7 Поля, отступы, границы

На любой HTML-элемент воздействует множество свойств каскадных таблиц стилей, определяющих, каким образом он будет отображен браузером. Некоторые из них, например границы и фоновый цвет, наглядны. Другие нельзя определить явно (например, отступы и поля), но они также обеспечивают форматирование. Понимая, как эти атрибуты работают, вы можете создать привлекательные колонки, меню, элементы навигации, а также управлять пространством вокруг них (веб-дизайнеры называют его *воздухом*). Это делается для того, чтобы ваши веб-страницы не казались беспорядочными и нечитаемыми и вообще выглядели профессионально.

Все свойства, описываемые в текущей главе, — основа *блочной модели* CSS, которая представляет одну из важнейших составляющих этого языка.

Понятие блочной модели

Когда вам говорят об абзаце текста или заголовке, вы представляете буквы, слова, предложения. Фотографии, логотипы и другие изображения должны ассоциироваться с элементом `img`. Браузер обрабатывает все элементы как небольшие *блоки*. Для него любой элемент — контейнер с содержимым: текстом, изображением или другими элементами (рис. 7.1). Область внутри границ, которая включает контент и отступы, может также иметь цвет фона. Он находится под границей, поэтому, когда вы используете пунктирную или точечную границу, цвет отображается в промежутках между отрезками или точками.

Блок элемента окружают следующие свойства.

- `padding` — *отступ*, пространство между контентом и границей.
- `border` — *граница*, линия вдоль каждого края блока. Граница может отображаться как для всех сторон сразу, так и для любой из них или комбинации сторон.
- `background-color` — *цвет фона*, заполняет пространство внутри границы, включая область отступа.
- `margin` — *поле*, отделяет один элемент от другого. К примеру, пространство, обычно отображаемое сверху и снизу абзацев текста, — это поля.

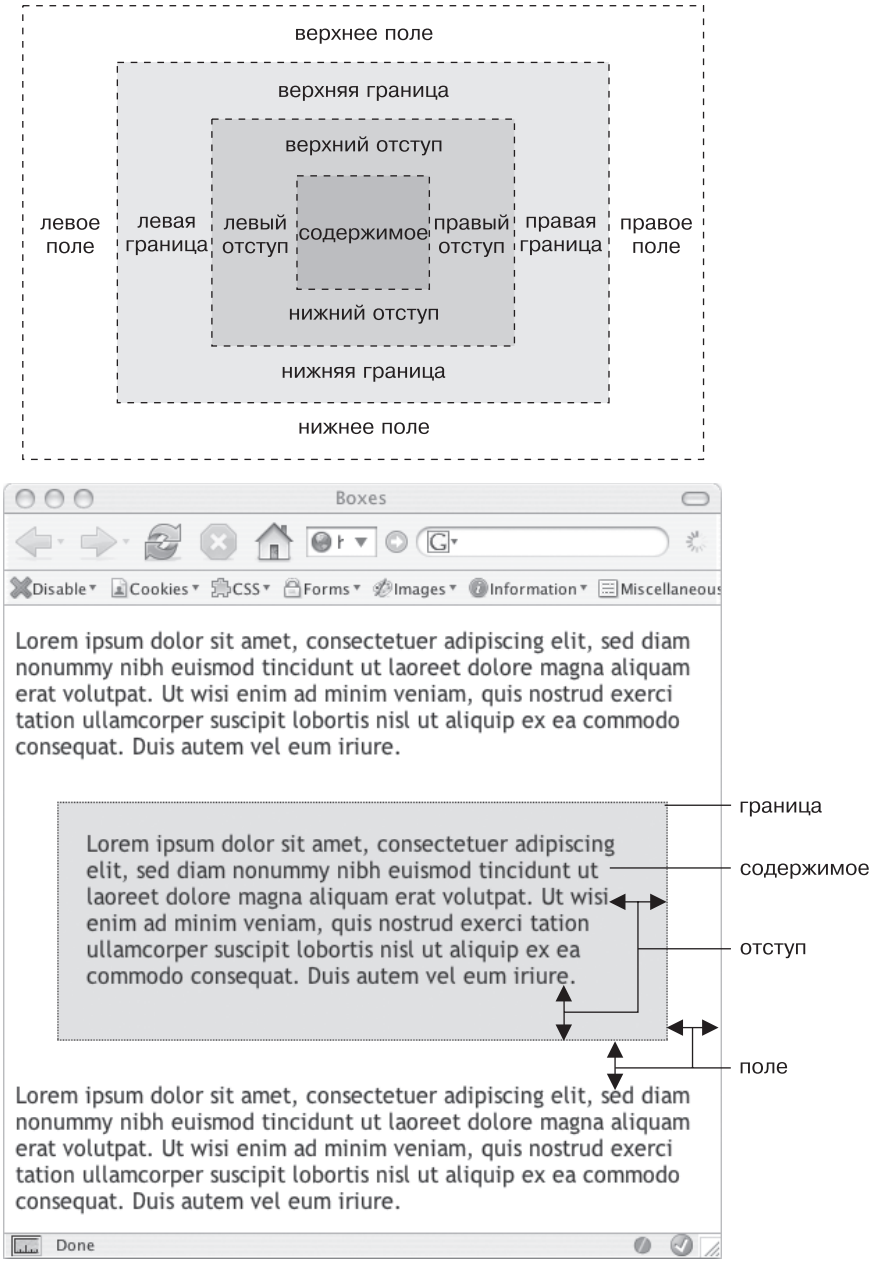

Рис. 7.1. Блочную структуру элемента образует его содержимое (например, несколько предложений текста), а также отступы, границы и поля

Для форматирования элемента можно использовать любые из этих свойств в любом сочетании или все сразу. Вы можете снабдить элемент исключительно полями или же добавить границы, поля и отступы. Или элемент может иметь границы и отступ, но

без полей и т. д. Если вы не настроите какое-либо из этих свойств, то браузер применит собственные настройки, которые вам могут как понравиться, так и нет. К примеру, хотя браузеры обычно не применяют отступы или границы к элементам на странице, некоторые элементы (например, заголовки и абзацы) имеют верхнее и нижнее поле.

ПРИМЕЧАНИЕ

Поскольку разные браузеры применяют отступы и поля разного размера, лучше всего сбрасывать значения этих свойств для всех элементов. Другими словами, используйте набор простых стилей, отвечающих за сброс CSS, для удаления отступов и полей из HTML-элементов. Потом, когда вы будете создавать дополнительные стили, добавляющие поля и отступы, вы сможете быть уверены в том, что страницы будут выглядеть одинаково в разных браузерах.

Управление размерами полей и отступов

Как поля, так и отступы добавляют промежутки вокруг содержимого элементов. Свойства `margin` и `padding` используются для отделения одного элемента веб-страницы от другого. Можно применять их, например, чтобы добавить пустое пространство между панелью навигации слева и основным контентом веб-страницы справа. Возможно, вы захотите отодвинуть границу от края фотографии (рис. 7.2).

На рис. 7.2 отступ разделяет два изображения друг от друга с помощью серого фона. Вы можете устанавливать границы, поля для каждой стороны изображения независимо друг от друга. Обратите внимание, что для нижних краев (оснований) фотографий установлены бóльшие по размеру отступы, чем для остальных.

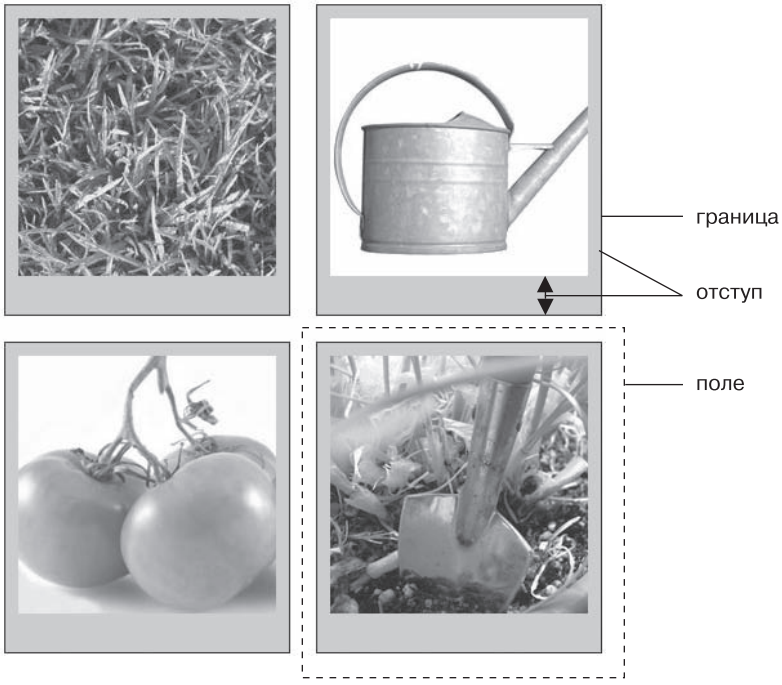

Рис. 7.2. Каждая фотография на этой веб-странице имеет поле размером 10 пикселов, то есть промежуток, отделяющий две соседние фотографии, составляет 20 пикселов

Свойства `padding` и `margin` производят одинаковый визуальный эффект, и, пока вы не добавите границу или цвет фона, вы не сможете сказать наверняка, каким свойством определен этот промежуток. Но если элемент имеет границу по периметру или цветной фон (подложку), вы заметите существенное различие этих свойств. Отступ добавляет промежуток между контентом и границей элемента и предотвращает появление эффекта заключения содержимого элемента в рамку. Он также включает область фона, поэтому пространство, занимаемое отступом, может быть свободно от содержимого (текста или фотографии), но заполнено фоновым цветом или изображением. А поля добавляют так называемые *средники* — промежутки между колонками, которые придают веб-странице более «воздушный» внешний вид.

Вы можете управлять полями или отступами каждого отдельного элемента независимо. Четыре свойства управляют соответствующими полями с каждой стороны элемента: `margin-top`, `margin-right`, `margin-bottom` и `margin-left`. Аналогично с отступами: `padding-top`, `padding-right`, `padding-bottom` и `padding-left`. Вы можете использовать любые единицы измерения, принятые в языке CSS, для определения размеров полей и отступов, например:

```
margin-right: 20px;
padding-top: 3em;
margin-left: 10%;
```

Пикселы и единицы em применяются и работают точно так же, как при форматировании текста (см. раздел «Изменение размера шрифта» главы 6). Поле размером 20 пикселов добавляет соответствующий пустой промежуток, отступ 3 em — промежуток, в три раза больший, чем размер шрифта форматируемого элемента.

Обычно используются значения в процентах. С их помощью можно гибко задавать значения границ и отступов, которые будут зависеть от ширины окна браузера, что идеально подходит для адаптивного дизайна (см. главу 15).

СОВЕТ

Чтобы удалить все пространство полей и отступов, используйте свойства со значением 0 (например, margin-top: 0 или padding-bottom: 0). Чтобы убрать все дополнительное пустое пространство с четырех сторон окна браузера, нужно присвоить свойствам margin и padding нулевые значения: margin: 0; padding 0;. Это позволит поместить баннер, заголовок, логотип или какой-то другой элемент веб-страницы вплотную у самого края окна браузера, без промежутков.

ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

Значения полей и отступов в процентах

При использовании процентов браузеры вычисляют размер полей и отступов на основе *ширины самого элемента-контейнера*, в который заключены форматируемые элементы. Рассмотрим самый простой случай, когда таким элементом-контейнером является `body`, который имеет ширину всего окна браузера. В данном случае значение в процентах в каждый конкретный момент времени вычисляется на основании текущей ширины окна. Допустим, оно составляет 760 пикселов. Тогда левое поле, равное 10 %, добавит промежуток 76 пикселов с левого края форматируемого элемента. Но если вы измените размеры окна браузера, то размер промежутка левого поля тоже изменится. Уменьшение до 600 пикселов изменит размер на 60 пикселов (10 % от 600 пикселов).

Однако элемент-контейнер не всегда равен ширине окна браузера. В последующих главах книги, когда мы будем создавать более сложный дизайн веб-страниц, вы увидите, что для разработки комплексного дизайна придется добавлять дополнительные элементы.

Возможно, вы захотите добавить в веб-страницу элемент `div` для группировки содержимого навигационной панели. Допустим, она имеет ширину 300 пикселов. `div` будет контейнером для всех остальных вложенных в него элементов. Таким образом, размер правого поля любого элемента, вложенного в `div` навигационной панели и установленного в размере 10 %, будет равен 30 пикселам.

При установке процентных значений верхнего и нижнего полей элементов ситуация еще более запутанная: эти значения вычисляются на основании ширины элемента-контейнера, а не его высоты. Таким образом, 20%-ное верхнее поле составит 20 % от ширины форматируемого элемента-контейнера.

Сокращенная запись свойств margin и padding

Нередко требуется одновременно установить одинаковые размеры полей или отступов для всех четырех сторон форматируемого элемента. Но последовательно набирать четыре различных свойства стиля (`margin-right`, `margin-left` и т. д.) утомительно и отнимает лишнее время. Здесь вы также можете использовать сокращенные варианты свойств `margin` и `padding` для быстрой установки всех четырех параметров одновременно:

```
margin: 0 10px 10px 20px;
padding: 10px 5px 5px 10px;
```

ПРИМЕЧАНИЕ

Если свойству присваивается значение 0, то совсем не нужно указывать единицу измерения. Например, наберите всего лишь margin: 0; вместо margin: 0px;.

Порядок определения четырех значений свойств `margin` и `padding` важен. Они должны указываться в следующей последовательности: сверху, справа, снизу и слева. Без учета этого у вас могут возникнуть проблемы с форматированием. Самый легкий способ запомнить очередность — сверху вниз по часовой стрелке.

Если вы хотите применить одинаковое значение свойства для всех четырех сторон, нет ничего проще — используйте единственное значение. Чтобы удалить все поля из заголовка `h1`, добавьте такой стиль:

```
h1 {
  margin: 0;
}
```

Кроме того, пользуйтесь сокращенной записью для добавления промежутков между содержимым и границами элемента:

```
padding: 10px;
```

ПРИМЕЧАНИЕ

Если нужно применить одинаковое значение свойства поля или отступа сверху и снизу элемента и одно и то же значение для левого и правого края, можно указать два значения. Так, объявление

margin: 0 2em; удаляет верхнее и нижнее поля, а левое и правое поля устанавливает равными 2 em. Точно так же, если верхние и нижние поля (или отступы) изменяются, а правые и левые остаются прежними, можно воспользоваться тремя значениями. Например, объявление margin: 0 2em 1em; установит верхнее поле равным 0, левое и правое — равными 2 em, а нижнее поле — 1 em.

Конфликты полей

В каскадных таблицах стилей не всегда справедливы математические расчеты. Вы сами можете в этом убедиться, когда нижнее поле одного элемента веб-страницы касается верхнего поля другого элемента. Вместо того чтобы объединить эти поля вместе, браузер использует большее из них (рис. 7.3, *вверху*). Предположим, значение нижнего поля маркированного списка установлено равным 30 пикселам, а значение верхнего поля следующего за ним абзаца составляет 20 пикселов. Вместо того чтобы сложить два значения, получив общий промежуток в размере 50 пикселов между списком и абзацем, браузер применяет *наибольшее* из двух значений — в данном случае 30 пикселов. Если вас такое поведение не устраивает, используйте вместо полей верхний или нижний отступ (см. рис. 7.3, *внизу*).

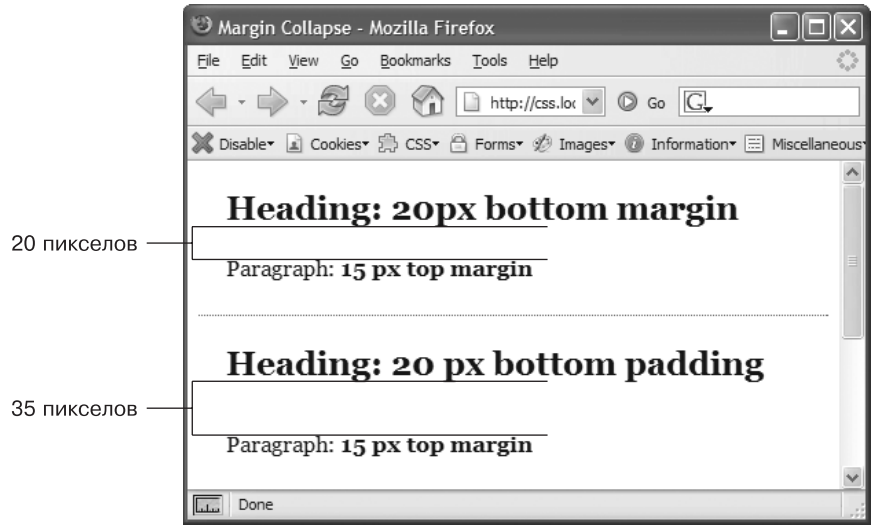

Рис. 7.3. При соприкосновении двух вертикальных полей меньшее из них игнорируется

На рис. 7.3, несмотря на то что и верхний заголовок имеет нижнее поле в размере 20 пикселов, а у расположенного ниже абзаца текста верхнее поле составляет 15 пикселов, браузер добавляет промежуток между ними, равный всего 20 пикселам. Чтобы получить тот промежуток, который вы хотели (35 пикселов), используйте вместо полей отступы, как показано в нижней части рисунка. Здесь для заголовка установлен нижний отступ 20 пикселов. Он складывается с верхним полем абзаца, равным 15 пикселам, и получается общий промежуток в размере 35 пикселов.

Ситуация еще более усугубляется, когда один элемент веб-страницы *вложен* в другой. Это может привести к затиранию отдельных частей. Допустим, вы добав-

ляете на веб-страницу «предупреждение» (заключенное в элемент `div`). Верхнее и нижнее поля устанавливаются равными 20 пикселам, чтобы отделить сообщение от заголовка сверху и от абзаца текста снизу. Пока все выглядит неплохо.

Но, предположим, вы вставляете заголовок в сообщение с предупреждением и, чтобы добавить небольшой промежуток между сообщением и верхним и нижним краем блока `div`, определяете для заголовка поле размером 10 пикселов. Вы, наверное, думаете, что добавили 10-пиксельный промежуток, но вы не правы (рис. 7.4, *вверху*). Вместо этого поле появляется *над* блоком `div`. В данном случае не имеет значения, какого размера поле применяется к заголовку, — поле все равно окажется *над* `div`.

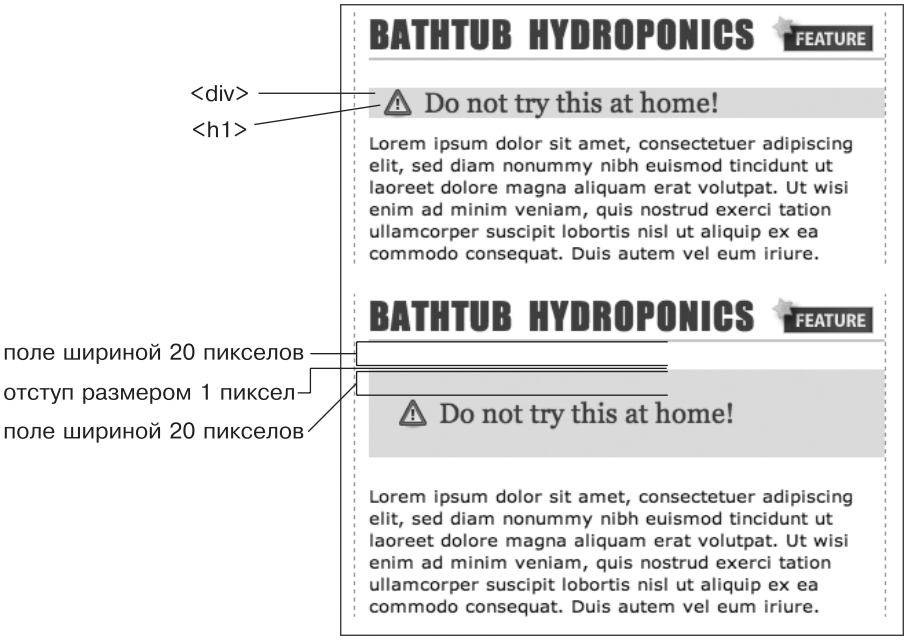

Рис. 7.4. Как бы ни соприкасались вертикальные поля, в любом случае произойдет конфликт

ПРИМЕЧАНИЕ

На профессиональном жаргоне CSS это явление называется схлопыванием полей. Оно означает, что два поля фактически превращаются в одно.

Есть два пути решения этой проблемы: добавить либо небольшой отступ, либо границу вокруг элемента `div`. Поскольку *между* этими двумя полями располагаются граница и отступ, поля больше не схлопываются и заголовок имеет небольшой отделяющий промежуток (см. рис. 7.4, *внизу*).

ПРИМЕЧАНИЕ

Горизонтальные (слева и справа) поля и поля между плавающими элементами не схлопываются. Между абсолютно и относительно позиционируемыми элементами, о которых вы узнаете в главе 13, также нет конфликта.

Удаление воздуха с помощью отрицательных значений

Большинство значений свойств в CSS должны быть положительными. Что же произойдет, если указать для размера шрифта текста отрицательное значение, например *минус 20 пикселов*? Вообще, отступы должны иметь положительные значения. Однако каскадные таблицы стилей допускают использование отрицательных значений для создания определенных визуальных эффектов.

Отрицательные значения полей вместо добавления пустого пространства между тем или иным элементом и соседними, наоборот, вызывают *удаление* этих промежутков. В результате может получиться абзац, накладывающийся на заголовок, выступающий из своего элемента-контейнера (боковой панели или другого элемента `div`) или даже совсем исчезающий за пределами окна браузера. Таким образом, можно с уверенностью сказать, что применение отрицательных значений полей дает немалую пользу.

Даже когда вы устанавливаете размеры полей, равные 0, между двумя заголовками, все равно остается небольшой промежуток (благодаря межстрочному интервалу, как описано в подразделе «Установка межстрочного интервала» раздела «Форматирование абзацев» главы 6). На самом деле это не так уж плохо, поскольку трудно читать предложения без дополнительных интервалов, сливающиеся друг с другом. Тем не менее разумное, умеренное использование небольших промежутков между заголовками поможет создать интересные визуальные эффекты. Второй заголовок на рис. 7.5 (который начинается со слов Raise Tuna) имеет верхнее поле размером 10 пикселов. Оно поднимает заголовок вверх, что обеспечивает небольшое наложение текста на пространство вышестоящего заголовка. Кроме того, левые и правые границы заголовка, начинающегося со слов Extra! Extra!, теперь соприкасаются с буквами большего заголовка, создавая эффект единой надписи.

Можно также использовать отрицательные значения полей для того, чтобы имитировать отрицательный отступ. В третьем заголовке на рис. 7.5, который начинается со слов The Extraordinary Technique, линия подчеркивания отображена прямо под текстом. Она на самом деле представляет собой *верхнюю* границу следующего абзаца (о том, как к тексту добавить границы, вы узнаете далее). Но, поскольку здесь определена верхняя граница с отрицательным значением, она располагается чуть выше текста абзаца и фактически находится под верхним заголовком. Обратите внимание на то, что хвост буквы Q заголовка буквально свисает под линией-границей. Поскольку отступ между содержимым (то есть Q) и границей не может быть отрицательным, вам не удастся поднять нижнюю границу так, чтобы она находилась ближе к тексту или любому другому содержимому, не говоря уже о наложении. И все же есть возможность добиться этого эффекта, применив границу с отрицательным значением к последующему, нижестоящему элементу веб-страницы.

СОВЕТ

Можно использовать либо верхнее поле абзаца с отрицательным значением, либо отрицательное нижнее поле заголовка. Оба варианта приведут к одному и тому же визуальному эффекту поднятия абзаца выше, ближе к тексту заголовка.

Рис. 7.5. Чтобы превратить верхнюю границу последнего абзаца в нижнюю границу вышестоящего заголовка, вернее, имитировать такой эффект, добавим небольшой отступ. Верхний отступ шириной 5 пикселов сместит абзац вниз относительно границы, в то время как левый отступ шириной 4 em сместит текст абзаца, оставив верхнюю границу у левого края

Отображение строчных и блочных элементов

Хотя браузеры и обрабатывают любой тег веб-страницы подобно блочному элементу, на самом деле они не все одинаковы. В CSS существует два различных типа элементов: *блочные* и *строчные*.

В *блочных* элементах создается разрыв строки перед элементом и после него. Например, абзац `p` создает блок, отделенный от элементов, расположенных выше и ниже его. Другими примерами являются заголовки, контейнеры `div`, таблицы, списки и элементы списков.

Строчные элементы не создают разрывов строк ни до, ни после себя. Они отображаются на одной строке с содержимым рядом стоящих элементов. Элемент `strong` — строчный. Слово, отформатированное с его помощью, будет расположено на одной строке с текстом, заключенным в другие строчные элементы, например `em`. Было бы очень странно, если бы отдельное слово в середине абзаца, выделенное `strong`, вдруг появилось на отдельной строке. Другими примерами строчных элементов являются `img` — для добавления изображений, `a` — для создания ссылок, различные элементы для разметки форм и т. д.

В большинстве случаев каскадные таблицы стилей одинаково работают со строчными и блочными элементами. Можно применять шрифты, цвет, фон, границы к обоим типам элементов. Однако поля и отступы строчных элементов браузеры обрабатывают по-другому. Если добавить поля или отступы слева или справа строчного элемента, то посредством установки верхнего или нижнего отступа или поля увеличить высоту строчного элемента не удастся. В верхнем абзаце на рис. 7.6 строчный элемент отформатирован с применением границ, фонового цвета и полей размером 20 пикселов со всех четырех сторон. Но браузер добавляет пустые промежутки только с левой и правой стороны элемента.

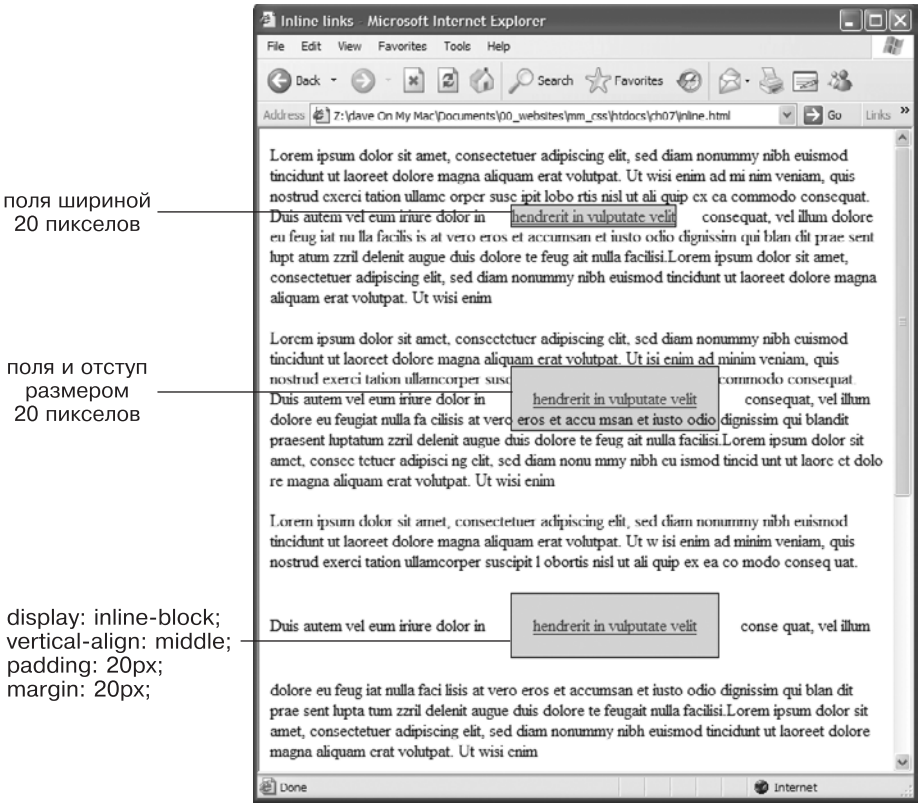

Рис. 7.6. Добавление к строчному элементу верхнего, нижнего поля или отступа не изменит высоту элемента: форматирование будет не таким, как вы ожидаете.

На рис. 7.6 в среднем абзаце фон и границы ссылки накладываются на текст, находящийся выше и ниже форматируемого. Цвет фона строчного элемента отображается поверх вышестоящей строки текста, но под следующей, так как браузер обрабатывает каждую строку так, будто она расположена в стеке, наверху по отношению к предыдущей. Как правило, это не представляет проблемы, поскольку строки текста обычно не накладываются. Если вы хотите, чтобы верхние и нижние поля были применимы к строчному элементу, используйте инструкцию `display:inline-block` (внизу). Она оставит элемент строчным, но он будет восприниматься как блочный, поэтому отступы, поля, границы, ширина и высота будут к нему применимы.

ПРИМЕЧАНИЕ

Есть одно исключение из этого правила: если поля или отступы применяются к строчным элементам `img`, элементы не изменяют своей высоты. Браузеры корректно меняют высоту контейнера элемента-изображения, чтобы подогнать указанные отступы и поля.

Иногда требуется, чтобы строчные элементы вели себя так же, как блочные, или наоборот. Маркированные списки представляют элементы в виде отдельных блоков, то есть каждый элемент списка располагается в стеке поверх следующего. Но что

делать, если вы хотите изменить поведение пунктов списка таким образом, чтобы все они располагались рядом друг с другом, на одной строке (как панель навигации, показанная на рис. 9.4)? Или, возможно, вы захотите, чтобы строчный элемент обрабатывался как блочный, например, изображение, встроенное в абзац текста, было расположено на отдельной строке, с верхним и нижним промежутками-интервалами.

В языке CSS есть команда, которая позволяет вам это сделать, — это свойство `display`. С его помощью можно заставить блочный элемент функционировать как строчный:

```
display: inline;
```

Или, наоборот, вы можете сделать так, чтобы строчные элементы, например изображение или ссылка, вели себя как блочные:

```
display: block;
```

Наконец, вы можете заставить элемент действовать и как блочный, и как строчный. Значение `inline-block` не создает разрывов ни до, ни после элемента и одновременно заставляет элемент подчиняться верхним и нижним полям и отступам, а также настройкам высоты:

```
display: inline-block;
```

ПРИМЕЧАНИЕ

У свойства display есть большое количество значений, многие из которых поддерживаются не всеми браузерами. Значение inline-block поддерживается во всех современных браузерах (см. рис. 7.6). Другое значение — none — обрабатывается в большинстве браузеров и имеет множество вариантов использования. Это значение выполняет одну простую функцию — полностью скрывает форматируемый элемент, чтобы он не отображался в окне браузера. Используя JavaScript-сценарии, вы можете скрыть элементы, чтобы они стали видимыми после присваивания свойству display значения inline или block. Сделать их видимыми можно и средствами каскадных таблиц стилей.

Добавление границ

Граница представляет собой обычную линию, которая очерчивает форматируемый элемент. Как показано на рис. 7.1, она располагается между отступом и полем элемента. С помощью границ, добавленных со всех сторон, можно заключить изображение в рамку или выделить баннер и т. д. Но совсем не обязательно применять границы, создающие очертания всего содержимого элемента. Точно так же, как вы добавляете границы с четырех сторон элемента, можно добавить требуемую границу с любой из сторон по отдельности. Эта гибкость обеспечивает добавление произвольных элементов дизайна. Например, добавьте границу слева от элемента, придайте ей толщину около 1 em, и она станет похожа на маркер в виде квадрата. Единственная граница с нижнего края абзаца производит тот же визуальный эффект, что и элемент `hr` (горизонтальная линия), выступая разделителем контента веб-страницы.

Вы можете управлять тремя различными свойствами любой из границ: `color` (цвет), `width` (ширина) и `style` (стиль). Значение `color` может быть представлено шестнадцатеричным числом, ключевым словом или значением в системе RGB (или RGBA). Ширина границы `width` — толщина линии, используемой для очерчивания.

Вы можете указывать любые единицы измерения каскадных таблиц стилей (кроме процентов) или ключевые слова `thin` (тонкая линия), `medium` (средняя) и `thick` (толстая). Самые распространенные и понятные единицы измерения для данного свойства — пикселы.

И наконец, свойство `style` управляет типом линии границы. Существует множество различных стилей. Примеры приведены на рис. 7.7. Вы можете также определить стиль с помощью ключевых слов. Например, значение `solid` рисует сплошную линию, а `dashed` — штриховую (пунктирную). В каскадных таблицах стилей для границ имеются следующие стили: `solid`, `dotted`, `dashed`, `double`, `groove`, `ridge`, `inset`, `outset`, `none` и `hidden`. Ключевые слова `none` и `hidden` работают одинаково: они полностью удаляют границы. Но значение `none` удобно использовать для удаления границы с одной стороны элемента.

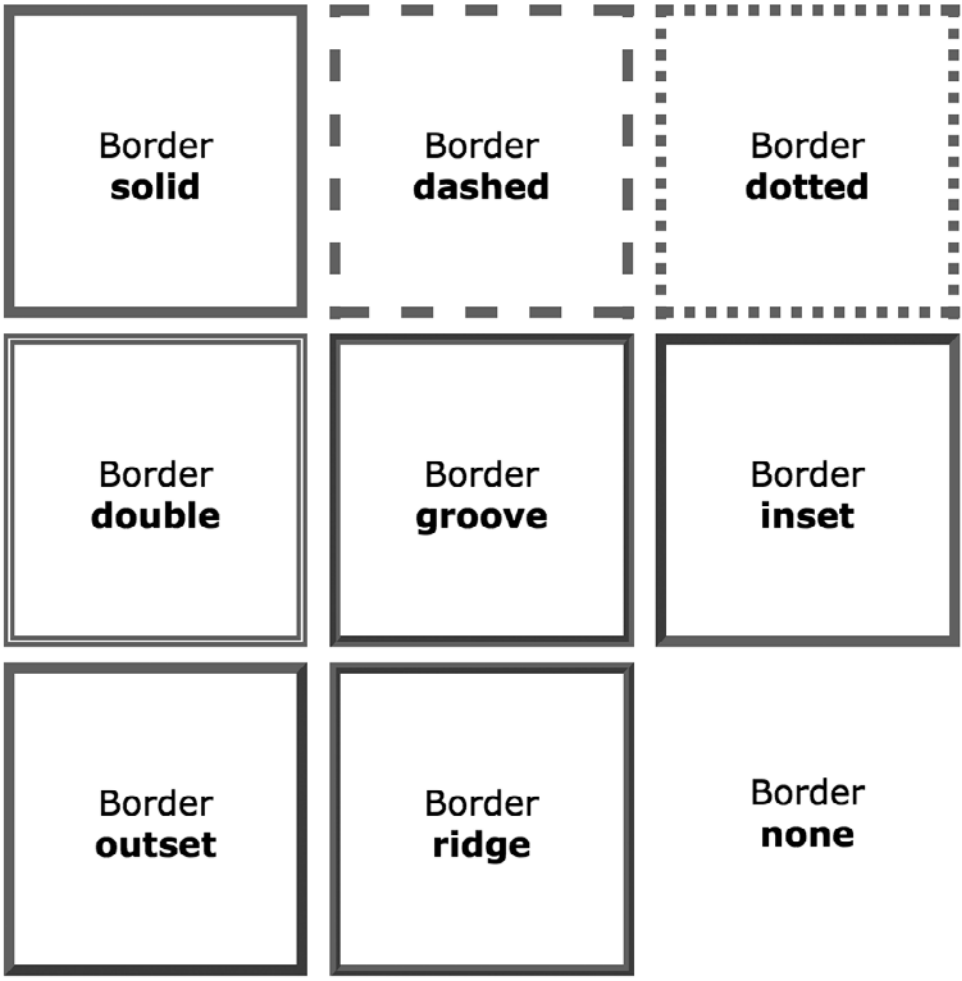

Рис. 7.7. Каскадные таблицы стилей предлагают девять различных стилей границ

Самая простая граница — сплошная (`solid`). Но если вы хотите добиться особого визуального эффекта, то можете сделать границу пунктирной или точечной, имитировать трехмерную рамку с помощью значений `groove`, `ridge`, `inset` и `outset`. Вы даже можете применить различные стили границ к разным сторонам элемента, например сделав верхнюю границу сплошной, а остальные три — пунктирными.

Сокращенная запись свойства border

Если вы взглянете на полный список свойств, доступных в языке CSS, то подумаете, что границы действительно сложны. Вообще, существует 20 разновидностей свойств границ, с которыми вы столкнетесь в следующих разделах книги, а также несколько относящихся к таблицам. Но все это — лишь варианты, обеспечивающие различные способы управления одними и теми же тремя свойствами: цвета, ширины (толщины) и стиля для каждой из четырех границ. Наиболее простое и понятное свойство — `border`, которое добавляет границы с заданными параметрами:

```
border: 4px solid rgb(255,0,0);
```

Стиль создает сплошную красную границу с толщиной линии 4 пиксела. Вы можете использовать это свойство для создания простейшей рамки, окаймляющей изображение, панель навигации или любой другой элемент, которые надо выделить в отдельный блок.

ПРИМЕЧАНИЕ

Последовательность указания свойств не имеет значения: border: 4px solid rgb(255,0,0); работает так же, как border: rgb(255,0,0) solid 4px;.

Форматирование границ по отдельности

Вы можете управлять границей с каждой стороны элемента отдельно, используя соответствующее свойство: `border-top`, `border-bottom`, `border-left` или `border-right`. Они работают точно так же, как стандартное свойство `border`, с тем исключением, что управляют границей только с одной стороны форматируемого элемента. Добавить красную пунктирную линию снизу можно, используя следующее объявление свойства:

```
border-bottom: 2px dashed red;
```

Вы можете объединять общее свойство `border` со свойствами отдельных границ. Например, использовать свойство `border-left`, чтобы определить основной, общий стиль, а затем выборочно настроить одну или несколько границ. Допустим, вы хотите, чтобы верхняя, левая и правая стороны абзаца имели одинаковый тип границы, а нижняя выглядела по-другому. Можно написать такие четыре строки кода:

```
border-top: 2px solid black;
border-left: 2px solid black;
border-right: 2px solid black;
border-bottom: 4px dashed #333;
```

Такого же эффекта можно достичь всего двумя строками кода:

```
border: 2px solid black;
border-bottom: 4px dashed #333;
```

Первая строка определяет общий вид всех четырех границ, а вторая переопределяет вид нижней границы. Преимущество не только в том, что легче написать две строки кода вместо четырех, но и в том, что изменить стиль будет проще. Если вы захотите сделать цвет верхней, левой и правой границ красным, то необходимо отредактировать единственную строку кода вместо трех:

```
border: 2px solid red;
border-bottom: 4px dashed #333;
```

При использовании этого сокращенного способа установки границ определяется общий вид всех четырех границ. Затем вид одной из границ замещается с помощью свойства конкретной границы, например `border-left`. Очень важно, чтобы свойства были написаны в определенной последовательности. В общем случае глобальные установки границ должны быть на первом месте, а установки отдельной границы — на втором:

```
border: 2px solid black;
border-bottom: 4px dashed #333;
```

Поскольку свойство нижней границы `border-bottom` указано вторым, оно частично замещает общие установки свойства `border`. Если бы свойство `border-bottom` было расположено первым, то `border` было бы отменено и все четыре границы стали бы одинаковыми. Последнее явное свойство может заместить любые аналогичные свойства, определенные в коде выше. Это пример работы механизма каскадности CSS, который мы рассматривали в главе 5.

Вы также можете использовать этот сокращенный способ определения границ, чтобы скрыть их с помощью ключевого слова `none`. Предположим, вы хотите установить границы только с трех сторон элемента (сверху, слева, снизу). Следующие две строки кода обеспечат такое форматирование:

```
border: 2px inset #FFCC33;
border-right: none;
```

Возможность тонкой настройки границ каждой из сторон форматируемого элемента является причиной большого количества разновидностей свойств границ. Остальные 15 свойств позволяют определять индивидуальные цвета, стили, толщину линий границ для каждой стороны. Например, можно переписать определение `border: 2px double #FFCC33;` следующим образом:

```
border-width: 2px;
border-style: double;
border-color: #FFCC33;
```

В этом варианте используются три строки кода вместо одной, поэтому вы, наверное, будете избегать такого способа. Однако каждая сторона имеет свой набор из трех свойств, которые удобно использовать для отмены одного. Правая граница: `border-right-width`, `border-right-style` и `border-right-color`. Левая, верхняя и нижняя границы имеют похожие свойства: `border-left-width`, `border-left-style` и т. д.

Вы можете изменить ширину единственной стороны границы так: `border-right-width: 4px;`. При таком подходе хорошо то, что, когда вы позже решите изменить границу на сплошную, нужно будет отредактировать только групповое свойство границы, изменив значение `dashed` на `solid`.

Кроме того, вы можете задать собственные значения для каждой стороны границы, используя свойства `border-width`, `border-style` и `border-color`. Например, правило `border-width: 10px 5px 15px 13px;` применит четыре различных значения ширины для каждой из сторон (верхней, правой, нижней и левой).

Допустим, вы хотите установить все четыре границы элемента в виде пунктирной линии толщиной 2 пиксела, но при этом нужно, чтобы каждая граница имела свой цвет (например, вы создаете сайт для детей). Ниже представлен способ быстро сделать это:

```
border: 2px dashed;
border-color: green yellow red blue;
```

Этот набор правил создает границы в виде двухпиксельной пунктирной линии со всех четырех сторон элемента, при этом верхняя граница будет иметь зеленый цвет, правая — желтый, нижняя — красный, а левая — синий.

ПРИМЕЧАНИЕ

Как правило, при использовании границ требуется добавлять отступы. Они обеспечивают промежутки между границами и содержимым элементов: текстом, изображениями, прочими элементами. Обычно границы отображаются слишком близко к содержимому элементов, только если вы не захотите разместить их вокруг изображения.

Установка цвета фона

В языке CSS имеются средства для добавления фона как для всей веб-страницы, так и для отдельного заголовка или любого другого элемента страницы. Используйте свойство `background-color` в сочетании с любым из действительных определений цветов, которые описаны в разделе «Форматирование текста цветом» главы 6. При желании вы можете окрасить фон веб-страницы в яркий зеленый цвет, указав следующий код:

```
body { background-color: rgb(109,218,63); }
```

Или можете создать класс, например, `.review` со свойством `background-color` и значением желаемого цвета фона, а затем применить его к элементу `body` в HTML-коде таким образом: `<body class="review">`.

ПРИМЕЧАНИЕ

Вы можете также поместить изображение на заднем плане в качестве фона веб-страницы и управлять его положением различными способами. Мы рассмотрим это в следующей главе. Кроме того, к фону любого элемента можно добавить цветовой градиент, о котором вы узнаете в следующей главе.

Цвет фона удобно применять для создания множества различных визуальных эффектов. Вы можете придать заголовку контрастность, рельефность, установив темный цвет для фона и светлый — для текста. Цвет фона также является отличным

средством для выделения таких обособленных частей веб-страницы, как панель навигации, баннер или боковая панель.

И не забывайте о ранее рассмотренном методе задания цвета по модели RGBA. С его помощью можно сделать фон полупрозрачным, позволяя просматриваться находящимся позади него цвету, текстурам или изображениям. Например, можно сделать желто-коричневый фон страницы. Затем, предположим, возникло желание придать находящемуся внутри элементу `div` более светлый оттенок желто-коричневого цвета. Вместо задания для фона элемента `div` сплошного цвета можно добавить белый цвет, а затем управлять прозрачностью этого цвета, чтобы через него просвечивались различные оттенки желто-коричневого цвета:

```
body {
  background-color: rgb(247,226,155);
}
.special-div {
  background-color: rgba(255,255,255,.75);
}
```

ПРИМЕЧАНИЕ

Когда вы пользуетесь одновременно фоновым цветом и границами, помните: если стиль границы — точечная или пунктирная линия (см. рис. 7.7), то фоновый цвет отображается в промежутках между точками или штрихами линий границ. Другими словами, браузеры размещают линию границ поверх цвета фона. Тем не менее вы можете обойти это поведение, присвоив свойству background-clip значение padding-box:

background-clip: padding-box;

Скругление углов

Как уже ранее упоминалось, браузеры рассматривают все элементы как строго прямоугольные блоки. Это становится очевидным при установке границы вокруг абзаца или элемента `div`. На этот случай есть возможность сгладить острые углы таких прямоугольников, добавив к стилям скругленные углы (рис. 7.8). В языке CSS представлено свойство `border-radius`, позволяющее дизайнерам добавлять скругления к одному или нескольким углам элемента. В самом простом варианте свойство `border-radius` получает одно значение, которое затем применяется ко всем четырем углам элемента:

```
.specialBox {
  background-color: red;
  border-radius: 20px;
}
```

Браузер использует предоставленное значение радиуса для выписывания круговой траектории в каждом углу элемента. Как показано на рис. 7.9, значение, равное расстоянию от центра окружности до ее края, является ее радиусом. В качестве единиц измерения чаще всего применяются пикселы и em, но можно использовать и проценты (хотя они ведут себя немного неожиданно).

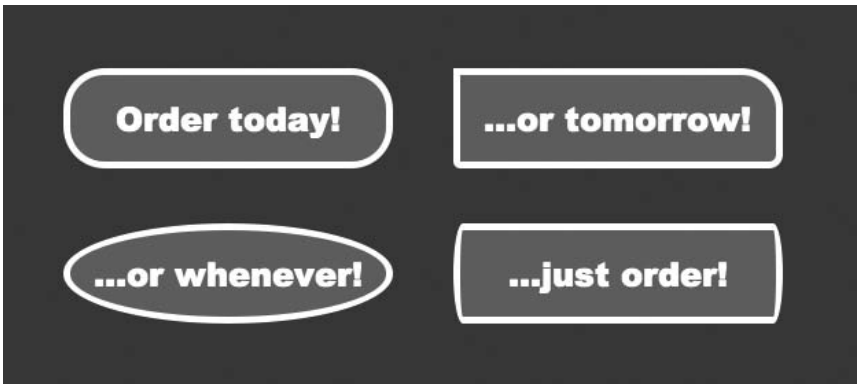

Рис. 7.8. В языке CSS позволяется скруглять углы элемента. Чтобы увидеть любое из этих удивительных закруглений, у элемента должны быть цветной фон или граница

При использовании единственного значения браузер рисует закругления одинакового радиуса для каждого угла элемента. Например, для верхнего левого изображения на рис. 7.8 используется следующее объявление:

```
border-radius: 30px;
```

Но указанием одного и того же значения для каждого угла настройки не ограничиваются. Для каждого угла можно предоставить отдельные значения, задав четыре параметра. Например, у верхнего правого блока на рис. 7.8 четыре разных угла. Объявление имеет следующий вид:

```
border-radius: 0 30px 10px 5px;
```

Сначала задается числовое значение для левого верхнего угла блока, а затем по часовой стрелке для всех остальных углов. То есть первое значение (`0` в примере на рис. 7.8) применяется к левому верхнему, второе (`30px`) — к правому верхнему, третье (`10px`) — к правому нижнему и четвертое (`5px`) — к левому нижнему углу. Можно задать только два значения, тогда первое число будет применено к левому верхнему и правому нижнему углам, а второе — к правому верхнему и левому нижнему углам.

Кроме рассматриваемых до сих пор абсолютно круглых углов (то есть представляющих собой часть окружности), можно задавать эллиптические углы, подобные тем, что показаны в двух нижних примерах на рис. 7.8. Для эллиптической формы угла требуется два значения радиуса: первое для радиуса от центра до левого или правого края, а второе — для определения расстояния от центра до верхнего или нижнего края. Например, чтобы добавить углы, показанные в левом нижнем углу на рис. 7.8, нужно создать следующее объявление:

```
border-radius: 40px/20px;
```

Значение `40px` задает горизонтальный радиус, а значение `20px` — вертикальный. Слеш между ними оповещает браузер о создании эллиптического угла. Можно сделать так, что у каждого из четырех углов будут разные вытянутые формы, указав различные значения каждого из радиусов эллипса. Это может показаться немного странным:

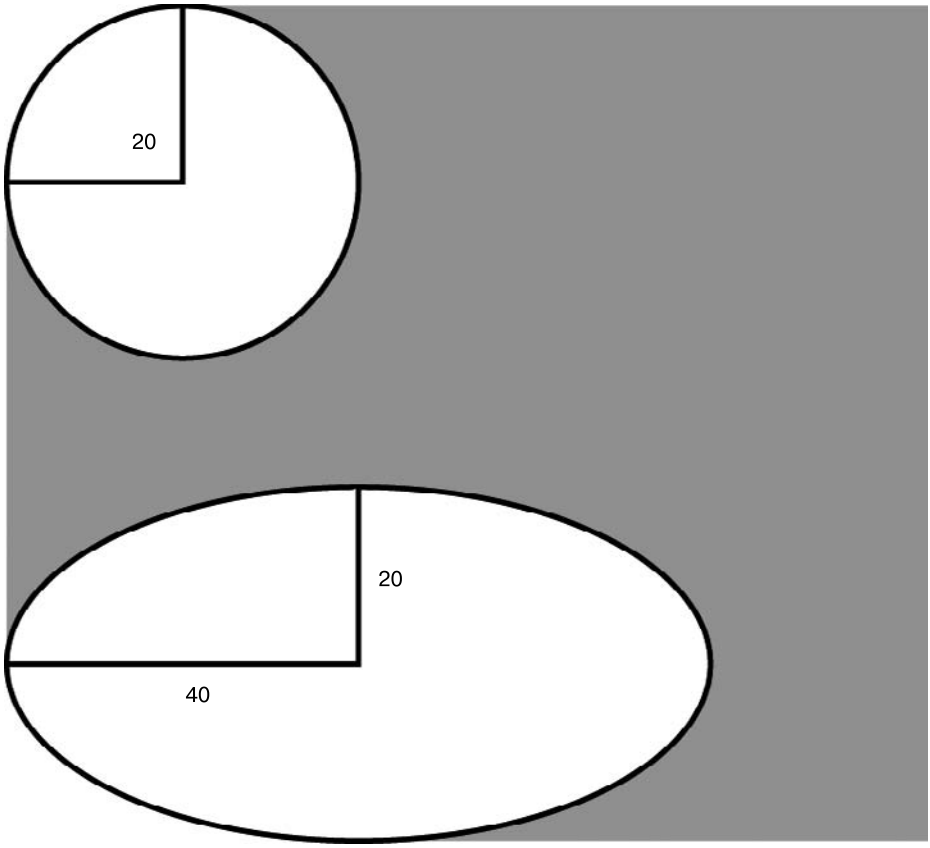

Рис. 7.9. Можно создавать не только скругленные *(вверху)*, но и эллиптические углы *(внизу)*

сначала указываются четыре значения для горизонтальных радиусов каждого из углов (начиная с левого верхнего угла), а затем после каждого из значений указываются обратный слеш (/) и четыре значения вертикального радиуса каждого угла:

```
border-radius: 40px 10px 20px 10px / 20px 30px 40px 20px;
```

С помощью аналогичного синтаксиса можно даже смешивать эллиптические и скругленные углы. Горизонтальный и вертикальный радиусы у закругленных углов одинаковые:

```
border-radius: 10px 10px 20px 10px / 10px 30px 40px 10px;
```

И наконец, если нужно пойти по более длинному пути, то для определения формы каждого угла можно воспользоваться отдельными свойствами. Например, правило:

```
border-radius: 1em 2em 1.5em .75em;
```

можно также записать следующим образом:

```
border-top-left-radius: 1em;
border-top-right-radius: 2em;
```

```
border-bottom-right-radius: 1.5em;
border-bottom-left-radius: .75em;
```

ПРИМЕЧАНИЕ

Браузер Internet Explorer 8 и более ранние версии не поддерживают свойство border-radius, поэтому при его объявлении будут показаны прямые углы.

Добавление тени

Как уже упоминалось, чтобы сделать текст объемным, к нему можно добавить еле заметную (или весьма заметную) отбрасываемую тень. Для добавления теней к блоку, обрамляющему элемент, используется свойство `box-shadow`, позволяющее, к примеру, блоку `div` казаться парящим над страницей (рис. 7.10). По сравнению с `text-shadow`, у этого свойства есть несколько дополнительных настроек. Например, можно сделать так, чтобы тень появлялась *внутри* блока, как показано в нижней части рис. 7.10.

Свойство `box-shadow` работает в большинстве современных браузеров, включая Internet Explorer 9. При этом IE 9 рисует тени заметно тоньше, чем другие браузеры. Кроме того, браузер IE 8 и более ранние версии не поддерживают это свойство и не отобразят тени у элементов.

Основной синтаксис свойства `box-shadow` показан на рис. 7.11. Первое значение задает горизонтальное смещение, то есть это значение приводит к перемещению тени влево или вправо от элемента. Положительное число приводит к перемещению тени вправо (верхний блок на рис. 7.10), а отрицательное число — влево.

Второе значение задает вертикальное смещение — позицию тени либо над элементом, либо под ним. При положительном значении тень помещается ниже нижнего края блока (верхний блок на рис. 7.10), а при отрицательном значении тень помещается над верхним краем блока.

ПРИМЕЧАНИЕ

Для задания значений тени нужно использовать пикселы или единицы em. Проценты не поддерживаются.

Третье значение задает радиус размытия тени. Оно определяет степень размытости и ширины тени. Значение `0` не придает эффекта размытости, то есть получается тень с четкими краями. Чем выше значение, тем более размытой и тусклой становится тень.

И наконец, последнее значение задает цвет отбрасываемой тени. Можно воспользоваться любым обозначением цвета, принятым в языке CSS, но RGBA-значения смотрятся намного лучше, потому что позволяют управлять прозрачностью тени, делая ее более реалистичной.

Для свойства `box-shadow` доступны два дополнительных значения: ключевое слово `inset` и значение расширения. Ключевое слово `inset` помещает тень внутрь блока (см. рис. 7.10, *внизу*). Для создания внутренней тени нужно добавить слово `inset` в качестве первого значения свойства `box-shadow`:

```
box-shadow: inset 4px 4px 8px rgba(0,0,0,.75);
```

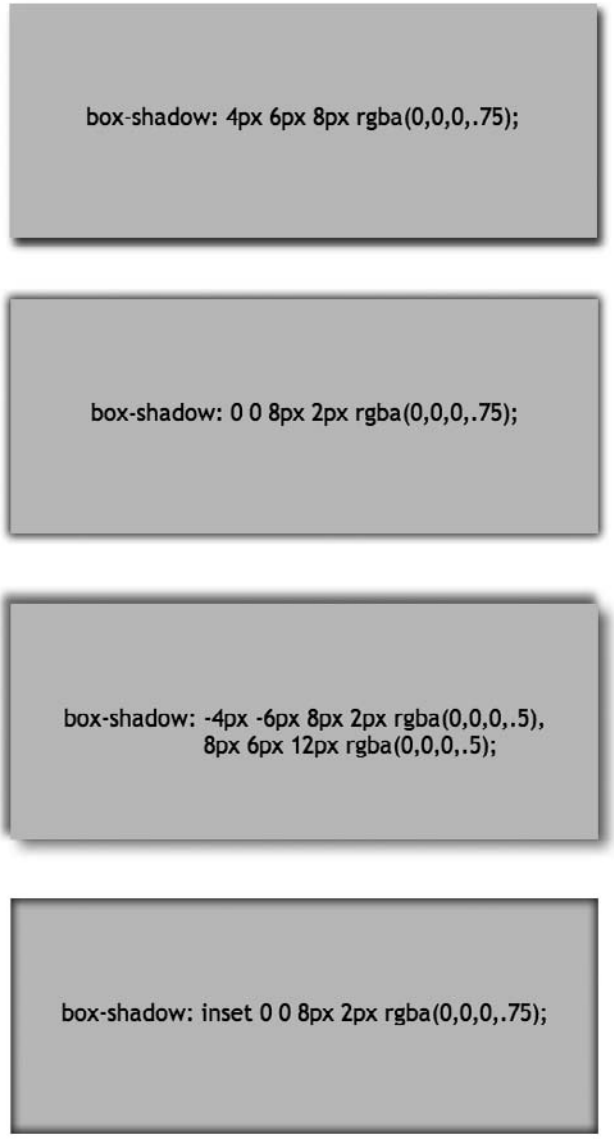

Рис. 7.10. Свойство box-shadow позволяет добавлять тени к элементам, создавая эффект парения над страницей

В качестве четвертого значения (между радиусом размытия тени и ее цветом) можно также добавить *расширение*. Это приведет к расширению тени на указанное значение. Иначе говоря, если добавить значение расширения, равное `10px`, браузер расширит тень на 10 пикселов в каждом направлении (в целом тень получается на 20 пикселов шире и на 20 пикселов выше). Это значение также задает позицию, в которой применяется радиус размытия. То есть, если добавлено расширение,

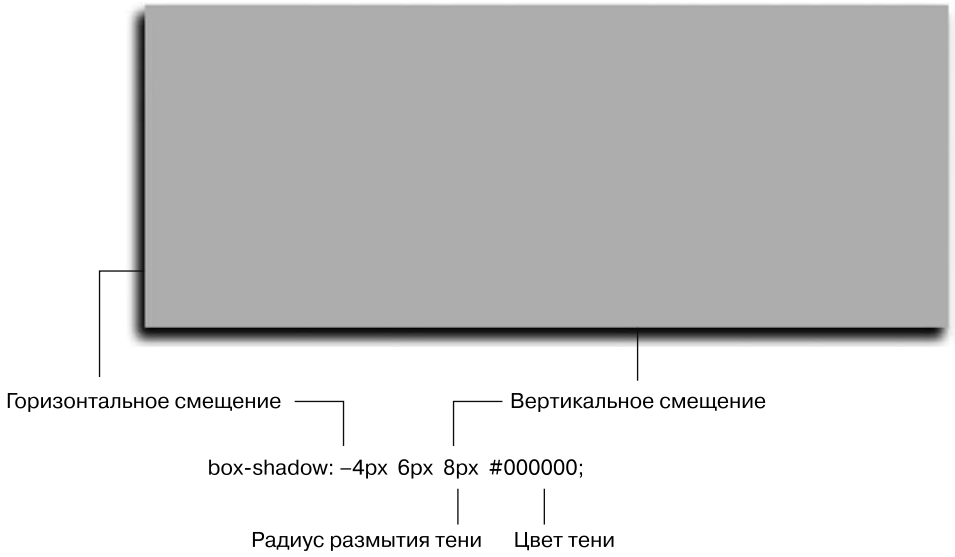

Рис. 7.11. Чаще всего тень размещается либо слева (при отрицательном значении, как показано на рисунке) элемента, либо справа (при положительном значении), либо выше верхнего края (при отрицательном значении), либо ниже нижнего края (при положительном значении) с ее размытием согласно указанному радиусу

размытие тени не начинается до тех пор, пока не будет применено значение расширения. В частности, это пригодится в том случае, когда нужно добавить тень вокруг всего элемента, добиваясь эффекта *свечения*.

Например, во втором сверху блоке на рис. 7.10 горизонтальное и вертикальное смещение равны `0`, для радиуса тени установлено значение `8px`, а для расширения — `2px`. Значение расширения выталкивает тень на 2 пиксела наружу относительно всех четырех сторон блока, а затем радиус тени, равный 8 пикселам, расширяет размытие еще на 8 пикселов. Значение расширения можно даже использовать для создания вокруг существующей границы второй, по-другому расцвеченной тени. Вот как выглядит пример кода для этого эффекта:

```
border: 10px solid rgb(100,255,30);
box-shadow: 0 0 0 10px rgb(0,33,255);
```

И наконец, можно даже применить к элементу несколько теней (третий блок на рис. 7.10). Для этого нужно указать запятую после первого набора установок тени, а затем добавить еще одну тень:

```
box-shadow: 10px 5px 8px #FF00FF,
-5px -10px 20px 5px rgb(0,33,255);
```

Количество теней не ограничено (только здравым смыслом).

ПРИМЕЧАНИЕ

Эффект тени заставляет браузер выполнять дополнительные вычисления. Используйте тень аккуратно. Кроме того, не забудьте проверить работу страницы с эффектом тени на мобильных устройствах, вычислительные ресурсы которых гораздо ниже, чем у настольных компьютеров и ноутбуков.

Изменение высоты и ширины

Рассмотрим еще два свойства, являющихся частью блочной модели CSS. Они предназначены для установки размеров объектов, таких как таблица, колонка, баннер, боковая панель. Свойства `height` и `width` определяют высоту и ширину области форматируемого элемента. Мы будем часто пользоваться ими при создании разметки, макета веб-страниц, как описано в части III. Они также применяются для разработки базового дизайна: назначения ширины таблиц, создания простейших боковых панелей или галерей эскизов.

Разработка стилей с этими свойствами не составляет сложностей. Наберите их со значением в любой единице измерения языка CSS, которые мы изучили. Например:

```
width: 300px;
width: 30%;
height: 20em;
```

Пикселы просты в использовании, понятны и удобны, обеспечивают точные ширину или высоту. Единица измерения *em* — это примерно то же самое, что и размер шрифта текста. Допустим, вы устанавливаете размер шрифта 24 пиксела; единица em для этого форматируемого элемента будет равна 24 пикселам, а если вы установите ширину равной 2 em, она составит 2 × 24 или 48 пикселов. Если вы не определите в стиле размер шрифта текста, то он будет взят из унаследованных параметров.

Процентные значения свойства `width` рассчитываются на основании ширины элемента-контейнера. Если вы установите ширину заголовка равной 75 % и этот заголовок не вложен ни в какие другие элементы веб-страницы с явно определенной шириной, то ширина заголовка составит 75 % от ширины окна браузера. Если посетитель изменит размер окна браузера, то ширина заголовка тоже изменится. Однако если заголовок заключен в блок `div` шириной 200 пикселов (к примеру, для создания колонки), то ширина этого заголовка составит 150 пикселов. Процентные значения в свойстве `height` работают точно так же, но расчет базируется на высоте элемента-контейнера, а не на его ширине.

Вычисление фактических размеров блочных элементов

Свойства `width` и `height` на первый взгляд кажутся довольно простыми и понятными, однако есть несколько нюансов, вводящих начинающих веб-дизайнеров в заблуждение. Прежде всего, существует различие между значениями ширины и высоты, которые вы явно указываете при написании стилей, и размером пространства, которое браузер фактически выделяет и использует для отображения элементов блочной модели CSS. Свойства `width` и `height` устанавливают ширину и высоту *области содержимого* форматируемого элемента — пространства, в котором заключены текст, изображения или другие вложенные элементы (см. рис. 7.1, чтобы вспомнить о том, где именно в блочной конструкции элементов CSS находится область содержимого).

Фактическая ширина элемента веб-страницы представляет собой область экрана (окна браузера), выделяемую для отображения. Она состоит из ширины полей, границ, отступов и явно указанного значения ширины в свойстве стиля, как показано на рис. 7.12.

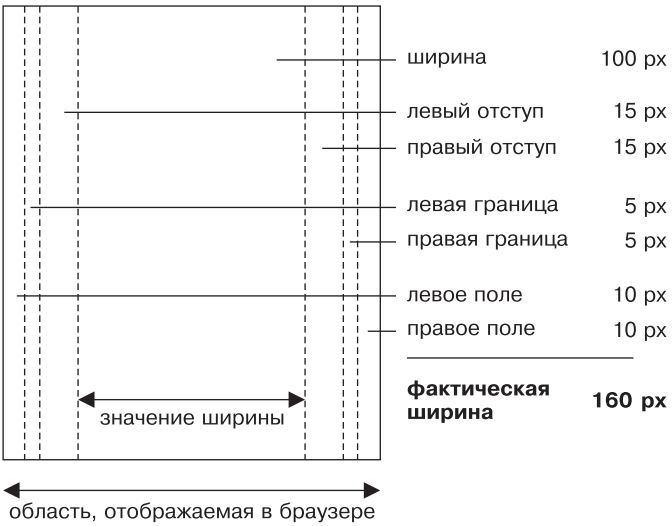

Рис. 7.12. Фактическая ширина блока форматируемого элемента вычисляется путем сложения значений свойств margin, border, padding и width. Высота экранного элемента рассчитывается исходя из суммы значения свойства height, верхних и нижних полей, верхней и нижней границ и верхнего и нижнего отступа

Допустим, вы назначили следующие свойства:

```
width: 100px;
padding: 15px;
border-width: 5px;
margin: 10px;
```

Если определено свойство `width`, вы всегда точно знаете, сколько места займет содержимое элемента — текст и изображения, заполняющие основное пространство элемента, — независимо от любых других установленных свойств. Вам не нужно ничего вычислять, потому что значение `width` и есть размер основного пространства для размещения содержимого (в представленном выше примере ширина равна 100 пикселам). Конечно, *придется* выполнить несложные арифметические операции, чтобы выяснить общий точный размер. В представленном выше примере для размещения форматируемого элемента отводится пространство шириной 160 пикселов: 20 пикселов для левого и правого полей, 10 — для левой и правой границ, 30 — для левого и правого отступа и 100 пикселов в качестве ширины основного содержимого.

Общее правило по регулированию высоты элементов на странице гласит так: *не делайте* этого! Многие подающие надежды веб-дизайнеры пытаются задать высоту абсолютно для всего, желая получить полный контроль над пикселами.

Но если вы полностью не уверены в точных размерах содержимого внутри элемента, то можете столкнуться с некоторыми нежелательными результатами (рис. 7.13). В этом примере блок с цитатой, который используется для того, чтобы акцентировать внимание на интересном отрывке из статьи, имеет ширину и высоту 100 пикселов. Когда в блок добавляется больше текста, чем может уместиться в такую высоту, его содержимое выходит за пределы. Даже если вы уверены, что текст, который вы разместили в блоке с фиксированной высотой, соответствует его размерам, посетитель может увеличить размер шрифта в своем браузере и высота текста, соответственно, станет больше высоты блока.

Другими словами, свойство `height` полезно для контроля высоты элемента `div`, содержащего, например, изображения, потому что в таком случае вы можете правильно определить его высоту. Однако если вы используете это свойство для элементов с текстом, не забудьте не только протестировать свои страницы в основных браузерах, но и проверить их при различных установленных размерах шрифта, изменяя его в браузерах.

Рис. 7.13. Когда устанавливается высота элемента (например, высота контейнера div правой боковой панели, как в данном примере), содержимое которого выше самого элемента, браузеры позволяют содержимому выйти за нижнюю границу элемента

СОВЕТ

Область баннера на странице — еще один подходящий кандидат для установки высоты. Обычно баннер имеет ограниченное содержимое: логотип, поле поиска, может быть, какие-нибудь навигационные кнопки. Зачастую у баннеров остается довольно много пустого пространства (помогающего привлечь внимание посетителя к ключевым элементам навигационной панели), поэтому указание высоты для баннера обычно не вызывает проблем.

Переопределение ширины блока с помощью свойства box-sizing

Как уже упоминалось, браузеры традиционно вычисляют ширину элемента, складывая значения свойств `border`, `padding` и `width`. Это не только заставит вас проводить математические вычисления, чтобы определить реальную ширину отображаемого элемента, но и может вызвать ряд других проблем. Особенно это касается случаев создания макетов с плавающими элементами с использованием процентных отношений. Подробности создания макетов с плавающими элементами нам еще предстоит изучить, но если говорить вкратце, каскадные таблицы стилей позволяют с помощью свойства `float` помещать элементы бок о бок, что дает возможность создавать разметки, состоящие из нескольких колонок.

Когда для нескольких колонок используется процентное отношение, могут возникать довольно странные проблемы. Предположим, есть две колонки (на самом деле два таких элемента, как `div`) и нужно, чтобы каждая занимала 50 % ширины окна. Соответственно, для двух колонок устанавливается ширина 50 %, но на тот момент, когда добавляется отступ или граница к одной из колонок, вы увеличиваете ее ширину, которая становится больше 50 % (если точнее, она составляет 50 % плюс значение левого и правого отступов и значение ширины левой и правой границ). В большинстве случаев это заставит вторую колонку опуститься *ниже* первой.

На этот случай каскадные таблицы стилей предлагают свойство, позволяющее изменить порядок вычисления браузером экранной ширины (и высоты) элемента. Свойство `box-sizing` предоставляет три варианта.

- `content-box` устанавливает ранее рассмотренный способ, с помощью которого браузеры всегда определяют экранную ширину и высоту элемента. То есть браузер добавляет ширину границ и толщину отступа к значениям, установленным для свойств ширины и высоты, чтобы определить экранную ширину и высоту заданного элемента. Поскольку это поведение используется по умолчанию, указывать что-либо для реализации варианта `content-box` не нужно.
- `padding-box` сообщает браузеру, что при установке для стиля свойства ширины или высоты они должны включать отступы как часть своего значения. Предположим, что есть элемент с отступами слева и справа, равными 20 пикселам. Его ширина установлена равной 100 пикселам. Браузер будет рассматривать ту часть, что приходится на отступы, частью этого 100-пиксельного значения. То есть ширина области содержимого будет равна всего лишь 60 пикселам (100 — 20 [левый отступ] — 20 [правый отступ]).
- `border-box` сообщает браузеру о необходимости включения в качестве составляющей части значений свойств `width` и `height` толщину как отступа, так и границ. Эта настройка решает ту проблему использования значений, выраженных в процентном отношении, о которой говорилось выше. То есть когда свойству `box-sizing` присвоено значение `border-box`, при установке ширины элемента, равной 50 %, этот элемент будет занимать до 50 % пространства, даже если к нему будут добавлены отступы и границы.

Если вам не нравится стандартный способ вычисления браузером элементов ширины и высоты, добавьте значение `border-box`. (Если, конечно, у вас нет какой-нибудь причины, по которой вы хотите включить в расчеты отступ, но не хотите включать туда еще и границу.) Чтобы воспользоваться свойством `box-sizing`, предоставьте ему одно из трех значений из списка. Например:

```
box-sizing: border-box;
```

Между прочим, многие веб-разработчики настолько оценили пользу настройки `border-box`, что создали универсальный селектор, применяемый к каждому элементу на странице:

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

В главе 14 будет показано, что особая польза от применения значения `border-box` проявится при решении определенных проблем, возникающих при верстке CSS-кода.

Управление блочными элементами с помощью свойства overflow

Когда содержимое форматируемого элемента имеет размеры больше определенных свойствами `width` и `height`, происходят странные вещи. На рис. 7.13 показано, что в браузерах содержимое отображается за пределами (выступает наружу) границ элемента, часто накладываясь на него.

Браузер использует в этой ситуации свойство `overflow`. В качестве значения можно указать одно из четырех ключевых слов, определяющих, как должно отображаться содержимое, выходящее за пределы блочного элемента.

- `visible` — это значение, используемое браузером по умолчанию. Указание этого ключевого слова имеет тот же эффект, что отсутствие установки свойства `overflow` (рис. 7.14, *вверху*).
- `scroll` — позволяет добавить полосы прокрутки (см. рис. 7.14, *посередине*). Параметр создает своего рода окно мини-браузера внутри веб-страницы, которое выглядит подобно устаревшим HTML-фреймам. Вы можете использовать ключевое слово `scroll`, чтобы вместить объемное содержимое в ограниченную область. При таком варианте полосы прокрутки отображаются *всегда*, даже если содержимое помещается внутри блока.
- `auto` — пользуйтесь этим значением, чтобы сделать полосы прокрутки необязательными. Оно выполняет ту же функцию, что и `scroll`, с одним исключением — полосы прокрутки в данном случае появляются только при необходимости.
- `hidden` — скрывает любое содержимое, выходящее за пределы блочного элемента (см. рис. 7.14, *внизу*). Это значение небезопасно, поскольку может привести к тому, что часть содержимого будет скрыта.

Рис. 7.14. Свойство overflow предоставляет три простых способа отображения текста, размеры которого не позволяют браузеру отобразить его внутри блочного элемента: visible показывает весь контент *(вверху)*, scroll и auto добавляют полосы прокрутки *(посередине)*, а hidden отображает только тот контент, который не выходит за границы области *(внизу)*

Задание максимальных и минимальных значений высоты и ширины

Если вы еще не поняли, хочу вас заверить, что каскадные таблицы стилей предлагают множество гибких решений. Кроме стандартных свойств `width` и `height`, можно воспользоваться еще четырьмя их вариантами.

- Свойство `max-width` устанавливает максимальную ширину элемента. Этот элемент может быть у́же установленного предела, но не может быть шире. Такой вариант пригодится в том случае, когда нужно, чтобы ваша страница меняла свои размеры, чтобы поместиться на экранах разной ширины, но при этом нужно, чтобы она не становилась настолько широкой, что это затруднит ее чтение на мониторах с большой диагональю. Предположим, на страницу добавлен следующий стиль:

  ```
  body {
    max-width: 1200px;
  }
  ```

  Этот стиль позволяет странице перестраивать контент, чтобы поместиться по ширине на небольших экранах смартфонов или планшетных компьютеров. Но на действительно больших настольных мониторах страница не должна быть шире 1200 пикселов, то есть она не может разрастаться в ширину, при которой ее контент уже невозможно читать.

- Свойство `max-height` работает во многом похоже на `max-width`, за исключением того, что оно применяется к высоте элемента. Но, как уже упоминалось, с высотой элемента лучше все же не связываться.

- Свойство `min-width` устанавливает минимальную ширину элемента. Элемент может растянуться шире значения минимальной ширины, но никогда не может стать уже этого значения. Если, к примеру, вы заметили, что при изменении размеров окна вашего браузера элементы становятся настолько узкими, что макет рассыпается, можно с помощью следующего кода установить ограничение минимальной ширины элемента:

  ```
  body {
    min-width: 760px;
  }
  ```

  Если посетитель уменьшит окно своего браузера менее чем до 500 пикселов по ширине, браузер добавит полосу прокрутки, не сужая элементы на странице до крайности.

- Свойство `min-height` работает точно так же, как и `min-width`, но применительно к высоте. Оно позволяет решить проблему, показанную на рис. 7.13. Ограничивая минимальную высоту, вы заставляете браузер сделать элемент по крайней мере имеющим определенную высоту. Если содержимое элемента выше, то браузер сделает выше весь элемент.

Управление обтеканием контента с помощью плавающих элементов

HTML-контент в обычном порядке отображается, начиная с верхнего края окна браузера по направлению к нижнему: один заголовок, абзац, блочные элементы друг за другом. Такой способ представления неинтересен, но благодаря каскадным таблицам стилей предоставляется возможность изменить дизайн в лучшую сторону. В части III мы рассмотрим много новых методов компоновки, размещения, упорядочения элементов, но уже сейчас вы можете добавить привлекательности своим веб-страницам посредством свойства `float`.

Свойство `float` перемещает любой элемент в нужную позицию, выравнивая его по левому или правому краю веб-страницы. В процессе перемещения содержимое, находящееся ниже позиционируемого плавающего элемента, смещается вверх и плавно обтекает сам плавающий элемент (рис. 7.15, *внизу*). Плавающие (или перемещаемые) элементы — идеальное средство для того, чтобы выделить дополнительную информацию из основного текста. Изображения могут смещаться к любому краю веб-страницы, обеспечивая перенос расположенного рядом текстового контента и его изящное обтекание вокруг изображений. Точно так же вы можете переместить боковую панель с относящейся к тексту информацией и элементы навигации к одной из сторон веб-страницы.

На рис. 7.15 приоритет элементов следующий: заголовок, абзац, контейнер `div` и т. д. Свойство `float` позволяет нарушать этот порядок, ввиду чего и является одним из самых мощных инструментов каскадных таблиц стилей. Широкий диапазон его применения не только включает простое позиционирование, перемещение изображений к одной из сторон абзаца, но и обеспечивает полный контроль по управлению размещением баннеров, меню, панелей навигации и других элементов веб-страниц.

Несмотря на то что свойство `float` может быть использовано для сложного, комплексного форматирования (об этом вы прочитаете в главе 13), его применение не представляет никаких сложностей. В качестве значения указывается одно из трех ключевых слов — `left`, `right` или `none`. К примеру:

```
float: left;
```

- `left` — перемещает форматируемый элемент влево, при этом содержимое, находящееся ниже, обтекает правый край элемента.
- `right` — перемещает элемент вправо.
- `none` — отменяет обтекание и возвращает объект в его обычную позицию. Это свойство определяет нормальное поведение элемента, поэтому используйте его, если хотите отменить обтекание слева или справа, примененное к другому стилю (правила взаимодействия нескольких стилей обсуждались в главе 5).

Плавающие элементы могут быть расположены у левого или правого края содержащего их *элемента-контейнера*. В некоторых случаях это означает, что элемент перемещается к левому или правому краю окна браузера. Однако если вы

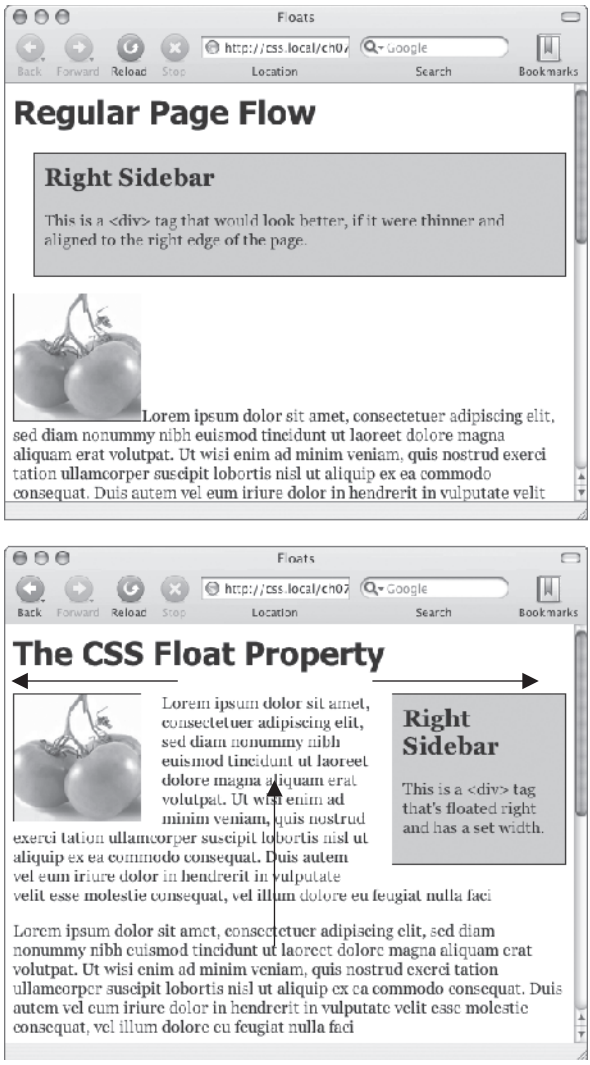

Рис. 7.15. Стандартная последовательность отображения HTML-контента слева направо, сверху вниз

перемещаете элемент, находящийся внутри другого элемента, для которого установлены значения ширины или позиции на веб-странице, то перемещение будет осуществлено к левому или правому краю *этого элемента*, который является контейнером для плавающего элемента. Например, на веб-странице может быть блочный элемент шириной 30 % от ширины окна браузера, который прилегает к правому краю окна браузера. Внутри может располагаться изображение, которое выровнено по левому краю. Иными словами, изображение примыкает к левому краю этого блока, а не окна браузера.

Вы можете применять свойство `float` к строчным элементам, например `img`. Надо сказать, использование выравнивания по левому или правому краю для фотографий — самый обычный, даже наиболее распространенный способ применения свойства `float`. Браузер обрабатывает строчные элементы точно так же, как блочные. Поэтому, добавляя отступы и поля в строчных элементах, вы избежите проблем, которые обычно возникают в такой ситуации.

Вы можете также использовать свойство `float` для блочных элементов, таких как заголовки или абзацы текста (или HTML5-элементы, например `article`, `section` и `aside`). Общая методика применения свойства `float` к элементу `div`, содержащему другие HTML-элементы, заключается в создании своего рода блока-контейнера. Так вы можете создавать боковые панели, плавающие цитаты и другие обособленные элементы веб-страниц (пример приведен в практикуме этой главы). При использовании свойства `float` для блочных элементов рекомендуется установить для них свойство `width` (на самом деле правила CSS требуют установки ширины для всех плавающих элементов, кроме изображений). Таким образом, вы можете управлять размером блока по горизонтали, оставив промежуток для текстового контента, расположенного после этого блочного элемента, и, соответственно, задать его поведение — обтекание.

ПРИМЕЧАНИЕ

Порядок написания HTML-кода оказывает большое влияние на отображение плавающих элементов веб-страницы. HTML-код плавающих элементов должен находиться прежде любого обтекающего HTML-контента. Допустим, вы создали веб-страницу, на которой имеется элемент заголовка h1, а за ним расположен абзац p. Внутри абзаца p вы вставили фотографию с помощью элемента img. Теперь, если вы установите выравнивание фотографии по правому краю, заголовок h1 и часть содержимого абзаца p будут по-прежнему отображены над фотографией. Только содержимое, следующее за элементом img, будет обтекать изображение с левой стороны.

Фон, границы и обтекание

К недовольству многих веб-дизайнеров, фон и границы не переносятся таким же образом, как другие элементы веб-страницы. Допустим, у вас есть боковая панель, примыкающая к правому краю веб-страницы. Содержимое страницы, расположенное под ней, как ему и положено, приподнимается выше и обтекает панель. Однако если этому контенту страницы назначены настройки фона или границ, они отображаются, фактически *проступая* под боковой панелью (рис. 7.16, *слева*). По сути, браузер окружает плавающий элемент только текстом, но не границами или фоном. Как это ни удивительно, но именно так работает механизм обтекания. Возможно, вы не будете следовать этим правилам; а захотите, чтобы границы, фоновый цвет или рисунок не отображались при достижении плавающего элемента (см. рис. 7.16, *справа*). С помощью каскадных таблиц стилей вы можете сделать и это.

Первый метод заключается в добавлении еще одного свойства в стиль, устанавливающий параметры фона и границ и оказывающий воздействие на плавающий элемент. Нужно добавить в этот стиль следующий код: `overflow: hidden;`. Свойство `overflow` (описано в предыдущем разделе) отменяет отображение фона или границ под плавающими элементами.

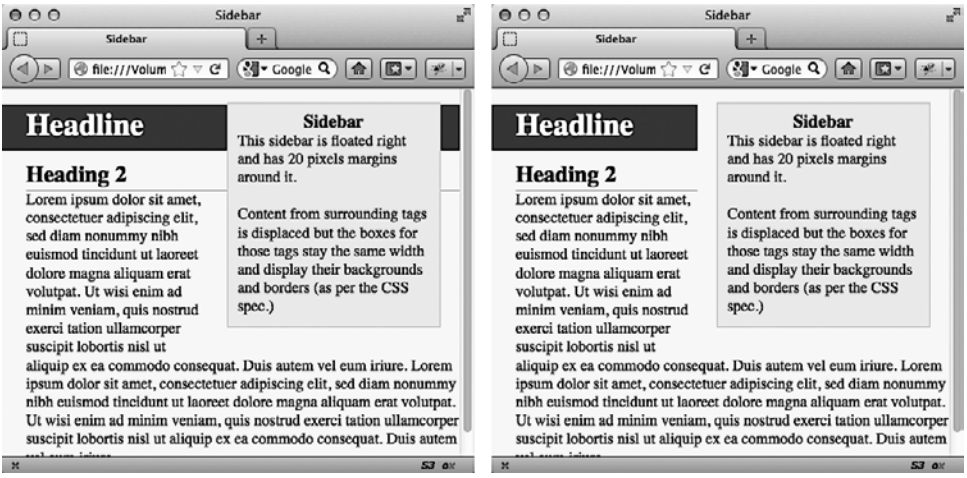

Рис. 7.16. Добавление свойства overflow: hidden; в стиль элемента h1 препятствует тому, чтобы атрибуты заголовка повлияли на отображение фона и границ под плавающим элементом (боковая панель)

Другой подход состоит в добавлении границ вокруг плавающего элемента. Если вы создаете достаточно толстую границу цвета фона страницы, то не будете даже догадываться о существовании границ, находящихся под плавающим элементом, так как они будут скрыты.

Запрет обтекания

Иногда требуется отобразить содержимое элемента таким образом, чтобы на него не влияла способность обтекания плавающего элемента. Например, веб-страница может содержать примечание с текстом авторских прав, которое в любом случае должно отображаться в самой нижней части окна браузера. Если у вас имеется высокая боковая панель, примыкающая к левому краю веб-страницы, то примечание с авторскими правами может подняться вверх и отобразиться с обтеканием вокруг плавающего элемента. Таким образом, вместо того чтобы быть внизу страницы, текст появится гораздо выше, на одном уровне с боковой панелью. Вам же нужно, чтобы для примечания с авторскими правами было отменено обтекание, установленное свойством плавающей боковой панели.

Сложности другого рода возникают при наличии нескольких плавающих элементов, расположенных рядом. Когда они имеют небольшую ширину, то поднимаются вверх, располагаясь на одном уровне. В противном случае элементы могут образовать некрасивую «пустоту» (рис. 7.17, *вверху*). В данном случае плавающие элементы *не должны* отображаться рядом друг с другом. Для решения этой проблемы в языке CSS предусмотрено свойство `clear`.

Свойство `clear` сообщает браузеру о том, что форматируемый элемент *не должен обтекать* плавающий элемент. Оно приводит к тому, что элемент смещается вниз

по тексту веб-страницы, располагаясь ниже плавающего элемента. Кроме того, вы можете запретить обтекание с любой стороны элемента (слева или справа) или полностью удалить все параметры обтекания.

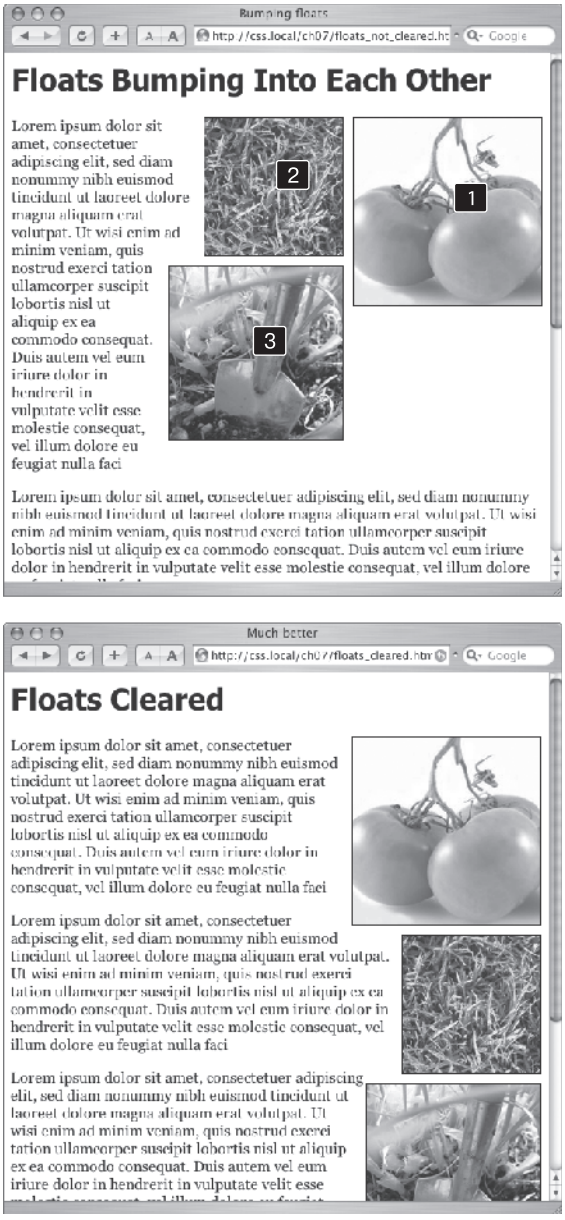

Рис. 7.17. Свойство clear, примененное к изображению *(2)*, предотвращает наложение на другое изображение *(1)*. Применение свойства clear: right; к фотографии *(3)* предотвращает ее размещение под фото *2*

Свойство `clear` допускает следующие значения:

- `left` — форматируемый элемент смещается вниз относительно плавающего с установленным обтеканием слева, но обтекание справа остается в силе;
- `right` — форматируемый элемент смещается вниз относительно плавающего с установленным обтеканием справа, обтекание слева остается в силе;
- `both` — вызывает перемещение форматируемого элемента вниз относительно плавающего с установленным обтеканием и слева и справа;
- `none` — полностью отменяет запрет обтекания — сообщает браузеру, что форматируемый элемент должен вести себя так, как предусмотрено, то есть он может обтекать плавающие элементы как с левой, так и с правой стороны.

В нашем примере текст примечания с авторским правом должен всегда отображаться внизу веб-страницы и не должен обтекать другое содержимое. Ниже приведен код класса, который делает это:

```
.copyright {
  clear: both;
}
```

На рис. 7.17 показано, как свойство `clear` может препятствовать беспорядочному отображению плавающих элементов различной высоты. Для всех трех представленных на рисунке фотографий установлено обтекание по правому краю. В верхней части рисунка фото помидоров *(1)* — первое изображение на веб-странице, оно расположено в верхнем правом углу. На отображение второй фотографии *(2)* оказывает влияние параметр обтекания первого изображения, и оно обтекает слева первое фото. Последнее изображение *(3)* слишком широкое, чтобы отобразиться рядом (на одном уровне) со вторым *(2)*, но все-таки оно пытается расположиться слева по отношению как к первому *(1)*, так и ко второму *(2)* изображениям. Однако ширина рисунка не позволяет этого сделать.

Применение здесь свойства `clear: right;` препятствует размещению фотографий на одном уровне (см. рис. 7.17, *внизу*). Запрет обтекания для второй фотографии не допускает ее отображения рядом с первой, а запрет обтекания для третьей предотвращает ее отображение на одном уровне со второй.

ПРИМЕЧАНИЕ

Применение параметров левого и правого обтеканий, а также запрет обтекания кажется достаточно сложным механизмом. Вообще, так оно и есть. В этом разделе приведены основные понятия. Мы снова вернемся к этой теме в главе 13, там вы познакомитесь с использованием обтекания в более сложных реализациях.

Практикум: поля, фон и границы

В этом практикуме мы исследуем элементы блочной модели CSS, потренируемся в настройке пространства вокруг объектов веб-страниц, применим к элементам красочные границы, научимся управлять размерами и обтеканием элементов веб-страниц.

Чтобы начать обучение, вы должны иметь в распоряжении файлы с учебным материалом. Для этого нужно загрузить файлы для выполнения заданий практи-

кума, расположенные по адресу 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 07.

Управление фоном и полями страницы

Начнем с простого HTML-файла, содержащего внутреннюю таблицу стилей со сбросом стандартных настроек стилей CSS. Пока на странице нет ничего интересного (рис. 7.18).

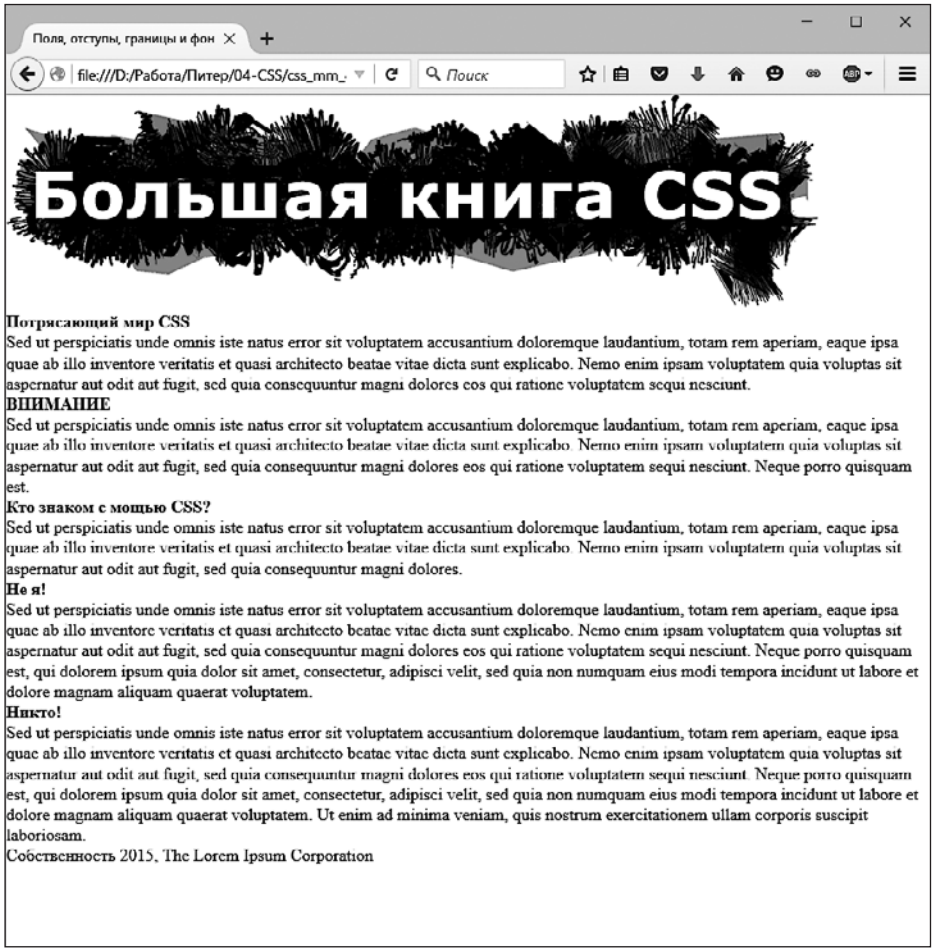

Рис. 7.18. Страница представляет собой простой HTML-документ и содержит единственный стиль, сбрасывающий множество встроенных настроек форматирования браузера. Она будет выглядеть намного лучше после изменения с использованием блочной модели CSS

СОВЕТ

Увидеть конечный результат практикума можно на рис. 7.21.

1. Откройте в редакторе HTML-кода страницу `main.html` из папки 07.

   Эта таблица стилей уже связана с файлом `index.html`, поэтому стили, добавленные в него, применятся и к странице. В таблице используется набор стилей, который рассматривался в главе 5, в подразделе «С чистого листа» раздела «Управление каскадностью». Он удаляет все поля, отступы, установленные значения размеров шрифта, начертания шрифта из наиболее значимых блочных элементов и устраняет множество кросс-браузерных проблем с отображением элементов, с которыми вы могли столкнуться из-за этих свойств.

   Наверное, наиболее важные свойства в нем — `margin` и `padding`, установленные в первом стиле. Существует достаточное количество кросс-браузерных странностей, связанных с этими двумя свойствами, поэтому вы должны всегда обнулять их и начинать с чистого листа. Другой распространенной альтернативой является таблица стилей, которая устраняет различия в отображении страницы в различных браузерах, но сохраняет основные поля (tinyurl.com/cop5s89).

2. В нижней части файла `main.css` щелкните кнопкой мыши сразу после комментария `/* end reset browser styles */` и добавьте селектор тега:

```
html {
  background-color: rgb(253,248,171);
}
```

   Этот стиль устанавливает светло-желтый фон для страницы. Если вы хотите задать цвет для фона веб-страницы, свойство `background-color` можно добавлять либо к элементу `html`, либо к `body`. Далее мы добавим поля, границы и другие свойства элемента `body`.

ПРИМЕЧАНИЕ

Возможно, вы привыкли вместо задания цвета по модели RGB использовать шестнадцатеричные значения (например, #FDF8AB). Чтобы осуществлять преобразования между этими двумя моделями, можно воспользоваться онлайн-конвертером по адресу colorhexa.com. Но предпочтительнее применять модель RGB, поскольку ее вариация RGBA, имеющая дополнительный параметр прозрачности, достаточно полезна, чтобы остановиться на одной системе указания цвета (RBG), а не смешивать две (RGB и шестнадцатеричные значения).

3. Добавьте еще один стиль в таблицу:

```
body {
  background-color: rgb(255,255,255);
  border: 3px solid rgb(75,75,75);
}
```

   Этот стиль добавляет к элементу `body` белый цвет фона и темно-серую сплошную границу шириной 3 пиксела. Поскольку `body` расположен внутри элемента `html`, браузер считает, что он находится «поверх» элемента `html`, и белый цвет покроет фон желтого цвета, установленный на предыдущем шаге. Далее мы зададим ширину элемента `body` и настроим его отступы и поля.

СОВЕТ

Обычно, если вы добавляете свойство background-color к элементу body, цвет заполняет все окно браузера. Однако если вы также добавите цвет фона для элемента html, фон body будет заполнять

только область контента. Чтобы увидеть это в действии, просмотрите веб-страницу после шага 4, затем удалите стиль для элемента html и просмотрите страницу снова. Странная, но полезная мелочь каскадных таблиц стилей.

4. Измените стиль элемента body, который вы только что создали, добавив пять новых свойств (изменения выделены полужирным шрифтом):

```
body {
  background-color: rgb(255,255,255);
  border: 3px solid rgb(75,75,75);
  max-width: 760px;
  margin-top: 20px;
  margin-left: auto;
  margin-right: auto;
  padding: 15px;
}
```

Свойство max-width ограничивает тело (элемент body) страницы 760 пикселами по ширине: если окно браузера посетителя окажется шире, чем 760 пикселов, он увидит фоновый цвет элемента html и область шириной 760 пикселов с белым фоном элемента body. Если же окно браузера станет у́же указанного размера, блок текста также сузится, заполняя окно, что упрощает просмотр страницы на небольших экранах планшетов и смартфонов.

Свойство margin-top добавляет 20 пикселов пространства от верхнего края окна браузера, смещая содержимое элемента body немного вниз, а левое и правое поля центрируют его, размещая в середине окна браузера. Значение auto — это еще один способ сообщить браузеру: «Разбирайся в этом сам», и, поскольку оно применяется как к левому, так и к правому полям, браузер создает одинаковые промежутки слева и справа.

ПРИМЕЧАНИЕ

Вы можете также использовать сокращенную запись свойства margin, устанавливая настройки полей, если наберете одну строку кода:

margin: 20px auto 0 auto;

Наконец, для того, чтобы предотвратить прикосновение содержимого элемента body к линии границы, отступ шириной 15 пикселов добавлен к внутренней части содержимого с помощью свойства padding. Другими словами, для изображения и текста был задан отступ 15 пикселов от всех четырех краев. Затем мы добавим свечение вокруг блока, воспользовавшись свойством box-shadow.

5. Отредактируйте только что созданный стиль элемента body, добавив к нему еще одно свойство после строки border, но перед max-width (изменения выделены полужирным шрифтом):

```
body {
  background-color: rgb(255,255,255);
  border: 3px solid rgb(75,75,75);
  box-shadow: 0 0 15px 5px rgba(44,82,100,.75);
  max-width: 760px;
```

```
  margin-top: 20px;
  margin-left: auto;
  margin-right: auto;
  padding: 15px;
}
```

Этот стиль добавляет к блоку свечение путем создания 15-пиксельной тени, помещенной непосредственно за блоком (то, что параметры тени начинаются с `0 0`, означает, что тень не имеет смещения вправо-влево или вверх-вниз и представляет собой фон). Значение `5px` определяет расширение тени, выталкивая ее на 5 пикселов дальше всех четырех углов. И наконец, значение `rgba` устанавливает темно-синий цвет с уровнем непрозрачности 75 % (то есть сквозь него можно видеть желтый фон).

В вашей таблице стилей уже много всего, поэтому пришло время проверить, как выглядит страница.

6. Сохраните файл и просмотрите веб-страницу в браузере.

   Вы должны увидеть белый блок с изображением, фрагмент текста и серый контур с синеватым свечением, плавающие на желтом фоне (рис. 7.19). В данном случае настройка ширины элемента `body`, а также добавление кода `margin-left: auto;` и `margin-right: auto;` помещает его прямо в центре окна браузера. Для выравнивания объекта по вертикали (сохранения одинаковых расстояний сверху и снизу от объекта до краев страницы), необходимо использовать нечто большее, чем свойства `margin`. Чтобы узнать способы выравнивания по вертикали и горизонтали, посетите страницу tinyurl.com/qhad4mq.

   Теперь следует уделить внимание тексту, чем мы и займемся далее.

Настройка интервалов между элементами

Поскольку стили, сбрасывающие стандартные настройки CSS, «оголили» текст на странице, вам необходимо создать новые стили, которые преобразили бы заголовки и абзацы. Начнем с элемента `h1` в верхней части страницы.

1. Вернитесь к файлу `main.html` в редакторе HTML-кода. Установите курсор после закрывающей скобки селектора `body`, нажмите клавишу Enter и добавьте следующий стиль:

```
h1 {
  font-size: 2.75em;
  font-family: Georgia, "Times New Roman", Times, serif;
  font-weight: normal;
  text-align:center;
  letter-spacing: 1px;
  color: rgb(133,161,16);
  text-transform: uppercase;
}
```

   Он использует множество свойств для форматирования текста, которые мы обсуждали в предыдущей главе. Верхний заголовок имеет высоту 2,75 em (44 пиксела в большинстве браузеров) и набран прописными буквами. Для него уста-

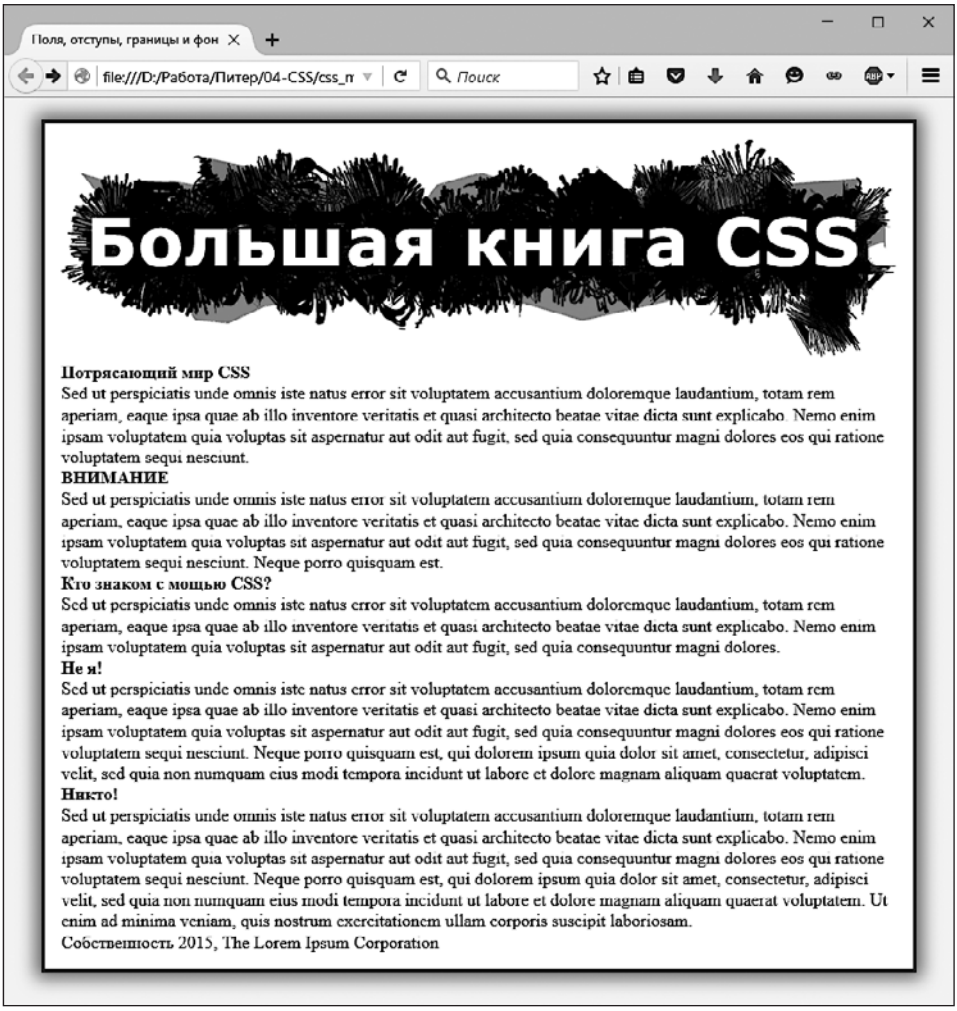

Рис. 7.19. Устанавливая значение auto для левого и правого полей любого элемента, имеющего заданную ширину, вы центрируете этот элемент

новлен шрифт Georgia зеленого цвета, есть небольшой промежуток между буквами. Свойство `text-align` обеспечивает выключку текста по центру блока. Очень интересным получается добавление фонового цвета для выделения заголовка.

СОВЕТ

Сохраняйте файл и просматривайте страницу в браузере после каждого шага из этого примера. Таким образом вы поймете, как приведенные здесь свойства CSS воздействуют на элементы, которые они форматируют.

2. Добавьте новое свойство к селектору тега `h1`, чтобы он выглядел следующим образом (изменения выделены полужирным шрифтом):

```
h1 {
  font-size: 2.75em;
  font-family: Georgia, "Times New Roman", Times, serif;
  font-weight: normal;
  text-align:center;
  letter-spacing: 1px;
  color: rgb(133,161,16);
  text-transform: uppercase;
  background-color: rgb(226,235,180);
}
```

Если вы просмотрите страницу сейчас, то увидите, что заголовок обрел светло-зеленый фон. Когда фон применяется к блочным элементам, таким как заголовок, он заполняет все доступное горизонтальное пространство (другими словами, цвет не только появляется за текстом «Потрясающий мир CSS», но и простирается вплоть до правого края окна).

Текст заголовка немного стесненный — буква «П», с которой он начинается, касается края фона. С помощью небольших отступов вы можете исправить это.

3. Добавьте еще одно свойство к селектору тега `h1`, чтобы он выглядел следующим образом (изменения выделены полужирным шрифтом):

```
h1 {
  font-size: 2.75em;
  font-family: Georgia, "Times New Roman", Times, serif;
  font-weight: normal;
  text-align:center;
  letter-spacing: 1px;
  color: rgb(133,161,16);
  text-transform: uppercase;
  background-color: rgb(226,235,180);
  padding: 5px 15px 2px 15px;
}
```

Сокращенное свойство `padding` предоставляет быстрый способ добавить отступы вокруг всех четырех сторон контента — в данном случае отступ шириной 5 пикселов добавляется над текстом, 15 пикселов — справа, 2 пиксела — внизу и 15 пикселов — слева.

Существует еще одна проблема с заголовком: из-за отступов, которые добавлены к элементу `body` (см. шаг 4 предыдущего задания), заголовок (включая фоновый цвет) отодвинут на 15 пикселов от левого и правого краев серой границы, окружающей содержимое. Заголовок станет выглядеть лучше, если его фоновый цвет будет касаться этого контура. Это не проблема: на помощь приходят отрицательные значения полей.

4. Добавьте последнее свойство к селектору тега `h1`, чтобы он выглядел следующим образом (изменения выделены полужирным шрифтом):

```
h1 {
  font-size: 2.75em;
  font-family: Georgia, "Times New Roman", Times, serif;
```

```
  font-weight: normal;
  text-align:center;
  letter-spacing: 1px;
  color: rgb(133,161,16);
  text-transform: uppercase;
  background-color: rgb(226,235,180);
  padding: 5px 15px 2px 15px;
  margin: 0 -15px 20px -15px;
}
```

Здесь сокращенное свойство `margin` устанавливает следующие размеры полей: 0 пикселов для верхнего, –15 пикселов для правого, 20 пикселов для нижнего и –15 пикселов для левого. Нижнее поле добавляет немного пространства между заголовком и абзацем, который идет за ним. Следующий прием заключается в использовании отрицательных значений для левого и правого полей. У нас есть возможность назначить отрицательные значения полей для какого-либо элемента. Это свойство «тянет» элемент по направлению поля — в данном случае, заголовок продлевается на 15 пикселов влево и 15 пикселов вправо, расширяясь и вытягиваясь поверх отступов элемента `body`.

5. Теперь вы немного отформатируете элемент `h2`. Добавьте следующий стиль после стиля `h1`:

```
h2 {
  font-size: 1.5em;
  font-family: "Arial Narrow", Arial, Helvetica, sans-serif;
  color: rgb(249,107,24);
  border-top: 2px dotted rgb(141,165,22);
  border-bottom: 2px dotted rgb(141,165,22);
  padding-top: 5px;
  padding-bottom: 5px;
  margin: 15px 0 5px 0;
}
```

Этот стиль добавляет базовое форматирование текста и пунктирные границы сверху и снизу заголовка. Чтобы добавить немного пространства между текстом заголовка и строками, используются небольшие отступы сверху и снизу. Наконец, свойство `margin` добавляет поля шириной 15 пикселов над заголовком и 5 пикселов под ним.

6. Сохраните файл и просмотрите страницу в браузере.

Заголовки выглядят прекрасно (рис. 7.20). Далее вы создадите боковую панель в правой части страницы.

Создание боковой панели

Боковые панели — элементы, применяемые в большинстве печатных журналов и газет. Они отделяют небольшие блоки информации, например перечни статей, и акцентируют на них внимание. Но для того, чтобы боковые панели были достаточно эффективными и полезными, они не должны прерывать потока основного

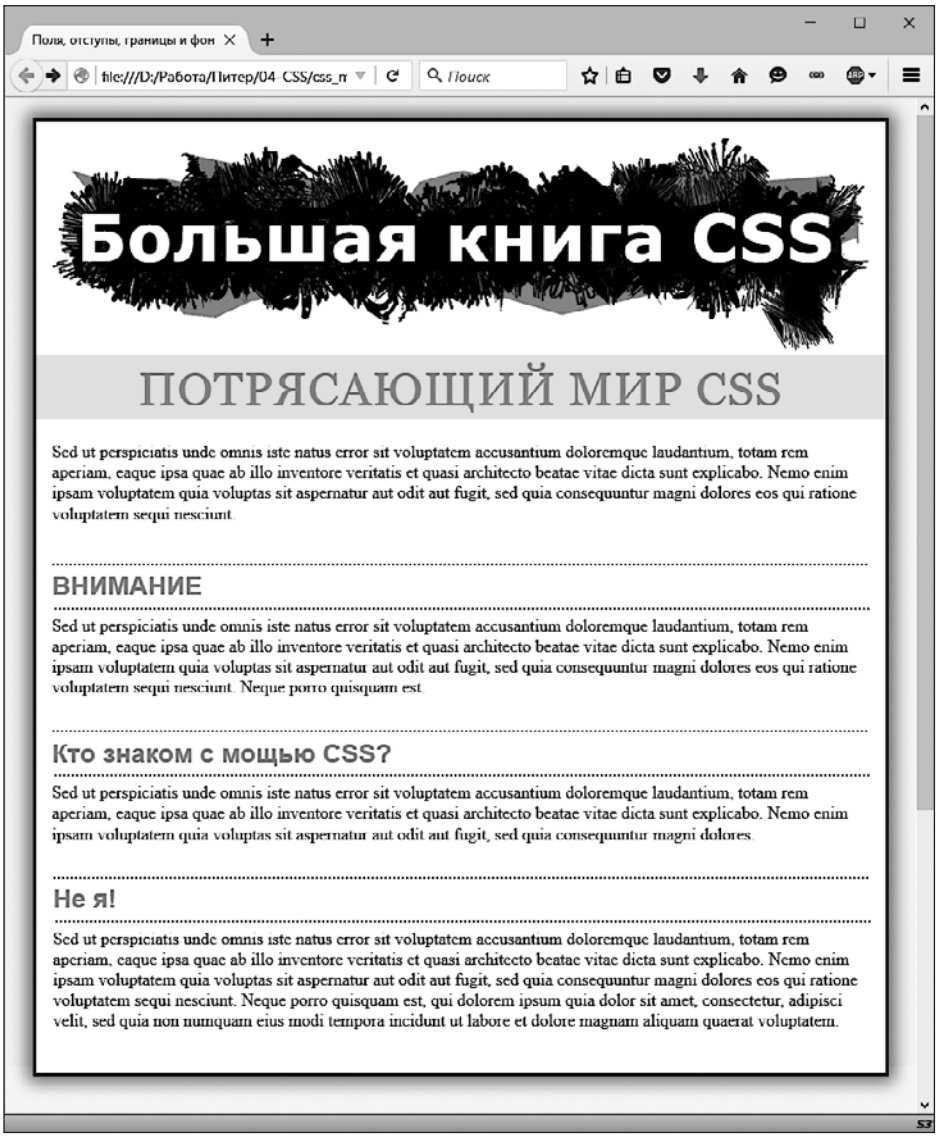

Рис. 7.20. С помощью нескольких стилей вы можете изменить фоновый цвет веб-страницы, добавить поля, регулировать интервалы между заголовками и абзацами

контента. Даже название «боковая панель» говорит о том, что этот блок должен быть расположен обособленно и примыкать к краю веб-страницы, что можно легко сделать средствами каскадных таблиц стилей.

1. Вернитесь к файлу `index.html` в редакторе HTML-кода.

   Сначала нужно изолировать область веб-страницы, составляющую боковую панель. Для этого прекрасно подходит элемент `div`. С его помощью можно заключить любой объем HTML-кода в отдельный блок.

2. Прокрутите страницу вниз и щелкните кнопкой мыши перед первым элементом h2 (с заголовком Внимание). Введите код и нажмите клавишу Enter.

   Этот HTML-код отмечает начало блока боковой панели и применяет к нему класс. Мы создадим класс .sidebar позже, а сначала нужно определить завершение блока боковой панели, закрыв контейнер тегом .

3. Перейдите к закрывающему тегу </p> абзаца, который следует сразу за элементом h2 (это тот тег </p>, который расположен перед строкой <h2>Кто знаком с мощью CSS?</h2>). Нажмите клавишу Enter и введите тег .

   Мы только что заключили заголовок и маркированный список в элемент div. Теперь создадим для него стиль.

4. Вернитесь к файлу main.css и добавьте после созданного ранее селектора h2 следующий код:

```
.sidebar {
  width: 30%;
  float: right;
  margin: 10px;
}
```

   Этот стиль устанавливает ширину области контента (в которой отображается текст) равной 30%. В этом случае ширина боковой панели составляет 30 % от ширины ее контейнера. Контейнером является элемент body, и его ширина не превысит 760 пикселов. Свойство float перемещает боковую панель в правую часть блока, а свойство margin добавляет 10 пикселов пространства вокруг панели.

   Если вы просмотрите веб-страницу в браузере, то увидите, что форма и положение блока боковой панели определены, но есть одна проблема: границы элементов h2 отображаются *под* самим блоком. Несмотря на то что плавающая боковая панель смещает текст заголовков, границы остаются на том же месте. Один из способов решить эту проблему — добавить цвет фона для боковой панели, чтобы не видеть границ h2 (но есть и другой способ, который вы будете использовать на шаге 8).

5. Добавьте еще одно свойство к стилю .sidebar, чтобы он выглядел следующим образом (изменения выделены полужирным шрифтом):

```
.sidebar {
  width: 30%;
  float: right;
  margin: 10px;
  background-color: rgb(250,235,199);
  padding: 10px 20px;
}
```

   Это свойство добавляет светло-оранжевый цвет к боковой панели и смещает текст от границ боковой панели, чтобы он не касался границ, которые вы собираетесь добавить.

6. Добавьте еще два свойства к стилю .sidebar, чтобы он выглядел следующим образом (изменения выделены полужирным шрифтом):

```
.sidebar {
  width: 30%;
  float: right;
  margin: 10px;
  background-color: rgb(250,235,199);
  padding: 10px 20px;
  border: 1px dotted rgb(252,101,18);
  border-top: 20px solid rgb(252,101,18);
}
```

Это пример удобной методики, описанной ранее. Если вы хотите, чтобы бóльшая часть границ вокруг элемента была одинаковой, можно сначала определить границу для всех четырех краев — в данном случае пунктирную оранжевую линию толщиной 1 пиксел вокруг всей боковой панели. Затем можно применить новые свойства для отдельных границ, которые вы хотите изменить, — в данном примере верхняя граница будет сплошной и будет иметь высоту 20 пикселов. Такой способ позволяет использовать всего две строки кода, а не четыре (border-top, border-bottom, border-left и border-right).

Затем мы добавим скругленные углы и тень, чтобы выделить эту боковую панель.

7. И наконец, добавьте еще два свойства к стилю .sidebar, придав ему следующий вид (изменения выделены полужирным шрифтом):

```
.sidebar {
  width: 30%;
  float: right;
  margin: 10px;
  background-color: rgb(250,235,199);
  padding: 10px 20px;
  border: 1px dotted rgb(252,101,18);
  border-top: 20px solid rgb(252,101,18);
  border-radius: 10px;
  box-shadow: 5px 5px 10px rgba(0,0,0,.5);
}
```

Свойство border-radius позволяет создать скругленные углы. В данном случае значение 10px предоставляет заметное скругление. Свойство box-shadow добавляет тень, отбрасываемую вниз и вправо от блока, придавая ему вид парящего над страницей. Теперь вы близки к завершению работы.

Заголовок внутри боковой панели выглядит не совсем так, как должен. К нему применяются те же свойства, что и к другим элементам h2 (из-за стиля тега h2, который вы создали в шаге 4). Границы отвлекают внимание, а верхнее поле излишне смещает заголовок вниз от верхней части боковой панели. Для решения проблемы вы можете использовать селектор потомков, чтобы переопределить эти свойства.

8. После стиля .sidebar в файле main.css добавьте селектор потомков следующим образом:

```
.sidebar h2 {
  border: none;
  margin-top: 0;
  padding: 0;
}
```

Из-за селектора потомков .sidebar этот стиль является доминирующим, то есть имеет большую *специфичность* по сравнению с простым стилем h2. Он стирает границу, полученную от оригинального стиля элемента h2, вместе с верхним полем и всеми отступами. Тем не менее, поскольку в этом стиле не определены размер, цвет и начертание шрифта, эти свойства по-прежнему передаются от стиля h2 — каскадность в действии!

Страница стала хорошо выглядеть, но границы элементов h2 все еще появляются под боковой панелью. На это не очень приятно смотреть, но все можно легко исправить.

9. Найдите стиль h2 и добавьте свойство overflow:

```
h2 {
  font-size: 1.5em;
  font-family: "Arial Narrow", Arial, Helvetica, sans-serif;
  color: rgb(249,107,24);
  border-top: 2px dotted rgb(141,165,22);
  border-bottom: 2px dotted rgb(141,165,22);
  padding-top: 5px;
  padding-bottom: 5px;
  margin: 15px 0 5px 0;
  overflow: hidden;
}
```

Присвоив свойству overflow значение hidden, вы скроете границы, которые проходят за пределами текста заголовка и под плавающим элементом.

10. Сохраните файл и просмотрите веб-страницу в браузере.

Таблица должна иметь вид, представленный на рис. 7.21.

Дополнительное задание

Чтобы закрепить полученные знания и навыки, попробуйте выполнить следующее практическое задание самостоятельно. Создайте для элемента p стиль, который бы смог приукрасить абзац: попробуйте добавить поля, указать цвет шрифта и т. д. Затем создайте класс для форматирования примечания с информацией об авторском праве, которое должно отображаться в нижней части страницы index.html (например, с именем .copyright). Добавьте в этот стиль верхнюю границу (над текстом примечания), измените цвет текста, уменьшите размер шрифта и измените регистр букв на прописные (используйте для этого свойство text-transform, описанное в разделе «Форматирование символов и слов» главы 6). После создания стиля добавьте соответствующий атрибут класса к элементу p в HTML-коде.

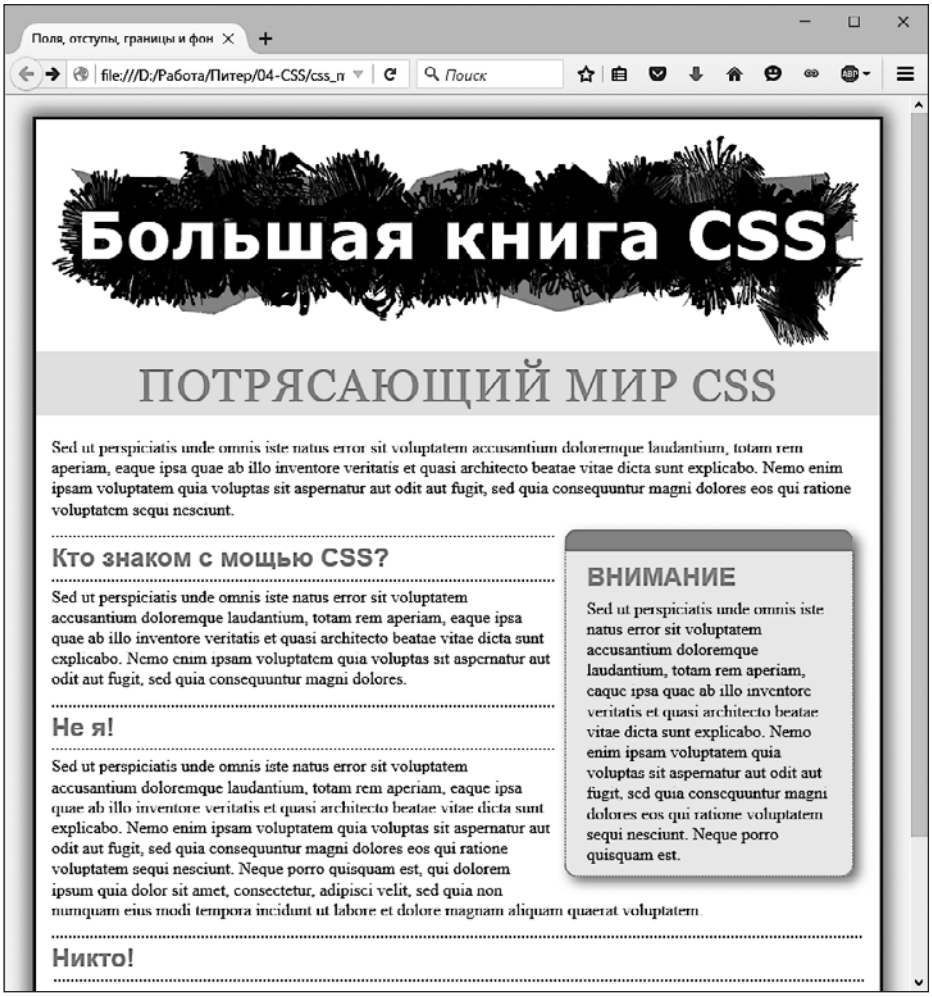

Рис. 7.21. Небольшой набор каскадных стилей придает непривлекательному HTML-коду элегантный дизайн. Обратите внимание, как плавающая боковая панель привлекает внимание и смещает в сторону основной текст