13 Макеты на основе обтекаемых элементов

Верстка на основе обтекаемых элементов использует преимущества свойства `float` для позиционирования элементов рядом друг с другом, создавая на веб-странице колонки. Как было описано в разделе «Управление обтеканием контента с помощью плавающих элементов» главы 7, вы можете использовать это свойство для создания эффекта обтекания, скажем, фотографии, но, когда вы применяете его к элементу `div`, оно становится мощным инструментом верстки страницы. Свойство `float` перемещает элемент в указанную сторону страницы (или другого блока с содержимым). Любой HTML-объект, который указан ниже обтекаемого элемента, изменяет свое положение на странице и обтекает его.

Свойство `float` принимает одно из трех значений: `left`, `right` или `none`. Чтобы выровнять изображение по правому краю страницы, вы можете создать класс и применить с его помощью стиль к элементу `img`:

```
.floatRight { float: right; }
```

То же самое свойство, примененное к элементу `div` с содержимым, позволяет создать боковую панель:

```
.sidebar {
  float: left;
  width: 25%;
}
```

На рис. 13.1 показаны эти два стиля в действии.

ПРИМЕЧАНИЕ

Значение none отменяет любое выравнивание и определяет элемент как обычный (необтекаемый). Это полезно только для отмены выравнивания, которое уже применено к элементу. Допустим, у вас есть элемент, к которому применен специфический класс, такой как .sidebar, и этот элемент смещен вправо. На одной из страниц вы, возможно, захотите, чтобы элемент с этим классом не смещался, а был определен в общем потоке страницы, допустим, в качестве примечания. Создавая более специфичный селектор (см. раздел «Управление каскадностью» главы 5) с кодом float: none, вы можете предотвратить смещение этого элемента.

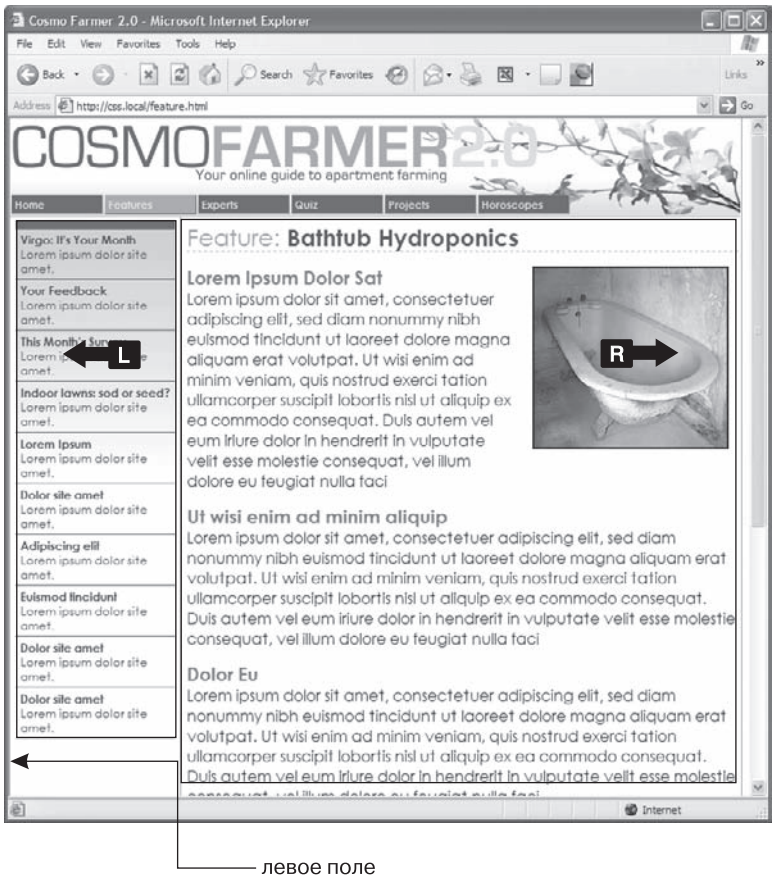

Рис. 13.1. Используйте свойство float для компоновки многоколоночной веб-страницы. Блок новостей выровнен по левому краю. У него есть фиксированная ширина, однако у основного контента ее нет, что делает этот дизайн резиновым. Основной раздел страницы расширяется, заполняя окно браузера. Вверху справа фотография с ванной выровнена по правому краю

Простой дизайн с двумя колонками наподобие показанного на рис. 13.1 требует выполнения всего нескольких действий.

1. Оберните каждую колонку элементом `div` с атрибутом класса или идентификатора.

   На рис. 13.1 заголовки новостей, перечисленные в левой части, обернуты в один контейнер (`<div class = "news">`), а основной контент страницы — в другой (`<div class = "main">`).

2. Выровняйте элемент `div` с боковой панелью по правому или левому краю.

   Когда вы работаете с обтекаемыми элементами, важен порядок исходного кода (порядок, в котором вы добавляете HTML-код в файле). HTML-код обтекаемого элемента должен быть указан перед HTML-кодом элемента, который оборачивает его.

На рис. 13.2 показаны три варианта двухколоночного дизайна. Схемы в левой части демонстрируют порядок HTML-кода страницы: элемент `div` баннера, за которым следует `div` боковой панели, и, наконец, элемент `div` основного контента. Справа вы видите фактический дизайн страницы. Боковая панель указана *ранее* основного контента в HTML-коде, так что она может переместиться или влево (*вверху* и *внизу*), или вправо *(посередине)*.

3. Установите ширину для обтекаемой боковой панели.

   Всегда ограничивайте ширину обтекаемых элементов. Таким образом вы позволите браузеру создать пространство для другого контента в результате обтекания.

   В качестве значения ширины может быть задан фиксированный размер, такой как 170 пикселов или 10 em. Вы также можете использовать проценты для резинового дизайна, основанного на ширине окна браузера (см. подраздел «Ключевые слова, проценты и единица измерения em» раздела «Изменение размера шрифта» главы 6, чтобы узнать обо всех «за» и «против» различных единиц измерения). Если боковой панели задана ширина 20 %, а окно браузера — 700 пикселов в ширину, то ширина боковой панели составит 140 пикселов. Если же посетитель сайта изменит размер окна до 1000 пикселов, то боковая панель увеличится до 200 пикселов. Боковые панели с фиксированной шириной легче проектировать, так как не нужно просматривать различные значения ширины, до которых боковая панель могла бы растянуться.

   Однако проценты позволяют вам поддерживать одинаковые пропорции между двумя колонками, что может быть визуально привлекательнее. Кроме того, проценты делают ваши конструкции гибкими, поскольку общие пропорции страницы могут подстраиваться под размер экрана, что играет важную роль при создании адаптивных веб-конструкций, речь о которых пойдет в следующей главе.

ПРИМЕЧАНИЕ

Если дизайн всей страницы указан с фиксированной шириной, значения ширины для боковой панели в процентах основаны на элементе с фиксированной шириной. Ширина не зависит от размера окна и остается постоянной.

4. Добавьте поле слева к основному контенту.

   Если боковая панель короче другого контента на странице, то текст из основной колонки обтекает панель снизу. Добавление поля слева, больше ширины боковой панели или равного ей, выравнивает основной контент страницы, создавая иллюзию второй колонки:

```
.main { margin-left: 27%; }
```

   Рекомендуется сделать поле слева чуть больше по ширине, чем боковая панель: так вы добавите промежуток между двумя элементами. Если в качестве значения ширины боковой панели вы используете проценты, то задавайте немного большее значение для поля слева.

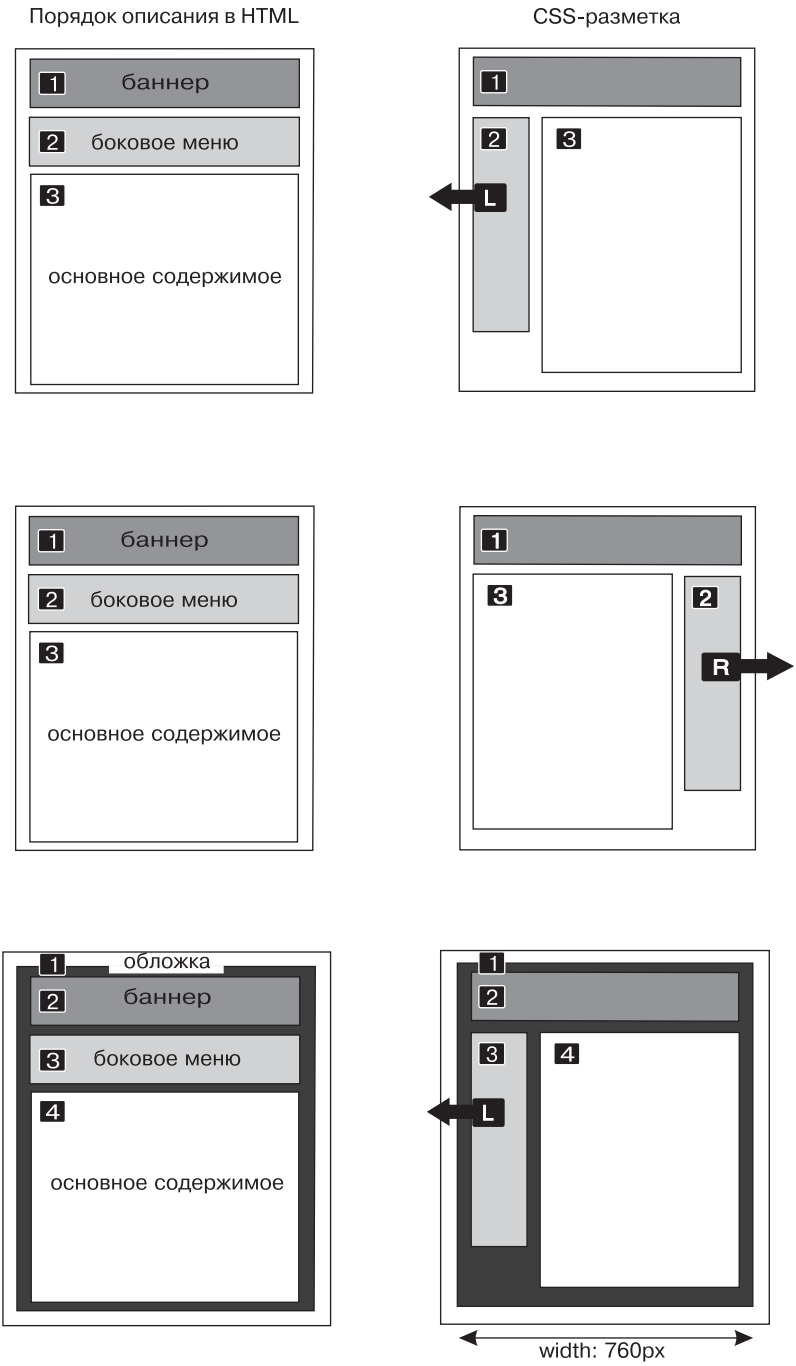

Рис. 13.2. Создание двухколоночного макета лишь вопрос выравнивания элемента div по левому краю *(вверху)*. Чтобы переместить боковую панель в правую часть страницы *(посередине)*, измените стиль боковой панели на float: right

Избегайте ограничения ширины контейнера `div` с основным контентом — браузеры расширяют его, чтобы он занимал доступное пространство. Даже если вы хотите создать фиксированный дизайн, вам не нужно ограничивать ширину основного контента, как вы узнаете в следующем разделе.

Использование обтекаемых элементов при верстке

Теперь, когда вы изучили простую двухколоночную резиновую верстку, можете применять ее бесчисленным множеством способов. Преобразовать ее в верстку с фиксированной шириной — просто. Надо обернуть все элементы внутри тела страницы *другим* элементом `div` (например, `<div class = "wrap">`), а затем создать стиль для этого нового элемента-контейнера, для которого определена ширина, скажем, 960 пикселов (см. рис. 13.2, *внизу*). Подобное ограничение ширины удерживает контент внутри контейнера.

ПРИМЕЧАНИЕ

Существует также вариант создания страницы с фиксированной шириной без применения дополнительного элемента div, оборачивающего все элементы: ограничьте ширину элемента body. Вы уже видели пример использования этого метода в практикуме главы 2.

Развернуть контент страницы на три колонки также не представляет сложности (рис. 13.3). Сначала добавьте другой элемент `div` между этими двумя колонками и выровняйте его по правому краю. Затем добавьте поле справа к средней колонке так, чтобы, если текст в ней окажется длиннее новой правой боковой панели, он не обтекал боковую панель.

В остальной части этого раздела рассматриваются многочисленные методы CSS-верстки с применением обтекаемых элементов.

Выравнивание всех колонок

В полной мере можно выровнять *каждую* колонку, а не только левую и правую боковые панели. Вы можете выровнять первую боковую панель и среднюю колонку по левому краю, а правую боковую панель — по правому краю, как показано на рис. 13.2, *вверху*. Этот подход позволяет размещать более трех колонок на странице. Вы можете выравнивать четыре колонки и более, пока есть пространство для всех обтекаемых элементов, чтобы они находились рядом друг с другом.

Если вы выравниваете все колонки в макете, то должны обратить пристальное внимание на ширину каждой из них. Если совокупная ширина всех колонок меньше доступного пространства, например, если окно браузера меньше или колонки обернуты другим элементом `div` с определенной шириной, то последняя колонка опускается вниз (как решать проблему выпадения обтекаемых элементов, вы можете прочитать в разделе «Решение проблем с обтекаемыми элементами» этой главы).

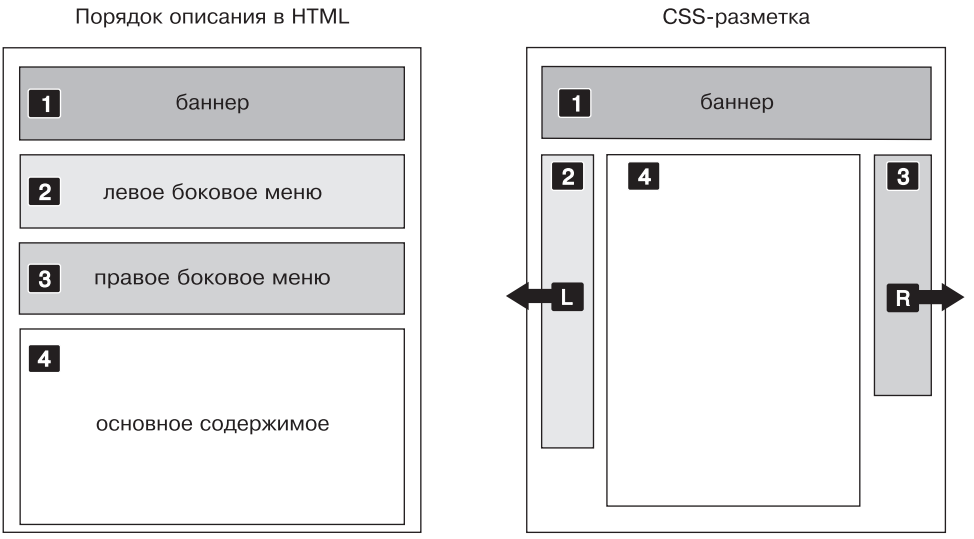

Рис. 13.3. Для трехколоночных макетов используется та же концепция, что и для двухколоночных. При трехколоночном дизайне вы выравниваете левую и правую панель и добавляете поля слева и справа к центральной колонке. Схема в левой части демонстрирует порядок следования HTML-кода, а справа вы можете увидеть, как выглядит веб-страница

Вдобавок выравнивая не только боковые панели, вы сможете изменить порядок разделов `div` в HTML-коде. Используйте, к примеру, левую схему на рис. 13.3, которая демонстрирует порядок следования элементов `div` для страницы. По принципу своей работы обтекаемые элементы должны появляться перед любым контентом, который окружает их, так что в этом примере область основного контента должна следовать *после* боковой панели.

В КУРС ДЕЛА!

Нужно ли снова изобретать колесо?

Если такие термины, как *резиновый макет* и *контейнерный* элемент, кажутся немного пугающими, не поддавайтесь панике. Прежде всего практикум в конце этой главы шаг за шагом проведет вас через весь процесс верстки веб-страниц с помощью каскадных таблиц стилей. Однако нет такого правила, которое бы информировало о том, как вы должны создавать свои собственные CSS-макеты с нуля. Во Всемирной паутине вы найдете множество шаблонных и протестированных макетов, которые можно применять в своих проектах. Сайт Layout Gala содержит 40 различных CSS-дизайнов, которые функционируют в большинстве популярных браузеров (tinyurl.com/nfesllb). Представленные макеты — каркасы, состоящие из элементов `div` и CSS-кода, который позиционирует их. Все, что вам нужно сделать, — заполнить их собственными стилями форматирования, такими как свойства шрифта и изображения.

Посетите также сайт Templated (templated.co), который содержит свыше 850 бесплатных CSS- и HTML-шаблонов. Эти современные шаблоны включают изображе-

ния, фоновые цвета и свойства типографики, которые принесут вашему сайту популярность.

Существует также немало *генераторов макетов* — эти онлайн-инструменты позволяют настраивать такие основные параметры, как количество колонок, вид макета (резиновый или фиксированный) и т. д. Сайт Layout Generator (pagecolumn.com) содержит простой инструмент, с помощью которого можно создать многоколоночный дизайн страницы, после чего сгенерированные HTML- и CSS-файлы можно будет скачать. В главе 16 вы узнаете о том, как использовать *модульные сетки* — CSS-файлы с колонками, организованными определенным способом, — для создания страниц со сложным дизайном.

Соблюдение порядка элементов `div` в HTML-коде может показаться чем-то не очень важным, пока вы не попытаетесь просмотреть веб-страницу без каскадных таблиц стилей, что имеет место для многих альтернативных браузеров, включая программы экранного доступа, которые произносят контент страницы вслух для слабовидящих посетителей. Без каскадных таблиц стилей весь контент боковой панели (он часто включает навигационные элементы, рекламу и другую информацию, не относящуюся к основному контенту страницы) появится перед содержимым, ради которого зашел посетитель. Неудобство, заключающееся в необходимости прокручивать одно и то же содержимое боковой панели на каждой странице, может отпугнуть многих посетителей. Кроме того, ваша страница будет неудобна пользователям с недостатками зрения, которым придется выслушивать, как их экранный диктор читает длинный список ссылок и рекламы, прежде чем получить реально необходимую информацию.

Если же и это никак не затрагивает ваш проект, то стоит поволноваться насчет поисковых систем. Многие из них ограничивают количество HTML-кода, считываемое при поиске сайта. На особенно длинной веб-странице они просто остановятся на определенном месте, возможно, упустив важное содержимое, которое *должно* быть проиндексировано. Кроме того, большинство поисковых систем придает большее значение тому HTML-коду, который находится в начале файла. Таким образом, если вас волнует релевантность вашего сайта при выдаче результатов поиска такими системами, то имеет смысл убедиться в том, что важный контент расположен как можно ближе к началу HTML-кода страницы.

На левой верхней схеме на рис. 13.4 HTML-код с основным контентом находится между левой и правой боковыми панелями, что лучше, чем помещать его после этих блоков. Вы можете даже поместить основной контент перед HTML-кодом *обеих* боковых панелей, оборачивая его и левую боковую панель одним элементом `div` и выравнивая его по левому краю. Затем *внутри* этого элемента `div` нужно выровнять основной контент по правому краю, а левую боковую панель — по левому краю (см. рис. 13.4, *внизу*). Теперь HTML-код основной колонки расположен перед остальными элементами `div`.

Вложенные обтекаемые элементы

Нижняя схема на рис. 13.4 иллюстрирует другую полезную методику — выравнивание элементов *внутри* обтекаемых элементов. Предположим, что `div`-оберток основного контента *(3)* и левой боковой панели *(4)* не существует, а были оставлены только

обертка колонки *(2)* и правая боковая панель *(5)*. У вас получится базовый двухколоночный дизайн, где одна колонка выровнена по левому краю, а другая — по правому краю. На самом деле это все еще двухколоночный дизайн, хотя две `div`-обертки (*3* и *4*) помещены внутри обертки колонки. Различие в том, что левая колонка сама разделена на две колонки.

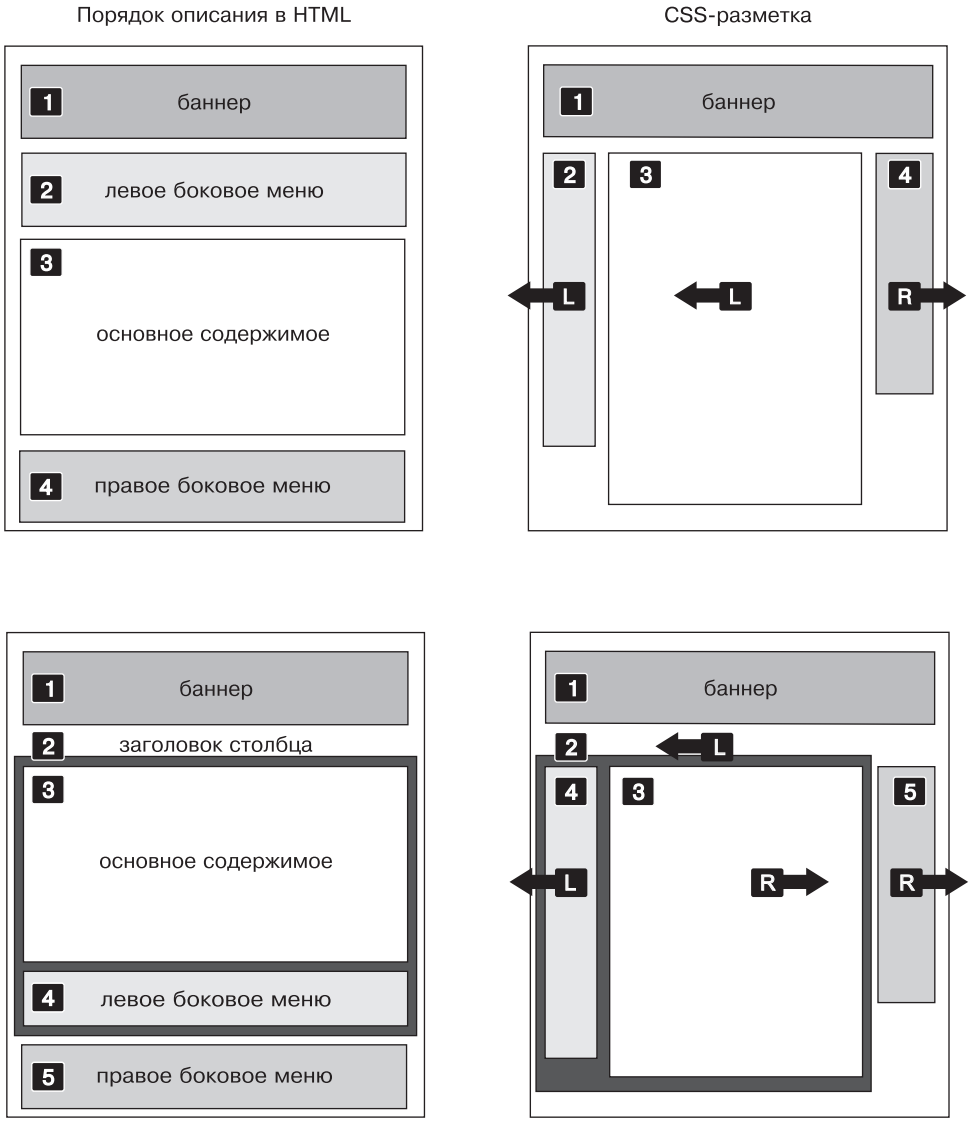

Рис. 13.4. Существует несколько способов сверстать страницу с обтекаемыми элементами. Гибкость каскадных таблиц стилей предоставляет множество вариантов создать многоколоночный макет

Хотя подобное выравнивание слегка смущает, оно бывает полезным во многих случаях. Во-первых, оно позволяет добавлять колонки внутрь других колонок. В трехколоночном макете, показанном на рис. 13.5, *вверху*, используется маленький блок для советов в средней колонке, внутри которого, в свою очередь, также созданы две колонки. Вкладывая одни выравниваемые элементы внутрь других, вы можете создавать достаточно сложные дизайны.

Решение проблем с обтекаемыми элементами

Когда вы начнете активно работать с каскадными таблицами стилей, то, как и многие веб-разработчики, вероятно, столкнетесь с некоторыми сложностями при работе с выравниваемыми (обтекаемыми) элементами. В этом разделе описаны распространенные проблемы и пути их решения.

Запрет обтекания и элементы-контейнеры

Выравниваемые элементы — мощные средства проектирования, поскольку позволяют содержимому обтекать их по краям. Выравнивание фотографии позволяет тексту, находящемуся ниже, подняться и обернуть изображение (см. рис. 13.1). Когда вы разрабатываете дизайны, основанные на выравниваемых колонках, иногда не требуется, чтобы содержимое передвигалось и оказывалось рядом с выровненным элементом. Например, вы хотите указать записи об авторском праве, контактную информацию или другие сведения в нижней части веб-страницы, ниже остального контента.

В двух и трехколоночных макетах, которые мы уже рассматривали, если основная колонка короче любой колонки с обтекаемой боковой панелью, нижний колонтитул может сместиться вверх, оказавшись рядом с левой выровненной колонкой (рис. 13.6, *слева*). Чтобы оставить нижний колонтитул под боковой панелью, вы можете использовать свойство `clear` (см. раздел «Управление обтеканием контента с помощью плавающих элементов» главы 7). Оно устанавливает, с какой стороны элемента запрещено его обтекание другими элементами. Вы можете отменить обтекание с левого края элемента. При этом все другие элементы на этой стороне опустятся вниз и будут располагаться под текущим (`clear: left;`). Аналогично свойство `clear: right;` отменяет обтекание с правой стороны элемента. Для нижнего колонтитула и других элементов, которые должны находиться в нижней части страницы, вы должны устранить обтекание *как* слева, *так и* справа:

```
footer { clear: both; }
```

Другая проблема возникает, если вы выравниваете один или несколько элементов внутри необтекаемого элемента-контейнера, такого как `div`. Если обтекаемый элемент выше, чем остальной контент в контейнере, он придерживается основания содержащего его объекта. Эта путаница особенно видна, если у элемента есть фоновый цвет или граница. В верхней части веб-страницы, показанной на рис. 13.7, используется элемент `div`, в котором определены заголовок `h1` и две колонки, созданные двумя обтекаемыми контейнерами `div`. Фон и граница, которые появляются только по краям

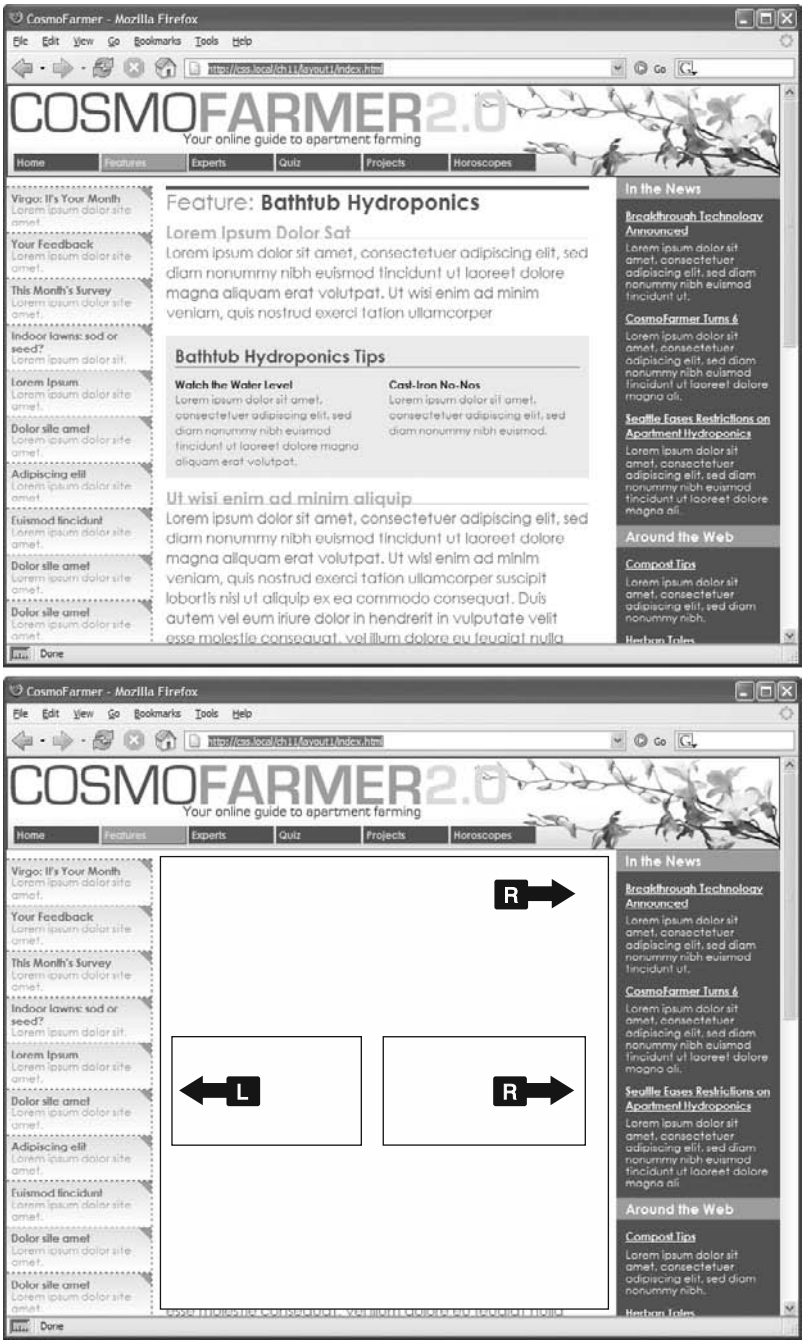

Рис. 13.5. *Вверху:* создавайте вложенные колонки выравниванием элементов внутри других обтекаемых элементов. *Внизу:* не важно, в каком направлении выравнивается контейнер (в этом случае — по правому краю), — вы лишь выравниваете две дополнительные колонки по левому и правому краю

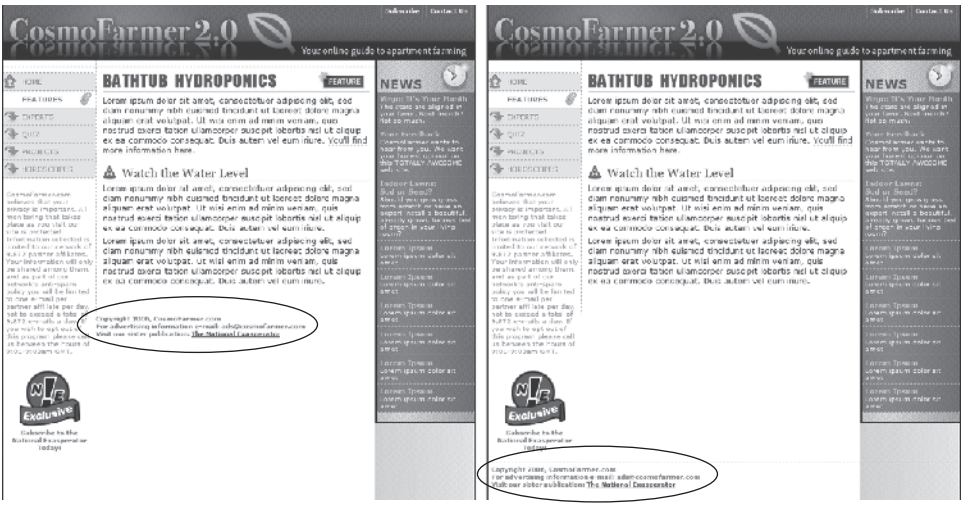

Рис. 13.6. Объект не всегда должен обтекать по краям выравниваемого элемента *(слева)*. Сведения об авторских правах и другие материалы, расположенные в нижней части страницы, всегда должны располагаться отдельно от обтекаемых элементов, находящихся рядом. Чтобы добиться этого, используйте свойство clear

заголовка, в действительности применяются ко всему элементу `div`, включая область с двумя колонками. Однако, поскольку колонки выравниваются, они выходят за пределы блока вместо того, чтобы расширять границы области.

ПРИМЕЧАНИЕ

Чтобы узнать, почему обтекаемые элементы могут выходить за пределы содержащих их блоков, посетите сайт tinyurl.com/369n3.

Пример подобной проблемы показан на рис. 13.7. В этом случае каждое изображение выровнено по левому краю внутри содержащего их элемента `div`, для которого задана граница. Поскольку изображения выше, чем их блоки, они выходят за пределы блоков. К сожалению, этот пример еще хуже, чем предыдущий, потому что каждый рисунок смещает нижнее изображение вправо, создавая ступенчатый эффект.

Избавиться от проблем, возникающих с обтекаемыми элементами, можно несколькими способами. Перечислим наиболее популярные из них.

- **Добавьте в конце div-контейнера запрещающий обтекание элемент.** Это решение — наиболее простое. Нужно лишь добавить элемент, например разрыв строки или горизонтальную линию, в конце контейнера `div`, содержащего обтекаемый элемент (то есть прямо перед закрывающим тегом `</div>`). Затем используйте свойство `clear`, чтобы установить этот дополнительный элемент под обтекаемыми элементами. Такой метод расширяет контейнер, выявляя его фон и границу. Вы можете указать разрыв строки — `<br>` (HTML) или `<br/>` (XHTML) — *перед* закрывающим тегом `</div>` и добавить к нему класс: `<br class = "clear"/>`. Затем создайте для него следующий стиль:

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

Недостаток метода заключается в добавлении дополнительного HTML-кода.

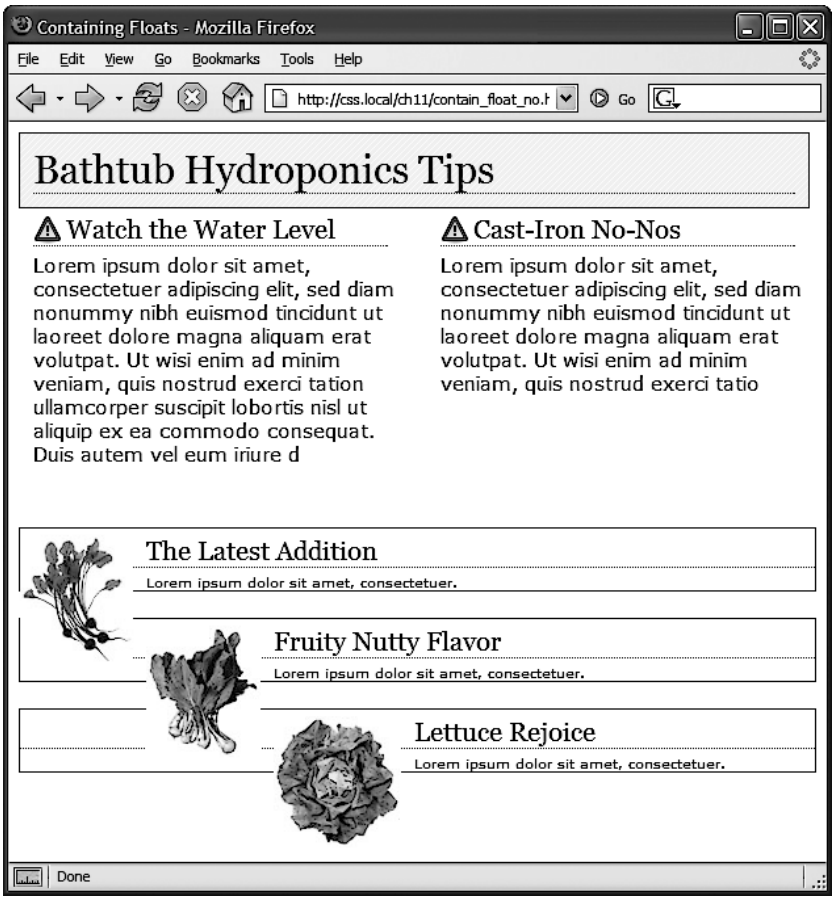

Рис. 13.7. Обтекаемый элемент может выйти за пределы содержащего его блока, если он выше его. Если для блока определены фоновый цвет или граница, то выходящие элементы могут выглядеть так, как будто даже не являются частью блока *(вверху)*

- **Выровняйте элемент-контейнер.** Более легкий путь состоит в том, чтобы выровнять элемент `div`, который содержит обтекаемые элементы. Выровненный контейнер `div` расширяется так, чтобы полностью вмещать любые обтекаемые элементы. На рис. 13.8, *вверху*, элемент `div`, содержащий заголовок и две обтекаемые колонки, выровнен по левому краю страницы. При необходимости вся его область — фон и границы — расширятся, чтобы соответствовать контенту внутри, включая обтекаемые элементы. Это странно, но это так.

  Если вы выбрали такой метод, убедитесь, что добавили свойство `clear` к любому элементу, который расположен после обтекаемого контейнера. Так вы гарантируете, что следующий элемент будет находиться под контейнером.

- **Используйте свойство overflow: hidden.** Другой распространенный метод состоит в добавлении следующего свойства в стиль блока-контейнера:

```
overflow: hidden;
```

- **Свойство overflow: hidden — одно из странностей каскадных таблиц стилей.** Оно провоцирует контейнер расширяться и вмещать обтекаемые элементы. В целом этот метод работает очень хорошо. Тем не менее, если у вас есть абсолютно позиционированные элементы внутри контейнера, они могут не отобразиться. Вы можете попасть в такую ситуацию, если у вас есть раскрывающийся список внутри другого элемента и, когда он раскрывается, кажется, что список находится за пределами элемента-контейнера. Если это так, используйте какой-либо другой способ из описанных на этих страницах.
- **Воспользуйтесь методом очистки потока (clearfix).** Эта технология, созданная Николасом Галлахером, заключается в добавлении к элементу, содержащему обтекаемый элемент, всего нескольких стилей и имени класса. Такой способ наиболее свеж в длинной эволюции методов, использующих псевдокласс `:after`. Чтобы воспользоваться им, нужно добавить в таблицу стилей следующий код:

```
.clear:after {
  content: " ";
  display: table;
  clear: both;
}
```

После добавления этих стилей в таблицу к `div`-контейнеру, *содержащему* выпадающие обтекаемые элементы, нужно лишь добавить класс: `<div class="clear">`. Если используются семантические элементы HTML5, например `article` или `footer`, класс нужно добавить и к ним: `<article class="clear">`. Взгляните на нижнее изображение на рис. 13.8. Эта технология работает очень надежно, но, в отличие от предыдущих двух технологий, вам потребуется наличие на странице дополнительного HTML-кода (класса).

ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

Простой способ создания нескольких колонок

В CSS представлен модуль *многоколоночной верстки*: он позволяет разделить единый элемент (например, заполненный текстом `div`-контейнер) на три, четыре колонки или более. Модуль предоставляет свойства каскадных таблиц стилей для определения количества колонок, пустого пространства между ними и добавления линий между колонками.

Такой способ верстки предназначен для имитации газетных или журнальных страниц, в которых одна длинная публикация простирается от верхней до нижней части одной колонки, продолжается во второй колонке и т. д. То есть он не предназначен для колонок не связанного между собой контента (например, боковой панели со ссылками и колонки, содержащей основной контент). Кроме того, для каждой отдельной колонки нельзя указать ширину, отличную от ширины других колонок, все они будут идентичны.

Верстка в несколько колонок хорошо смотрится на страницах журналов, у которых страница имеет определенный размер и вы можете видеть ее всю, что упрощает чтение текста в колонке до конца страницы и переход к верхней части следующей колонки в верхней части страницы. Но в случае с веб-страницей размер экрана по высоте устройства посетителя вам неизвестен, поэтому длинная колонка текста может заставить посетителя прокручивать страницу вниз, чтобы добраться до нижней части, затем прокручивать ее вверх, чтобы прочитать следующую

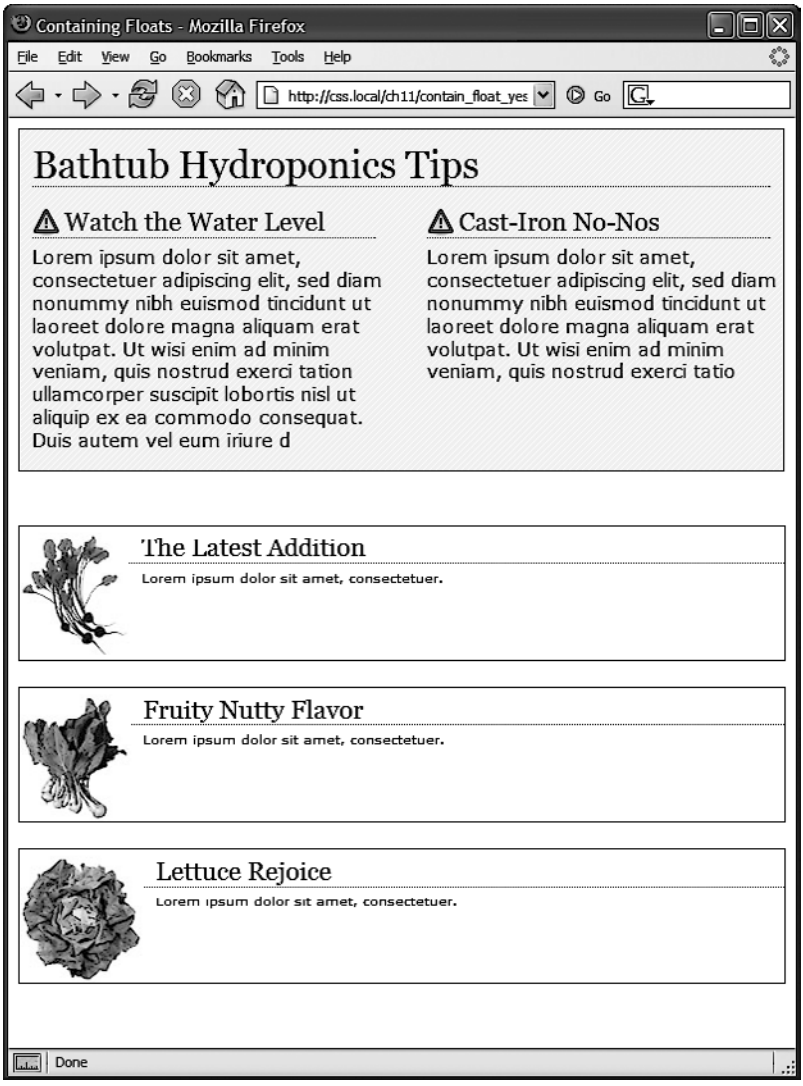

Рис. 13.8. Не позволяйте обтекаемым элементам выпадать из нужных позиций на странице. Есть несколько способов сохранить обтекаемые элементы внутри содержащего их контейнера

ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

колонку. Без учета размера колонок вы заставите людей прокручивать страницу вниз-вверх, чтобы прочитать ее контент.

Тем не менее технология создания нескольких колонок хорошо подойдет для длинных маркированных списков с небольшими фрагментами текста в пунктах. Вместо одного длинного маркированного списка можно будет распределить пункты по странице в нескольких колонках, экономя драгоценное вертикальное пространство.

Для задания количества колонок можно воспользоваться свойством `column-count`, для настройки про-

межутков меду ними — свойством `column-gap`, а для отображения линий между колонками — свойством `column-rule`. Чтобы установить стиль, размер и цвет линий между колонками, используется точно такой же синтаксис, как и для границ (см. раздел «Добавление границ» главы 7).

Эти свойства применяются к элементу, содержащему контент, который нужно разбить на колонки. Предположим, что имеется несколько абзацев текста в контейнере `div`. При применении этих свойств к такому `div`-контейнеру абзацы будут заливаться в несколько колонок. Если, к примеру, этому контейнеру присвоен класс с именем `multicol`, то для получения трехколоночной конструкции с промежутками между колонками 1 em и пунктирными линиями можно создать следующий стиль:

```
.multicol {
  column-count: 3;
  column-gap: 1em;
  column-rule: 1px dotted black;
}
```

Следует напомнить, что Internet Explorer 9 и более ранние версии этого браузера не поддерживают многоколоночные свойства. Кроме того, нужно использовать вендорные префиксы для браузеров Chrome и Safari (`-webkit-column-count`, `-webkit-column-gap` и `-webkit-column-rule`) и Firefox (`-moz-column-count`, `-moz-columngap` и `-moz-column-rule`). А браузеры Internet Explorer 10 (и версии выше), Edge и Opera поддерживают эти свойства без префиксов.

Существуют и другие свойства для задания нескольких колонок, о которых можно прочитать на официальной странице Консорциума W3C: tinyurl.com/cjf2ad. Кроме того, простое руководство по созданию нескольких колонок доступно по адресу tinyurl.com/pg96bv3, а интерактивное средство для генерации многоколоночной верстки — по адресу tinyurl.com/ok33y7b.

Заполнение колонок по высоте

HTML-таблицы не лучшее средство для верстки веб-страниц. Они требует большого объема кода, их трудно обновлять и они не работают так же хорошо в альтернативных браузерах, например тех, что используются в смартфонах. Но, что касается верстки, у таблиц есть один плюс — возможность создавать колонки одинаковой высоты. Их применение позволяет добавлять фоновый цвет или изображение к отдельной колонке и заполнять колонками всю высоту страницы. Фоны двух боковых панелей, показанных на рис. 13.9, *вверху*, заполняют всю высоту, создавая сплошные, широкие полосы с обеих сторон страницы.

Обтекаемые элементы не так великолепно справляются с задачей. Ячейки таблицы в строке всегда одной и той же высоты, чего не скажешь о `div`-контейнерах. Высота обтекаемого элемента обычно определяется его контентом. Когда его мало, сам элемент тоже мал. Поскольку фоновое изображение или фоновый цвет заполняют только обтекаемый элемент, у вас могут получиться колонки другого цвета, не достигающие основания страницы, как выделено на рис. 13.9, *внизу*.

ПРИМЕЧАНИЕ

Гибкая блочная модель решает проблему разности высоты ячеек в строке. О ней вы узнаете в главе 15.

Но, как и у большинства проблем, связанных с каскадными таблицами стилей, доступно несколько обходных путей. Наиболее действенным из них является метод *псевдоколонок*. Секрет состоит в том, чтобы добавить фоновые изображения

к элементу, который *оборачивает* низкую боковую панель и другие колонки на странице. Скажем, в вашем HTML-коде есть два элемента `div`, в которых содержится контент левой боковой панели и основной контент страницы:

```
<div class="sidebar">Контент боковой панели</div>
<div class="main">Основной контент страницы, в этой колонке много текста и она на-
много выше боковой панели.</div>
```

Элемент `div` боковой панели выравнивается по левому краю страницы, и ширина его равна 170 пикселам. Поскольку на боковой панели немного текста, она короче, чем основной контент. Предположим, что вы оборачиваете данный HTML-код в `div`-контейнер следующим образом:

```
<div class="wrapper">
<div class="sidebar">Контент боковой панели</div>
<div class="main">Основной контент страницы, в этой колонке много текста и она на-
много выше боковой панели.</div>
</div>
```

Внешний контейнер увеличится и станет таким же высоким, как самый высокий элемент внутри него, поэтому, если контент `div`-контейнера с классом `main` очень велик, обертывающий блок `div` с классом `wrapper` будет такой же высоты. В этом все волшебство: создайте стиль для обертки `div` с фоновым изображением, ширина которой равна ширине боковой панели, подобрав нужный фоновый цвет. Таким образом, если фоновое изображение повторяется по вертикали, оно сформирует сплошную полосу размером, равным высоте `div`-контейнера (рис. 13.9, *вверху*).

```
.wrapper { background: url(images/col_bg.gif) repeat-y left top; }
```

Браузеры отображают это фоновое изображение прямо *позади боковой панели*, создавая иллюзию того, что у панели есть фоновый цвет.

ПРИМЕЧАНИЕ

Вы не ограничены сплошным цветом. Поскольку допускается использовать изображения, можно создать декоративный узор, который будет чередоваться до конца страницы.

Применение этого метода к двум колонкам немного сложнее. Во-первых, нужно добавить два контейнера-обертки:

```
<div class="wrapper1">
<div class="wrapper2">
<div class="sidebar1">Контент первой боковой панели</div>
<div class="sidebar2">Контент второй боковой панели</div>
<div class="main">Основной контент страницы, в этой колонке много текста и она на-
много выше боковых панелей.</div>
</div>
</div>
```

ПРИМЕЧАНИЕ

Если обертка и каждая колонка имеют фиксированную ширину, вы можете создать вид псевдоколонки для левой и правой панели с одним изображением и div-оберткой. Для этого сделайте рисунок таким же широким, как и div-обертка, и с левой стороны залейте цвет по ширине левой боковой панели, а с правой — цвет по ширине правой панели. Центральная часть должна быть цвета центральной колонки.

Рис. 13.9. Колонки во всю высоту страницы с фоновым цветом — распространенный прием в веб-дизайне. Боковые панели слева и справа *(вверху)* демонстрируют, как сплошной фоновый цвет помогает визуально разделить области страницы. Когда фон боковой панели резко прерывается *(внизу)*, появляется пустое пространство, которое выглядит непривлекательно

Если первая боковая панель отображается в левой части страницы, а вторая — в правой, то создается два стиля. Примените один стиль к первому контейнеру `div`, чтобы добавить фон к левой боковой панели; другой же стиль примените ко второму контейнеру `div`, добавив фон к правой боковой панели (рис. 13.10, *внизу*).

```
.wrapper1 { background: url(images/col1_bg.gif) repeat-y left top; }
.wrapper2 { background: url(images/col2_bg.gif) repeat-y right top; }
```

При добавлении фонового изображения к правой колонке убедитесь, что вы помещаете изображение вверху справа во второй обертке так, что оно простирается позади второй боковой панелью в правой части страницы.

Основная проблема псевдоколонок связана с тем, что их очень трудно заставить работать, если для всех колонок ширина задана в процентах. Если ширина боковых панелей задана в процентах от ширины окна браузера, они могут быть у́же или шире в зависимости от характеристик монитора посетителя. Техника псевдо-колонок требует помещения изображения в элемент, являющийся контейнером. Это изображение имеет определенную ширину и не будет масштабироваться при изменениях ширины окна браузера, а соответственно, и ширины колонок.

Один из удачных обходных маневров — использование линейных градиентов (см. раздел «Использование градиентных фонов» главы 8) в элементах-контейнерах. Градиент становится для колонок цветом фона внутри элемента-контейнера. С его помощью можно также создать эффект сплошного цвета.

Как уже говорилось, линейные градиенты позволяют устанавливать цветовые узлы, то есть позиции, в которых появляется новый цвет. Если первый и второй цветовой узлы имеют одинаковый цвет, к примеру белый, вместо градиента с переходами цвета вы получите сплошной белый фон. Кроме того, можно установить второй цветовой узел в позиции первого, чтобы второй цвет отображался сразу после первого без какого-либо градиентного эффекта.

Предположим, что есть трехколоночный дизайн. Первая колонка имеет ширину 25 %, вторая — 50 %, третья — 25 %. Нужно, чтобы у первой колонки фоновым цветом был красный, у второго — белый, а у третьего — синий.

1. Заключите все три колонки в элемент-контейнер:

```
<div class="wrapper">
  <div class="sidebar1">…сюда помещается содержимое…</div>
  <div class="main">…сюда помещается содержимое…</div>
  <div class="sidebar2">…сюда помещается содержимое…</div>
</div>
```

   Контейнер является тем самым элементом, к которому добавляется градиент. Кроме того, если все три колонки внутри контейнера обтекаемы, то для содержания всех этих обтекаемых элементов нужно будет воспользоваться одним из методов, рассмотренных ранее в разделе «Решение проблем с обтекаемыми элементами».

2. Добавьте линейный градиент с цветовыми узлами в соответствии с шириной колонок:

```
.wrapper {
  background-image: linear-gradient(left,
```

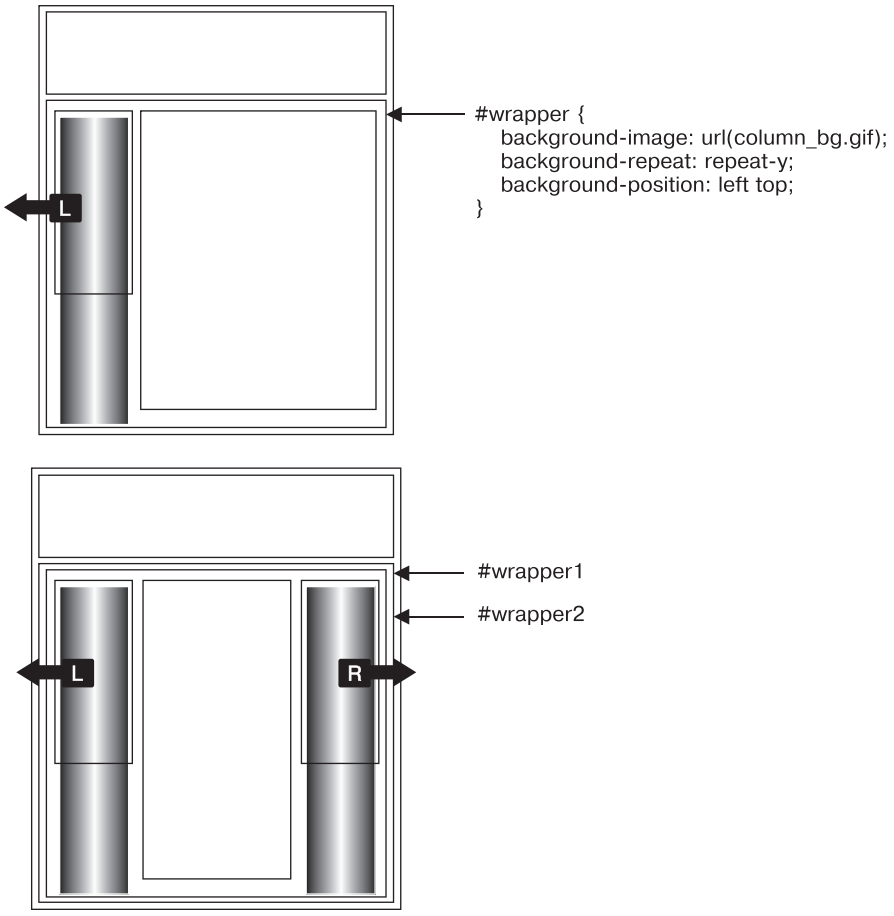

Рис. 13.10. Для креативного решения проблем каскадных таблиц стилей иногда необходимо выходить за пределы элемента

```
    red 0%,
    red 25%,
    white 25%,
    white 75%,
    blue 75%,
    blue 100%);
}
```

Красный цвет займет пространство от 0 % (от левого края контейнера) до 25 %. Поскольку в обоих узлах используется один и тот же цвет, перехода не будет. Затем на отметке 25 %, там же, где закончился красный цвет, начинается белый цвет, поэтому градиента опять не будет. Белый сплошной цвет займет пространство до 75 % отметки. Затем в позиции, где заканчивается белый цвет, начнется синий цвет, который займет пространство до 100 % (до правого края контейнера). То есть получатся три колонки со сплошным фоновым цветом, ширина которых будет изменяться по мере изменения ширины окна браузера.

Недостаток использования линейного градиента заключается в том, что он поддерживает только сплошные цвета (и, разумеется, градиенты, если вы того пожелаете), поэтому вы не можете использовать в качестве фона изображения или границы по краям каждой колонки. Кроме того, Internet Explorer 9 и ранние версии этого браузера не поддерживают градиенты.

ПРИМЕЧАНИЕ

Есть еще несколько способов отобразить колонки с одинаковой высотой. Обзор различных технологий можно найти в блоге эксперта по каскадным таблицам стилей Криса Койера по адресу tinyurl.com/p5swv4s.

ПРИМЕЧАНИЕ

Существует техника для создания элемента, находящегося позади обтекаемой колонки, которая заключается в использовании псевдокласса :before. Подробнее о ней можно прочитать по адресу tinyurl.com/q7lglhu.

Предотвращение выпадений обтекаемых элементов

Может случиться так, что внезапно одна из колонок на вашем сайте опустится ниже других (рис. 13.11, *вверху*). Как видно из рисунка, на странице достаточно пространства для размещения всех колонок рядом друг с другом, но этого не получилось. В данном случае произошло *выпадение обтекаемого элемента*. Если ширина обтекаемых элементов хотя бы на пиксел шире содержащего их блока (например, контейнера `div` или даже окна браузера), то последний элемент опускается ниже других (см. рис. 13.11, *вверху*). Фактическая высота элемента — это комбинация множества свойств каскадных таблиц стилей. На среднем изображении видно, что контур по краям области основного контента слишком широк (выделено на рисунке), чтобы позволить правой боковой панели занять желаемое расположение. Скорректировав значения ширины, отступов или полей всех элементов, вы можете решить проблему (см. рис. 13.11, *внизу*).

Обтекаемая колонка опускается вниз из-за недостатка пространства. Будьте осторожны, если вы ограничиваете ширину *каждой* колонки. Если ширина доступного пространства в окне браузера (или контейнера в фиксированном макете) меньше общей ширины колонок, то обтекаемый элемент может опуститься вниз. Кроме того, не забывайте о блочной модели CSS: как обсуждалось в разделе «Изменение высоты и ширины» главы 7, ширина элемента, отображаемого в окне браузера, не определяется лишь значением присвоенного свойства `width`. Отображаемая ширина любого элемента — это комбинация его размеров контента, а также границ, отступов и полей слева и справа. Для соответствия колонкам окно браузера (или контейнер) должно подстроить совокупность данных размеров.

Возьмем, например, простой трехколоночный дизайн, представленный на рис. 13.11. Как вы можете видеть на верхнем изображении, эти три колонки не располагаются рядом друг с другом. Рассмотрим, из-за чего возникла проблема.

- **div-обертка.** Контейнер `div` с фиксированной шириной заключает в себя весь контент. Его ширина равна 760 пикселов, таким образом, ширина всех колонок не может быть в сумме больше, чем это значение.

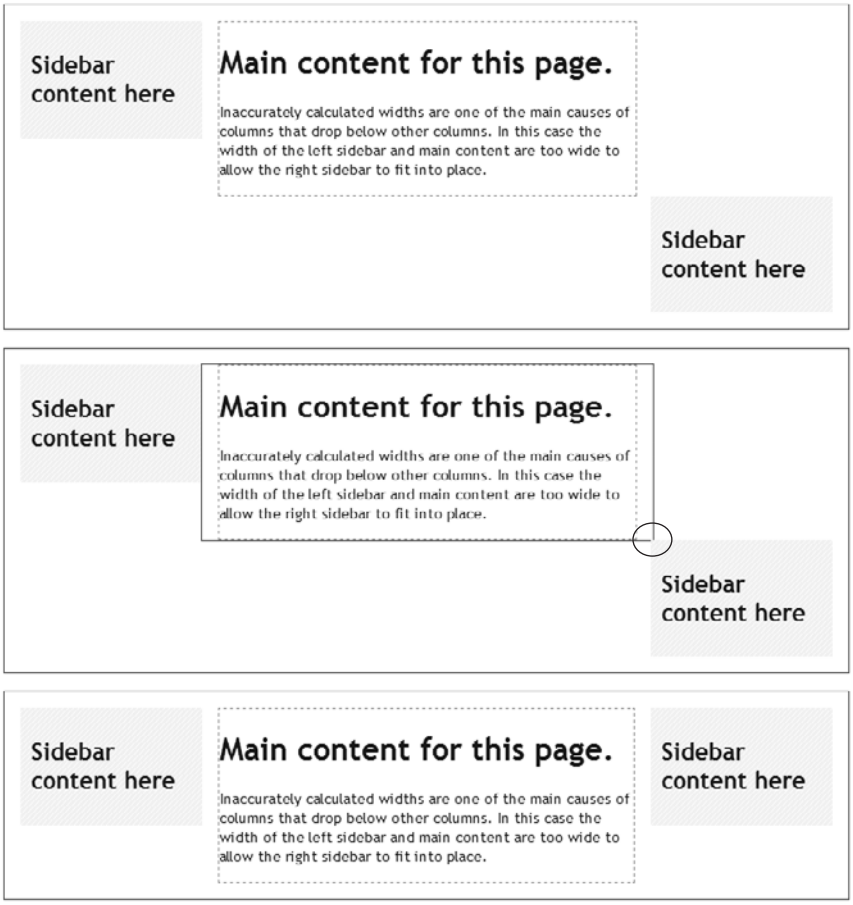

Рис. 13.11. Чтобы разрушить дизайн страницы, достаточно ошибки в 1–2 пиксела

- **Первая боковая панель (в левой части).** Ширина — 150 пикселов, но у панели также есть отступы по 10 пикселов, из-за чего общая ширина составляет 170 пикселов (150 пикселов панели + 10 пикселов отступа слева + 10 пикселов отступа справа).
- **Основной контент.** Имеет ширину 390 пикселов, а также включает по 1 пикселу обеих границ и по 15 пикселов полей слева и справа, из-за чего общая ширина равна 422 пикселам (390 пикселов ширины контента + 1 пиксел левой границы + 1 пиксел правой границы + 15 пикселов левого поля + 15 пикселов правого поля).
- **Вторая боковая панель (в правой части).** Ширина этого элемента равна 150 пикселам. Он также включает в себя по 10 пикселов левого и правого отступов; в результате получается 170 пикселов, как и в случае первой боковой панели.
- **Фактическая ширина всех добавленных элементов составляет 762 пиксела.** Это на 2 пиксела больше ширины `div`-обертки. На рис. 13.11, *посередине*, показана

рамка по краям контейнера div для основного контента, которая представляет его полную ширину *вместе* с полями. Всего двух лишних пикселов ширины (выделено на рисунке) достаточно для того, чтобы заставить колонку опуститься вниз. Решение заключается в удалении 2 пикселов из значения ширины любого элемента. К примеру, измените ширину левого и правого поля основного контента с 15 до 14 пикселов, что добавит дополнительное пространство, необходимое для размещения всех трех колонок, которые теперь будут находиться рядом.

Подведем итог: выпадение обтекаемых элементов вызывается недостатком пространства для вмещения всех колонок.

Способ упрощения вычислений заключается в удалении границ или отступов div-контейнеров или элементов, содержащих колонки. Тогда при установке трем колонкам значения ширины 20, 60 и 20 % соответственно вы будете знать, что они поместятся рядом друг с другом, поскольку вместе составляют 100 % и нарушающие это положение вещей отступы или границы отсутствуют. Если нужны дополнительные отступы, их можно добавить к элементам, находящимся внутри колонок, например присвоить одинаковые отступы слева и справа заголовкам, абзацам и другим элементам внутри div-контейнера. Это потребует дополнительной работы, но предотвратит потенциальное выпадение обтекаемых элементов, вызванное превышением суммарной ширины колонок в 100 %.

Если нужны еще и границы, можно применить прием вложения div-контейнеров:

```
<div class="column1">
  <div class="innerColumn">
  … здесь размещен контент …
  </div>
</div>
```

Затем устанавливается ширина внешнего div-контейнера, в данном случае имеющего класс column1, и добавляются отступы и границы к внутреннему div-контейнеру с классом innerColumn. Устанавливать ширину внутреннего div-контейнера не нужно, он заполнит всю ширину внешней колонки автоматически.

Предотвращение выпадения обтекаемого элемента с помощью свойства box-sizing

Основной причиной выпадения обтекаемых элементов является странный способ, который используется браузерами для вычисления фактической экранной ширины элемента. Например, вы устанавливаете ширину 100 пикселов, но браузер визуализирует элемент шириной 122 пиксела, потому что вы добавили также 10-пиксельный левый и правый отступы и 1-пиксельную границу по краям элемента. К счастью, есть CSS-свойство, которое позволяет избавиться от столь неприятного способа расчета.

Свойство box-sizing позволяет применить другую модель расчета фактической экранной ширины элемента. Ему можно присвоить одно из трех значений.

- content-box приводит к обычной работе браузера: ширина элемента учитывает только размер контента — значение свойства width.

```
box-sizing: content-box;
```

- `padding-box` позволяет включить левый и правый отступ в расчет ширины элемента. То есть экранная ширина элемента складывается из значения свойства `width`, а также левого и правого отступов. Любые границы по краям элемента не учитываются.

  ```
  box-sizing: padding-box;
  ```

- `border-box` включает отступы, границы и значение свойства `width`. В общем, это то, что вам нужно. При задании этого значения расчеты упрощаются, что помогает избежать выпадения обтекаемых элементов, особенно при использовании ширины, заданной в процентном отношении в сочетании со значениями в пикселах ширины границ и отступов:

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

  Такую схему поддерживает большинство браузеров, включая Internet Explorer 8.

Некоторые веб-разработчики предлагают устанавливать для всех элементов значение `border-box`, чтобы они измерялись одинаково. Для этого используется универсальный селектор, который нужно поместить в верхней части таблицы стилей вместе с кодом сброса CSS:

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

Практикум: многоколоночные макеты

В этом практикуме мы рассмотрим, как создавать многоколоночные макеты, основанные на обтекаемых элементах. Вы создадите трехколоночный резиновый и фиксированный дизайны. Кроме того, вы научитесь применять ряд других методов достижения аналогичного результата.

Чтобы начать обучение, вы должны иметь в распоряжении файлы с учебным материалом. Для этого нужно загрузить файлы для выполнения заданий практикума, 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 13.

Структурирование HTML-кода

Первый шаг в верстке макета на основе каскадных таблиц стилей — идентификация различных элементов на странице. Вы делаете это, обертывая фрагменты HTML-кода в контейнеры `div`, каждый из которых представляет отдельный элемент страницы.

1. Откройте файл `index.html` в редакторе HTML-кода и установите курсор на пустой строке после HTML-комментария: `<!-- первая боковая панель -->`.

   Как вы можете видеть, часть HTML-разметки уже сделана: на данный момент созданы баннер и нижний колонтитул. Прежде чем создавать какие-либо стили, вы должны добавить структуру и контент на страницу. Далее вы добавите элемент `aside` для левой боковой панели.

2. Добавьте открывающий тег <aside> для левой боковой панели: <aside class = "sidebar1">. Затем нажмите клавишу Enter, чтобы перейти на новую строку.

   В примере применяется HTML5-элемент aside, но для создания такой колонки можно воспользоваться и элементом div.

   Если бы вы создавали веб-страницу с нуля, то в этом пункте пришлось бы добавлять HTML-код для боковой панели на странице и, возможно, определять список публикаций сайта, ссылки на родственные сайты и т. д. В данном случае вам не придется этого делать. Код неупорядоченного списка ссылок уже набран в отдельном файле. Осталось только скопировать его и добавить на страницу.

3. Откройте файл sidebar1.txt, скопируйте его содержимое, а затем вернитесь к файлу index.html. Вставьте скопированный HTML-код после тега <aside>, который вы создали в шаге 2 (или тега , если вы решили использовать его).

   Боковая панель почти готова. Осталось лишь закрыть элемент aside.

4. Сразу же после кода, который вы только что добавили, введите код </aside>.

   Вы только добавили на страницу первый элемент макета. Чуть позже мы отформатируем его так, чтобы он был похож на колонку. Но сначала вы должны добавить еще немного кода.

5. Установите курсор на пустую строку после следующего HTML-комментария: <!-- основной контент -->, а затем введите код <article class = "main">.

   Этот раздел будет хранить главное содержимое страницы. Нужный HTML-код вы также возьмете в другом файле.

6. Откройте файл main.txt, скопируйте его содержимое, вернитесь к файлу index.html и вставьте скопированный код после тега <article>, который вы только что создали. Добавьте закрывающий тег </article> точно так же, как в шаге 4. Сохраните HTML-файл.

Это весь HTML-код, который нужен для создания дизайна. Теперь пришло время переключиться на создание каскадной таблицы стилей.

CSS-верстка

Если вы просмотрите страницу, то увидите, что для баннера, навигационных кнопок и текста стили уже созданы. Так получилось потому, что к странице присоединена внешняя таблица стилей с базовым форматированием. Далее нужно создать стили для форматирования колонок на странице.

1. Откройте в редакторе HTML-кода файл styles.css.

   Поскольку веб-страница использует внешнюю таблицу стилей, новые стили будут добавляться в этот CSS-файл. Теперь вы работаете с двумя файлами: HTML и CSS, поэтому перед просмотром страницы в браузере нужно убедиться в том, что сохранены оба файла.

2. Перейдите в конец CSS-файла и найдите комментарий /* стили из практикума главы 13 */. Добавьте ниже этого комментария следующий код:

```
.sidebar1 {
  float: left;
  width: 20%;
}
```

Стиль выравнивает боковую панель по левому краю страницы и задает ей значение ширины, равное 20 %. Свойство `width` играет в этом стиле важную роль: если только вы не выравниваете изображение, для которого задана ширина, всегда нужно ограничивать ширину обтекаемого (выравниваемого) элемента. В ином случае браузер установит ширину на основе контента внутри обтекаемого элемента, что приведет к противоречивым результатам. Здесь ширина задана в процентном отношении, стало быть, она определяется шириной данного контейнера. В нашем случае контейнером является элемент `body`, который заполняет всю ширину окна браузера. Поэтому экранная ширина боковой панели будет зависеть от ширины окна браузера, используемой посетителем.

3. Сохраните HTML- и CSS-файлы и просмотрите файл `index.html` в браузере.

   Боковая панель теперь представляет собой колонку, выровненную по левому краю. Когда текст основной колонки достигает основания боковой панели, он обтекает основание боковой панели, как показано на рис. 13.12. Хоть это и типично для обтекаемых элементов, это не то, что нам нужно сейчас. Чтобы основной контент отобразился в виде отдельной колонки, следует добавить достаточное по размеру поле слева. Это позволит отделить основной контент от боковой панели.

4. Создайте стиль для второй колонки:

```
.main {
  margin-left: 22%;
}
```

   Поскольку ширина боковой панели составляет 20 %, поле размером 22 % смещает основной контент на дополнительные 2 %, создавая промежуток между двумя колонками. Дополнительное пустое пространство не только повышает читабельность текста, но и улучшает внешний вид страницы.

   Взгляните на страницу, и увидите, что получился двухколоночный дизайн.

Добавление колонки

Как вы могли заметить, двухколоночный дизайн создать несложно. Добавив третью колонку, вы сможете преподнести своим посетителям еще большее количество информации. Подобный дизайн также несложно создавать — действия в этом случае практически такие же, как в предыдущей части этого практикума.

1. Откройте файл `sidebar2.txt`. Скопируйте из него весь HTML-код, а затем вернитесь к файлу `index.html`.

   HTML-код для этой колонки помещается после элемента `article` с основным контентом.

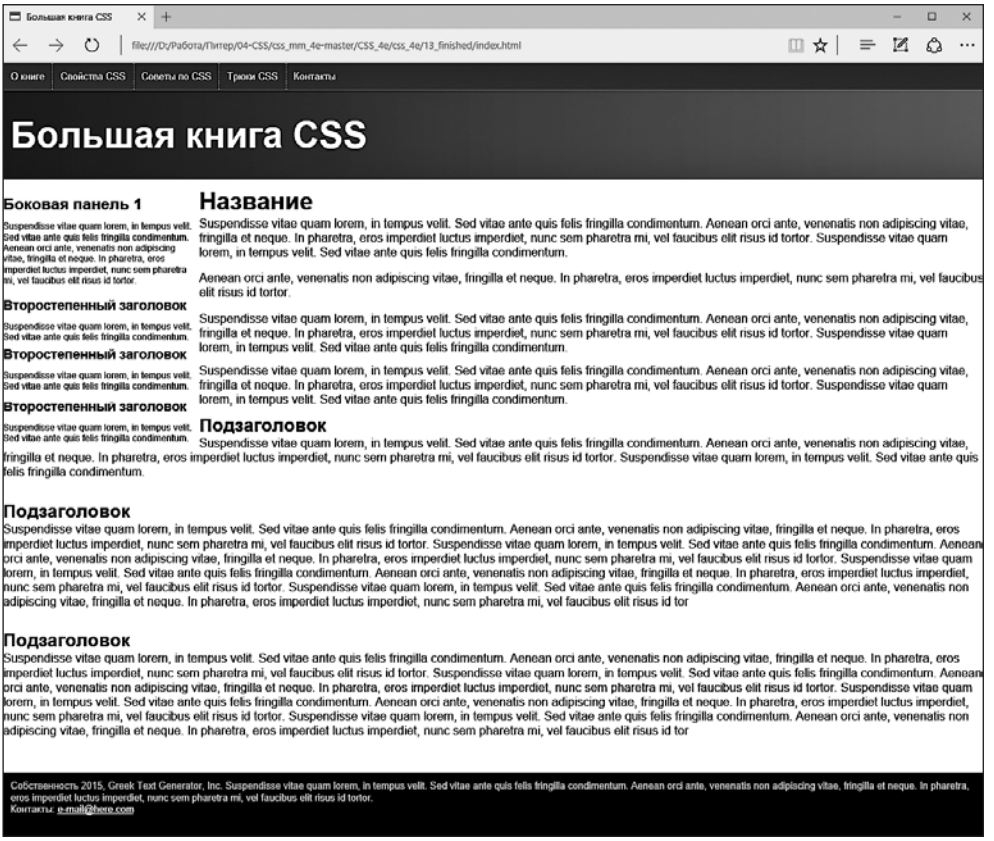

Рис. 13.12. Обтекаемый элемент в действительности не создает колонку на странице. Он лишь смещает любой контент, которое обтекает его, до позиции, где он заканчивается. После этого контент занимает свое место под элементом

2. Найдите ближе к концу файла HTML-комментарий `<!-- вторая боковая панель -->`. Щелкните кнопкой мыши на пустой строке ниже этого комментария.

   Зачастую, когда для структурирования контента страницы используется много `div`-контейнеров, найти нужный закрывающий тег `</div>` бывает нелегко. Вот почему HTML-комментарии, такие как этот, помогают определить HTML-код на странице.

3. Введите код `<aside class="sidebar2>`, нажмите клавишу Enter и вставьте HTML-код, который вы скопировали в шаге 1. Вновь нажмите клавишу Enter и добавьте закрывающий тег `</aside>`. Сохраните HTML-файл.

   Закрыв элемент `div`, вы завершили HTML-код третьей колонки на странице. Теперь приступим к форматированию этой колонки.

4. Вернитесь в редактор HTML-кода к файлу `styles.css`. Под стилем `.main`, который вы создали в шаге 4 в предыдущем подразделе, добавьте следующий код:

```
.sidebar2 {
  float: right;
  width: 20%;
}
```

Указав данный стиль, вы выровняете колонку по правому краю страницы, чтобы по обе стороны основного контента располагались боковые панели.

5. Сохраните все файлы и просмотрите файл `index.html` в браузере.

   Теперь вы должны увидеть нечто странное. Вторая боковая панель отображается ниже основного контента и даже накладывается на нижний колонтитул. Проблема связана с порядком следования HTML-кода. Когда выравнивается элемент, то его обтекает и появляется после него только тот HTML-код, который *следует* в коде за этим обтекаемым элементом. То есть, пока HTML-код второй боковой панели следует за основным контентом, он появляется не рядом с ним, а после него.

   Справиться со сложившейся ситуацией можно двумя способами. Можно переместить HTML-код второй боковой панели (второй элемент `aside`), указав его перед HTML-кодом основного контента (перед элементом `article`). Тогда первая боковая панель переместится влево, вторая — вправо, а основной контент поднимется и разместится между ними.

   Можно выровнять основной контент. Если установить его ширину таким образом, чтобы ширина всех трех колонок не превышала 100 %, все они будут располагаться рядом. В текущем примере будет применен именно этот вариант.

6. Отредактируйте стиль `.main` следующим образом:

```
.main {
  float: left;
  width: 60%;
}
```

   Если сохранить CSS-файл и просмотреть страницу `index.html` в браузере, проявится еще одна проблема — внезапно все снова будет испорчено! Спокойствие: это нижний колонтитул попытался обернуть все обтекаемые элементы и создал тем самым хаос. Как уже ранее говорилось, когда элемент выравнивается и его обтекает другой элемент, фоновый цвет и границы этого элемента фактически расширяются под обтекаемый элемент. Странно, все вроде правильно, а результат разочаровывает. Вполне очевидно, что для решения проблемы нужно, чтобы нижний колонтитул опустился ниже обтекаемых элементов.

7. После стиля `.sidebar2` добавьте следующий код:

```
footer {
  clear: both;
}
```

   Как уже упоминалось, свойство `clear` может опустить элемент ниже колонок. В данном случае оно выталкивает нижний колонтитул ниже колонок (рис. 13.13).

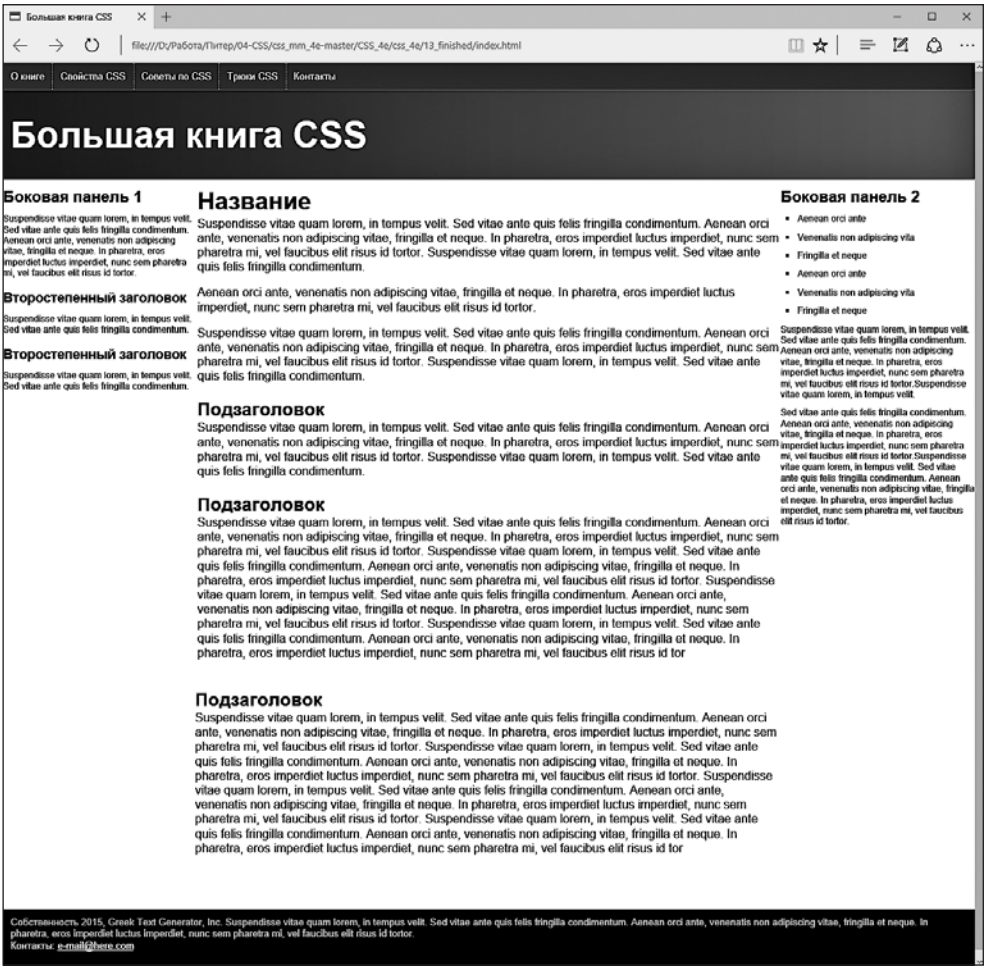

Рис. 13.13. Используя обтекаемые элементы, можно выстроить все три элемента рядом в три колонки

Добавление разрядки

Три колонки у вас уже есть, но текст выглядит слишком скученно. Эти три колонки практически касаются друг друга, и текст на правой боковой панели слишком близко прижимается к краю окна браузера. Исправить ситуацию помогут небольшие отступы.

1. Добавьте отступы в стилях `.sidebar1`, `.main` и `.sidebar2`, придав их коду следующий вид:

```
.sidebar1 {
  float: left;
  width: 20%;
  padding: 0 20px 0 10px;
```

```
}

.main {
  float: left;
  width: 60%;
  padding: 0 20px 0 20px;
}

.sidebar2 {
  float: right;
  width: 20%;
  padding: 0 10px 0 10px;
}
```

Здесь используется сокращенная запись свойства `padding`. Числа представляют верхний, правый, левый и нижний отступы соответственно. То есть в стиль `.sidebar1` добавлен верхний нулевой отступ, 20 пикселов отступа справа, нижний нулевой отступ и 10 пикселов отступа слева.

Если сохранить файл `styles.css` и просмотреть в браузере страницу `index.html`, станет заметна еще одна проблема — выпадение обтекаемого элемента. В результате добавления отступов каждая колонка стала шире, а поскольку общая ширина колонок (20 + 60 + 20) % уже составила в сумме 100 %, дополнительные отступы опустили третью колонку под первые две. Это нужно срочно исправить!

Проблему можно решить несколькими способами. Во-первых, можно убрать отступы из этих стилей и добавить их ко всем внутренним элементам. То есть добавить 10 пикселов правого и левого отступов к элементам `h2`, `h3`, `p` и `ul`. Но это довольно трудоемкий процесс.

Во-вторых, можно убрать отступы из стилей в CSS-файле, а затем в документе `index.html` добавить в каждую колонку `div`-контейнер следующего вида:

```
<aside class="sidebar1">
  <div class="innerColumn">
    … Сюда помещается контент …
  </div>
</aside>
```

Затем в файле `styles.css` нужно создать стиль для добавления отступов:

```
.innerColumn {
  padding: 0 20px 0 10px;
}
```

Поскольку в стиле `.innerColumn` ширина не задается, контейнер просто разрастается, чтобы занять колонку, а отступы смещают все, что находится внутри него (заголовки, абзацы и т. д.), вовнутрь на 10 пикселов. Недостаток такого подхода состоит в необходимости добавления дополнительного кода.

Существует еще один, более простой и широко поддерживаемый браузерами способ.

2. Добавьте в верхней части таблицы стилей еще один стиль (сразу после комментария `/* сброс стилей браузера */`):

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

В этом стиле используется универсальный селектор. Благодаря ему свойство `box-sizing` применяется к каждому элементу страницы. Присвоение этому свойству значения `border-box` инструктирует браузеры учитывать размер отступов и границ вместе со значением свойства `width`. То есть дополнительные отступы не добавляются к установленному ранее значению ширины. Тем самым предотвращаются любые выпадения обтекаемых элементов, поскольку колонки составляют не более 100 % ширины окна браузера.

И наконец, добавьте границы, чтобы отделить колонки друг от друга.

3. Отредактируйте стиль `.main`, добавив в него свойства левой и правой границ:

```
.main {
  float: left;
  width: 60%;
  padding: 0 20px 0 20px;
  border-left: dashed 1px rgb(153,153,153);
  border-right: dashed 1px rgb(153,153,153);
}
```

Эти свойства приведут к добавлению прямых линий с каждой стороны раздела основного контента. Если теперь просмотреть страницу в браузере, она должна выглядеть, как показано на рис. 13.14.

Ограничение ширины

В настоящее время у страницы резиновый дизайн, означающий, что контент расширяется, чтобы заполнить всю ширину окна браузера. Но, допустим, вам нужно, чтобы страница все время оставалась одной и той же ширины, так как вас не устраивает, как она выглядит на широкоформатных мониторах или при очень маленьком размере окна браузера. Изменить резиновый дизайн на фиксированный легко. Начните с добавления HTML-кода.

1. Вернитесь в редактор HTML-кода к файлу `index.html`. Сразу же за открывающим тегом `<body>` добавьте новый элемент `div`:

```
<div class="pageWrapper">
```

Таким образом вы заключите всю страницу в блок, который будет использоваться для управления шириной страницы. Вы должны убедиться, что этот контейнер закрыт.

2. Добавьте закрывающий тег `</div>` перед `</body>`:

```
</div>
</body>
```

Теперь, когда созданный контейнер включает в себя все содержимое страницы, вы можете управлять ее шириной, изменяя ширину данного контейнера.

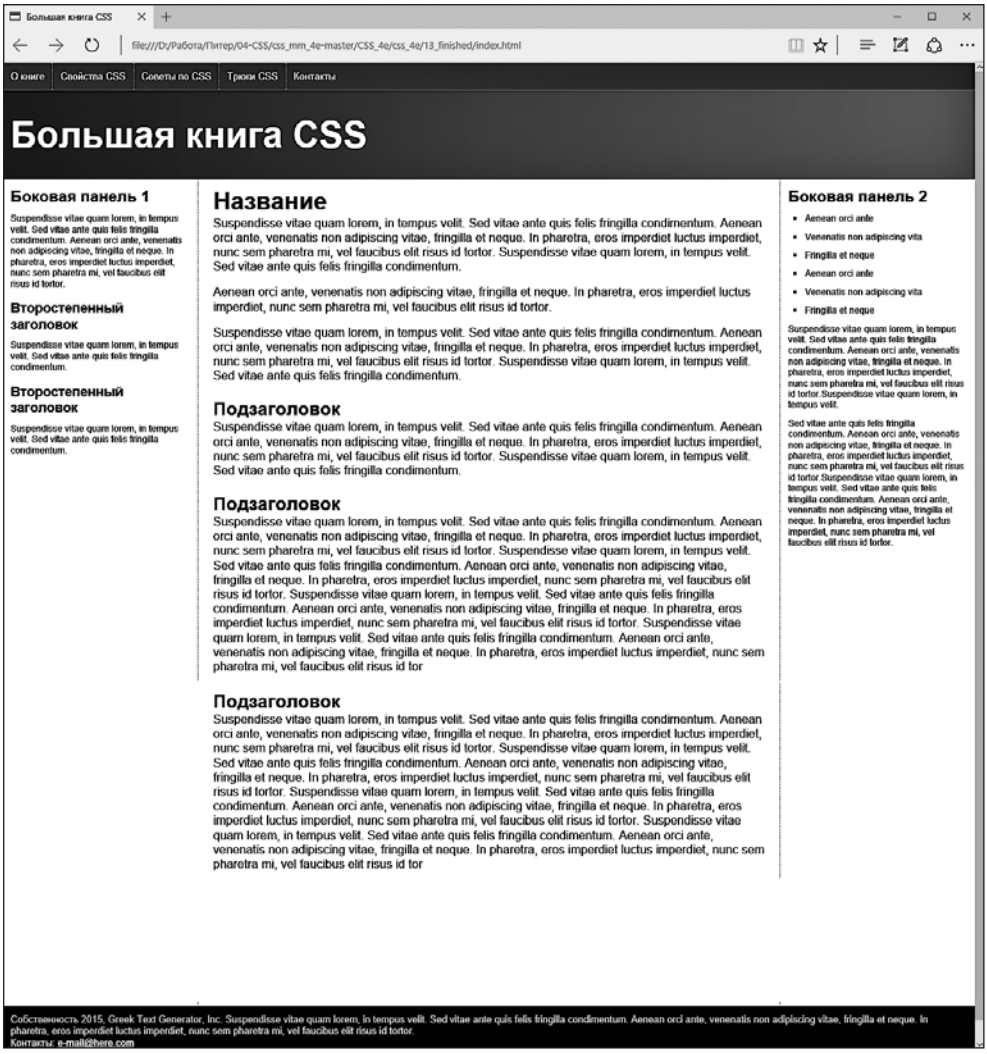

Рис. 13.14. Добавление отступов и границ визуально отделяют колонки друг от друга

3. Сохраните HTML-файл и перейдите к редактированию файла `styles.css`. Добавьте еще один стиль:

```
.pageWrapper {
  width: 960px;
}
```

Если сохранить CSS- и HTML-файлы и просмотреть страницу `index.html` в браузере, вы увидите, что ее контент действительно заключен в пространство шириной 960 пикселов. Если сделать ширину окна браузера меньше 960 пикселов, появятся полосы прокрутки.

Но вам, разумеется, не нужно устанавливать точную ширину. Если хотите, чтобы страница поместилась в более узком (скажем, имеющем ширину 760 пикселов) окне браузера, лучше избегать ограничения ширины. Настоящая проблема заключается в том, что страницу становится трудно читать в слишком широком окне браузера. Другой подход заключается в использовании свойства `max-width`, которое не позволяет `div`-контейнеру увеличиваться свыше определенной величины, но не препятствует более узкому значению его ширины для помещения на небольших экранах. Раз уж вы за это взялись, нужно также центрировать `div`-контейнер в окне браузера.

4. Измените только что созданный в файле `styles.css` стиль `.pageWrapper`, чтобы он приобрел следующий вид:

```
.pageWrapper {
  width: 960px;
  margin: 0 auto;
}
```

Свойство `max-width` присуще резиновому макету, но только до конкретного предела. В данном случае, когда окно браузера шире 1200 пикселов, `div`-контейнер не будет увеличиваться. Свойству `margin` присвоено значение `0 auto`, обеспечивающее нулевые поля сверху и снизу и автоматические поля слева и справа. Последняя настройка (`auto`) позволяет браузеру автоматически определять размер полей путем равномерного деления пространства между левой и правой сторонами и центрируя таким образом `div`-контейнер в окне браузера. Теперь страница должна приобрести вид, показанный на рис. 13.15.

Рис. 13.15. Использование свойства max-width вместо width позволяет получить резиновый дизайн, помещающийся в окнах браузеров различной ширины, но при этом не приобретающий излишнюю ширину, затрудняющую чтение на широкоформатных мониторах с высоким разрешением

Резиновый/фиксированный макет

Страница выглядит вполне привлекательно, но могла бы выглядеть лучше, если бы черный фон для навигационной панели вверху и для колонтитула внизу, а также темно-фиолетовый градиент в баннере распространялись на ширину всей страницы (рис. 13.16). Поскольку навигационная панель, баннер и нижний колонтитул находятся внутри `div`-контейнера `pageWrapper`, эти фоновые цвета прекращают отображаться, когда окно браузера становится шире 1200 пикселов. Вместо этого нужно оставить часть страницы — элементы с фоновым цветом — резиновой, при этом ограничив ширину элементов с основным контентом.

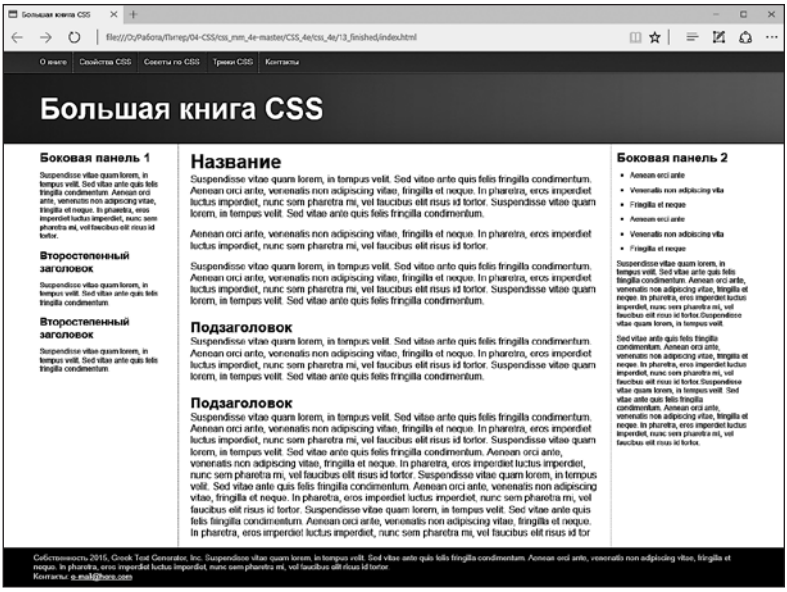

Рис. 13.16. Верстка многоколоночных макетов ничуть не сложнее создания обтекаемых HTML-элементов: три колонки, три обтекаемых элемента. Дополнительные колонки можно добавлять с помощью дополнительных HTML-контейнеров, выравнивая их по левому или по правому краю

Фоновый цвет для навигационной панели применяется к элементу `nav`, фиолетовый градиент — к элементу `header`, черный фон в нижнем колонтитуле — к элементу `footer`. Чтобы фон каждого из этих элементов распространился на всю ширину страницы, они не должны быть ограничены `div`-контейнером `pageWrapper`. Поэтому сначала нужно удалить этот стиль.

1. Удалите в файле `styles.css` недавно созданный стиль `.pageWrapper`.

   Контейнер `pageWrapper` в HTML-коде можно оставить. Этот дополнительный HTML-код не станет обузой и может понадобиться вам позже для применения других стилей.

   Страница вернется к своему полностью резиновому дизайну. Теперь нужно ограничить только навигационную панель, текст баннера, текст нижнего колонтитула

и основной контент, чтобы они не превышали по ширине 1200 пикселов. Для этого нужно немного углубиться в HTML-код и посмотреть, с какими элементами следует поработать.

Взгляните на HTML-код файла `index.html` и найдите элемент `nav`. Внутри этого элемента вы увидите кнопки навигации, созданные с помощью простого неупорядоченного списка. Именно это нам и нужно. Для списка можно установить максимальную ширину 1200 пикселов и центрировать его на странице, позволив элементу `nav` (и его черному фону) распространиться на всю ширину окна браузера. То же самое касается текста баннера внутри элемента `h1`. Для него также можно установить максимальную ширину и поля. В нижнем колонтитуле можно взять под контроль элемент `p`.

2. Добавьте в файл `styles.css` следующий код:

```
nav ul, header h1, footer p {
  max-width: 1200px;
  margin: 0 auto;
}
```

Этот групповой селектор нацелен на панель навигации, баннер и нижний колонтитул, но не на те элементы, в которых они содержатся. Теперь, если сохранить CSS-файл и просмотреть в браузере страницу `index.html`, вы увидите что элементы навигации, название и нижний колонтитул не становятся шире 1200 пикселов. Но основной контент по-прежнему заполняет все пространство окна браузера. Чтобы исправить ситуацию, нужно заключить три колонки в еще один `div`-контейнер для создания группы, размером и выравниванием которой можно управлять.

ПРИМЕЧАНИЕ

Другой способ заключается в том, чтобы добавить контейнер div в элемент header и обернуть им элементы nav и h1, а второй контейнер div вставить в элемент footer. Можно не ограничивать ширину элементов header и footer, ограничивая при этом ширину вложенных div-контейнеров.

3. Откройте в редакторе HTML-кода файл `index.html`. Непосредственно перед комментарием `<!-- первая боковая панель -->` добавьте код `<div class="contentWrapper">`:

```
<div class="contentWrapper">
<!-- первая боковая панель -->
```

Теперь этот `div`-контейнер нужно закрыть.

4. Перейдите в конец HTML-файла. Между закрывающим тегом `</aside>` и открывающим тегом `<footer>` добавьте закрывающий тег `</div>`, чтобы HTML-код приобрел следующий вид:

```
</aside>
</div>
<footer>
```

И наконец, можно добавить это новое имя класса к стилю, созданному на шаге 2.

5. Добавьте новый класс к групповому селектору в файле styles.css:

```
nav ul, header h1, footer p, .contentWrapper {
  max-width: 1200px;
  margin: 0 auto;
}
```

Вид завершенной страницы показан на рис. 13.16. Полная версия этого урока находится в папке 13_finished.