

# 15 Адаптивный веб-дизайн

Веб-дизайнерам всегда приходилось решать задачу разработки сайтов под различные размеры экрана, от дисплеев ноутбуков с разрешением по горизонтали 760 пикселей до огромных широкоэкранных мониторов. С ростом количества смартфонов и планшетных компьютеров разработка в расчете на широкий диапазон экранов с различным разрешением стала еще актуальнее. Некоторые компании зашли так далеко, что стали создавать отдельные сайты, предназначенные только для мобильных устройств (рис. 15.1, *вверху*). Но при дефиците времени, средств и технических знаний для разработки двух сайтов и программного обеспечения для веб-сервера, предоставляющего нужный сайт тому или иному устройству, сайты исключительно для мобильных устройств вам не по плечу.

К счастью, есть еще один, более простой подход, позволяющий создать один сайт, адаптируемый к устройствам с различным разрешением экрана (см. рис. 15.1, *внизу*). Эта технология, получившая название «адаптивный веб-дизайн», использует ряд различных приемов, заставляющих страницу изменять макет на основе ширины окна браузера. Например, на смартфоне страницу можно скомпоновать в одну удобную для чтения колонку, помещающуюся на узком экране (см. рис. 15.1, *внизу слева*), а на более широких мониторах поддерживать компоновку в несколько колонок (см. рис. 15.1, *внизу справа*).

## Основы адаптивного веб-дизайна

Читать веб-страницу, состоящую из четырех колонок на экране смартфона, имеющем разрешение по горизонтали 320 пикселей, очень трудно. Не менее трудно читать одну колонку текста, растянутую на 2560 пикселей огромного компьютерного монитора. *Адаптивный веб-дизайн* (термин введен Итаном Маркоттом) является попыткой решить эту проблему. Адаптивный веб-дизайн позволяет изменять всю компоновку страницы на основе ширины окна браузера (наряду с другими факторами), допуская создание наиболее легко читаемых представлений для каждого устройства, не испытывая необходимости в создании нескольких версий одного и того же сайта. Адаптивный веб-дизайн не является единой технологией или методом. В нем собраны воедино несколько методов CSS- и HTML-верстки для создания веб-страниц, макеты которых адаптируются к различным экранам.



**Рис. 15.1.** На смартфоне страница может появиться в виде длинной колонки, в окне браузера компьютера та же самая страница будет использовать всю ширину экрана и включать в себя несколько колонок и более крупные иллюстрации

#### ПРИМЕЧАНИЕ

Свои подходы к адаптивному веб-дизайну Итан Маркотт излагает в книге «Отзывчивый веб-дизайн». Можно также прочитать статью, посвященную адаптивному веб-дизайну, этого же автора по адресу [tinyurl.com/c68oc3m](http://tinyurl.com/c68oc3m).

В адаптивном веб-дизайне объединены три основные концепции: гибкие сетки для компоновки, гибкая среда для изображений и видеоконтента и медиазапросы

CSS, предназначенные для создания различных стилей для экранов с разным разрешением. Гибкие сетки позволяют избавиться от фиксированных дизайнов. Поскольку у смартфонов экраны бывают с самым разным разрешением, создавать страницы с фиксированной шириной нет никакого смысла, тут нужна страница, способная расширяться и сужаться, чтобы поместиться на экране конкретного устройства. Создание гибкой среды для передачи мультимедийного контента позволяет подбирать масштаб изображениям и видеоматериалам, чтобы они поместились на соответствующем экране — большие фотографии на больших мониторах, фотографии поменьше на небольших экранах и т. д.

И наконец, медиазапросы являются технологией каскадных таблиц стилей, позволяющей на основе создавшихся условий отправлять браузеру различные стили. Например, для экрана, имеющего разрешение по горизонтали менее 480 пикселей, можно отправить один набор стилей, а для экрана с разрешением по горизонтали 760 пикселей — другой. Но одной только шириной дело не ограничивается: можно разрабатывать стили, применяемые только к планшетным устройствам при альбомном просмотре или к устройствам с экранами с высокой плотностью пикселей (например, к смартфонам iPhone и планшетам iPad с дисплеем Retina).

## Создание адаптивного дизайна веб-страницы

Если у вас есть смартфон под управлением операционной системы iOS или Android, присмотритесь к нему повнимательнее. Откройте браузер и перейдите на страницу [nytimes.com](http://nytimes.com). Вы попадете на версию сайта, предназначенную специально для мобильных устройств — [mobile.nytimes.com](http://mobile.nytimes.com). Тем не менее на ней доступна ссылка на версию сайта, предназначенную для компьютеров. Вы увидите контент, подобный показанному на рис. 15.2 (если только издание *New York Times* с момента написания книги не сделало свой сайт адаптивным). Это крупноформатный, многоколоночный дизайн, втиснутый в небольшое пространство экрана смартфона. Поскольку производители смартфонов понимают, что большинство сайтов созданы для экранов компьютерных мониторов, они заставили свои браузеры вести себя немного непривычно для вас. Мобильные браузеры не отображают страницу на все 100 %; если бы они это сделали, то страница шириной 960 пикселей не поместилась бы на экране и вы бы увидели только часть этой страницы. Затем, чтобы увидеть всю страницу, вам пришлось бы смахивать экран в разных направлениях. Вместо этого, чтобы страница поместилась на экране, браузеры смартфонов уменьшают масштаб. Конкретный коэффициент уменьшения варьируется в зависимости от характеристики той или иной модели смартфона. Например, браузер Safari смартфона iPhone работает так, будто экран действительно имеет разрешение по горизонтали 980 пикселей, и уменьшает страницу, чтобы она уместилась в этих 980 пикселях.

Впрочем, подобное поведение мобильных браузеров позволяет неплохо справиться с большинством сайтов, но с адаптивным веб-дизайном оно сочетается не очень хорошо. Так как адаптивные сайты предназначены для получения хорошего вида на смартфонах, уменьшать масштаб на экране не нужно, поскольку текст станет слишком мелким для чтения. К счастью, существует довольно простой способ

отмены такого поведения в браузерах мобильных устройств. Нужно в раздел заголовка вашей веб-страницы добавить следующий код (самое подходящее место для этого будет непосредственно перед элементом `title`):

```
<meta name="viewport" content="width=device-width">
```

Метатеги языка HTML предоставляют дополнительную информацию о содержимом страницы и могут передавать браузерам дополнительные инструкции о способах отображения страницы на экране. В данном случае `viewport` обозначает экран браузера, а для атрибута `content` устанавливается ширина окна браузера, равная разрешению по горизонтали экрана смартфона. То есть браузерам мобильных устройств, склонным к уменьшению масштаба, предписывается этого не делать, настроив ширину экрана на текущее разрешение по горизонтали экрана смартфона.



**Рис. 15.2.** Сайты, разработанные для браузеров компьютеров, на смартфонах выглядят как небольшие почтовые марки. Посетителям приходится увеличивать масштаб изображения и, чтобы все прочитать, перетаскивать страницу по экрану

## Медиазапросы

Язык CSS включает понятие *медиазапросов*. Медиазапросы позволяют назначать страницам стили на основе размера (ширины и высоты) окна целевого браузера. С помощью этого метода можно создавать пользовательские стили для браузеров смартфонов, планшетов и компьютеров и тем самым настраивать представление вашего сайта таким образом, чтобы он выглядел наилучшим образом на экране каждого типа устройств.

Весь смысл адаптивного дизайна состоит в том, чтобы дать посетителям вашего сайта наиболее читаемое и привлекательное представление. Обычно это означает настройку дизайна под наилучший вид в окнах браузеров различной ширины. Многие разработчики ориентируются на три наиболее распространенных устройства просмотра веб-контента: смартфоны, планшеты и компьютеры. При условии широкого диапазона размеров экранов этих устройств (могут быть смартфоны как с маленьким экраном, так и с большим, 8-дюймовые и 10-дюймовые планшетные компьютеры и т. д.) единого значения ширины области просмотра для всех этих устройств не существует. И целью является нормальный внешний вид страницы при разном разрешении экрана по горизонтали. Вы можете протестировать различные дизайнерские решения в окнах разной ширины, чтобы увидеть, когда четырехколоночный дизайн должен превращаться в одно- или двухколоночный.

## Стратегии использования медиазапросов

Хотя для решения о внесении изменений в дизайн, позволяющего придать ему наиболее удачный внешний вид на различных устройствах, нами рекомендуется метод проб и ошибок, существует ряд общих дизайнерских изменений, на которые обычно нацелены медиазапросы.

- **Изменение количества колонок.** Несколько колонок контента неплохо смотрятся на больших мониторах (и даже на планшетных компьютерах в альбомном режиме), но не подходят для смартфонов. Кроме того, четырех колонок, скорее всего, будет многовато для большинства планшетных компьютеров с книжной ориентацией экрана, поэтому сведение дизайна страницы к двум или трем колонкам, по всей видимости, вполне подходит для медиазапросов, нацеленных на планшетные компьютеры. Исключение обтекаемых элементов в стилях медиазапросов, направленных на планшетные компьютеры, позволяет складывать контейнеры с контентом страницы друг на друга. Эта технология будет опробована в практикуме этой главы.
- **Гибкая ширина.** Фиксированный дизайн можно использовать для браузеров компьютеров — именно так годами и поступали дизайнеры. Но на более узких экранах планшетных компьютеров и смартфонов фиксированные элементы не поместятся. Страница шириной 960 пикселей будет слишком велика для смартфонов с экранами с разрешением по горизонтали 320 или 480 пикселей. Для смартфонов и планшетов более удачным подходом будет присвоение свойству `width` контейнеров `div` с контентом значения `auto` или `100%`. Эти установки превратят дизайн вашей страницы из фиксированного в резиновый,

или гибкий. Иными словами, независимо от разрешения экрана смартфона `div`-контейнеры поместятся на нем полностью. Если держать смартфон iPhone в книжной ориентации (при разрешении экрана по горизонтали 320 пикселей), а затем быстро повернуть в альбомную (изменив тем самым разрешение экрана по горизонтали до значения 480 пикселей), `div`-контейнеры со свойством `width: auto` или `width: 100%` просто изменят свой размер, чтобы поместиться в новом пространстве.

- **Сжатие пустых пространств.** Обширные промежутки между заголовками, изображениями и другими элементами страницы позволяют свободно компоновать дизайн для огромных мониторов, но приводят к хаосу и нерациональному использованию пространства на небольших экранах смартфонов, принуждая посетителей прибегать к прокрутке. Сужение полей и отступов позволяет разместить на таких небольших экранах больше полезной информации.
  - **Настройка размеров шрифта.** Контраст между крупными, полужирными заголовками и набранным мелким шрифтом основным текстом неплохо выглядит на компьютерных мониторах. Но на портативных устройствах излишне крупные заголовки труднее читаются и совершенно необоснованно занимают полезное пространство. А вот незначительное увеличение шрифта обычного текста на смартфонах зачастую облегчает его чтение. Иначе говоря, создавая стили медиазапросов, обращайте внимание на размеры шрифтов.
  - **Изменение навигационных меню.** У вас может быть красиво оформленная горизонтальная панель навигации, занимающая всю верхнюю часть веб-страницы и состоящая из десятка кнопок, направляющих посетителей к разным разделам вашего сайта. К сожалению, по мере сужения окна браузера эти кнопки могут не поместиться на весь экран. Они будут разнесены на две, три и более строки. Пример такого явления вы увидите в практикуме в конце главы. Возможно, в том, что навигационная панель займет не одну, а несколько строк экрана, не будет ничего страшного, но ведь эта панель может занять в верхней части страницы слишком много места, заставляя пользователей задействовать прокрутку, чтобы добраться до первых строк реального контента.
- Увы, каскадные таблицы стилей не предлагают простых и понятных решений. На многих сайтах для динамического превращения навигационного меню в раскрывающийся список используется JavaScript, тогда это меню занимает небольшое пространство экрана (чтобы этому научиться, обратитесь по адресу [tinyurl.com/q96ertm](http://tinyurl.com/q96ertm)). Но есть и другие решения. Обзор различных подходов, применяемых на некоторых сайтах, представлен на сайтах [tinyurl.com/l9ox9sv](http://tinyurl.com/l9ox9sv) и [tinyurl.com/oyvmpd6](http://tinyurl.com/oyvmpd6).

- **Скрытие лишнего контента на портативных устройствах.** Многие разработчики скрывают некоторый контент на мобильных версиях сайтов. На компьютерном мониторе просмотр нескольких колонок и сотен строк текста дается довольно легко, а вот большое количество информации на смартфоне может показаться совершенно излишним. Каскадные таблицы стилей можно использовать для того, чтобы скрыть контент, который, на ваш взгляд, не нужно показывать пользователям мобильных устройств, для чего следует присвоить свойству `display` значение `none`.

Но все же нужно иметь в виду, что, скрывая контент, вы отстраняете посетителя от той информации, которая представляется на вашем сайте. Тем, кто ранее посещал ваш сайт с компьютера, а теперь посещает его со смартфона, будет крайне неприятно увидеть, что совсем недавно просматриваемая ими важная информация теперь куда-то исчезла. Кроме того, даже если вы скроете контент с помощью каскадных таблиц стилей, сам HTML-код никуда не денется, заставляя смартфон впустую тратить время и трафик на загрузку неиспользуемого HTML-кода.

- **Использование фоновых изображений.** Если поместить на экран 960-пиксельный баннер, то ни один смартфон не покажет его без уменьшения масштаба. Можно, конечно, представить достаточно небольшое изображение, способное поместиться на экране смартфона, или же воспользоваться вместо него фоновыми изображениями. Можно, например, создать div-контейнер и добавить к нему следующий класс: `<div class="logo">`. Затем в таблице стилей для браузера компьютера установить ширину/высоту div-контейнера, соответствующую размеру большого логотипа, используя свойство `background-image` для вставки изображения в фон. Например:

```
.logo {  
  width: 960px;  
  height: 120px;  
  background-image: url(images/large_logo.png)  
}
```

Затем можно добавить еще один стиль, используемый смартфонами, который изменяет размеры div-контейнера и устанавливает другое фоновое изображение:

```
.logo {  
  width: 100%;  
  height: 60px;  
  background-image: url(images/small_logo.png)  
}
```

В разделе «Адаптивные изображения» далее в этой главе вы научитесь масштабировать изображения, вставляемые в HTML-код с помощью элемента `img`, чтобы они помещались в окна браузеров различной ширины.

## Создание точек останова

Медиазапросы позволяют отправлять браузерам различные стили на основе ширины окон этих браузеров. Например, браузеру можно сообщить следующее: «Если твой экран шире 480 пикселей, применяй те стили» или «Если твой экран шире 480 пикселей, но уже 769 пикселей, применяй эти стили». Различные значения ширины, указываемые вами, — 480, 769 и т. д. — в адаптивном дизайне часто называют *точками останова*. А вообще-то, с каких значений нужно начинать разбивать дизайн на точки останова?

Проще всего это определить, если взять готовый дизайн для компьютера и открыть страницу в браузере. Нажав и удерживая кнопку мыши на краю окна браузера,



перетаскивайте указатель и медленно уменьшайте ширину окна. В определенный момент времени страница приобретет совершенно неприглядный вид. К примеру, станет тесно ее четырем колонкам. Та позиция, в которой дизайн теряет приемлемый внешний вид, становится хорошим кандидатом на точку останова, то есть этот размер вполне подходит для определения нового медиазапроса и для загрузки новых стилей, чтобы удалить одну или две колонки.

Нередко создают три набора медиазапросов для трех различных точек останова — один для смартфонов, другой для планшетов, а третий — для компьютерных мониторов. Конкретные значения точек останова зависят от конкретного дизайна (а также от конкретного устройства), но чаще всего отправной точкой служит экран, имеющий разрешение по горизонтали меньше 480 пикселей, который получает один набор стилей, экран с разрешением по горизонтали между 481 и 768 пикселями получает второй набор стилей, а устройства с разрешением по горизонтали больше 768 пикселей получают дизайн, предназначенный для компьютеров. Но все это в конечном счете остается на ваше усмотрение. Некоторые дизайнеры допускают расширение зоны планшетных компьютеров до 1024 пикселей, а стили для компьютеров начинают отправлять тем браузерам, ширина окна которых превышает 1024 пикселя.

Некоторые дизайнеры даже доходят до определения четырех или пяти точек останова, чтобы их творения хорошо смотрелись в более широком диапазоне экранов. Подробности создания этих точек останова с использованием медиазапросов будут даны в подразделе «Создание медиазапросов» далее.

## Приоритет мобильных устройств или компьютеров?

Следует рассмотреть еще один вопрос: с расчетом на какое устройство нужно приступать к разработке дизайна? Вам не нужно создавать три отдельных набора стилей, по одному для каждого из устройств, на которые вы нацелены. Вы можете и должны сначала создать *исходный* дизайн, то есть дизайн, работающий без медиазапросов. Затем можно создать стили медиазапросов для замены исходных стилей и переформатирования страницы под конкретное разрешение экрана по горизонтали. Для этого существует два основных подхода.

- **Стратегия Desktop first:** предпочтение компьютерам. Дизайн сайта можно разрабатывать с прицелом на компьютеры. Создайте все требуемые колонки. Отработайте дизайн до совершенства, чтобы он хорошо выглядел на большом мониторе. Теперь это будет ваш исходный дизайн, и все стили можно собрать во внешнюю таблицу стилей и привязать ее к страницам вашего сайта обычным образом. Затем добавьте медиазапросы для планшетов и смартфонов. Стили этих медиазапросов будут подстраивать «компьютерный» дизайн под новые условия — удалять колонки, уменьшать шрифт заголовков и т. д.
- **Стратегия Mobile first:** предпочтение мобильным системам. Можно сделать все наоборот и сначала разработать дизайн для мобильных систем. Теперь в обычную внешнюю таблицу помещаются основные стили, предназначенные для



устройств с малыми экранами, а потом в медиазапросах дизайн дорабатывается для планшетов и компьютеров путем добавления колонок и других изменений для больших экранов.

Какой бы метод ни был выбран, нужно использовать обычную внешнюю таблицу стилей, привязанную к веб-странице обычным образом. В эту таблицу включаются все стили, являющиеся общими применительно к *различным* устройствам. Например, для всех версий сайта требуется одинаковая цветовая палитра и одинаковые шрифты. Можно также использовать одинаковые стили для ссылок, изображений и других HTML-элементов. Иными словами, вам не нужно создавать для каждого устройства три абсолютно отдельных набора стилей, начните с одного набора, применяемого ко всем браузерам, как смартфонов, так и планшетов и компьютеров, а затем уточните дизайн для устройств, на которые нацелены медиазапросы.

## Создание медиазапросов

*Запрос*, по сути, представляет собой вопрос, заданный браузеру: «Равна ли ширина окна браузера 480 пикселям?» Если ответ положительный, браузер использует таблицу стилей именно для устройства с данным разрешением экрана по горизонтали (предоставляемая вами таблица стилей, рассмотренная ранее). Код, выполняющий эту задачу, схож с кодом привязки любой другой внешней таблицы стилей:

```
<link href="css/small.css" rel="stylesheet" media="(width: 480px)">
```

К этой стандартной ссылке на таблицу стилей добавился еще один атрибут *media*, где определены условия, при которых браузер использует указанную таблицу. В примере выше браузер загружает внешнюю таблицу стилей *small.css*, когда посетитель просматривает ваш сайт с помощью браузера, ширина окна которого составляет 480 пикселей. Скобки, в которые заключен запрос, — *(width: 480px)* — обязательны. Если их не указать, браузер запрос проигнорирует.

Конечно, 480 пикселей очень точное значение, а как быть, если посетитель пользуется смартфоном с экраном разрешением по горизонтали, скажем, 300 пикселей? Поэтому рекомендуется использовать в медиазапросе диапазон значений. Например, может понадобиться применить конкретный стиль для экранов, разрешение по горизонтали которых *меньше или равно* 480 пикселям. Это можно сделать с помощью следующего кода:

```
<link href="css/small.css" rel="stylesheet" media="(max-width:480px)">
```

Запись *(max-width:480px)* эквивалентна высказыванию «для экранов, имеющих разрешение по горизонтали не более 480 пикселей». Поэтому стили из файла *small.css* будут применены, к примеру, к экранам с разрешением по горизонтали 480, 320 и 200 пикселей.

Существует также вариант *min-width*, определяющий, имеет ли браузер минимальную указанную ширину окна. Этот вариант применяется при нацеливании на устройство, экран которого больше смартфона или планшета. Например, чтобы применить

стили на устройствах с разрешением по горизонтали более 768 пикселей, что превосходит разрешение экрана по горизонтали многих планшетных компьютеров, нужно использовать следующий код:

```
<link href="css/large.css" rel="stylesheet" media="(min-width:769px)">
```

Чтобы эта таблица стилей была применена, окно браузера должно быть шириной не менее 769 пикселей, что на 1 пиксел больше, чем 768 пикселей, составляющих разрешение экрана по горизонтали ряда планшетных устройств.

И наконец, можно установить как максимальное, так и минимальное значение ширины экрана целевых устройств, чтобы в них попадали браузеры устройств *между* смартфонами и компьютерами. Например, чтобы создать набор стилей для планшетного компьютера, имеющего экран с разрешением по горизонтали 768 пикселей, можно воспользоваться следующим CSS-кодом:

```
<link href="css/medium.css" rel="stylesheet" media="(min-width:481px) and (max-width:768px)">
```

Иными словами, окно браузера должен быть шириной не менее 481 и не более 768 пикселей. К примеру, файл `medium.css` не будет применяться на смартфонах с разрешением экрана по горизонтали 320 пикселей, а также на компьютерных мониторах с разрешением экрана по горизонтали 1024 пикселя.

---

#### ПРИМЕЧАНИЕ

Медиазапросы каскадных таблиц стилей способны не только на проверку ширины окна браузера. Текущие стандарты медиазапросов позволяют проверять высоту и ориентацию (то есть узнать, в каком положении находится смартфон или планшет посетителя: в горизонтальном или вертикальном) и даже определять типа экрана устройства: цветной или монохромный. Получить дополнительные сведения о медиазапросах можно на сайте Консорциума W3C по адресу [tinyurl.com/3xjdbg](http://tinyurl.com/3xjdbg).

---

## Добавление медиазапросов в таблицу стилей

Показанная выше технология является одним из способов использования медиазапросов с элементом `link` для загрузки различных таблиц стилей на устройствах с экранами с разным разрешением по горизонтали. Но медиазапросы можно также добавлять и внутри единой таблицы стилей. Возможно, вам захочется сделать это, чтобы не пришлось, к примеру, добавлять в HTML-файл несколько элементов `link`, или понадобится хранить все стили медиазапросов в основной таблице стилей. Большинство веб-дизайнеров используют этот подход вместо нескольких отдельных файлов для каждого медиазапроса.

Существует два способа добавления медиазапросов в таблицу стилей.

- **Использование правила `@import`.** Эта технология уже рассматривалась в главе 2. Правило `@import` позволяет загружать дополнительные внешние таблицы стилей во внутреннюю либо во внешнюю таблицу стилей. Правило `@import` можно также использовать с медиазапросом. Предположим, нужно загрузить внешнюю таблицу стилей `small.css`, содержащую стили для экранов с разрешением по горизонтали не более 320 пикселей. Для этого добавьте следующее правило `@import` непосредственно в таблицу стилей:

```
@import url(css/small.css) (max-width:320px);
```

## ПРИМЕЧАНИЕ

Правила `@import` должны помещаться в начало таблицы стилей. Их нельзя указывать после каких-либо стилей. В результате могут возникать проблемы с каскадностью, при которых стили, определенные во внешней таблице стилей и загруженные с помощью правила `@import`, будут замещаться более поздними стилями в таблице стилей. Этой проблемы можно избежать созданием одной внешней таблицы стилей, которая содержит только правила `@import`. Первая из них приведет к загрузке основной таблицы стилей, предназначенной для всех устройств, а вторая и третья приведут к загрузке таблиц стилей с использованием медиазапросов:

```
@import url(css/base.css); /* без медиазапроса, для всех */
@import url(css/medium.css) (min-width:481px) and (max-width:768);
@import url(css/small.css) (max-width: 480px);
```

- **Внедрение медиазапроса в таблицу стилей.** Медиазапрос можно также внедрить непосредственно в таблицу стилей:

```
@media (max-width: 480px) {

    body {

        /* сюда помещаются свойства стиля*/

    }

    .style1 {

        /* сюда помещаются свойства стиля */

    }

}
```

Правило `@media` функционирует как своеобразный контейнер для всех стилей, соответствующих запросу. Следовательно, в этом примере стили `body` и `.style1` применяются только к тем устройствам, разрешение экрана по горизонтали которых не превышает 480 пикселей. Использование встраиваемых правил `@media` позволяет организовать все ваши стили в одну таблицу стилей. Рекомендуется начинать внешние таблицы с тех стилей, которые не содержатся в медиазапросах, отдавая сначала предпочтение стилям либо для компьютеров, либо для мобильных устройств, а затем добавляя медиазапросы для всех остальных устройств. Этот подход наиболее общий, используемый большинством веб-дизайнеров.

## Основная структура таблицы стилей

Можно сказать, что существует множество различных способов использования медиа-запросов, нацеленных на различные устройства: с приоритетом, отдаваемым компьютерам, с приоритетом, отдаваемым мобильным системам, в отдельных таблицах стилей, в единой таблице стилей и т. д. По мере накопления опыта вы поймете, какой из способов больше подойдет для вашего проекта. Но вначале нужно всегда создавать единую внешнюю таблицу стилей, включая в нее стили для компьютерных

мониторов, а затем добавляя медиазапросы с изменениями этого дизайна основного уровня под планшетные устройства и смартфоны. Схематично структура для такого файла должна иметь следующий вид:

```
/* Сброс стилей браузера */
/* стили для компьютеров и базовые стили для всех устройств */
body {
    /* свойства тела страницы */
}

/* только для устройств со средним размером экрана */
@media (min-width: 481px) and (max-width:768px) {
    body {
        /* только для планшетов */
    }
}

/* только для устройств с малым размером экрана */
@media (max-width:480px) {
    body {
        /* только для смартфонов */
    }
}
```

Вы увидите эту структуру в действии в практикуме в конце текущей главы, а базовую таблицу стилей можно найти в файле `desktop_first.css`, который находится в папке 15 заданий практикума по адресу [github.com/mrightman/css\\_4e](https://github.com/mrightman/css_4e). Даже притом, что в этот файл включены медиазапросы, он содержит обыкновенную таблицу стилей, которая привязывается к веб-странице обычным образом. Например, если сохранить этот файл в его текущем виде, нужно добавить в раздел заголовка веб-страницы следующий HTML-код:

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

## Приоритет мобильных устройств

Если вы решили отдать предпочтение мобильным системам, то сначала нужно создать набор стилей, нацеленный на браузеры мобильных систем, а затем добавить медиазапросы для изменения дизайна под браузеры планшетов и компьютеров. Основная структура таблицы стилей при таком подходе будет иметь следующий вид:

```
/* Конец кода сброса стилей браузера */

/* сброс стилей браузера */
/* стили для мобильных устройств и базовые стили для всех устройств */
body {
    /* свойства тела страницы */
}
```

```
/* только для устройств со средним размером экрана */
@media (min-width: 481px) and (max-width:768px) {
  body {
    /* только для планшетов */
  }
}

/* только для устройств с крупным размером экрана */
@media (min-width:769px) {
  body {
    /* только для компьютеров */
  }
}
```

Начальную основную таблицу стилей можно найти в файле `mobile_first.css`, который находится в папке 15 с примерами для этой главы.

## Гибкие сетки

Не стоит поддаваться соблазну разработки отдельных фиксированных дизайнов для поддержки конкретных устройств, к примеру шириной 375 пикселей для iPhone 6, шириной 768 пикселей для планшетов iPad в книжной ориентации и шириной 1000 пикселей для компьютерных мониторов. Не делайте этого. Хотя смартфоны iPhone 6 и приобрели широкую популярность, на них свет клином не сошелся. Есть еще и другие модели iPhone, и смартфоны под управлением операционной системы Android, имеющие всевозможные формы и размеры, а стало быть, и разрешение экрана. Не сомневаюсь, что вам попадались и многие другие устройства с необычными размерами, да и планшетные устройства имеют весьма разнообразные разрешения экрана. То есть универсального значения ширины страниц для смартфонов и планшетных устройств попросту нет, поэтому лучше всего создавать страницы с изменяемой шириной.

Основным компонентом адаптивного веб-дизайна являются гибкие сетки. Это не что иное, как резиновый макет, который рассматривался в главе 12 и у которого общая ширина страницы изменяется, чтобы поместиться на экранах с разным разрешением по горизонтали. В большинстве случаев это означает, что свойству ширины присваивается значение 100%. Но с учетом пользователей компьютеров, возможно, потребуется использовать свойство `max-width`, чтобы страница не становилась крайне широкой на мониторах с высоким разрешением.

Кроме того, в процентах должна измеряться ширина и отдельных колонок, вместо пикселей или единиц `em`. Колонки по отдельности также должны становиться шире или уже, чтобы вписываться в изменяющуюся ширину страницы.

### СОВЕТ

---

В следующей главе вы познакомитесь с некоторыми бесплатными системами гибких сеток. Фактически с помощью одной из самых популярных систем под названием `Skeleton` очень легко создавать многоколоночные веб-страницы на основе гибких сеток.

---

Предположим, что нужно создать двухколоночный дизайн, в котором первая колонка занимает одну треть ширины страницы, а вторая — две трети. Можно начать с создания следующего HTML-кода:

```
<div class="columns">
  <div class="one-third">
    ...контент...
  </div>
  <div class="two-thirds">
    ...контент...
  </div>
</div>
```

Затем можно добавить несколько стилей для создания резинового дизайна:

```
.columns {
  width: auto; /* same as 100% */
  max-width: 1200px;
}
.columns:after {
  content: "";
  display: table;
  clear: both;
}
.one-third {
  float: left;
  width: 33%;
}
.two-thirds {
  float: left;
  width: 67%;
}
```

Первый стиль — `.columns` — устанавливает ширину `div`-контейнера, содержащего колонки. Присвоение свойству `width` значения `auto` — то же самое, что и установка значения `100%`, поскольку свойство `max-width` будет ограничивать блок от приобретения слишком большой ширины. Второй стиль — `.columns:after` — помогает управлять двумя обтекаемыми колонками (подробности, как это работает, можно найти в разделе «Решение проблем с обтекаемыми элементами» главы 13). И наконец, последние два стиля устанавливают ширину двух `div`-контейнеров равной 33 % (в ширину одной трети его контейнера) и 67 % (две трети), а также выравнивают их по левому краю, чтобы они появлялись рядом.

## Значимость HTML-кода

В книжной ориентации смартфона на его экране, при условии сохранения читабельности страницы, попросту не хватит пространства не только для трех, а даже и для двух колонок контента. Поэтому многие разработчики для отображения контента страницы на экране смартфона компонуют его в одну большую колонку. Для этого нужно удалить из созданных колонок все обтекаемые элементы. Например, если

создается трехколоночный дизайн для компьютерных мониторов и в нем используются обтекаемые элементы для позиционирования колонок друг рядом с другом, нужно присвоить свойству `float` таких элементов значение `none`. Тогда они отобразят HTML-контент в обычном порядке — один блочный элемент за другим.

Такое поведение придает порядку следования исходного HTML-кода особую важность. Например, у страницы могут быть две боковые панели, первая с перечнем ссылок на родственные сайты, а вторая — с рекламой товаров, продаваемых вашей компанией. При этом основной контент, ради которого, собственно, и посещают вашу страницу, содержится в средней колонке. Один из способов компоновки всего этого по колонкам заключается в выравнивании первой боковой панели по левому краю, а второй боковой панели — по правому краю, позволяя основной колонке обтекать две прочие колонки и находиться в центре.

В понятиях HTML это будет означать, что сначала отображаются два `div`-контейнера боковых панелей, а затем следует элемент, содержащий основной контент. Если адаптировать эту страницу под мобильные устройства, удалив указание на то, что боковые панели являются обтекаемыми элементами, то получится, что две боковые панели отобразятся *ранее* основного контента. Вашей аудитории придется прокручивать страницу вниз, чтобы пропустить рекламу и ссылки и добраться до нужного ей контента.

Лучшим решением было бы поместить контейнер с основным контентом перед боковыми панелями. Как уже упоминалось в разделе «Использование обтекаемых элементов при верстке» главы 13, этот метод может потребовать добавления дополнительных контейнеров и выравнивания всех элементов, включая `div`-контейнер с основным контентом.

Одним словом, нужно соблюдать соответствующий порядок следования HTML-элементов, прежде чем создавать многоколоночный дизайн для компьютеров. Проще всего понять, что происходит, если просмотреть страницу без применения к ней каскадных таблиц стилей. Тогда вы увидите все `div`-контейнеры и другие блочные элементы в порядке их следования и сможете понять, как будет выглядеть страница в виде одной колонки на экране смартфона.

#### ПРИМЕЧАНИЕ

---

Как можно создать трехколоночный дизайн для компьютеров при условии размещения наиболее важного контента в верхней части страницы, показано в практикуме в конце этой главы.

---

## Сброс блочной модели

Как уже объяснялось в подразделе «Предотвращение выпадений обтекаемых элементов» раздела «Решение проблем с обтекаемыми элементами» главы 13, при ограничении ширины в процентном отношении возникает угроза выпадения обтекаемых элементов, если общая ширина колонок в одном ряду превышает 100 %, из-за чего последняя колонка выпадает под другие колонки. Из-за способа, используемого браузерами для вычисления ширины элементов, увеличение толщины границы по периметру `div`-контейнера или добавление к нему внутренних отступов приводит к тому, что ширина `div`-контейнера на экране становится больше указанной в каскадной таблице стилей.



Например, если для одной колонки указана ширина 33 %, а для другой — 67 % (как в примере, приведенном ранее), они должны уместиться рядом друг с другом, поскольку их общая ширина составляет 100 %. Но если к колонке добавлена граница толщиной 1 пиксел, то ее общая ширина станет 100 % + 2 пиксела (учитываются левая и правая границы). Теперь эта колонка будет слишком большой, чтобы поместиться в окне, и вторая колонка выпадет под первую.

Существует несколько способов преодоления этого затруднительного положения, и все они были рассмотрены в подразделе «Предотвращение выпадений обтекаемых элементов» раздела «Решение проблем с обтекаемыми элементами» главы 13. Но наиболее очевидным решением станет предписание браузеру *засчитывать* в общую ширину и высоту элемента границы и отступы в качестве части его вычислений в рамках блочной модели. То есть браузер можно инструктировать включать границы и отступы в состав свойства `width`, чтобы добавление дополнительных отступов или границ не приводило к увеличению ширины (или высоты) элемента. Поскольку такой подход неплохо бы применить ко всем элементам, лучше воспользоваться универсальным селектором, позволяющим сбросить исходные параметры блочной модели для всех элементов, имеющихсся на странице:

```
* {  
  box-sizing: border-box;  
}
```

Лучше всего поместить этот стиль в код каскадной таблицы стилей, используемый для сброса исходных установок страницы (см. раздел «Управление каскадностью» главы 5).

## Преобразование фиксированного макета в гибкие сетки

При разработке совершенно нового дизайна подбор процентных соотношений несложен. В конце концов, если нужны четыре колонки одинаковой ширины, для каждой из них устанавливается значение 25%:

```
width: 25%;
```

Но если приходится иметь дело с фиксированным дизайном и нужно преобразовать его в резиновый, ситуация усложняется. Для начала представим, что вы разработали страницу с фиксированной шириной 960 пикселей. Либо контент страницы заключен в элемент `div`, для которого задана ширина 960 пикселей, либо такая ширина установлена для элемента `body`. В любом случае теперь нужно, чтобы этот контейнер был полностью резиновым. Для этого достаточно изменить код:

```
width: 960px;
```

на

```
width: auto;
```

Присвоение свойству `width` элемента значения `auto` — практически то же самое, что и использование кода `width: 100%;`; ширина этого элемента будет идентична ширине его контейнера.

Затем нужно преобразовать значения свойств `width` колонок из пикселей в проценты. Чтобы упростить вычисление, гуру адаптивного веб-дизайна Итан Маркотт придумал замечательную формулу: *цель / контекст = результат*. Проще говоря: «возьмите ширину элемента, подвергаемого преобразованию (в пикселах), и разделите ее на ширину контейнера этого элемента (в пикселах)». В результате получится дробное значение, которое нужно перевести в проценты.

Рассмотрим пример. Предположим, что на этой странице шириной 960 пикселей имеются две колонки: боковая панель шириной 180 пикселей и основная колонка шириной 780 пикселей. В CSS-файле имеется следующий код:

```
.sidebar {  
  float: left;  
  width: 180px;  
}  
.main {  
  float: left;  
  width: 780px;  
}
```

Конечно, в нем присутствует множество другого CSS-кода, задающего границы, фоновые цвета и т. д., но в данном случае нас интересуют только свойства ширины. Приступая к преобразованию боковой панели, нужно взять значение ее ширины, равное 180 пикселям, и разделить на значение ширины контейнера этой боковой панели, то есть 960 пикселей. Получившийся результат — 0,1875 — нужно умножить на 100 и получить процентное значение: 18,75 %. Точно так же для основной колонки: 780 делится на 960 и получается 0,8125. При умножении этого результата на 100 получаем 81,25 %. То есть нужно внести следующие изменения в код:

```
.sidebar {  
  float: left;  
  width: 18.75%;  
}  
.main {  
  float: left;  
  width: 81.25%;  
}
```

Полученные значения округлять не следует. То есть не превращайте 18,75 % в 19 %, поскольку это, скорее всего, приведет к выпадению обтекаемого элемента. Браузеры хорошо справляются с десятичными значениями, и вы спокойно можете использовать любые значения, возвращаемые калькулятором. Вполне допустима, скажем, ширина, заданная значением 25,48488 %.

То же самое применимо и к вложенным колонкам. Предположим, что в показанной выше основной колонке имеется один раздел, состоящий из двух обтекаемых `div`-контейнеров, создающих две дополнительные колонки внутри основной. Оба `div`-контейнера имеют одинаковую ширину, равную 390 пикселям, поэтому для вычисления их ширины в процентном отношении берется значение в пикселах — 390 — и делится на ширину их контейнера, которая в данном примере составляет 780 пикселей. В результате получается значение 0,5, умножение которого на

100 выдаст 50 %. (Но, по сути, эти вычисления не нужны. Ведь у вас имеются две расположенные рядом колонки одинаковых размеров, и вы знаете, что каждая из них занимает половину доступного пространства, то есть 50 %.)

После редизайна и вычисления процентных значений нужно помнить, что общая ширина всех колонок в одном ряду не должна превышать 100 %.

#### СОВЕТ

Ту же формулу можно применить для преобразования значений размеров из пикселей в единицы em. Предположим, что ширина текстового абзаца составляет 18 пикселей (цель). Исходный размер обычного текста (контекст) составляет 16 пикселей. При делении 18 на 16 получаем новый размер в единицах em: 1.125em.

## Гибкие изображения

Наряду с тем, что гибкая компоновка позволит создать дизайн, который будет прекрасно отображаться на экранах с разрешением широкого спектра устройств, при добавлении на страницы изображений возникает проблема. Хотя колонки в гибком дизайне по мере уменьшения окна сжимаются, изображений обычно это не касается. Это может привести к выпадению изображений за обозначенные пределы и к тому, что они перестанут вписываться в ширину колонки (рис. 15.3).

К счастью, можно придать гибкости и изображениям. Для этого нужно выполнить два шага: создать новый стиль CSS и внести ряд изменений в HTML-код.

1. Для начала добавьте в таблицу стилей следующий код:

```
img { max-width: 100%; }
```

Этот код ограничит максимальный размер любого изображения значением 100 % от ширины контейнера этого изображения. То есть изображение не сможет стать шире колонки, div-контейнера или любого HTML-элемента, внутри которого оно находится.

Но этого еще недостаточно для обеспечения гибкости изображений. Обычно при добавлении элемента `img` указываются атрибуты его высоты и ширины. Именно эти размеры используются браузером при визуализации изображения. Если присвоить значение свойству `max-width`, изображение не станет шире колонки, но его высота по-прежнему будет точно соответствовать значению, указанному в HTML-коде. Таким образом, изображение подстроится под ширину колонки, а его высота не изменится, что приведет к искажению этого изображения. Решение вполне очевидно: нужно удалить из HTML-кода атрибуты `width` и `height`.

2. Найдите на странице все элементы `img` и удалите из них атрибуты `height` и `width`. То есть измените HTML-код:

```

```

следующим образом:

```

```

Во многих редакторах HTML-кода есть режим поиска и замены, который может ускорить поиск и удаление этих атрибутов.



**Рис. 15.3.** Если колонка становится уже, чем находящееся внутри нее изображение, это изображение выходит за пределы колонки, перекрывая другие колонки и контент страницы

Разумеется, этот подход предполагает, что все ваши изображения будут заполнять колонку, внутри которой они находятся.

Во многих случаях вам потребуется, чтобы изображения были меньше этого размера. Например, можно выровнять фотографию по левому краю основной колонки, а текст будет обтекать изображение по правому краю. Для работы с изображениями, размер которых задается по-разному, можно создать различные классы с иными значениями свойства `max-width` и применить эти классы к конкретным элементам `img` в HTML-коде. Итак, требуется, чтобы изображение было выровнено по левому краю колонки, а его размер составлял 40 % от ширины колонки. Сначала создадим стиль класса:

```
.imgSmallLeft {
  float: left;
  max-width: 40%;
}
```

Затем применим этот класс к элементу `img`:

```

```

Более гибким подходом будет разделить ограничение размера и установку выравнивания элемента по разным классам:

```
.imgSmall {
  max-width: 40%;
}
```

```
.imgLeft {  
  float: left;  
}
```

А затем применить к изображению оба класса:

```

```

Путем использования двух классов можно применить класс `imgSmall` к любым изображениям, даже к тем, которые вы решили выровнять по правому краю или вообще не выравнивать, но ограничить их ширину.

#### ПРИМЕЧАНИЕ

Изменить размер фоновых изображений можно с помощью свойства `background-size`.

## Недостатки гибких изображений


У гибких изображений есть одна проблема. При просмотре страницы на смартфоне размеры колонки и изображения могут ужаться до ширины существенно меньшей, чем в окне браузера компьютера. Это означает, что изображения на экране будут сильно уменьшены. Хотя качество изображения от этого не пострадает, пользователи смартфонов будут вынуждены загружать графические файлы размером намного больше необходимого. Тем самым будет неоправданно расходоваться трафик на большие файлы изображений, в которых нет необходимости. К сожалению, пока эта проблема не имеет полноценного решения, и фактически сторонники адаптивного веб-дизайна считают ее одной из наиболее существенных проблем.

### В КУРС ДЕЛА!

#### Тестирование адаптивных макетов

Поскольку адаптивный дизайн предназначен для ответа экранам различных устройств просмотра, нужно найти способ просматривать ваши страницы на экранах с различным разрешением. Проще всего протестировать ваши медиазапросы путем просмотра страницы на компьютере при изменении окна браузера. Перетащите край окна браузера, сделав его уже, и посмотрите, что получится в каждой точке останова или на каждом установленном вами медиазапросе. Этот прием работает неплохо, но не все браузеры позволяют уменьшать размер окна до ширины 320 пикселей, характерной для некоторых смартфонов.

Кроме того, браузер Google Chrome содержит простой инструмент, который позволяет имитировать просмотр страницы на различных устройствах с изменением высоты и ширины экрана. Чтобы открыть инструмент

разработчика, нажмите кнопку  и выберите команду меню **Дополнительные инструменты** ► **Инструменты разработчика** (**Developer** ► **Developer Tools**). Затем нажмите кнопку с изображением смартфона, расположенную в верхнем левом углу панели **Developer Tools** (**Инструменты разработчика**), чтобы открыть новое представление текущей страницы. Используйте раскрывающийся список **Device** (**Устройство**) в верхней части страницы, чтобы выбрать новое устройство, например **Apple iPhone 6** или **Nokia Lumia 520**. По окончании закройте панель **Developer Tools** (**Инструменты разработчика**), чтобы вернуться к обычному режиму просмотра веб-страницы.

Существует также несколько веб-инструментов, позволяющих просматривать страницы в окнах разных размеров. Сайт [responsivepx.com](http://responsivepx.com) позволяет вве-

сти URL-адрес страницы во Всемирной паутине, затем указывать различные параметры ширины и высоты экрана. Этот сайт открывает страницу во фрейме установленной ширины. Браузер применяет любые медиа-запросы, поэтому вам предоставляется возможность просматривать результат их выполнения.

Онлайн-инструмент [responsinator.com](http://responsinator.com) предоставляет ряд предустановок ширины окна, соответствующих таким популярным устройствам, как iPhone, Samsung Galaxy, iPad, Kindle и т. д. Нужно лишь указать URL-адрес тестируемого сайта, и его страница появится внутри эмулируемых экранов. Ресурс работает только со страницами, опубликованными во Всемирной паутине.

Если у вас есть смартфон или планшет, страницу можно просмотреть непосредственно с вашего ком-

пьютера на этих устройствах, используя приложение Adobe Dreamweaver ([tinyurl.com/o92twzj](http://tinyurl.com/o92twzj)). Команда Предварительный просмотр на устройстве (Device Preview) предоставляет замечательную возможность просматривать дизайн страницы в процессе его разработки. Можно также выгрузить страницу на веб-сервер, подключить к Интернету смартфон и посетить эту страницу, чтобы просмотреть ее. Разумеется, пока вы не обзаведетесь десятком разных моделей смартфонов, вы не узнаете, какое впечатление создается от вашей веб-страницы у тех людей, которые пользуются теми или иными мобильными браузерами.

Наконец, вы можете подписаться на службу Browser-Stack ([browserstack.com](http://browserstack.com)), которая позволяет увидеть, как будет выглядеть веб-страница в различных операционных системах, браузерах и на разных устройствах, включая смартфоны и планшеты.

К числу других людей и организаций, работающих над проблемой гибких изображений, относится Консорциум W3C. Ознакомиться с наиболее популярными в настоящее время решениями можно на сайте [tinyurl.com/q43644b](http://tinyurl.com/q43644b). К сожалению, пока не существует решения, которое бы подошло для всех браузеров, поэтому можно проигнорировать все волнения насчет больших файлов изображений для мобильных систем и ждать, пока не появится решение этой проблемы. Или же можно воспользоваться весьма удачным, но непростым решением, которое называется *адаптивные изображения*. Для передачи изображений подходящего к каждому устройству размера используются команды на языке JavaScript и PHP. То есть браузеры мобильных устройств для своих небольших экранов получают изображения, меньшие по объему, а браузеры компьютеров — более крупные изображения. Широко этот вопрос рассмотрен на сайте [adaptive-images.com](http://adaptive-images.com).

## Видео и Flash-контент

При использовании HTML-элемента `video` или внедрении Flash-контента для масштабирования таких элементов вместе с их контейнерами также можно воспользоваться приемом со свойством `max-width`. Добавьте в свою таблицу стилей следующий стиль:

```
img, video, embed, object {
    max-width: 100%;
}
```

К сожалению, он не в состоянии справиться с видеоконтентом, внедренным с помощью элементов `iframe` (а это наиболее часто используемый способ добавления на страницу видеоконтента с сайтов видеохостинга типа YouTube или Vimeo).

Чтобы внедрять видеоконтент с сайта YouTube, прочтите статью [tinyurl.com/q2y7k94](http://tinyurl.com/q2y7k94) или обратитесь к онлайн-инструменту [embedresponsively.com](http://embedresponsively.com). Укажите URL-адрес видеоролика на сайте YouTube или Vimeo, и вы сможете сгенерировать HTML-код, который необходимо добавить в качестве адаптивной версии видеоконтента на своей веб-странице.

## В КУРС ДЕЛА!

### Когда пиксел уже не пиксел?

Многим пиксел представляется в виде отдельной точки на экране или на мониторе. Это так и есть. Но благодаря новым дисплеям с высокой плотностью пикселей, таким как разработанные компанией Apple Retina-дисплеи, теперь о пикселях нужно иметь двойное представление.

Устройства наподобие iPhone 6 Plus имеют разрешение  $1920 \times 1080$  пикселей и здесь довольно большое количество пикселей помещено в область, меньшую по размеру, чем у большинства других экранов с таким разрешением. Каждый дюйм экрана составляет 401 пиксел. У компьютерных мониторов обычно приходится около 100 пикселей на дюйм. То есть у экрана iPhone в три раза больше пикселей на дюйм, чем у многих компьютерных мониторов. В результате получаются потрясающе четкие и резкие изображения.

Но в результате этого проблема возникает у веб-дизайнеров. Если для шрифта текста задан размер 16 пикселей, позволяющий ему хорошо смотреться на компьютерном мониторе (высота текста составляет около 0,16 дюйма), то он будет абсолютно нечитаем на смартфоне с Retina-дисплеем, поскольку высота текста составит 0,04 дюйма.

К счастью, браузеры в устройствах с Retina-дисплеями так поступать не будут. Они сделают так, чтобы каждый пиксел занимал на экране такого дисплея *несколько* пикселей и высота текста размером 16 пикселей на самом деле отображалась с использованием *более* 16 пикселей. Смартфоны и другие устройства с Retina-дисплеями различают *пиксели экрана устройства*, то есть точки на экране, и *пиксели веб-страниц*.

Пиксел веб-страницы вычисляется на основе плотности пикселей на экране и расстояния от глаз пользователя до экрана. Поскольку смартфон держат ближе к лицу, чем монитор, элементы на его экране крупнее элементов с таким же размером на мониторе с диагональю, к примеру, 28 дюймов.

На экране смартфона iPhone 1 пиксел веб-страницы фактически представлен 2 пикселями устройства. Поэтому текст высотой 16 пикселей на устройстве фактически занимает высоту 32 пикселя. У разных устройств, например у смартфонов под управлением операционной системы Android, разная плотность пикселей, поэтому для определения количества пикселей экрана, приходящихся на 1 пиксел веб-страницы, используются разные вычисления.

## Практикум: адаптивный веб-дизайн

В этом практикуме мы возьмем дизайн, созданный в главе 13 (с добавлением нескольких изображений), и наделим его адаптивными свойствами. Будет продемонстрирован подход, при котором предпочтение отдано дизайну для компьютеров, то есть исходные стили лучше всего будут работать в браузерах компьютеров, а затем к ним будут добавлены медиазапросы, адаптирующие внешний вид под экраны среднего размера (экраны планшетов) и под небольшие экраны (экраны смартфонов).

Чтобы начать обучение, вы должны иметь в распоряжении файлы с учебным материалом. Для этого загрузите файлы для выполнения заданий практикума,



расположенные по адресу [github.com/mrightman/css\\_4e](https://github.com/mrightman/css_4e). Перейдите по ссылке и загрузите ZIP-архив с файлами (нажав кнопку **Download ZIP** в правом нижнем углу страницы). Файлы текущего практикума находятся в папке 15.

## Изменение HTML-кода

При адаптации макета под небольшой экран вы превращаете свой трехколоночный дизайн в одноколоночный, где все блоки с контентом следуют друг за другом. Проблема HTML-кода страницы, которая была создана ранее в этой книге, заключается в том, что сначала в нем следует левая боковая панель, поэтому при преобразовании дизайна в одноколоночный эта боковая панель появится ранее основного контента.

Будет лучше, если посетители сайта сначала увидят основной контент, а затем, прокручивая страницу, — дополнительную информацию, находящуюся на бывших боковых панелях. Чтобы получить такой результат, нужно добавить некоторый HTML-код и переместить часть уже существующего кода. Для этого будет использоваться технология, рассмотренная в подразделе «Предотвращение выпадений обтекаемых элементов» раздела «Решение проблем с обтекаемыми элементами» главы 13 (см. рис. 13.4): установка контейнера с основным контентом ранее первой боковой панели с последующей оберткой основного контента и боковой панели в новый div-контейнер. Этот новый div-контейнер нужно выровнять по левому краю, основной контент внутри него — по правому краю контейнера, а боковую панель — по левому. Такое выравнивание позволит поддерживать единую визуальную компоновку в стиле компьютерных мониторов (слева боковая панель, в центре основной контент, а справа — вторая боковая панель), а для дизайна под смартфоны основной контент будет следовать ранее двух бывших боковых панелей.

1. Откройте файл `index.html`, который находится в папке 15.

Его код в точности соответствует финальной странице из урока главы 13. Сначала нужно переместить основной контент вверх.

2. Найдите в HTML-коде комментарий `<!-- основной контент -->` и выделите его и весь HTML-код ниже, включая закрывающий тег `</article>`.

То есть выделите этот комментарий и весь код ниже до комментария `<!-- вторая боковая панель -->`.

3. Вырежьте текст и поместите его в буфер обмена с помощью команды меню **Редактировать ▸ Вырезать** (**Edit ▸ Cut**) (или воспользуйтесь аналогичным способом, предоставляемым вашим редактором HTML-кода).

Далее этот код будет вставлен перед кодом первой боковой панели.

4. Найдите ближе к началу файла элемент `<div class="contentWrapper">`. Добавьте под ним перед комментарием `<!-- первая боковая панель -->` пустую строку и воспользуйтесь командой меню **Редактировать ▸ Вставить** (**Edit ▸ Paste**), чтобы вставить основной контент ранее кода боковой панели.

Теперь нужно будет добавить div-контейнер, чтобы обернуть им основной контент и первую боковую панель.

5. После элемента `<div class="contentWrapper">` добавьте элемент `<div class="columnWrapper">`. HTML-код должен приобрести следующий вид:

```
<div class="contentWrapper">
<div class="columnWrapper">
<!-- основной контент -->
```

Затем нужно закрыть этот div-контейнер.

6. Найдите закрывающий тег `</aside>`, принадлежащий коду первой боковой панели. Он находится непосредственно перед комментарием `<!-- вторая боковая панель -->`. Добавьте тег `</div>` после тега `</aside>`, чтобы данный фрагмент HTML-кода приобрел следующий вид:

```
</aside>
</div>
<!-- вторая боковая панель -->
```

Пока это будут все изменения, вносимые в HTML-код. Если вы запутались или хотите проверить свою работу, файл с именем `new-source-order.html`, содержащий все эти изменения HTML-кода, можно найти в папке 15.

Если просмотреть эту страницу в браузере, можно увидеть трехколоночный дизайн, но с основным контентом слева и первой боковой панелью в центре (рис. 15.4). Это проблему можно решить с помощью дополнительного CSS-кода.

7. Откройте файл `styles.css`, который находится в папке 15.

В нем содержится код каскадной таблицы стилей, созданный во время изучения предыдущей главы. Сначала нужно добавить новый стиль для контейнера колонок и выровнять его по левому краю, чтобы он располагался рядом с правой боковой панелью.

8. Добавьте ближе к концу файла перед стилем `.sidebar1` следующий код:

```
.columnWrapper {
  float: left;
  width: 80%;
}
```

Значение свойства `width` здесь формируется из ширины левой боковой панели (20 %) и основной колонки (60 %). На самом деле здесь создается двухколоночный дизайн: первую колонку представляет ранее созданный элемент `div`, а вторую — правая боковая панель. Контейнер основного контента и левая боковая панель, по сути, являются двумя колонками внутри `div`-контейнера, в которую заключена эта колонка, то есть вложенными колонками, как уже демонстрировалось на рис. 13.4.

Далее нужно будет настроить обтекаемые элементы и изменить ширину основного контента и первой боковой панели.

9. Измените значение свойства `width` в стиле `.sidebar` на 25%. Код стиля должен приобрести такой вид:

```
.sidebar1 {
  float: left;
```

```
width: 25%;
padding: 0 20px 0 10px;
}
```

Изначально эта боковая панель занимала 20 % всей ширины страницы, но теперь, когда она находится внутри `div`-контейнера, оборачивающего колонки, нужно подогнать ширину колонок боковой панели и основного контента, чтобы они умещались в 80 % от ширины страницы, выделенных контейнеру колонок. То есть указанное процентное отношение относится не ко всей странице, а к контейнеру колонок, который занимает только 80 % ширины страницы.

Чтобы вычислить новое процентное отношение, нужно взять предыдущее значение — 20 %, разделить его на ширину контейнера — 80 %, а затем умножить результат на 100. Результат деления 20 на 80 равен 0,25. Умножение этого результата на 100 позволяет получить значение 25 %. Та же технология должна быть применена и к изменению ширины основного контента.



**Рис. 15.4.** Страница по-прежнему имеет трехколоночный дизайн, но теперь основной контент находится в левой части окна, а левая боковая панель находится в центре.

Вернуть эти колонки на их исходные места можно с помощью пары простых стилей

- Найдите стиль `.main` и присвойте свойству `float` значение `right`, а свойству `width` — значение 75%, придав стилю следующий вид:

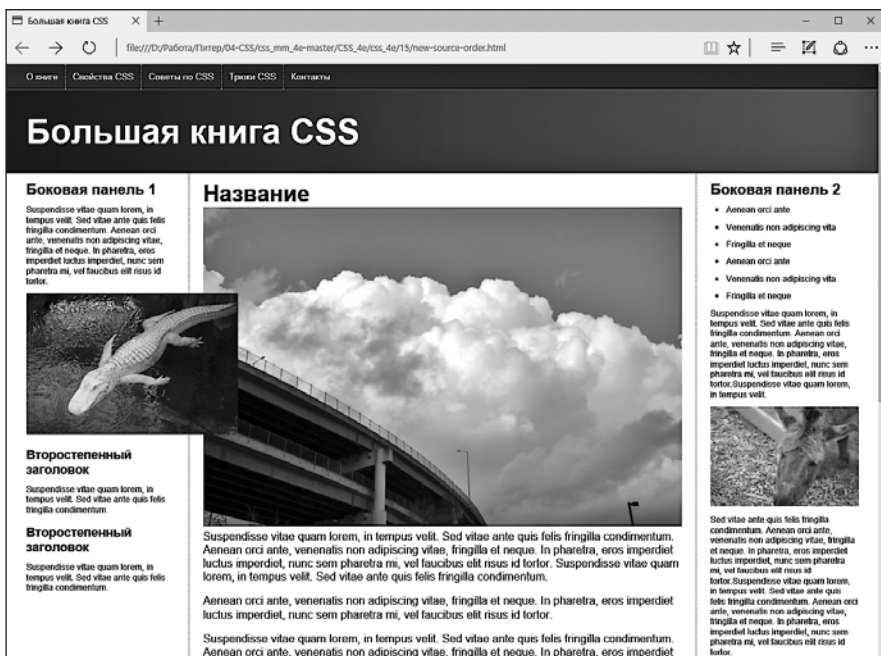
```
.main {
  float: right;
  width: 75%;
  padding: 0 20px;
```

```
border-left: dashed 1px rgb(153,153,153);
border-right: dashed 1px rgb(153,153,153);
}
```

Теперь элемент выровнен по правому краю, поэтому он появляется справа от первой боковой панели, в результате чего основной контент снова помещается в центр. Ширина вычисляется путем деления старого значения ширины элемента на значение ширины его контейнера:  $60 / 80 = 0,75$ , или 75 %. Если сохранить файлы и просмотреть файл `index.html` в браузере, он должен выглядеть так, как показано на рис. 15.5.

#### ПРИМЕЧАНИЕ

Дизайн страницы в этом примере не станет шире 1200 пикселей благодаря свойству `max-width`, которому присвоено значение 1200px в групповом селекторе `nav ul, header h1, footer p, .contentWrapper`. Если нужно, чтобы страница заполняла окно браузера независимо от его ширины, удалите объявление `max-width: 1200px`; из этого группового селектора.



**Рис. 15.5.** Возвращаясь к исходному состоянию: к трехколоночному дизайну.

Но теперь порядок следования исходного HTML-кода страницы больше подходит для создания одноколоночного дизайна для смартфонов. Некоторые изображения не помещаются в отведенное для них пространство и выпадают за пределы колонок.

Далее вы займетесь устранением этого недостатка

## Адаптивные изображения

Вы успешно справились с созданием резинового макета. Если теперь просмотреть страницу и изменить размеры окна, можно увидеть, что по мере уменьшения раз-

мера окна браузера колонки становятся меньше. К сожалению, некоторые изображения шире колонок и выпадают за их пределы. Как уже ранее упоминалось, составной частью адаптивного веб-дизайна является еще и придание изображениям возможности адаптации к изменениям ширины окна браузера. Для реализации такой возможности нужно немного дополнить CSS-код и убрать часть HTML-кода.

1. Перейдите к файлу `styles.css` в редакторе HTML-кода. В нижней части таблицы стилей добавьте стиль для элемента `img`:

```
img {  
    max-width: 100%;  
}
```

Этот стиль ограничивает максимальную ширину изображения 100 % от ширины контейнера. Следовательно, для колонки, имеющей на экране ширину 200 пикселей, изображение будет шириной 200 пикселей. Если посетитель изменит размеры окна браузера и колонки станут меньше, то изображение также уменьшится, чтобы поместиться в колонке.

Но все это будет функционировать при условии удаления свойств изображения `width` и `height` из HTML-кода.

2. Удалите для каждого из четырех элементов `img`, имеющихся в файле `index.html`, атрибуты `height` и `width`.

У нас имеются четыре изображения — `clouds.jpg`, `jellyfishy.jpg`, `gator.jpg` и `mule.jpg`. Удалите из соответствующих элементов `img` атрибуты `width` и `height`. Например, код:

```

```

должен принять вид:

```

```

Если сохранить CSS- и HTML-файлы и просмотреть страницу в браузере, можно будет увидеть, что по мере того, как окно браузера уменьшается, изображения меняют свой размер. Но два изображения в основной колонке все же слишком большие. Использование объявления `max-width: 100%` для элемента `img` приводит к созданию гибких изображений, но во многих случаях вам не нужно, чтобы изображения становились такими же по ширине, как и сама колонка. В случае с двумя изображениями в основной колонке они будут в целом лучше смотреться, если займут только половину размера основной колонки. Этого можно добиться путем применения к этим элементам определенных классов и добавления стилей.

3. Добавьте в конце файла `styles.css` следующие стили:

```
img.half {  
    max-width: 50%;  
}  
img.left {  
    float: left;  
    margin: 0 10px 10px 0;
```

```
}  
img.right {  
  float: right;  
  margin: 0 0 10px 10px;  
}
```

Первый стиль присваивает свойству `max-width` значение 50%, что соответствует половине ширины колонки. Другие два стиля выравнивают элемент изображения по левому или правому краю колонки, позволяя тексту обтекать изображения, и создают небольшие поля. Чтобы воспользоваться этими стилями, к элементам изображений нужно добавить имена классов.

4. Найдите в файле `index.html` элемент `img` изображения `clouds.jpg` — `` — и добавьте код `class="half right"`, чтобы получился следующий код:

```

```

Добавление к элементу более одного класса — довольно устоявшаяся и полезная практика. Таким образом можно объединять простые модульные стили для создания более сложных конструкций дизайна. В данном случае класс `half` изменяет размер изображения, а класс `right` выравнивает его по правому краю. Простым изменением значения `right` на `left` можно выровнять изображение по левому краю колонки, сохраняя ограничение размера, заданное в стиле `half`.

5. Добавьте в элемент изображения `jellyfish.jpg` атрибут `class="half left"`, придав HTML-коду следующий вид:

```

```

Теперь изображения получили нужные размеры, и страница должна приобрести вид, показанный на рис. 15.6.

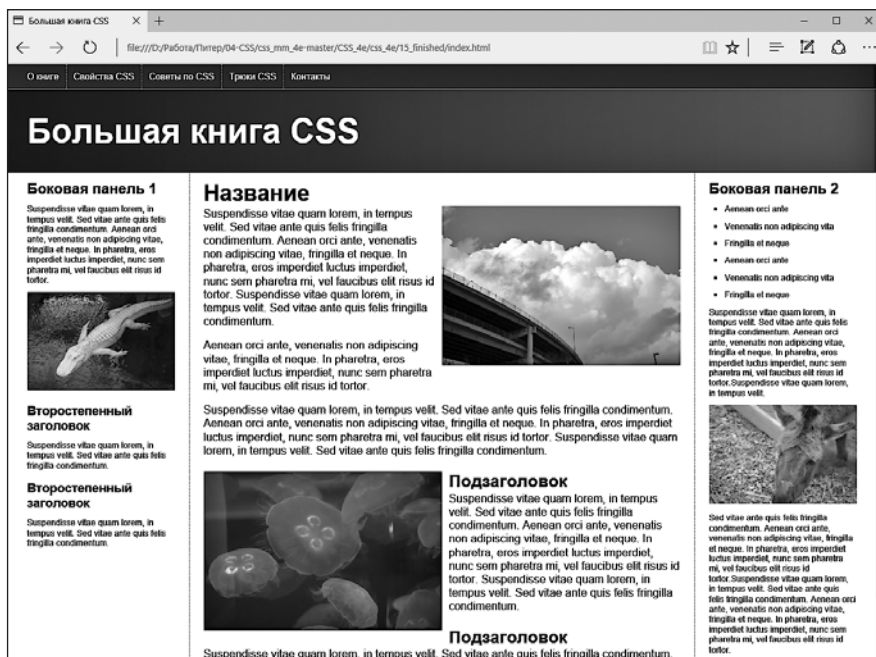
По желанию можно создать дополнительные стили для других размеров и даже указать размеры в пикселах, позволяя нужным изображениям изменять размеры до их полной ширины, но не более.

## Добавление стилей для планшетов

На данный момент дизайн страницы не меняется на всех устройствах просмотра: компьютерных мониторах, планшетах и смартфонах. Это резиновый дизайн, который меняет размер контента, чтобы уместить его в окне браузера. Но в определенный момент боковые панели становятся настолько узкими, что текст на них уже не прочитать. Первый добавляемый вами медиазапрос будет нацелен на устройства с экранами, разрешение по горизонтали которых находится в диапазоне 480–768 пикселей с перемещением правой боковой панели в нижнюю часть страницы, превращая трехколоночный дизайн в более удобочитаемый двухколоночный.

1. Откройте в редакторе HTML-кода файл `styles.css`. Перейдите в конец файла и добавьте следующий код:

```
@media (min-width: 481px) and (max-width:768px) {  
}
```



**Рис. 15.6.** Создание адаптивных изображений не составляет особого труда.

Нужно лишь убрать из HTML-кода атрибуты высоты и ширины изображений и присвоить значение свойству `max-width`. Если к изображениям нужно применить другие размеры, создайте классы с требуемыми значениями свойства `max-width` и примените их к элементам `img`

Это первый медиазапрос. Он предназначен для устройств с экранами, имеющими разрешение по горизонтали не менее 481 и не более 768 пикселей. В следующем упражнении этого практикума будет создан медиазапрос для устройств, экранное разрешение по горизонтали которых не превышает 480 пикселей. Поэтому данный медиазапрос исключает такие устройства, как и устройства с экранным разрешением по горизонтали, превышающим 768 пикселей. Иными словами, любые стили, помещаемые в этот раздел таблицы, не будут применяться к браузерам компьютеров (пока вы не уменьшите их окно до слишком узкого) и большинства смартфонов.

Первое, что нужно сделать, — это удалить свойство `float` из стиля правой боковой панели.

2. Внутри медиазапроса, добавленного при выполнении предыдущего шага, создайте следующий стиль:

```
@media (min-width: 481px) and (max-width:768px) {
  .sidebar2 {
    float: none;
    width: auto;
  }
}
```



Этот код отменяет выравнивание боковой панели и присваивает свойству `width` значение `auto` (замещая значение `20%` из стиля `.sidebar2`, ранее определенного в данной таблице). Но этот стиль не заставит боковую панель опуститься ниже двух других колонок. Поскольку `div`-контейнер колонок выровнен по левому краю, вторая боковая панель будет обтекать его. Вам нужно с помощью свойства `clear` освободить боковую панель от обтекания.

3. Отредактируйте стиль, созданный на предыдущем шаге, добавив в него код, выделенный полужирным шрифтом:

```
.sidebar2 {  
  float: none;  
  width: auto;  
  clear: both;  
  border-top: 2px solid black;  
  padding-top: 10px;  
}
```

Свойство `clear` (рассмотренное в главе 7) опускает эту боковую панель ниже двух других колонок. Кроме этого, здесь добавляется характерная верхняя граница для еще большего визуального отделения этого контейнера от верхних колонок.

Теперь, если просмотреть страницу и уменьшить ширину окна браузера до 768 пикселей, вы увидите, что две колонки не распространяются на всю ширину страницы (рис. 15.7). Причина в том, что ранее на шаге 8 предыдущей последовательности действий для контейнера этих колонок было установлено значение ширины, равное 80 % ширины страницы. Вам нужно сбросить этот стиль в медиазапросе для планшетных устройств.

4. Добавьте после стиля `.sidebar2` следующий код:

```
.columnWrapper {  
  width: auto;  
}
```

Убедитесь в том, что этот стиль находится внутри медиазапроса. Он сбрасывает ширину контейнера колонок, присваивая свойству `width` значение `auto` (то же самое, что и `100 %`), поэтому распространяется на всю ширину страницы. Поскольку теперь у нас только две колонки, правую границу, применявшуюся к колонке основного контента, можно удалить.

5. Добавьте еще один стиль после класса `.columnWrapper`. Весь медиазапрос должен приобрести следующий вид (изменения выделены полужирным шрифтом):

```
@media (min-width: 481px) and (max-width:768px) {  
  .sidebar2 {  
    float: none;  
    width: auto;  
    clear: both;  
    border-top: 2px solid black;  
    padding-top: 10px;  
  }  
  .columnWrapper {
```

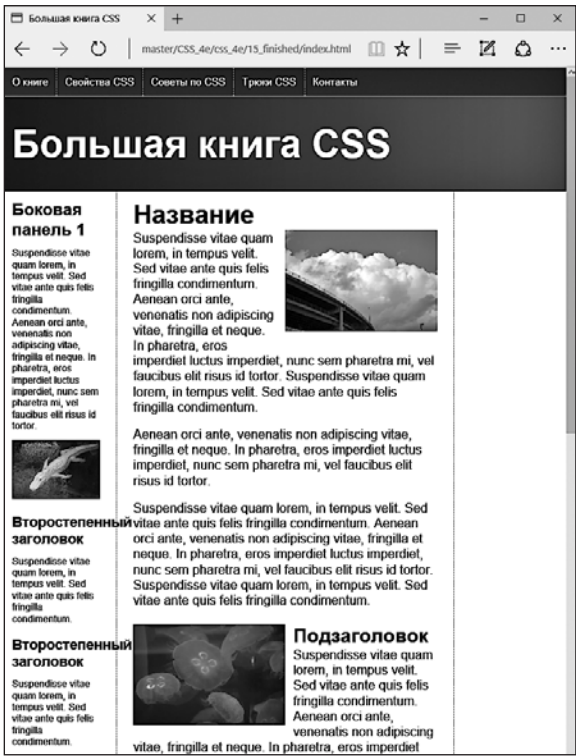


Рис. 15.7. От трехколоночного дизайна к двухколоночному с помощью одного медиазапроса и трех простых стилей

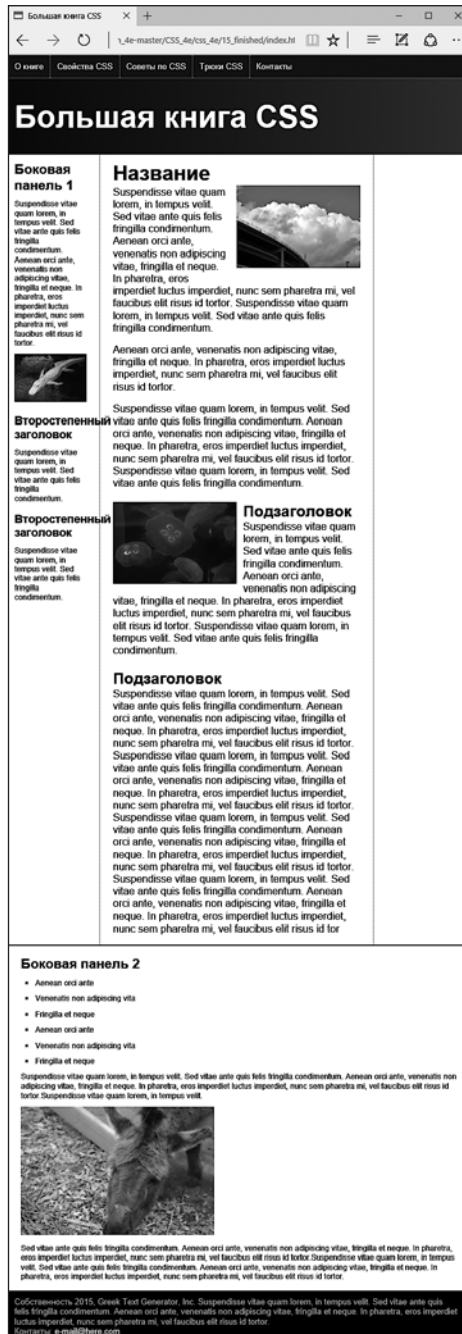
- ```
width: auto;
}
.main {
  border-right: none;
}
}
```
6. Сохраните файл `styles.css` и просмотрите файл `index.html` в браузере. Уменьшайте ширину окна браузера до появления только двух колонок. Страница должна иметь вид, показанный на рис. 15.8.

## Добавление стилей для смартфонов

И наконец, настал черед добавить стили для устройств, разрешение экрана которых по горизонтали составляет менее 480 пикселей.

1. Откройте в редакторе HTML-кода файл `styles.css`. Перейдите в конец файла и добавьте еще один медиазапрос:

```
@media (max-width:480px) {
}
```



**Рис. 15.8.** Теперь дизайн зависит от ширины окна браузера — три колонки превращаются в две, а вторая боковая панель смещается вниз. Ширина контента страницы соответствует ширине окна браузера. Это гарантирует, что колонки будут обладать достаточной шириной для удобства чтения даже на устройствах с небольшим экраном

Укажите этот медиазапрос после предыдущего. Теперь мы нацелимся на устройства, разрешение экрана которых по горизонтали составляет менее 480 пикселей. Сначала нужно будет *линеаризовать* дизайн, то есть удалить все обтекаемые элементы, чтобы контент выстроился друг за другом в одну колонку для повышения его читабельности на небольшом экране.

2. Внутри этого медиазапроса добавьте следующий стиль:

```
.columnWrapper, .main, .sidebar1, .sidebar2 {  
  float: none;  
  width: auto;  
}
```

Он удалит обтекаемые (то есть выровненные) элементы из контейнера, обертывающего колонки, и из всех колонок. Присвоение свойству `width` значения `auto` позволяет элементам заполнять свои контейнеры полностью. Таким образом, теперь три колонки становятся простыми блочными элементами, располагаемыми друг за другом, с основным контентом в начале и двумя боковыми панелями следом.

Область основного контента все еще имеет границы, создаваемые стилями, ранее добавленными в CSS-файл. В трехколоночном дизайне эти левые и правые границы служили визуальными разделителями колонки основного контента и левой/правой боковых панелей. В этом последовательном дизайне они не нужны, поэтому их можно удалить. С другой стороны, желательно добавить визуальное разделение между блоками контента, которые теперь следуют друг за другом.

3. После только что добавленного стиля (внутри медиазапроса `@media (max-width:480px) { }`) введите следующий код:

```
.main {  
  border: none;  
}  
.sidebar1, .sidebar2 {  
  border-top: 2px solid black;  
  margin-top: 25px;  
  padding-top: 10px;  
}
```

Представленный здесь групповой селектор добавляет границу над каждой боковой панелью, а также верхние поле и отступ для четкого разделения различных областей контента страницы.

Сохраните CSS-файл и просмотрите в браузере файл `index.html`. Нажав и удерживая кнопку мыши на краю окна браузера, перетащите мышью так, чтобы уменьшить ширину окна до 480 пикселей. Теперь вы увидите дизайн, состоящий из одной колонки. Если просмотреть страницу на смартфоне, можно заметить, что заголовок выглядит слишком крупным (рис. 15.9). Кроме того, элементы навигации смотрятся не очень хорошо. Далее мы займемся устранением этих проблем.



**Рис. 15.9.** Даже если получена подходящая для экрана смартфона ширина страницы, нам еще есть чем заняться. Например, заголовком, который больше подходит по размеру для браузера компьютера, а на экране смартфона выглядит огромным, занимая слишком много полезного пространства

4. Добавьте внутри медиазапроса `@media (max-width:480px)` { стиль для элемента `h1`:
 

```
header h1 {
  font-size: 1.5em;
}
```

Этот стиль уменьшает размер шрифта текста в элементах `h1`, чтобы он поместился на одной строке. Уменьшение размера заголовков при преобразовании дизайна для просмотра на небольших экранах смартфонов зачастую отнюдь не лишено смысла. Можно также прийти к выводу, что для повышения читабельности следует увеличить размер шрифта основного текста. Для настройки дизайна страницы для смартфонов существует множество способов, а каким должен быть наилучший внешний вид — решать вам.

Теперь займитесь панелью навигации. Пунктирные границы, разделяющие кнопки, смотрятся плохо и сбивают с толку. Кроме того, кнопки выровнены по левому краю, создавая при этом несбалансированную асимметрию. Они будут лучше смотреться по центру и без границ.

5. После стиля `header h1`, созданного на предыдущем шаге, добавьте следующий код:

```
nav {  
    text-align: center;  
}
```

Элементы навигации содержатся внутри HTML5-элемента `nav`. Присвоение свойству `text-align` значения `center` позволяет центрировать весь текст, находящийся внутри элемента `nav`. Но элементы навигации в данный момент выровнены по левому краю, поэтому это только первый шаг. (Использование обтекаемых элементов для создания горизонтальной панели навигации подробно рассмотрено в разделе «Создание панелей навигации» главы 9.)

6. Добавьте после стиля `nav` еще два стиля:

```
nav li {  
    float: none;  
    display: inline-block;  
}  
  
nav a {  
    float: none;  
    display: inline-block;  
    border: none;  
}
```

Вы отменяете выравниваете как элементов списка, так и самих ссылок. Кроме того, вы превращаете эти элементы в строчные блоки. Как уже упоминалось, использование строчных блоков в элементах навигации позволяет размещать ссылки рядом друг с другом, сохраняя при этом отступы и поля. Кроме того, применение строчных блоков — единственный способ центрирования этих кнопок.

7. В окончательном варианте код медиазапроса `@media (max-width:480px)` { должен иметь следующий вид:

```
@media (max-width:480px) {  
    .columnWrapper, .main, .sidebar1, .sidebar2 {  
        float: none;  
        width: auto;  
    }  
    .main {  
        border: none;  
    }  
    .sidebar1, .sidebar2 {  
        border-top: 2px solid black;  
    }  
}
```

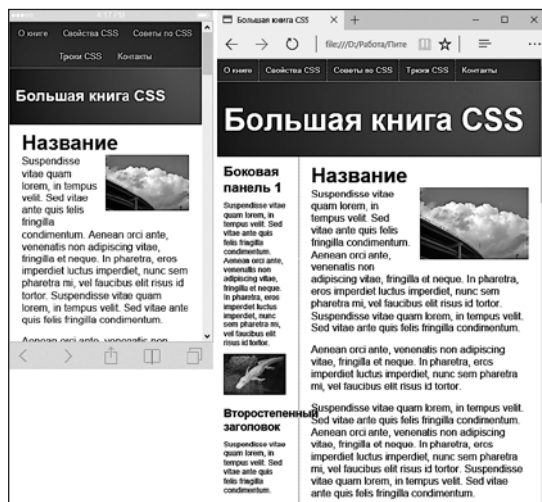
```

margin-top: 25px;
padding-top: 10px;
}
header h1 {
    font-size: 1.5em;
}
nav {
    text-align: center;
}
nav li {
    float: none;
    display: inline-block;
}

nav a {
    float: none;
    display: inline-block;
    border: none;
}
}

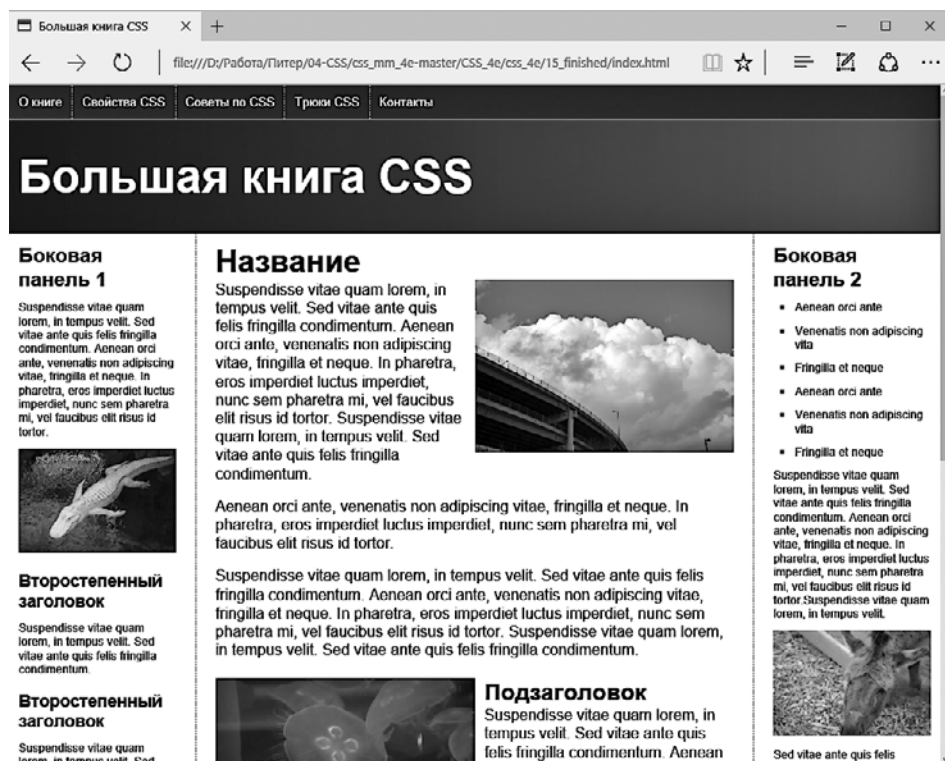
```

Сохраните все ваши файлы и просмотрите файл `index.html` в браузере. Измените размер окна браузера, чтобы его ширина стала меньше 480 пикселей, и изучите дизайн, предназначенный для смартфонов. Увеличивайте размер окна, пока в поле зрения не появится дизайн, состоящий из двух колонок, а затем продолжайте изменение, пока не увидите версию для компьютеров, имеющую три колонки. Вы должны видеть дизайны, показанные на рис. 15.10 и 15.11. Полную версию файлов этого практикума вы найдете в папке `15_finished`.



**Рис. 15.10.** Одна и две колонки, и все на одной веб-странице. Благодаря адаптивному дизайну отдельную веб-страницу можно преобразовать так, чтобы она выглядела наилучшим образом на экране устройства с любым разрешением





**Рис. 15.11.** Три колонки на одной веб-странице. Благодаря адаптивному дизайну отдельную веб-страницу можно преобразовать так, чтобы она отлично выглядела на экране устройства с любым разрешением