились с читателями Head First для создания веб-общества
                    Starbuzz, которое очень быстро растет и помогает удовлетворять потребность в кофеине
                    абсолютно новой группы клиентов.
                </p>
                <h1>НАПИТКИ КАФЕ STARBUZZ</h1>
                 <p>
```

Далее следуют элементы для верхнего колонтитула и для области основного содержимого.

Это все еще продолжается область основного содержимого.

```
            Кафе Starbuzz предлагает большой выбор кофеиносодержащих напитков, включая наши .
            <a href="beverages.html#house" title="Домашняя смесь">Домашняя смесь</a>,
            <a href="beverages.html#mocha" title="Кофе мокко">кофе мокко</a>,
            <a href="beverages.html#cappuccino" title="Капучино">капучино</a>,
            и любимый нашими клиентами
            <a href="beverages.html#chai" title="Чай">чай</a>.
        </p>
        <p>
            Мы также предлагаем вам множество сортов кофе в зернах и молотого кофе,
            которые вы можете взять с собой. Сегодня вы также можете заказать себе кофе,
            используя специальную форму <a href="form.html" title="The Bean Machine">
            Bean Machine</a>, и наслаждаться напитками кафе Starbuzz прямо у себя дома.
        </p>

</div>
```

```
<div id="sidebar">
        <p class="beanheading">
            <img src="images/bag.gif" alt="Bean Machine bag" />
            <br />
            ОН-ЛАЙН-ЗАКАЗ через
            <a href="form.html">BEAN MACHINE</a>
            <br />
            <span class="slogan">
                БЫСТРЫЙ <br />
                СВЕЖЕПРИГОТОВЛЕННЫЙ <br />
                ПРЯМО К ВАМ ДОМОЙ <br />
            </span>
        </p>
        <p>
            Чего вы ждете? Вы можете заказать любой сорт кофе через Интернет с помощью нашей
            новой автоматической формы Bean Machine. Как это работает? Просто щелкните на ссылке
            Bean Machine, оформите заказ, и ваш кофе будет обжарен, перемолот (если вы этого
            захотите), упакован и доставлен к вашей двери.
        </p>
</div>
```

Это элемент для Bean Machine. Мы присвоили ему идентификационное имя sidebar. Что бы это могло обозначать?

```
<div id="footer">
        &copy; 2005, Кафе Starbuzz
        <br />
        Все торговые марки, которые упоминаются на данном сайте,
        являются собственностью их владельцев.
</div>
```

И наконец, у нас есть элемент для нижнего колонтитула страницы.

```
</body>
</html>
```

А теперь посмотрим на стиль

Давайте внимательно посмотрим на CSS-код, который стилизует новую Starbuzz-страницу. Внимательно просмотрите все CSS-правила. Страница выглядит достаточно стильно, а CSS очень прост и понятен вам.

```
body {
        background-color: #b5a789;
        font-family:      Georgia, "Times New Roman", Times, serif;
        font-size:        small;
        margin:           0px;
}

#header {
        background-color: #675c47;
        margin:           10px;
        height:           108px;
}

#main {
        background: #efe5d0 url(images/background.gif) top left;
        font-size:  105%;
        padding:    15px;
        margin:     0px 10px 10px 10px;
}

#sidebar {
        background: #efe5d0 url(images/background.gif) bottom right;
        font-size:  105%;
        padding:    15px;
        margin:     0px 10px 10px 10px;
}

#footer {
        background-color: #675c47;
        color:            #efe5d0;
        text-align:       center;
        padding:          15px;
        margin:           10px;
        font-size:        90%;
}

h1 {
        font-size: 120%;
        color:     #954b4b;
}

.slogan { color: #954b4b;}

.beanheading {
        text-align:  center;
        line-height: 1.8em;
}
```

Сначала мы просто задаем основные свойства для элемента body: цвет фона, шрифты, а также устанавливаем для него поле шириной 0 пикселов. Так мы гарантируем отсутствие свободного пространства по краям страницы.

Затем идут правила для каждого логического раздела. В каждом из них мы определяем размер шрифта, добавляем отступы, а для элементов с идентификаторами main и sidebar задаем фоновое изображение.

Затем мы устанавливаем размер и цвет шрифта для заголовков.

И такой же цвет шрифта для класса slogan, который используется в элементе с идентификатором sidebar. Мы также определяем свойства для класса beanheading, который применяется там же.

```
a:link {
        color:                  #b76666;
        text-decoration:        none;
        border-bottom:          thin dotted #b76666;
}
a:visited {
        color:                  #675c47;
        text-decoration:        none;
        border-bottom:          thin dotted #675c47;
}
```

В последних двух CSS-правилах страницы Starbuzz мы используем псевдоклассы a:link и a:visited для стилизации ссылок.

Обратите внимание, что мы получаем интересный эффект в виде точечной линии под ссылками, задавая нижнюю границу вместо подчеркивания. Это отличный пример того, как можно использовать границу для строчных элементов.

Для описания свойства border-bottom мы использовали сокращение.

Переведем Starbuzz на следующий уровень

Цель такова: на сайте кафе Starbuzz сместить один раздел, а именно врезку Bean Machine, вправо, чтобы получилась красивая страница, состоящая из двух колонок. Вы уже делали нечто подобное для гостевой, верно? С учетом этого даем вам такое задание.

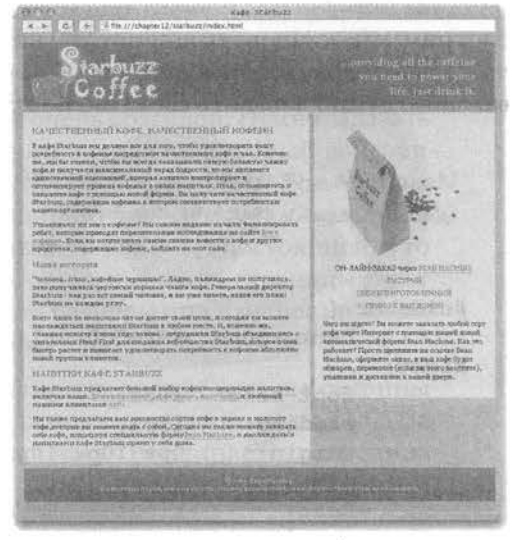
1. Используя атрибут **id**, присвойте уникальное имя элементу, который хотите сделать плавающим. Это уже сделано.

2. Убедитесь, что XHTML-код для вашего элемента находится прямо под описанием того элемента, под которым он должен «плавать». В данном случае это верхний колонтитул Starbuzz.

3. Установите для элемента ширину.

4. Сместите элемент влево или вправо. Кажется, нам было нужно, чтобы он был смещен вправо.

Итак, приступим. Всего за несколько простых шагов мы сделаем такое, за что генеральный директор Starbuzz пришлет пару чашек чая от заведения.



Мы получили отличную страницу, состоящую из двух отдельных колонок.

Поместите врезку прямо под верхний колонтитул

Когда вы создаете плавающий элемент, вам нужно переместить его XHTML-код прямо под тот элемент, под которым он должен «плавать». В данном случае врезка должна располагаться под верхним колонтитулом страницы. Итак, откройте XHTML-код в текстовом редакторе, найдите элемент ** ** с идентификатором **sidebar** и переместите его прямо под элемент ** ** с идентификатором **header**. Вы найдете нужный XHTML-код в файле `index.html` из папки `chapter12/starbuzz`. Когда справитесь с этим, сохраните изменения и обновите страницу.



Теперь врезка должна располагаться под разделом с основным содержимым.

Задайте ширину врезки и сделайте ее плавающей

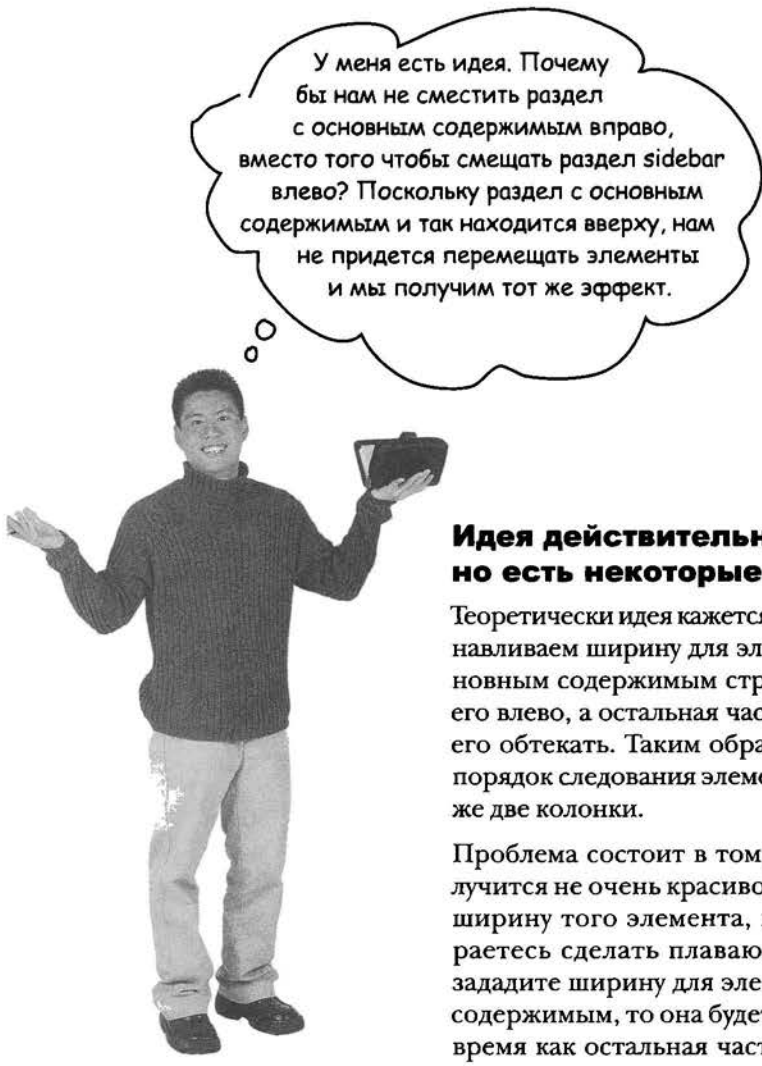
Рассмотрим, как установить ширину 280 пикселов для элемента с идентификатором **sidebar**. И чтобы сделать его плавающим, нужно добавить свойство **float**.

Используем селектор идентификатора, чтобы выбрать элемент с идентификатором sidebar.

```
#sidebar {
        background: #efe5d0 url(images/background.gif) bottom
right;
        font-size:  105%;
        padding:    15px;
        margin:     0px 10px 10px 10px;
        width:      280px;
        float:      right;
}
```

Мы задаем ширину области содержимого 280 пикселов.

Затем смещаем элемент с идентификатором sidebar вправо. Помните, он сместится вправо под верхним колонтитулом страницы настолько, насколько это возможно. Кроме того, элемент sidebar будет удален из общего потока. Все остальные элементы из XHTML, описанные после sidebar, будут обтекать его.



Идея действительно хороша, но есть некоторые проблемы.

Теоретически идея кажется хорошей. Мы устанавливаем ширину для элемента ** ** с основным содержимым страницы и смещаем его влево, а остальная часть страницы будет его обтекать. Таким образом мы сохраним порядок следования элементов и получим те же две колонки.

Проблема состоит в том, что страница получится не очень красивой. Нужно задавать ширину того элемента, который вы собираетесь сделать плавающим. Если же вы зададите ширину для элемента с основным содержимым, то она будет постоянной, в то время как остальная часть страницы будет менять свои размеры при изменении ширины окна браузера.

Обычно ширина врезок меньше ширины раздела с основным содержимым. И когда неосновная часть расширяется, страница выглядит просто ужасно. Итак, в хорошем дизайне должен расширяться раздел с основным содержимым, а не врезка.

Давайте все же проверим, как будет выглядеть страница, если принять эту идею за основу. Кроме того, поговорим еще немного о том, почему стоит заботиться даже о порядке, в котором расположены ваши разделы.

Тест для Starbuzz

Убедитесь, что вы добавили в файл `starbuzz.css` из папки `chapter12/starbuzz` новые свойства для врезки, и обновите страницу Starbuzz. Посмотрим, что получилось.

Что ж, выглядит неплохо, но если вы вернетесь на три страницы назад, то заметите, что мы получили не совсем то, что хотели.

Основное содержимое и врезка находятся слева и справа соответственно, но они все же не выглядят как две отдельные колонки.

Посмотрите, как фоновые изображения двух разделов наложились друг на друга. Кроме того, между колонками нет видимой границы.

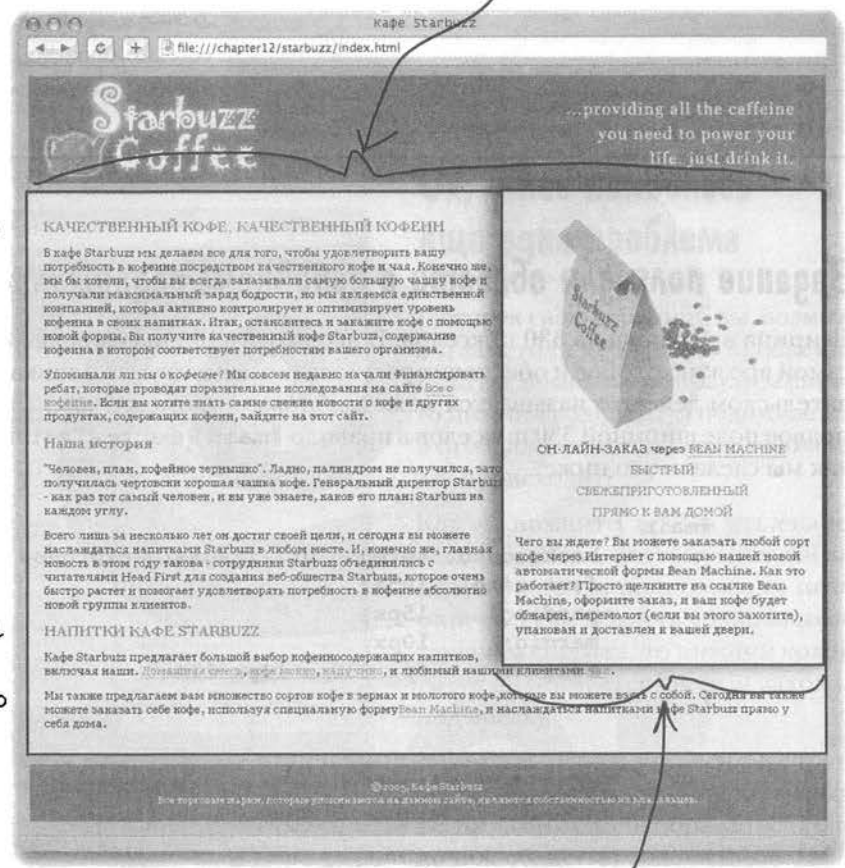


Текст обтекает врезку слева и снизу, что также не создает эффекта двух колонок на странице. Хмм, все получилось так же как в гостевой, и нам стоило этого ожидать.

Решение проблемы двух колонок

Вы сидите и ждете, что мы примчимся к вам и принесем волшебное свойство, которое решит эту проблему? Так вот, этого не случится. Это тот момент в CSS, когда процесс разметки страницы становится более похожим на искусство или, по крайней мере, на использование набора определенных технических приемов, чем на использование набора свойств, которые могут решить любую задачу. Итак, попытаемся решить эту проблему, применив общеизвестный технический прием. Как вы увидите, он несовершенен, но в большинстве случаев дает хороший результат. После этого мы покажем вам еще несколько методов, позволяющих достичь того же результата, а именно получить страницу, разделенную на две колонки. Здесь важно, чтобы вы понимали технические приемы и то, как они работают, могли применять их для решения собственных проблем и даже упрощать их в тех случаях, когда это необходимо.

Первое, что вам нужно запомнить, — врезка «плавает» на странице, а основное содержимое ее обтекает.



Что будет, если мы зададим для области с основным содержимым правое поле, которое будет иметь такую же ширину, как и вся врезка? Тогда ее содержимое растянется почти до самой врезки, но все же не вплотную.

В результате появится расстояние между двумя колонками. Поскольку поля прозрачны и не имеют фонового изображения, сквозь него будет показан фон самой страницы. А это как раз то, что нам нужно (вернитесь на несколько страниц назад, и вы увидите это).

Создадим поле шириной, равной ширине врезки.

Возьми в руку карандаш

Нужно задать для правого поля раздела с основным содержимым такую же ширину, как и у врезки. Но чему же она равняется? Мы надеемся, что вы еще не забыли то, что учили в прошлой главе. Приводим здесь информацию, которая понадобится для вычисления ширины врезки. Проверьте свой ответ в конце главы.

```
#sidebar {
        background: #efe5d0 url(images/background.gif) bottom right;
        font-size:  105%;
        padding:    15px;
        margin:     0px 10px 10px 10px;
        width:      280px;
        float:      right;
}
```

В этом правиле вы найдете все составляющие, необходимые для вычисления ширины врезки.

Задание поля для области с основным содержимым

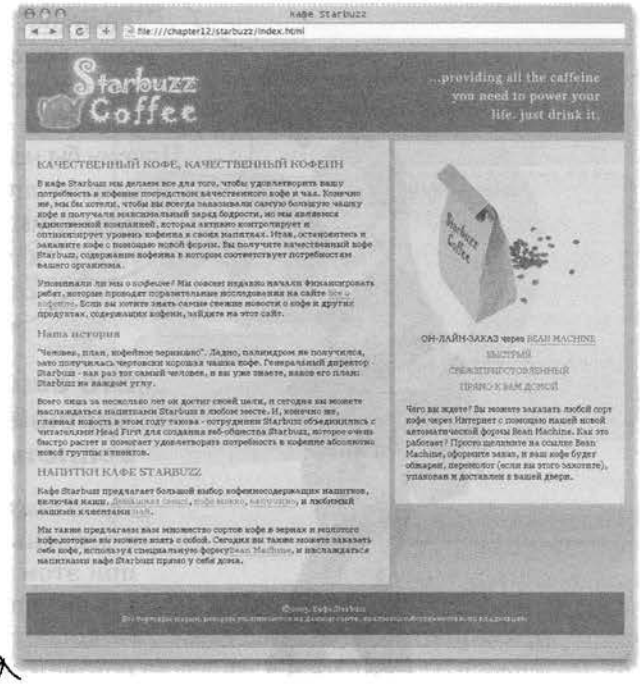
Ширина врезки равна 330 пикселов. В нее входят 10 пикселов левого поля самой врезки, которое и обеспечит расстояние между двумя колонками (в издательском деле это называется межстолбцовым промежутком). Добавьте правое поле шириной 330 пикселов в правило **#main** в файле `starbuzz.css`, как мы сделали это ниже:

```
#main {
        background: #efe5d0 url(images/background.gif) top left;
        font-size:  105%;
        padding:    15px;
        margin:     10px;
        margin:     0px 330px 10px 10px;
}
```

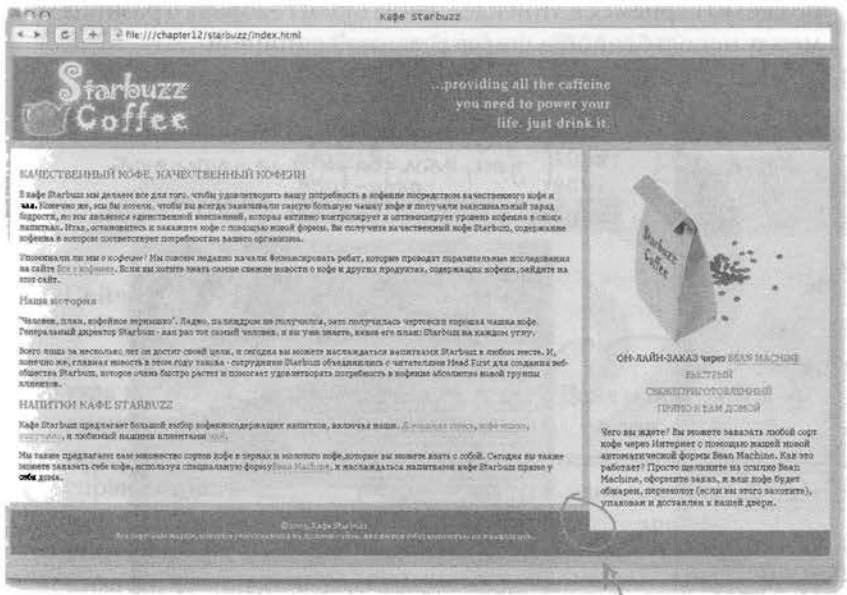
Мы задаем размер правого поля 330 пикселов, что равняется ширине врезки.

Тест

Как обычно, сохраните файл `starbuzz.css` и обновите страницу `index.html`. Теперь вы увидите промежуток между двумя колонками. Давайте еще раз разберемся с тем, как это работает. Врезка «плавает» справа, так что она смещается в эту сторону настолько, насколько это возможно, и весь ее элемент ** ** удаляется из общего потока и плавает по верху страницы. Теперь элемент ** ** для основного содержимого все еще занимает всю ширину окна браузера (потому что он является блочным элементом), но мы задали для него поле такой же ширины, как у врезки. В результате получаются две красивые колонки. Вы знаете, что блок элемента ** ** с идентификатором **main** все еще располагается прямо под врезкой, но мы никому об этом не скажем, если вы сами этого не сделаете.



Увеличив поле элемента с идентификатором main, мы создаем иллюзию двухколоночной разметки.



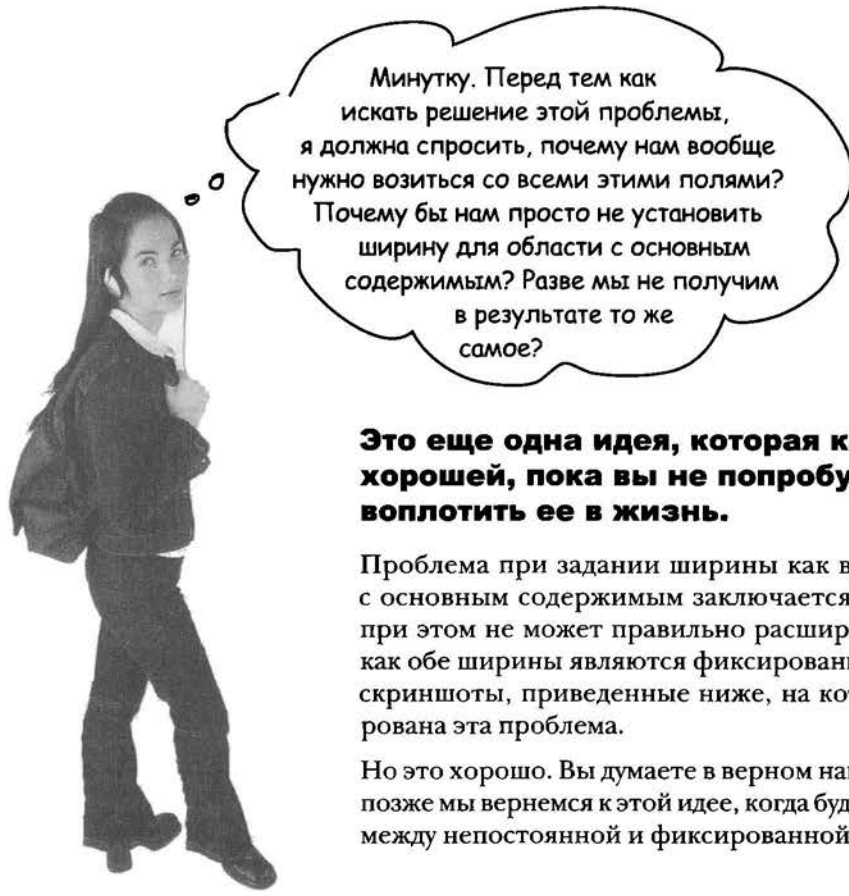
У нас появилась проблема. Если вы расширяете окно браузера, то нижний колонтитул и врезка частично перекрывают друг друга.

Ох, у нас появилась еще одна проблема

Тестируя свою страницу, вы, возможно, заметили небольшую проблемку. Если вы делаете окно браузера шире, то нижний колонтитул поднимается и отображается под врезкой. Почему так происходит?

Как вы помните, врезка удалена из общего потока, поэтому нижний колонтитул почти полностью ее игнорирует. Если область содержимого слишком коротка, то нижний колонтитул поднимается вверх и заходит под врезку.

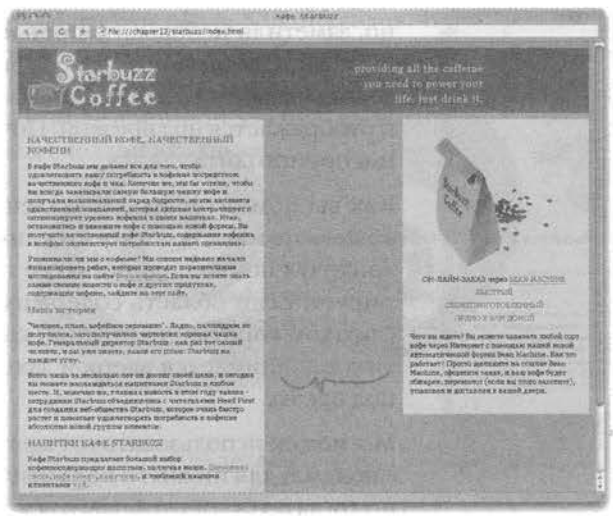
Мы можем использовать тот же трюк с полем и для нижнего колонтитула, но тогда он будет отображаться только под областью содержимого, а не по всей ширине страницы. Итак, что же теперь делать?



Это еще одна идея, которая кажется хорошей, пока вы не попробуете воплотить ее в жизнь.

Проблема при задании ширины как врезке, так и области с основным содержимым заключается в том, что страница при этом не может правильно расширяться и сужаться, так как обе ширины являются фиксированными. Посмотрите на скриншоты, приведенные ниже, на которых продемонстрирована эта проблема.

Но это хорошо. Вы думаете в верном направлении, и немного позже мы вернемся к этой идее, когда будем говорить о разнице между непостоянной и фиксированной шириной.



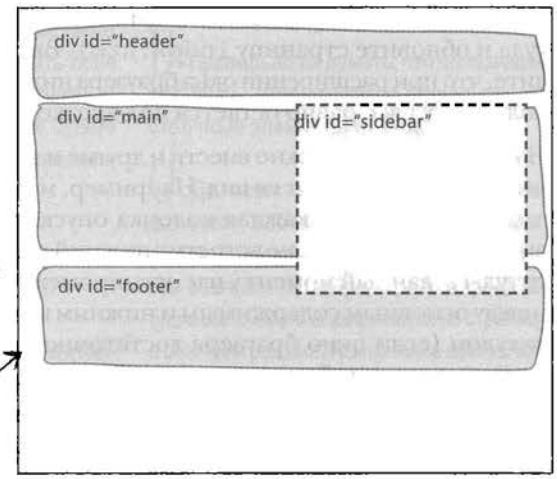
Когда окно браузера расширяется, они полностью разделяются.

Когда окно браузера становится уже, эти два раздела начинают накладываться друг на друга.



Вернемся к решению проблемы с наложением

Знаете, мы все-таки примчимся к вам и принесем с собой решение для вашей проблемы, но не привыкайте к этому. Решение – свойство **clear**. Посмотрим, как оно работает...



Вот то, что мы имеем на данный момент. Элемент с идентификатором main достаточно короткий, так что с идентификатором footer поднимается вверх и перекрывает с именем sidebar.

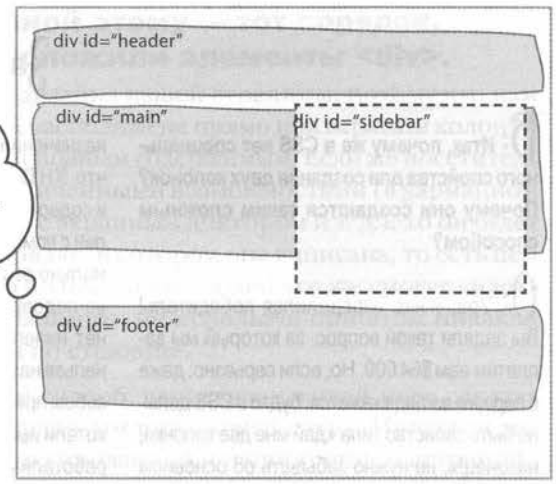
Это происходит потому, что врезка была удалена из общего потока. Итак, браузер просто располагает элементы с идентификаторами main и footer как обычно, игнорируя врезку. Однако не забывайте, что, когда браузер заливает строчные элементы, он учитывает границы врезки, а их содержимое обтекает ее, не перекрывая.

Эту проблему можно решить с помощью CSS-свойства **clear**. Задав его, вы укажете браузеру, чтобы с левой, правой или сразу с обеих сторон элемента не было никакого плавающего содержимого, так как элемент заливается на страницу с общим потоком. Давайте попробуем...

```
#footer {
        background-color: #675c47;
        color:            #efe5d0;
        text-align:       center;
        padding:          15px;
        margin:           10px;
        font-size:        90%;
        clear:            right;
}
```

Здесь мы добавляем новое свойство в правило для нижнего колонтитула. Оно говорит, что с правой стороны элемента не может быть никакого плавающего содержимого.

Сейчас, когда браузер располагает элементы на странице, он смотрит, нет ли справа от нижнего колонтитула плавающего элемента. Если он видит такой элемент, то смещает нижний колонтитул вниз до тех пор, пока справа от него не будет никаких элементов. Теперь неважно, насколько широким вы сделаете окно браузера, нижний колонтитул всегда будет располагаться под врезкой.



Даже и не думайте о том, чтобы поместить справа от меня плавающий элемент.

Теперь нижний колонтитул располагается под врезкой и справа от него нет плавающих элементов.

Тест

Итак, вперед! Добавьте в файле `starbuzz.css` свойство **`clear`** в правило для нижнего колонтитула и обновите страницу `index.html`. Вы увидите, что при расширении окна браузера нижний колонтитул все равно остается под врезкой.

На этой странице можно внести и другие изменения, которые улучшат ее вид. Например, можно сделать так, чтобы каждая колонка опускалась вниз до тех пор, пока не встретит нижний колонтитул. На данный момент у нас есть промежуток между основным содержимым и нижним колонтитулом (если окно браузера достаточно большой ширины) либо между врезкой и нижним колонтитулом (если ширина окна средняя).

К сожалению, исправить это нелегко, и в этой главе мы даже не будем пытаться это сделать. Разметка в CSS — искусство, и ни одно из решений не является совершенным. Если все сделано правильно, то CSS-разметка даст вам отличную веб-страницу.

Далее мы рассмотрим еще несколько способов разметки страницы в CSS с использованием **`float`**.

В CSS есть множество методов для реализации любых замыслов разработчика, и каждый из них имеет свои сильные и слабые стороны.

Теперь наши проблемы с нижним колонтитулом решены. Он всегда будет расположен под врезкой, независимо от ширины окна браузера.



Часто Задаваемые Вопросы

В: Итак, почему же в CSS нет специального свойства для создания двух колонок? Почему они создаются таким сложным способом?

О: Ура, у нас определился победитель! Вы задали такой вопрос, за который мы заплатим вам $64 000. Но, если серьезно, даже с первого взгляда кажется, будто в CSS должно быть свойство типа «дай мне две колонки, наконец!», не нужно забывать об основном назначении XHTML и CSS. Предполагается, что XHTML — это формат для структуры и содержимого, который может быть оформлен с помощью CSS, но который должен нормально отображаться даже на устройствах, не поддерживающих CSS. Таким образом, нет ничего удивительного в том, что CSS нельзя назвать единственным и лучшим способом презентации документов. Если бы мы хотели иметь такой способ, то, скорее всего, работали бы в Microsoft Word. Но CSS дает несколько замечательных инструментов для создания разметок, которые отлично выглядят и удобны в использовании.

В: Можно ли сделать так, чтобы элемент «плавал» по центру?

О: Нет, CSS позволяет создавать элементы, плавающие только справа или слева. Хорошенько вдумавшись, вы поймете, что, если бы такая возможность все-таки была,

строчному содержимому пришлось бы обтекать плавающий по центру элемент с двух сторон. Хотя это, вероятнее всего, было бы выполнимо, читаемость документа явно бы ухудшилась и вряд ли бы он хорошо смотрелся.

В: Сворачиваются ли поля для плавающих элементов?

О: Нет, и очень легко понять почему. В отличие от блочных элементов, которые заливаются на страницу сплошным потоком, плавающие элементы — всего лишь плавающие элементы. Другими словами, поля плавающих элементов не соприкасаются с полями элементов из потока, поэтому они не могут сворачиваться.

Но это затрагивает один важный момент, который влечет за собой ошибки в разметках. Если есть область с основным содержимым и врезка, то достаточно часто для них обеих задают верхние поля. Затем, если врезку делают плавающей, то она все еще имеет поле, но оно больше не будет соединяться с тем, что находится над этой врезкой. Таким образом, вы легко можете получить поля разных размеров для области с основным содержимым и для врезки, если не будете помнить о том, что поля плавающих элементов никогда не сворачиваются.

В: Можно ли сделать строчный элемент плавающим?

О: Да, конечно, можно. Самый лучший и наиболее часто встречающийся пример этого — плавающие изображения. Попробуйте сделать плавающим либо слева, либо справа изображение из абзаца, и вы увидите, что текст начнет обтекать его с соответствующей стороны. Не забудьте добавить отступы, чтобы вокруг изображения появилось немного свободного места, и, если хотите, добавьте границу. Вы также можете сделать плавающим любой другой строчный элемент, но такое встречается редко.

В: Правильно ли думать, что плавающие элементы игнорируются блочными, но строчные элементы их видят?

О: Да, можно думать и так. Строчное содержимое, вложенное в блочный элемент, всегда обтекает плавающий элемент, принимая во внимание его границы, в то время как блочные элементы заливаются на страницу в обычном режиме. Исключение составляет тот случай, когда вы задаете свойство clear для блочного элемента, в результате чего он смещается вниз до тех пор, пока справа, слева или с обеих сторон (в зависимости от значения свойства) не будет никаких плавающих элементов.



Действительно. Виной этому — тот порядок, в котором мы расположили элементы .

Это один из недостатков дизайна нашей страницы, потому что нам нужно, чтобы врезка была расположена прямо под верхним колонтитулом и перед разделом с основным содержимым. Если же посетитель использует браузер с ограниченными возможностями (в карманном ПК, мобильном телефоне, с экранным диктором и т. д.), то он будет видеть страницу в том порядке, в котором она написана, то есть первой будет врезка. Однако большинство людей все же смогут видеть область основного содержимого, не используя при этом никаких средств для перемещения по странице.

Итак, рассмотрим другой способ: вернемся к вашей идее сделать область с основным содержимым плавающей слева. При этом мы поговорим о дизайне с фиксированной и непостоянной шириной.



Упражнение Смотри, ма, нет CSS!

Хотите знать, как будут выглядеть ваши страницы для посетителей, использующих браузеры, которые не поддерживают CSS? Откройте файл `index.html` и удалите элемент **`<link>`** из **`<head>`**, сохраните изменения и обновите страницу в браузере. Теперь вы сможете увидеть действительный порядок элементов (или услышать, если используете экранный диктор). Итак, вперед! Попробуйте это сделать сами. Только не забудьте затем вернуть все на свои места (в конце концов, в этой главе мы учим CSS).

Это страница Starbuzz без CSS. По большому счету мы в хорошей форме. Страница все еще хорошо читается, хотя раздел Bean Machine идет перед основным содержимым, а это все-таки не то, чего мы хотим.



Левее, выше, правее...

На странице Starbuzz поменяем роли для области основного содержимого и врезки, чтобы теперь область с основным содержимым стала плавающей слева. Посмотрим, как это работает, а затем сделаем так, чтобы это работало *действительно хорошо*. Вот как мы изменим страницу всего за несколько простых шагов.

Шаг первый: начнем с врезки.

По сути, мы меняем роли области основного содержимого и врезки. Теперь область основного содержимого будет иметь фиксированную ширину и станет плавающей, в то время как врезка будет обтекать ее. Мы также будем использовать кое-какие технические приемы для работы с полями, чтобы визуально разделить две области. Но прежде, чем менять CSS, откройте файл `index.html` и переместите элемент ** ** с идентификатором **sidebar** под ** ** с идентификатором **main**. После этого внесите следующие изменения в CSS-правило для идентификатора **sidebar**:

```
#sidebar {
        background:   #efe5d0 url(images/background.gif) bottom right;
        font-size:    105%;
        padding:      15px;
        margin:       0px 10px 10px 470px;
        width:        280px;
        float:        right;
}
```

Поскольку сейчас врезка будет идти в потоке после основного содержимого, нам необходимо переместить большое поле во врезку. Общая ширина области содержимого составляет 470 пикселов. Можете вычислить ее сами, когда у вас будет свободное время. Вычисляйте ее так же, как вы делали это для врезки. Мы собираемся задать области основного содержимого ширину 420 пикселов.

Задаем фиксированную ширину для элемента с основным содержимым, так что удалите из элемента с идентификатором sidebar свойства width и float.

Шаг второй: позаботимся об области с основным содержимым.

Теперь нам нужно сделать плавающим элемент ** ** с идентификатором **main**. Вот как это делается:

```
#main {
        background: #efe5d0 url(images/background.gif) top left;
        font-size:    105%;
        padding:      15px;
        margin:       0px 10px 10px 10px;
        width:        420px;
        float:        left;
}
```

Меняем правое поле с 330 пикселов обратно на 10 пикселов

Нужно задать точную ширину, потому что мы хотим сделать этот элемент плавающим. Пусть она будет 420 пикселов.

Мы хотим, чтобы элемент с идентификатором main «плавал» слева.

Шаг третий: позаботимся о нижнем колонтитуле.

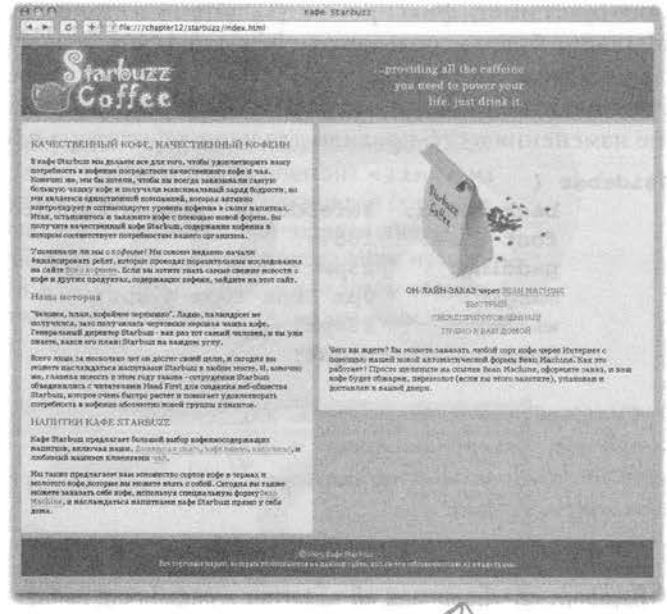
Теперь нам просто нужно привести в порядок нижний колонтитул, чтобы поменять значение свойства **clear** с **right** на **left**.

```
#footer {
        background-color: #675c47;
        color:            #efe5d0;
        text-align:       center;
        padding:          15px;
        margin:           10px;
        font-size:        90%;
        clear:            left;
}
```

Измените значение свойства clear с right на left. Таким образом нижний колонтитул никогда не перекроется областью основного содержимого.

Быстрый тест

Мы уже говорили, что, если сделать область основного содержимого плавающей слева, то будут кое-какие проблемы. Прежде чем продолжить работу, выполните небольшой тест и посмотрите, как это будет работать. Внесите изменения в файле `starbuzz.css` и обновите в браузере страницу `index.html`. Внимательно рассмотрите, как отображается страница, если сделать окно браузера узким, средней ширины или широким.



На самом деле все выглядит достаточно хорошо, и теперь элементы расположены в правильном порядке. Но не очень хорошо то, что врезка может растягиваться; она смотрелась намного лучше, когда имела фиксированную ширину. Часто врезки используются для навигации, поэтому не стоит делать их растягивающимися.

Когда мы сделали элемент с идентификатором sidebar плавающим справа, дизайн стал достаточно красивым и область основного содержимого могла растягиваться. Однако, сделав область основного содержимого плавающей слева, мы пришли к тому, что дизайн стал выглядеть гораздо хуже, а растягиваться начала врезка.

МОЗГОВОЙ ШТУРМ

Если рассматривать это с точки зрения дизайна, первый вариант был лучше, в то время как с точки зрения содержимого лучше выбрать второй вариант (из-за порядка расположения элементов ** **). Можно ли объединить преимущества этих двух вариантов: чтобы врезка имела фиксированную ширину, но элемент ** ** с идентификатором **main** все же шел первым в XHTML? Какие дизайнерские решения могут помочь получить этот результат?

Дизайны с фиксированной и непостоянной шириной



Все дизайны, с которыми мы работали до сих пор, называются дизайнами с непостоянной шириной, потому что они растягиваются, чтобы соответствовать размерам окна браузера. Такая разметка удобна, потому что, когда содержимое растягивается, заполняется все свободное место в окне браузера и место на экране применяется с пользой. Но иногда бывает важно, чтобы разметка была фиксированной. При этом пользователь изменит размер окна браузера, а страница будет выглядеть так, как должна. Подобные разметки *блокируют элементы*, фиксируя их положения на странице так, что они вообще не могут перемещаться. Это позволяет избавиться от множества проблем, которые появляются в результате изменения размеров окна браузера. Попробуем создать фиксированную разметку.

Чтобы перейти от текущей страницы к фиксированной, нужно лишь добавить кое-что в XHTML-код и написать одно правило в CSS.

Изменения в XHTML

В свой XHTML-код вы добавите новый элемент ** ** с идентификатором **allcontent**. Как можно догадаться по названию, этот ** ** будет включать в себя все содержимое страницы. Итак, поместите открывающий тег ** ** перед элементом ** ** с идентификатором **header**, а закрывающий тег — после элемента ** ** с идентификатором **footer**.

```
<body>
  <div id="allcontent">
    <div id="header">
      ... остальной XHTML будет здесь ...
    </div>
  </div>
</body>
```

Добавьте новый с идентификатором allcontent и заключите в него все элементы, вложенные в <body>.

Этот элемент закрывает с идентификатором footer.

Изменения в CSS

Теперь мы будем использовать этот ** **, чтобы заключить все элементы и содержимое внутри элемента ** ** с идентификатором **allcontent** в область фиксированной ширины, равную 800 пикселам. Рассмотрим CSS-правило, которое делает это:

```
#allcontent {
        width:              800px;
        padding-top:        5px;
        padding-bottom:     5px;
        background-color:   #675c47;
}
```

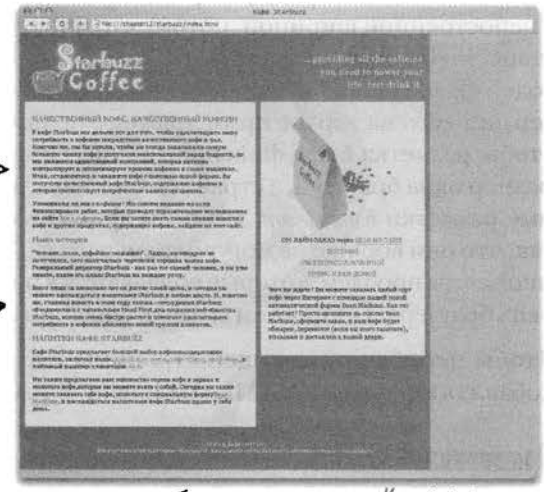
Зададим ширину элемента allcontent равной 800 пикселов. Это произведет следующий эффект: все находящееся внутри его будет заключено в область шириной 800 пикселов.

Поскольку мы впервые оформляем этот элемент , определим для него отступы и фоновое изображение. Вы увидите, что это поможет объединить всю страницу.

Внешний элемент с идентификатором allcontent всегда будет иметь ширину 800 пикселов, даже если размеры браузера будут меняться. Таким образом, мы эффективно заморозим размеры элемента вместе со всем, что находится внутри его.

Тест для фиксированного дизайна

Итак, вперед! Добавьте это правило в самый конец файла `starbuzz.css` и обновите страницу `index.html`. Теперь вы поймете, почему мы называем это фиксированной разметкой. Она не меняется при изменении размеров браузера.



Теперь элемент с идентификатором allcontent имеет ширину 800 пикселов, независимо от того, как меняется размер окна браузера. Поскольку остальные элементы находятся внутри элемента с идентификатором allcontent, они так же будут расположены в пространстве шириной 800 пикселов. Итак, в сущности, ширина страницы заморожена и теперь составляет 800 пикселов.

Это, конечно же, решает проблему с растягиванием врезки, а страница выглядит достаточно красиво.

Однако мы еще не все сделали, поэтому продолжим работу.



Что находится между разметками с фиксированной и непостоянной шириной? Гибкая разметка, конечно же!

Фиксированная разметка имеет кое-какие преимущества, но она явно плохо выглядит, если окно браузера очень широкое. Далее мы исправим это, и вы увидите, что такой дизайн часто используется в Сети. Такая разметка – промежуточная между фиксированной и непостоянной. Она имеет соответствующее имя: *гибкая*. Гибкие разметки фиксируют ширину области содержимого на странице, но центрируют ее в окне браузера. На самом деле будет намного проще изменить имеющуюся разметку на гибкую и дать вам возможность самим с ней поэкспериментировать, а не объяснять, как это все работает.

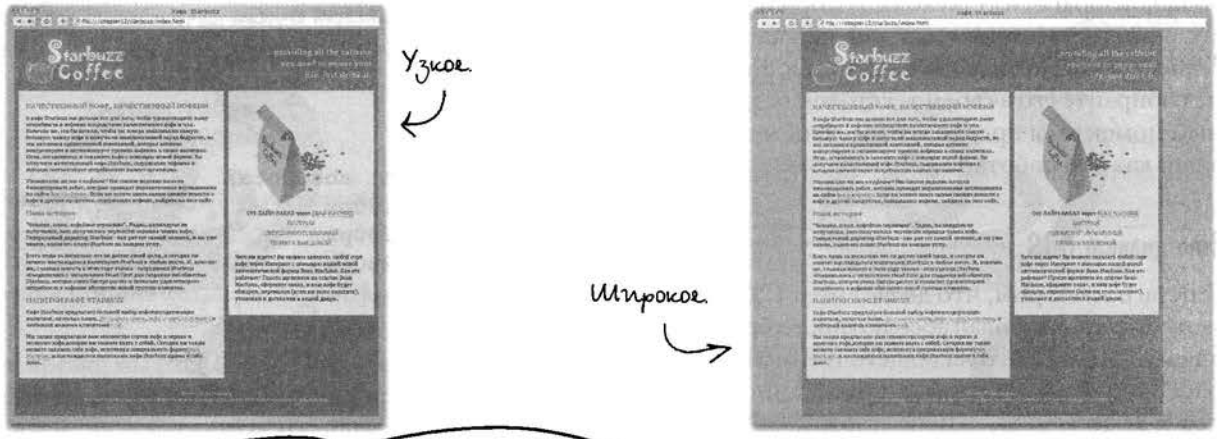
```
#allcontent {
        width: 800px;
        padding-top:    5px;
        padding-bottom: 5px;
        background-color: #675c47;
        margin-left:  auto;
        margin-right: auto;
}
```

Вместо того чтобы изменять размер левых и правых полей элемента с идентификатором allcontent, мы устанавливаем для них значение auto.

Когда речь шла о задании ширины auto для области основного содержимого, браузер растягивал ее настолько, насколько это было нужно. Для полей с шириной auto браузер выясняет, какой фактически должен быть размер, а также делает левое и правое поля одинаковыми, чтобы содержимое было центрировано.

Тест для гибкой разметки

Добавьте два свойства **margin** в файл `starbuzz.css` и обновите страницу. Теперь поэкспериментируйте с размерами окна браузера. Неплохо, ведь так?



Итак, если мы хотим, чтобы наше содержимое отображалось в правильном порядке, нам придется либо смириться с расширяющейся врезкой, либо использовать гибкую разметку. Или есть и другие способы добиться этого?



В CSS существует множество подходов к выбору разметки, и каждый имеет свои достоинства и недостатки. На самом деле мы как раз собирались рассмотреть другой известный способ создания двухколоночной разметки, при котором порядок содержимого остается правильным и есть возможность избежать некоторых проблем разметки с непостоянной шириной. Однако, как вы сами увидите, в нем тоже есть свои минусы.

В этот раз мы вообще не будем делать элементы плавающими. Вместо этого мы будем использовать особую возможность CSS, которая позволяет строго определять *месторасположение* элементов на странице. Это *абсолютное позиционирование*. Вы можете использовать его не только для создания мультиколоночной разметки, но и для некоторых дополнительных эффектов оформления, пример чему мы также рассмотрим.

Вернемся к исходному XHTML и CSS, с которого начали эту главу. Вы можете найти свежие копии этих файлов в папке `chapter12/absolute`. Еще раз взгляните на эти файлы, чтобы убедиться, что помните, как они выглядели сначала. Вернемся к первоначальному набору элементов ** **: для верхнего колонтитула, для области основного содержимого, для нижнего колонтитула и для врезки. Вспомните также, что в исходном XHTML-коде элемент ** ** с идентификатором **sidebar** располагался под областью основного содержимого, как раз там, где нам нужно.

Как работает абсолютное позиционирование

Сначала рассмотрим, что такое абсолютное позиционирование и как оно работает. Здесь приводится небольшой фрагмент CSS-кода, который располагает на странице элемент ** ** с идентификатором **sidebar** с применением абсолютного позиционирования. Пока не набирайте его в своем текстовом редакторе. На данный момент мы просто хотим, чтобы вы прочувствовали, как это работает.

В первую очередь мы используем свойство position, чтобы показать, что элемент будет позиционирован абсолютно.

```
#sidebar {
    position: absolute;
    top:   100px;
    right: 200px;
    width: 280px;
}
```

Затем задаем свойства top и right.

Определяем ширину для элемента .

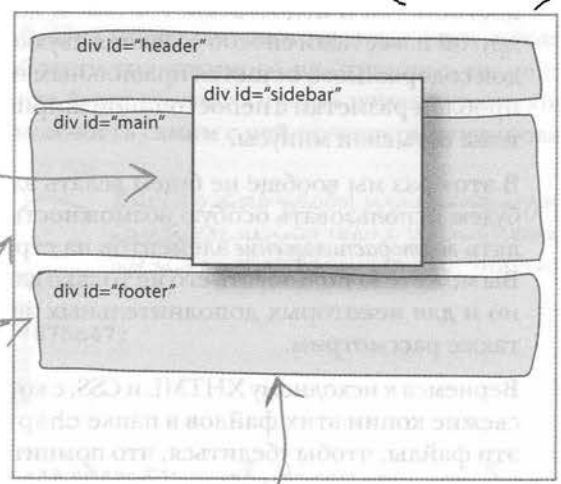
Что делает CSS

Теперь посмотрим, что делает этот CSS-код. Если элемент позиционирован абсолютно, то браузер в первую очередь удаляет его из общего потока. Затем браузер располагает элемент в месте, заданном свойствами **top** и **right** (можете использовать свойства **bottom** и **left**). В данном случае врезка будет расположена на расстоянии 100 пикселов от верхней границы страницы и 200 пикселов от его правой границы. Кроме того, мы определили ширину для элемента ** **, как делали это, когда он был плавающим.

Врезка расположена на расстоянии 200 пикселов от правой границы страницы

Врезка расположена на расстоянии 100 пикселов от верхней границы страницы

Поскольку врезка теперь позиционирована абсолютно, она удаляется из общего потока и располагается на странице так, как это задают свойства top, left, right или bottom.



Поскольку врезка покинула общий поток, другие элементы вообще не знают, что она есть на странице, и полностью ее игнорируют.

Строчное содержимое элементов из общего потока не обтекает блок абсолютно позиционированного элемента. Они вообще не обращают внимания на то, что этот элемент присутствует на странице.

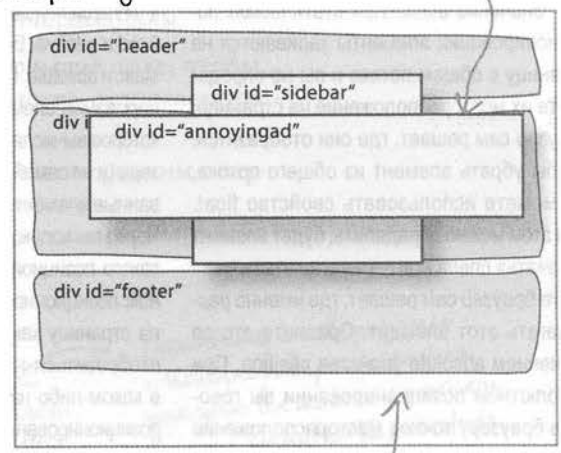
Еще один пример абсолютного позиционирования

Рассмотрим еще один пример. Допустим, у нас есть еще один элемент ** ** с идентификатором **annoyingad**. Мы можем позиционировать его следующим образом:

```
#annoyingad {
    position: absolute;
    top:    150px;
    left:   100px;
    width: 400px;
}
```

Элемент располагается на расстоянии 100 пикселов от левой границы страницы и 150 пикселов от верхней. Он также немного шире врезки, его ширина составляет 400 пикселов.

Теперь у нас есть второй абсолютно позиционированный элемент , расположенный на расстоянии 100 пикселов от левой границы страницы и 150 пикселов от верхней.



Обратите внимание, что с идентификатором annoyingad находится над элементом с идентификатором sidebar.

Как и врезку, мы расположили элемент ** ** с идентификатором **annoyingad** в конкретном месте на странице. Все элементы из общего потока, находящиеся ниже, не имеют ни малейшего представления о существовании над ними абсолютно позиционированного элемента. Это немного отличается от ситуации с плавающим элементом, потому что в том случае элементы общего потока заставляли все строчное содержимое учитывать границы плавающего элемента.

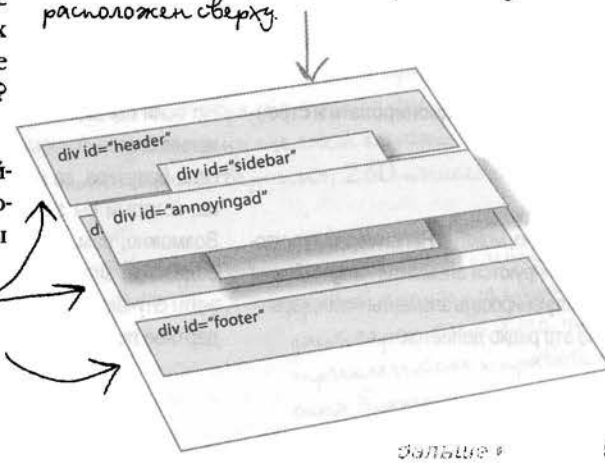
Абсолютно позиционированные элементы не оказывают никакого действия на другие элементы.

Кто сверху?

Еще один интересный момент, касающийся абсолютно позиционированных элементов: вы можете накладывать их друг на друга слоями. Но если у вас есть несколько абсолютно позиционированных элементов, расположенных в одном и том же месте страницы, как вам узнать порядок следования слоев? Другими словами, кто будет сверху?

У каждого позиционированного элемента есть свойство **z-index**, которое определяет его месторасположение на странице. Через несколько страниц вы увидите, как задается это свойство.

Элементы с идентификаторами sidebar и annoyingad накладываются друг на друга на странице, при этом раздел annoyingad имеет больший z-index, чем sidebar, поэтому он расположен сверху.



Все элементы с идентификаторами header, main и footer заливаются на страницу в общем потоке.

Часто Задаваемые Вопросы

В: Какое значение свойства position используется по умолчанию?

О: Значение static. При статическом позиционировании элементы заливаются на страницу в общем потоке и вы не определяете их месторасположение на странице; браузер сам решает, где они отобразятся. Чтобы убрать элемент из общего потока, вы можете использовать свойство float. При этом можно определить, будет элемент «плавать» слева или справа, но в конечном счете браузер сам решает, где именно располагать этот элемент. Сравните это со значением absolute свойства position. При абсолютном позиционировании вы говорите браузеру точное месторасположение элемента.

В: Я могу позиционировать только элементы ?

О: Вы можете абсолютно позиционировать любые элементы: и блочные, и строчные. Просто помните, что, когда элемент позиционирован абсолютно, он удаляется из общего потока.

В: Итак, я могу позиционировать и строчные элементы?

О: Да, конечно, можете. Например, очень часто позиционируются элементы <img>. Можно также позиционировать элементы , и др., но это редко делается.

В: Существуют ли другие значения свойства position, кроме static и absolute?

О: На самом деле их четыре: static, absolute, fixed и relative. Вы уже слышали о значениях static и absolute. При фиксированном позиционировании элемент располагается в месте, которое вычисляется относительно окна браузера (а не самой страницы), так что фиксированные элементы никогда не перемещаются. Через несколько страниц вы увидите пример такого позиционирования. При относительном позиционировании элемент заливается на страницу как обычно, но перед тем, как отобразиться на странице, он смещается в каком-либо направлении. Относительное позиционирование часто применяется для более современного позиционирования элементов и создания дополнительных эффектов. Пример такого позиционирования вы также еще увидите.

В: Нужно ли мне задавать ширину для абсолютно позиционированных элементов, как я делал это для плавающих элементов?

О: Ширину для абсолютно позиционированных элементов задавать не обязательно. Но если вы этого не сделаете, то по умолчанию блочный элемент займет всю ширину окна браузера, за исключением того отступа, который вы задали слева или справа. Возможно, вам именно это и будет нужно. Задавайте значение свойства width только в том случае, если хотите изменить это стандартное поведение.

В: При позиционировании элементов можно использовать только пикселы?

О: Нет, положение элементов часто задается через проценты. Если вы будете использовать проценты, то месторасположение ваших элементов будет меняться при изменении размера окна браузера. Итак, например, если ширина окна браузера равняется 800 пикселам, а для элемента левая позиция задана как 10 %, то он будет расположен на расстоянии 80 пикселов от левой границы страницы. Но если вы измените ширину окна браузера на 400 пикселов, то это расстояние до левой границы страницы будет уменьшено до 10 % от 400 пикселов, то есть до 40 пикселов.

Кроме того, в процентах часто задается ширина. Если вам не нужна четко установленная ширина для элементов или их полей, можете использовать проценты, чтобы и область основного содержимого, и врезка имели гибкие размеры. Так делается во многих двух- и трехколоночных разметках.

В: Нужно ли мне знать, как задавать свойство z-index, чтобы использовать абсолютное позиционирование?

О: Нет, свойство z-index применяется в более сложных ситуациях, где необходимо использование CSS. Это касается очень сложных сценариев веб-страниц и немного выходит за рамки материала этой книги. Но это свойство — часть работы абсолютного позиционирования, так что неплохо бы знать о нем (и вы действительно увидите случаи, в которых знание z-index необходимо).

Использование абсолютного позиционирования

Сейчас мы будем создавать двухколоночную версию страницы Starbuzz с помощью технических приемов, похожих на те, что мы применяли в версии страницы с плавающим элементом. Однако на этот раз мы будем использовать абсолютное позиционирование. Вот что нам нужно будет сделать.

1. Сначала мы сделаем элемент ** ** с идентификатором **sidebar** абсолютно позиционированным. При этом мы поместим его в том же самом месте, где он был расположен, будучи плавающим.
2. Затем мы зададим для области содержимого большое поле, чтобы врезка могла располагаться поверх пространства, занимаемого этим полем.
3. И наконец, мы хорошенько протестируем все это и сравним с версией страницы, на которой был плавающий элемент.

Изменение CSS для Starbuzz-страницы

Помните, мы вернулись к исходной версии файла, которую вы можете найти в папке chapter12/absolute

Вы можете работать и не в папке absolute, например скопировать файлы index.html и starbuzz.css в папку starbuzz и работать в ней, как это сделали мы.

Наш XHTML-код уже готов к использованию, а ** ** с идентификатором **sidebar** находится как раз там, где нужно (под областью основного содержимого). Нам остается внести несколько изменений в CSS, и мы получим абсолютно позиционированную врезку. Откройте файл `starbuzz.css` и внесите несколько правок для врезки.

```
#sidebar {
    background: #efe5d0 url(images/background.gif) bottom right;
    font-size:  105%;
    padding:    15px;
    margin:     0px 10px 10px 10px;
    position:   absolute;
    top:        128px;
    right:      0px;
    width:      280px;
}
```

Итак, теперь мы указываем, что врезка позиционирована абсолютно и должна находиться на расстоянии 128 пикселов от верхней границы страницы и 0 пикселов от правой. Мы также хотим, чтобы врезка имела фиксированную ширину, поэтому сделаем ее такой же, как в «плавающей версии»: 280 пикселов.

Откуда берется цифра 128, вы узнаете через минутку...

Расстояние 0 пикселов от правой границы страницы даст гарантию того, что врезка «приклеится» к правой стороне окна браузера.

Теперь нужно подкорректировать с идентификатором main

На самом деле нам не так уж много придется исправлять. Мы просто добавим поле, как делали это в версии с плавающим элементом. Итак, измените размер правого поля для элемента ** ** с идентификатором **main** и сделайте его равным 330 пикселам.

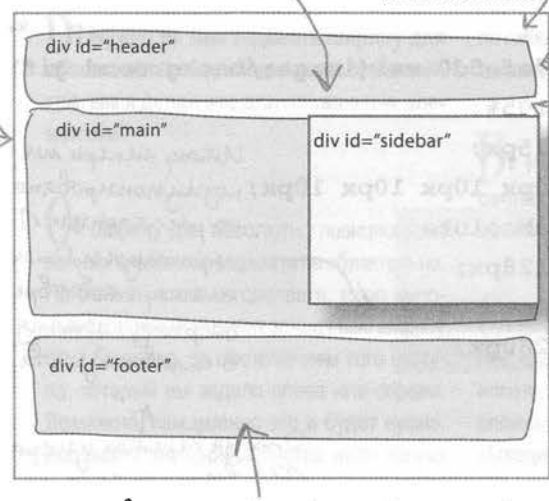
```
#main {
    background: #efe5d0 url(images/background.gif) top left;
    font-size:  105%;
    padding:    15px;
    margin:     0px 330px 10px 10px;
}
```

Мы определяем свободное место, на котором будет расположена врезка, задавая элементу с идентификатором main большое поле. Это точно такой же технический прием, который мы использовали для плавающего элемента. Единственное отличие заключается в способе расположения элемента с идентификатором sidebar.

Итак, вам нужно изменить размер поля, а затем сохранить эти изменения. Однако перед тем, как тестировать страницу, представьте, как она будет выглядеть с абсолютно позиционированной врезкой.

Мы располагаем врезку на расстоянии 128 пикселов от верха страницы и вплотную к правой ее стороне. Не забывайте, что врезка тоже имеет правое поле шириной 10 пикселов, так что фоновый цвет будет просвечиваться сквозь него, как и прежде.

Элемент с идентификатором main заливается на страницу сразу после верхнего колонтитула, так что его верхняя граница будет располагаться на том же уровне, что и верхняя граница врезки. У него также имеется правое поле такого же размера, как и у врезки, поэтому все его строчное содержимое не будет вплотную подходить к врезке слева. Помните, что элементы общего потока вообще не знают о существовании абсолютно позиционированных элементов, поэтому строчное содержимое элементов общего потока не будет обтекать абсолютно позиционированный элемент.



Возможно, вы думаете о том, что произойдет с нижним колонтитулом. Поскольку элементы общего потока ничего не знают об абсолютно позиционированных элементах, мы больше не можем использовать свойство clear.

Протестируем страницу с абсолютным позиционированием

Убедитесь, что вы сохранили изменения в CSS, и обновите страницу `index.html` в браузере. Посмотрим на результаты.

Ого, все выглядит точно так же, как когда мы использовали плавающий элемент, однако вы знаете, что врезка позиционирована абсолютно.



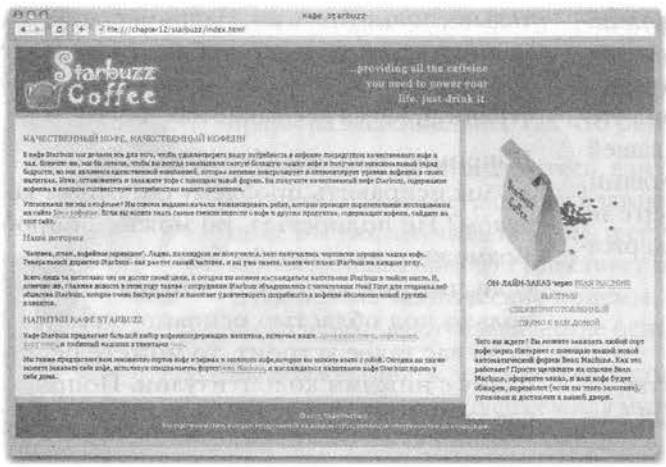
Когда вы изменяете размер окна браузера, врезка все равно находится на расстоянии 128 пикселов от верха страницы и вплотную к ее правому краю.

Кроме того, врезка имеет правое поле шириной 10 пикселов, поэтому между ней и краем страницы есть небольшое расстояние.

Область основного содержимого имеет правое поле, по ширине полностью совпадающее с шириной врезки. В результате врезка отображается именно в свободном пространстве, задаваемом этим полем.

Между двумя колонками все еще есть межстолбцовый промежуток.

Однако у нас снова появилась проблема с нижним колонтитулом. Если ширина окна браузера достаточно велика, то абсолютно позиционированная врезка перекрывает часть нижнего колонтитула. К сожалению, мы не можем обратиться к свойству **`clear`** на этот раз, так как элементы общего потока игнорируют присутствие на странице абсолютно позиционированных элементов.



Если окно браузера достаточно широкое, то высота области основного содержимого уменьшается и врезка перекрывает часть нижнего колонтитула.

Что мы можем сделать, или Не могли бы вы рассказать, как создать двухколоночную разметку, которая не будет постоянно портиться?

Вы знаете, что одна из главных причин появления CSS — разделение структуры и стиля. Верно? Кроме того, CSS выполняет большую работу, чтобы вы имели возможность создавать XHTML-документы, которые могут использоваться в множестве браузеров (даже в экранных дикторах и браузерах, отображающих только текст), не вставляя при этом ненужные стили в XHTML. Но это не означает, что CSS является полнофункциональным языком для разметки страницы. Скорее он дает некоторые интересные инструменты, которые можно использовать, чтобы систематизировать и позиционировать элементы XHTML-документов. В зависимости от того, где отображается страница, вы можете получить различные результаты. Кроме того, если вы расширите окно браузера, разметка может испортиться.

Итак, на чем мы остановились? В этой главе мы рассмотрели несколько способов создания двухколоночной страницы. Ни один из них не совершенен, и во всех приходится чем-то жертвовать. Давайте еще раз быстро просмотрим примеры различных разметок.

Плавающая разметка

Ах, какая прелесть, помните свой первый вариант двухколоночной Starbuzz-страницы? Вы использовали свойство **`float`**, а также свойство **`clear`** для нижнего колонтитула, и жизнь была прекрасна. Единственная проблема заключалась в том, что часто порядок расположения разделов на вашей странице плохо воспринимался пользователями, работающими в особых браузерах, например с экранными дикторами, которые читают содержимое страницы вслух.

Гибкая разметка

Сначала мы создали фиксированную разметку, заключив все содержимое страницы в новый элемент **`<div>`** фиксированного размера, а затем сделали ее гибкой, позволив полям растягиваться благодаря значению **`auto`**. В результате наша разметка стала выглядеть великолепно (в множестве веб-страниц используется именно такой дизайн). Это также решило проблему с упорядочением разделов. Недостаток данной разметки состоит в том, что содержимое страницы не растягивалось, чтобы занять всю ширину окна браузера (хотя многие разработчики вообще не рассматривают это как недостаток).

Разметка с абсолютным позиционированием

Наконец, мы поставили цель оставить разметку непостоянной, но чтобы содержимое располагалось в нужном порядке. Итак, мы использовали абсолютное позиционирование и на самом деле достигли своей цели. Однако в этом варианте тоже был свой недостаток: поскольку мы не могли использовать свойство **`clear`** с абсолютными элементами, нижний колонтитул перекрывался врезкой, если ширина окна браузера становилась достаточно большой.

Мы сделали все, что могли? Наверное, да. Если один из этих дизайнов устраивает вас, то замечательно, пользуйтесь им. Например, многих людей полностью устраивают гибкие разметки. Но вы всегда можете усовершенствовать собственную страницу, внеся кое-какие изменения.

Например, возьмем абсолютный дизайн. Можем ли мы исправить проблему с нижним колонтитулом? Не полностью, но можно попробовать. Возможно, ваш дизайн будет выглядеть лучше, если нижний колонтитул будет отображаться только под областью основного содержимого. Если вас такое устроит, то можете решить проблему с нижним колонтитулом. Попробуем это сделать.

Один компромисс, на который вы можете пойти, чтобы решить проблему с нижним колонтитулом

Чтобы проверить это решение на практике, просто задайте для нижнего колонтитула правое поле такого же размера, как и у области основного содержимого:

```
#footer {
        background-color: #675c47;
        color:            #efe5d0;
        text-align:       center;
        padding:          15px;
        margin:           10px 330px 10px 10px;
        font-size         90%;
}
```



Это решение приводит к тому, что элемент с идентификатором footer отображается только под областью основного содержимого. Это позволяет предотвратить наложение на него врезки.

Если вы сохраните эти изменения и обновите страницу, то увидите, что теперь нижний колонтитул отображается только под областью основного содержимого и никогда не перекрывается врезкой. Это наилучшее решение? Нет, но тоже неплохо. И как мы уже сказали, создание CSS-разметки — это своего рода искусство. Чтобы создать хорошую разметку, необходимо *экспериментировать*, *анализировать* и следить за CSS-разметками, которые создаются другими людьми (в конце главы вы найдете несколько ссылок на хорошие форумы по CSS).

Итак, все это, конечно, замечательно, но что я все-таки должна делать? Использовать гибкую разметку или разметку с непостоянной шириной? Применять относительное или абсолютное позиционирование?

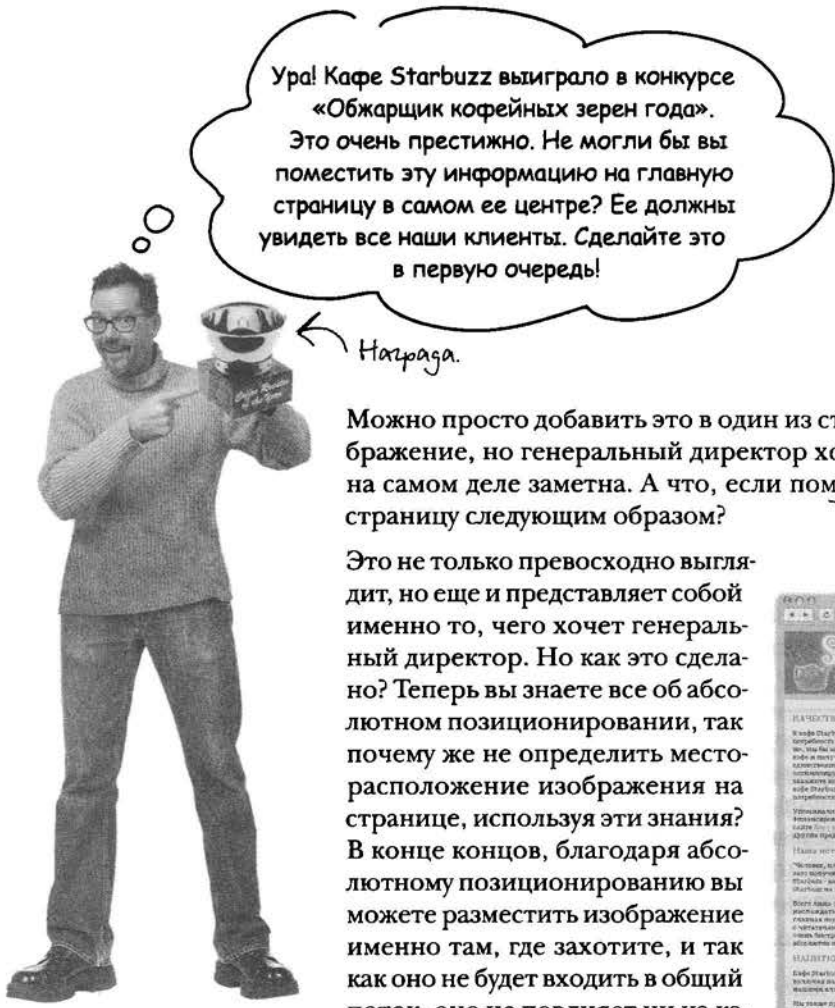


Решение, какую разметку использовать, зависит от того, какая из них будет лучше подходить для вашей страницы. Для некоторых страниц замечательно подходит фиксированный размер области содержимого и поля с непостоянной шириной, и такая страница может лучше выглядеть в широком окне браузера. Для других страниц вы, возможно, не захотите использовать ширину окна браузера максимально. Так что решайте, что лучше именно для вас.

Когда вы с этим определитесь, вам останется выяснить, какой метод использовать для создания страницы: плавающий, абсолютный или какую-то их комбинацию. Вы уже выучили основы, и сейчас пришло время для экспериментов, так как на данный момент существует множество подходов и каждый день появляются новые. Технические приемы, с которыми вы ознакомились в этой главе, часто используются в качестве основ для очень замысловатых дизайнов.

Вы должны знать, что в целом использование плавающих элементов считается наиболее гибким решением для мультиколоночных разметок. Просто помните, что нужно быть внимательными к последовательности расположения содержимого страницы и учитывать то, какой дизайн применяется.

дальше ▸



Можно просто добавить это в один из старых абзацев на странице как изображение, но генеральный директор хочет, чтобы эта информация была на самом деле заметна. А что, если поместить информацию о награде на страницу следующим образом?



Это не только превосходно выглядит, но еще и представляет собой именно то, чего хочет генеральный директор. Но как это сделано? Теперь вы знаете все об абсолютном позиционировании, так почему же не определить месторасположение изображения на странице, используя эти знания? В конце концов, благодаря абсолютному позиционированию вы можете разместить изображение именно там, где захотите, и так как оно не будет входить в общий поток, оно не повлияет ни на какие элементы на странице. Кажется, это небольшое изменение позволит легко вставить нужное изображение и не нарушит имеющуюся структуру страницы.

Давайте попробуем это сделать. Начнем с добавления нового элемента ** ** сразу под верхним колонтитулом (генеральный директор считает, что очень важно, чтобы изображение находилось в самом начале).

```
<div id="award">
    <img src="images/award.gif"
    alt="Roaster of the Year award" />
</div>
```

Вот этот содержит изображение награды

Позиционирование награды

Мы хотим, чтобы изображение награды было расположено примерно посередине страницы, когда ширина окна браузера составляет 800 пикселов (это ширина изображения верхнего колонтитула и обычная ширина окна браузера), и просто перекрывало ** ** с основным содержимым.

Итак, будем использовать свойства **top** и **left** для позиционирования награды: 30 пикселов от верхней границы страницы и 365 пикселов от левой.

```
#award {
    position: absolute;
    top:      30px;
    left:     365px;
}
```

Мы задаем абсолютную позицию для элемента с идентификатором award, которая составляет 30 пикселов от верхней границы страницы и 365 пикселов от левой.

Добавьте этот CSS-код в файл `starbuzz.css`, сохраните изменения и обновите веб-страницу. Вы увидите, что изображение появилось именно там, где мы хотели. Обязательно поэкспериментируйте с размером окна браузера, чтобы посмотреть, где отображается рисунок.

Небольшой дефект

Вы заметили небольшой дефект, когда меняли размер окна браузера? Возможно, вы увидели, что ** ** с идентификатором **sidebar** перекрывает изображение (это зависит от вашего браузера). Что происходит? Помните, что каждый абсолютно позиционированный элемент имеет свойство **z-index**, которое описывает, будут ли одни элементы накладываться на другие? В некоторых браузерах по умолчанию элементу с изображением награды будет присвоено значение свойства **z-index** меньше, чем у элемента ** ** с идентификатором **sidebar**. Чтобы исправить это, вам придется задать значение **z-index** для награды больше, чем для врезки.

В некоторых браузерах врезка будет накладываться на изображение награды и частично перекрывать ее.

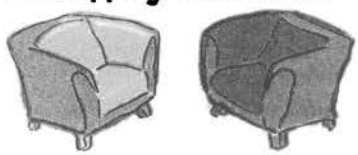



```
#award {
    position: absolute;
    top:      30px;
    left:     365px;
    z-index:  99;
}
```

Зададим для элемента с идентификатором award действительно большое значение свойства z-index, чтобы быть уверенными, что оно больше значения z-index для врезки.

Итак, двигайтесь дальше! Добавьте свойство **z-index** в файл `starbuzz.css`. Сохранив изменения и обновив страницу, вы увидите, что теперь ** ** с идентификатором **award** отображается поверх врезки, а дефект с наложением исправлен.

Беседа у камина



Вечерний диалог: **Плавающий и Абсолютный рассуждают на тему, кто из них лучше подходит для создания разметок.**

Плавающий

Абсолютный, заметил ли ты, что многие люди прибегают к моей помощи при создании разметок?

Все знают, что я лучше подхожу для CSS-разметки. Работать со мной намного проще. Разве ты этого не заметил? Нужно лишь добавить одно маленькое свойство **`float`** в нужное место CSS-кода, и бах! Ты получаешь две колонки.

Это мелочи. Я хочу сказать, что, используя меня, люди не возятся с вычислением расстояний в пикселах. Чтобы выяснить, где будет расположен определенный блок содержимого, можно просто сделать его плавающим, а в остальном положиться на меня.

А как насчет проблемы с нижним колонтитулом? Ты всегда накладываешь блоки друг на друга, не так ли? Если читатели недостаточно аккуратны, то у них может получиться так, что одни блоки веб-страниц перекрывают большие участки других блоков. Я, по крайней мере, учитываю удобное свойство **`clear`**.

Абсолютный

Ты в этом уверен? Знаешь, я тоже используюсь в множестве страниц.

Хмм, помнится, еще нужно использовать свойства **`width`** и **`margin`**, чтобы все заработало правильно...

Минуточку. Приходится перемещать *весь* блок врезки в другое место XHTML-кода, чтобы все сработало как надо. Я бы не назвал это «в остальном положиться на тебя». Это большой кусок работы. Используя меня, по крайней мере не нужно беспокоиться о том, в каком порядке расположено содержимое. Я сделаю все правильно независимо от этого.

Эй, ты тоже накладываешься на другие элементы.

Плавающий

Но не забывай, что строчное содержимое обтекает меня, как это и должно быть.

Ты кое-что забываешь. Я более гибкий и даю людям отличный способ создавать разметку их страниц. Я уверен, что могу сделать любую разметку, какую способен сделать ты.

Хммм...

Ладно, может быть, я и не могу сделать *всего* того, что можешь ты, но я думаю, что людям намного проще делать стандартные разметки, используя меня, и это устраивает большинство из них.

Да, раньше я имел такую славу, но более современные версии браузеров достаточно хорошо ладят со мной. И теперь, когда веб-разработчики это осознают, они предпочитают работать со мной, как я уже говорил в самом начале.

А я никогда и не говорил, что ты редко применяешься. Но посмотри на те дизайны, в которых используюсь я.

Ладно, но ведь это не так уж и важно, не так ли? Так или иначе, у меня появилась работа, мне нужно бежать.

Абсолютный

Знаешь, иногда люди *хотят*, чтобы элементы располагались прямо поверх других элементов. С моей помощью можно позиционировать элементы где угодно, а не только слева или справа, как предлагаешь ты. Ты, наверное, не думал об этом, но на самом деле ты не предоставляешь больших возможностей.

Правда? Я думаю, у тебя нет способа, с помощью которого ты смог бы реализовать эту замечательную штуку с наградой.

Согласись с этим! На самом деле ты не такой гибкий, как я.

Не знаю, я слышал, что в старых браузерах у тебя появляется куча ошибок. Это может расстроить начинающих веб-разработчиков.

Я не думаю, что ты хорошо осведомлен о том, где я применяюсь. Меня очень часто используют на веб-страницах.

Эй, не думай, что ты такой безупречный. Возможно, ты и неплох для веб-разметок, но точно не подходишь для графического дизайна.

Подумай над этим, Плавающий.

Еще кое-что, что вы должны знать об абсолютном позиционировании

До сих пор, когда вы использовали свойства **left**, **right**, **top** и **bottom** для определения положения элемента, все эти расстояния вычислялись относительно границ страницы, не так ли? Теперь нам придется немного углубиться в детали.

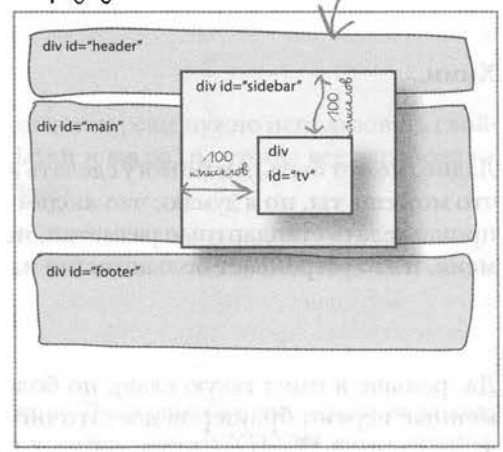
Когда вы позиционируете элемент, вы задаете его месторасположение относительно ближайшего элемента-родителя, который тоже позиционирован. Сейчас вы, скорее всего, говорите: «Что? Я не позиционировал ничего, кроме врезки. Как же тогда в моем XHTML-коде может появиться позиционированный элемент-родитель?» Хотите — верьте, хотите — нет, но существует элемент **<html>**, который браузер сам позиционирует при создании страницы.

Однако пока пропустим этот момент. Допустим, вы хотите абсолютно позиционировать другой элемент ** ** внутри врезки:

Элемент с идентификатором tv позиционируется относительно элемента с идентификатором sidebar, а не относительно страницы

```
<div id="sidebar">

    <div id="tv">
        <img src="AsSeenOnTv.gif" alt="Что смотреть по ТВ" />
    </div>

    <p class="beanheading">
        ... остальная часть XHTML-кода здесь ...
    </p>
    ...
</div>
```

Это новый , вложенный во врезку.



Если мы позиционируем элемент ** ** с идентификатором **tv**, то его ближайшим позиционированным элементом-родителем будет ** ** с идентификатором **sidebar**. В результате позиционирование будет выполняться относительно границ врезки, а не страницы.

```
#tv {
    position: absolute;
    top:   100px;
    left:  100px;
    width: 100px;
}
```

Вам полезно будет знать следующее: если вы примете участие в беседе о ближайших позиционированных родителях на своей следующей вечеринке с коктейлями, просто говорите «ближайший блок, который уже позиционирован». Такую терминологию используют профессионалы.

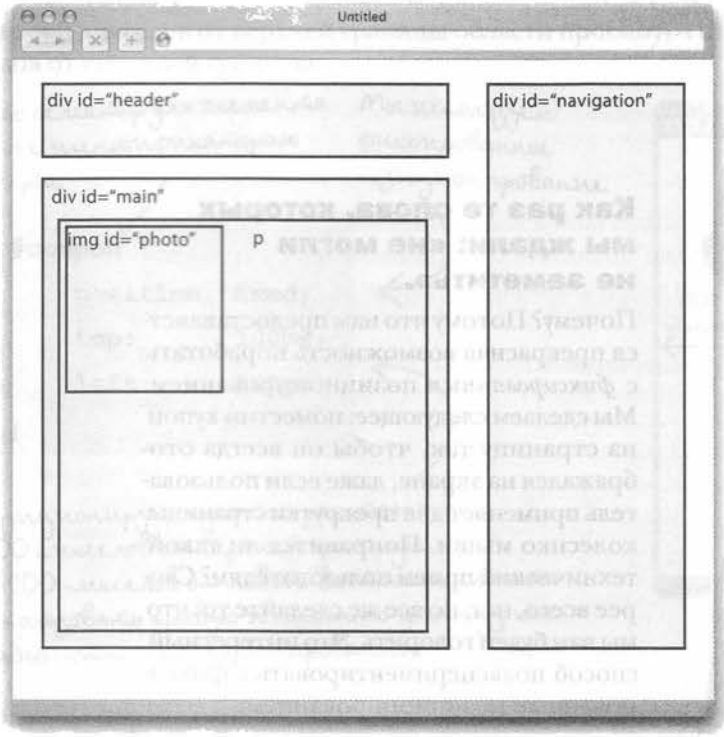


Если вы позиционируете элемент относительно элемента <html>, то, возможно, свойство bottom применится не так, как вы ожидаете. Может быть, вы думаете, что bottom — это нижняя граница самой веб-страницы, но элемент <html> определяет это как нижнюю границу окна браузера. Итак, если вы хотите абсолютно позиционировать элемент относительно нижней границы самой страницы, а не окна браузера, то вам нужно поместить элемент внутрь другого элемента, который растягивается до самого низа страницы и является позиционированным. Единственный способ так сделать — вложить свой элемент в относительно позиционированный элемент в самом низу страницы. С относительным позиционированием мы разберемся чуть позже в этой главе.

Возьми в руку карандаш

Пришло время проверить все ваши знания о плавающих и абсолютно позиционированных элементах на практике!

Взгляните на веб-страницу, приведенную ниже. На ней есть четыре элемента, каждый со своим идентификационным именем. Ваша задача — правильно поставить в соответствие каждому элементу CSS-правило (см. справа) и вписать идентификатор для каждого селектора. Проверьте свои ответы в конце главы.



Впишите имена селекторов, чтобы дополнить CSS.

```
_____________ {
   margin-top:  140px;
   margin-left: 20px;
   width:       500px;
}

_____________ {
   position: absolute;
   top:      20px;
   left:     550px;
   width:    200px;
}

_____________ {
   float: left;
}

_____________ {
   position: absolute;
   top:      20px;
   left:     20px;
   width:    500px;
   height:   100px;
}
```



Как раз те слова, которых мы ждали: «не могли не заметить».

Почему? Потому что нам предоставляется прекрасная возможность поработать с *фиксированным* позиционированием. Мы сделаем следующее: поместим купон на страницу так, чтобы он всегда отображался на экране, даже если пользователь применяет для прокрутки страницы колесико мыши. Понравится ли такой технический прием пользователям? Скорее всего, нет, но все же сделайте то, что мы вам будем говорить. Это интересный способ поэкспериментировать с фиксированным позиционированием.



Как работает фиксированное позиционирование

По сравнению с абсолютным, фиксированное позиционирование достаточно простое. Здесь вы задаете месторасположение элемента точно так же, как и в абсолютном позиционировании, но положение вычисляется относительно границ *окна браузера*, а не относительно *страницы*. Это имеет один интересный эффект: если вы определяете месторасположение элемента, используя фиксированное позиционирование, то он всегда остается там, куда вы его поместили, и не перемещается даже тогда, когда вы используете для прокрутки страницы колесико мыши.

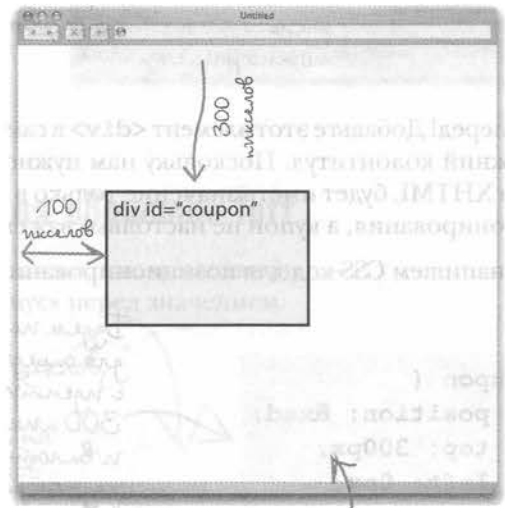
Впечатлите своих друзей и коллег по работе, сославшись на окно браузера как на область просмотра. Попробуйте сделать это, и W3C одобрительно кивнет вам головой.

Допустим, у вас есть элемент ** ** с идентификатором **coupon**. Вы можете фиксированно позиционировать этот ** ** на расстоянии 300 пикселов от верхней границы области просмотра и 100 пикселов от его левой границы.

Это селектор для элемента с идентификатором coupon.

Мы используем фиксированное позиционирование.

```
#coupon {
    position: fixed;
    top:      300px;
    left:     100px;
}
```



Позиционируем купон на расстоянии 300 пикселов от верхней границы и 100 пикселов от левой. Вы также можете использовать правую и нижнюю границу, как в абсолютном позиционировании.

На этом месте внутри области просмотра будет расположен элемент.

Когда вы определите месторасположение элемента, начнется самое интересное: при использовании для прокрутки страницы колесика мыши элемент не двигается. Поменяйте размер окна браузера — элемент не двигается. Поднимите свой монитор и потрясите его — элемент не двигается. Ладно, последнее — это шутка. Но суть в том, что фиксированно позиционированные элементы не двигаются; они располагаются на одном и том же месте, пока страница открыта.

Мы уверены, что сейчас вы думаете о том, сколько всего забавного можно сделать с помощью фиксированного позиционирования, но у вас есть задание, которое нужно выполнить. Итак, поместим этот купон на страницу Starbuzz.



К сожалению, 6-я и более ранние версии Internet Explorer не поддерживают фиксированное позиционирование. Если вы используете старые версии этого браузера, то не сможете увидеть правильное расположение купона на веб-странице кафе Starbuzz.

Размещение купона на странице

Теперь нам нужно поместить на страницу купон «Бесплатный кофе». Начнем с создания элемента ** **, который будет включать в себя этот купон:

Это элемент с идентификатором coupon.

Внутри будет изображение купона, которое вы найдете в папке chapter12/starbuzz/images.

```
<div id="coupon">
  <a href="freecoffee.html" title="Щелкните здесь, чтобы получить бесплатный кофе">
    <img src="images/ticket.gif" alt="Купон на бесплатный кофе Starbuzz" />
  </a>
</div>
```

Мы вложили изображение в элемент <a>, чтобы пользователи могли щелкнуть на этом изображении и попасть на страницу с купоном, который можно распечатать.

Итак, вперед! Добавьте этот элемент ** ** в самый конец вашего файла `index.html` сразу под нижний колонтитул. Поскольку нам нужно позиционировать элемент, его расположение в XHTML будет иметь значение только в тех браузерах, которые не поддерживают позиционирования, а купон не настолько важен, чтобы помещать его в самом верху.

Теперь напишем CSS-код для позиционирования купона.

Будем использовать фиксированное позиционирование для определения месторасположения элемента с идентификатором coupon и расположим его на расстоянии 300 пикселов от верхней границы окна просмотра и вплотную к его правой границе. Таким образом, нужно задать расстояние 0 пикселов от левой границы.

```
#coupon {
    position: fixed;
    top: 300px;
    left: 0px;
}
```

Кроме того, нужно стилизовать изображение и ссылки. В противном случае может оказаться, что вокруг изображения появилась граница, так как оно активизируется щелчком кнопкой мыши. Итак, установим значение none свойства border для изображения и сделаем то же самое для посещенных и непосещенных ссылок.

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

#coupon a:link {
    border: none;
}
#coupon a:visited {
    border: none;
}
```

Помните, что в CSS есть правило, которое убирает эффекты декорирования текста и вместо этого использует границу для подчеркивания текста. В данном примере мы переопределяем это правило для ссылок в элементе с идентификатором coupon и говорим, что не хотим, чтобы для ссылок использовалась граница. Вернитесь назад и посмотрите на исходный CSS-код, если забыли, как выглядят остальные правила для ссылок.

Размещение купона на странице

Добавьте новые правила для купона в файл `starbuzz.css`, сохраните изменения и обновите страницу. Возможно, вам придется уменьшить размеры окна браузера, чтобы увидеть, что купон остается на своем месте, даже когда для просмотра страницы применяется колесико мыши. Щелкнув кнопкой мыши на купоне, вы попадете на страницу `freecoffee.html`.



Знаете, выглядит замечательно, но станет еще более привлекательным, если сместить купон влево. В результате будет казаться, будто он выглядывает из окна просмотра. Для создания такого эффекта можно воспользоваться программой для редактирования изображений и обрезать левый край купона. Кроме того, можно использовать отрицательные значения смещения, чтобы левая сторона изображения располагалась за пределами порта просмотра.

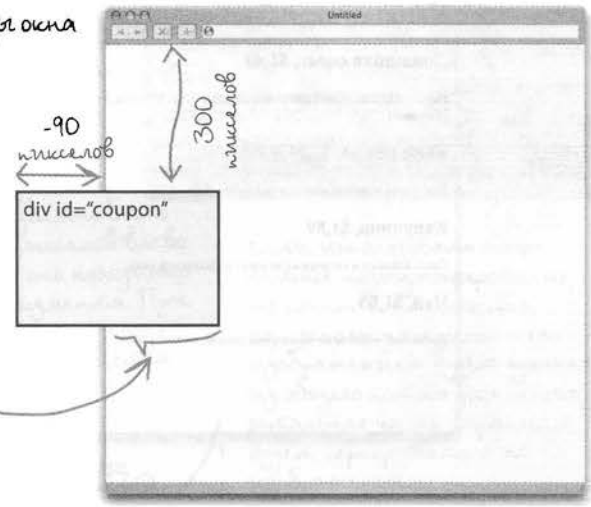
Все правильно, *вы можете сделать это.*

Использование отрицательного значения свойства left

Задайте отрицательное значение для свойства точно так же, как вы задавали положительные значения, только поставьте знак «минус» перед значением.

```
#coupon {
    position: fixed;
    top: 300px;
    left: -90px;
}
```

Задав значение -90 пикселов, мы говорим браузеру расположить элемент на расстоянии 90 пикселов слева от левой границы окна просмотра.



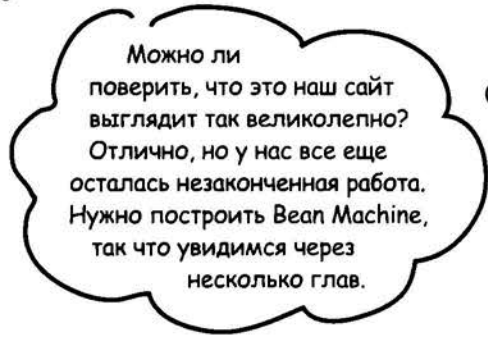
Браузер с удовольствием поместит изображение левее левой границы окна, и будет видна только часть изображения, которая попадает в окно просмотра.

Положительный тест для отрицательных значений

Убедитесь, что задали отрицательное значение для свойства **left**, сохраните изменения и обновите страницу. Разве это не симпатично? Поздравляем, вы только что создали первый спецэффект в CSS. Джордж Лукас, берегись!



Просто помните, что использование фиксированного позиционирования элемента, при котором под ним прячется часть содержимого страницы, вряд ли можно назвать удобным для пользователя, но это ВЕСЕЛО.









ОГО! Какая огромная разница!

Знакомство с относительным позиционированием

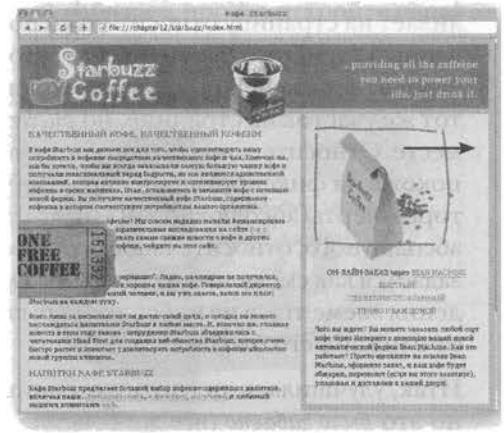
Пришло время рассмотреть *относительное позиционирование*. Надо сказать, что это самый одинокий вариант позиционирования, так как сегодня вы встретите немного людей, использующих его в дизайнах своих страниц. Однако новые дизайны появляются каждый день, поэтому, если вы увидите относительное позиционирование, то должны знать, как оно работает.

Элемент, позиционированный относительно, является частью общего потока страницы, но в последний момент, как раз перед тем, как отобразить этот элемент, браузер смещает его. Это отличает относительное позиционирование от абсолютного и фиксированного. Как это работает, мы рассмотрим на примере пакетика с кофе на странице Starbuzz. Сместим его в сторону, чтобы казалось, что те кофейные зерна, которые высыпаются из пакетика, высыпаются и за пределы страницы.

На текущий момент мы можем абсолютно позиционировать данный элемент, но при этом нужно будет найти способ заранее зарезервировать место на странице, которое он будет занимать, так как при абсолютном позиционировании элемент полностью удаляется из общего потока.

Именно здесь вступает в действие относительное позиционирование. Мы можем *оставить* элемент в общем потоке, что обеспечит ему место на странице, а затем сместить туда, где он действительно должен быть отображен. Давайте попробуем.

Мы хотим взять пакетик с кофе Starbuzz и сместить его примерно на 100 пикселов вправо.





Это новое правило, которое выбирает изображение. Здесь мы указываем селектор потомка, чтобы выбрать только те изображения, которые находятся внутри элемента с идентификатором beanheading.

```
.beanheading img {
    position: relative;
    left: 120px;
}
```

Обратите внимание, что изображение — строчный элемент, и это нормально. Вы можете использовать любые методы позиционирования строчных элементов и даже сделать их плавающими.

Итак, здесь мы указываем, что изображение должно быть смещено на 120 пикселов влево от того места, где оно находится в общем потоке документа. При задании величины смещения можно использовать свойства right, top и bottom.

Затем мы указываем, что позиционирование относительное, и задаем величины всех смещений для нашего изображения. Смещение вычисляется относительно того места, на котором элемент находится в общем потоке.

Если использовать абсолютное позиционирование, то пакетик сместится, но, поскольку элемент с его изображением будет удален из общего потока документа, остальная часть страницы будет перекрываться этим изображением.

Добавьте это правило в свой CSS-код, а затем сохраните изменения и обновите страницу.

дальше ▸

Тест



Обновив страницу Starbuzz, вы увидите пакетик с кофе в правой части врезки. Интересно, что изображение немного выходит *за пределы* врезки на ее поле и далее за пределы страницы. Как это работает? Как вы уже видели, браузер сначала заливает на страницу относительно позиционированный элемент, а потом смещает его туда, где он будет отображен. В результате элемент занимает тот же участок на странице, но *рисуется* в другом месте. Относительное позиционирование чем-то похоже на фиксированное, но содержит и некоторые черты абсолютного. Однако, в отличие от абсолютного, относительное позиционирование задается как смещение от реального местонахождения элемента, а не в абсолютных координатах от ближайшего блока.

Итак, улучшилась ли в итоге страница? Вряд ли, но это *было забавно* (все же советуем вам убрать относительное позиционирование перед тем, как показывать страницу генеральному директору).



Скорее всего, нет.

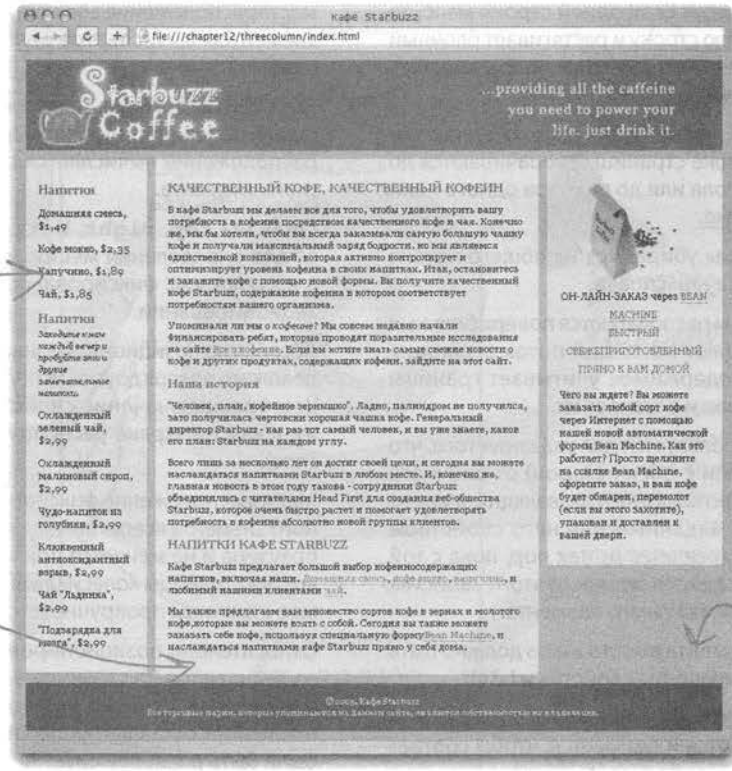
Независимо от того, как вы настраиваете поля и отступы, вы все равно не сможете сделать так, чтобы изображение выводилось вне блока, в котором оно расположено. И зачем вообще выбирать сложный путь для решения этой задачи?

Мы достигли наилучшего эффекта всего лишь с помощью двух строк CSS. Можно использовать относительное позиционирование, чтобы отобразить элемент вдали от содержащего его блока в потоке, чего нельзя сделать с помощью полей и отступов.

Три колонки и более...

Хотя на протяжении всей главы мы рассматривали двухколоночные разметки, основной целью было все же изучение свойств **float** и **clear**, а также знакомство с различными видами позиционирования, предлагаемыми CSS. Теперь, когда вы знаете основы, самое время подумать над трехколоночными (или любыми другими) разметками. Итак, глава подошла к концу.

Но подождите! Прежде чем окончательно завершить ее, давайте продумаем, как может работать трехколоночная разметка (если вы хотите на примере посмотреть, как она работает, просто откройте папку `chapter12/threecolumn`).



Дизайн разработан на основе технических приемов, которые вы уже понимаете. Чтобы узнать намного больше, чем мы здесь показали, вы можете посмотреть, как другие разработчики использовали CSS для создания интересных дизайнов. Мы рекомендуем вам не останавливаться на достигнутом и оглядеться вокруг. Посетите несколько наших любимых ресурсов о CSS-дизайне, перечисленных на:

`http://headfirstlabs.com/books/hfhtml/chapter12/cssdesign.html`

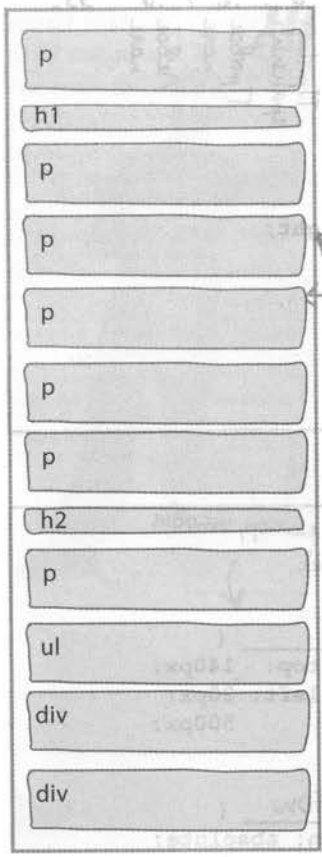
ПОВТОРИМ выученное



- Браузеры размещают элементы на странице, используя поток.
- Блочные элементы заливаются на страницу сверху вниз, с разрывами строки между ними. По умолчанию каждый блочный элемент занимает всю ширину окна браузера.
- Строчные элементы внутри блочных заливаются слева направо, сверху вниз. Если одной строки мало, то браузер создает новую строку и растягивает блочный элемент в вертикальном направлении, чтобы вместить все строчное содержимое.
- Смежные верхнее и нижнее поля двух соседних элементов в общем потоке страницы сворачиваются до размера большего поля или до размера одного поля, если поля одинаковые.
- Плавающие элементы убираются из общего потока и размещаются слева или справа.
- Плавающие элементы располагаются поверх блочных элементов и не влияют на общий поток страницы. Однако строчное содержимое учитывает границы плавающих элементов и обтекает их.
- Свойство **clear** используется для указания того, что слева или справа (или с обеих сторон) от блочного элемента не может располагаться плавающий элемент. Блочный элемент с заданным для него свойством **clear** будет смещаться вниз до тех пор, пока с той стороны, которая указана в значении этого свойства, не будет ни одного плавающего элемента.
- Для плавающего элемента вместо **auto** должно быть задано конкретное значение свойства **width**.
- В разметке с непостоянной шириной содержимое страницы растягивается и сжимается, чтобы соответствовать размерам окна браузера.
- В разметке с постоянной шириной область содержимого фиксирована, не растягивается и не сжимается при изменении размеров окна браузера. Преимущество этой разметки в том, что можно лучше контролировать дизайн, но ценой того, что ширина окна браузера будет задействована не полностью.
- В гибкой разметке ширина содержимого фиксирована, но при изменении размеров окна браузера растягиваются и сжимаются поля. В такой разметке содержимое обычно находится в центре страницы. Она имеет те же преимущества, что и разметка с постоянной шириной, но часто выглядит более привлекательно.
- Для свойства **position** может быть задано четыре значения: **static**, **absolute**, **fixed** и **relative**.
- Фиксированное позиционирование используется по умолчанию. В этом случае элементы заливаются на страницу в общем потоке.
- Абсолютное позиционирование позволяет поместить элемент на странице там, где нужно. По умолчанию месторасположение абсолютно позиционированных элементов вычисляется относительно границ страницы.
- Если абсолютно позиционированный элемент вложен в другой позиционированный элемент, то его месторасположение вычисляется относительно родительского элемента.
- Свойства **top**, **right**, **bottom** и **left** применяются для определения месторасположения элементов в абсолютном, фиксированном и относительном позиционировании.
- Абсолютно позиционированные элементы можно расположить поверх других, используя свойство **z-index**. Чем больше значение свойства **z-index**, тем ближе к вам на экране расположен элемент (в верхнем слое).
- Месторасположение фиксированно позиционированного элемента всегда вычисляется относительно окна браузера и не меняется при использовании для прокрутки страницы колесика мыши. Остальное содержимое страницы прокручивается под этим элементом.
- Относительно позиционированные элементы сначала заливаются на страницу в общем потоке, а затем смещаются на указанное расстояние в заданном направлении, оставляя пустым то место, где они должны были быть расположены.
- При использовании относительного позиционирования свойства **left**, **right**, **top** и **bottom** указывают смещение от месторасположения элемента, если он залит в общем потоке.
- Один и тот же дизайн часто может быть создан с использованием плавающих и абсолютно позиционированных элементов.
- Применение плавающих элементов — гибкое решение для создания многоколоночных дизайнов. Оно позволяет обеспечить отсутствие плавающих элементов со стороны других элементов, чего нельзя достичь при абсолютном позиционировании.

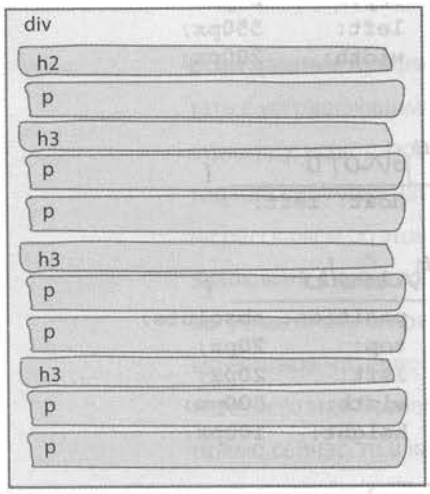


Решение упражнения



Все блочные элементы в файле lounge.html заливаются сверху вниз, и между ними добавляется разрыв строки.

У этих трех элементов есть вложенные в них блочные элементы.



Мы не просили вас подумать над этим, но если вы захотите пойти дальше, то посмотрите, как они зальются на страницу.



Поработайте браузером

Откройте файл lounge.html и найдите все блочные элементы. Залейте каждый из них на страницу слева. Сосредоточьтесь на блочных элементах, вложенных непосредственно в элемент body. Вы можете проигнорировать свойство float в своем CSS-коде, так как пока не знаете, что оно делает. Далее приведено решение.



Решение упражнения

Переместите элемент ** ** на его изначальное место — под списком музыкальных композиций, затем сохраните файл и обновите страницу. Где теперь «плавает» элемент? Вы должны были увидеть раздел с напитками под разделом с музыкальными композициями.

Элемент «плавает» справа, прямо под списком музыкальных композиций, а остальной XHTML-код (а это нижний колонтитул) обтекает его.



Возьми в руку карандаш

Решение

Нужно было задать для правого поля раздела с основным содержимым такую же ширину, как и у врезки. Но чему же она равняется? Мы надеемся, что вы еще не забыли то, что учили в прошлой главе. Приводим здесь информацию, которая понадобится для вычисления ширины врезки. А вот решение.

15 + 15 + 280 + 0 + 0 + 10 + 10 = 330

Левый отступ · Правый отступ · Область содержимого · Левая граница · Правая граница · Правое поле · Левое поле

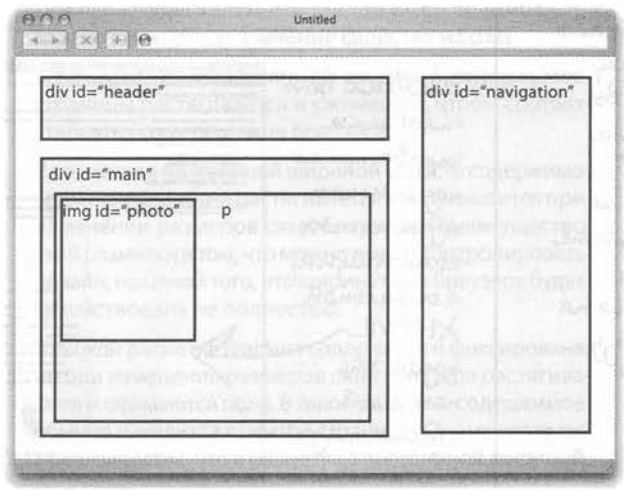
```
#sidebar {
        background: #efe5d0 url(images/background.gif) bottom right;
        font-size:  105%;
        padding:    15px;
        margin:     0px 10px 10px 10px;
        width:      280px;
        float:      right;
}
```

Возьми в руку карандаш

Решение

Пришло время проверить все ваши знания о плавающих и абсолютно позиционированных элементах на практике!

Взгляните на веб-страницу, приведенную ниже. На ней есть четыре элемента, каждый со своим идентификационным именем. Ваша задача — правильно поставить в соответствие каждому элементу CSS-правило (см. справа) и вписать идентификатор для каждого селектора. Далее приведено решение. Вы все сделали правильно?



Впишите селектор, чтобы дополнить CSS.

```
#main {
    margin-top:  140px;
    margin-left: 20px;
    width:       500px;
}

#navigation {
    position: absolute;
    top:      20px;
    left:     550px;
    width:    200px;
}

#photo {
    float: left;
}

#header {
    position: absolute;
    top:      20px;
    left:     20px;
    width:    500px;
    height:   100px;
}
```

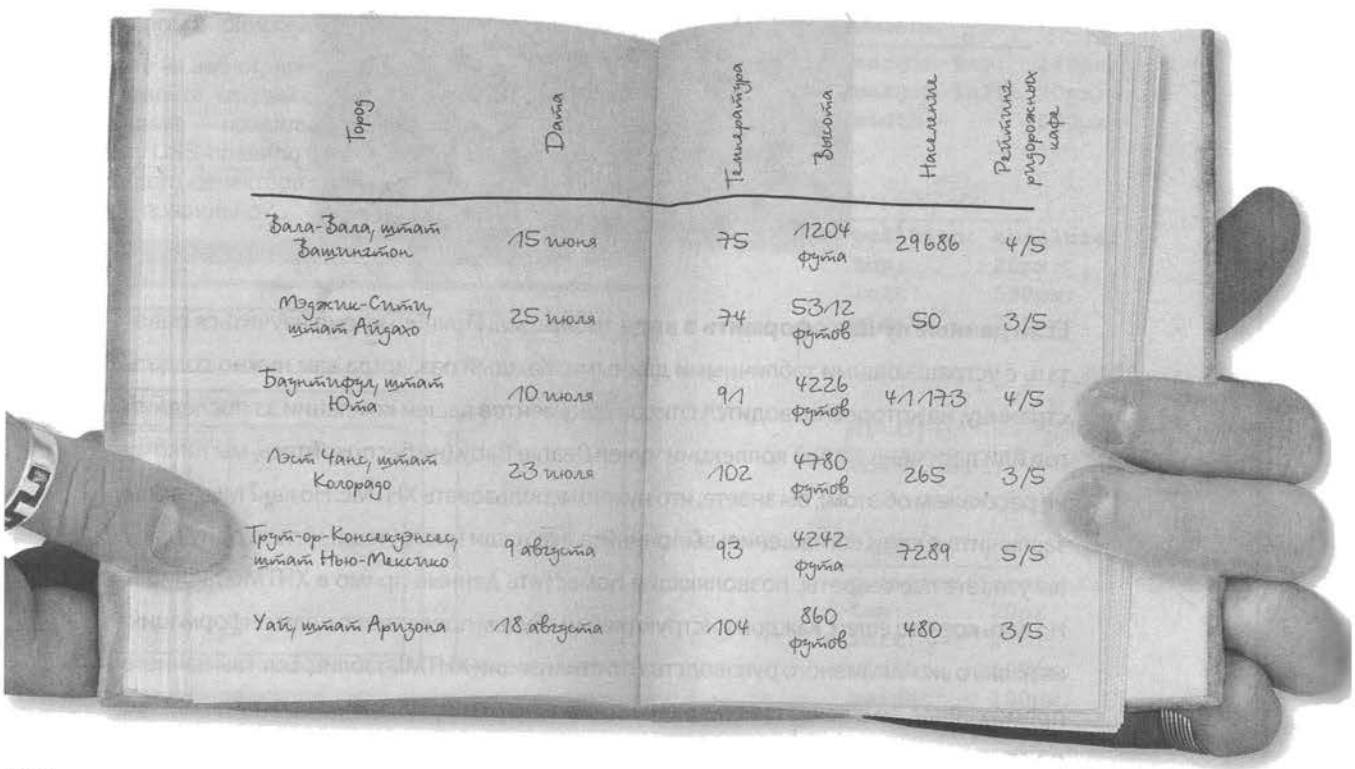
Представление в табличной форме



Если данные лучше оформить в виде таблицы... Пришло время научиться работать с устрашающими табличными данными. Каждый раз, когда вам нужно создать страницу, на которой выводится список документов вашей компании за последний год или перечень вашей коллекции чучел Beanie Baby (не беспокойтесь, мы никому не расскажем об этом), вы знаете, что нужно использовать XHTML. Но как? Мы готовы заключить с вами соглашение: выполняйте все наши инструкции — и за одну главу вы узнаете все секреты, позволяющие поместить данные прямо в XHTML-таблицы. Но есть кое-что еще: с каждой инструкцией мы будем представлять вам информацию из нашего эксклюзивного руководства по стилизации XHTML-таблиц. Если вы начнете прямо сейчас, то в качестве специального бонуса мы представим вам руководство по стилизации XHTML-списков. Не сомневайтесь, соглашайтесь прямо сейчас!

Эй, ребята, я только что создал в своем дневнике небольшую таблицу городов. Я собирался поместить ее на сайт, но не смог найти хорошего способа сделать это с помощью заголовков, блочных цитат или абзацев. Вы можете мне помочь?

О.о



Город	Дата	Температура	Высота	Население	Рейтинга придорожных кафе
Вала-Вала, штат Вашингтон	15 июня	75	1204 фута	29686	4/5
Мэджик-Сити, штат Айдахо	25 июля	74	5312 футов	50	3/5
Бьютифул, штат Юта	10 июля	91	4226 футов	41173	4/5
Лэйк Чанс, штат Колорадо	23 июля	102	4780 футов	265	3/5
Трут-ор-Консеквэнсес, штат Нью-Мексико	9 августа	93	4242 фута	7289	5/5
Уайт, штат Аризона	18 августа	104	860 футов	480	3/5

Как мы создаем таблицы в XHTML

Тони прав, вы на самом деле еще не видели хорошего способа использования XHTML для изображения таблиц. Возможно, вы думаете, что для создания таблиц можно использовать CSS и элементы ** **, но в XHTML есть элемент **<table>**, который заботится обо всех таблицах. Прежде чем погрузиться в изучение этого элемента, рассмотрим, из чего состоят таблицы.

У нас есть столбцы

В этой строке находятся заголовки.

Кроме того, у нас есть строки.

Город	Дата	Температура	Высота	Население	Рейтинг придорожных кафе
Вала-Вала, штат Вашингтон	15 июня	75°	1204 фута	29686	4/5
Мэджик-Сити, штат Айдахо	25 июня	74°	5312 футов	50	3/5
Баунтифул, штат Юта	10 июля	91°	4226 футов	41173	4/5
Лэст Чанс, штат Колорадо	23 июля	102°	4780 футов	265	3/5
Трут-ор-Консекуэнсес, штат Нью-Мексико	9 августа	93°	4242 фута	7289	5/5
Уай, штат Аризона	18 августа	104°	860 футов	480	3/5

Каждый кусочек данных называют ячейкой. Иногда также просто говорят «табличные данные».

МОЗГОВОЙ ШТУРМ

Если вас попросят сделать XHTML-таблицу, то как вы будете проектировать элементы, которые могут быть использованы для ее создания, включая заголовки, строки, столбцы и сами табличные данные?

Как создать таблицу, используя XHTML

Прежде чем взяться за сайт Тони и вносить в него изменения, сделаем так, чтобы таблица работала в отдельном XHTML-файле. Мы уже начали создавать таблицу и ввели заголовки и две первые строки в файл `table.html` из папки `chapter13/journal/`. Просмотрите его:

Просто небольшой фрагмент CSS-кода, чтобы можно было видеть структуру таблицы в браузере. Пока не волнуйтесь об этом.

```
<!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="ru" xml:lang="ru">
<head>
   <meta http-equiv="content-type" content="text/html; charset=windows-1251" />
   <style type="text/css">
      td, th {border: 1px solid black;}
   </style>
   <title>Тестирование таблицы Тони</title>
</head>
<body>
   <table>
      <tr>
         <th>Город</th>
         <th>Дата</th>
         <th>Температура</th>
         <th>Высота</th>
         <th>Население</th>
         <th>Рейтинг придорожных кафе</th>
      </tr>
      <tr>
         <td>Вала-Вала, штат Вашингтон</td>
         <td>15 июня</td>
         <td>75</td>
         <td>1204 фута</td>
         <td>29686</td>
         <td>4/5</td>
      </tr>
      <tr>
         <td>Мэджик-Сити, штат Айдахо</td>
         <td>25 июня</td>
         <td>74</td>
         <td>5312 футов</td>
         <td>50</td>
         <td>3/5</td>
      </tr>
   </table>
</body>
</html>
```

Мы используем тег <table>, чтобы указать, что начинается таблица.

Это первая строка, которую мы начинаем тегом <tr>.

Каждый элемент <th> — это название столбца таблицы.

Обратите внимание, что заголовки таблицы перечисляются один за другим. Может показаться, что они должны составить столбец, но на самом деле определяется строка с заголовками. Взгляните на список Тони еще раз и обратите внимание на то как в нем располагаются заголовки.

Каждый элемент <tr> формирует строку таблицы

Это начало второй строки, которая описывает город Вала-Вала.

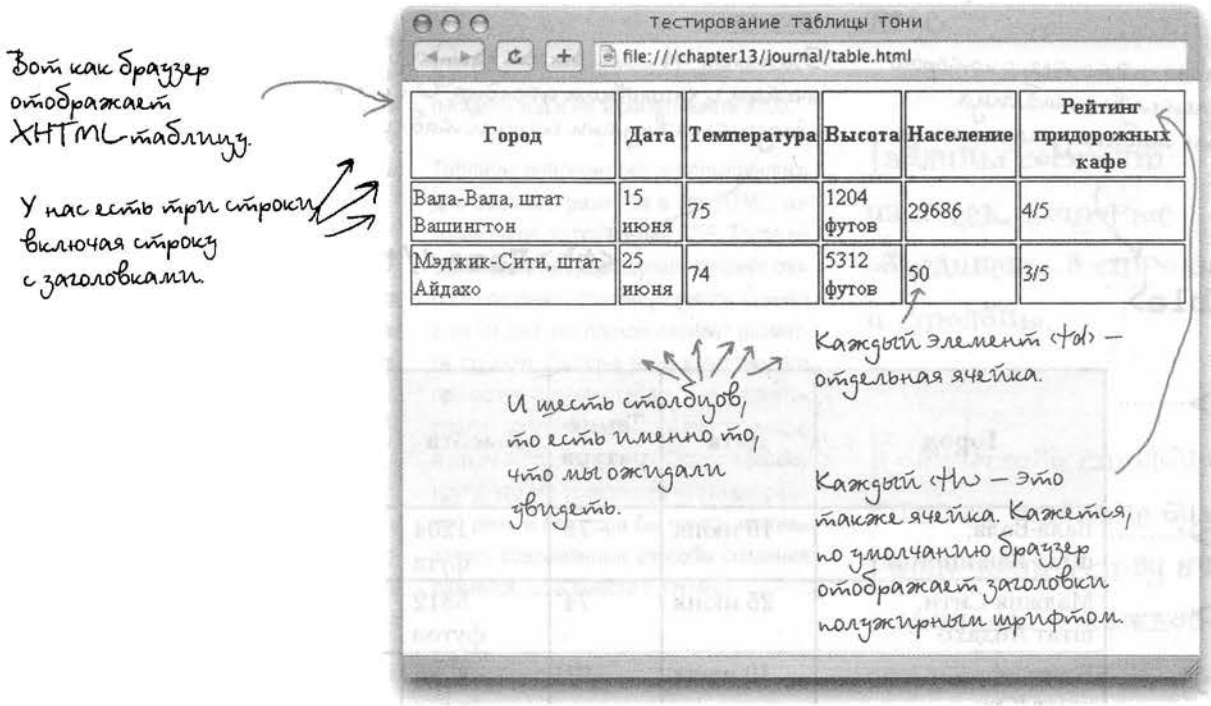
Каждый элемент <td> содержит данные одной табличной ячейки, и все ячейки находятся в разных столбцах.

Все эти элементы <td> составляют строку.

Это третья строка. И вновь каждый элемент <td> содержит данные одной ячейки.

Что создает браузер

Посмотрим, как браузер отображает эту XHTML-таблицу. Предупреждаем вас: это будет не самая красивая таблица, но она уже будет нормально выглядеть. О ее внешнем виде мы позаботимся очень скоро, а пока убедимся в том, что вы усвоили самое главное.





Упражнение

Сначала напечатайте XHTML-код для тестирования таблицы Тони, приведенный на предыдущей странице. Хотя печатать его будет скучно, это поможет вам усвоить и запомнить структуру тегов **<table>**, **<tr>**, **<th>** и **<td>**. Когда закончите, быстро протестируйте код, а затем добавьте оставшиеся данные из таблицы Тони. И снова протестируйте.

Разделение таблицы



Вы видели, что для создания одиночной таблицы используются четыре элемента: <table>, <tr>, <th> и <td>. Рассмотрим каждый по отдельности, чтобы понять, какую роль они играют в создании таблицы.

Тег <table> — это тег, с которого начинается вся таблица. Создавая таблицу, начинайте с него.

Элемент <th> содержит данные одной ячейки с заголовком столбца. Он должен находиться внутри строки таблицы.

Тег </tr> заканчивает строку таблицы.

```
<table>
<tr>
<th>Дата</th>
</tr>
</table>
```

Город	Дата	Температура	Высота	Население	Рейтинг придорожных кафе
Вала-Вала, штат Вашингтон	15 июня	75°	1204 фута	29686	4/5
Мэджик-Сити, штат Айдахо	25 июня	74°	5312 футов	50	3/5
Баунтифул, штат Юта	10 июля	91°	4226 футов	41173	4/5
Лэст Чанс, штат Колорадо	23 июля	102°	4780 футов	265	3/5
Трут-ор-Консекуэнсес, штат Нью-Мексико	9 августа	93°	4242 фута	7289	5/5
Уай, штат Аризона	18 августа	104°	860 футов	480	3/5

Каждый элемент <tr> задает строку таблицы. Итак, все табличные данные одной строки вкладываются в элемент <tr>.

```
<td>9 августа</td>
</table>
```

Элемент <td> содержит данные одной ячейки таблицы. Он должен находиться внутри строки таблицы.

Тег </table> завершает таблицу.

Часто Задаваемые Вопросы

В: Почему нет отдельного элемента для столбца таблицы? Мне кажется, что это очень важно.

О: Разработчики XHTML решили дать вам возможность задавать таблицы через строки, а не через столбцы. Но обратите внимание, что, указывая элементы <td> для каждой строки, вы все равно неявным образом определяете каждый столбец.

В: Что будет, если в одной из моих строк недостаточное количество элементов? Иначе говоря, если их меньше, чем столбцов в таблице?

О: Самый простой способ справиться с этим — оставить ячейку пустой, то есть написать <td></td>. Если вы просто пропустите ячейку, то таблица будет выстроена неправильно, поэтому нужно задать все ячейки данных, даже если они пустые.

В: Если я хочу, чтобы заголовки таблицы были расположены в левом столбце, а не в верхней строке, могу ли я это реализовать?

О: Конечно. Вам просто нужно будет поместить элементы с заголовками в отдельные строки, вместо того чтобы располагать их все в первой строке. Если первыми элементами каждой строки будут элементы <th>, то первый столбец будет состоять из одних заголовков.

В: Мой друг показал мне классный трюк: он сделал разметку всей страницы с помощью таблицы. Ему даже не пришлось использовать CSS!

О: Отправляйтесь в CSS-тюрьму. Вы пропускаете ход и не зарабатываете $200.

Таблицы повсеместно использовались для создания разметок в эру HTML, которая была до появления CSS. Тогда на самом деле не существовало лучшего способа создания сложных разметок. Однако в наши дни это плохой вариант разметки страниц. Сегодня всем известно, что при использовании таблиц для разметок сложно получить желаемый вид страницы, а затем поддерживать ее. Скажите своему другу, что его технология — это вчерашний день и ему пора бы начать использовать современные способы создания разметок: CSS вместе с XHTML.

В: Разве таблицы — это не способ изображения элементов? Что случилось с разделением содержимого и структуры?

О: Не совсем. С помощью таблиц вы определяете взаимосвязь между действительными табличными данными. Мы будем использовать CSS, чтобы видоизменять форму страниц.

Таблицы позволяют определять табличные данные в вашем HTML-коде.

Таблицы состоят из ячеек, которые объединены в строки и столбцы.

Количество столбцов в вашей таблице будет равно количеству ячеек с данными в каждой строке.

Самое главное: таблицы не предназначены для разметки элементов; это работа CSS.



Поработайте браузером

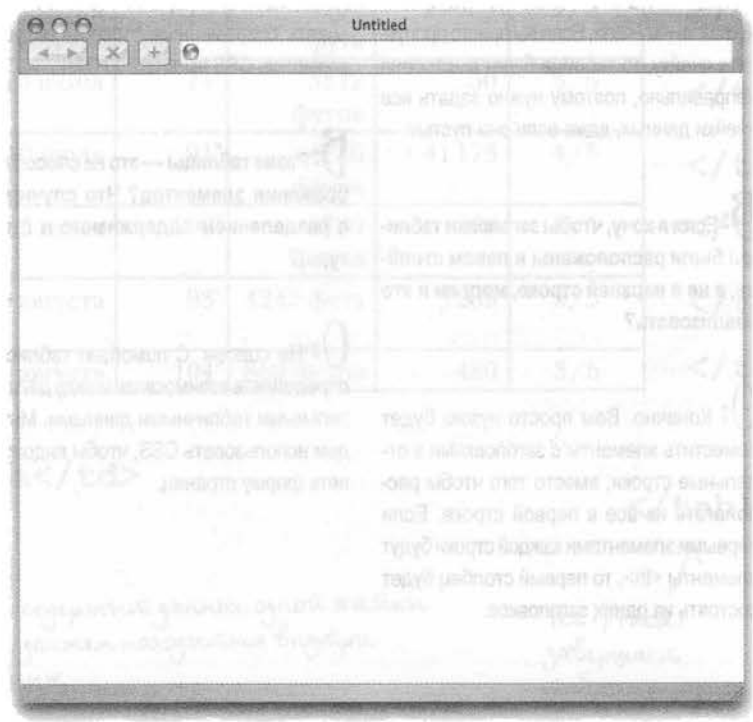
Слева вы найдете XHTML-код для таблицы. Ваша задача — поставить себя на место браузера, отображающего эту страницу. Выполнив упражнение, проверьте правильность решения в конце главы.

```
<table><tr><th>Исполнитель</th>
<th>Альбом</th></tr><tr>
<td>Enigma</td><td>Le Roi Est Mort,
Vive Le Roi!</td></tr> <tr><td>LTJ
Bukem</td>
<td>Progression Sessions 6</td>
</tr><tr>
<td>Timo Maas</td>
<td>Pictures</td></tr></table>
```

Это просто XHTML-таблица.

Ох! Кому-то придется научиться структурировать свой XHTML-код.

Нарисуйте таблицу здесь.



Добавление заголовка и краткого описания таблицы

Есть несколько вещей, которые вы можете сделать сразу же, чтобы усовершенствовать свою таблицу, — можете добавить заголовок и краткое описание.

```
<table summary="В этой таблице хранятся данные о тех
городах, которые я посетил во время путешествия. Я указал
в ней дату посещения города, температуру воздуха в этот
день, высоту над уровнем моря и количество населения этого
города. Я также включил в таблицу рейтинг придорожных кафе,
в которых обедал, по шкале от 1 до 5.">
    <caption>
        Города, которые я посетил во время
        путешествия на скутере по США
    </caption>
    <tr>
        <th>Город</th>
        <th>Дата</th>
        <th>Температура</th>
        <th>Высота</th>
        <th>Население</th>
        <th>Рейтинг придорожных кафе</th>
    </tr>
    <tr>
        <td>Вала-Вала, штат Вашингтон</td>
        <td>15 июня</td>
        <td>75</td>
        <td>1204 фута</td>
        <td>29686</td>
        <td>4/5</td>
    </tr>
    <tr>
        <td>Мэджик-Сити, штат Айдахо</td>
        <td>25 июня</td>
        <td>74</td>
        <td>5312 фута</td>
        <td>50</td>
        <td>3/5</td>
    </tr>
    .
    .
    .
</table>
```

Далее идут остальные строки таблицы

Краткое описание не появляется на экране. Оно используется для представления информации пользователям и выглядит как небольшой отрывок текста, который экранные дикторы прочитывают для пользователей вслух, чтобы описать таблицу.

Название отображается в окне браузера. По умолчанию в большинстве браузеров оно отображается над страницей.

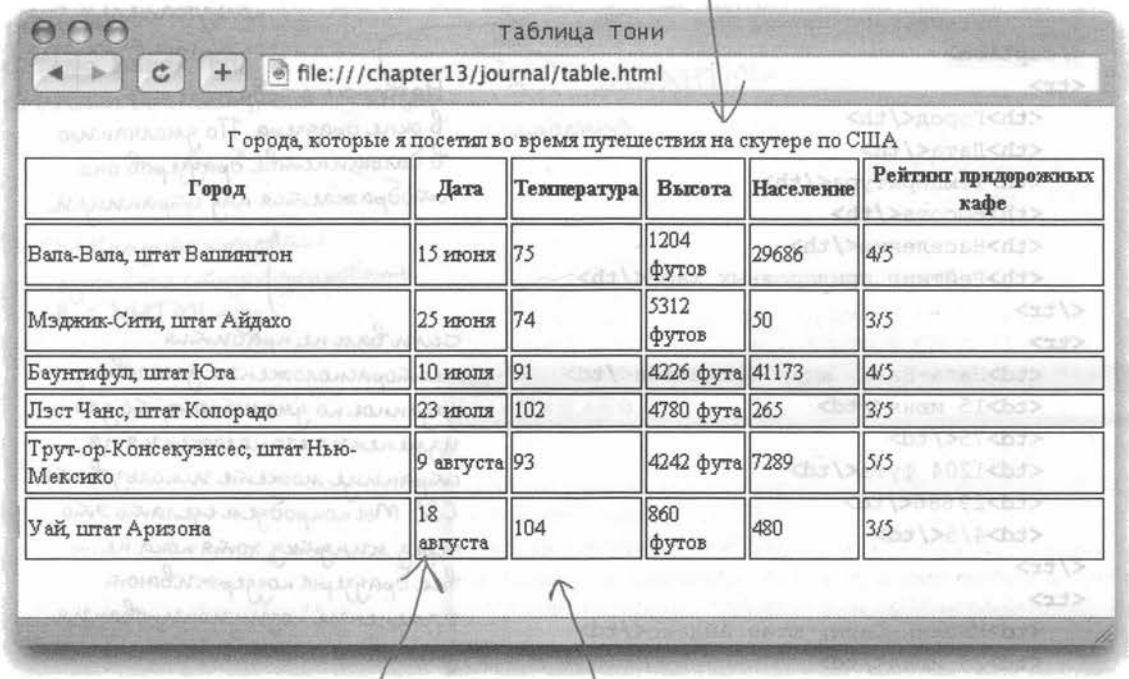
Если вам не нравится месторасположение заголовка, заданное по умолчанию, то для изменения его положения на странице можете использовать CSS. Мы попробуем сделать это через минутку, хотя пока не все браузеры поддерживают изменение позиционирования заголовка.

Сделаем тест... и подумаем о стиле

Добавьте заголовок и краткое описание к вашей таблице. Сохраните изменения и обновите страницу.

Вы не увидите краткого описания. Оно предназначено в основном для экранных дикторов, которые прочитывают его для людей с ослабленным зрением, представляя им больше информации о таблице

Заголовок расположен над таблицей. Скорее всего, он бы лучше смотрелся под ней.



Города, которые я посетил во время путешествия на скутере по США

Город	Дата	Температура	Высота	Население	Рейтинг придорожных кафе
Вала-Вала, штат Вашингтон	15 июня	75	1204 футов	29686	4/5
Мэджик-Сити, штат Айдахо	25 июня	74	5312 футов	50	3/5
Баунтифул, штат Юта	10 июля	91	4226 фута	41173	4/5
Лэст Чанс, штат Колорадо	23 июля	102	4780 фута	265	3/5
Трут-ор-Консекуэнсес, штат Нью-Мексико	9 августа	93	4242 фута	7289	5/5
Уай, штат Аризона	18 августа	104	860 футов	480	3/5

Нам однозначно нужно добавить отступы для ячеек с табличными данными, чтобы они лучше читались.

Если добавить немного оранжевого цвета, то таблица действительно отлично впишется в общую картину.

Линии границы совсем некрасивые и какие-то «тяжелые». Можно использовать намного более «легкие» границы для ячеек, хотя было бы здорово нарисовать темную границу вокруг всей таблицы

Перед тем как мы перейдем к стилизации, поместим таблицу на страницу Тони

Прежде чем стилизовать новую таблицу Тони, нужно поместить ее на главную страницу. Вспомните, что для главной страницы уже заданы семейства и размеры шрифтов, а также множество других стилей, которые унаследует наша таблица. Итак, пока мы не поместим таблицу на главную страницу, мы не узнаем, как она на самом деле будет выглядеть.

Итак, начните с открытия файла `journal.html` из папки `chapter13/journal`, найдите запись от 20 августа и внесите нижеприведенные изменения. Когда справитесь с этим, переходите к следующей странице, не тестируя то, что получилось.

```
<h2>20 августа, 2005</h2>
<p>
   <img src="images/segway2.jpg" alt="Я и мой скутер в Нью-Мексико" />
</p>

<p>
Итак, я уже проехал 1200 миль и побывал в некоторых
интересных местах:
</p>

<ol>
   <li>Вала Вала, штат Вашингтон</li>
   <li>Мэджик Сити, штат Айдахо</li>
   <li>Бьютифул, штат Юта</li>
   <li>Лост Чанс, штат Колорадо</li>
   <li>Трут ор Консекуэнсес, штат Нью-Мексико</li>
   <li>Уай, штат Аризона</li>
</ol>
```

Это старый список городов. Удалите его, потому что вместо него мы будем использовать таблицу.

```
<table summary="В этой таблице хранятся данные о тех городах, которые я посетил во время
  путешествия. Я указал в ней дату посещения города, температуру воздуха в этот день,
  высоту над уровнем моря и количество населения этого города. Я также включил в таблицу
  рейтинг придорожных кафе, в которых обедал, по шкале от 1 до 5.">
   <caption>Города, которые я посетил во время путешествия на скутере по США</caption>
   <tr>
      <th>Город</th>
      <th>Дата</th>
      <th>Температура</th>
      <th>Высота</th>
      <th>Население</th>
      <th>Рейтинг придорожных кафе</th>
   </tr>
   .
   .
   .
</table>
```

Здесь будет новая таблица. Самый простой способ поместить ее сюда — скопировать из предыдущего файла.

Стилизуем таблицу

Теперь нам нужно скопировать правила стиля для таблицы в файл `journal.css`. Поскольку мы все равно хотим их полностью изменить, давайте просто добавим новые стили. Добавьте новые правила, описанные ниже, в самый конец файла с таблицей стилей.

```
body {
    font-family: Verdana, Geneva, Arial, sans-serif;
    font-size: small;
}
h1, h2 {
    font-weight: normal;
    color: #cc6600;
    border-bottom: thin dotted #888888;
}
h1 {
    font-size: 170%;
}
h2 {
    font-size: 130%;
}
blockquote {
    font-style: italic;
}
```

Это те стили, которые на данный момент применяются на веб-странице Тони. Мы добавляли их в главе 9. Под ними мы добавим новые правила стиля для таблицы.

```
table {
    margin-left: 20px;
    margin-right: 20px;
    border: thin solid black;
    caption-side: bottom;
}

td, th {
    border: thin dotted gray;
    padding: 5px;
}

caption {
    font-style: italic;
    padding-top: 8px;
}
```

Сначала мы стилизуем таблицу. Добавим левое и правое поля и тонкую черную границу для таблицы

Переместим заголовок под таблицу.

Кроме того, изменим вид границы для ячеек с данными и сделаем ее более легкой. Нарисуем границу точечной линией серого цвета.

Добавим отступы для ячеек, чтобы между их границей и содержимым было свободное место.

Это правило для стилизации заголовка. Мы меняем стиль шрифта на курсивный и добавляем небольшой отступ сверху.

Тест для стилизованной таблицы

Мы внесли сразу много изменений. Убедитесь, что вы их сохранили, и проверьте валидность файлов. Затем откройте страницу `journal.html` в своем браузере.

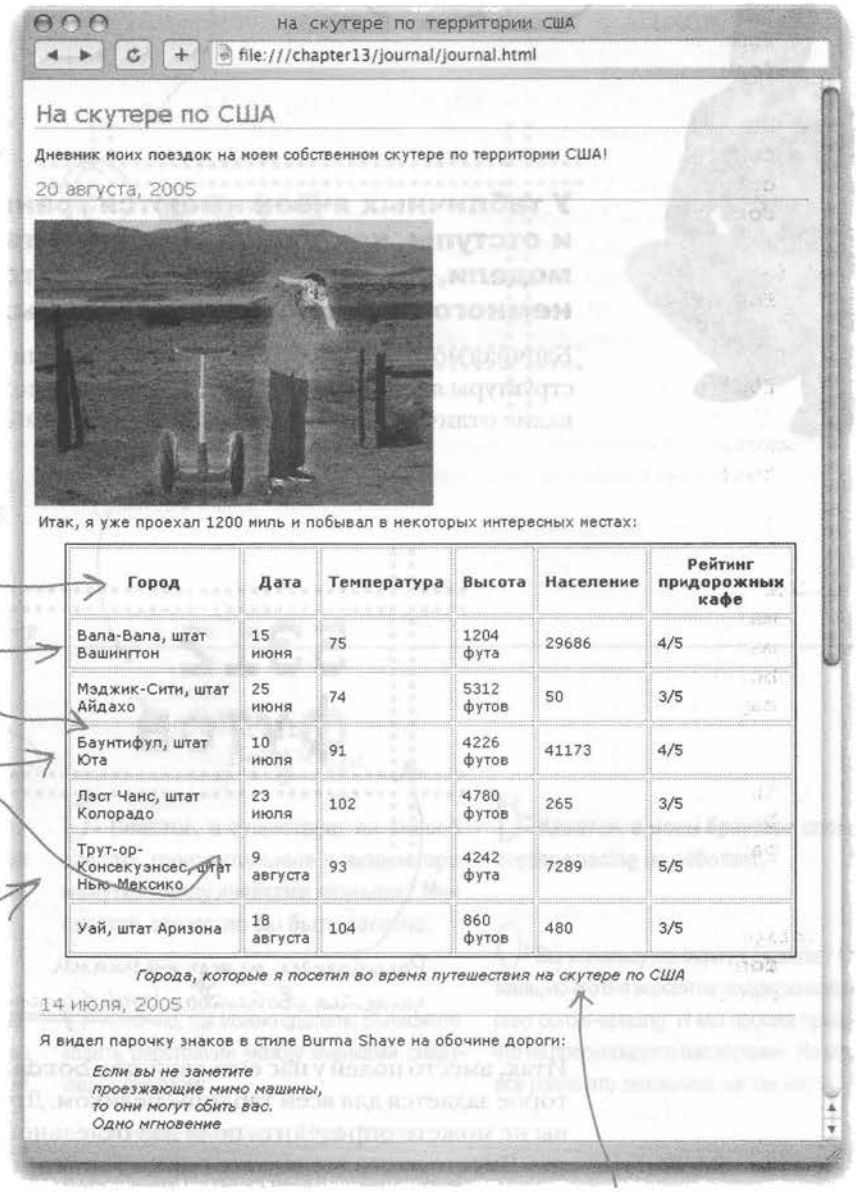
После того как вы стилизовали таблицу, она выглядит совсем иначе. Она унаследовала некоторые стили от основной страницы дневника Тони.

Теперь все используемые шрифты принадлежат семейству sans-serif и имеют меньший размер. Это из тех правил стиля, которые уже были заданы для страницы ранее.

Теперь у нас есть темная граница и точечные линии.

У нашей таблицы есть поля и небольшие отступы, заданные для каждой ячейки.

Однако эти точечные линии сильно бросаются в глаза и отвлекают внимание. Кроме того, нет ничего хорошего в том, что между каждыми двумя ячейками они дублируются.



Город	Дата	Температура	Высота	Население	Рейтинг придорожных кафе
Вала-Вала, штат Вашингтон	15 июня	75	1204 фута	29686	4/5
Мэджик-Сити, штат Айдахо	25 июня	74	5312 футов	50	3/5
Баунтифул, штат Юта	10 июля	91	4226 футов	41173	4/5
Лэст Чанс, штат Колорадо	23 июля	102	4780 футов	265	3/5
Трут-ор-Консекуэнсес, штат Нью-Мексико	9 августа	93	4242 фута	7289	5/5
Уай, штат Аризона	18 августа	104	860 футов	480	3/5

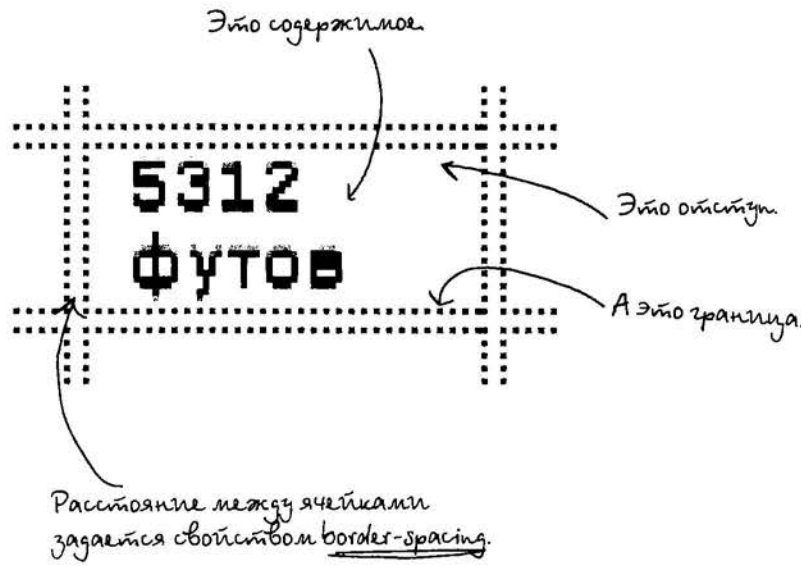
Помните, что в браузерах, которые не поддерживают свойство caption-side, заголовок будет расположен над таблицей.

дальше »



У табличных ячеек имеются граница и отступы, как у любого элемента в блочной модели, но что касается полей, то ячейки немного отличаются от остальных элементов.

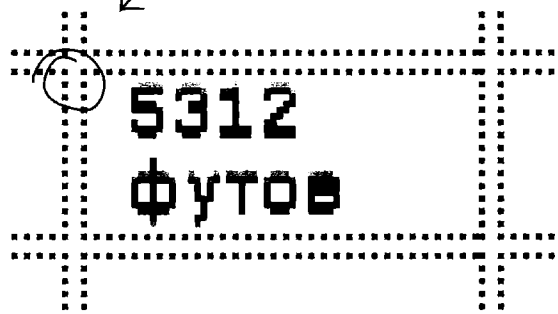
Блочная модель может быть использована для представления структуры ячейки, но когда дело доходит до полей, то есть кое-какие отличия. Рассмотрим одну из ячеек таблицы Тони:



Итак, вместо полей у нас есть свойство **`border-spacing`**, которое задается для всей таблицы целиком. Другими словами, вы не можете определить поле для отдельной ячейки таблицы. Вместо этого вы задаете один общий промежуток, который будет использован как поле между всеми ячейками.

Возьми в руку карандаш

Дублированные точечные линии в таблице Тони очень сильно бросаются в глаза. Было бы лучше, если бы вокруг каждой ячейки была одинарная точечная линия, что не отвлекало бы внимания от содержимого таблицы. Вы можете придумать способ сделать это, используя те знания о стилях, которые имеете на данный момент? Попробуйте выполнить это и проверьте свое решение в конце главы.



Часто Задаваемые Вопросы

В: Промежуток между ячейками определяется для всей таблицы, в то время как поля задаются для каждого элемента индивидуально?

О: Именно. У ячеек таблицы нет полей. У них есть лишь свободное пространство вокруг границ, которое задается для всей таблицы целиком. Вы не можете определять промежутки между различными ячейками индивидуально.

В: Понятно, а существует ли способ сделать горизонтальные и нижние промежутки между ячейками разными? Мне кажется, это могло бы быть полезно.

О: Конечно, так можно сделать. Вы можете задать расстояние между ячейками следующим образом:

```
border-spacing: 10px 30px;
```

В результате вы установите горизонтальные промежутки шириной 10 пикселов, а вертикальные — высотой 30 пикелов.

В: Кажется, в моем браузере свойство border-spacing не работает.

О: Вы используете Internet Explorer? Очень жаль, но его 6-я версия не поддерживает свойство border-spacing. И мы просим прощения, что не предупредили вас заранее. Но ведь вы все равно это запомните, не так ли?

Добавление границ

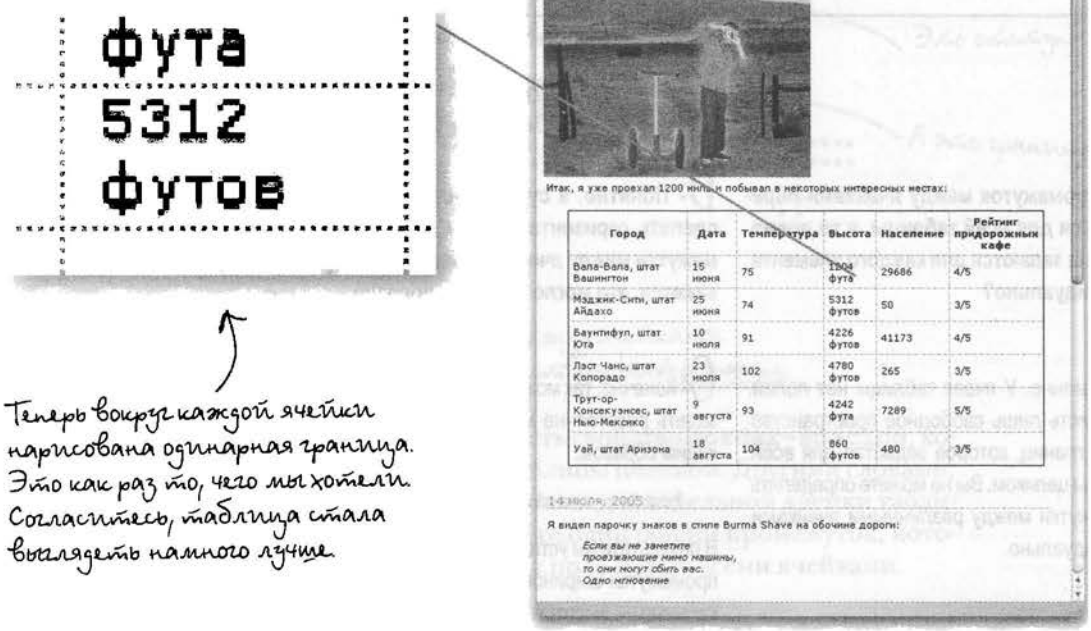
Есть еще один способ решить проблему с границами. Вы можете использовать CSS-свойство **border-collapse**, позволяющее объединить границы между ячейками так, чтобы расстояния между ячейками не было вовсе. Если вы это сделаете, браузер будет игнорировать все промежутки между ячейками (а также между ячейками и границей самой таблицы), которые вы задали для таблицы. Он также будет соединять две соседние границы в одну.

Вы можете установить свойство **border-collapse**, как показано ниже. Внесите эти изменения в файл `journal.css`:

```
table {
    margin-left: 20px;
    margin-right: 20px;
    border: thin solid black;
    caption-side: bottom;
    border-collapse: collapse;
}
```

Добавьте свойство border-collapse и установите для него значение collapse.

Сохраните файл и обновите страницу; затем оцените изменения во внешнем виде границы.



Теперь вокруг каждой ячейки нарисована одинарная граница. Это как раз то, чего мы хотели. Согласитесь, таблица стала выглядеть намного лучше.

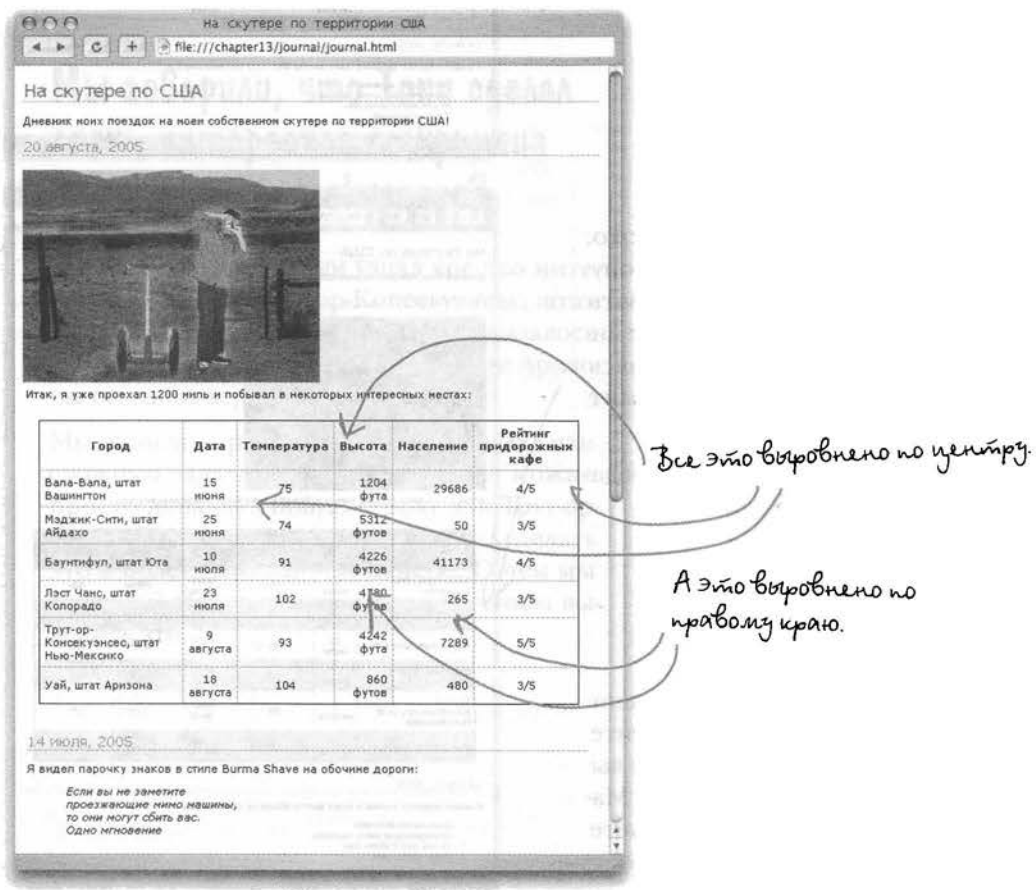
Возьми в руку карандаш

Вы уже становитесь отличным специалистом в XHTML и CSS, поэтому мы уверены, что вы справитесь с этим заданием.

Нужно еще больше приукрасить таблицу и начать лучше с решения некоторых проблем с выравниванием текста. Мы хотим, чтобы дата, температура и рейтинг придорожных кафе были выровнены по центру. А как насчет выравнивания по правому краю для высоты и населения? Как вы будете это делать?

Вот подсказка: создайте два класса, один — для текста, выровненного по центру, другой — для текста, выровненного по левому краю. Затем просто используйте свойство **`text-align`** для каждого. Наконец, добавьте соответствующий класс в каждый элемент **`<td>`**.

Возможно, это прозвучит достаточно грубо для вас, но делайте все шаг за шагом. Конечно же, вы найдете решение в конце главы, но прежде, чем посмотреть его, уделите время тому, чтобы выполнить задание самостоятельно.

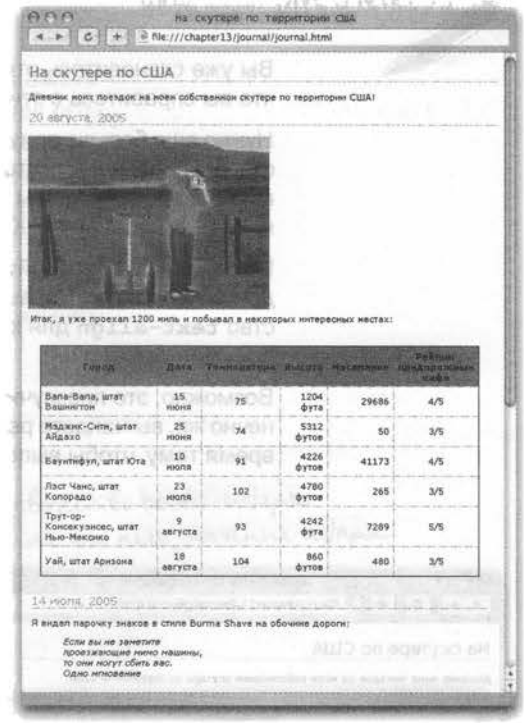


Как насчет цвета?

Вы знаете, что Тони нравится оранжевый цвет, и нет причины не добавить его в таблицу. Это не только будет замечательно выглядеть, но и позволит существенно улучшить читаемость таблицы. Как и для других элементов, вам нужно установить свойство **background-color** для ячеек таблицы, чтобы изменить их цвета (обратите внимание, что все те знания, которые вы получили об XHTML и CSS, начинают объединяться!). Вот как это делается:

```
th {
    background-color: #cc6600;
}
```

Добавьте это новое правило в файл `journal.css` и обновите страницу. Вот что вы увидите.



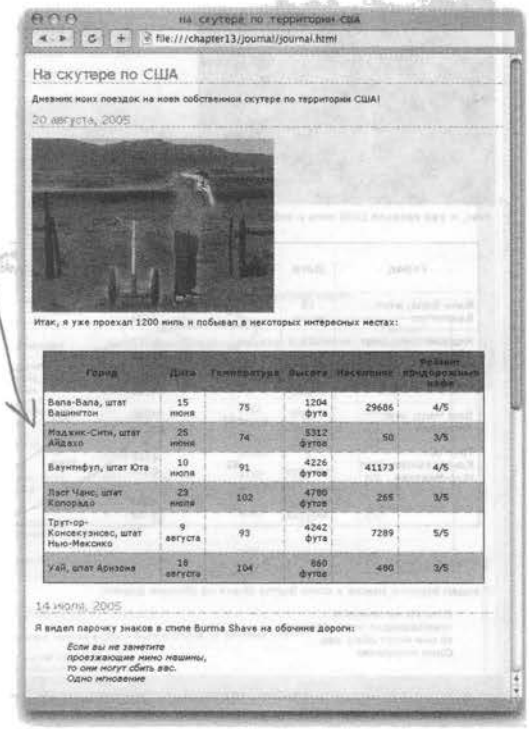
Как насчет того, чтобы раскрасить строки таблицы?

До сих пор цвета смотрелись достаточно неплохо. Итак, переведем их на уровень выше. Самый популярный способ разукрашивать таблицы – задавать для строк различные цвета, что позволяет легче визуально отделять строки одна от другой, не путаясь в том, какому столбцу и какой строке принадлежит каждая ячейка. Проверьте это.

Сложно ли это сделать в CSS? Нет. Сначала определите новый класс, например **cellcolor**:

```
.cellcolor {
    background-color: #fcba7a;
}
```

Теперь остается добавить атрибут **class** со значением **cellcolor** в каждую строку, которую вы хотите раскрасить в этот цвет. Итак, в данном примере вы находите открывающие теги **<tr>** для городов Мэджик-Сити, Лэст Чанс и Уай и в каждый добавляете **class="cellcolor"**.





Упражнение

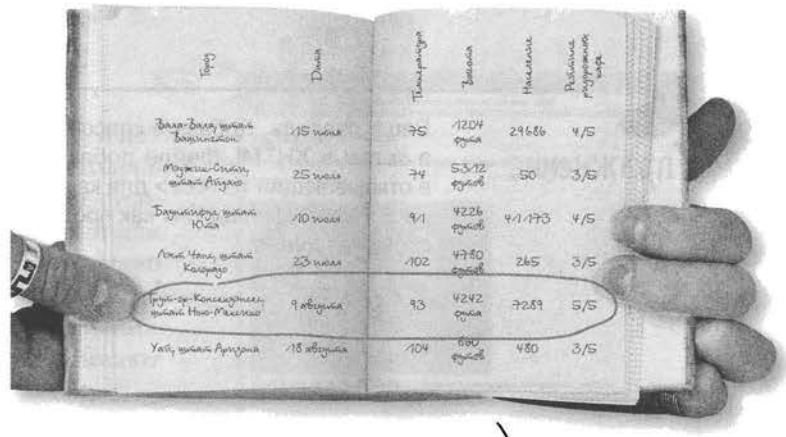
Ваша очередь. Добавьте класс **cellcolor** в файл `journal.css`, а затем в XHTML-файле добавьте строку **class="cellcolor"** в открывающий тег **<tr>** для каждой строки, которая должна быть раскрашена. Перед тем как продолжить, проверьте правильность своего решения.

Мы говорили, что Тони сделал очень интересное открытие в Трут-ор-Консекуэнсес?

Надо сказать, что Тони узнал кое-что интересное в городе Трут-ор-Консекуэнсес, штат Нью-Мексико. На самом деле это показалось ему настолько интересным, что после Аризоны он снова вернулся туда.

Мы, конечно, рады за Тони, но он задал нам сложную задачку с таблицей. Хотя можно просто добавить новую строку для Трут-ор-Консекуэнсес, мы все же хотели бы сделать это в более изысканной манере. О чем мы говорим? Переверните страницу, чтобы выяснить это.





Посмотрим на таблицу Тони еще раз

Поскольку Тони возвращался в Нью-Мексико, то он добавил новую запись от 27 августа прямо под первоначальной записью для Трут-ор-Консекуэнсес. Он также повторно использовал некоторые ячейки, информация в которых не менялась (отличный технический прием для уменьшения информации в таблице). Вы можете видеть, что, когда он добавил новую строку, ему осталось записать те данные, которые отличались при его втором посещении этого города (дата, температура и рейтинг кафе).

Здесь отмечены оба визита Тони в Трут-ор-Консекуэнсес.

Город	Дата	Температура	Высота	Население	Рейтинг придорожных кафе
Вала-Вала, штат Вашингтон	15 июня	75°	1204 фута	29 686	4/5
Мэджик-Сити, штат Айдахо	25 июня	74°	5312 футов	50	3/5
Баунтифул, штат Юта	10 июля	91°	4226 футов	41173	4/5
Лэст Чанс, штат Колорадо	23 июля	102°	4780 футов	265	3/5
Трут-ор-Консекуэнсес, штат Нью-Мексико	9 августа	93°	4242 фута	7289	5/5
	27 августа	98°			4/5
Уай, штат Аризона	18 августа	104°	860 футов	480	3/5

Теперь эти данные занимают ДВЕ строки в таблице

Но как же нам отразить это в XHTML? Кажется, придется добавить новую строку и просто дублировать в ней название города, население и высоту... Не будем забегать вперед, ведь у нас есть специальная технология. Используя XHTML-таблицы, вы можете сделать так, чтобы ячейка охватывала несколько строк (или столбцов). Посмотрим, как это работает.

Как сделать, чтобы ячейка охватила несколько строк

Что это значит – ячейка охватывает несколько строк? Снова посмотрим на записи для города Трут-ор-Консекуэнсес в таблице Тони. Ячейки с названием города, высотой и населением охватывают *две строки*, а не одну, в то время как ячейки с датой, температурой и рейтингом придорожных кафе – одну строку, что является стандартным поведением ячеек.

Город	Дата	Температура	Высота	Население	Рейтинг придорожных кафе
Вала-Вала, штат Вашингтон	15 июня	75°	1204 фута	29686	4/5
Мэджик-Сити, штат Айдахо	25 июня	74°	5312 футов	50	3/5
Баунтифул, штат Юта	10 июля	91°	4226 футов	41173	4/5
Лэст Чанс, штат Колорадо	23 июля	102°	4780 футов	265	3/5
Трут-ор-Консекуэнсес, штат Нью-Мексико	9 августа	93°	4242 фута	7289	5/5
	27 августа	98°			4/5
Уай, штат Аризона	18 августа	104°	860 футов	480	3/5

Ячейки с датой, температурой и рейтингом придорожных кафе занимают по одной строке.

Эти ячейки охватывают две строки.

Как же реализовать это в XHTML? На самом деле это намного легче, чем вы думаете. Нужно использовать атрибут **rowspan**, чтобы указать, как много строк займет ячейка, а затем удалить соответствующий элемент с данными из других строк, которые покрывает эта ячейка. Проще понять это на примере.

В таблице есть две строки с данными о Нью-Мексико.

Для ячеек, содержащих те данные, которые не изменились при повторном визите (город, высота и население), мы добавляем атрибут rowspan. Он указывает, что ячейка с этими данными охватывает две строки.

```
<tr>
  <td rowspan="2">Трут-ор-Консекуэнсес, штат Нью-Мексико</td>
  <td class="center">9 августа</td>
  <td class="center">93</td>
  <td rowspan="2" class="right">4242 фута</td>
  <td rowspan="2" class="right">7289</td>
  <td class="center">5/5</td>
</tr>
<tr>

  <td class="center">27 августа</td>
  <td class="center">98</td>


  <td class="center">4/5</td>
</tr>
```

Город указывать не нужно, так как в предыдущей строке мы задали для него атрибут rowspan.

То же самое для высоты и населения.

Во второй строке мы указываем только те данные, которых пока нет (дата, температура и новый рейтинг).

Кто и что делает?

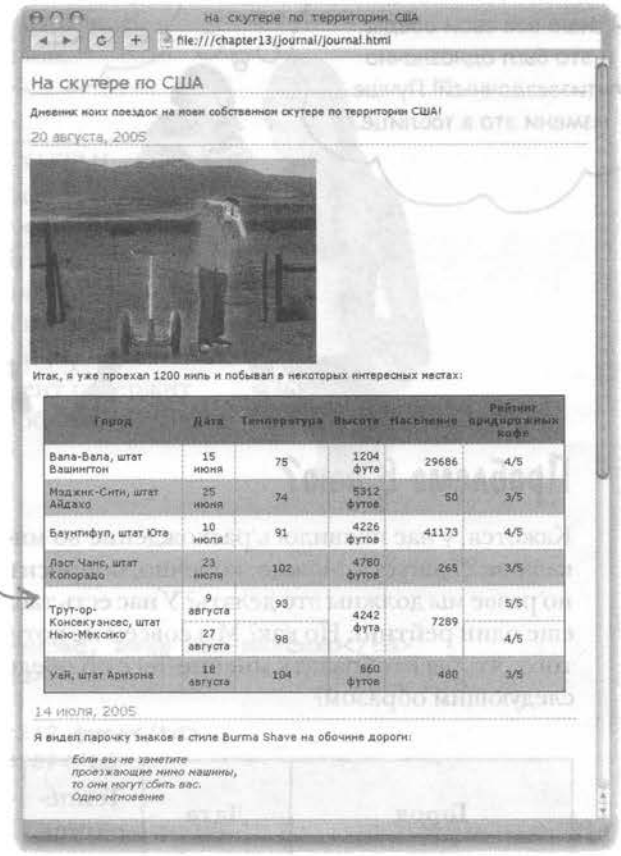
Для того чтобы удостовериться, что вы хорошо все усвоили, мы попросим вас нарисовать стрелку от каждого элемента **<td>** к соответствующей ему ячейке таблицы. Проверьте свои ответы, прежде чем продолжать двигаться вперед.

```
<tr>
   <td rowspan="2">Трут-ор-Консекуэнсес, Нью-Мексико</td>
   <td class="center">9 августа</td>
   <td class="center">93</td>
   <td rowspan="2" class="right">4242 фута</td>
   <td rowspan="2" class="right">7289</td>
   <td class="center">5/5</td>
</tr>
<tr>

   <td class="center">27 августа</td>
   <td class="center">98</td>


   <td class="center">4/5</td>
</tr>
```

Трут-ор-Консекуэнсес, штат Нью-Мексико	9 августа	93°	4242 фута	7289	5/5
	27 августа	98°			4/5

Новая и усовершенствованная таблица

Внесите соответствующие изменения в таблицу из файла `journal.html` и протестируйте страницу. Взгляните на таблицу. Подумайте о том, что именно вы делаете с таблицей: вы используете XHTML-код, чтобы указать, что определенные ячейки охватывают несколько строк, а затем удаляете лишние элементы **<td>** из следующих строк.



Теперь таблица выглядит замечательно, и в ней нет избыточной информации.

Часто Задаваемые Вопросы

В: Вы сказали, что можно сделать так, чтобы одна ячейка охватывала несколько столбцов?

О: Конечно, можно. Просто добавьте атрибут colspan в элемент <td> и укажите в нем количество столбцов. С помощью rowspan вы объединяете строки, а посредством colspan удаляете элементы с табличными данными из *той же* строки (поскольку вы объединяете столбцы, а не строки).

В: Можно ли для одного и того же элемента <td> использовать и атрибут colspan, и атрибут rowspan?

О: Конечно, можно. Только не забудьте привести в порядок все элементы <td> таблицы. Другими словами, вам нужно будет удалить элементы <td>, которые охватывает ваша ячейка.

В: Неужели вам действительно кажется, что ячейки, охватывающие несколько строк, смотрятся лучше?

О: Они определенно уменьшают количество информации в таблице, что обычно хорошо. Если вы посмотрите на некоторые реально существующие таблицы, то увидите, что ячейки, объединяющие несколько строк или столбцов, используются достаточно часто, так что здорово уметь делать это в XHTML. Но если вам больше нравилась предыдущая версия таблицы, можете поменять свой XHTML-код и вернуться к ней.

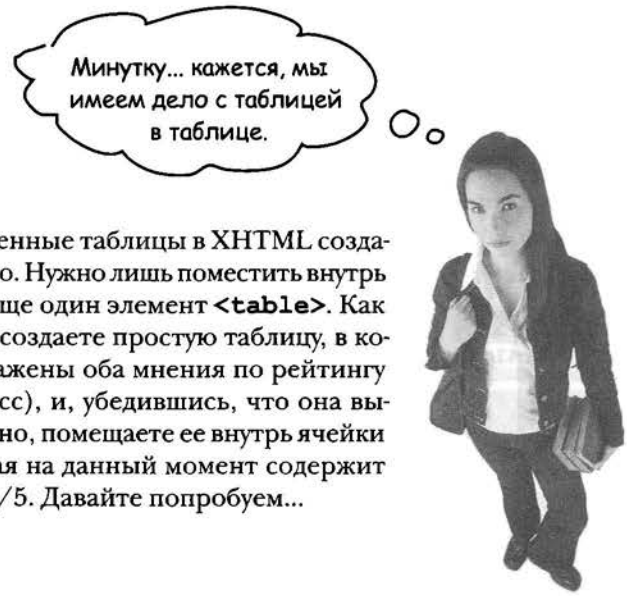


Проблема в раю?

Кажется, у нас появилось расхождение во мнениях о рейтинге обеда в придорожном кафе за 27 августа. Можно, конечно, попросить Тони и Тесс прийти к единому мнению, но разве мы должны это делать? У нас есть таблица, и мы должны суметь добавить в нее еще один рейтинг. Но как? Мы совсем не хотим добавлять еще одну запись только для того, чтобы отобразить мнение Тесс об обеде. Хммм... Почему бы нам не сделать это следующим образом?

Город	Дата	Температура	Высота	Население	Рейтинг придорожных кафе
Вала-Вала, штат Вашингтон	15 июня	75°	1204 фута	29686	4/5
Мэджик-Сити, штат Айдахо	25 июня	74°	5312 футов	50	3/5
Баунтифул, штат Юта	10 июля	91°	4226 футов	41173	4/5
Лэст Чанс, штат Колорадо	23 июля	102°	4780 футов	265	3/5
Трут-ор-Консекуэнсес, штат Нью-Мексико	9 августа	93°	4242 фута	7289	5/5
	27 августа	98°			Тесс 5/5 Тони 4/5
Уай, штат Аризона	18 августа	104°	860 футов	480	3/5

Почему бы не поместить в таблицу оба рейтинга? В результате у нас будет более точная информация.



Так и есть. Вложенные таблицы в XHTML создаются элементарно. Нужно лишь поместить внутрь элемента **<td>** еще один элемент **<table>**. Как это сделать? Вы создаете простую таблицу, в которой будут отражены оба мнения по рейтингу кафе (Тони и Тесс), и, убедившись, что она выводится правильно, помещаете ее внутрь ячейки таблицы, которая на данный момент содержит рейтинг Тони: 4/5. Давайте попробуем...

```
<tr>
   <td rowspan="2">Трут-ор-Консекуэнсес, штат Нью-Мексико</td>
   <td class="center">9 августа</td>
   <td class="center">93</td>
   <td rowspan="2" class="right">4242 фута</td>
   <td rowspan="2" class="right">7289</td>
   <td class="center">5/5</td>
</tr>
<tr>
   <td class="center">27 августа</td>
   <td class="center">98</td>
   <td>
     ~~4/5~~
     <table>
        <tr>
           <th>Тесс</th>
           <td>5/5</td>
        </tr>
        <tr>
           <th>Тони</th>
           <td>4/5</td>
        </tr>
     </table>
   </td>
</tr>
```

Сначала удалите старый рейтинг...

...и поместите в это место таблицу. В ней будут храниться оценки обеда по пятибалльной шкале: одна оценка Тесс, другая — Тони. Мы используем их имена в качестве заголовков таблицы, а рейтинги — в качестве ячеек.

Тест для Вложенной таблицы

Итак, вперед! Введите данные для новой таблицы. При вводе часто случаются опечатки, поэтому проверьте документ, а затем обновите страницу. Вы увидите новую вложенную таблицу.

На скутере по территории США

file:///chapter13/journal/journal.html

На скутере по США

Дневник моих поездок на моем собственном скутере по территории США!

20 августа, 2005



Итак, я уже проехал 1200 миль и побывал в некоторых интересных местах:

Город	Дата	Температура	Высота	Население	Рейтинг придорожных кафе
Вала-Вала, штат Вашингтон	15 июня	75	1204 фута	29686	4/5
Мэджик-Сити, штат Айдахо	25 июня	74	5312 футов	50	3/5
Баунтифул, штат Юта	10 июля	91	4226 футов	41173	4/5
Лэст Чанс, штат Колорадо	23 июля	102	4780 футов	265	3/5
Трут-ор-Консекуэнсес, штат Нью-Мексико	9 августа	93	4242 фута	7289	5/5
	27 августа	98			Тесс 5/5 Томи 4/5
Уай, штат Аризона	18 августа	104	860 футов	480	3/5

Города, которые я посетил во время путешествия на скутере по США

14 июля, 2005

Я видел парочку знаков в стиле Burma Shave на обочине дороги:

Ого, выглядит здорово. Единственное замечание: цветной фон для заголовков вложенной таблицы – это лишнее. Давайте выделим имена полужирным шрифтом, но уберем цветной фон.



МОЗГОВОЙ ШТУРМ
ПОВЫШЕННАЯ СЛОЖНОСТЬ

Пришло время вспомнить все, что изучено к этому моменту. Вам нужно поменять цвет фона только для заголовков вложенной таблицы (Тони и Тесс) и при этом не затронуть цвет фона заголовков главной таблицы. Как? Нужно написать селектор, который выбирает только заголовки вложенной таблицы.

Город	Дата	Температура	Высота	Население	Рейтинг придорожных кафе
Вала-Вала, штат Вашингтон	15 июня	75	1204 фута	29686	4/5
Мэджик-Сити, штат Айдахо	25 июня	74	5312 футов	50	3/5
Баунтифул, штат Юта	10 июля	91	4226 футов	41173	4/5
Лэст Чанс, штат Колорадо	23 июля	102	4780 футов	265	3/5
Трут-ор-Консекуэнсес, штат Нью-Мексико	9 августа	93	4242 фута	7289	5/5
	27 августа	98			Тесс 5/5 Тони 4/5
Уай, штат Аризона	18 августа	104	860 футов	480	3/5

Нужно изменить цвет фона ячеек с заголовками для вложенной таблицы, сделав его белым.

Напишите селектор, выбирающий только заголовки вложенной таблицы

```
_____ {
    background-color: white;
}
```



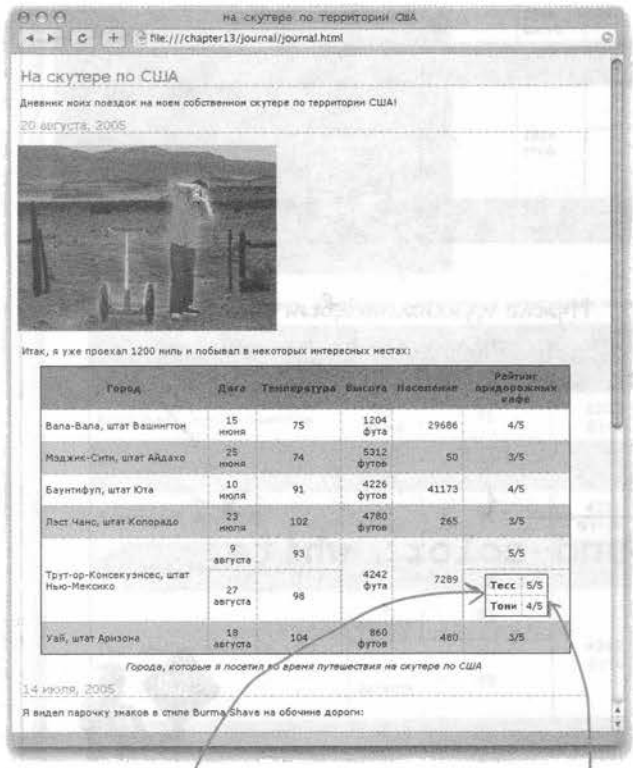
Стоп! Не переворачивайте страницу, пока не выполните это упражнение.

Переопределение CSS для заголовков вложенной таблицы

Вы можете выбрать только элементы **<th>** вложенной таблицы, используя селектор потомка. Добавьте новое правило в CSS-код, в котором используется селектор **table table th**, чтобы изменить цвет фона для заголовков вложенной таблицы:

```
table table th {
    background-color: white;
}
```

Теперь сохраните изменения в файле `journal.css` и обновите страницу.



Город	Дата	Температура	Высота	Население	Рейтинг придорожных кафе
Вала-Вала, штат Вашингтон	15 июня	75	1204 фута	29686	4/5
Мэджик-Сити, штат Айдахо	25 июня	74	5312 футов	50	3/5
Баунтифул, штат Юта	10 июля	91	4226 футов	41173	4/5
Ласт Чанс, штат Колорадо	23 июля	102	4780 футов	265	3/5
Трут-ор-Консекуэнсес, штат Нью-Мексико	9 августа	93	4242 фута	7289	5/5
	27 августа	98			Тесс 5/5, Тони 4/5
Уай, штат Аризона	18 августа	104	860 футов	460	3/5

Теперь элементы «th» во вложенной таблице имеют белый фон.

Но обратите внимание, что для них все еще задан полужирный шрифт, так как мы не переопределяли это свойство.

Часто Задаваемые Вопросы

В: Чтобы выполнить упражнение «Мозговой штурм» повышенной сложности, я создал класс nestedtable и сделал его членом каждый заголовок вложенной таблицы. Затем я создал такое правило:

```
.nestedtable {
    background-color: white;
}
```

Можно ли выполнить это упражнение и таким образом?

О: Существует множество способов для выполнения подобных задач в CSS, и, конечно, ваше решение также является эффективным и верным. Хотелось бы лишь подчеркнуть, что, используя селектор потомка, мы смогли обойтись без каких-либо изменений в XHTML. А если бы Тони и Тесс продолжали добавлять разные оценки обедов и в других городах? Тогда для каждой оценки вам пришлось бы добавлять в класс соответствующий элемент <th>. При использовании же нашего способа стилизация происходит автоматически.

МОЗГОВОЙ ШТУРМ

Нужно, чтобы фоновые цвета строк таблицы с рейтингом Тони и Тесс были разными, например, голубым и розовым соответственно. Можете ли вы придумать несколько способов реализовать это?

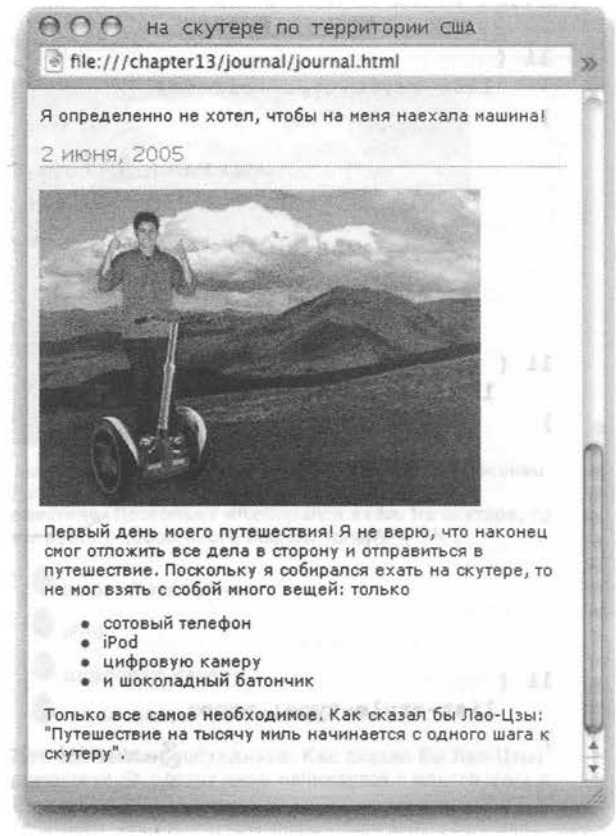
Доведем сайт Тони до совершенства

Страница Тони выглядит действительно здорово, но есть еще одна область, стилизации которой мы пока не уделили внимания, — список вещей, которые он приготовил с собой в поездку. Вы найдете этот список в записи от 2 июня:

Мы рассматриваем только отрывок XHTML-кода, содержащий запись от 2 июня.

```
.
.
.
  <h2>2 июня, 2005</h2>

  <p>
    <img src="images/segway1.jpg"
         alt="The first day of the trip" />
  </p>

  <p>
    Первый день моего путешествия! Я не
    верю, что наконец смог отложить все
    дела в сторону и отправиться
    в путешествие. Поскольку я собирался
    ехать на скутере, то не мог взять
    с собой много вещей: только
  </p>
  <ul>
    <li>сотовый телефон</li>
    <li>iPod</li>
    <li>цифровую камеру</li>
    <li>шоколадный батончик</li>
  </ul>
  <p>
    Только все самое необходимое. Как
    сказал бы Лао-Цзы: <q>Путешествие
    на тысячу миль начинается с одного
    шага к скутеру</q>
  </p>
</body>
</html>
```

Это нижняя часть дневника Тони из файла journal.html. Помните этот список в первой записи дневника?



Вот так этот список выглядит на данный момент.

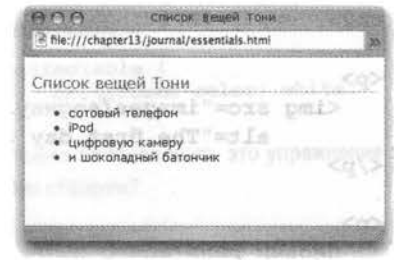
Стилизация списка

Вы, вероятно, уже поняли, что, зная основы CSS (шрифт, текст, цвет и другие свойства), можно стилизовать что угодно. Это распространяется и на списки. На самом деле есть всего несколько свойств, которые применяются исключительно для списков. Основное свойство для списков называется **list-style-type** и позволяет определять вид используемых маркеров. Рассмотрим несколько способов задания маркеров.

Здесь мы определяем стиль для элементов <li>. Можно также задавать его для элементов <ul>, и он унаследуется элементами <li>.

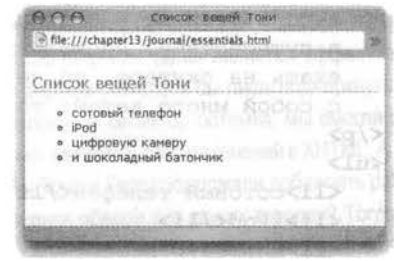
```
li {
    list-style-type: disc;
}
```

Disc – тип маркеров, используемый по умолчанию.



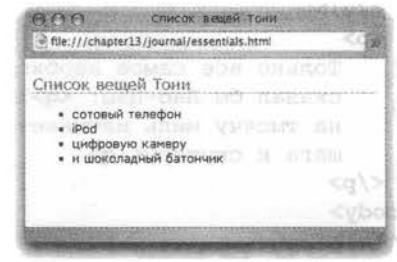

```
li {
    list-style-type: circle;
}
```

Значение circle свойства list-style-type устанавливает простой круглый маркер.



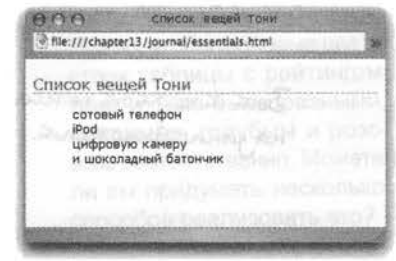

```
li {
    list-style-type: square;
}
```

Значение square определяет квадратный маркер.




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

Значение none вообще удаляет маркеры



Если нужен маркер особой формы?



Неужели вы думаете, что Тони будет довольствоваться обычным маркером и не захочет придумать свой собственный? К счастью, в CSS есть свойство **list-style-image**, с помощью которого вы можете использовать изображение в качестве маркера списка. Попробуем сделать это для списка Тони.

Это свойство list-style-image, в качестве значения которого мы задаем URL.

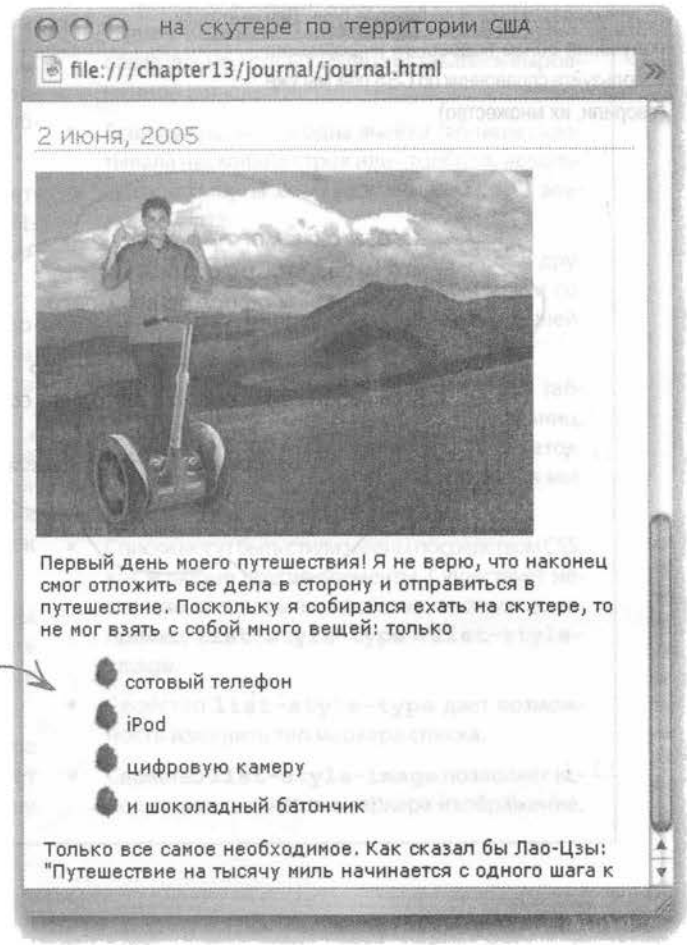
Рисунок backpack.gif представляет собой маленькое изображение рюкзака. Кажется, нам это подходит, не так ли?

```
li {
    list-style-image: url(images/backpack.gif);
    padding-top: 5px;
    margin-left: 20px;
}
```

Чтобы добавить свободное пространство слева от элементов списка, мы используем поле. Кроме того, задаем небольшой отступ, чтобы в верхней части списка тоже было свободное пространство.

И последний тест...

Это будет последнее изменение сайта Тони. Добавьте новое правило стиля для элементов списка в CSS-код, а затем обновите страницу.



Это список, в котором на месте маркеров стоит изображение. Кроме того, здесь есть дополнительное поле и отступ.

Часто Задаваемые Вопросы

В: Как насчет нумерованных списков? Что можно сделать, чтобы изменить их стиль?

О: Нумерованные и маркированные списки стилизуются одинаково. Конечно же, в нумерованных списках используются не маркеры, а числа или буквы. Благодаря CSS вы можете задавать в качестве маркеров для нумерованных списков цифры десятичной системы счисления, римские цифры или буквы алфавита (например, a, b, c). Для этого применяется свойство list-style-type. Для получения более подробной информации используйте справочник по CSS (как мы уже говорили, их множество).

В: Как можно контролировать способ переноса текста на новую строку в списке? Я имею в виду, как можно проверить, переносится текст под маркер или правее, под начало предыдущей строки?

О: Для этого предназначено свойство list-style-position. Если вы зададите для него значение inside, то после переноса на новую строку текст будет расположен под маркером. Если укажете значение outside, то текст будет выровнен по левому краю предыдущей строки.

В: Вы уверены, что это именно так? Кажется, должно быть наоборот.

О: Да, вот что *на самом деле* говорят эти значения: если вы устанавливаете значение inside свойства line-style-position, то маркеры как бы располагаются *внутри* элементов вашего списка, а переносимый на новую строку текст отображается под ними. Если используете значение outside, то маркеры как бы находятся *вне* элементов вашего списка и переносимый на новую строку текст просто отображается под текстом предыдущей строки.



ПОВТОРИМ выученное



- XHTML-таблицы используются для структуризации табличных данных.
- Используйте все следующие элементы HTML: **<table>**, **<tr>**, **<th>** и **<td>**, чтобы создать таблицу.
- Элемент **<table>** задает таблицу и содержит ее остальные элементы.
- Таблицы определяются построчно с помощью элемента **<tr>**.
- Каждая строка состоит из одной или нескольких ячеек с данными, задаваемыми элементом **<td>**.
- Используйте элемент **<th>** для тех ячеек, которые играют роль заголовков строк или столбцов.
- Каждой строке таблицы ставится в соответствие строка **<tr>...</tr>** в вашем HTML-коде, а каждому столбцу — **<td>...</td>** внутри строки.
- Вы можете представить пользователям дополнительную информацию о своих таблицах, используя атрибут **summary** элемента **<table>** и элемент **<caption>**.
- У ячеек могут быть отступы и границы. Кроме того, можно задать промежутки между ячейками (расстояния между границами соседних ячеек, а также между границами крайних ячеек и границей всей таблицы).
- Так же, как вы задавали отступы, границы и поля для других элементов, можете задавать отступы, границы и промежутки для ячеек в CSS.
- Свойство **border-collapse** — специальное CSS-свойство для таблиц, которое позволяет объединить границы соседних ячеек в одну. В результате таблица будет лучше смотреться и читаться.
- Вы можете изменить выравнивание данных в ячейках таблицы, используя CSS-свойства **text-align** и **vertical-align**.
- Можно определить цвет таблицы благодаря свойству **background-color**. Фоновый цвет может быть задан для всей таблицы целиком, для отдельной строки или ячейки с данными.
- Если для какой-то ячейки у вас нет данных, оставляйте элемент **<td>** без содержимого. Использовать элемент **<td>...</td>** нужно обязательно, чтобы таблица оставалась выровненной и ячейки не сместились.
- Если нужно, чтобы одна ячейка таблицы охватывала несколько строк или столбцов, используйте атрибуты **rowspan** или **colspan** элемента **<td>**.
- Вы можете вложить одну таблицу внутрь другой, поместив элемент **<table>** вместе со всем его содержимым внутрь ячейки внешней таблицы.
- Таблицы нужно использовать лишь для табличных данных, а не для разметки веб-страниц. Для создания мультиколоночных разметок применяйте CSS-позиционирование, как мы рассказывали в главе 12.
- Списки могут быть стилизованы посредством CSS, как и любые другие элементы. Существует несколько специальных свойств для списков, например, **list-style-type** и **list-style-image**.
- Свойство **list-style-type** дает возможность изменить тип маркера списка.
- Свойство **list-style-image** позволяет использовать в качестве маркера изображение.



Решение упражнения

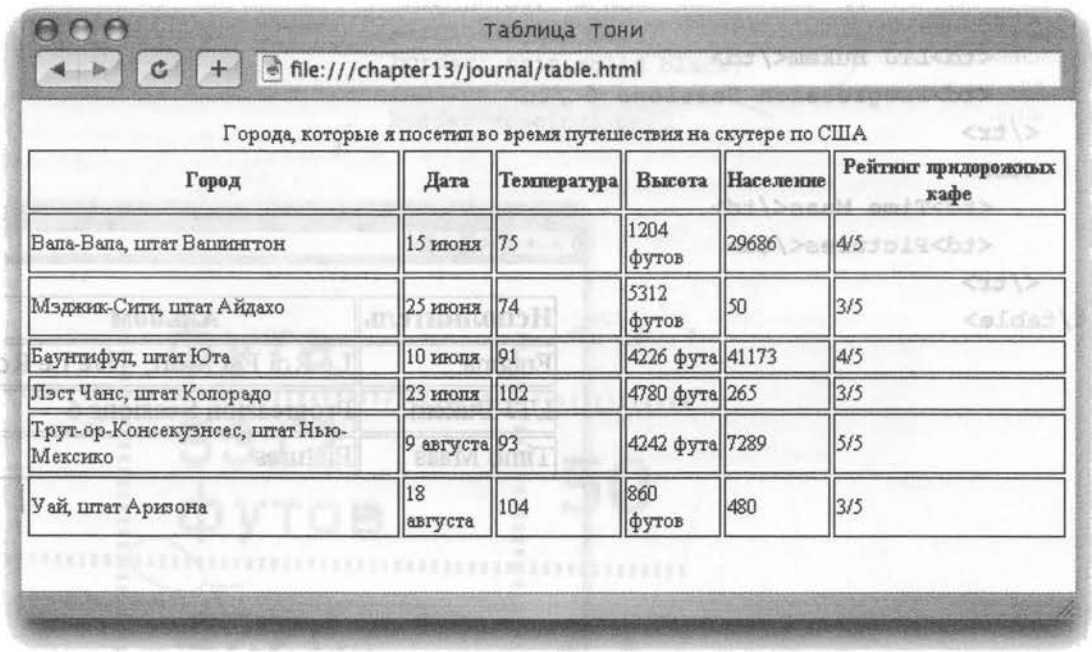
Сначала напечатайте XHTML-код для тестирования таблицы Тони, приведенный на предыдущей странице. Хотя печатать его будет скучно, это поможет вам усвоить и запомнить структуру тегов **<table>**, **<tr>**, **<th>** и **<td>**. Когда закончите, быстро протестируйте код, а затем добавьте оставшиеся данные из таблицы Тони. И снова протестируйте.

```
<!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="ru" xml:lang="ru">
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />
   <style type="text/css">
        td, th {border: 1px solid black;}
   </style>
   <title>Тестирование таблицы Тони</title>
</head>
<body>
   <table>
      <tr>
         <th>Город</th>
         <th>Дата</th>
         <th>Температура</th>
         <th>Высота</th>
         <th>Население</th>
         <th>Рейтинг придорожных кафе</th>
      </tr>
      <tr>
         <td>Вала-Вала, штат Вашингтон</td>
         <td>15 июня</td>
         <td>75</td>
         <td>1204 фута</td>
         <td>29 686</td>
         <td>4/5</td>
      </tr>
      <tr>
         <td>Мэджик-Сити, штат Айдахо</td>
         <td>25 июня</td>
         <td>74</td>
         <td>5312 футов</td>
         <td>50</td>
         <td>3/5</td>
      </tr>
      <tr>
         <td>Баунтифул, штат Юта</td>
         <td>10 июля</td>
         <td>91</td>
         <td>4226 футов</td>
         <td>41173</td>
         <td>4/5</td>
      </tr>
      <tr>
         <td>Лэст Чанс, штат Колорадо</td>
         <td>23 июля</td>
         <td>102</td>
         <td>4780 футов</td>
         <td>265</td>
         <td>3/5</td>
      </tr>
      <tr>
         <td>Трут-ор-Консекуэнсес, штат Нью-Мексико</td>
         <td>9 августа</td>
```



Решение упражнения

```
        <td>93</td>
        <td>4242 фута</td>
        <td>7289</td>
        <td>5/5</td>
      </tr>
      <tr>
        <td>Уай, штат Аризона</td>
        <td>18 августа</td>
        <td>104</td>
        <td>860 футов</td>
        <td>480</td>
        <td>3/5</td>
      </tr>
    </table>
</body>
</html>
```



Города, которые я посетил во время путешествия на скутере по США

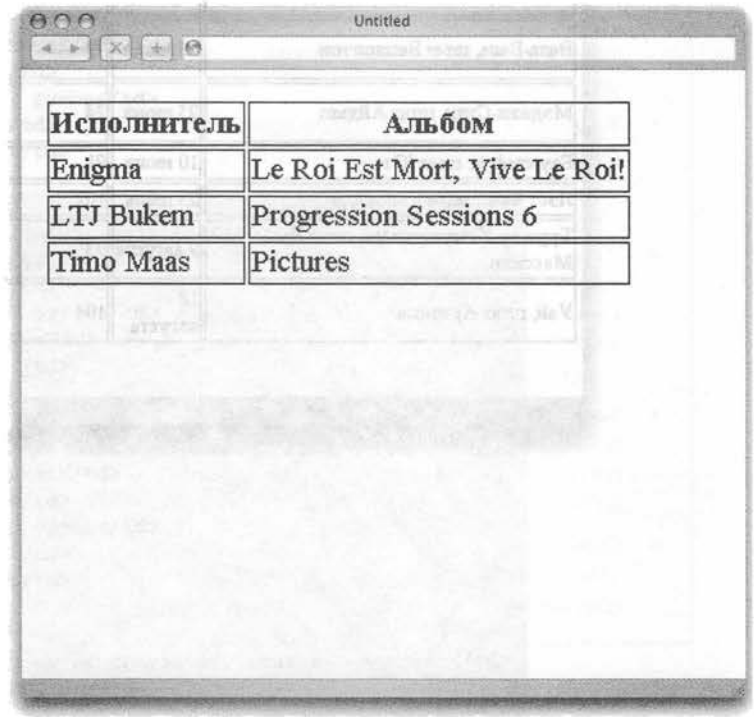
Город	Дата	Температура	Высота	Население	Рейтинг придорожных кафе
Вала-Вала, штат Вашингтон	15 июня	75	1204 футов	29686	4/5
Мэджик-Сити, штат Айдахо	25 июня	74	5312 футов	50	3/5
Баунтифул, штат Юта	10 июля	91	4226 фута	41173	4/5
Лэст Чанс, штат Колорадо	23 июля	102	4780 фута	265	3/5
Трут-ор-Консекуэнсес, штат Нью-Мексико	9 августа	93	4242 фута	7289	5/5
Уай, штат Аризона	18 августа	104	860 футов	480	3/5



Поработайте браузером

Слева вы найдете XHTML-код для таблицы. Ваша задача — поставить себя на место браузера, отображающего эту страницу. Вот решение.

```
<table>
   <tr>
      <th>Исполнитель</th>
      <th>Альбом</th>
   </tr>
   <tr>
      <td>Enigma</td>
      <td>Le Roi Est Mort, Vive Le Roi!</td>
   </tr>
   <tr>
      <td>LTJ Bukem</td>
      <td>Progression Sessions 6</td>
   </tr>
   <tr>
      <td>Timo Maas</td>
      <td>Pictures</td>
   </tr>
</table>
```

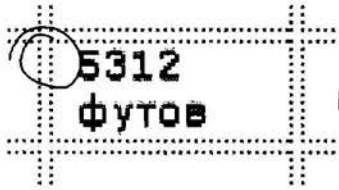


Возьми в руку карандаш

Решение

Дублированные точечные линии в таблице Тони очень сильно бросаются в глаза. Было бы лучше, если бы вокруг каждой ячейки была одинарная точечная линия, что не отвлекало бы внимания от содержимого таблицы. Вы можете придумать способ сделать это, используя те знания о стилях, которые имеете на данный момент?

Можно задать значение 0 для свойства **border-spacing**, что позволит убрать промежутки между границами.



Мы можем использовать свойство border-spacing, чтобы установить ширину промежутка между границами, равную 0. Тогда две линии будут расположены вплотную друг к другу.

```
table {
    margin-left: 20px;
    margin-right: 20px;
    border: thin solid black;
    caption-side: bottom;
    border-spacing: 0px;
}
```



Это лучше, но мы все же имеем две линии, а не одну, хотя они и расположены вплотную друг к другу. В результате ширина границ удваивается. Нам бы больше понравилось, если бы между ячейками была одинарная граница. Не так ли?

Возьми в руку карандаш

Решение

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

Это два класса. Первый — для выравнивания по центру, второй — для выравнивания по левому краю.

```
<table summary="В этой таблице хранятся данные о тех городах, которые
я посетил во время путешествия. Я указал в ней дату посещения города,
температуру воздуха в этот день, высоту над уровнем моря и количество
населения этого города. Я также включил в таблицу рейтинг придорожных
кафе, в которых обедал, по шкале от 1 до 5.">
 <caption>Города, которые я посетил во время путешествия на скутере по США</caption>
   <tr>
      <th>Город</th>
      <th>Дата</th>
      <th>Температура</th>
      <th>Высота</th>
      <th>Население</th>
      <th>Рейтинг придорожных кафе</th>
   </tr>
   <tr>
      <td>Вала-Вала, штат Вашингтон</td>
      <td class="center">15 июня</td>
      <td class="center">75</td>
      <td class="right">1204 фута</td>
      <td class="right">29 686</td>
      <td class="center">4/5</td>
   </tr>
   <tr>
      <td>Мэджик-Сити, штат Айдахо</td>
      <td class="center">25 июня</td>
      <td class="center">74</td>
      <td class="right">5312 футов</td>
      <td class="right">50</td>
      <td class="center">3/5</td>
   </tr>
   .
   .
   .
</table>
```

Здесь вы просто добавляете каждый элемент `<td>` в соответствующий класс!



Решение упражнений

Чтобы задать разные фоновые цвета для строк таблицы, вы просто добавляете атрибут **class="cellcolor"** в открывающий тег **<tr>** для соответствующих строк.

```
<tr class="cellcolor">
    <td>Мэджик-Сити, штат Айдахо</td>
    ...
</tr>
```



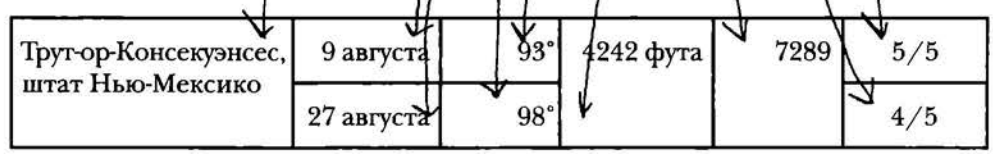


Решение упражнений

Кто и что делает?

Для того чтобы удостовериться, что вы хорошо все усвоили, мы попросим вас нарисовать стрелку от каждого элемента `<td>` к соответствующей ему ячейке таблицы. Проверьте свои ответы, прежде чем продолжать двигаться вперед.

```
<tr>
   <td rowspan="2">Трут-ор-Консекуэнсес, штат Нью-Мексико</td>
   <td class="center">9 августа</td>
   <td class="center">93</td>
   <td rowspan="2" class="right">4242 фута</td>
   <td rowspan="2" class="right">7289</td>
   <td class="center">5/5</td>
</tr>
<tr>

   <td class="center">27 августа</td>
   <td class="center">98</td>


   <td class="center">4/5</td>
</tr>
```





Решение упражнений



МОЗГОВОЙ ШТУРМ
ПОВЫШЕННАЯ СЛОЖНОСТЬ

Пришло время вспомнить все, что изучено к этому моменту. Вам нужно поменять цвет фона только для заголовков вложенной таблицы (Тони и Тесс) и при этом не затронуть цвет фона заголовков главной таблицы. Как? Нужно написать селектор, который выбирает только заголовки вложенной таблицы.

Мы хотим изменить цвет фона ячеек с заголовками для вложенной таблицы, сделав его белым. Вот как вы можете это сделать.

Город	Дата	Температура	Высота	Население	Рейтинг придорожных кафе
Вала-Вала, штат Вашингтон	15 июня	75	1204 фута	29686	4/5
Маджик-Сити, штат Айдахо	25 июня	74	5312 футов	50	3/5
Баунтифул, штат Юта	10 июля	91	4226 футов	41173	4/5
Лэст Чанс, штат Колорадо	23 июля	102	4780 футов	265	3/5
Трут-ор-Консекуэнсес, штат Нью-Мексико	9 августа	93	4242 фута	7289	5/5
	27 августа	98			Тесс 5/5 Тони 4/5
Уай, штат Аризона	18 августа	104	860 футов	480	3/5

(1) Начните с выбора внешней таблицы

(2) Затем выберите внутреннюю страницу.

(3) Теперь выберите заголовок таблицы

(1) (2) (3)

```
table table th {
    background-color: white;
}
```

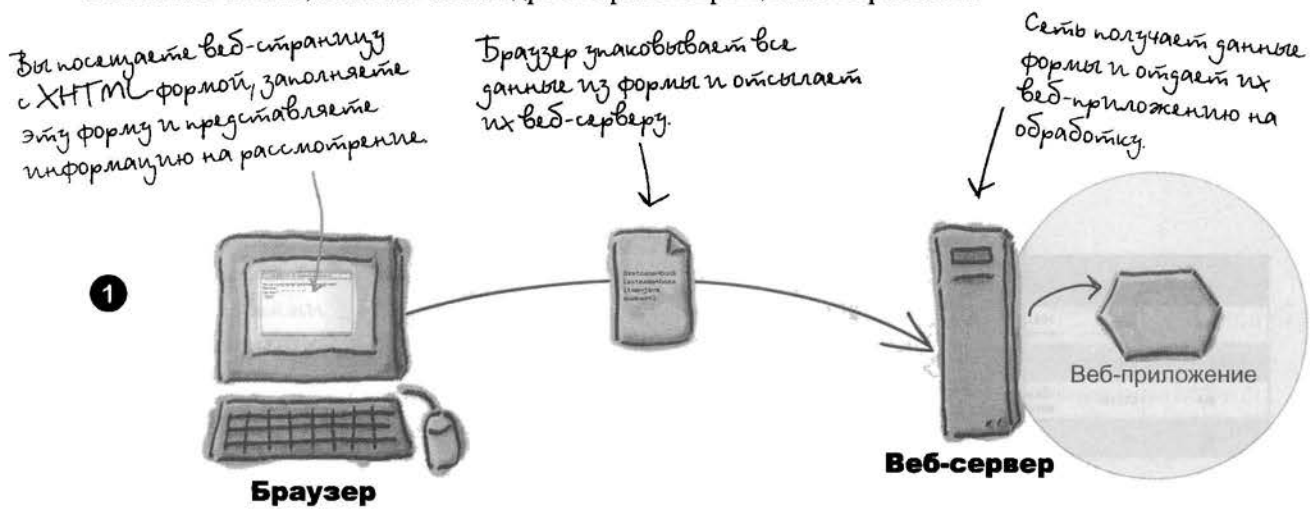
Переход на интерактивный режим

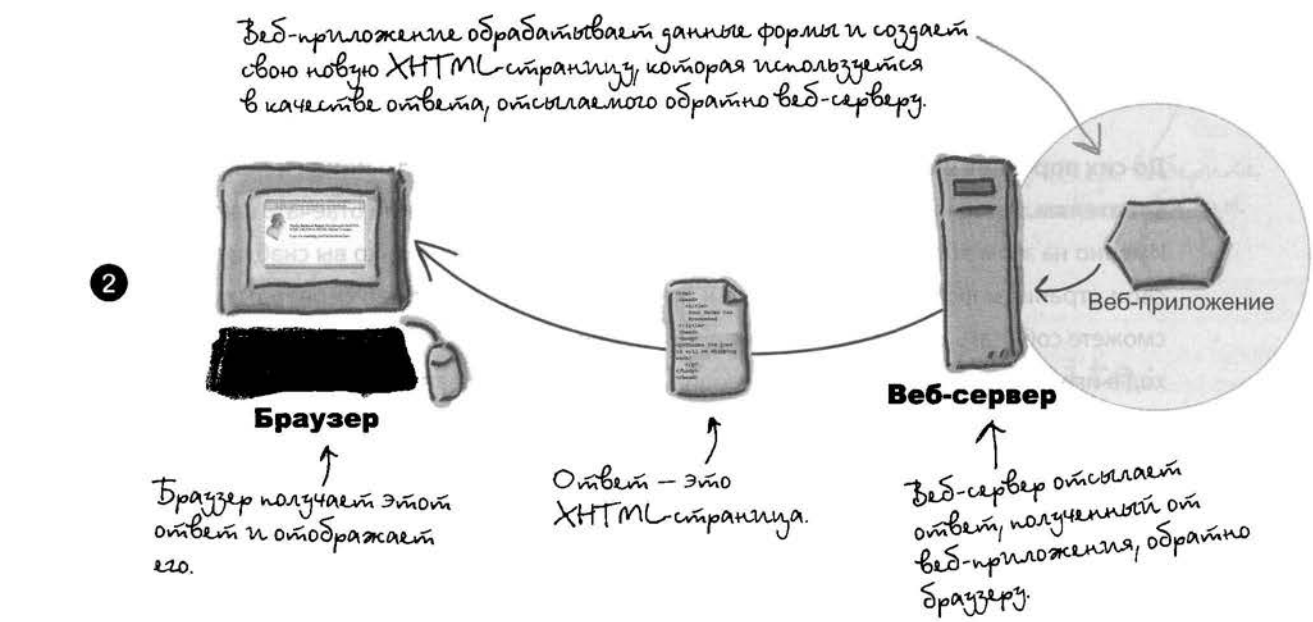


До сих пор ваше веб-общение было односторонним: от веб-страницы к пользователям. Разве не было бы здорово, если бы пользователи смогли отвечать вам? Именно на этом этапе вступают в действие формы XHTML. Как только вы снабдите свои страницы формами (прибегнув к определенной помощи веб-сервера), вы сможете собирать отзывы клиентов, принимать по Сети заказы, делать следующий ход в игре в режиме онлайн или проводить интернет-голосования. В этой главе вы познакомитесь с целой группой XHTML-элементов, предназначенных для создания веб-форм. Вы также узнаете кое-что о том, что происходит на сервере для поддержки форм и как сделать сами формы стильными.

Как работают формы

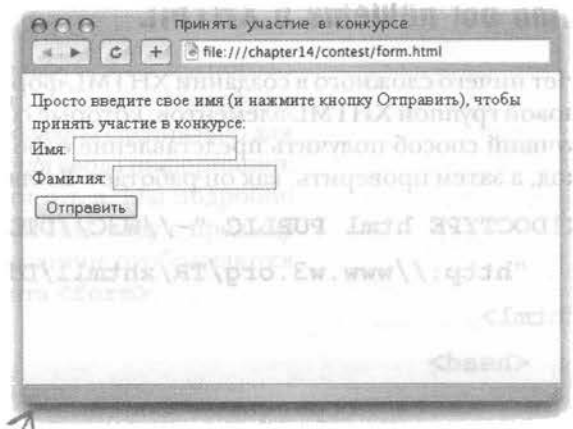
Если вы пользуетесь Интернетом, то точно знаете, как выглядят формы. Но вы, скорее всего, никогда не задумывались о том, что им приходится делать с XHTML. По своей сути, форма — это веб-страница с областями ввода, в которых вы можете печатать свою информацию. Когда форма *представляется на рассмотрение*, информация из нее упаковывается и отсылается веб-серверу для обработки веб-приложением. Что вы получаете после обработки? Другую веб-страницу в качестве ответа, конечно же. Подробно рассмотрим, как это работает.





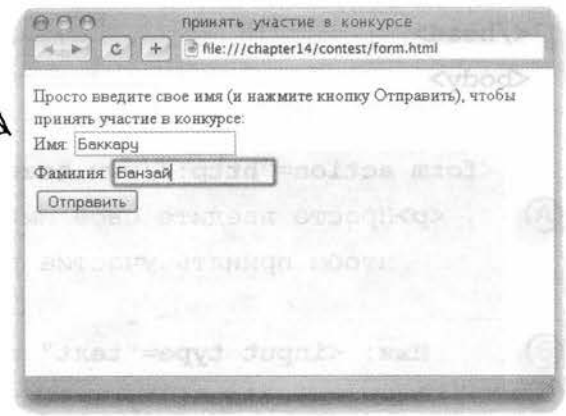
Как формы работают в браузере

Для браузера форма – всего лишь фрагмент XHTML-кода на странице. Вы увидите, что можете легко создавать формы, всего лишь добавляя несколько новых элементов. Рассмотрим, как работает форма с точки зрения браузера.



Браузер загружает страницу

Браузер, как обычно, загружает XHTML-код для страницы, а когда вдруг встречает элементы формы, создает на странице элементы управления, что позволяет вам вводить разнообразные типы данных. *Элемент управления* – это в основном то, что позволяет вам вводить данные, например кнопка, окно для ввода текстовых данных, раскрывающийся список.

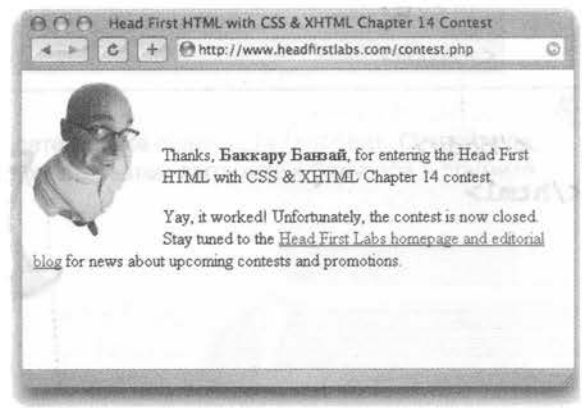


Вы вводите данные

Вы используете элементы управления для ввода данных. В зависимости от типа элемента, ввод происходит по-разному. Вы можете ввести единичную строку текста в текстовый элемент управления или выбрать одно из нескольких положений в элементе управления типа «переключатель». Очень скоро мы рассмотрим разные виды элементов управления.

Вы представляете форму на рассмотрение

Вы представляете форму на рассмотрение, нажав кнопку Submit (Отправить). Этим вы даете браузеру сигнал, что ему нужно упаковать все данные и отослать их серверу.



Сервер реагирует

Как только сервер получает данные формы, он отправляет их соответствующему веб-приложению на обработку. В качестве результата этой обработки выдается новая XHTML-страница, которая возвращается браузеру. Поскольку это обычный XHTML-код, то браузер отображает его для вас.

Что Вы пишете в XHTML

Нет ничего сложного в создании XHTML-форм. В этой главе вы встретитесь с абсолютно новой группой XHTML-элементов, которые сообща работают для создания форм. Самый лучший способ получить представление о формах – посмотреть на небольшой XHTML-код, а затем проверить, как он работает. Взгляните на эту форму:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
   <head>
      <meta http-equiv="Content-Type"
            content="text/html; charset=ISO-8859-1" />
      <title>Enter the Contest</title>
   </head>
   <body>

      <form action="http://www.headfirstlabs.com/contest.php" method="POST">
Ⓐ        <p>Просто введите свое имя (и нажмите кнопку Отправить),
            чтобы принять участие в конкурсе: <br />

Ⓑ        Имя: <input type="text" name="firstname" value="" /> <br />
Ⓒ        Фамилия: <input type="text" name="lastname" value="" /> <br />
Ⓓ        <input type="submit" />

         </p>
      </form>

   </body>
</html>
```

Все это вам уже хорошо знакомо.

Это форма.

У нас есть сам элемент <form>.

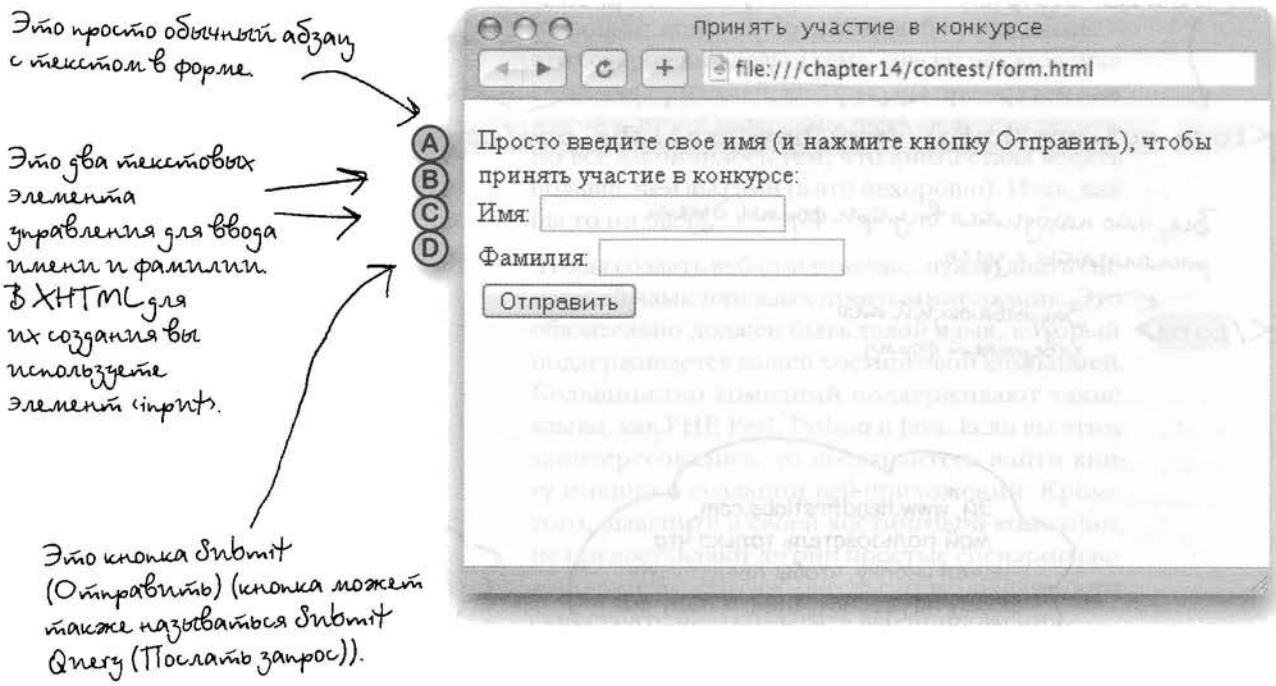
И набор вложенных в него элементов.



Пока просто внимательно рассмотрите форму и ее содержимое. На протяжении этой главы мы разберемся со всеми деталями.

Что создает браузер

Сюрприз! Для создания формы применяется элемент **<form>**. Сейчас мы поговорим немного о блочных элементах, которые могут входить в элемент **<form>**, но есть также набор абсолютно новых элементов, которые созданы специально для форм. Каждый из них предоставляет различные способы для ввода информации: поля для ввода текста, флажки, раскрывающиеся списки и т. д. Мы подробно рассмотрим все эти элементы, но сначала вернемся к XHTML-коду с предыдущей страницы и посмотрим, как на приведенной ниже странице отображаются элементы и содержимое, расположенные внутри элемента **<form>**.





Упражнение

Форму для конкурса вы найдете в папке `chapter14/contest`. Откройте ее, еще раз хорошенько рассмотрите, затем загрузите в браузере и примите участие в конкурсе.

Как работает элемент <form>

Посмотрим на **<form>** не только как на элемент, содержащий все элементы, образующие форму, но и как на элемент, который говорит браузеру, куда отсылать данные вашей формы после того, как вы представите ее на рассмотрение (и какой метод браузер должен использовать для отсылки).

Это открывающий тег. Все, что имеет отношение к форме, располагается внутри.

Атрибут action содержит URL веб-сервера...

...и название веб-приложения, которое будет обрабатывать данные.

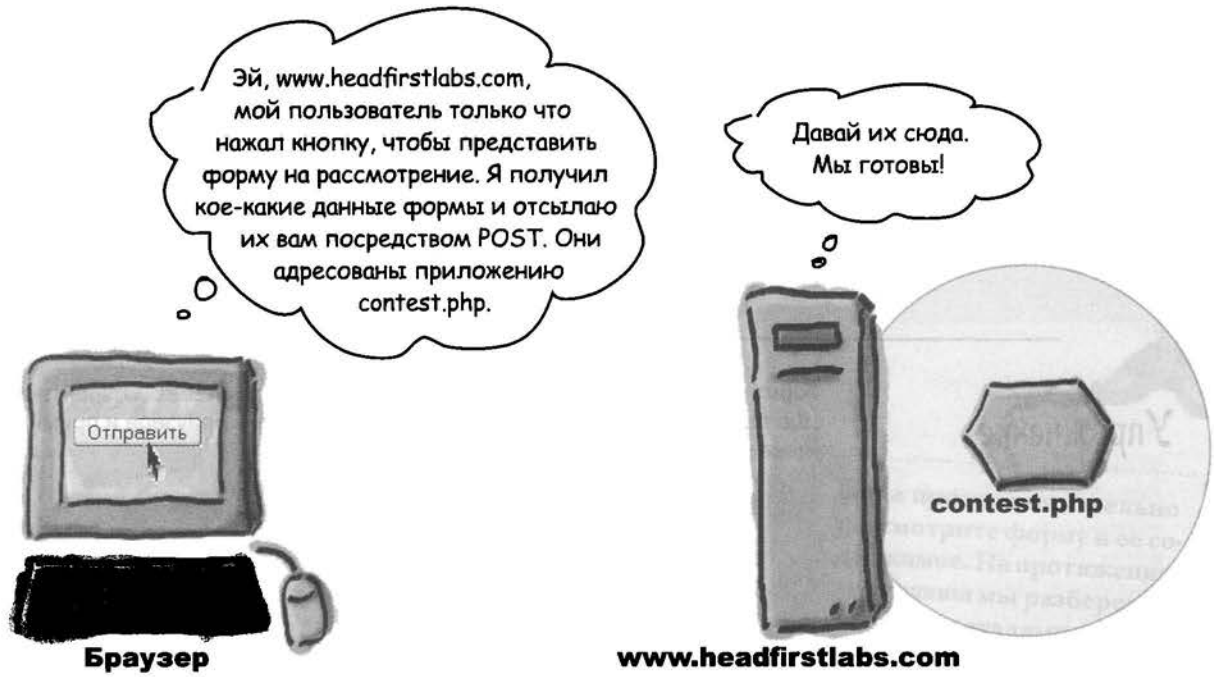
Атрибут method указывает, каким образом данные формы будут отправлены на сервер. Мы будем использовать наиболее распространенный: POST. Чуть позже мы поговорим и о других способах отсылки данных, а также о преимуществах и недостатках применения POST.

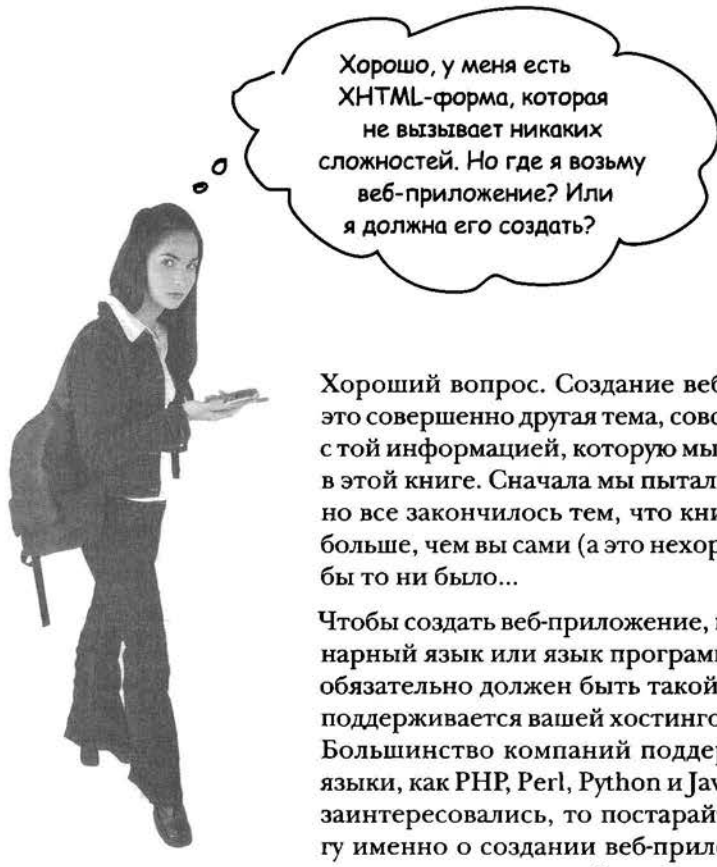
```
<form action="http://www.headfirstlabs.com/contest.php" method="POST">
```

Все, что находится внутри формы, будет располагаться здесь.

```
</form>
```

Закрывающий тег завершает форму.





Хороший вопрос. Создание веб-приложений – это совершенно другая тема, совсем не связанная с той информацией, которую мы предлагаем вам в этой книге. Сначала мы пытались ее осветить, но все закончилось тем, что книга стала весить больше, чем вы сами (а это нехорошо). Итак, как бы то ни было...

Чтобы создать веб-приложение, нужно знать сценарный язык или язык программирования. Это обязательно должен быть такой язык, который поддерживается вашей хостинговой компанией. Большинство компаний поддерживают такие языки, как PHP, Perl, Python и Java. Если вы этим заинтересовались, то постарайтесь найти книгу именно о создании веб-приложений. Кроме того, выясните в своей хостинговой компании, не предоставляют ли они простые сценарии своим клиентам. Это избавит вас от необходимости самостоятельно создавать веб-приложения.

Что касается этой главы, мы уже разработали все веб-приложения, которые вам понадобятся. Вам нужно лишь правильно указать эти приложения в URL-адресе в атрибуте **`action`** элемента **`<form>`**.

Что может входить в форму

Вы можете поместить в форму практически любой блочный элемент, но это не то, что важно для нас на данный момент. Нас интересуют *специальные элементы формы, которые создают элементы управления* в браузере. Мы приводим здесь краткую информацию о наиболее часто используемых элементах формы. Начнем с элемента формы **<input>**, который имеет много функций.

text input

Элемент <input> типа text используется для ввода одной строки текста. Благодаря дополнительным атрибутам вы можете задать максимальное количество символов и ширину управляющего элемента.

Имя: Баккару

Элемент <input> со значением text атрибута type создает на странице браузера элемент управления для ввода текста из одной строки.

Используйте атрибут type, чтобы указать, что вам нужен ввод информации типа text.

Для большинства элементов требуется название, используемое веб-приложениями. Чуть позже мы рассмотрим, как это работает.

```
<input type="text" name="fullname" />
```

Элемент <input> пустой, так что его нужно заканчивать символами « />».

Обратите внимание, что в обоих случаях используется один и тот же XHTML-элемент, но с различными значениями атрибута type.

submit input

Элемент <input> типа submit создает кнопку, нажав которую вы можете представить форму на рассмотрение. Когда вы нажимаете эту кнопку, браузер отсылает форму веб-приложению на обработку.

Отправить

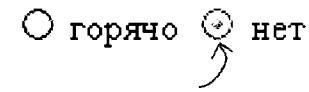
По умолчанию кнопка называется Submit (Отправить) или Submit Query (Отправить запрос), хотя это название можно изменить (как это сделать, мы покажем чуть позже).

```
<input type="submit" />
```

Для кнопки Submit (Отправить) укажите значение submit для атрибута type элемента <input>.

radio input

Элемент <input> типа radio создает переключатели, которые являются взаимоисключающими. Это похоже на старые переключатели в машинах: если вы выбираете одно положение, то все остальные становятся неактивными.



Элемент управления radio позволяет выбрать только одно положение из предлагаемых.

Указывайте элемент <input> типа radio для каждого положения переключателя.

Все положения, имеющие отношение к данному переключателю, должны иметь одно имя управляющего элемента...

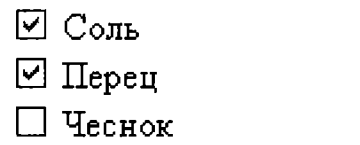
...но каждое положение должно иметь свое значение.

```
<input type="radio" name="hotornot" value="горячо" />
<input type="radio" name="hotornot" value="нет" />
```

То же самое здесь: мы все еще используем элемент <input>, просто с другим значением атрибута type.

checkbox input

Элемент <input> типа checkbox создает флажки, которые могут быть либо установленными, либо снятыми. Вы можете использовать сразу несколько флажков, при этом можно установить любое их количество.



Можно не установить ни одного флажка или установить любое их количество. Это отличает флажки от переключателей.

Как и для переключателей, вы используете по одному элементу <input> со значением checkbox атрибута type для каждого положения.

Взаимосвязанные флажки также используют одно и то же имя.

Каждый флажок имеет собственное значение.

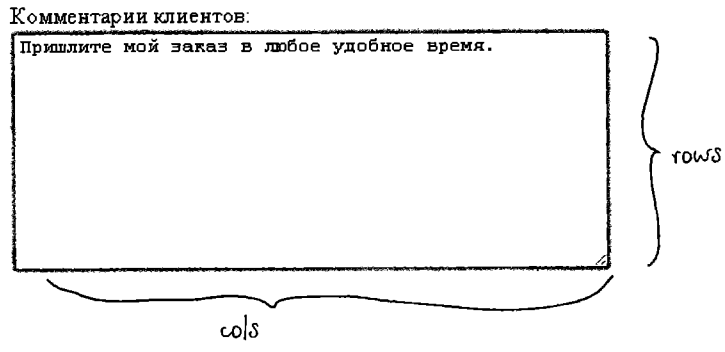
```
<input type="checkbox" name="spice" value="соль" />
<input type="checkbox" name="spice" value="перец" />
<input type="checkbox" name="spice" value="чеснок" />
```

Что может входить в форму (часть II)

Итак, не каждый элемент формы – это элемент **<input>**. Существует несколько других элементов, например **<select>** для меню и **<textarea>** для ввода многострочного текста. Почему бы нам не познакомиться с ними перед тем, как продолжать обучение? После этого вы будете знать 90 % информации об элементах форм (и 99 % элементов форм, которые станете часто использовать).

textarea

Элемент <textarea> создает многострочное текстовое поле для ввода данных. Если вы вводите больше текста, чем может вместиться в одну строку, то справа появляется полоса для прокрутки.



Элемент <textarea> – это не пустой элемент, что есть у него есть и открывающий, и закрывающий теги.

Используйте атрибут name, чтобы присвоить элементу уникальное имя.

Атрибут cols говорит браузеру, какой ширины должно быть текстовое поле (ширина определяется количеством символов, которые поместятся в этом поле).

```
<textarea name="comments" rows="10" cols="48"></textarea>
```

Атрибут rows говорит браузеру, какой высоты должно быть текстовое поле (высота также задается количеством символов, которые поместятся в этом поле).

Текст, находящийся между открывающим и закрывающим тегами, будет по умолчанию отображаться в поле ввода.

select

Элемент <select> создает на веб-странице раскрывающийся список. Он предоставляет возможность выбирать предлагаемые варианты ответов. Элемент <select> работает в паре с элементом <option>, создающим меню.



Элемент <select> создает раскрывающийся список, который выглядит следующим образом (хотя его внешний вид может очень сильно варьироваться в разных браузерах).

Элемент <select> должен окружать все пункты меню, чтобы сгруппировать их.

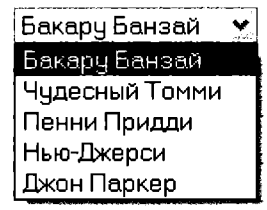
Как и в других элементах формы, присвойте элементу <select> уникальное имя, используя атрибут name.

```
<select name="characters">
    <option value="Баккару">Баккару Банзай</option>
    <option value="Томми">Чудесный Томми</option>
    <option value="Пенни">Пенни Придди</option>
    <option value="Джерси">Нью-Джерси</option>
    <option value="Джон">Джон Паркер</option>
</select>
```

option

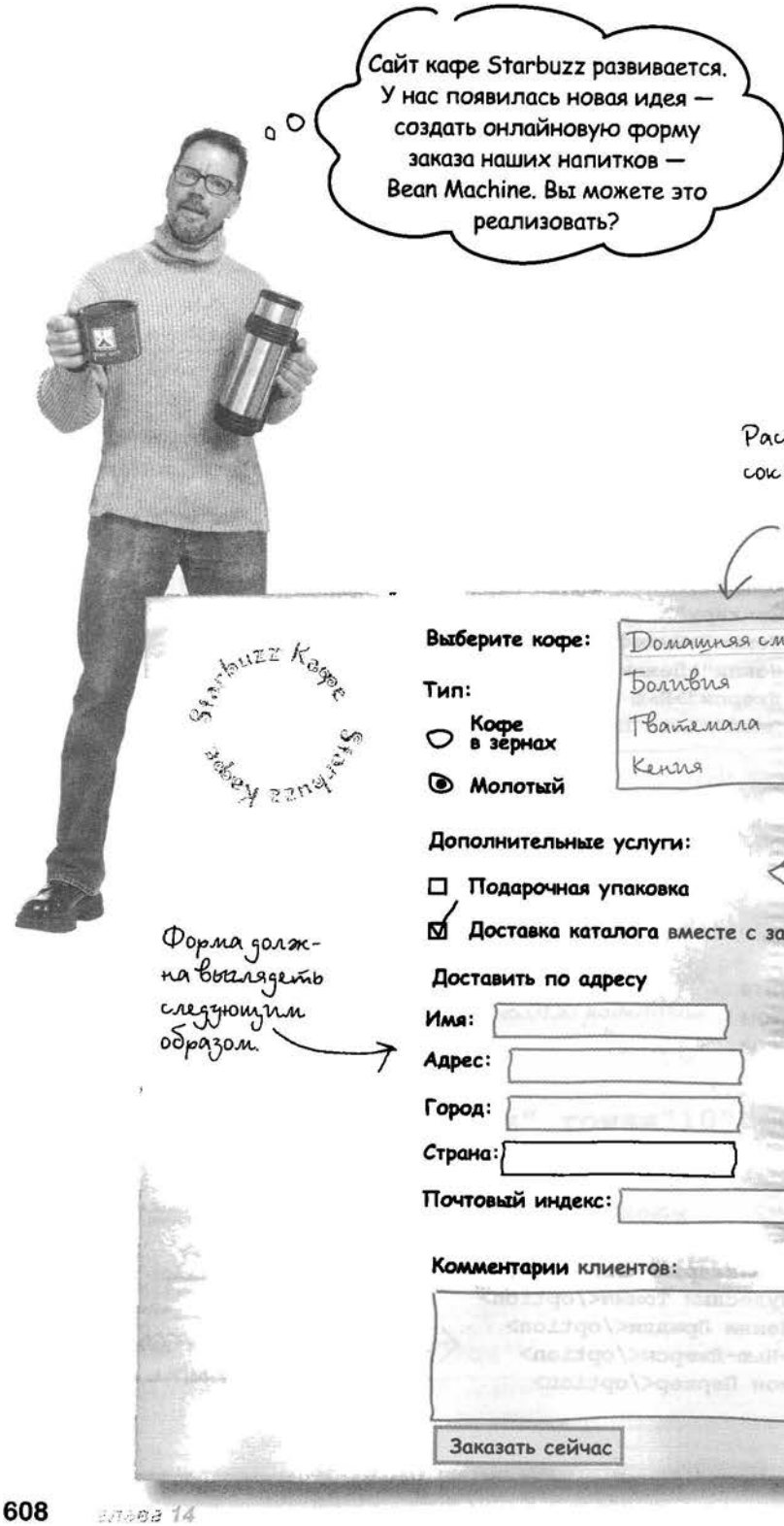
Чтобы создать меню, используйте элемент <option> в паре с элементом <select>. Задавайте элемент <option> для каждого пункта меню.

После того как вы щелкнете на меню, появится список его пунктов.




```
<select name="characters">
    <option value="Баккару">Баккару Банзай</option>
    <option value="Томми">Чудесный Томми</option>
    <option value="Пенни">Пенни Придди</option>
    <option value="Джерси">Нью-Джерси</option>
    <option value="Джон">Джон Паркер</option>
</select>
```

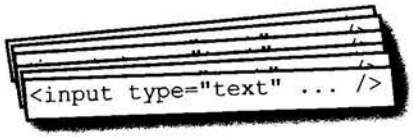
Содержимое элемента <option> используется для описания пунктов меню. Каждый пункт также включает представляющий его атрибут value.

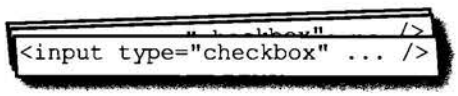

Сайт кафе Starbuzz развивается. У нас появилась новая идея — создать онлайновую форму заказа наших напитков — Bean Machine. Вы можете это реализовать?
Starbuzz Coffee
Starbuzz Coffee
Раскрывающийся список с напитками.
Выберите кофе:
Домашняя смесь
Боливия
Гватемала
Кения
Тип:
Кофе в зернах
Молотый
Молотый кофе или кофе в зернах (можно выбрать только одно).
Дополнительные услуги:
Подарочная упаковка
Доставка каталога вместе с заказом
Подарочная упаковка или включить каталог (не выбирайте ни одного или выберите один или два).
Форма должна выглядеть следующим образом.
Доставить по адресу
Имя:
Адрес:
Город:
Страна:
Почтовый индекс:
Информация о заказчике и его адрес, состоящие из пяти текстовых полей.
Комментарии клиентов:
Поле для комментариев клиентов.
Заказать сейчас
Кнопка отправления заказа.



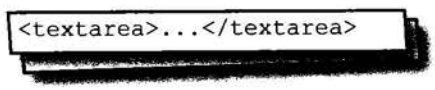
Магниты для разметки

Ваша задача — взять магниты с элементами формы и расположить их над соответствующим элементом управления на схеме. Для выполнения работы вам не понадобятся все предлагаемые магниты — некоторые останутся незадействованными. Проверьте правильность своего ответа в конце главы и только потом продолжите читать далее.

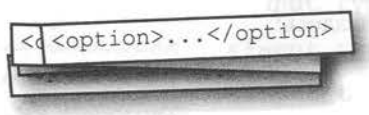


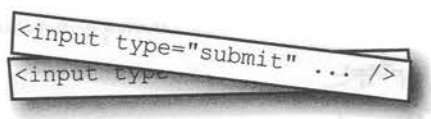


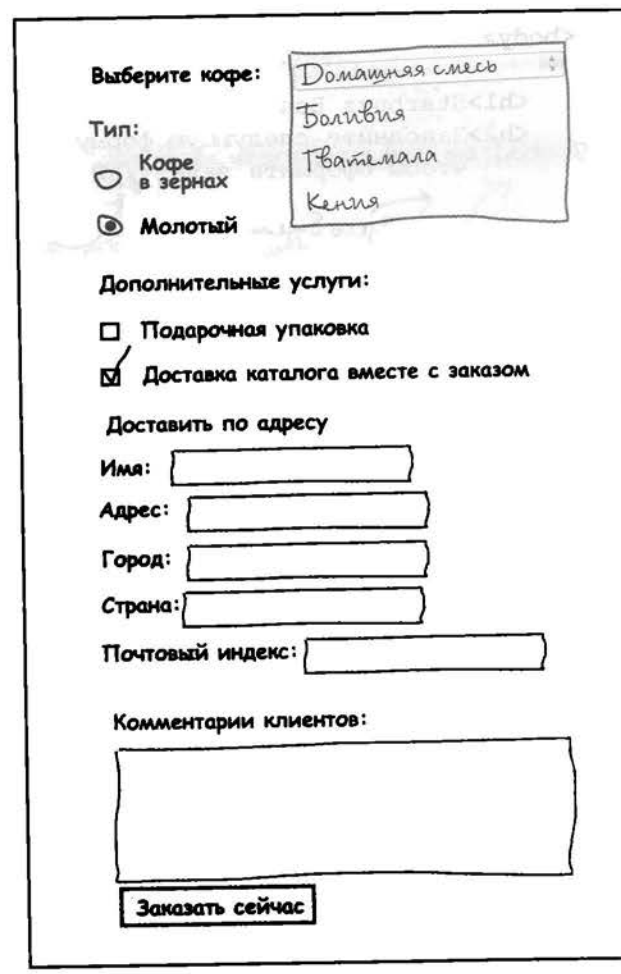












Подготовка к созданию формы для Bean Machine

Перед тем как мы начнем создавать эту форму, откройте папку `chapter14/starbuzz` и найдите в ней файл `form.html`. Откройте его и внимательно рассмотрите код. Все, что содержится в этом файле, — это основы XHTML.

```
<!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="ru" xml:lang="ru" >
   <head >
      <meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />
      <title>Starbuzz Bean Machine</title>
   </head>
   <body>

      <h1>Starbuzz Bean Machine</h1>
      <h2>Заполните следующую форму и нажмите кнопку Отправить,
          чтобы оформить заказ</h2>

   </body>
</html>
```

До сих пор у нас были лишь заголовок для страницы и инструкции.

Здесь будет расположена форма.

Сейчас мы создадим форму, к которой не будет применен общий стиль сайта Starbuzz. Таким образом мы сконцентрируемся лишь на XHTML-форме. Над стилем поработаем позже.

Из чего состоит элемент <form>

Пришло время использовать элемент **<form>**. Создавая этот элемент, в первую очередь нужно помнить, что он представляет собой URL для веб-приложения, которое будет обрабатывать данные вашей формы. Мы уже позаботились об этом для вас. Нужное веб-приложение вы найдете здесь:

```
http://www.starbuzzcoffee.com/processorder.php
```

Этот URL-адрес указывает на сайт starbuzzcoffee...

...а веб-приложение processorder.php находится на их сервере. Это приложение уже знает, как обрабатывать заказы из тех форм, которые мы будем создавать.

Добавление элемента <form>

Если вы уже знаете URL веб-приложения, которое будет обрабатывать вашу форму, то остается лишь вставить его в атрибут **action** *вашего элемента* **<form>** *(далее работайте самостоятельно и внесите в свой XHTML-код соответствующие изменения):*

```
<!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="ru" xml:lang="ru" >
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />
      <title>Starbuzz Bean Machine</title>
   </head>
   <body>
      <h1>Starbuzz Bean Machine</h1>
      <h2>Заполните следующую форму и нажмите кнопку Отправить,
          чтобы оформить заказ</h2>

      <form action="http://www.starbuzzcoffee.com/processorder.php" method="POST">

      </form>
   </body>
</html>
```

Это элемент form.

Атрибут action содержит URL веб-приложения.

Помните, что для передачи данных формы серверу мы используем метод POST. Подробнее об этом позже

Добавьте также закрывающий тег </form>.

Пока все идет нормально, но пустой элемент **<form>** вряд ли принесет много пользы. Вернитесь к схеме формы и взгляните на нее еще раз. Предстоит добавить много различных элементов, но начнем с простого: создадим часть формы, посвященную адресу доставки. Если вы помните, она состоит из набора текстовых полей. Вы уже знаете, что представляют собой текстовые поля ввода, но давайте разберемся с ними более детально. Поля ввода для формы Starbuzz выглядят следующим образом:

Мы используем <input> для нескольких разных элементов управления. Атрибут type указывает тип элемента управления.

```
<input type="text" name="name" />
<input type="text" name="address" />
<input type="text" name="city" />
<input type="text" name="state" />
<input type="text" name="zip" />
```

У нас есть по одному элементу для каждого поля ввода в форме: имя, адрес, город, страна и почтовый индекс.

В данном случае задан тип text, потому что этот элемент управления будет текстовым полем.

Атрибут name выступает в роли идентификатора данных, которые вводит пользователь. Обратите внимание, что все их значения различны. Посмотрим, как это работает...

Как работают атрибуты name элементов формы

Об атрибуте **name** вам обязательно нужно знать следующее: он играет роль связующего звена между вашей формой и веб-приложением, которое ее обрабатывает. Рассмотрим, как это работает.

У каждого элемента управления типа input вашей формы есть атрибут name.

Вводя элементы формы в свой XHTML-файл, вы присваиваете им уникальные имена. Вы уже видели, как это делается для текстовых полей:

```
<input type="text" name="name" />
<input type="text" name="address" />
<input type="text" name="city" />
<input type="text" name="state" />
<input type="text" name="zip" />
```

Обратите внимание, что в данном случае атрибут name элемента имеет значение name.

Каждый элемент <input> получает индивидуальное имя.

Когда вы представляете форму на рассмотрение, браузер упаковывает все данные, используя уникальные имена.

Допустим, вы ввели свое имя, адрес, город, страну и почтовый индекс в форму и нажали кнопку Submit (Отправить). Браузер берет каждый кусок данных и помечает его тем именем, которое вы использовали в качестве значения атрибута name. Затем браузер отправляет имена и значения серверу следующим образом.

То, что вы ввели в форму.

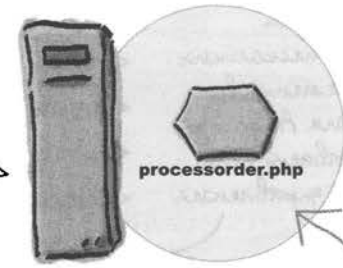
Имя: Баккару Банзай
Адрес: Институт Банзай
Город: Лос-Анджелес
Страна: США
Почтовый индекс: 90050

Уникальные имена для каждого элемента формы

```
name = Баккару Банзай
address = Институт Банзай
city = Лос-Анджелес
state = США
zip = 90050
```

Каждое уникальное имя получает значение, которое берется из данных, вводимых в форму.

Информация, которую браузер упаковывает для сервера.



www.starbuzzcoffee.com

Она нужна веб-приложению, чтобы данные формы были помечены соответствующим образом.

Часто Задаваемые Вопросы

В: В чем разница между элементом <input> типа text и элементом <textarea>?

О: Вы можете использовать элемент <input> типа text для ввода таких текстовых данных, которые вмещаются в одну строку, например, имя или почтовый индекс, а элемент <textarea> применяйте для ввода более длинного текста.

В: Могу ли я изменить надпись на кнопке Submit (Отправить) на какую-нибудь другую?

О: Да, просто используйте в элементе атрибут value и присвойте ему нужное значение. Вы также можете использовать атрибут value для элемента input типа text, чтобы определить для него текст, который будет отображаться по умолчанию.

В: Есть ли какие-то ограничения по объему информации, который я могу ввести в элемент <input> типа text или в элемент <textarea>?

О: Браузеры задают определенные ограничения на количество информации, которое можно вводить в элементы <input> или в <textarea>. Однако вряд ли вам когда-то понадобится больше информации, чем допускают эти ограничения. Если вы все же хотите ограничить количество символов, которое ваши пользователи могут ввести в элемент <input>, используйте атрибут maxlength и задайте ему значение, равное определенному количеству символов. Например, maxlength="100" не позволит пользователям ввести более 100 символов. Однако для <textarea> в XHTML нет способа ограничить количество символов, которое могут ввести пользователи.

В: Я до сих пор не понял, как имена ставятся в соответствие данным формы.

О: Вы знаете, что каждый элемент формы имеет уникальное имя и у каждого элемента есть свое значение. Когда вы нажимаете кнопку Submit (Отправить), браузер берет все имена вместе с их значениями и отсылает серверу. Например, когда вы вводите почтовый индекс 90050 в элемент <input> типа text с именем zip, браузер отсылает серверу "zip = 90050", после того как форма представляется на рассмотрение.

В: Как веб-приложение узнает, какие имена я собираюсь использовать в своей форме? Другими словами, как мне выбирать имена для элементов формы?

О: Хороший вопрос. На самом деле все происходит наоборот: это вы должны знать, какие имена ожидает увидеть ваше веб-приложение, и, учитывая это, писать свою форму. Если вы используете веб-приложение, написанное кем-то другим, то этот кто-то должен вам сказать, какие имена нужно использовать, или представить вам такую информацию в документации для приложения. Неплохо бы начать с того, чтобы попросить о помощи вашу хостинговую компанию.

В: Почему у элемента <option> нет атрибута name? У всех остальных элементов форм он есть.

О: Хорошее замечание. Все элементы <option> на самом деле являются частью меню, которое создается элементом <select>. В действительности нам нужно лишь одно имя для всего меню, и оно уже присвоено элементу <select>. Иначе говоря, элементам <option> не нужен атрибут name, потому что в элементе <select> уже указано имя для всего меню целиком. Не забывайте, что при представлении формы на рассмотрение на сервер под этим именем отсылается только то значение, которое пользователь выбрал в меню.

В: Разве вы не говорили, что имя каждого элемента формы должно быть уникальным? Но все элементы <input> типа radio имеют одинаковое имя.

О: Верно. Положения для переключателей всегда используются группами. Подумайте над этим: если вы установите одно положение переключателя, то остальные станут неактивными. Итак, чтобы браузер знал, что положения переключателя взаимосвязаны, для них используется одно имя. Допустим, у вас есть переключатель под названием "color" (цвет) с положениями "красный", "зеленый" и "синий". Во всех значениях указаны цвета, и выбран может быть только один, так что в указании одного общего имени для всего набора есть логика.

В: Как насчет флажков? Они работают так же, как переключатели?

О: Да. Единственное отличие состоит в том, что вы можете установить сразу несколько флажков. Когда браузер отсылает данные формы серверу, он комбинирует значения всех флажков в одно и отсылает их под именем элемента checkbox. Допустим, у вас есть флажки под общим названием "spice" (специи) со значениями "соль", "перец" и "чеснок", и вы установили их все. В результате браузер отошлет серверу выражение "spice = соль&перец&чеснок".

В: Неужели я действительно должен знать, как данные попадают на сервер?

О: Вам нужно знать лишь те имена и типы элементов формы, которые ожидает веб-приложение. Хотя иногда очень полезно разбираться в том, как все это работает, вам не обязательно знать все подробности того, в каком виде данные отсылаются на сервер.

Вернемся к размещению элементов <input> в XHTML

Теперь у нас есть элементы **<input>** внутри формы. Взгляните на дополнения, описанные ниже, и внесите соответствующие изменения в свой файл.

Начнем с того, что поместим все внутрь элемента <p>.

Вы можете вкладывать непосредственно в форму только блочные элементы.

Это ТОЛЬКО фрагмент формы form.html.

```
<form action="http://www.starbuzzcoffee.com/processorder.php" method="POST">
  <p>Доставить по адресу: <br />
     Имя: <input type="text" name="name" /> <br />
     Адрес: <input type="text" name="address" /> <br />
     Город: <input type="text" name="city" /> <br />
     Страна: <input type="text" name="state" /> <br />
     Индекс: <input type="text" name="zip" /> <br />
  </p>
  <p>
     <input type="submit" value="Заказать сейчас" />
  </p>
</form>
```

Это все элементы <input>: по одному для каждого элемента типа text в разделе формы под названием «Доставить по адресу».

Мы добавили метку для каждого поля, чтобы пользователь знал, что вводить.

Вы также должны знать, что <input> — это строчный элемент, поэтому если вы хотите, чтобы между элементами <input> были разрывы строки, придется добавить элементы <br />. Именно поэтому вам также нужно вложить их в элемент <p>.

Наконец, пользователям нужна кнопка, после нажатия которой данные будут отправлены на сервер. Добавьте ее на форму, указав элемент <input> типа submit в самом конце кода. Кроме того, задайте для нее значение "Заказать сейчас", в результате чего надпись на кнопке поменяется.

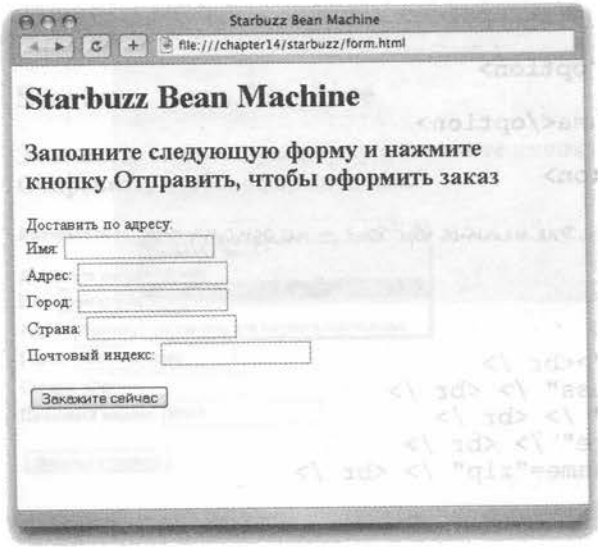
После того как вы внесете все эти изменения, сохраните файл `form.html` и двигайтесь дальше!

Не забудьте проверить свой XHTML-код на валидность. Элементы формы тоже должны быть проверены!

Тест для формы

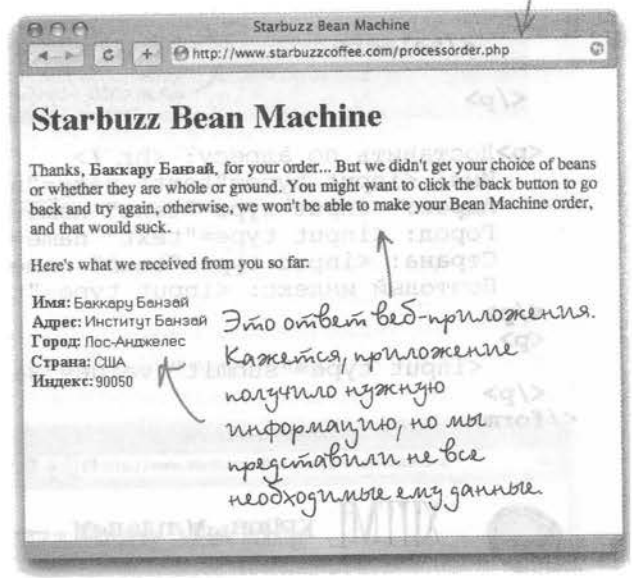
Обновите страницу, заполните текстовые поля и представьте форму на рассмотрение. Когда вы это сделаете, браузер упакует данные и отошлет их по адресу, указанному в атрибуте **action**. В данном случае это **www.starbuzzcoffee.com**.

Вы же не думаете, что мы дадим вам пример, который в реальности не работает? Сайт starbuzzcoffee.com готов принять вашу форму на обработку. Не упустите этой возможности!



Это форма.

Обратите внимание на то, как изменился URL в адресной строке браузера, после того как вы представите форму на рассмотрение (вы увидите его в атрибуте action).



Это ответ после рассмотрения формы приложением.

Это ответ веб-приложения. Кажется, приложение получило нужную информацию, но мы представили не все необходимые ему данные.

Добавим в форму еще несколько элементов <input>

Кажется, веб-приложение не позволит нам двигаться дальше, пока мы не укажем, какие сорта кофе хотим заказать, должен ли этот кофе быть молотым или в зернах. Добавим возможность такого выбора, поместив в форму элемент **<select>**. Помните, что этот элемент содержит перечень параметров, каждый из которых можно выбрать в раскрывающемся списке. Кроме того, в соответствие каждому пункту списка ставится значение. Когда форма представляется на рассмотрение, значение выбранного пункта списка отсылается серверу. Переверните страницу и посмотрите, как правильно добавить элемент **<select>**.

Добавление элемента <select>

```
<form action="http://www.starbuzzcoffee.com/processorder.php" method="POST">
  <p> Выберите кофе:
    <select name="beans">
      <option value="Смесь">Домашняя смесь</option>
      <option value="Боливия">Боливия</option>
      <option value="Гватемала">Гватемала</option>
      <option value="Кения">Кения</option>
    </select>
  </p>
  <p>Доставить по адресу: <br />
    Имя: <input type="text" name="name" /><br />
    Адрес: <input type="text" name="address" /> <br />
    Город: <input type="text" name="city" /> <br />
    Страна: <input type="text" name="state" /> <br />
    Почтовый индекс: <input type="text" name="zip" /> <br />
  </p>
  <p>
    <input type="submit" value="Заказать сейчас" />
  </p>
</form>
```

Это наш новый элемент <select>. Он имеет уникальное имя.

Внутрь мы помещаем элементы <option> — по одному для каждого сорта кофе.



HTML крупным планом

Давайте подробно рассмотрим элемент **<option>**.

У каждого элемента option есть свое значение.

Содержимое элемента используется в качестве пункта раскрывающегося списка.

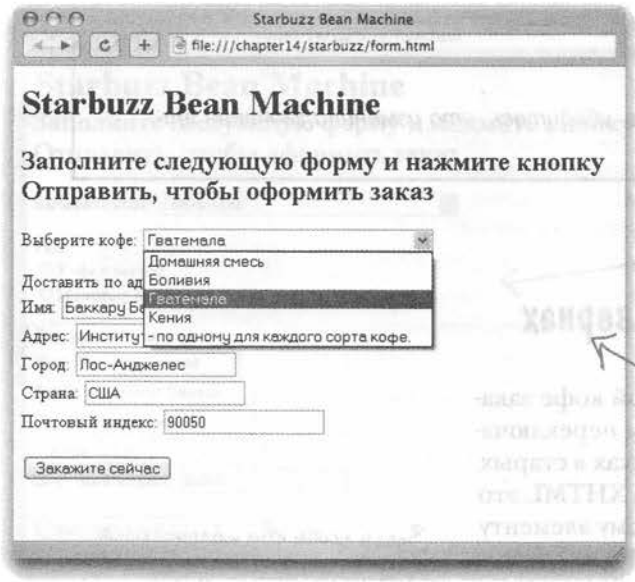
```
<option value="House Blend">Домашняя смесь</option>
```

Когда браузер упаковывает имена и значения формы, он использует имя элемента <select> вместе со значением выбранного элемента option.

В данном случае браузер отошлет серверу выражение beans = "Домашняя смесь".

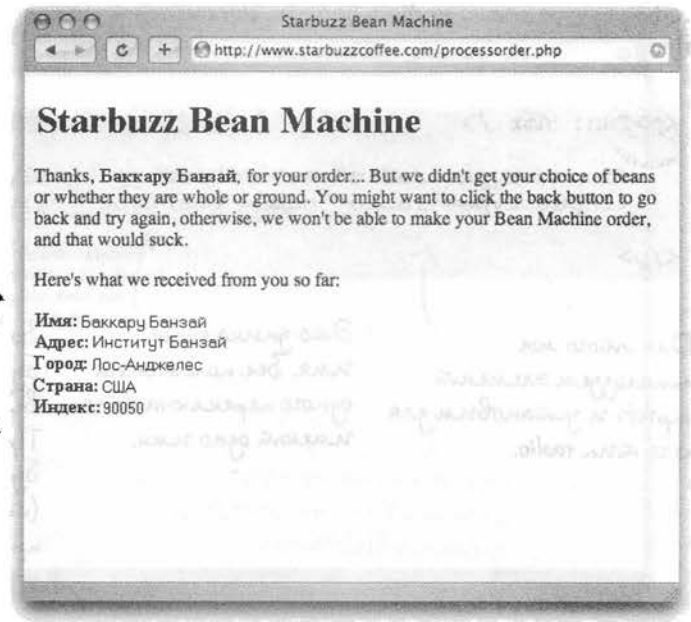
Тест для элемента <select>

Теперь посмотрим, как работает элемент **<select>**. Обновите страницу, и вы увидите, что появился раскрывающийся список. Выберите любимый сорт кофе, заполните остальные поля формы и сделайте заказ.



Это форма, дополненная элементом <select>. Обратите внимание, что в нем есть все параметры.

Мы еще не представили веб-приложению всю необходимую ему информацию, но оно по крайней мере успешно приняло те данные, которые были введены в форму.



Это результат выбора <select>.

Это данные, введенные в текстовые поля.



Измените значение атрибута **name** элемента **<select>** на **thembeans**. Обновите форму и сделайте новый заказ. Как это повлияло на ответ, который вы получили от веб-приложения?

После того как вы выполните это упражнение, убедитесь, что изменили значение атрибута name обратно на beans.

Предоставьте клиенту выбор, хочет он молотый кофе или кофе в зернах

Клиент должен иметь возможность выбрать, какой кофе заказать: молотый или в зернах. Для этого мы создаем переключатели. Они действуют как кнопки на радиоприемниках в старых автомобилях: можно нажать только одну из них. В XHTML это работает следующим образом: вы создаете по одному элементу **<input>** типа **radio** для каждого положения переключателя. В данном случае вам нужны два положения: одно для молотого кофе, другое – для кофе в зернах. Рассмотрим, как это выглядит:

Здесь есть два положения переключателя: одно для кофе в зернах, другое – для молотого кофе.

```
<p>Тип: <br />

   <input type="radio" name="beantype" value="зерна"  /> Кофе в зернах <br />
   <input type="radio" name="beantype" value="молотый" /> Молотый кофе

</p>
```

Для этого мы используем элемент <input> и установим для него тип radio.

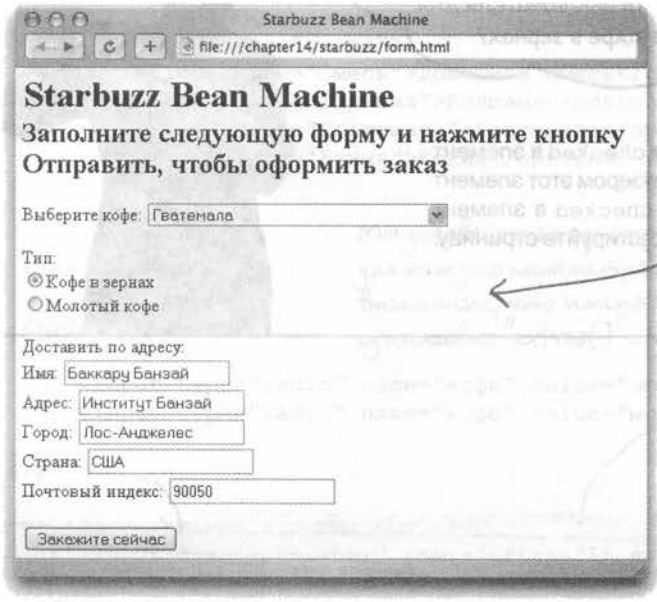
Это уникальное имя. Все положения одного переключателя имеют одно имя.

Вот значения, которые будут отправлены веб-приложению. Причем отправлено будет только одно из них (то, которое выбрано при представлении формы на рассмотрение).

Обратите внимание, что для переключателей метки часто ставятся справа от самого элемента.

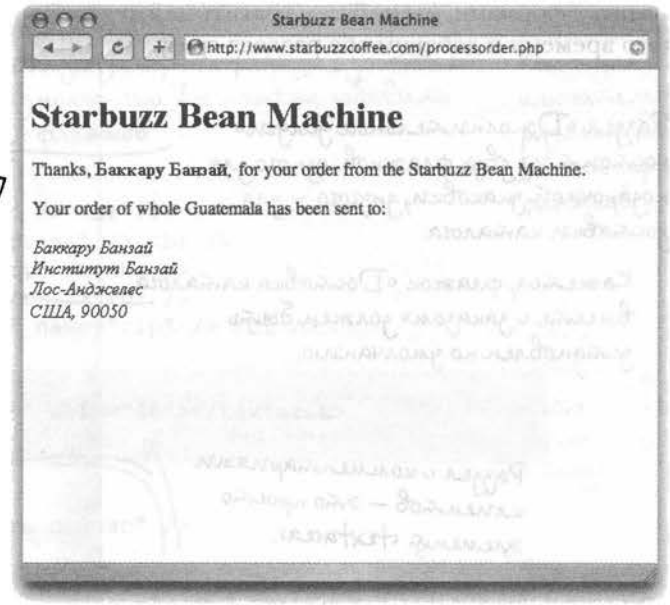
Стилизация переключателей

Возьмите код для переключателя с предыдущей страницы и вставьте его в свой XHTML-код, прямо под абзац, содержащий элемент **<select>**. Убедитесь, что обновили страницу, и снова представьте форму на рассмотрение.



Скорее всего, после обновления страницы не будет выбрано ни одно положение переключателя, но это зависит от того, какой браузер вы используете.

Ого! Кафе Starbuzz приняло наш заказ, хотя мы даже не завершили работы над формой. Нам все еще нужно добавить сведения о подарочной упаковке и область для ввода комментариев клиентов.



Как же могло случиться, что заказ был успешно принят, а на форме еще не все элементы? Все зависит от того, как запрограммировано веб-приложение. В данном случае оно настроено на обработку заказа даже в том случае, если вместе с остальными данными не представлена информация о подарочной упаковке, каталоге и комментариях клиента. Чтобы узнать, какие именно элементы формы требуются веб-приложению, спросите об этом разработчика или прочитайте в документации.



Упражнение

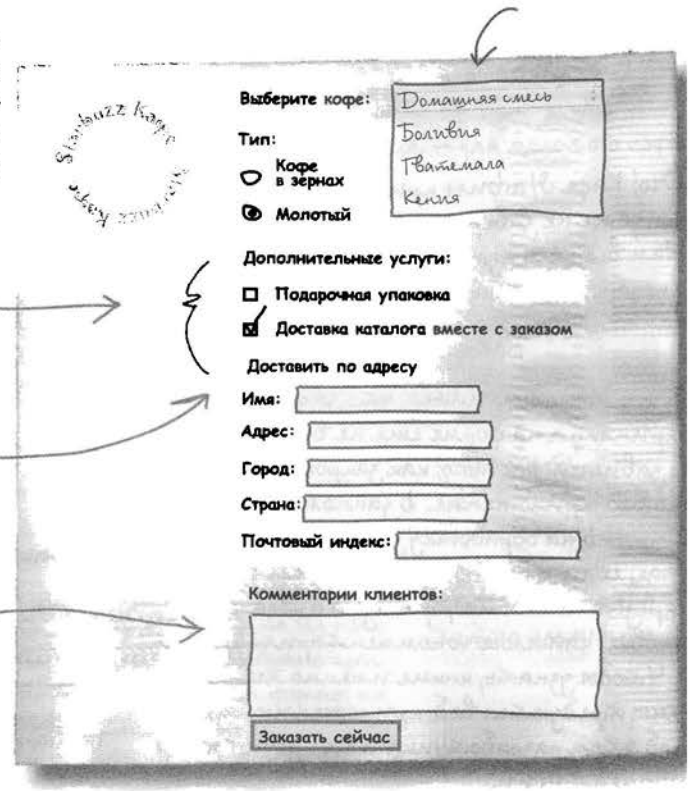
80% наших клиентов заказывают молотый кофе. Можете ли вы сделать так, чтобы, когда пользователь загружает страницу, по умолчанию было выбрано положение переключателя для заказа кофе в зернах?



Если вы добавите атрибут `checked` со значением `checked` в элемент `input` типа `radio`, то при отображении формы браузером этот элемент будет выбран по умолчанию. Добавьте атрибут `checked` в элемент `<input>` типа `radio` под названием `зерна` и протестируйте страницу. Решение упражнения вы найдете в конце главы.

Дополнение формы

Вы уже почти все сделали. Осталось добавить в форму два раздела: раздел «Дополнительные услуги» с двумя флажками и раздел для комментариев клиентов. Поскольку вы уже научились работать с формами, мы не будем тратить на это много времени и добавим оба раздела сразу.



Раздел «Дополнительные услуги» состоит из двух флажков: одного для подарочной упаковки, другого — для доставки каталога.

Кажется, флажок «Доставка каталога вместе с заказом» должен быть установлен по умолчанию.

Раздел с комментариями клиентов — это просто элемент «textarea».

Добавление флажков и многострочного текстового поля

Вы знаете, что делать: просмотрите новый XHTML-код и добавьте его в файл `form.html`.

```
<form action="http://www.starbuzzcoffee.com/processorder.php" method="POST">
   <p> Выберите кофе:

        <select name="beans">
        <option value="Смесь">Домашняя смесь</option>
        <option value="Боливия">Боливия</option>
        <option value="Гватемала">Гватемала</option>
        <option value="Кения">Кения</option>
        </select>

   </p>

   <p>Тип: <br />

       <input type="radio" name="кофе" value="зерна" />Кофе в зернах<br />
       <input type="radio" name="кофе" value="молотый" checked="checked" />Молотый кофе

   </p>

   <p>Дополнительные услуги: <br />
       <input type="checkbox" name="extras[]" value="упаковка" />Подарочная упаковка<br />
       <input type="checkbox" name="extras[]" value="каталог" checked="checked" />Доставка
       каталога вместе с заказом
   </p>

   <p>Доставить по адресу: <br />
       Имя: <input type="text" name="name" /> <br />
       Адрес: <input type="text" name="address" /> <br />
       Город: <input type="text" name="city" /> <br />
       Штат: <input type="text" name="state" /> <br />
       Почтовый индекс: <input type="text" name="zip" /> <br />

   <p>Комментарии клиентов:<br />
       <textarea name="comments" rows="10" cols="48"></textarea>
   </p>

   <p>
       <input type="submit" value="Заказать сейчас" />
   </p>
</form>
```

Мы добавили по одному флажку для каждого параметра. Обратите внимание, что имена элементов одинаковы: "extras[]"...

...но их значения разные.

Мы используем атрибут checked, чтобы указать, что флажок о доставке каталога должен быть установлен по умолчанию. Вы можете добавить атрибут checked сразу для нескольких флажков.

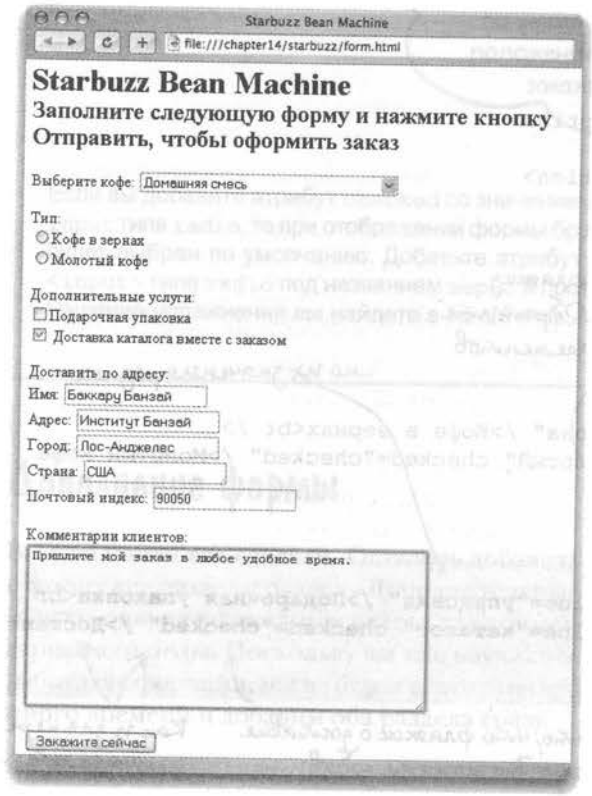
Как и для переключателей, мы разместили метки справа от флажков.

Это многострочное текстовое поле

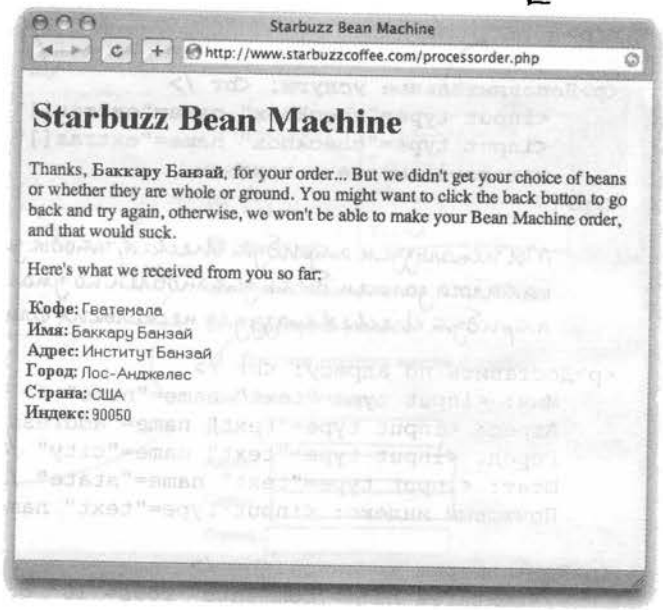
Мы указали, что хотим, чтобы оно было 10 символов в высоту и 48 символов в ширину.

Завершающий тест

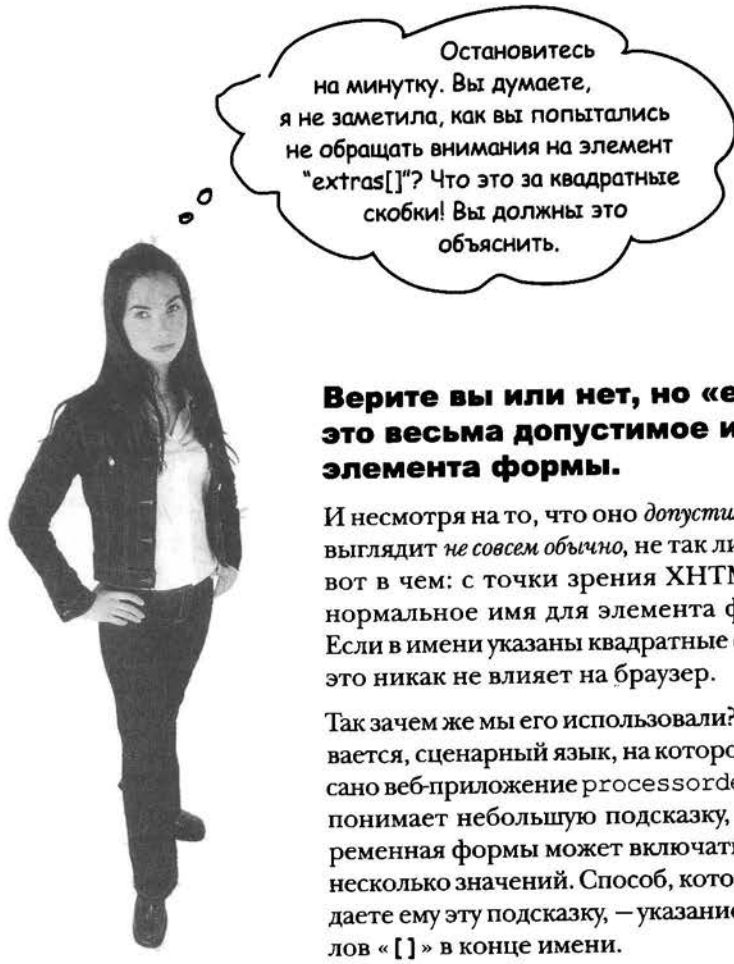
Сохраните изменения, обновите страницу и оцените новую форму. Не находите, что она достаточно хорошо выглядит?



Вот то, что вы получите после представления информации на рассмотрение. Веб-приложение получило все данные формы и использовало их на странице с ответом. Убедитесь, что можете найти на ней все данные, которые ввели в форме.



Попробуйте использовать различные комбинации данных в форме, предлагаемой на рассмотрение (с подарочной упаковкой и без нее, с каталогом и без, с различными сортами кофе и т. д.), и посмотрите, как все будет работать.



Верите вы или нет, но «extras[]» — это весьма допустимое имя для элемента формы.

И несмотря на то, что оно *допустимо*, оно выглядит *не совсем обычно*, не так ли? Дело вот в чем: с точки зрения XHTML это нормальное имя для элемента формы. Если в имени указаны квадратные скобки, это никак не влияет на браузер.

Так зачем же мы его использовали? Оказывается, сценарный язык, на котором написано веб-приложение `processorder.php`, понимает небольшую подсказку, что переменная формы может включать в себя несколько значений. Способ, которым вы даете ему эту подсказку, — указание символов «**[]**» в конце имени.

Итак, если вы ограничиваетесь только изучением XHTML, можете полностью об этом забыть. Однако лучше сохраните эту информацию в самом дальнем уголке сознания на тот случай, если когда-нибудь в будущем захотите написать форму, которая будет использовать PHP-приложение.



Поработайте браузером

Ниже приведена XHTML-форма, а справа — те данные, которые ввел в нее пользователь. Ваша задача — стать на время браузером и поставить в соответствие каждому имени элемента формы значение, введенное пользователем. Выполнив это упражнение, проверьте, все ли вы сделали правильно.

```
<form action="http://www.chooseyourmini.com/choice.php" method="POST">
   <p>Ваша информация: <br />

      Имя: <input type="text" name="name" /><br />
      Почтовый индекс: <input type="text" name="zip" /><br />

   </p>
   <p>Какую модель вы хотите? <br />
     <select name="model">
         <option value="купер">Мини Купер</option>
         <option value="куперS">Мини Купер S</option>
         <option value="откидной">Мини Купер с откидным верхом</option>
     </select>
   </p>
   <p>Какой цвет вы хотите? <br />
     <input type="radio" name="color" value="чилийский" />Чилийский красный <br />
     <input type="radio" name="color" value="синий" />Ярко-синий
   </p>
   <p>Какие дополнительные параметры вы хотите?  <br />
     <input type="checkbox" name="caroptions[]" value="двигатель" />Спортивный двигатель
     <br />
     <input type="checkbox" name="caroptions[]" value="сиденья" />Спортивные сиденья
   </p>

   <p>
     <input type="submit" value="Заказать сейчас" />
   </p>

</form>
```

Это форма.



Эта форма заполнена.

Поставьте в соответствие каждому полю формы его значение и впишите свои ответы сюда.

```
name = "Баккару Банзай"
zip = _____
model = _____
color = _____
caroptions[] = _____
```



Есть два основных метода, которые использует браузер: POST и GET.

POST и **GET** выполняют одну и ту же работу – доставляют данные вашей формы из браузера на сервер. Но они делают это различными способами. **POST** упаковывает переменные формы и незаметно для пользователя отсылает их серверу. В то же время **GET** также упаковывает переменные формы, но, прежде чем отправить запрос серверу, присоединяет их в конец URL-адреса.

POST

При использовании метода POST все данные формы отсылаются серверу как часть запроса и не видны пользователю.



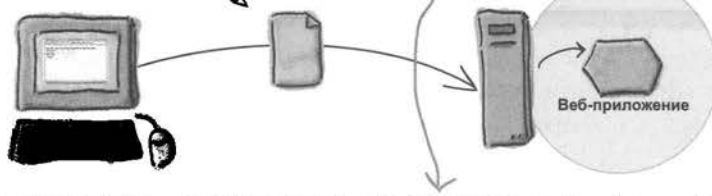
В адресной строке браузера пользователь видит только URL веб-приложения.

```
http://www.headfirstlabs.com/contest.php
```

GET

В самом запросе нет никаких данных формы.

При использовании метода GET данные формы добавляются к самому URL-адресу, так что пользователь их видит.



Обратите внимание, что данные формы добавляются в конец URL-адреса. Это то, что пользователь видит в адресной строке браузера.

```
http://www.headfirstlabs.com/contest.php?firstname=buckaroo&lastname=banzai
```

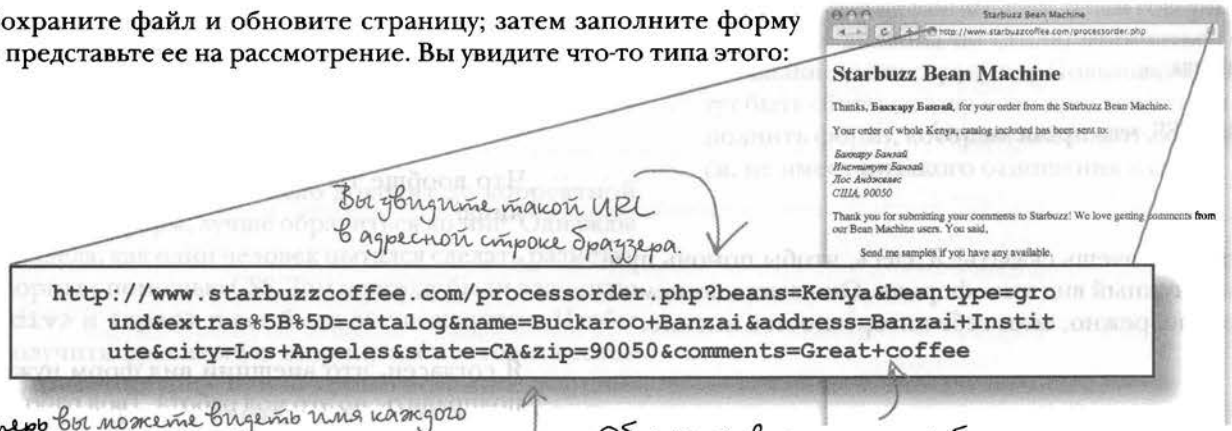
Проверим GET на практике

Нет лучшего способа понять, как работает метод GET, чем увидеть его в действии. Откройте файл `form.html` и внесите небольшое изменение:

Просто изменим метод с POST на GET.

```
<form action="http://www.starbuzzcoffee.com/processorder.php" method="GET">
```

Сохраните файл и обновите страницу; затем заполните форму и представьте ее на рассмотрение. Вы увидите что-то типа этого:



Вы увидите такой URL в адресной строке браузера.

```
http://www.starbuzzcoffee.com/processorder.php?beans=Kenya&beantype=gro
und&extras%5B%5D=catalog&name=Buckaroo+Banzai&address=Banzai+Instit
ute&city=Los+Angeles&state=CA&zip=90050&comments=Great+coffee
```

Теперь вы можете видеть имя каждого элемента формы, а также его значение, прямо здесь, в URL-адресе.

Обратите внимание, что браузер кодирует некоторые символы, например пробелы. Веб-приложение автоматически декодирует их, когда получит запрос.

Часто Задаваемые Вопросы

В: Почему метод называется GET («получатель»), хотя мы, наоборот, отсылаем что-то серверу?

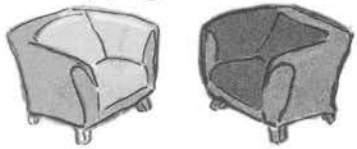
О: Хороший вопрос. Какова основная работа браузера? *Получать* веб-страницы от сервера. Когда вы используете метод GET, браузер просто получает веб-страницу, как делает это всегда. Кроме того, в случае с формой он присоединяет кое-какие данные в конец URL-адреса. Надо также отметить, что браузер воспринимает это как обычный запрос. Когда вы используете метод POST, браузер, наоборот, создает небольшой пакет данных и отправляет их серверу.

В: В чем же преимущества и недостатки использования методов POST и GET?

О: Есть несколько отличий, которые на самом деле имеют большое значение. Если вы хотите, чтобы пользователи могли напрямую ссылаться на веб-страницы, которые являются результатом представления формы на рассмотрение, то лучше используйте метод GET, потому что не существует способа сослаться на страницу, являющуюся результатом рассмотрения формы методом POST. Когда вам может такое понадобиться? Допустим, у вас есть веб-приложение, которое возвращает список результатов поиска. Тогда можно сделать так, чтобы пользователи могли ссылаться на эти результаты и снова просматривать их без необходимости вновь заполнять всю форму. С другой стороны, если ваше веб-приложение обрабатывает заказы, то вы не захотите, чтобы пользователи ссылались на результаты таких обработок. Иначе каждый раз, когда они будут возвращаться к этой странице, заказ будет снова и снова представляться на рассмотрение. Ситуация, в которой вы точно никогда не захотите использовать метод GET, — когда данные в вашей форме являются конфиденциальными, например хранят номер кредитной карты или пароль. Поскольку URL легко увидеть, конфиденциальная информация может быть запросто получена другими людьми, если они просмотрят историю вашего браузера или если каким-то образом GET попадет в закладки. Наконец, если вы используете элемент <textarea>, вы просто обязаны применять метод POST, так как наверняка пересылаете большие объемы данных. Метод GET накладывает ограничения на длину запроса, который может содержать всего 256 символов. В то же время метод POST не имеет никаких ограничений на размер пакета пересылаемых данных.

дальше ›

Беседа у камина



Вечерний диалог: **Таблица и CSS спорят о том, как делать разметку для форм.**

Таблица

Эй, CSS, что происходит?

Что ты хочешь сказать? Я здесь, чтобы помочь придать нужный вид этим формам. Они выглядят немного... небрежно, если тебе интересно мое мнение.

Да, и знаешь, некоторые люди как раз полагают, что элементы формы — это и есть табличные данные. Кроме того, я придаю формам отличный внешний вид намного лучше, чем ты.

Я. Я делаю так, что формы выглядят аккуратно и красиво. Кроме того, я как следует выравниваю метки и элементы формы.

Я согласна, что я не могу добавить эти мелкие штрихи, которые можешь добавить ты, но они ведь не так уж и важны. Чтобы сделать формы дружественными пользователю, главное — расположить метки и соответствующие им элементы так, чтобы это имело смысл и пользователи не путались в том, куда вводить нужную информацию. Все остальное пользователя совершенно не заботит.

CSS

Что вообще ты делаешь в этой главе, Таблица?

Я согласен, что внешний вид форм нужно подправить, но это моя работа. Твоя работа — табличные данные, помнишь это?

Кто-кто лучше?

Ты знаешь, я тоже могу позиционировать элементы. Наши читатели уже разобрали главу о позиционировании и знают, как «выровнять все как следует», используя CSS. Тем не менее даже если ты в состоянии хорошо выровнять элементы, то, конечно же, не сможешь раскрасить элементы, добавить отступы и изменить шрифты.

Таблица

Возможно. Но когда дело доходит до корректной разметки форм, лучше обратиться ко мне. Однажды я видела, как один человек пытался сделать разметку формы с помощью CSS. Там повсюду были элементы ** ** и **** и все было очень запутано. Чтобы получить правильную ширину элементов и всевозможных отступов, нужно задавать сложное позиционирование и выполнять утомительные вычисления размеров полей... У меня заболела голова, хотя я всего лишь посмотрела на это.

Во всяком случае, используя мои строки и ячейки с данными, легче выяснить, что и где будет расположено. Если же применять твои методы позиционирования, это невозможно понять.

Но ведь пользователи вводят *данные* в элементы управления формы, не так ли? Формы созданы для того, чтобы получать информацию от того человека, который их использует. Почему же это не табличные данные?

Ладно, я хочу сказать, что раз я могу сделать эту работу лучше тебя, то и используй меня для ее выполнения. Я буду рада помочь.

Вот что я тебе скажу: почему бы нам не предоставить пользователям самим решать, что применять для разметки форм?

CSS

Да ты вообще не понимаешь, о чем говоришь. Формы не должны по стилю отличаться от остальной части страницы. Пользователи могут быть сбиты с толку, если предложить им заполнить форму, которая, как может показаться, не имеет никакого отношения к сайту.

А как насчет твоих элементов **<tr>** и **<td>**, засоряющих XHTML? Чем они лучше?

Ты, видимо, не читала главу 12. И я думаю, что элементы таблицы просто-напросто нельзя назвать табличными данными... Они являются XHTML-элементами, а не данными.

Хмм... Допустим, я отчасти с тобой согласен, но все же думаю, что это в корне неверно — использовать таблицы для разметки. Разметка — это *презентация* страницы, а это *моя* работа.

Лучше меня? Тише-тише, ты заходишь слишком далеко...

Возьми в руку карандаш

GET или POST

Для каждого описания обведите карандашом либо GET, либо POST, в зависимости от того, какой метод лучше всего подходит в указанной ситуации. Если вы думаете, что подойдет любой метод, обведите оба. Но будьте готовы отстоять свое мнение.

GET **POST** *Форма для ввода имени пользователя и пароля.*

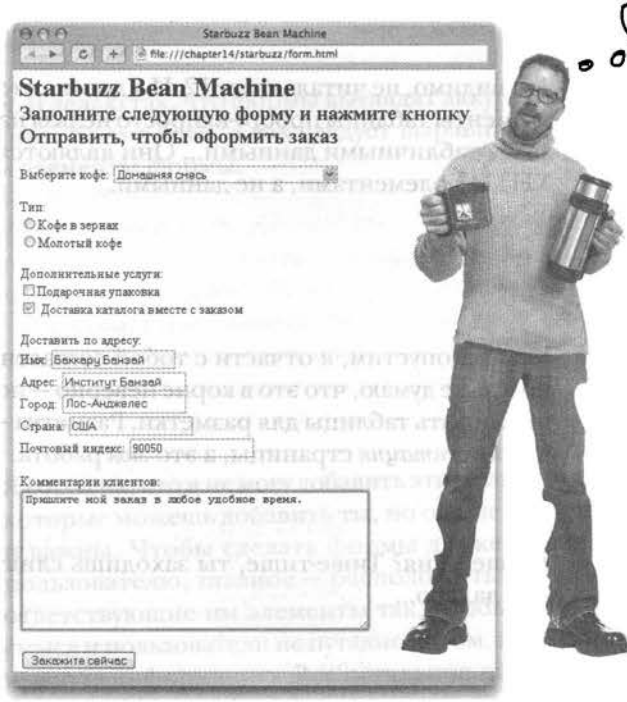
GET **POST** *Форма для заказа компакт-дисков.*

GET **POST** *Форма для поиска новостей.*

GET **POST** *Форма для отправки отзывов на книгу.*

GET **POST** *Форма для получения информации о пособиях по государственному идентификационному номеру.*

GET **POST** *Форма для отправки отзывов клиентов.*



Я хотел сказать, что вы отлично поработали над разделом Bean Machine! Новая форма на самом деле увеличит объемы продаж нашего кофе. Вам осталось немного стилизовать этот раздел, и мы будем готовы предоставить нашим пользователям возможность онлайн-заказа.

МОЗГОВОЙ ШТУРМ

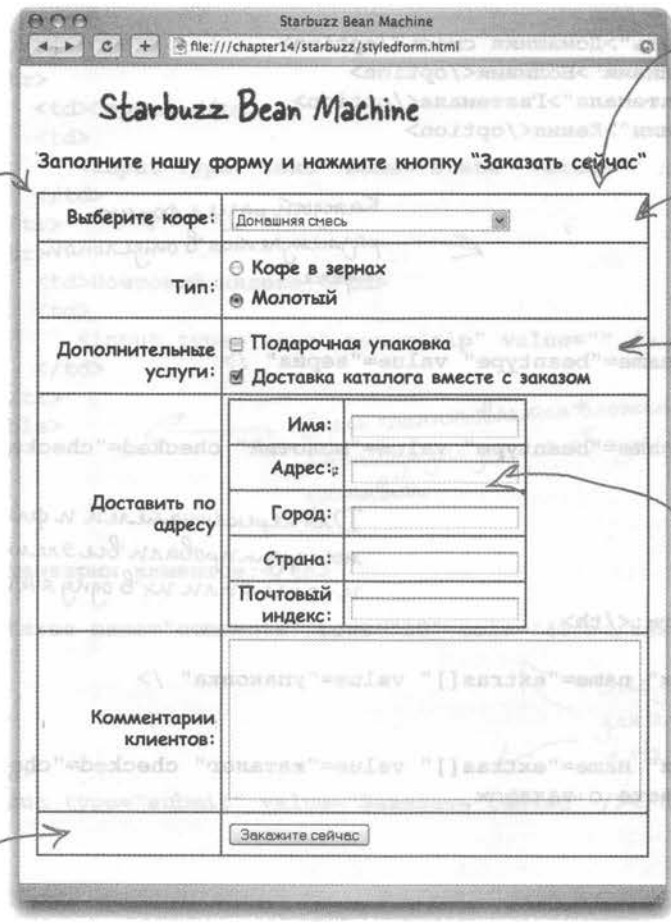
Как вы стилизуете эту форму, применяя все свои знания в области XHTML и CSS?

Использовать таблицы или нет

Вы встретите много людей, которые ответят на этот вопрос утвердительно, и не меньше тех, кто ответит отрицательно. Будете ли вы использовать CSS для разметки своих форм? Или, может быть, лучше применять таблицы? Суровая реальность состоит в том, что создавать разметку форм с помощью CSS достаточно сложно. И если вы все же решите потратить свою энергию и время на создание разметки формы, используя CSS, мы, конечно же, с радостью одобрим ваш выбор и посмотрим на вас с восхищением. Однако разметки многих форм на самом деле уже являются таблицами. Так почему бы не использовать таблицы, чтобы сделать разметку формы, а CSS предоставить работу по стилизации? Таким образом можно получить преимущества *обеих* сторон.

Начнем с разметки...

Начнем того, что поместим форму в таблицу. Взгляните на форму, приведенную ниже, и вы увидите, что ее структура отлично соответствует таблице. Более того, она стала выглядеть как настоящая форма, а не как беспорядочный набор элементов для ввода информации. Обратите также внимание, что мы использовали вложенную таблицу в разделе «Доставить по адресу».



Это эскиз простой таблицы, состоящей из двух столбцов и шести строк, по одной для каждого основного раздела формы.

Метки для каждого элемента формы помещаются в левый столбец.

Все элементы для ввода информации мы поместили в правый столбец.

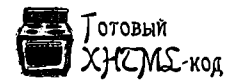
Обратите внимание, что переключатель и флажки расположены в отдельных ячейках с данными.

В разделе «Доставить по адресу» есть пять элементов <input> типа text, так что мы сгруппировали их и поместили в одну вложенную таблицу. Вложенная таблица имеет такую же основу разметки, что и главная таблица: два столбца и пять строк, по одной строке для каждой пары «метка/элемент».

Правая ячейка для строки с кнопкой Заказать сейчас пуста. Нет метки, которую можно поместить сюда.

дальше ▸

Размещение элементов формы в таблице



Теперь, когда вы знаете, как расположить элементы формы в таблице, вам предстоит протестировать свои навыки по созданию XHTML-таблиц. Итак, начните печатать прямо сейчас!

Это была шутка. Мы не будем заставлять вас все это печатать. В конце концов, эта глава посвящена формам, а не таблицам. Мы уже набрали весь код сами: вы найдете его в файле `styledform.html`, который находится в папке `chapter14/starbuzz`. Даже несмотря на то, что он выглядит достаточно сложно, все не так плохо. Ниже мы добавили несколько замечаний, чтобы выделить основные части.

Это элемент <form>. Нам не нужно помещать эту часть в таблицу.

```
<form action="http://www.starbuzzcoffee.com/processorder.php" method="POST">

<table>
   <tr>
      <th>Выберите кофе:</th>
      <td>
         <select name="beans">
            <option value="Смесь">Домашняя смесь</option>
            <option value="Боливия">Боливия</option>
            <option value="Гватемала">Гватемала</option>
            <option value="Кения">Кения</option>
         </select>
      </td>
   </tr>

   <tr>
      <th>Тип:</th>
      <td>
         <input type="radio" name="beantype" value="зерна" />
         Кофе в зернах
         <br />
         <input type="radio" name="beantype" value="молотый" checked="checked" />
         Молотый
      </td>
   </tr>

   <tr>
      <th>Дополнительные услуги:</th>
      <td>
         <input type="checkbox" name="extras[]" value="упаковка" />
         Подарочная упаковка
         <br />
         <input type="checkbox" name="extras[]" value="каталог" checked="checked" />
         Доставка каталога вместе с заказом
      </td>
   </tr>
```

Здесь начинается таблица.

В каждой строке основной таблицы есть по две ячейки: <th> для метки и <td> — для элемента формы

Каждый раздел формы размещается в отдельной строке.

Для переключателей и флажков мы сгруппировали все элементы и поместили их в одну ячейку таблицы

```
<tr>
   <th>Доставить по адресу:</th>
   <td>
      <table>
         <tr>
            <td>Имя:</td>
            <td>
               <input type="text" name="name" value="" />
            </td>
         </tr>
         <tr>
            <td>Адрес:</td>
            <td>
               <input type="text" name="address" value="" />
            </td>
         </tr>
         <tr>
            <td>Город:</td>
            <td>
               <input type="text" name="city" value="" />
            </td>
         </tr>
         <tr>
            <td>Страна:</td>
            <td>
               <input type="text" name="state" value="" />
            </td>
         </tr>
         <tr>
            <td>Почтовый индекс:</td>
            <td>
               <input type="text" name="zip" value="" />
            </td>
         </tr>
      </table>
   </td>
</tr>

<tr>
   <th>Комментарии клиентов:</th>
   <td>
      <textarea name="comments" rows="10" cols="48"></textarea>
   </td>
</tr>

<tr>
   <th></th>
   <td><input type="submit" value="Заказать сейчас" /></td>
</tr>
</table>
</form>
```

Для данных об адресе доставки мы создали вложенную таблицу в ячейке основной таблицы. Мы сделали это так, чтобы можно было хорошо выровнять метки для каждого элемента <input> типа text в разделе формы «Доставить по адресу».

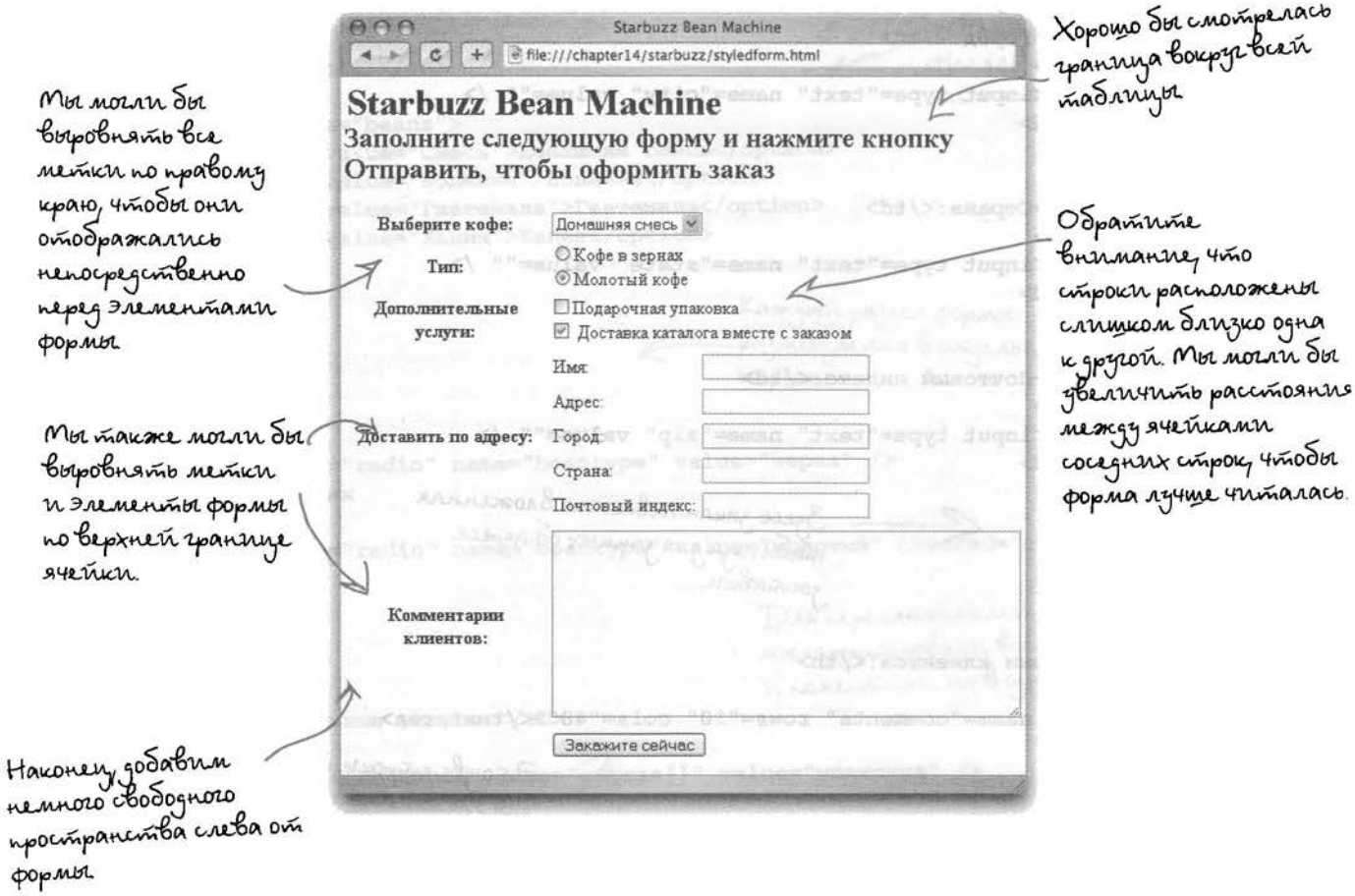
Здесь заканчивается вложенная таблица для данных об адресе доставки.

Это две строки основной таблицы для элементов <input> типа submit и <textarea>.

Тест для формы в виде таблицы

Откройте страницу `styledform.html` в браузере и посмотрите на форму Starbuzz Bean Machine в виде таблицы. Она выглядит лучше, не так ли? Все метки и элементы формы выровнены, и это выглядит более профессионально.

Теперь мы можем использовать CSS, чтобы усовершенствовать внешний вид формы в некоторых ее местах. Посмотрим, что можно изменить.



Стилизация формы и таблицы с помощью CSS



Осталось добавить несколько правил стиля для XHTML, и все будет готово. Поскольку наша форма — это часть сайта Starbuzz, мы будем использовать стили из файла `starbuzz.css` и создадим таблицу стилей под названием `styledform.css`, в которую добавим новые правила стиля для формы Bean Machine. Вы уже видели весь этот CSS-код. Мы не используем никаких индивидуальных правил стиля для форм и таблиц: все это уже было сделано в предыдущей главе.

Вы найдете CSS-код в файле `styledform.css`, который находится в папке `chapter14/starbuzz`.

В некоторых стилях мы будем полагаться на CSS для всего Starbuzz-сайта, но изменим шрифт для элемента body на sans-serif, добавим фоновый рисунок Starbuzz и поля для элемента body.

```
body {
    background: #efe5d0 url(images/background.gif) top left;
    font-family: Verdana, Helvetica, Arial, sans-serif;
    margin: 20px;
}
```

Свойства шрифта будут унаследованы всеми элементами страницы, включая текст в таблице и форме.

```
table {
    border: thin dotted #7e7e7e;
    padding: 10px;
}
```

Мы добавляем границу вокруг таблицы и отступы.

```
th {
    text-align: right;
    vertical-align: top;
    padding-right: 10px;
    padding-top: 2px;
}
```

Метки формы находятся в заголовках таблицы. Нужно выровнять их по верхнему и правому краю так, чтобы они хорошо отображались рядом с элементами формы, расположенными в правом столбце таблицы. Кроме того, добавим для них небольшой отступ, чтобы создать вокруг немного свободного пространства.

```
td {
    vertical-align: top;
    padding-bottom: 15px;
}
```

Содержимое ячеек с данными по умолчанию уже выровнено по левому краю. Это как раз то, что нам необходимо. Однако нам также нужно выровнять их и в вертикальном направлении, чтобы они отображались на одном уровне с заголовками таблицы. Здесь мы тоже добавляем небольшие отступы, чтобы между строками было свободное место.

```
table table {
    border: none;
    padding: 0px;
}

table table td {
    text-align: right;
    padding-bottom: 0px;
}
```

Эти два правила переопределяют некоторые свойства, которые мы задали выше для элементов table и td. Почему? Потому что мы не хотим, чтобы у вложенной таблицы была граница, а хотим, чтобы промежутки между ее ячейками были уже, поэтому убираем отступы. Нам также нужно выровнять метки формы в ячейках вложенной таблицы по правому краю (они не являются заголовками основной таблицы, как остальные заголовки, поэтому не будут выровнены правилом, указанным выше).

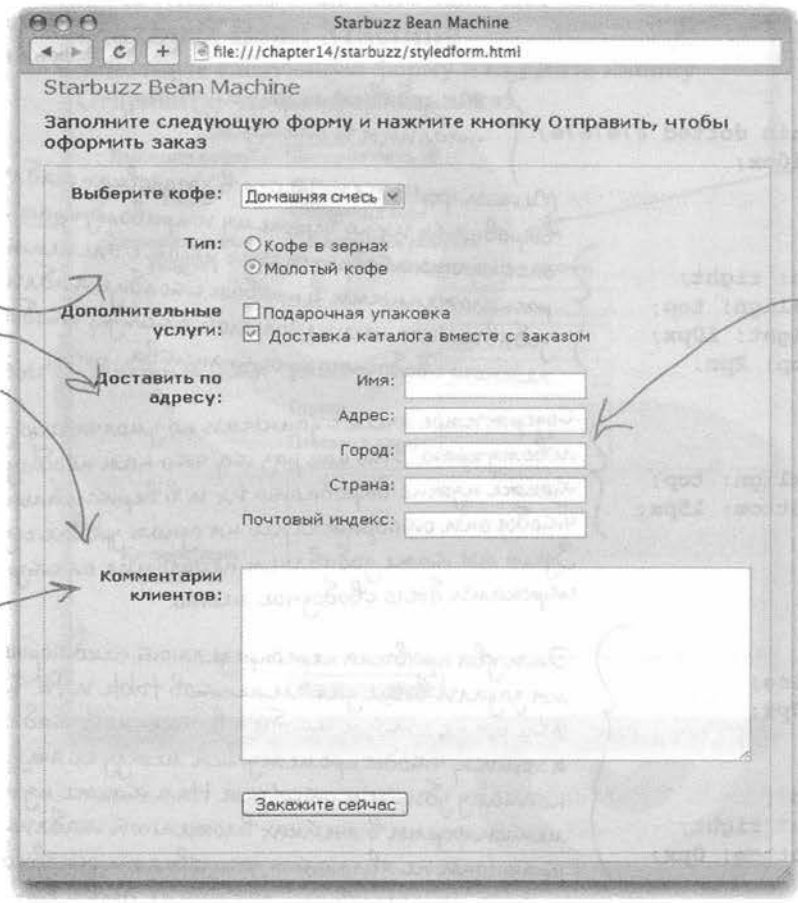
Последний тест

Вы добавите *два* элемента **<link>** в элемент **<head>** в файле `styledform.html`. Эти элементы ссылаются на таблицу стилей Starbuzz из главы 12 – `starbuzz.css` и на новую таблицу стилей – `styledform.css`. Убедитесь, что добавили элементы **link** в правильном порядке: сначала `starbuzz.css`, затем `styledform.css`. Создав ссылки на обе таблицы стилей, сохраните файл и обновите страницу. Вы увидите шикарную стильную версию Starbuzz Bean Machine.

Ого, как сильно все меняется при добавлении небольшого стиля!

Если вы хотите немного расширить свои навыки в области XHTML и CSS, попробуйте добавить нижний и верхний колонтитулы Starbuzz на страницу Bean Machine и сделайте так, чтобы форма выглядела еще более привлекательно с этими элементами.

Форма теперь полностью соответствует остальному сайту Starbuzz.



Метки выровнены по верхнему и правому краям элементов формы. Благодаря этому сразу становится понятно, какая метка к какому элементу относится.

Промежутки между строками таблицы сильно влияют на ее внешний вид, и информация в ней читается намного проще.

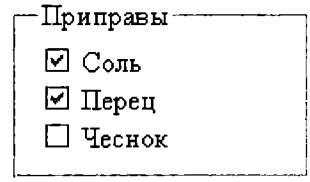
Обратите внимание, что у вложенной таблицы нет границы и промежутки между ее ячейками уже. Так произошло вследствие работы тех правил, которые переопределяют свойства основной таблицы.

Что еще может входить в форму

Мы использовали почти все элементы, которые обычно применяются в формах, но есть еще несколько элементов, которые вы, возможно, захотите указывать в своих формах. Мы опишем их сейчас на тот случай, если в будущем вы захотите продолжить создавать формы самостоятельно.

Элементы <fieldset> и <legend>

Когда ваши формы начинают увеличиваться в размерах, полезно визуально сгруппировать элементы. Конечно, для этого вы можете использовать элементы ** ** и CSS, но в XHTML предусмотрен элемент **<fieldset>**, который может быть использован для группировки взаимосвязанных элементов. Он работает в паре с элементом **<legend>** следующим образом:



В браузере элементы fieldset и legend выводятся так. Вы увидите, что браузеры отображают их по-разному.

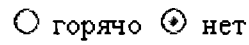
Элемент <fieldset> окружает набор элементов input.

Элемент <legend> задает метку для всей группы.

```
<fieldset>
   <legend>Condiments</legend>
      <input type="checkbox" name="spice" value="соль" />
         Соль  <br />
      <input type="checkbox" name="spice" value="перец" />
         Перец <br />
      <input type="checkbox" name="spice" value="чеснок" />
         Чеснок
</fieldset>
```

Элемент <label>

До сих пор вы помечали элементы формы обычным текстом, но в XHTML есть специальный элемент **<label>**, который представляет больше информации о структуре страницы и позволяет проще стилизовать метки с помощью CSS. Кроме того, этот элемент может помочь экранным дикторам для людей с ослабленным зрением правильно распознавать элементы формы.



По умолчанию метки ничем не отличаются от обычного текста. Но если вплотную заняться ими, то можно очень сильно изменить внешний вид меток.

Чтобы использовать элемент <label>, сначала добавьте атрибут id в свой элемент формы.

```
<input type="radio" name="hotornot" value="горячо" id="hot" />
<label for="hot">горячо</label>

<input type="radio" name="hotornot" value="нет" id="not" />
<label for="not">нет</label>
```

Затем добавьте элемент <label> и присвойте его атрибуту for соответствующее идентификационное имя.

Вы можете использовать <label> для любого элемента формы.

Элемент <input> типа password

Элемент **<input>** типа **password** работает как обычный элемент **<input>** типа **text**, за исключением того, что вводимый вами текст зашифрован. Это может быть удобно в формах, где есть необходимость вводить пароли, секретные коды или иную информацию, которую не должны видеть другие люди. Запомните: информация из формы *не будет* отсылаться от браузера веб-приложению безопасным способом, если вы только специально об этом не позаботитесь. Для получения большей безопасности свяжитесь со своей хостинговой компанией.

```
<input type="password" name="secret" />
```

Элемент <input> типа password работает точно так же, как элемент <input> типа text, за исключением того, что текст, который вы вводите, зашифрован.

Элемент <input> типа file

Это абсолютно новый тип элемента **input**, о котором мы еще не говорили. Если вам необходимо отослать веб-приложению целый файл, вы используете элемент **<input>**, но задаете для него тип **file**. В результате элемент **<input>** создаст для формы управляющий элемент, который позволит выбрать файл. После того как форма будет представлена на рассмотрение, содержимое этого файла будет передано серверу вместе с остальными данными. Помните: ваше веб-приложение должно ожидать загрузки файла. Обратите также внимание, что в данном случае нужно использовать метод POST.



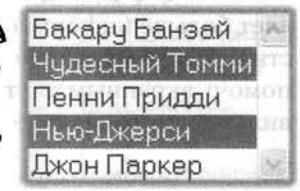
Вот так выглядит элемент input типа file в различных браузерах.

```
<input type="file" name="doc" />
```

Чтобы создать элемент input типа file, просто укажите этот тип.

Параметр multiple

Это не новый элемент, а всего лишь новый способ использования элементов. Если вы добавите атрибут **multiple** со значением **multiple** в свой элемент **<select>**, то превратите меню с однозначным выбором в такое меню, где можно выбрать сразу несколько пунктов. Вы получите меню, все пункты которого сразу отображаются на экране (если этих пунктов много, то добавляется полоса прокрутки). Можете выбрать более одного пункта, удерживая нажатой клавишу Ctrl (Windows) или Command (Mac) в то время, как делаете свой выбор.



Указав параметр multiple, вы можете выбрать сразу несколько пунктов меню одновременно.

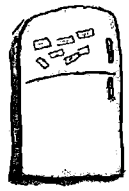
```
<select name="characters" multiple="multiple">
    <option value="Баккару">Баккару Банзай</option>
    <option value="Томми">Чудесный Томми</option>
    <option value="Пенни">Пенни Придди</option>
    <option value="Джерси">Нью-Джерси</option>
    <option value="Джон">Джон Паркер</option>
</select>
```

Просто добавьте атрибут multiple со значением multiple, чтобы преобразовать меню с однозначным выбором в меню с многозначным выбором.



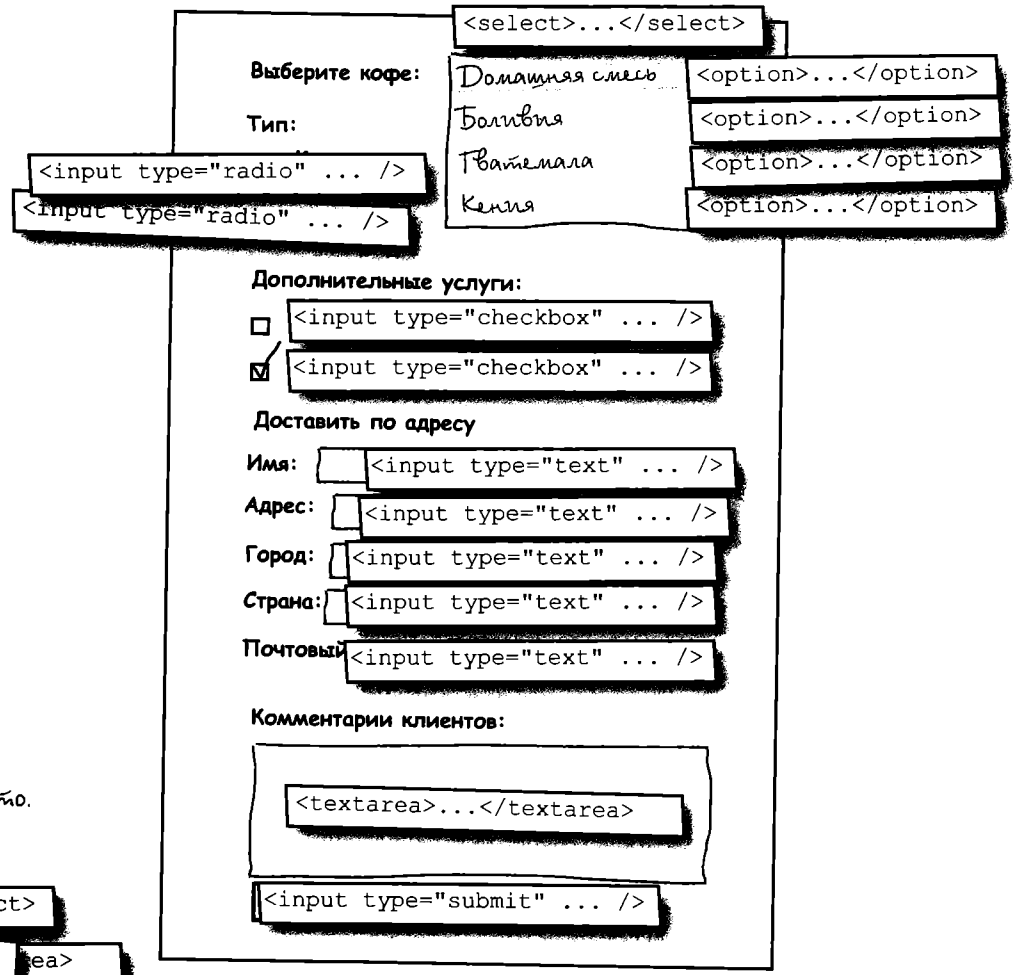
ПОВТОРИМ выученное

- Элемент **<form>** определяет форму, и все ее элементы вкладываются в него.
- Атрибут **action** содержит URL-адрес веб-приложения.
- Атрибут **method** задает метод отправки данных: POST либо GET.
- POST упаковывает данные формы и отсылает их как часть запроса.
- GET упаковывает данные формы и присоединяет их к URL-адресу.
- Используйте метод POST, если данные формы должны быть конфиденциальными. Кроме того, этот метод применяется, если информации слишком много, например, в форме использованы элементы **<textarea>** или элемент **<input>** типа **file**.
- Используйте метод GET для таких запросов, на которые пользователь может ссылаться, а также создавать закладки.
- Элемент **<input>** может создавать множество различных элементов управления для ввода информации на веб-странице в зависимости от значения его атрибута **type**.
- Тип **text** создает элемент управления для ввода однострочного текста.
- Тип **submit** создает кнопку Submit (Отправить).
- Тип **radio** создает одно положение переключателя. Все положения, имеющие одинаковое имя, создают переключатель.
- Тип **checkbox** создает один флажок на странице. Вы можете создать целый набор флажков, задав нескольким флажкам одно и то же имя.
- Элемент **<textarea>** создает многострочное текстовое поле для ввода информации.
- Элемент **<select>** создает меню, состоящее из одного или нескольких элементов **<option>**. Элементы **<option>** определяют пункты меню.
- Если вы поместите какой-нибудь текст внутрь **<textarea>**, то он по умолчанию будет отображаться в соответствующем поле для ввода на веб-странице.
- Атрибут **value** элемента **<input>** типа **text** может быть использован для указания текста, который будет изначально отображен в элементе управления для ввода однострочного текста.
- Используя атрибут **value** для кнопки Submit (Отправить), можно изменить надпись на этой кнопке.
- Когда веб-форма представляется на рассмотрение, значения ее данных ставятся в соответствие именам элементов формы и все имена и значения отсылаются серверу парами.
- Для создания разметки форм часто применяются таблицы; в таком случае формы имеют табличную структуру. После создания разметки нужно использовать CSS для стилизации форм, а именно для добавления цвета, стиля шрифта и т. д.
- XHTML позволяет структурировать элементы формы благодаря элементу **<fieldset>**.
- Элемент **<label>** может быть использован для добавления меток к элементам формы таким образом, чтобы было понятно, какая метка какому элементу соответствует.

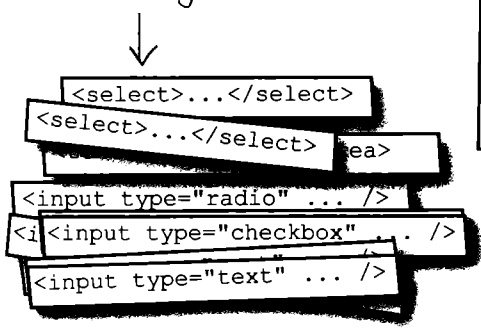


Магниты для разметки. Решение

Ваша задача — взять магниты с элементами формы и расположить их над соответствующим элементом управления на схеме. Для выполнения работы вам не понадобятся все предлагаемые магниты — некоторые останутся незадействованными. Далее приведено решение.



Нам не понадобится это.





Решение упражнения



```
name = "Баккару Банзай"
zip = "90050"
model = "с откидным верхом"
color = "чилийский красный"
caroptions[] = "спортивные сиденья"
```



Возьми в руку карандаш

Решение

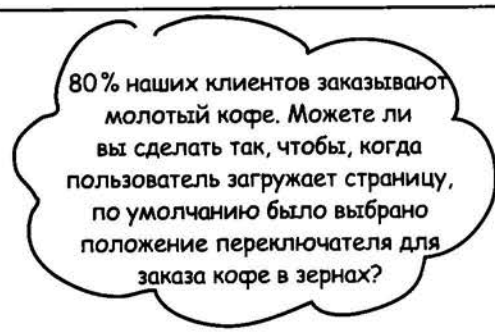
GET или POST

Для каждого описания обведите карандашом либо GET, либо POST, в зависимости от того, какой метод лучше всего подходит в указанной ситуации. Если вы думаете, что подойдет любой метод, обведите оба. Но будьте готовы отстоять свое мнение.

GET	**POST**	*Форма для ввода имени пользователя и пароля.*
GET	**POST**	*Форма для заказа компакт-дисков.*
GET	**POST**	*Форма для поиска новостей.*
GET	**POST**	*Форма для отправки отзывов на книгу.*
GET	**POST**	*Форма для получения информации о пособиях по государственному идентификационному номеру.*
GET	**POST**	*Форма для отправки отзывов клиентов.*



Решение упражнения





Если вы добавите атрибут **checked** со значением **checked** в элемент **<input>** типа **radio**, то при отображении формы браузером этот элемент будет выбран по умолчанию. Добавьте атрибут **checked** в элемент **<input>** типа **radio** под названием **зерна** и протестируйте страницу. Далее приведено решение.

Это просто раздел для формы файла form.html.

```
<form action="http://www.starbuzzcoffee.com/processorder.php" method="POST">
   <p>Выберите кофе:

        <select name="beans">

        <option value="Смесь">Домашняя смесь</option>

        <option value="Боливия">Боливия</option>

        <option value="Гватемала">Гватемала</option>

        <option value="Кения">Кения</option>

        </select>

   </p>

   <p>Тип: <br />

       <input type="radio" name="beantype" value="зерна"  /> Кофе в зернах <br />
       <input type="radio" name="beantype" value="молотый" checked="checked" /> Молотый
       кофе
   </p>
   <p>Доставить по адресу: <br />
       Имя: <input type="text" name="name" /> <br />
       Адрес: <input type="text" name="address" /> <br />
       Город: <input type="text" name="city" /> <br />
       Страна: <input type="text" name="state" /> <br />
       Почтовый индекс: <input type="text" name="zip" /> <br />
       <input type="submit" value="Заказать сейчас" />
   </p>
</form>
```

Это новый атрибут, который устанавливает переключатель в положение «Молотый кофе».

Топ-10 тем, которые не были освещены в этой книге



В книге мы рассмотрели очень много тем, и вы уже почти закончили свое обучение по ней. Однако мы не можем со спокойной душой отпустить вас, не рассказав еще кое-что напоследок. Как бы мы ни старались, но не сможем уместить в это маленькое приложение все то, что вам нужно знать, поэтому мы сначала включили в него все, что вам нужно знать об XHTML и CSS (все то, что не было освещено в предыдущих главах), уменьшив размер шрифта на 0,00004. Все вместилось, но прочесть это было нереально. В итоге мы все же выбросили большую часть информации и поместили в приложение только десять самых важных тем.

На самом деле это последний раздел книги.

#1 Дополнительные типы селекторов

Хотя вы уже ознакомились с большинством часто используемых селекторов, мы расскажем еще о нескольких, о которых вы, возможно, захотите знать.

Псевдоэлементы

Вы уже все знаете о псевдоклассах, а псевдоэлементы очень на них похожи. Они могут использоваться для выбора тех *частей элемента*, которые не получается окружить элементами ** ** или ****. Например, псевдоэлемент **first-letter** можно применять для выбора первой буквы текста блочного элемента, что позволяет добавлять буквицы. Существует еще псевдоэлемент, который называется **first-line** и применяется для выбора первой строки абзаца. Рассмотрим, как используются оба этих элемента для выбора первой буквы и первой строки элемента **<p>**:

Для псевдоэлементов используется тот же синтаксис, что и для псевдоклассов.

```
p:first-letter {
      font-size: 3em;
}

p:first-line {
      font-style: italic;
}
```

Здесь делаем первую букву абзаца большой, а шрифт первой строки делаем курсивным.

Селекторы атрибутов

Селекторы атрибутов не очень хорошо поддерживаются нынешними версиями браузеров, но будут более широко поддерживаться в дальнейшем. Селекторы атрибутов работают соответственно их названию: это такие селекторы, с помощью которых можно выбрать элементы по значениям их атрибутов. Они применяются следующим образом:

Этот селектор выбирает все изображения, у которых есть атрибут width.

```
img[width] { border: black thin solid; }

img[height="300"] { border: red thin solid; }

image[alt~="flowers"] { border: #ccc thin solid; }
```

Данный селектор выбирает все изображения, у которых есть атрибут height с заданным для него значением 300.

Этот селектор выбирает все изображения, у которых есть атрибут alt, в значение которого входит слово flowers.

Выбор по элементам того же уровня вложенности

Вы можете выбирать элементы, основываясь на предшествующих элементах такого же уровня вложенности. Допустим, вы хотите выбрать только те абзацы, для которых определены предшествующие элементы **<h1>** на том же уровне вложенности. Используйте следующий селектор:

Напишите предшествующий элемент, затем символ «+» и нужный элемент, находящийся на том же уровне вложенности.

```
h1+p {
  font-style: italic;
}
```

Селектор выберет абзацы, следующие сразу же после элементов <h1>.

Сочетание селекторов

Вы уже видели в книге примеры того, как могут сочетаться селекторы. Например, вы можете взять селектор класса и использовать его как часть селектора потомка:

```
.blueberry p { color: purple; }
```

Здесь мы выбираем абзацы — потомки элементов, являющихся членами класса blueberry.

Ниже приведен пример того, как можно создать достаточно сложный комбинированный селектор. Давайте подробно разберем его.

(1) Начните с определения содержимого элемента, который хотите выбрать, например, так:

```
div#greentea > blockquote
```

Здесь мы используем селектор потомка, в котором указывается, что элемент с идентификатором greentea должен быть родительским элементом для <blockquote>.

(2) Затем укажите элемент, который хотите выбрать:

```
div#greentea > blockquote p
```

контекст — элемент

Далее мы добавляем <p> как тот элемент, который нужно выбрать из контекста элемента <blockquote>. Элемент <p> должен быть потомком <blockquote>, который, в свою очередь, должен являться дочерним для элемента с идентификационным именем greentea.

(3) Затем укажите псевдоклассы или псевдоэлементы:

```
div#greentea > blockquote p:first-line { font-style: italic; }
```

контекст — элемент

Затем мы добавляем псевдоэлемент first-line, чтобы выбрать только первую строку абзаца.

Это достаточно сложный селектор! Составляйте свои селекторы используя такой же метод.

Дальше >

#2 Фреймы

HTML позволяет разделять веб-страницу на несколько *фреймов*, каждый из которых применяется для отображения вложенной страницы. Возможно, вы уже видели страницы с фреймами, которые позволяют открывать дополнительные страницы, оставляя нетронутыми верхний колонтитул и навигацию оригинальной страницы. Фреймы относятся к так называемой «старой школе HTML», так как они влекут за собой проблемы с навигацией и не способствуют удобству использования страниц. Кроме того, W3C не рекомендует их использовать. Тем не менее иногда вы будете встречать страницы с фреймами.

Чтобы создать на странице набор фреймов, используйте элементы **<frameset>** и **<frame>**:

Создает набор фреймов в строку, где первый фрейм занимает 30% ширины окна браузера, последний – 20%, а средний – все остальное место.

Вы также можете задать набор фреймов в столбец или как набор строк и столбцов.

```
<frameset rows="30%, *, 20%">
   <frame name="header" src="header.html" />
   <frame name="content" src="content.html" />
   <frame name="footer" src="footer.html" />
</frameset>
```

Для каждого фрейма используется элемент frame. Каждый элемент frame задает название фрейма и имя HTML-файла, который отображается во фрейме.

Вы также можете ссылаться на фрейм, указав его имя в элементе **<a>**. Для этого присвойте его атрибуту **target** значение в виде имени фрейма:

****новый комментарий****.

Существует также родственный элемент, который называется **<iframe>** и широко поддерживается в современных версиях браузеров. Строчный элемент **<iframe>** позволяет поместить элемент в любом месте страницы и применяется следующим образом:

Создает строчный фрейм для страницы newcontent.html.

```
<iframe name="inlinecontent" src="newcontent.html"
        width="500" height="200" />
```

Наконец, вы должны знать, что для работы с фреймами вам нужно указывать на странице DOCTYPE, который содержит слово **frameset**. Предполагается, что такой DOCTYPE переходной, поэтому невозможно одновременно использовать фреймы и придерживаться какого-нибудь строгого стандарта. Для HTML 4.01 указывайте такое определение документа:

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

а для XHTML 1.0 используйте:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
                "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

#3 Мультимедиа и флэш

Браузеры могут воспроизводить на веб-странице звук, видео или даже интерактивное содержимое, например флэш-приложения. HTML поддерживает такие типы информации благодаря элементу **<object>**, который отвечает за встраивание в веб-страницы внешнего содержимого (однако может понадобиться специальная программа для просмотра).

Кроме того, нужно упомянуть об элементе **<object>** и его близком родственнике — элементе **<embed>**. К сожалению, они так и не были полностью восстановлены со времен войны между браузерами, поэтому использовать их намного сложнее, чем это должно быть. Если вы хотите добавить на свою страницу мультимедийную информацию, мы советуем вам посетить сайт создателя нужного формата мультимедийных данных и убедиться, что вы используете рекомендуемые там настройки. Хотя встраивание мультимедийных данных сложнее, чем это должно быть, не пугайтесь и учтите, что благодаря всем знаниям в области HTML, которые вы уже имеете, вам не понадобится много времени для добавления на страницы звуков, анимации и видео.

Это простой пример вставки Quicktime-фильма с использованием элемента **<object>**:

Это открывающий тег <object>. Как видите, для добавления данных соответствующего формата требуется множество специализированных тегов и атрибутов, позволяющих указать нужную программу просмотра.

Вы можете вложить элемент <embed> в элемент <object>, чтобы предоставить браузеру несколько вариантов на выбор. Если браузер не поддерживает внешний элемент <object>, он будет пытаться работать с <embed>.

```
<object classid="clsid:02BF25D5-8C17-4B23-BC80-D3488ABDDC6B"
        codebase="http://www.apple.com/qtactivex/qtplugin.cab"
        height="200"
        width="300">
   <param name="src" value="buckaroo.mov">
   <param name="autoplay" value="true">
   <param name="controller" value="true">
   <embed height="200"
          width="300"
          src="buckaroo.mov"
          pluginspage="http://www.apple.com/quicktime/download/"
          type="video/quicktime"
          controller="true"
          autoplay="true">К сожалению, ваш браузер не поддерживает этот тип видео</embed>
</object>
```

Здесь вложен элемент <embed> для совместимости со старыми версиями браузеров.

Информация мультимедийного типа на ваших страницах может произвести неизгладимое впечатление на пользователей. Однако применение элементов **<object>** может привести к нежелательным последствиям, поэтому внимательно изучите документацию производителя программы просмотра, чтобы выяснить, каким образом эта программа встраивается в страницу.

#4 Сервисные программы для создания Веб-страниц

Теперь, когда вы знаете XHTML и CSS, самое время выяснить, хотите ли вы пользоваться такими сервисными программами, как Dreamweaver, GoLive и FrontPage. Все эти программы пытаются предоставить вам сервис типа «Что видишь, то и получаешь». Мы уверены: вы достаточно много знаете о XHTML и его поддержке браузерами, чтобы осознать, что использование сервисных программ хотя и дает быстрые результаты, все же иногда не очень удобно. Однако стоит отметить, что эти программы могут предоставить вам некоторые очень полезные возможности, даже если в основном вы пишете XHTML-код сами.

- Окно для ввода кода со встроенной функцией проверки синтаксиса и отслеживания наиболее часто встречающихся ошибок.
- Возможность предварительного просмотра и тестирования страницы перед ее размещением в Сети.
- Менеджер сайта, позволяющий приводить в порядок ваши страницы, а также синхронизирующий локальные изменения с сайтом на сервере. Обратите также внимание, что он обычно берет на себя всю работу с FTP.

Однако сервисные программы имеют и недостатки.

- Часто они не успевают за стандартами, которые поддерживаются нынешним программным обеспечением, поэтому, чтобы ваш XHTML- и CSS-код был современным, вам придется писать его самостоятельно.
- Программы не придерживаются строгих стандартов и могут позволять достаточно небрежно писать на XHTML и CSS, поэтому не забывайте проходить процедуру валидации (с помощью специальных сервисных программ для валидации).

Помните, что для создания страниц вы можете использовать обычный текстовый редактор и сложные сервисные программы вместе, поэтому старайтесь применять программы в тех случаях, когда это имеет смысл.

Некоторые сервисные программы, которые мы предлагаем рассмотреть

- Macromedia Dreamweaver.
- Adobe GoLive.
- Microsoft FrontPage.
- GNU Emacs (открытый код).

#5 Клиентские сценарии

HTML-страницы не обязательно должны быть пассивными документами: они могут иметь *исполняемое* содержимое. Такое содержимое задает режим работы ваших страниц и создается посредством написания программ на специальном языке сценариев. Существует немного подобных языков, которые работают в браузерах, и несомненный лидер среди них – JavaScript. Посмотрите на следующий фрагмент кода, чтобы получить первое представление о том, как поместить исполняемое содержимое на ваши страницы.

Это новый HTML-элемент <script>, с помощью которого можно написать код прямо внутри HTML. Обратите внимание, что мы задали тип JavaScript.

```
<script type="text/javascript">
   function validBid(form) {
      if (form.bid.value > 0) return true;
      else return false;
   }
</script>
```

Это небольшой сценарий на JavaScript, который проверяет предлагаемую пользователем цену, чтобы убедиться, что она не равна 0 долларов и менее.

В XHTML вы можете создать форму, в которой используется сценарий для проверки цены. При этом цена будет проверяться до того, как форма будет предоставлена на рассмотрение.

```
<form onsubmit="return validBid(this);" method="post" action="contest.php">
```

Это новый атрибут onsubmit элемента <form>, который запускает сценарий при нажатии на форме кнопки Submit (Отправить).

Что еще могут делать сценарии

Как вы уже поняли, валидация введенных пользователем данных – широко используемое и полезное действие, которое часто выполняется посредством JavaScript. Но это лишь малая часть того, что вы можете сделать с помощью JavaScript. На самом деле этот язык имеет доступ ко всему дереву элементов документа (к тому самому дереву, с которым мы работали в главе 3) и позволяет программным способом изменять элементы в дереве и их значения. Что это означает? Вы можете написать сценарий, меняющий вид вашей веб-страницы в зависимости от того, какие действия совершает пользователь. Рассмотрим, что можно делать, используя JavaScript.

- Создавать интерактивные игры.
- Динамически изменять изображение в то время, когда пользователь проводит над ним указатель мыши.
- Запоминать в браузере информацию о пользователе, чтобы иметь возможность вспомнить ее при следующем визите.
- Позволять пользователям выбирать различные стили страницы.
- Отображать случайную цитату из набора цитат.
- Показывать, сколько дней осталось до Рождества.

#6 Серверные сценарии

Многие сайты не создаются вручную, а генерируются веб-приложениями, работающими на сервере. Например, представьте себе систему для онлайн-заказов, в которой сервер генерирует страницы по мере того, как вы проходите все стадии заказа. Или представьте форум, где имеются страницы, сгенерированные сервером и основанные на сообщениях форума, которые хранятся в какой-то базе данных. Для обработки формы, которую вы создали в главе 14 для Starbuzz Bean Machine, мы также использовали веб-приложение.

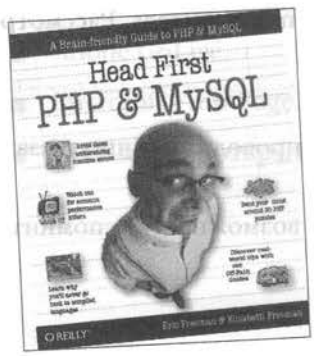
Многие хостинговые компании позволяют создавать собственные веб-приложения путем написания серверных сценариев и программ. Рассмотрим, что можно сделать с их помощью.

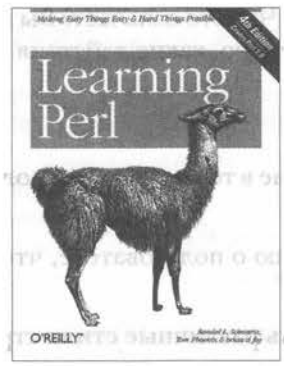
- Создать интернет-магазин, в котором будет представлен перечень предлагаемых товаров, а также предусмотрена система заказа в режиме онлайн и возможность рассчитаться кредитной картой.
- Персонализировать ваши страницы для пользователей по их предпочтениям.
- Размещать на странице последние новости, события и информацию.
- Позволять пользователям искать сайт напрямую.
- Давать посетителям возможность помогать вам в создании сайта.

Для создания веб-приложений вам нужно знать определенный серверный язык сценариев или язык программирования. Существует множество конкурирующих языков, и вы, скорее всего, услышите разные мнения о достоинствах и недостатках каждого из них. На самом деле веб-языки имеют нечто общее с автомобилями: вы можете ездить на чем угодно, и каждая модель будет иметь свои сильные и слабые стороны (цена, простота использования, размер, экономия и т. д.).

Веб-языки постоянно развиваются и совершенствуются. PHP, Python, Perl, Java Server Pages (JSPs) – это наиболее часто применяемые языки. Если вы новичок в программировании, то обратите внимание на PHP. Это, наверное, самый простой язык, который подойдет для начинающего разработчика. Если вы уже имеете какой-то опыт в программировании, попробуйте писать на JSP. Если вы более ориентированы на технологии Microsoft, вам, возможно, лучше обратиться к VB.net и ASP.net.

Предлагаем ознакомиться с некоторыми книгами, с которых вы можете начать свое обучение языкам веб-программирования.







#7 Поговорим о поисковых службах

Многие пользователи будут находить ваши веб-страницы благодаря поисковым службам (например, Google и Yahoo!). Однако в каком-то случае вы можете захотеть, чтобы сайты не отображались в списках, создаваемых поисковыми службами. Для этого можно использовать XHTML. В других случаях вы захотите сделать все возможное, чтобы ваш сайт оказывался как можно выше в списке найденных, когда пользователи ищут определенные термины. Здесь мы дадим вам несколько основных подсказок на тему, как сделать результаты работы поисковых служб наиболее выгодными для ваших страниц. Но помните, что все поисковые службы работают по-разному и каждая из них учитывает различные факторы, когда определяет порядок следования сайтов в списке результатов, создаваемом после поиска.

Как максимально высоко расположить сайт в списках, создаваемых поисковыми службами

Поисковые службы используют в своих списках встречающиеся на ваших страницах сочетания слов и фраз для определения порядка сайтов. Чтобы расположить сайт как можно выше в этих списках и помочь поисковым службам определить основную задачу вашей страницы, начните с использования двух тегов **<meta>** в элементе **<head>**: один нужен для перечисления ключевых слов, другой — для описания содержимого вашей страницы. *Ключевое слово* — это обычное слово (или несколько слов), которое описывает сайт, например «кофе» или «дневник путешествия».

```
<meta name="description" content="Это будет описанием содержимого вашей страницы.
Главные ключевые слова и фразы должны быть помещены в это описание." />

<meta name="keywords" content="ключевая фраза_1, ключевая фраза_2, ключевая фраза_3
и т. д." />
```

Многие поисковые службы считают слова в заголовках и атрибутах **alt** и **title** более значимыми, чем весь остальной текст, поэтому указывайте в них лаконичную и содержательную информацию.

Наконец, учитывается количество ссылок на ваш сайт с других страниц. Чем больше таких ссылок, тем выше в списке будет расположен ваш сайт. Итак, делайте все, чтобы другие люди как можно чаще ссылались на ваш сайт.

Как сделать так, чтобы ваш сайт не отображался в списках, создаваемых поисковыми службами

Вы можете потребовать, чтобы поисковые службы игнорировали ваши страницы, но нет никакой гарантии, что это требование будет выполнено всеми службами. Единственный способ сделать так, чтобы ваш сайт на самом деле никто не смог найти, — сделать его приватным (обсудите это с вашей хостинговой компанией). Но если вы просто хотите потребовать, чтобы ваш сайт не отображался в списках, создаваемых основными поисковыми службами, просто поместите тег **<meta>** в элемент **<head>** следующим образом:

```
<meta name="robots" content="noindex,nofollow" />
```

Этот тег <meta> говорит поисковым службам игнорировать данную страницу, а также все остальные страницы сайта, на которые на ней встречаются ссылки.

#8 Подробнее о таблицах стиля для печати

Как вы видели в главе 10, можно использовать атрибут **media** для элемента **<link>**, чтобы указать альтернативный тип устройства. Если вы зададите для этого атрибута в таблице стилей значение **print**, таблица стилей будет использоваться при печати документа. Для этого указывайте элемент **<link>** следующим образом:

```
<link rel="stylesheet" type="text/css" media="print" href="forprint.css" />
```

Атрибут media в элементе <link> говорит браузеру, что он должен использовать эту таблицу стилей, когда выводит данную веб-страницу на печать.

Это ссылка на таблицу стилей, используемую при печати. Она не будет применяться при просмотре вашей веб-страницы на экране компьютера, а будет использована только тогда, когда вы распечатываете страницу.

Далее, когда пользователь посещает вашу страницу и выбирает в своем браузере команду Print (Печать), браузер применяет таблицу стилей `forprint.css`, прежде чем распечатать страницу. Это позволяет вам стилизовать страницы таким образом, чтобы они лучше смотрелись на бумаге.

Приводим здесь несколько рекомендаций, которые лучше учитывать при разработке таблиц стилей для печати.

- В областях с текстом задайте белый цвет фона, чтобы текст лучше читался на бумаге.
- В таблице стилей, предназначенной для печати, укажите размер шрифта в пунктах, а не в пикселах, процентах или em. Размер в пунктах применяется специально для печатного текста. Обычно используются шрифты размером 12 пунктов.
- В то время как шрифты sans-serif легче читаются на экране компьютера, на бумаге удобнее читать шрифты семейства serif. Вы можете использовать таблицу стилей для печати, чтобы поменять семейство шрифтов.
- Если у вас вокруг основного содержимого страницы есть навигационные меню, врезки или другие подобные элементы, можете спрятать их для печатной версии страницы, особенно когда они не являются необходимыми для понимания основной информации. Это можно сделать, указав свойство **display** со значением **none** для любого элемента.
- Если вы позиционировали элементы на веб-странице, то, возможно, захотите убрать свойства позиционирования, чтобы содержимое распечаталось в порядке сверху вниз, что более логично при чтении информации на бумаге.
- Если вы задавали конкретную ширину для некоторых элементов веб-страницы, то, возможно, захотите сделать ее гибкой, указав поля или другие свойства. Если вся ваша веб-страница имеет определенную ширину, то она может не поместиться на бумагу целиком или будет плохо на ней смотреться.

Чтобы создать хорошую таблицу стилей для печати, посмотрите на первоначальное содержимое страницы и убедитесь, что оно хорошо выглядит в распечатанном виде, хорошо вмещается в размеры бумажной страницы и легко читается. Лучший способ удостовериться, что страница будет хорошо выглядеть в распечатанном виде, — протестировать таблицу стилей, распечатав страницу.

#9 Страницы для мобильных устройств

Вы хотите, чтобы ваши веб-страницы можно было просматривать на таких мобильных устройствах, как сотовые телефоны и карманные компьютеры (КПК)? Если да, то при создании веб-страниц вам нужно помнить о некоторых вещах. В то время как мобильные устройства становятся все более «навороченными», поддержка ими XHTML и CSS все еще несовершенна. Одни поддерживают CSS, другие — нет; одни на самом деле хорошо отображают XHTML-страницы, другие их портят. Все, что вы можете сделать, — предупредить возможные проблемы и ожидать более благоприятных времен, когда страницы на мобильных устройствах будут поддерживаться лучше.

Кроме того, используя атрибут **media** элемента **<link>**, вы можете сослаться на специфическую таблицу стилей **handheld**.

```
<link rel="stylesheet" type="text/css" media="handheld" href="formobile.css" />
```

Чтобы создать таблицу стилей для мобильных устройств, используйте атрибут media и задайте для него значение handheld.

К сожалению, поддержка таблицы стилей типа **handheld** все еще очень ограниченна, поэтому, даже если на вашей веб-странице есть ссылка на такую таблицу, это не означает, что браузер вашего телефона на самом деле будет ее использовать. Итак, вам нужно запомнить некоторые общие технические приемы дизайна страниц, чтобы ваши веб-страницы выглядели одинаково хорошо на экране как компьютера, так и мобильного устройства.

- Помните, что многие мобильные сервисы все еще ограничивают количество данных, которые могут передаваться устройству. Это веская причина писать простой, правильный XHTML-код и использовать CSS для стилизации веб-страниц.
- Делайте навигацию простой и очевидной. Это значит, что лучше использовать текстовые ссылки и избегать специальных эффектов, для которых требуется применение мыши и клавиатуры.
- Уменьшите масштаб вашей страницы настолько, насколько это возможно. Если у вас есть таблица стилей типа **handheld**, используйте ее для максимального уменьшения размеров шрифтов, полей и отступов.
- Помните, что многоколоночные разметки часто игнорируются в маленьких устройствах, поэтому внимательно следите за порядком элементов в XHTML.
- Многие мобильные устройства не поддерживают фреймы и всплывающие окна, поэтому избегайте их использования.
- Наконец, всегда тестируйте свои страницы на максимальном количестве различных мобильных устройств, чтобы узнать, как они на самом деле отображаются.

Хотя в наши дни поддержка, возможно, и ограниченна, если вы уже сегодня выработаете у себя привычку писать альтернативную таблицу стилей для типа **handheld**, вы будете хорошо подготовлены к тому дню, когда страницы на мобильных устройствах станут поддерживаться лучше.

#10 Блоги

Блоги похожи на персональные веб-страницы, только написаны в виде дневника, например как веб-дневник Тони. Многие люди, создающие блоги, используют сервисные интернет-программы, которые берут на себя заботу о деталях управления записями блога. Эти сервисные программы также предоставляют заранее заготовленные шаблоны, позволяющие выбрать из множества вариантов, как будет выглядеть ваш блог. Они предлагают различные фоновые цвета, размеры шрифтов и даже фоновые изображения. Кроме того, они дают возможность доработать выбранный вами шаблон и создать для него собственный уникальный стиль. Для этого применяется XHTML и CSS, как вы, скорее всего, уже догадались.

Это фрагменты XHTML- и CSS-кода из образца блога, взятого из популярного сервиса для публикации блогов Blogger.com. Как видите, в них использованы те же самые элементы и свойства, которые вы изучали в этой книге. И они даже придерживаются самых новых стандартов: их шаблоны написаны на XHTML 1.0 Strict, поэтому очень хорошо, что вы уже умете писать на строгом XHTML. Рассмотрим эти фрагменты подробнее...

Сервисная программа для публикации блогов использует XHTML 1.0 Strict, поэтому здесь использованы DOCTYPE и атрибуты <html>, которые вы уже видели ранее.

```
<!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>

  <title><$BlogPageTitle$></title>

  <$BlogMetaData$>
```

Эти <$...$> – переменные шаблона блога. В них вносится название блога, а также другое содержимое, изменяемое при создании блога и добавлении нового сообщения. Вы должны оставлять эти переменные неизменными, так как они необходимы для того, чтобы корректно отображать содержимое вашего блога.

```
  <style type="text/css">
    body {
      margin: 0px;
      padding: 0px;
      text-align: center;
      color: #554;
      background: #689C54 url(http://www.blogblog.com/dots_dark/bg_minidots.gif) top center repeat;
      font-family: "Lucida Grande", lucida, helvetica, sans-serif;
      font-size: small;
    }
    .
    .
    .
  </style>
</head>
<body>
  .
  .
</body>
</html>
```

Это вершина таблицы стилей, которая задает внешний вид блога. Здесь удалены поля и отступы для элемента <body>, заданы используемый по умолчанию цвет текста и фоновое изображение, а также установлены свойства шрифта.

Здесь есть множество других стилей. Каждое правило стиля контролирует такие настройки, как шрифт, используемый для записи блога, заголовки, цвета – все то, что вы привыкли стилизовать.

В XHTML хранятся все составляющие блога: заголовки, записи, даты и т. д. Каждая область содержимого также будет иметь переменную <$...$> для получения текста ваших сообщений.

Majordomo

Хостинг. Домены. Серверы.

(812) 335-35-45 (495) 727-22-78
www.majordomo.ru

Входит в пятерку крупнейших хостинг-провайдеров России.
На рынке с 2000 года. Полный комплекс услуг,
связанных с размещением Вашего сайта в сети Интернет.

Registrant

Крупнейший аккредитованный регистратор
доменных имен в зоне .ru и .su
в Северо-Западном регионе России.

(812) 335-35-45 (495) 727-22-78
www.registrant.ru

Полный комплекс услуг, связанных с регистрацией, продлением и
переоформлением Вашего домена.

(812)407-8-407

COMLINK
TELECOM

УНИВЕРСАЛЬНЫЙ ОПЕРАТОР СВЯЗИ

ТЕЛЕФОНИЯ
ИНТЕРНЕТ
ХОСТИНГ
Wi-Fi

www.comlinktel.ru