

# 3 Селекторы: выбор форматируемых элементов

Каждый CSS-стиль состоит из двух частей: селектора и блока объявления (о нем говорилось в предыдущей главе), который, в свою очередь, содержит правила форматирования — свойства цвета, размера шрифта текста и т. д. Они относятся лишь к оформлению. Возможность фокусировки каскадных таблиц стилей на отдельных элементах заключается как раз в самых первых символах, начинающих определение любого стиля, — селекторах. Именно селектор контролирует дизайн веб-страницы (рис. 3.1), *определяя* элемент, который вы хотите изменить. Другими словами, он используется для форматирования множества элементов одновременно. Если дать *более* подробное описание, то селекторы позволяют выбрать один или несколько схожих элементов для их последующего изменения с помощью свойств в блоке объявления. Селекторы CSS — большой потенциал для создания дизайна веб-страниц.

```
h1 {  
    font-family: Arial, sans-serif;  
    color: #CCCCFF;  
}
```

**Рис. 3.1.** Здесь первая часть стиля — селектор — определяет элементы, подлежащие форматированию. В данном случае h1 означает «все заголовки первого уровня (элементы h1) веб-страницы»

---

## ПРИМЕЧАНИЕ

Если вы хотите немного попрактиковаться на примерах, прежде чем изучать теоретический материал по теме селекторов, обратитесь к практикуму в конце этой главы.

---

## Селекторы тегов

Селекторы, которые используют для применения стилей к определенным HTML-элементам, называются *селекторами тегов* (или *селекторами элементов*). Они

представляют собой весьма эффективное средство проектирования дизайна веб-страниц, поскольку определяют стиль всех экземпляров конкретного HTML-элемента. С их помощью можно быстро изменять дизайн веб-страницы с небольшими усилиями. Например, если надо отформатировать все абзацы текста, используя шрифт одного начертания, размера, а также цвета, то вам просто нужно создать стиль с селектором `p` (применительно к элементу `p`). Он определяет, каким образом браузер отобразит отдельно взятый элемент (в данном случае `p`).

До появления CSS, чтобы отформатировать текст, вам пришлось бы заключить его по всем абзацам в элемент `font` многократно. Этот процесс занял бы много времени, не говоря о том, что код HTML-страниц при этом увеличится в объеме до невероятных размеров, страницы будут загружаться медленнее, их обновление будет занимать много времени. С селекторами тегов вам фактически ничего не нужно делать с HTML-кодом, вы создаете CSS-стили и позволяете браузеру сделать все остальное.

Селекторы исключительно просто определить в CSS-стилях, так как они наследуют имя формируемых элементов — `p`, `h1`, `table`, `img` и т. д. Например, на странице, представленной на рис. 3.2, селектор `h2` (*вверху*) определяет представление каждого заголовка второго уровня страницы (*внизу*), которых в данном случае три (см. метки в левой части окна браузера).

```
h2 {  
  font-family:"Century Gothic", "Gill Sans", sans-serif;  
  color:#000000;  
  margin-bottom:0;  
}
```

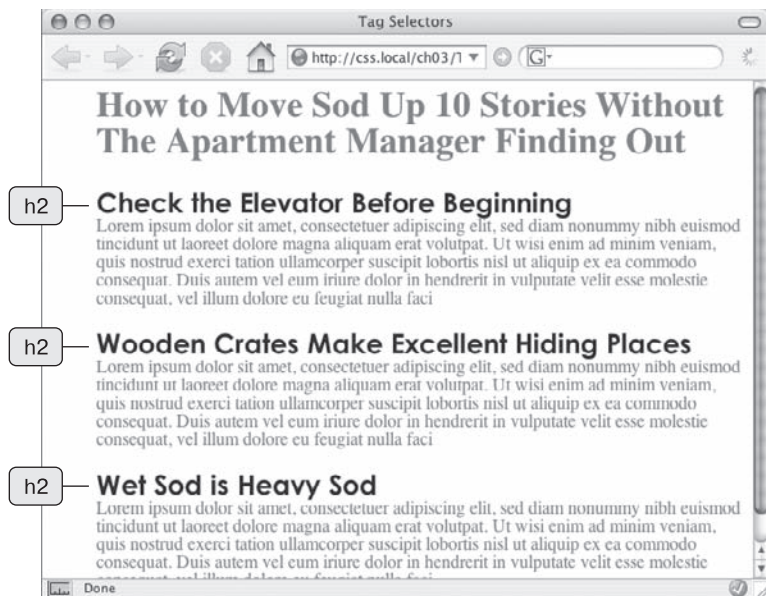


Рис. 3.2. Селектор тега воздействует на все экземпляры указанного элемента веб-страницы

## ПРИМЕЧАНИЕ

Как видно из рис. 3.2, селекторы тегов указываются без символов < и >, которые окружают теги HTML-элементов. Поэтому при написании правила, например, для тега <p> просто вводите его имя: p.

Однако и здесь есть свои недостатки. Как сделать так, чтобы не все абзацы веб-страницы выглядели одинаково? Селекторами тегов этого добиться не удастся, потому что они не предоставляют достаточную информацию браузеру. Например, нужно задать различия между элементом p, выделенным определенным цветом и кеглем, и элементом p, для которого вы хотите оставить шрифт черного цвета. Каскадные таблицы стилей предоставляют сразу несколько способов решения данной проблемы, самый простой из которых — использование *классов (селекторов классов)*.

## Классы: точное управление

Если вы хотите, чтобы какие-то элементы выглядели не так, как другие родственные им элементы на той же веб-странице, например, хотите задать для одного или двух рисунков красную рамку, оставив все остальные изображения без нее, то можете использовать селектор классов. Если вы привыкли работать со стилями в текстовых редакторах, таких как Microsoft Word, то классы покажутся вам хорошо знакомыми. Вы создаете селектор, назначая ему имя, а затем применяете его лишь к тем HTML-элементам, которые хотите отформатировать. Например, вы можете создать класс .copyright и с его помощью выделить абзац, содержащий информацию об авторских правах, не затрагивая остальные абзацы.

Классы позволяют указать конкретный элемент веб-страницы, независимо от тегов. Предположим, вы хотите отформатировать одно или несколько слов абзаца. В данном случае применяется форматирование не ко всему элементу p, а лишь к фрагменту абзаца. Таким образом, вам нужно использовать класс для выделения определенного текста. Можно применить изменения к множеству элементов, окруженных различными HTML-тегами.

Например, вы можете придать какому-то абзацу и заголовку второго уровня (элемент h2) одинаковый стиль, как показано на рис. 3.3. В отличие от селекторов тегов, которые ограничивают вас существующими на веб-странице HTML-элементами, с помощью этого метода вы можете создать любое количество классов и поместить их в выбранные позиции.

Посмотрите на рис. 3.3. Вы можете отформатировать один экземпляр заголовка h2 (с текстом Wet Sod is Heavy Sod). Класс .special сообщает браузеру о необходимости применения стиля к единственному элементу h2. Создав его один раз, вы можете пользоваться им и в дальнейшем для любых элементов. В примере такой стиль применен к верхнему абзацу.

## ПРИМЕЧАНИЕ

Если вы хотите применить класс всего к нескольким словам текста, содержащегося в произвольном элементе HTML-кода (подобно среднему абзацу на рис. 3.3), используйте элемент span. Для более детального ознакомления с ним смотрите врезку «В курс дела!» далее в этой главе.



**Рис. 3.3.** Классы позволяют целенаправленно изменять дизайн фрагментов веб-страниц

Вы, вероятно, обратили внимание на точку, с которой начинается имя каждого класса: `.copyright`, `.special`. Это одно из правил, которые необходимо иметь в виду при именовании классов.

- **Все имена классов должны начинаться с точки.** С ее помощью браузеры находят классы в каскадной таблице стилей.
- **При именовании классов разрешается использование только латинских букв, цифр, дефисов и знаков подчеркивания.**
- **Имя после точки всегда должно начинаться с латинской буквы.** Например, `.9lives` — неправильное имя класса, а `.crazy8` — правильное. Можно называть классы, скажем, именами `.copy-right` и `.banner_image`, но не `.-bad` или `._as_bad`.

- **Имена классов чувствительны к регистру.** Например, `.SIDEBAR` и `.sidebar` рассматриваются языком CSS как различные классы.

Классы создаются так же, как селекторы тегов. После имени указывается блок объявления, содержащий все необходимые свойства:

```
.special {  
  color:#FF0000;  
  font-family:"Monotype Corsiva";  
}
```

Поскольку селекторы тегов распространяются на все типы веб-страницы, их достаточно определить в таблице стилей и они начинают работать. Форматируемые HTML-элементы уже присутствуют в коде веб-страницы. Чтобы воспользоваться преимуществами, которые обеспечивают классы, требуется выполнить еще несколько действий, ведь использование классов — двухэтапный процесс. После того как вы создали класс, надо указать, где вы хотите его применить. Для этого добавьте атрибут `class` к HTML-элементу, который следует отформатировать.

Допустим, вы создали класс `.special` с целью выделения определенных элементов веб-страницы. Чтобы применить этот стиль к абзацу, добавьте атрибут `class` в тег `<p>`, как показано ниже:

```
<p class="special">
```

---

#### ПРИМЕЧАНИЕ

Вы не должны указывать точку перед именем класса в HTML-коде (в атрибуте `class`). Она требуется только в имени селектора таблицы стилей.

---

Таким образом, когда браузер сталкивается с этим элементом, он знает, что правила форматирования, содержащиеся в стиле `.special`, необходимо применить к данному тексту. Вы можете также использовать класс только в части абзаца или заголовка, добавив элемент `span`. Например, чтобы выделить только одно слово в абзаце, используя стиль `.special`, можно написать следующее:

```
<p>Добро пожаловать в ресторан <span class="special">Медуза</span>,  
    который удивит вас изысками морской кухни.</p>
```

Создав класс, можно применить его практически к любому элементу веб-страницы. Вообще, вы можете применять один и тот же класс к разным элементам, создав, к примеру, стиль `.special` с особым шрифтом и цветом для элементов `h2`, `p` и `u1`.

**Один элемент, несколько классов.** Вы можете применять не только один и тот же класс к различным элементам, но и несколько классов к одному элементу. Хотя сначала может показаться, что создание нескольких классов и добавление их имен к одному элементу — лишняя работа, это лишь общий подход.

Приведу примеры, когда бы вы могли применить несколько классов к одному элементу: представьте себе, что вы разрабатываете интерфейс для управления корзиной покупок посетителя сайта. Интерфейс требует разнообразных кнопок, каждая из которых выполняет какое-либо действие. Одна кнопка может быть использована для удаления товара из корзины, другая позволяет добавлять товар в нее, а третья служит для изменения количества товаров.

Будучи педантичным дизайнером, вы хотите, чтобы кнопки не только обладали некоторым сходством, например имели закругленные углы и одинаковый шрифт, но и различались: кнопка для удаления товара из корзины была красной, а кнопки для добавления — зелеными и т. д. Для достижения согласованности дизайна вы можете создать два класса. Один класс будет применяться ко всем кнопкам, а второй — к их определенным типам.

Для начала создадим класс `.btn` (сокращение от английского слова `button` — «кнопка»):

```
.btn {  
  border-radius: 5px;  
  font-family: Arial, Helvetica, serif;  
  font-size: .8 em;  
}
```

Далее необходимо создать дополнительные классы для каждого типа кнопок:

```
.delete {  
  background-color: red;  
}  
.add {  
  background-color: green;  
}  
.edit {  
  background-color: grey;  
}
```

Затем, применяя несколько классов к элементу, вы можете комбинировать стили, тем самым соблюдая согласованность между кнопками, но сохраняя уникальный вид каждого типа.

```
<button class="btn add">Добавить</button>  
<button class="btn delete">Удалить</button>  
<button class="btn edit">Изменить</button>
```

Браузеры и язык HTML не испытывают никаких проблем в случаях, когда к одному элементу применено несколько классов. Все, что нужно сделать, — добавить атрибут `class` к HTML-элементу и в качестве его значения — имя каждого класса, разделяя их запятой. Браузер объединит свойства из различных классов, применив законченные стили к элементам. В данном примере для текста каждой из кнопок будет использоваться шрифт Arial размером 0,8 em, а углы самих кнопок будут закруглены. Тем не менее цвета каждой из кнопок будут различны: кнопка **Добавить** — зеленого цвета, кнопка **Удалить** — красного, а кнопка **Изменить** — серого.

Преимущество этого подхода заключается в том, что, если вы решите, что углы кнопок больше не должны быть закруглены или что для текста на кнопках должен использоваться другой шрифт, вам нужно лишь изменить стиль `.btn`, чтобы обновить внешний вид всех кнопок. С другой стороны, если вы решите, что кнопка **Изменить** должна быть желтой, а не серой, изменение стиля `.edit` повлияет только на нее и не затронет другие кнопки.

## Идентификаторы: отдельные элементы веб-страницы

В языке CSS идентификаторы предназначены для *идентификации* уникальных частей веб-страниц, таких как шапка, панель навигации, область основного контента и т. д. Как и при использовании классов, вы создаете идентификатор (ID-селектор), придумав ему имя, а затем применяете его к HTML-коду своей веб-страницы. Так в чем же различие? Как описано во врезке «В курс дела!», идентификаторы имеют дополнительное специфическое применение. Например, в веб-страницах с использованием JavaScript-сценариев или в очень объемных страницах. Другими словами, существует несколько вынужденных причин для использования идентификаторов.

### ПРИМЕЧАНИЕ

В сообществе веб-разработчиков наметился тренд к отказу от идентификаторов в каскадных таблицах стилей. Причины этого явления требуют более глубокого знания языка CSS, чем то, которое может быть получено при чтении данной книги на текущий момент, поэтому здесь идентификаторы будут встречаться редко, а все причины того, почему применение идентификаторов нельзя признать удачным решением, будут изложены позже.

### В КУРС ДЕЛА!

#### HTML-элементы `div` и `span`

В главе 1 были кратко описаны `div` и `span` — два универсальных HTML-элемента, которые можно использовать в любых ситуациях. Когда нет HTML-элемента, который бы однозначно указывал, где следует разместить класс или идентификатор форматирования, для заполнения промежутков используйте `div` и `span`.

Логическое *деление* страницы на такие фрагменты, как шапка, панель навигации, боковые панели и колонтитул, обеспечивает элемент `div`. Вы также можете его использовать, чтобы охватить любой фрагмент, включающий несколько последовательных элементов веб-страницы, в том числе заголовки, маркированные списки, абзацы (программисты называют эти элементы *блочными*, потому что они формируют логически законченный блок контента с разрывами строк до и после такого блока). Элемент `div` функционирует как элемент абзаца: указываем открывающий тег `<div>`, далее добавляем некоторый текст, рисунок или какой-либо другой контент, а затем закрываем код тегом `</div>`.

Элемент `div` имеет уникальную способность включать в себя *несколько* блочных элементов, являясь средством группировки (к примеру, логотип и панель навигации или блок новостей, составляющий боковую панель). После

группировки содержимого веб-страницы таким образом вы можете применить определенное форматирование исключительно к элементам, находящимся внутри этого фрагмента `div`, или переместить (позиционировать) весь отмеченный фрагмент содержимого в конкретное место веб-страницы, например в правую часть окна браузера. О компоновке макетов средствами каскадных таблиц стилей рассказывается в части III книги.

Например, вы добавили на веб-страницу фотографию, у которой есть подрисовочная подпись. Вы можете для группировки обернуть оба объекта в элемент `div` (с применением к нему класса):

```
<div class="photo">

<p>Я, мама и папа в отпуске в Антарктике.</p>
</div>
```

В зависимости от того, как вы объявите стиль, класс `.photo` может добавить декоративную рамку, цвет фона и прочее сразу ко всему блоку, включающему и фотографию, и подпись к ней. В части III описываются еще более мощные способы применения `div`, включая вложенные конструкции из этих элементов.



## В КУРС ДЕЛА!

Последние версии языка HTML содержат множество блочных элементов, которые работают схожим с `div` образом, но предназначены для контента определенного типа. Например, для отображения картинки или подрисовочной подписи вместо элемента `div` вы можете использовать элемент `figure`. Тем не менее, поскольку Internet Explorer 8 не поддерживает HTML5-элементы (см. врезку «Как заставить браузер Internet Explorer 8 поддерживать HTML5-элементы» в главе 1), для группировки нескольких HTML-элементов в одно целое многие дизайнеры до сих пор используют элементы `div`.

Кроме того, новые HTML5-элементы предназначены добавлять «смысл» вашему HTML-коду. Например, элемент `article` используется для автономных блоков текста, таких как статья в журнале. Тем не менее не вся разметка имеет смысл, поэтому в своей рабо-

те для простой группировки элементов вы по-прежнему будете прибегать к элементам `div`.

С другой стороны, элемент `span` позволяет применять классы и идентификаторы к фрагменту — *части* HTML-элемента. С его помощью вы можете выхватить из абзаца отдельные слова и фразы (которые часто называются *строчными* элементами), чтобы форматировать их отдельно друг от друга.

В данном примере класс форматирования с именем `.companyName` **форматирует строчные элементы** "Cosmo-Farmer.com", "Disney" и "ESPN":

```
<p>Добро пожаловать на сайт <span class="companyName">CosmoFarmer.com</span>, дочерней компании таких корпораций, как <span class="companyName">Disney</span> и <span class="companyName">ESPN</span>...шутка.</p>
```

Хотя многие веб-разработчики уже не пользуются идентификаторами так часто, как раньше, все же следует знать, что они собой представляют и как работают. Применение идентификаторов не представляет сложности. Если в классах перед именем указывается точка (`.`), то в случае с идентификаторами сначала должен быть указан символ решетки (`#`). Во всем остальном руководствуйтесь теми же правилами, что и для классов (см. выше). Следующий пример устанавливает цвет фона, ширину и высоту элемента:

```
#banner {
  background: #CC0000;
  height: 300px;
  width: 720px;
}
```

Применение идентификаторов в HTML схоже с использованием классов, но требует другого атрибута с соответствующим именем — `id`. Например, для применения показанного выше стиля к элементу `div` нужно написать такой HTML-код:

```
<div id="banner">
```

Точно так же для указания того, что последний абзац страницы — единственный с информацией об авторских правах, вы можете создать идентификатор с именем `#copyright` и применить его к тегу этого абзаца:

```
<p id="copyright">
```

## ПРИМЕЧАНИЕ

Символ решетки (`#`) используется только при описании стиля в таблице. При вставке же имени идентификатора в HTML-тег символ `#` указывать не нужно: `<div id="banner">`.



**ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ****Специальное применение идентификаторов**

У идентификаторов имеется несколько преимуществ перед классами. Эти особенности фактически не имеют никакого отношения к каскадным таблицам стилей, поэтому могут вам и не понадобиться. Но если вам это интересно, приведу их.

- Одним из простых для JavaScript-программистов способов определения частей страницы и манипуляции ими является применение идентификатора к элементу страницы с последующим использованием JavaScript-сценариев со ссылками на этот идентификатор. Специалисты часто применяют идентификаторы к заполняемым элементам форм (например, к текстовым полям) для получения имени посетителя сайта и т. д. Это позволяет JavaScript получить доступ к полям форм и совершать с ними разные манипуляции, например, когда посетитель нажимает кнопку отправки формы, убеждаться в том, что поля заполнены.
- Идентификаторы также дают возможность создавать ссылки на определенные части веб-страниц,

при этом быстро перемещаясь по ним. Если у вас есть алфавитный словарь терминов, можно использовать идентификатор для создания указателя по буквам алфавита. При щелчке кнопкой мыши на букве «З» посетители сразу же переходят к словам, начинающимся с этой буквы. Для этого вам не придется использовать CSS — достаточно и HTML.

Сначала добавьте атрибут `id` в позицию на странице, на которую вы хотите ссылаться. Например, в указателе вы можете добавить элемент `h2` с буквой из алфавита. Добавьте соответствующий идентификатор к каждому элементу `h2`: `<h2 id="Z">З</h2>`. Чтобы создать ссылку в HTML, добавьте символ `#` и имя идентификатора в конец URL-адреса — `index.html#Z`. Эта ссылка указывает непосредственно на элемент с идентификатором `Z` на странице `index.html` (использование идентификатора таким способом оказывает действие, аналогичное применению элемента привязки `a` в языке HTML: `<a name="Z">З</a>`).

## Форматирование групп элементов

Иногда необходимо быстро применить одинаковое форматирование сразу к нескольким различным элементам веб-страницы. Например, нужно, чтобы все заголовки имели один и тот же цвет и шрифт. Создание отдельного стиля для каждого заголовка определенного уровня — слишком объемная работа. А если вы потом захотите изменить цвет всех заголовков одновременно, то придется все и обновлять. Для решения этой задачи лучше использовать *групповой селектор*, который позволяет применить стиль, описав его лишь один раз, одновременно ко всем элементам.

## Создание групповых селекторов

Для работы с групповым селектором создайте список, в котором один селектор отделен от другого запятыми. Таким образом, получаем:

```
h1, h2, h3, h4, h5, h6 { color: #F1CD33; }
```

Здесь приведены только селекторы тегов, но можно использовать любые типы селекторов (и их сочетания). Например, стиль группового селектора, который определяет одинаковый цвет шрифта для элементов `h1`, `p` и любых других,

принадлежащих классу `.copyright`, а также для элемента с идентификатором `#banner`, выглядит так:

```
h1, p, .copyright, #banner { color: #F1CD33; }
```

#### ПРИМЕЧАНИЕ

Если вам нужно применить к нескольким элементам веб-страницы схожие и в то же время несколько различающиеся стили, то можно создать групповой селектор с общими командами и отдельные стили с уникальным форматированием для каждого индивидуального элемента. Другими словами, два (или больше) стиля могут форматировать один и тот же элемент. Это и является мощной особенностью языка CSS (по этой теме см. главу 5).

## Универсальный селектор

Групповой селектор можно рассматривать как подручное средство для применения одинаковых свойств различным элементам. Каскадные таблицы стилей предоставляют *универсальный селектор* `*` для выборки всех элементов веб-страницы вместо указания каждого тега *по отдельности*. Например, если вы хотите, чтобы все отобразилось полужирным шрифтом, нужно добавить следующий код:

```
a, p, img, h1, h2, h3, h4, h5 ...и т. д... { font-weight: bold; }
```

Использование символа `*` — более быстрый способ сообщить CSS о выборке всех HTML-элементов веб-страницы:

```
* { font-weight: bold; }
```

Кроме того, вы можете использовать универсальный селектор в составе селектора потомков (также называемого вложенным селектором): применяете стиль ко всем элементам-потомкам, подчиненным определенному элементу веб-страницы. Например, `.banner *` выбирает все элементы внутри элемента, имеющего атрибут `class` со значением `.banner` (более подробно о селекторах потомков вы узнаете чуть позже).

Универсальный селектор не определяет тип элементов, поэтому трудно описать его воздействие на HTML-элементы сайта. По этой причине дизайнеры со стажем используют для форматирования различных элементов такую особенность языка CSS, как *наследование*. Она описана в следующей главе.

Тем не менее некоторые веб-дизайнеры используют универсальный селектор как способ очистки *всего* пространства (воздуха) вокруг блочных элементов. Как вы увидите позже, с помощью свойства `margin` можно добавить пространство вокруг элемента, а свойства `padding` — пространство между границей элемента и его содержимым. Для разных элементов браузеры автоматически добавляют воздух различного размера, поэтому один из способов начать с чистого листа заключается в том, чтобы удалить все пространство вокруг элементов с определенным стилем:

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

## Форматирование вложенных элементов

Выбор типа селектора в каждом случае обусловлен конкретной целью. Селекторы тегов можно быстро и просто создать, но они придают всем экземплярам форматируемого элемента одинаковый внешний вид. Это бывает необходимо, например, чтобы все заголовки второго уровня вашей веб-страницы выглядели одинаково. Классы и идентификаторы обеспечивают большую гибкость независимого форматирования отдельно взятых элементов страницы, но создание стилей для классов или идентификаторов также требует добавления атрибута `class` или `id` с соответствующим значением к HTML-элементам, которые нужно отформатировать. Это не только усложняет работу, но и увеличивает объем кода HTML-файла. Все, что нужно в этом случае, — объединить простоту использования селекторов тегов с точностью классов и идентификаторов. И такое средство в CSS есть — это *селекторы потомков* (также называемые *вложенными селекторами*).

Они используются для того, чтобы согласованно отформатировать целый набор элементов, но только когда те расположены в определенном фрагменте веб-страницы. Их работу можно описать так: «Эй вы, элементы привязки на панели навигации, прислушайтесь. У меня есть для вас указания по форматированию. А все остальные элементы привязки, проходите мимо, вам здесь нечего смотреть».

Селекторы потомков позволяют форматировать элементы в зависимости от их связей с другими элементами. Чтобы разобраться, как они работают, придется немного покопаться в языке HTML. Понимание работы селекторов потомков позволит получить представление о функционировании других типов селекторов, работа которых описана далее.

---

### ПРИМЕЧАНИЕ

Селекторы потомков на первый взгляд могут показаться немного сложными, однако это одна из наиболее важных технологий для эффективного и тонкого применения CSS. Запаситесь терпением, и вы скоро освоите их.

---

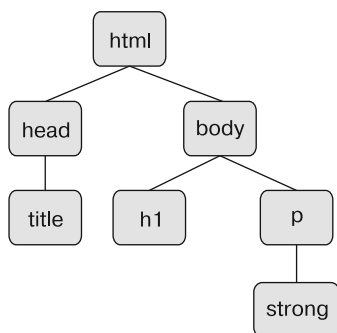
## Дерево HTML

HTML-код, на котором написана любая веб-страница, похож на генеалогическое дерево. Первый используемый HTML-элемент — `html` — похож на главного прародителя всех остальных. Из него выходят `head` и `body`, следовательно, `html` является *предком* названных элементов.

Элемент, расположенный внутри другого, — его *потомок*. Элемент `title` в следующем примере — потомок `head`:

```
<html>
  <head>
    <title>Простой документ</title>
  </head>
  <body>
    <h1>Заголовок</h1>
    <p>Абзац с <strong>важным </strong>текстом.</p>
  </body>
</html>
```

Представленный выше HTML-код можно изобразить в виде схемы (рис. 3.4). Здесь показаны отношения между элементами веб-страницы. Сначала располагается `html`; от него отходят два раздела, представленные `head` и `body`. Они содержат и другие, которые, в свою очередь, могут включать еще элементы. Таким образом, рассматривая, какие из них являются вложенными, можно схематически изобразить любую веб-страницу.



**Рис. 3.4.** HTML-код состоит из вложенных элементов. Связь между элементами, отображающая способ вложения одного в другой, представляет своего рода генеалогическое дерево

Схемы в форме дерева помогают выяснить и проследить, как CSS «видит» взаимодействие элементов веб-страницы. Функционирование многих селекторов, описанных в настоящей главе, включая селектор потомков, основывается на их родственных отношениях. Рассмотрим самые важные из них.

- **Предок.** Как описано в начале раздела, HTML-элемент, который заключает в себе другие элементы, — это предок. На рис. 3.4 это `html`, в то время как `body` — предок для всех содержащихся в нем элементов: `h1`, `p` и `strong`.
- **Потомок.** Элемент, расположенный внутри одного или более типов, — потомок. На рис. 3.4 `body` — потомок `html`, в то время как `p` — потомок одновременно и для `body`, и для `html`.
- **Родительский элемент.** Он связан с другими элементами *более низкого* уровня и находится выше в дереве. На рис. 3.4 `html` является родительским только для `head` и `body`. Элемент `p` — родительский по отношению к `strong`.
- **Дочерний элемент.** Элемент, непосредственно подчиненный другому элементу более высокого уровня, является дочерним. На рис. 3.4 оба элемента — `h1` и `p` — дочерние по отношению к `body`, но `strong` не является дочерним для `body`, так как он расположен непосредственно внутри элемента `p`, он является дочерним для этого элемента.
- **Родственный элемент.** Элементы, являющиеся дочерними для одного и того же родительского элемента, называются *родственными*, как братья или сестры. На рис. 3.4 они расположены рядом друг с другом. Элементы `head` и `body` — элементы одного уровня, как и `h1` и `p`.

На этом в CSS «родственные отношения» заканчиваются.

## Создание селекторов потомков

Селекторы потомков позволяют использовать дерево HTML, форматируя элементы по-разному, в зависимости от того, где они расположены. Например, на веб-странице имеется элемент `h1` и вы хотите выделить слово внутри этого заголовка с помощью элемента `strong`. Проблема в том, что большинство браузеров отобразит весь заголовок полужирным шрифтом, поэтому посетитель страницы не сможет увидеть разницы между выделенным словом и другими частями заголовка. Создание селектора тега — не очень хорошее решение: так вы измените цвет *всех* вхождений элемента `strong` веб-страницы. Селектор же потомков позволит вам изменить цвет элемента *только в том случае*, если он расположен внутри заголовка первого уровня.

Решение проблемы выглядит следующим образом:

```
h1 strong { color: red; }
```

В данном случае *любой* текст, окруженный тегами элемента `strong`, находящегося внутри элемента `h1`, будет выделен красным цветом, но на другие экземпляры элемента `strong` на веб-странице этот стиль не повлияет. Вы могли добиться того же результата, создав класс, например `.StrongHeader`. Но в таком случае понадобилось бы вносить изменения в HTML, добавляя новый класс к элементу `strong` внутри заголовка. Подход, основанный на селекторах потомков, позволяет обойтись без лишней работы при создании стилей.

Селекторы потомков форматируют вложенные элементы веб-страницы, следуя тем же правилам, которым подчиняются элементы-предки и элементы-потомки в дереве HTML.

Вы создаете селектор потомков, объединяя селекторы вместе (согласно ветви дерева, которую нужно отформатировать), помещая самого старшего предка слева, а форматируемый элемент — справа. На рис. 3.5 все элементы веб-страницы — потомки элемента `html`. Элемент может происходить от множества других. Например, первый элемент `a` диаграммы является потомком `strong`, `p`, `body` и `html`. Обратите внимание на три ссылки (a) — элементы маркированного списка и еще одну, расположенную внутри абзаца. Чтобы отформатировать их иначе, вы можете создать стиль с селектором потомков:

```
li a { font-family: Arial; }
```

Это правило гласит: «Нужно отформатировать все ссылки (a), которые расположены в элементах списка (li), используя шрифт Arial». Вообще, селекторы потомков могут включать более двух элементов. Ниже представлены правильные определения для элементов `a`, находящихся в маркированных списках (см. рис. 3.5):

```
ul li a
body li a
html li a
html body ul li a
```

Все четыре селектора, представленные выше, имеют один и тот же эффект, что лишний раз демонстрирует отсутствие необходимости полностью описывать всех предков элемента, который вы хотите отформатировать. Например, во втором примере (`body li a`) определение `ul` не требуется. Этот селектор выполняет свои функции,

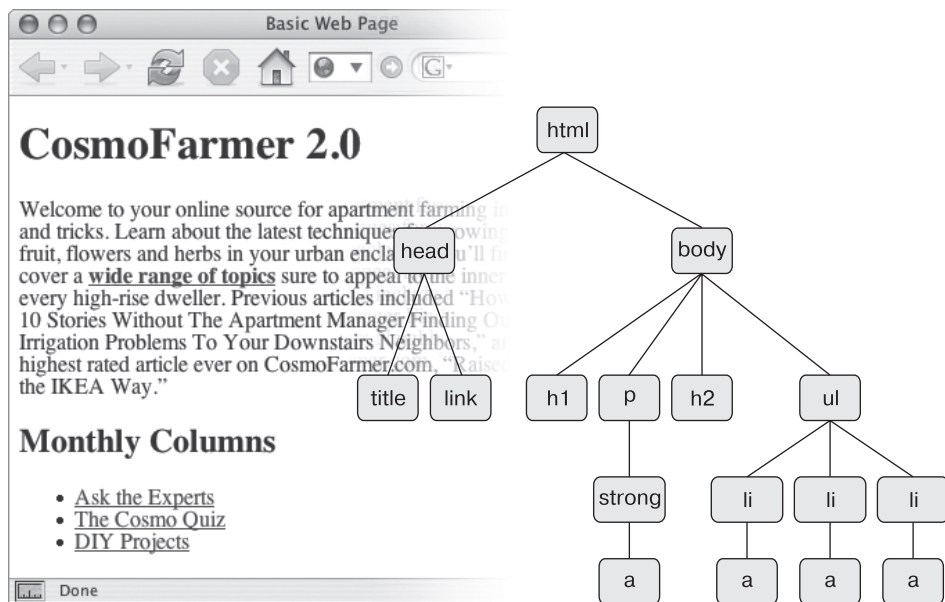


Рис. 3.5. Простейшая диаграмма в виде дерева, представляющая структуру веб-страницы

если есть `a` — потомок `li`. Его одинаково просто применить как к элементу `a`, расположенному внутри `em`, так и к `a` в `strong` или `li` и т. д.

В общем случае вы должны использовать кратчайший селектор потомка, который выполнит необходимое действие. Поскольку все элементы находятся внутри элементов `html` и `body`, включать их в селектор потомков нет причин.

#### ПРИМЕЧАНИЕ

Количество селекторов, включаемых вами в селектор потомков, влияет на то, как стиль взаимодействует с другими, конфликтующими стилями. Данное поведение называется специфичностью и описывается в главе 5.

Вы не ограничены использованием только селекторов тегов. Можно комбинировать различные типы селекторов в селекторе потомков. К примеру, нужно, чтобы ссылки были окрашены в желтый цвет, когда они находятся во вводимом элементе (который определен классом с именем `intro`). Этого можно добиться с помощью следующего селектора:

```
.intro a { color: yellow; }
```

Краткий перевод звучит так: «Нужно применить данный стиль ко всем ссылкам (`a`) — потомкам другого элемента, относящегося к классу с именем `intro`».

## Создание модулей

Селекторы потомков часто используются для форматирования *модуля* кода, то есть коллекции HTML-элементов, выполняющих на странице конкретную функцию. Предположим, что на странице присутствует `div`-контейнер, используемый для

вывода перечня корпоративных новостей. HTML-код может иметь следующий вид:

```
<div>
  <h2>Наша компания великолепна!</h2>
  <p>Дополнительная информация о достижениях нашей компании</p>
  <h2>Другая новость</h2>
  <p>Текст другой новости...</p>
  <h2>...и т. д...</h2>
  <p>...и т. д... </p>
</div>
```

Если вставить атрибут `class` в открывающий тег `<div>` — `<div class="news">`, вы можете создать селекторы потомков, по-разному форматирующие HTML-элементы внутри раздела новостей. Например:

```
.news h2 { color: red; }
.news p { color: blue; }
```

Теперь содержимое элементов `h2` внутри раздела новостей будет отформатировано красным цветом шрифта, а абзацев — синим. Можно даже создать (что часто и делается) селекторы потомков с несколькими именами классов. Предположим, что вы создаете страницу, предоставляющую каталог адресов сотрудников какой-нибудь организации. Вы можете заключить каждую контактную информацию в ее собственный `div`-контейнер, а затем улучшить внешний вид элементов внутри этого контейнера с помощью атрибута `class`:

```
<div class="contact">
  <p class="name">Василий Теркин</p>
  <p class="phone">495-555-1234</p>
  <p class="address">Советская, 5</p>
</div>
```

Затем можно создать несколько селекторов потомков для назначения стиля только этим элементам контактной информации:

```
.contact .name { font-weight: bold; }
.contact .phone { color: blue; }
.contact .address { color: red; }
```

---

#### ПРИМЕЧАНИЕ

Иногда в таблице стилей можно увидеть следующий код:

```
p.intro
```

Может показаться, что это похоже на селектор потомка, поскольку в нем присутствует как имя HTML-элемента, так и имя класса, но это не так. Между `p` и `.intro` нет пробела, значит, чтобы этот стиль работал, класс `intro` должен быть применен конкретно к элементу `p` (`<p class="intro">`). Если добавить пробел, вы получите другой эффект:

```
p.intro { color: yellow; }
```

Это несколько иной вариант, выбирающий любой элемент, форматируемый под класс `.intro`, который, в свою очередь, является потомком элемента `p`. Иными словами, он выбирает не абзац, а другой элемент внутри абзаца. В общем, чтобы максимально сохранить гибкость стилей вашего класса, лучше оставить в покое HTML-элемент (иными словами, использовать только `.intro` вместо `p.intro`).

---



## Псевдоклассы и псевдоэлементы

Иногда требуется выбрать фрагмент веб-страницы, в котором вообще нет элементов, но в то же время его достаточно просто идентифицировать. Это может быть первая строка абзаца или ссылка, на которую наведен указатель мыши. Каскадные таблицы стилей предоставляют для этих целей псевдоклассы и псевдоэлементы.

### Форматирование ссылок

Ниже представлены четыре псевдокласса, которые позволяют форматировать ссылки в зависимости от того, какое действие над ними выполняет посетитель веб-страницы.

- `a:link` — обозначает любую ссылку, по которой посетитель веб-страницы еще не переходил и на которую не наведен указатель мыши. Это обычный стиль непосещенных гиперссылок.
- `a:visited` — обозначает любую ссылку, по которой посетитель веб-страницы уже переходил. Она сохраняется в истории браузера. Можно создать для этого типа стиль, отличный от обычного, чтобы сообщить посетителю, что он уже посещал эту страницу. (Ограничения, связанные с этим селектором, будут рассмотрены во врезке в начале главы 9.)
- `a:hover` — позволяет изменять вид ссылки, на которую посетитель навел указатель мыши. Вы можете добавить визуальные эффекты трансформации (ролlover-эффект), которые служат для улучшения визуального восприятия, например, на кнопки панели навигации.

Кроме того, можно использовать псевдокласс `:hover` для применения стилей к элементам веб-страниц, отличным от ссылок. Например, вы можете применить его для выделения фрагмента текста, заключенного в теги элемента `p` или `div`, каким-либо эффектом форматирования в тот момент, когда посетитель веб-страницы перемещает указатель мыши поверх этого фрагмента. В этом случае вместо `a:hover` для добавления эффекта наведения указателя поверх ссылки вы можете создать стиль с именем `p:hover`, добавляющий эффект при наведении указателя мыши поверх абзаца. А если вы хотите добавить стиль к элементам с классом `highlight`, создайте стиль `highlight:hover`.

- `a:active` — позволяет определить, как будет выглядеть ссылка *во время* выбора ее посетителем веб-страницы. Другими словами, это стиль во время кратковременного щелчка кнопкой мыши.

В главе 9 описывается, как спроектировать дизайн ссылок при использовании этих селекторов для хорошего восприятия посетителями сайта (например, элементов навигации для перехода по разделам сайта и т. д.).

---

#### ПРИМЕЧАНИЕ

Если вы не собираетесь заниматься веб-дизайном профессионально, то не утруждайтесь чтением последующих разделов о селекторах. Многие специалисты вообще никогда ими не пользуются. Все, что вы уже изучили, — селекторы тегов, селекторы потомков, классы, идентификаторы и т. д. — по-

эволюит создавать красивые, функциональные, легко обновляемые, профессиональные с точки зрения дизайна сайты. Если вы готовы к практической части, то переходите сразу к практикуму данной главы. Вы можете закончить изучение теоретического материала позже в любое удобное для вас время.

---

## Форматирование фрагментов абзаца

На этапе становления Всемирной паутины не стоял вопрос типографического дизайна (в стиле глянцевого журналов) страниц и сайтов, никто и не думал о том, что текст должен выглядеть красиво. Теперь же это важно. В CSS для форматирования текста предусмотрено два псевдоэлемента — `:first-letter` и `:first-line`. Их использование придаст вашим веб-страницам изящное оформление, которым печатные издания обладают уже на протяжении многих лет.

Псевдоэлемент `:first-letter` позволяет создавать *буквицу* — начальный символ абзаца, который выделяется из остального текста, как в начале книжной главы.

Форматирование первой строки с помощью псевдоэлемента `:first-line` отличным от основного абзаца цветом притягивает посетителей веб-страницы изяществом оформления и простотой визуального восприятия содержимого сайта (в главе 6 вы узнаете все о форматировании текста, там же вы найдете подробное описание этих двух псевдоэлементов).

---

### ПРИМЕЧАНИЕ

В последней версии CSS синтаксис для псевдоэлементов претерпел изменения. В CSS 2.1 псевдоэлементы начинались с одинарного двоеточия:

`:first-letter`

В версии CSS3, чтобы отличить такие псевдоклассы, как `:hover`, от псевдоэлементов, добавлено еще одно двоеточие. Поэтому `:first-letter` и `:first-line` стали теперь выглядеть как `::first-letter` и `::first-line`. К счастью, чтобы сохранить поддержку более старых сайтов, браузеры продолжают поддерживать версию псевдоэлементов с одинарным двоеточием. Это хорошо, потому что Internet Explorer 8 не поддерживает синтаксис с двумя двоеточиями, поэтому пока можно остановиться на одинарном, поскольку все остальные браузеры используют их как и раньше.

---

## Дополнительные псевдоклассы и псевдоэлементы

В спецификации по языку CSS определены еще несколько мощных псевдоклассов и псевдоэлементов. Эти селекторы очень хорошо поддерживаются во всех браузерах, кроме устаревших.

### **:focus**

Псевдокласс `:focus` подобен `:hover`, с той лишь разницей, что `:hover` применяется, когда посетитель помещает указатель мыши поверх ссылки, а `:focus` — когда нажимает клавишу **Tab** или щелкает кнопкой мыши на текстовом поле (то есть требуется акцентировать внимание посетителя на конкретном (текущем) элементе веб-страницы). Щелчок на заполняемой форме программисты называют *фокусировкой*. Это единственный способ для дизайнера веб-страницы узнать, на чем сосредоточено внимание посетителя, с каким элементом страницы он имеет дело.

Селектор `:focus` полезен в основном для обеспечения обратной связи с посетителями сайта. Например, для смены цвета фона заполняемого поля формы, чтобы указать, где именно нужно вводить данные (текстовые поля, поля ввода пароля, текстовые области — везде можно использовать `:focus`). Этот стиль задает светло-желтый цвет любому текстовому полю, на котором посетитель щелкает кнопкой мыши или к которому переходит нажатием клавиши **Tab**:

```
input:focus { background-color: #FFFFCC; }
```

Селектор `:focus` задает эффект стиля только на время, пока элемент находится в фокусе. Если посетитель переходит к другому полю или в другую позицию веб-страницы, то свойство CSS прекращает форматировать элемент страницы.

---

#### СОВЕТ

Информация о поддержке браузерами селекторов приведена на сайте [caniuse.com](http://caniuse.com).

---

## :before

Псевдоэлемент `:before` выполняет такую функцию, которая не присуща ни одному другому селектору: он позволяет добавлять сообщение, предшествующее определенному элементу веб-страницы. Допустим, вы хотите поместить слово **ПОДСКАЗКА!** перед абзацами, чтобы визуальнo выделить их (по аналогии с названиями врезок в этой книге). Вместо многократного набора текста в HTML-коде вашей веб-страницы вы можете сделать это с помощью селектора `:before`. Кроме того, такое решение позволит уменьшить объем кода веб-страницы. Так, если вы решите изменить сообщение с **ПОДСКАЗКА!** на **ЭТО НУЖНО ЗНАТЬ!**, можно быстро отформатировать его на всех страницах сайта, один раз изменив текст в таблице стилей. Однако недостаток состоит в том, что сообщение невидимо для браузеров, которые не поддерживают CSS или псевдоэлемент `:before`.

Для работы псевдоэлемента нужно создать класс (скажем, с именем `.tip`) и применить его к абзацам, которым должно предшествовать данное сообщение, например `<p class="tip">`. Добавьте текст сообщения в таблицу стилей:

```
.tip:before {content: "ПОДСКАЗКА!" }
```

Всякий раз, когда браузеру встречается класс `.tip` внутри элемента `p`, он будет добавлять перед абзацем сообщение **ПОДСКАЗКА!**.

Текст, который добавляется этим селектором, называют *сгенерированным контентом*, поскольку браузер создает его на лету. В исходном коде HTML-страницы этой информации не содержится. Браузеры все время генерируют контент, например маркеры в маркированных списках или цифры в нумерованных. Для этого также можно использовать селектор `:before`.

Селектор `:before` поддерживается браузером Internet Explorer 8 и выше, а также другими основными браузерами (как и селектор `:after`).

## :after

Селектор `:before` добавляет сгенерированное содержимое перед определенным элементом, а `:after` — после. Например, им вы можете пользоваться для добавления заключительных кавычек (") после процитированного материала.

---

ПРИМЕЧАНИЕ

---

Псевдоэлементы `:before` и `:after` схожи с `:first-line` и `:first-letter`. Как уже упоминалось, в CSS3 к псевдоэлементам добавилось двойное двоеточие, поэтому `:before` и `:after` в CSS3 теперь называются как `::before` и `::after`. К счастью, браузеры поддерживают и более старую нотацию, поэтому вы можете продолжать использовать `:before` и `:after`, что добавляет преимуществ при работе в Internet Explorer 8.

---

## ::selection

Этот селектор, появившийся в CSS3, ссылается на элементы, которые посетитель выбрал на странице. Например, когда посетитель, нажав и удерживая кнопку мыши над текстом, перетаскивает указатель мыши, браузер выделяет этот текст и посетитель сможет скопировать выделенный фрагмент. Обычно браузеры добавляют за текстом синий фон. Internet Explorer изменяет при этом цвет шрифта текста на белый. Но вы можете управлять цветом фона и текста, указав описываемый селектор. Например, если нужно сделать выбранный текст белым на фиолетовом фоне, можно добавить в таблицу CSS следующий стиль:

```
::selection {
  color: #FFFFFF;
  background-color: #993366;
}
```

С помощью данного селектора можно установить лишь свойства `color` и `background-color`, поэтому вы не сможете изменить кегль, шрифт, поля и внести другие визуальные изменения.

---

ПРИМЕЧАНИЕ

---

Версии с одинарным двоеточием для псевдоэлемента `selection` не существует, поэтому вы должны использовать двойное двоеточие. Иными словами, `::selection` работает, а `:selection` — нет.

---

Этот селектор работает в браузерах Internet Explorer 9, Opera, Chrome и Safari, но не поддерживается в Internet Explorer 8 или Firefox. Однако вы можете добавить поддержку для Firefox, дополнив имя селектора так называемым *вендорным префиксом*:

```
::-moz-selection {
  color: #FFFFFF;
  background-color: #993366;
}
```

Для обеспечения работоспособности в Firefox и других браузерах в таблице стилей нужно указывать оба стиля, которые можно просто расположить друг за другом. (Подробности, касающиеся вендорных префиксов и необходимости их использования, изложены во врезке в главе 7.)

Если вы действительно хотите кого-нибудь поразить, можно указать другой цвет фона для текста, выделенного внутри конкретного элемента. Например, чтобы сделать красным на розовом фоне текст, который находится только внутри абзацев, нужно добавить селектор элемента `p` перед кодом `::selection`:

```
p::selection {  
  color: red;  
  background-color: pink;  
}
```

## Селекторы атрибутов

Каскадные таблицы стилей обеспечивают возможность форматирования элементов на основе выборки любых содержащихся в них атрибутов. Например, вы хотите выделить рамкой важные изображения веб-страницы, при этом не форматировать логотип, кнопки и другие небольшие изображения, в коде которых также присутствует элемент `img`. Вы должны понимать, что если у всех рисунков есть описания с атрибутом `title`, то это способствует использованию *селектора атрибутов* для выделения из массы изображений только нужных.

С помощью селекторов атрибутов вы можете выбрать элементы с конкретными свойствами. Вот пример, в котором выделены все элементы `img` с атрибутом `title`:

```
img[title]
```

Первая часть селектора — имя элемента (`img`); атрибут указывается далее в квадратных скобках: `[title]`.

Каскадные таблицы стилей не ограничивают использование селекторов атрибутов именами элементов: вы можете комбинировать их с классами. Например, селектор `.photo[title]` выбирает все элементы класса `.photo` с HTML-атрибутом `title`.

Если необходима более детальная выборка, то есть возможность найти элементы, которые имеют не только определенный атрибут, но и нужное значение. Например, если вы хотите подсветить ссылки, указывающие на определенный URL-адрес, создайте следующий селектор атрибута:

```
a[href="http://www.cafesoylentgreen.com"]{  
  color: green;  
  font-weight: bold;  
}
```

Уточнение селектора конкретным значением очень полезно при работе с заполняемыми формами. Многие элементы форм имеют одинаковые названия, даже если выглядят и функционируют по-разному. Будь то флажок, текстовое поле, кнопка отправки данных или какие-то другие элементы формы — все они содержат `input`. Значение атрибута `type` — вот что позволяет наделить тот или иной элемент формы соответствующими функциональными возможностями. Например, код `<input type="text">` создает текстовое поле, а `<input type="checkbox">` — флажок.

Чтобы выбрать только текстовые поля в форме веб-страницы, используйте следующее выражение:

```
input[type="text"]
```

Селектор атрибута очень разносторонний. Он не только позволяет находить элементы с определенным значением атрибута (например, все элементы формы со

значением `checkbox` атрибута `type`), но и выбирать элементы со значением атрибута, *начинающимся* с какого-либо значения, *заканчивающимся* им или *содержащим* его. Хотя эта возможность на первый взгляд может показаться излишней, на самом деле она достаточно полезная.

Представьте, что вы хотите создать стиль, который бы выделял внешние ссылки (ведущие за пределы вашего сайта), сообщая пользователю: «Ты покинешь этот сайт, если перейдешь по ссылке». Принимая во внимание то, что абсолютные ссылки внутри собственного сайта не используются, мы определяем, что любая внешняя ссылка будет начинаться со значения `http://` — первой части любой абсолютной ссылки.

Тогда селектор будет выглядеть так:

```
a[href^="http://"]
```

Символы `^=` означают «начинается на», так что вы можете использовать этот селектор для форматирования любой ссылки, начинающейся со значения `http://`. С его помощью вы легко форматируете ссылку, указывающую на `http://www.google.com` либо `http://www.piter.com`, то есть любую внешнюю ссылку.

---

#### ПРИМЕЧАНИЕ

Этот селектор не будет работать при использовании защищенного SSL-соединения, то есть когда ссылка начинается со значения `https://`. Чтобы создать стиль, учитывающий данную проблему, вам понадобится группа селекторов:

```
a[href^="http://"], a[href^="https://"]
```

---

Встречаются также ситуации, когда вам необходимо выбрать элемент с атрибутом, *заканчивающимся* определенным значением. И снова для этой задачи пригодны ссылки. Скажем, вы хотите добавить небольшой значок после ссылок, указывающих на PDF-файлы. Поскольку они имеют расширение PDF, вы знаете, что ссылка на эти документы будет заканчиваться также этими символами, например `<a href = "download_instructions">`. Значит, чтобы выбрать только такие ссылки, нужен следующий селектор:

```
a[href$=".pdf"]
```

А стиль целиком будет выглядеть так:

```
a[href$=".pdf"] {  
    background: url(doc_icon.png) no-repeat;  
    padding-left: 15px;  
};
```

Не беспокойтесь о том, что не знаете конкретные свойства этого стиля — вы прочтете о них далее в книге. Только обратите внимание на один интересный селектор: `$=` означает «заканчивается на». Вы можете использовать его для форматирования ссылок на документы Word (`[a href$=".doc"]`), фильмы (`[a href$=".mp4"]`) и т. д.

И наконец, вы можете выбирать элементы с атрибутами, содержащими конкретное значение. Например, вам нужно выделить фотографии сотрудников повсюду на сайте. Вы хотите, чтобы у всех фотографий был общий стиль, допустим зеленая рамка и серый фон. Один из способов сделать это — создать класс, например `.photo`,

и добавлять вручную атрибут класса к соответствующим элементам `img`. Однако, если вы задали для фотографий последовательные названия, это не самый быстрый метод.

Например, каждую из фотографий вы называли, используя при этом слово `photo`: `ivanov_photo`, `petrov_photo` и т. д. В каждом файле встречается слово `photo`, поэтому и атрибут `src` тега `<img>` содержит это слово. Вы можете создать селектор специально для этих изображений:

```
img[src*="photo"]
```

Выражение переводится как «выберите все изображения повсюду, атрибут `src` которых содержит слово `photo`». Это простой и изящный способ форматирования фотографий сотрудников.

## Дочерние селекторы

Подобно применению селекторов потомков, описанных ранее, в CSS можно форматировать вложенные элементы с помощью *дочернего* селектора, который использует дополнительный символ — угловую скобку (`>`) — для указания отношения между двумя элементами. Например, `body > h1` выбирает любой элемент `h1`, дочерний по отношению к `body`.

В отличие от селектора потомков, который применяется ко *всем* потомкам (то есть вложенным элементам), дочерний селектор позволяет определить конкретные дочерний и родительский элемент. На рис. 3.6 вы можете видеть два элемента `h2`. Использование селектора потомков `body h2` привело бы к выбору обоих, так как они являются вложенными по отношению к `body`. Но только второй — дочерний элемент `body`. Первый `h2` непосредственно вложен в `div`, который и является родительским. Поскольку у `h2` различные родительские элементы, то для того, чтобы добраться до каждого из них отдельно, можно использовать дочерний селектор. Для выбора только первого элемента `h2` используйте код `body > h2`, а для выбора второго — `div > h2`.

Для выбора дочерних элементов в CSS3 также включены некоторые весьма специфические псевдоклассы. Они позволяют точнее настроить селекторы под многие различные компоновки HTML-кода.

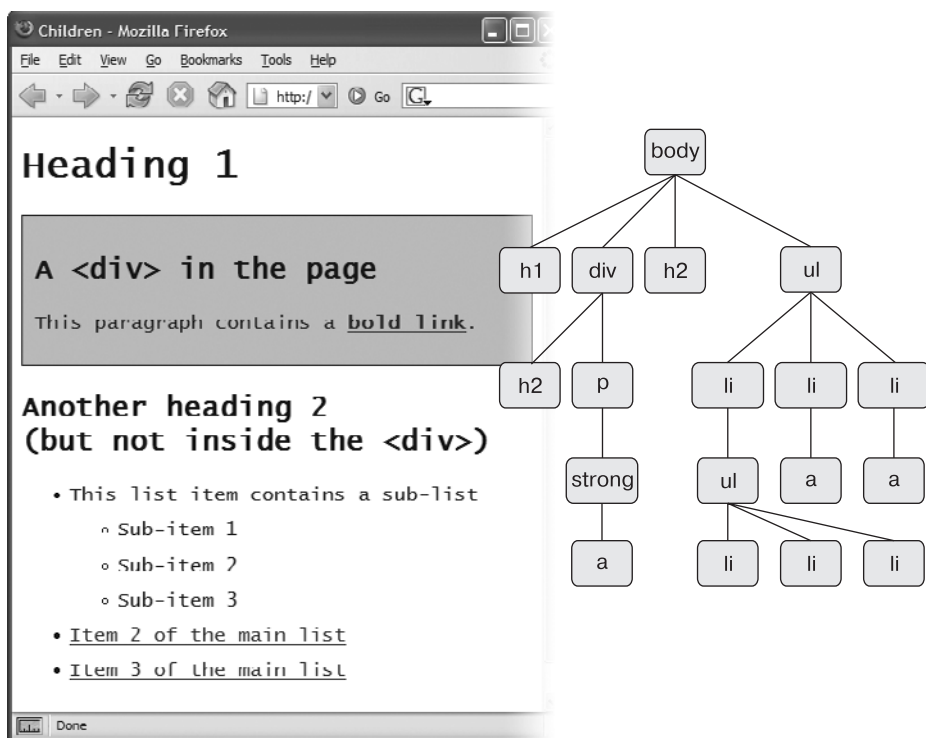
### :first-child

Возвращаясь на время к аналогии с семейным деревом HTML, вспомним, что такое дочерний элемент: это любой элемент, непосредственно заключенный в другой элемент. (Например, на рис. 3.6 элементы `h1`, `div`, `h2` и `ul` являются дочерними по отношению к `body`.) Псевдоэлемент `:first-child` позволяет выбрать и форматировать только первый из них, независимо от того, сколько их вообще может быть.

Если нужно выбрать *первый* элемент `h1` на странице, показанной на рис. 3.6, создайте следующий селектор:

```
h1:first-child
```





**Рис. 3.6.** Диаграмма в виде дерева (справа) показывает отношения между HTML-элементами (слева)

Этот селектор применяется к любому элементу `h1`, являющемуся первым дочерним элементом. На рис. 3.6 результат будет очевиден: там только один элемент `h1` и он является первым элементом на странице. Следовательно, он является дочерним для элемента `body`. Но `:first-child` может вызвать путаницу. Например, если вы измените элемент `h2` внутри `div`, показанного на рис. 3.6, на элемент `h1`, то `h1:first-child` выберет оба заголовка `h1`: тот, который непосредственно находится внутри элемента `body`, и тот `h1`, который находится внутри `div`-контейнера (поскольку этот заголовок `h1` является первым дочерним элементом `div`).

## **:last-child**

Этот псевдоэлемент похож на рассмотренный ранее `:first-child`, но выбирает последний дочерний элемент. Например, для форматирования последнего элемента списка нужно воспользоваться селектором `li:last-child` (рис. 3.7).

## **:only-child**

Существует также селектор для элемента, который является единственным дочерним для другого элемента. Допустим, в вашей таблице стилей есть такой стиль:

## Child Selectors

**li:first-child**

one  
two  
three  
four  
five  
six

**li:last-child**

one  
two  
three  
four  
five  
six

**li:nth-child(odd)**

one  
two  
three  
four  
five  
six

**li:nth-child(even)**

one  
two  
three  
four  
five  
six

**li:nth-child(2)**

one  
two  
three  
four  
five  
six

**li:nth-child(5)**

one  
two  
three  
four  
five  
six

**li:nth-child(3n)**

one  
two  
three  
four  
five  
six

**li:nth-child(2n)**

one  
two  
three  
four  
five  
six

**li:nth-child(3n+1)**

one  
two  
three  
four  
five  
six

**li:nth-child(4n+2)**

one  
two  
three  
four  
five  
six

**li:nth-child(n+3)**

one  
two  
three  
four  
five  
six

**li:nth-child(-n+3)**

one  
two  
three  
four  
five  
six

**Рис. 3.7.** Многочисленные дочерние селекторы в CSS предоставляют вам различные способы выбора дочерних элементов. Эти селекторы хорошо подходят для выделения первого, последнего или чередующихся элементов списка

```
p:only-child {
  color: red;
}
```

Он указывает, что цвет текста должен быть красным, но только в том случае, если внутри элемента находится лишь один абзац. Например, если в вашем элементе `div` находится три абзаца, то стиль не будет применен, поскольку внутри элемента `div` находится три дочерних элемента. Тем не менее, если из элемента `div` вы удалите два абзаца, оставшийся абзац станет красным.

Все несколько запутанно, но следует помнить, что данный стиль работает только в том случае, когда конкретный элемент является единственным дочерним для другого элемента. Иными словами, недостаточно, чтобы элемент являлся единственным в своем роде. Если у него существует другой родственный элемент, то селектор не сработает. Если вы добавите элемент `ul` внутрь `div` вместе с `p`, то абзац больше не будет единственным дочерним объектом. Элемент `ul` будет таким же дочерним, поэтому селектор `p:only-child` не будет применен.

## :nth-child

Этот комплексный селектор очень полезен. К примеру, с его помощью можно легко и просто форматировать каждую вторую строку в таблице, каждый третий элемент в списке или придать свой стиль любому сочетанию чередующихся дочерних элементов (см. рис. 3.7). Для определения того, какой из дочерних элементов нужно выбирать, этот селектор требует значение. Проще всего указать ключевое слово — либо `odd`, либо `even`, — позволяющее выбрать чередующиеся нечетные или четные дочерние элементы соответственно. Например, если нужно предоставить один фоновый цвет для каждой четной строки таблицы и другой фоновый цвет для каждой нечетной, можно создать два следующих правила:

```
tr:nth-child(odd) { background-color: #D9F0FF; }  
tr:nth-child(even) { background-color: #FFFFFF; }
```

Теперь у вас есть действительно простой способ для окрашивания чередующихся строк таблицы (рис. 3.8). Но у псевдоэлемента `:nth-child()` в рукаве спрятано еще немало козырей.

Можно также выбрать определенный дочерний элемент, указав его номер. Например, если вы хотите выделить пятый элемент в списке, задайте цифру 5 в селекторе `:nth-child()`:

```
li:nth-child(5)
```

Этот стиль выделяет только один дочерний элемент. Если вы хотите выделить, скажем, каждый третий элемент в списке, используйте цифру (в данном примере это 3) и букву `n`:

```
li:nth-child(3n)
```

Здесь `n` — это множитель, поэтому запись `3n` означает каждый третий дочерний элемент, начиная с третьей строки (см. рис. 3.7).

Но как быть, если вы хотите выделить каждый третий дочерний элемент списка, начиная со второго дочернего элемента? Представим, что вам нужно выделить каждую третью ячейку таблицы (элемент `td`) внутри строки, начиная со второй ячейки таблицы (см. рис. 3.8). Для этого нужно применить следующий стиль:

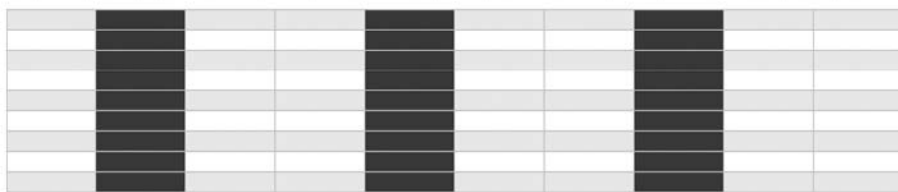
```
td:nth-child(3n+2) { background-color:#900; }
```

Число перед `n` — множитель: запись `3n` означает каждый третий элемент, `4n` — каждый четвертый элемент и т. д. Символ `+` и следующее за ним число (`+2` в данном примере) указывают, с какого элемента нужно начинать выделение, следовательно, `+2` означает, что нужно начинать со второго дочернего элемента, а `+5` говорит о том, что нужно начинать выделение с пятого дочернего элемента. Таким образом, использование псевдоэлемента `:nth-child(5n+4)` приведет к выделению каждого пятого дочернего элемента, начиная с четвертого дочернего элемента.

Вы также можете использовать отрицательные значения `n`, что приведет к выбору всех предшествующих дочерних элементов *в обратном порядке*. Например, последний список на рис. 3.7 использует следующий селектор:

```
li:nth-child(-n+3)
```

## Alternating Table Rows



**Рис. 3.8.** Простой способ окрашивания ячеек в таблице. Вы даже можете форматировать таблицу, выделяя чередующиеся столбцы, нацеливаясь на каждый второй элемент `td` в строке или, как в данном случае, на каждый третий столбец, начиная со второго

Он означает «начать с третьего элемента в списке и выделить каждый предшествующий ему элемент». Как можно увидеть, селектор `nth-child` сложный, но достаточно мощный инструмент, который позволяет выделять бесконечное разнообразие дочерних элементов.

## Дочерние псевдоклассы

Каскадные таблицы стилей включают селектор, который работает во многом похоже на дочерний, рассмотренный в предыдущем разделе, но применяется к дочерним элементам с *HTML-элементом* определенного типа. Предположим, что вам нужно определенным образом отформатировать первый абзац на боковой панели, но только на тех страницах, где эта боковая панель начинается с элемента `h2`, и на других страницах, где она начинается с элемента `p`. Псевдоэлемент `:first-child` для выбора этого абзаца применять нельзя, потому что в некоторых случаях абзац является *вторым* дочерним элементом (который следует за `h2`). Тем не менее он всегда является первым абзацем (элементом `p`) на этой боковой панели, даже если перед ним идут какие-нибудь другие элементы, следовательно, его можно выбрать с помощью селектора `first-of-type`.

### ПРИМЕЧАНИЕ

Псевдоклассы `:last-child`, `:first-of-type` и `:nth-child()` поддерживаются всеми современными браузерами, включая Internet Explorer 9 и выше. Но в Internet Explorer 8 они не работают.

## `:first-of-type`

Селектор работает так же, как и `:first-child`, но применяется к дочернему элементу, имеющему определенный элемент. Предположим, что у вас есть боковая панель с классом `sidebar`. Для форматирования первого абзаца этой боковой панели используется следующий селектор:

```
.sidebar p:last-of-type
```

Обратите внимание на букву `p` в коде `p:first-of-type`. Она обозначает элемент, который вы собираетесь отформатировать.

## :last-of-type

Селектор работает так же, как и `:last-child`, но применяется к последнему экземпляру элемента определенного типа. Например, если нужно на боковой панели `div` определенным образом отформатировать последний абзац, но вы не уверены, что за абзацем нет каких-либо других элементов (например, неупорядоченного списка, заголовка или рисунка). Этот стиль имеет следующий вид:

```
.sidebar p:last-of-type
```

### ПРИМЕЧАНИЕ

Следует помнить, что указанные селекторы тегов также должны быть дочерними по отношению к конкретному элементу. Следовательно, код `p:first-of-type` означает «первый дочерний элемент, являющийся элементом абзаца».

## :nth-of-type

Работает так же, как и `:nth-child()`, но применяется к чередующимся дочерним элементам, имеющим определенный элемент. Этот селектор может пригодиться при наличии больших абзацев текста, усеянных фотографиями. Элемент `img` является строчным, поэтому у вас может быть абзац `p` с несколькими изображениями `img` внутри. И если вам захочется чередовать появление изображений то слева, то справа, как показано на рис. 3.9, это можно сделать с помощью таких правил:

```
img:nth-of-type(odd) { float: left; }  
img:nth-of-type(even) { float: right; }
```

Как видите, для `:nth-of-type()` используются такие же ключевые слова (`odd` или `even`) и формулы (например,  $2n+1$ ), как и для `:nth-child()`.

Между прочим, `:nth-of-type()` можно также использовать для чередующихся строк таблицы:

```
tr:nth-of-type(odd) { background-color: #D9F0FF; }  
tr:nth-of-type(even) { background-color: #FFFFFF; }
```

Что касается CSS-селекторов, то здесь всегда имеется несколько способов добраться до HTML-элемента. Обычно их набирается более пяти!

### ЧАВО

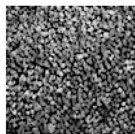
#### Придание спискам привлекательного внешнего вида

*Когда нужно использовать дочерние селекторы? Исходя из текста главы существует достаточное количество селекторов для выборки практически любого элемента веб-страницы. Так для чего нужны остальные селекторы?*

На самом деле существуют некоторые сложности дизайна веб-страниц, где дочерние селекторы просто незаменимы и никакие другие не смогут с этим спра-

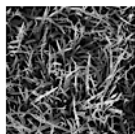
виться. Такая ситуация встречается на большинстве сайтов. В любом маркированном списке присутствует несколько пунктов — элементов списка (см. рис. 3.6). Здесь можно использовать дочерние селекторы, чтобы визуально упорядочить данные в категориях (пунктах) и подкатегориях (подпунктах). Вы форматируете элементы первого уровня списка одним способом, а второго — другим. Содержимое, представленное

## Type Selectors



Ullamco laboris nisi sed do eiusmod tempor incididunt excepteur sint occaecat. In reprehenderit in voluptate velit esse cillum dolore ut labore et dolore magna Ullamco laboris nisi mollit anim id est laborum. Cupidatat non proident, excepteur sint occaecat qui officia deserunt. Velit esse cillum dolore ut aliquip ex ea commodo consequat. Eu fugiat nulla pariatur. In reprehenderit in voluptate. In reprehenderit in voluptate lorem ipsum dolor sit amet, caliqua. Sunt in

culpa quis nostrud exercitation mollit anim id est laborum. Sed do eiusmod tempor incididunt lorem ipsum dolor sit amet, excepteur sint occaecat. Ut aliquip ex ea commodo consequat. Ut enim ad minim veniam, consectetur adipisicing elit, sed do eiusmod tempor



incididunt. Cupidatat non proident. Velit esse cillum dolore sed do eiusmod tempor incididunt ut enim ad minim veniam. Consectetur adipisicing elit, ullamco laboris nisi in reprehenderit in voluptate. Duis aute irure dolor quis nostrud exercitation eu fugiat nulla pariatur. Sed do eiusmod tempor incididunt consectetur adipisicing elit, ut labore et dolore magna aliqua. In reprehenderit in voluptate

ut aliquip ex ea commodo consequat. Sunt in culpa sed do eiusmod tempor incididunt duis aute irure dolor. Ut enim ad minim veniam, lorem ipsum dolor sit amet, ullamco laboris nisi. Sed do eiusmod tempor incididunt eu fugiat nulla pariatur. Quis nostrud exercitation consectetur adipisicing elit, in reprehenderit in voluptate. Updatat non proident. Ut enim ad minim veniam, consectetur adipisicing elit, sunt in culpa.



**Рис. 3.9.** Используя селектор `:nth-of-type()`, можно легко выбрать любые изображения внутри элемента, чередуя их расположение то справа, то слева

### ЧАВО

таким способом, выглядит четко, профессионально и читабельно.

Сначала создайте класс для самого верхнего — внешнего — уровня вложенности элементов списка и назовите его, скажем, `.mainList`. Для первого уровня вы можете использовать шрифт `sans-serif`, имеющий немного больший размер по сравнению с основным текстом веб-страницы, возможно, в другом цвете. Последующие категории могут быть представлены Times для лучшего восприятия. При большом объеме текста форматирование каждого уровня подкатегорий с небольшим отличием позволяет посетителям веб-страниц визуально ориентироваться в материале.

Теперь примените класс `.mainList` к первому элементу `ul`: `<ul class="mainList">`. Затем используйте дочерний селектор (`ul.mainList > li`) для выбора элементов списка только первого уровня и придания пунктам необходимого форматирования. Данный

стиль будет применен только к элементу `li`, являющемуся дочерним по отношению к `ul` и принадлежащему к классу `.mainList`. Затем для задания стиля дочерним элементам `li` любых последующих вложенных элементов `ul` воспользуйтесь таким селектором: `ul.mainList > li > ul > li`. (Селекторы потомков, как `ul li`, в отличие от этого, выбирают элементы списка всех неупорядоченных списков на странице: и вложенных, и невложенных.)

Вам нужно будет обратить внимание и на понятие наследования, которое будет рассмотрено в следующей главе. Как правило, конкретные CSS-свойства, примененные к одному элементу, наследуются элементами, находящимися внутри него. Поэтому, если даже вы используете дочерний селектор, нацеленный на чей-либо дочерний элемент, свойства могут перейти на другие элементы внутри этого дочернего элемента. Один из способов избежать такого развития событий заключается в применении селектора `:not()`.

## Родственные селекторы

Родительско-дочерние отношения — не единственная форма родственных связей в дереве HTML. Иногда требуется выбрать элемент, относящийся к группе родственных элементов одного уровня с общим родителем. Элемент, который следует сразу же за другим элементом, в HTML называется *смежным родственным элементом* того же уровня. На рис. 3.6 элемент `div` является смежным по отношению к `h1`, а элемент `p` — по отношению к `h2` и т. д.

Используя смежный родственный селектор, можно, к примеру, придать первому абзацу после каждого заголовка форматирование, отличное от следующих абзацев. Предположим, вы хотите удалить отступ, который автоматически появляется перед элементом `p`, чтобы между заголовком и абзацем не было промежутка. Или хотите придать абзацу, как небольшому вводному описанию, другой цвет и размер шрифта.

Смежный родственный селектор использует знак `+` для соединения одного элемента с другим. Поэтому, чтобы выбрать все первые абзацы, следующие за любым заголовком `h2`, используйте селектор `h2 + p` (пробелы необязательны, так что `h2+p` также будет прекрасно работать). Последний элемент в селекторе (в данном случае `p`) — элемент, который нужно отформатировать, но только при условии, что он следует сразу за смежным для него элементом `h2`.

Есть и другой родственный селектор, который называется *общим родственным сборным селектором*. Он обозначается знаком тильды (`~`) и означает следующее: «Нужно выбрать все родственные элементы этого типа». Например, если селектором `h2 + p` задается выбор отдельного абзаца `p`, который следует сразу за заголовком `h2`, то селектором `h2 ~ p` задается выбор *всех* элементов `p`, родственных (то есть находящихся на одном уровне) по отношению к заголовку `h2`. Честно говоря, вы можете так и не найти этому селектору достойного применения, но в CSS весьма разнообразный синтаксис.

## Селектор `:target`

Селектор `:target()` забавен. С его помощью можно создавать на самом деле интересные эффекты, например применять стиль к элементу страницы *после* того, как будет выполнен щелчок кнопкой мыши на другом элементе. Этот элемент интерактивности обычно выполняется с помощью сценариев JavaScript. Селектор зависит от использования конкретных идентификаторов, как это показано во врезке «HTML-элементы `div` и `span`» в разделе «Идентификаторы: отдельные элементы веб-страницы» этой главы. Идентификаторы используются для связи с определенным местом на странице.

Например, вы работаете с веб-страницей `index.html`. На ней находится элемент `div` с идентификатором `signupForm`. Если на этой странице (или на другой) расположена ссылка, которая указывает на объект `index.html#signupForm`, то после щелчка кнопкой мыши на ней браузер перейдет к этому элементу `div`. Если объект `div` находится в нижней части очень длинной страницы, то браузер прокрутит



страницу, отобразив его. Такие селекторы иногда называют *внутренними ссылками* и используют в качестве предметных указателей на веб-страницах. Щелкнув на такой ссылке, посетитель перейдет к области страницы, в которой встречается искомое слово.

Но вы не должны использовать эту функцию для перехода к другой области веб-страницы. Всякий раз, когда в URL в адресной строке браузера встречается символ # и следующий за ним идентификатор, элемент с указанным идентификатором становится селектором target. Поэтому вы можете применить конкретный стиль к элементу только в том случае, когда его идентификатор присутствует в URL-адресе.

Ниже приведен пример работы селектора :target. Представьте, что следующий код находится в верхней части веб-страницы:

```
<button>
  <a href="#signupForm">Подпишитесь на нашу рассылку</a>
</button>
<form id="signupForm">
  <label for="email">Укажите свой адрес электронной почты</label>
  <input type="email" id="email">
  <input class="btn" type="submit" value="Подписаться">
</form>
```

Когда посетитель сайта щелкнет кнопкой мыши на ссылке (элемент a), форма станет целевой. Другими словами, вы можете использовать один стиль для формы в ее обычном состоянии и другой после того, как посетитель выполнит щелчок на ссылке. Например, вы могли бы сначала скрыть форму (о свойствах CSS, отвечающих за отображение объектов, вы прочитаете в разделе «Принципы работы свойств позиционирования» главы 14) с помощью такого правила:

```
#signupForm {
  display: none;
}
```

Оно скрывает форму, поэтому, когда страница будет загружена, посетитель не увидит ее. Но, когда он щелкнет на ссылке **Подпишитесь на нашу рассылку**, форма станет целевой и вы сможете отформатировать ее с помощью следующего кода:

```
#signupForm:target {
  display: block;
}
```

Другими словами, если в адресной строке браузера отображается только URL, например index.html, то браузер использует первое правило, скрывая форму. Но если адрес выглядит, к примеру, следующим образом: index.html#signupForm, то браузер использует селектор target и форма отображается.

---

#### ПРИМЕЧАНИЕ

В галерее по адресу [tinyurl.com/ltqzifu](http://tinyurl.com/ltqzifu) представлены поистине замечательные примеры использования селектора :target.

---

## Селектор `:not()`

Селектор `:not()`, также известный как *псевдокласс отрицания*, позволяет выбрать что-либо отличное от другого. Например, можно применить класс к абзацу — `<p class="classy">` — и создать селектор, позволяющий отформатировать этот абзац:

```
.classy { color: red; }
```

А что делать, если понадобится выбрать все абзацы, *за исключением* тех, которым присвоен класс `classy`? Именно здесь пригодится селектор `:not()`. Чтобы указать, что *не* нужно выбирать, селектор помещается в круглые скобки. Например:

```
p:not(.classy) { color: blue; }
```

Этот стиль форматирует текст синим цветом во всех абзацах, к которым не применен класс `classy`.

Селектор `:not()` может быть полезен при использовании селекторов атрибутов. Например, ранее было показано, что селектор атрибутов можно использовать, чтобы выбрать все ссылки, указывающие за пределы вашего сайта:

```
a[href^="http://"]
```

Как вы уже, наверное, заметили, этот селектор не выбирает конкретно все ссылки, указывающие за пределы вашего сайта, он просто выбирает все ссылки, использующие абсолютные URL-адреса, то есть начинающиеся со значения `http://`. Для многих сайтов это одно и то же, поскольку они используют для указания на другие страницы этого же сайта ссылки относительно документа или главной страницы сайта, а для указания ссылок на другие сайты применяются абсолютные URL. Но в некоторых случаях абсолютные URL-адреса могут использоваться для указания на страницу внутри вашего сайта.

Например, некоторые системы управления контентом (в частности, WordPress) используют для указания на публикации блогов внутри сайта абсолютные URL-адреса. В этом случае, если нужно придать стиль ссылкам, которые ведут за пределы вашего сайта, следует усовершенствовать селектор атрибута, задействовав также селектор `:not()`. Предположим, доменное имя вашего сайта `mysite.com`. Для выбора ссылки, указывающей за пределы вашего сайта, нужно выбрать все абсолютные ссылки, *не* указывающие на домен `mysite.com`. Вот как это можно сделать:

```
a[href^="http://"]:not([href="http://mysite.com"])
```

Если перевести на нормальный язык, этот селектор говорит: «Нужно выбрать все ссылки, значение атрибута `href` которых начинается с текста `http://`, но не те, которые начинаются с `http://mysite.com`». Если вспомнить предыдущий материал, в селекторе атрибута символы `^=` означают директиву «начинается с». То же самое можно написать еще короче:

```
a[href^="http://"]:not([href*= "mysite.com"])
```

В селекторе атрибута символы `*=` означают правило «содержит», тем самым любой абсолютный URL-адрес, содержащий значение `mysite.com`, будет исключен. В том числе адреса `http://www.mysite.com` и `http://mysite.com`.

В отношении селектора `:not()` действуют несколько ограничений.

- С селектором `:not()` можно использовать только *простые селекторы*. Другими словами, можно применять селекторы тегов (такие как `html` или `p`), универсальный селектор (`*`), классы (например, `.footer`), идентификаторы (например, `#banner`) или псевдоклассы (`:hover`, `:checked`, `:first-child` и т. д.). Таким образом, все следующие селекторы можно считать правильными:

```
.footnote:not(div)
img:not(.portrait)
div:not(#banner)
li:not(:first-child)
```

- Нельзя использовать селекторы потомков (такие как `div p a`), псевдоэлементы (такие как `::first-line`), групповые селекторы или комбинации (такие как родственный смежный селектор `h2 + p`).
- Нельзя в одной строке указывать несколько селекторов `:not()`. Например, следующий код будет неправильным:

```
a[href^="http://"]:not([href*="google.com"]):not([href="yahoo.com"])
```

Другими словами, значение `:not()` в селекторе можно использовать только один раз.

## Практикум: использование селекторов

В оставшейся части этой главы вы потренируетесь в создании разных селекторов и увидите, как они влияют на дизайн веб-страницы. Практикум начнем с представления основных типов, а затем перейдем к более современным стилям.

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](https://github.com/mrightman/css_4e). Перейдите по ссылке и загрузите ZIP-архив с файлами (нажав кнопку **Download ZIP** в правом нижнем углу страницы). Файлы текущего практикума находятся в папке 03.

### ВОЗМУЩЕННОЕ ЧАВО

#### Используйте внутренние таблицы стилей

*Почему в этом практикуме мы пользуемся внутренними таблицами стилей? Ведь в главе 2 книги рекомендуется применять внешние!*

Да, внешние каскадные таблицы стилей используются для создания быстро загружаемых, «производительных» сайтов. Однако внутренние таблицы стилей упрощают разработку одиночных веб-страниц, таких, как в этом практикуме. В данном случае гораздо удобнее работать с одним файлом, вместо того чтобы переключаться между файлом внешней таблицы стилей и веб-страницей.

Кроме того, вы можете пользоваться предварительным просмотром результатов своей работы без постоянного обновления кэша браузера.

Ввиду вышесказанного я рекомендую использовать внешние таблицы стилей для сайтов. Если вы собираетесь использовать стили, созданные в этом уроке, в дальнейшем, а не только в учебных целях — на здоровье. Но в рамках текущего упражнения, чтобы выполнять задания быстро и легко, вы будете пользоваться одиночным HTML-файлом и внутренней таблицей стилей.

1. Откройте файл `selector_basics.html`, расположенный в папке **03**, в редакторе HTML-кода.

Страница состоит из основных HTML-элементов (рис. 3.10). В этом практикуме мы попытаемся придать ей изящный вид. Сначала вы свяжете страницу со шрифтом Google, который уже использовали в предыдущей главе.

2. Введите следующий код в пустой строке сразу после элемента `</title>`:

```
<link href='http://fonts.googleapis.com/css?family=Kurale' rel='stylesheet'>
```

Этот код связывается с внешней таблицей стилей, размещенной на сервере Google. Она загружает шрифт Kurale и тем самым дает вам возможность использовать его на своей странице (подробнее об использовании веб-шрифтов читайте в разделе «Использование веб-шрифтов» главы 6). Теперь необходимо добавить внутреннюю таблицу стилей.

3. Сразу после добавленного на предыдущем шаге элемента `link` нажмите клавишу **Enter** для перехода на новую строку и введите тег `<style>`. Дважды нажмите клавишу **Enter** и введите `</ >`.

Эти открывающий и закрывающий теги элемента `style` нужны для размещения внутренней таблицы стилей. Очень полезно набирать их сразу парой, чтобы случайно не забыть о закрывающем теге. Теги сообщают браузеру, что между ними находятся команды языка CSS. HTML-код теперь должен выглядеть следующим образом (код, который вы только что добавили, выделен полужирным шрифтом):

```
<title>Селекторы</title>
<link href='http://fonts.googleapis.com/css?family=Kurale' rel='stylesheet'>
<style>

</style>
```

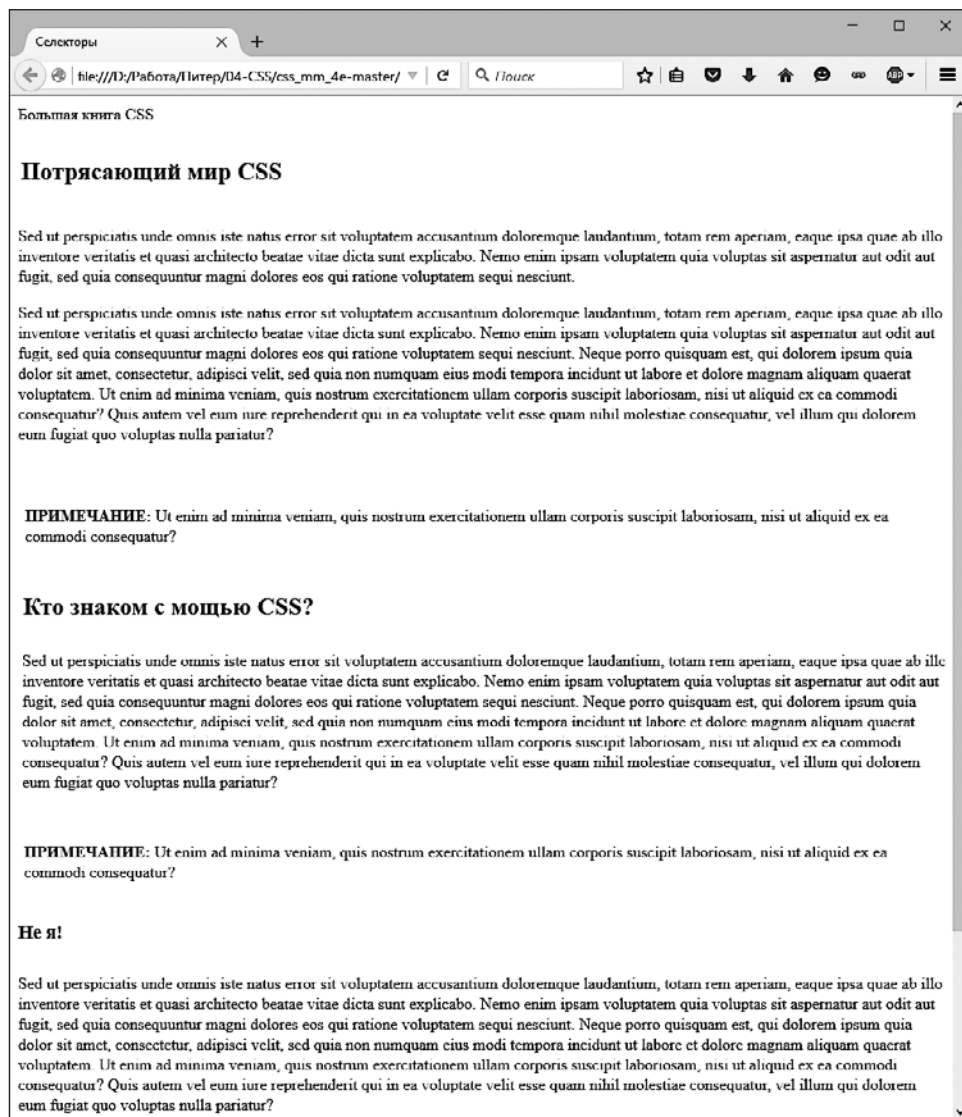
Теперь нужно добавить селектор тега (если вы выполнили все упражнения практикума в прошлой главе, то уже умеете это делать).

4. Перейдите к строке между открывающим и закрывающим тегами элемента `style`, а затем наберите код `body {`. Дважды нажмите клавишу **Enter** и введите `}`.

Как я уже отмечал, желательно указывать закрывающую скобку сразу же, как только вы добавили открывающую. Чтобы создать селектор тега, просто укажите имя HTML-элемента, который следует отформатировать. Он будет применен ко всем абзацам текста, заключенным в элемент `body`. Теперь необходимо изменить фоновый цвет и поле вокруг страницы.

5. Щелкните кнопкой мыши между скобками `{` и `}` стиля `body` и добавьте три правила форматирования — цвет, отступ и поле:

```
body {
  background-color: rgb(50,122,167);
  padding: 0 20px 20px 20px;
  margin: 0;
}
```



**Рис. 3.10.** Простой HTML-текст смотрится в браузере чересчур аскетично. Но с помощью каскадных таблиц стилей вы можете из серой страницы, показанной на этом рисунке, получить потрясающе красивую (рис. 3.11), выполнив три десятка простых шагов

Нажимайте клавишу **Enter**, чтобы поместить каждое CSS-свойство на отдельную строку. Кроме того, полезно сделать отступы клавишей **Tab**, чтобы улучшить визуальное восприятие кода CSS (некоторые разработчики вместо табуляции используют два пробела, а вы можете выбрать то, что вам больше нравится).

Свойство цвета фона, которое мы только что изменили, обозначается как `rgb()` — это один из способов указать значения красного, зеленого и синего цветов. В дан-

ном случае цвет фона темно-синий. Он затрудняет чтение текста, поэтому необходимо изменить цвет текста элементов абзаца.

#### ПРИМЕЧАНИЕ

Если вы новичок в веб-дизайне, то имена свойств и их значения вам пока незнакомы. Так что просто набирайте их в том виде, как показано в практических шагах. Обо всех этих свойствах вы узнаете в главе 6.

6. Добавьте другой стиль под только что созданным стилем `body`.

```
p {  
  color: rgba(255,255,255,.6);  
  font-size: 1em;  
  font-family: "Kurale", Arial, Helvetica, sans-serif;  
}
```

С помощью трех свойств CSS мы изменили форматирование всех абзацев (всех элементов `p`) — определили цвет, размер и шрифт текста. На этот раз, чтобы указать цвет, мы использовали значение `rgba()`. Дополнительная буква `a` в свойстве `rgb` позволяет создать частично прозрачный цвет. В данном примере для абзаца мы использовали белый цвет (его значения — 255,255,255) с непрозрачностью 60 % (значение `.6`). Сквозь текст проступает синий цвет, поэтому он кажется светло-синим.

Самое время посмотреть, что у нас получилось.

7. Откройте страницу в браузере для предварительного просмотра.

До изменения настроек текст веб-страницы отображался шрифтом с засечками (таким как Times New Roman). Теперь, если каскадные таблицы стилей функционируют должным образом, вы увидите семь абзацев, для которых задан светло-синий цвет текста и шрифт Kurale.

## Создание группового селектора

Нередко бывает так, что несколько различных элементов веб-страницы должны иметь одинаковый внешний вид. Вероятно, и вы хотите, чтобы все заголовки отображались шрифтом одного вида и цвета. Вместо того чтобы создавать отдельные стили и дублировать одни и те же атрибуты и параметры для каждого элемента `h1`, `h2` и т. д., вы можете собрать и сгруппировать несколько элементов в единственный селектор.

1. Вернитесь к своему HTML-редактору с открытым файлом `selector_basics.html`. Сейчас мы добавим новый стиль сразу после только что созданного стиля элемента `p`.
2. Щелкните кнопкой мыши после закрывающей фигурной скобки селектора тега `p`, нажмите клавишу **Enter** для начала новой строки и наберите код:

```
h1, h2, h3 {  
  
}
```

Как описано в этой главе ранее, групповой селектор — это просто список. Данный стиль создает одинаковое форматирование элементов h1, h2 и h3 веб-страницы.

- Щелкните на пустой строке между открывающей { и закрывающей } фигурными скобками и добавьте пять CSS-свойств:

```
color: rgb(255,255,255);
font-family: Arial, "Palatino Linotype", Times, serif;
border-bottom: 2px solid rgb(87,185,178);
padding-top: 10px;
padding-bottom: 5px;
```

Здесь вы задаете цвет и тип шрифта для заголовков, добавляете линию границы под заголовками, устанавливаете отступы снизу и сверху, используя свойство padding. Оно добавляет дополнительное пространство от краев элемента без воздействия на фон или рамку, то есть вы добавляете немного пространства под заголовком и между нижней границей текста и рамкой под ним.

- Сохраните файл и просмотрите его работу в браузере.

Заголовок h1 в начале веб-страницы и заголовки h2 и h3 ниже на странице имеют одинаковое начертание и цвет шрифта, а также зеленую рамку сверху (см. рис. 3.11). Элемент h1 выглядит немного мелковатым, но мы легко можем увеличить его.

- Вернитесь к файлу selector\_basics.html в текстовом редакторе. Под только что созданным групповым селектором добавьте еще один стиль:

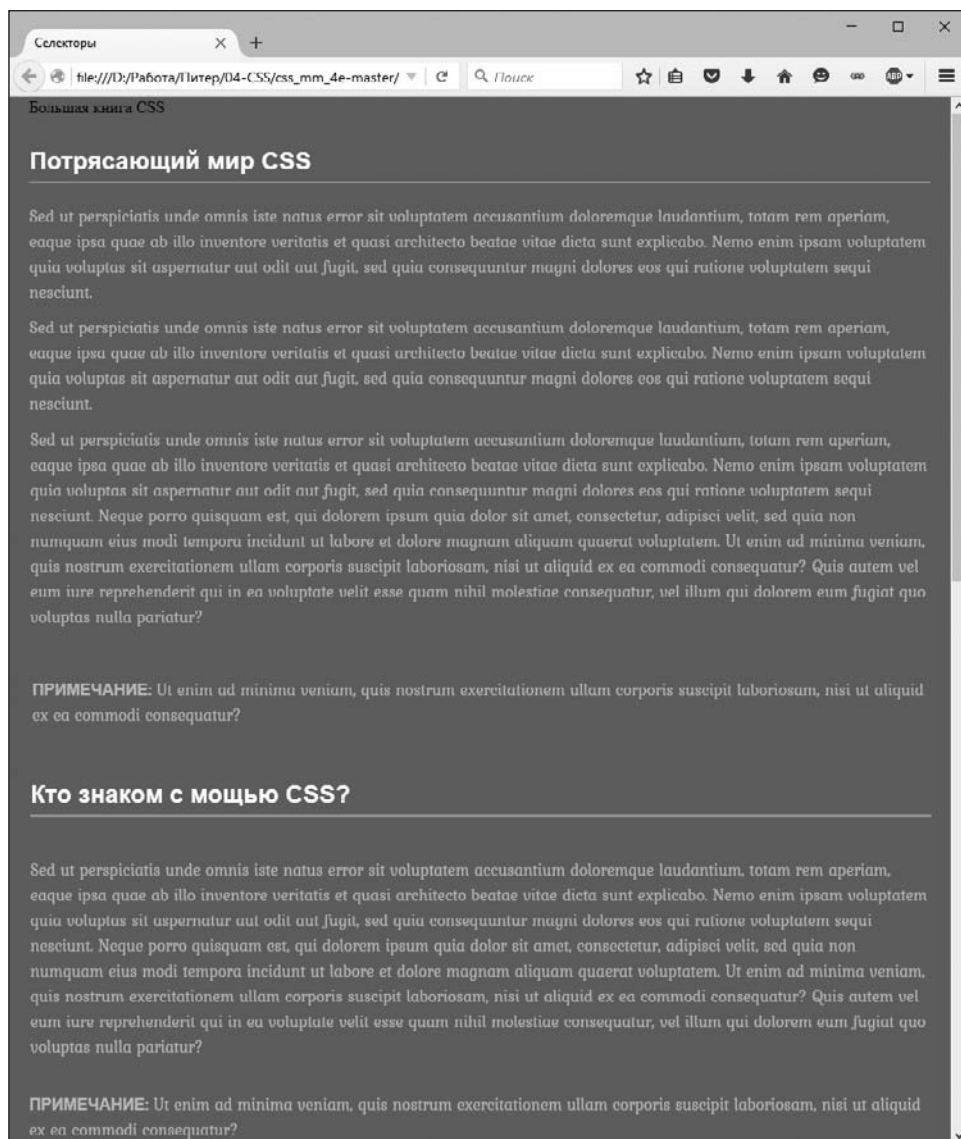
```
h1 {
    font-size: 2em;
}
```

Этот стиль увеличит размер шрифта. Em — это размер шрифта, который используется в браузерах по умолчанию, поэтому запись 2em означает двойной стандартный размер шрифта. Обратите внимание, как удобно применять сразу несколько стилей к одному элементу: правило h1, h2, h3 и h1 в данном случае. В этом примере групповой селектор и новый селектор тега применяются к элементу h1. Такой процесс в языке CSS называется *каскадом*. Подробнее об этом вы прочитаете в главе 5.

## Создание и применение идентификаторов

Идентификаторы предназначены для изменения стиля определенного элемента. Вы создаете стиль и добавляете атрибут id к открывающему HTML-тегу на странице, а затем применяете свойства созданного стиля к этому одиночному элементу. Часто идентификаторы используются для определения элементов формы, создания ссылок на страницах (см. врезку «HTML-элементы div и span» в данной главе), а также помогают применять JavaScript-сценарии для управления элементами на странице. Хотя многие веб-дизайнеры в настоящее время стараются не использовать идентификаторы (почему, вы узнаете в разделе «Управление каскадностью» главы 5), желательно знать, как с ними работать.





**Рис. 3.11.** Простой селектор тега может коренным образом преобразовать внешний вид всех входящих элементов, выполнив форматирование текста веб-страницы. В данном случае групповой селектор делает даже больше, изменяя формат каждого экземпляра заголовков трех различных уровней

В текущем примере мы создадим стиль, который определяет вид текста Большая книга CSS (CSS: The Missing Manual) в самом верху страницы. Этот текст играет роль логотипа, и вы создадите специальный идентификатор для его форматирования.

1. Вернитесь к HTML-редактору с открытым файлом `selector_basics.html`. Добавим после последнего созданного класса `h1` новый стиль.

- Щелкните кнопкой мыши после закрывающей фигурной скобки предыдущего стиля, нажмите клавишу **Enter** для создания новой строки и введите код `#logo {`. Идентификаторы всегда начинаются с символа решетки (`#`). Имя стиля указывает на тип элемента страницы, который является логотипом сайта.

- Нажмите клавишу **Enter** еще раз и введите следующий код:

```
font-family: Baskerville, Palatino, sans-serif;
font-size: 2em;
color: rgba(255,255,255,.8);
font-style: italic;
text-align: center;
margin-bottom: 30px;
background-color: rgb(191,91,116);
border-radius: 0 0 10px 10px;
padding: 10px;
```

Этот код похож на длинный список свойств, но он лишь устанавливает некоторые свойства шрифта, фоновый цвет страницы и добавляет отступы для текста логотипа.

- Завершите определение стиля, введя закрывающую фигурную скобку. Код стиля полностью должен выглядеть так:

```
#logo {
  font-family: Baskerville, Palatino, sans-serif;
  font-size: 2em;
  color: rgba(255,255,255,.8);
  font-style: italic;
  text-align: center;
  margin-bottom: 30px;
  background-color: rgb(191,91,116);
  border-radius: 0 0 10px 10px;
  padding: 10px;
}
```

Если вы сохраните файл и просмотрите его в браузере, то не увидите никаких изменений. Это обусловлено тем, что данный стиль не делает *ничего*, пока вы его не примените. Для этого нужно добавить атрибут `id` к HTML-коду веб-страницы, обозначая фрагмент, который следует отформатировать.

- Найдите на веб-странице открывающий тег `<div>` с текстом Большая книга CSS. Он расположен после тега `<article>`. Добавьте в тег `<div>` код `id="logo"` следующим образом:

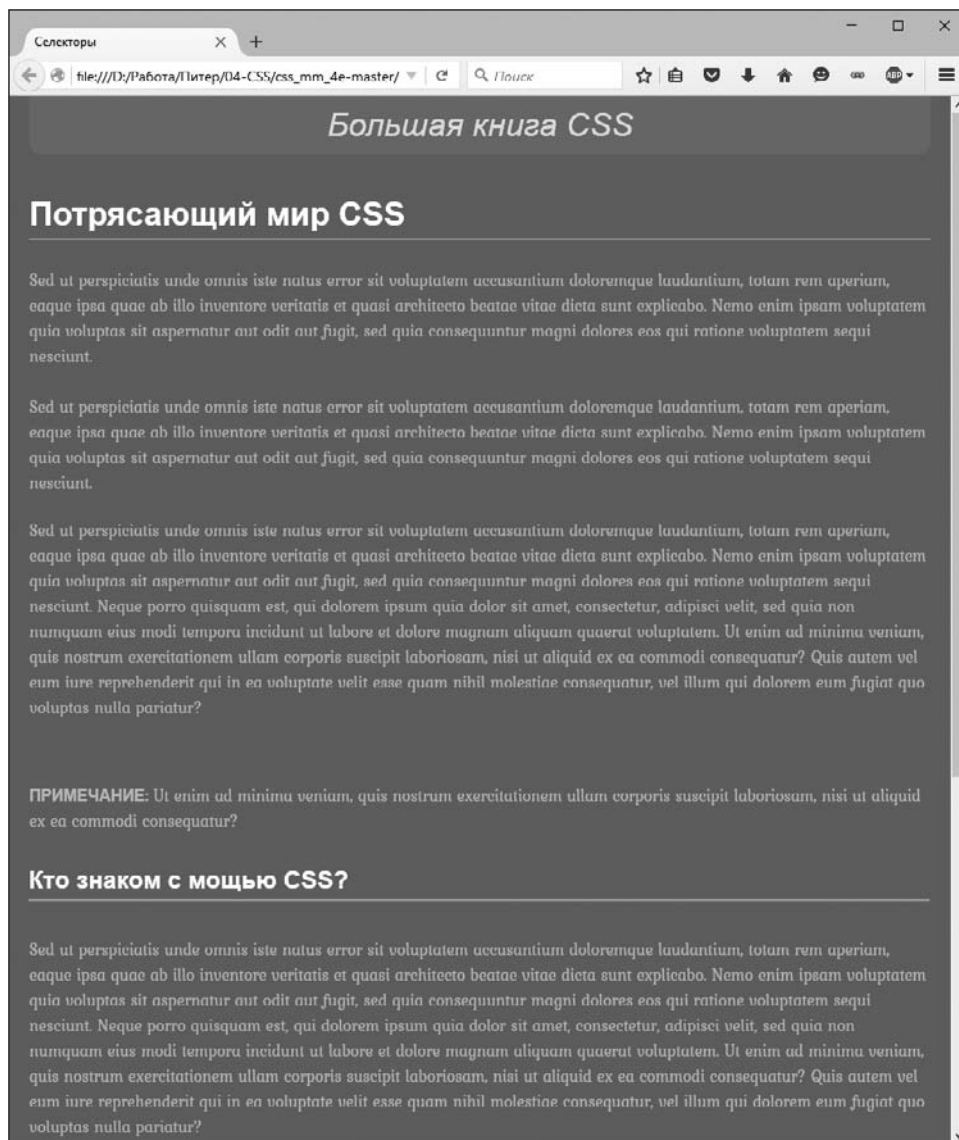
```
<div id="logo">
  Большая книга CSS
</div>
```

Теперь элемент `div` будет отформатирован в соответствии со стилями `#logo`. Как это часто случается при работе с CSS, существует много способов добиться одного и того же результата. Вы могли бы использовать класс и применить его к элементу `div`. Но в данном случае вы используете идентификатор, по-

сколькю назначение стиля — идентификация логотипа на странице — соответствует предназначению идентификаторов.

6. Сохраните страницу и просмотрите ее в браузере.

Теперь текст **Большая книга CSS (CSS: The Missing Manual)** находится в небольшом прямоугольнике в верхней части страницы, располагается по центру и выделен цветом (рис. 3.12).



**Рис. 3.12.** Идентификаторы не единственный способ форматирования отдельных элементов, таких как логотип в верхней части страницы

## Создание и применение классов

Селекторы тегов выполняют свои функции быстро и эффективно, но они, можно сказать, совсем неразборчивы в деталях. Что же делать, если вы хотите отформатировать один элемент `p` на веб-странице иным способом, чем все остальные такие элементы? Решение проблемы — использование классов.

1. Вернитесь к текстовому редактору с открытым файлом `selector_basics.html`. Добавьте вслед за последним созданным стилем еще один.
2. Щелкните кнопкой мыши после закрывающей фигурной скобки селектора `#logo`, нажмите клавишу **Enter** и введите код:

```
.note {  
  
}
```

Имя `note` (примечание) для стиля выбрано не случайно. Оно соответствует его функциям: стиль выделяет абзацы, содержащие дополнительные примечания для посетителей вашего сайта. Создав класс один раз, вы можете применить его ко всем примечаниям веб-страницы (сайта), например к третьему абзацу.

3. Щелкните на пустой строке между открывающей `{` и закрывающей `}` фигурными скобками и добавьте к стилю следующий перечень свойств:

```
color: black;  
border: 2px solid white;  
background-color: rgb(69,189,102);  
margin-top: 25px;  
margin-bottom: 35px;  
padding: 20px;
```

Обратите внимание, что для определения цвета шрифта и границ вы не использовали параметр `rgb()`. В языке CSS существует несколько способов определения цвета, включая ключевые слова, такие как `white` (белый), `black` (черный) или `orange` (оранжевый). Вы узнаете об этом подробнее в разделе «Форматирование текста цветом» главы 6.

Когда вы просмотрите эту веб-страницу, вы не увидите изменений. Как и в случае с идентификаторами, классы не возымеют никакого эффекта на веб-странице, пока стиль не будет применен к HTML-коду.

4. В HTML-коде веб-страницы найдите абзац `p`, текст которого начинается со слова **ПРИМЕЧАНИЕ**, окруженного тегами элемента `strong`.

Чтобы применить класс к этому элементу, добавьте атрибут `class`, за которым должно следовать имя класса, в данном случае `note`.

5. Щелкните кнопкой мыши сразу за именем открывающего тега `p`, нажмите клавишу **Пробел**, а затем введите код `class="note"`. HTML-код теперь должен иметь такой вид (только что набранный код отмечен полужирным шрифтом):

```
<p class="note"><strong>ПРИМЕЧАНИЕ:</strong>
```

Убедитесь, что *не* ввели атрибут с точкой: `class=".note"`. Точка требуется только во время определения имени класса в таблице стилей; в HTML-коде она не нуж-

на. Повторите этот шаг для второго абзаца (он расположен перед элементом h3 с текстом Не я!).

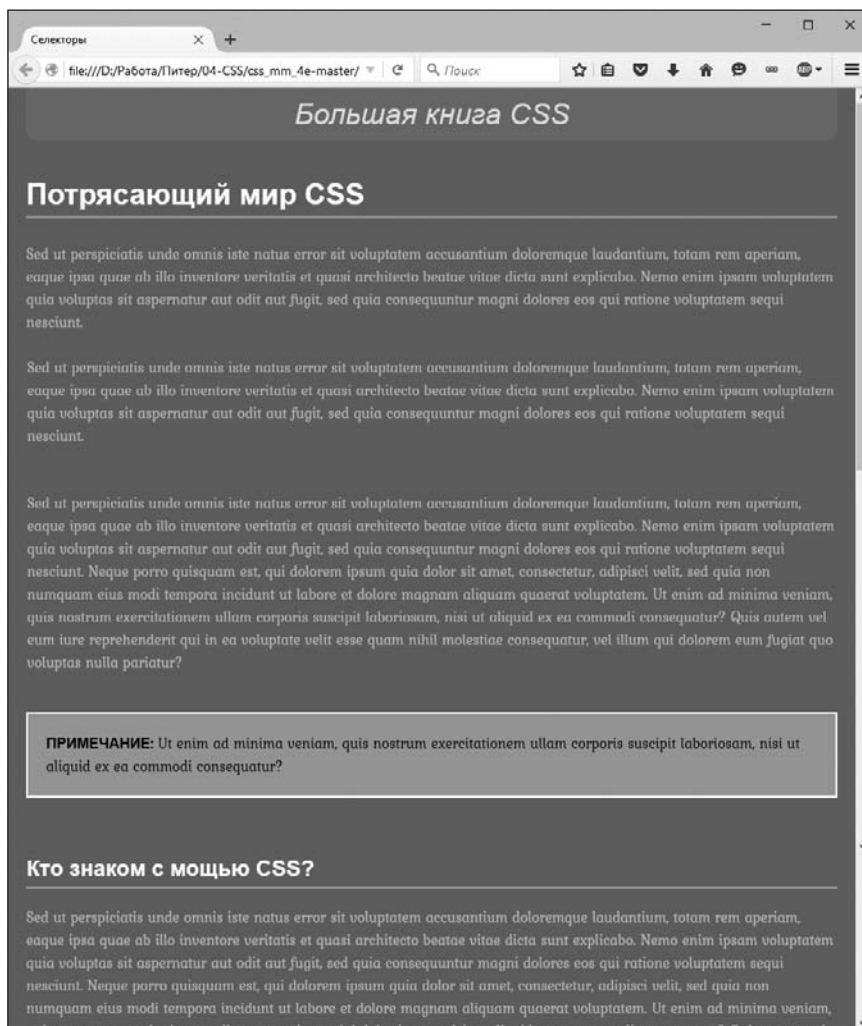
#### ПРИМЕЧАНИЕ

Несмотря на имя, которое вы присвоили классу, вы можете применить его к любым другим элементам, а не только к p. Если данное форматирование относится, например, к заголовку h2, то HTML-код должен выглядеть следующим образом:

```
<h2 class="note">
```

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

Абзац с примечанием красиво подсвечен на странице (рис. 3.13).



**Рис. 3.13.** С помощью классов вы можете выполнить ювелирное форматирование элементов веб-страницы

---

**ПРИМЕЧАНИЕ**

Если ваша веб-страница не похожа на изображенную на рис. 3.13, возможно, вы указали имя какого-либо свойства или его значение с ошибкой. Проверьте код по шагам. Кроме того, удостоверьтесь в том, чтобы каждая пара «свойство: значение» была завершена точкой с запятой и в самом конце определения стиля присутствовала закрывающая фигурная скобка. Если ваш стиль не работает должным образом, то, скорее всего, в нем не хватает именно этих символов. Это самая распространенная ошибка.

---

## Создание селекторов потомков

На странице `selectors_basics.html` вы применили класс `note` к двум абзацам. Каждый из них начинается словом **Примечание:**, выделенным полужирным начертанием (на самом деле это слово находится внутри элемента `strong`). Но что делать, если вы хотите отформатировать эти слова еще и ярко-оранжевым цветом? Вы могли бы создать стиль для элемента `strong`, но он затронет все эти элементы на странице, в то время как вы хотите изменить только те, которые находятся внутри примечаний. Одним из решений было бы создание класса, например `.noteText`, и применение его к каждому из элементов `strong` внутри примечаний. Но вы наверняка забудете применить класс, если у вас много таких страниц с примечаниями.

Лучший способ решить эту проблему — создать селектор потомков, который относится только к нужным нам элементам `strong`. К счастью, сделать это совсем не сложно.

1. Вернитесь к HTML-редактору и файлу `selector_basics.html`. Создайте новую строку для указания стиля с селектором потомков.

Щелкните кнопкой мыши после закрывающей фигурной скобки стиля `.note` и нажмите клавишу **Enter**.

2. Введите код `.note strong {`.

Последнее слово селектора — `strong` — это и есть элемент, который вы хотите отформатировать. При этом стиль отформатирует элемент `strong` только в том случае, если он расположен внутри другого элемента, к которому применен класс `.note`. Стиль не возымеет никакого эффекта на элементы `strong`, находящиеся, например, в тексте других абзацев, в списках или заголовках первого уровня.

3. Нажмите клавишу **Enter**, введите код `color: #white;`, затем нажмите клавишу **Enter** снова, чтобы создать еще одну новую строку. Закончите стиль символом закрывающей фигурной скобки.

Конечный вариант стиля должен иметь следующий вид:

```
.note strong {  
  color: white;  
}
```

4. Сохраните HTML-файл и просмотрите страницу в браузере.

Слово **Примечание:** должно быть окрашено в оранжевый цвет в каждом из примечаний на странице.

Селекторы потомков — одно из самых мощных средств языка CSS. Профессиональные веб-дизайнеры достаточно интенсивно используют их для целенаправленного форматирования отдельных элементов, при этом не засоряя HTML-код классами. В книге они применяются повсеместно, поэтому у вас будет возможность получить о них более полное представление.

## Последние штрихи

Текст на этой странице расширяется, чтобы заполнить окно браузера при изменении его размера. Чтобы увидеть данный эффект, откройте страницу и измените размер окна браузера. Обратите внимание, что по мере растягивания окна строки текста становятся шире. Если ваш монитор достаточно велик, то вы заметите, что при достижении определенной ширины строки текста слишком длинные, чтобы читать их с комфортом. К счастью, вы можете ограничить ширину контента страницы, чтобы она не превышала определенную величину.

1. Вернитесь к HTML-редактору с открытым файлом `selector_basics.html`. Создайте новую строку для нового стиля.

Если вы только что завершили предыдущие шаги, щелкните кнопкой мыши сразу после закрывающей скобки стиля `.note` и затем нажмите клавишу `Enter`.

2. Добавьте еще один стиль:

```
article {  
    max-width: 760px;  
}
```

Это еще один тип селектора. Он применяет HTML5-элемент `article`, который используется чтобы придать контенту вид записи в блоге, как показано на этой странице.

Значение параметра `max-width` определяет максимальную ширину элемента и означает, что максимальная ширина элемента статьи не будет превышать 760 пикселей. Сохраните файл и просмотрите его в браузере. Если вы растянете окно браузера шире 760 пикселей, то увидите, что по бокам от текста появился синий фон, а сам текст больше не расширяется.

С другой стороны, если вы уменьшите окно браузера, сделав его ширину меньше 760 пикселей, то строки текста сожмутся. В этом и заключается преимущество свойства `max-width` — оно позволяет ограничить максимальную ширину, не устанавливая минимальную. Это свойство очень полезно при разработке сайтов для экранов различных размеров, таких как настольные компьютеры, ноутбуки, планшеты и смартфоны. Это важный элемент *адаптивного дизайна*, с которым вы познакомитесь в главе 17.

Теперь при увеличении ширины окна браузера максимальная ширина текста ограничена. Кроме этого, было бы неплохо выровнять его по центру страницы вместо привязки к левому краю.

3. Добавьте еще одно свойство к стилю `article`. Код должен выглядеть следующим образом:



```
article {  
  max-width: 760px;  
  margin: 0 auto;  
}
```

Значение свойства `margin` определяет расстояние между элементом и другими элементами вокруг него. Подробнее о свойстве `margin` вы узнаете в разделе «Управление размерами полей и отступов» главы 7. Отмечу лишь, что в данном примере значения левого и правого поля определяются автоматически, то есть браузер сам вычисляет величину левого и правого поля элемента `article`. Когда ширина окна браузера превысит 760 пикселей, элемент `article` прекратит расширяться, поэтому браузер автоматически добавит пустое пространство слева и справа от элемента, по сути выравнивая его по центру относительно своего окна.

Сейчас вы добавите еще один дополнительный стиль — смежный родственный селектор — для форматирования абзаца, следующего сразу после первого заголовка (этого же эффекта можно достигнуть, создав класс и применив его к этому абзацу, но смежный родственный селектор не требует изменения HTML-кода).

#### 4. Добавьте последний стиль:

```
h1+p {  
  color: rgb(255,255,255);  
  font-size: 1.2em;  
  line-height: 140%;  
}
```

Он будет применен к любому абзацу, следующему *сразу* за элементом `h1`, то есть к первому абзацу после верхнего заголовка страницы. Он не будет применен ко второму или последующим абзацам. С помощью этого селектора можно легко изменить внешний вид вводного абзаца, чтобы выделить его визуально и обозначить начало статьи.

Стиль изменяет цвет и размер шрифта. Свойство `line-height` (о котором вы прочитаете в разделе «Форматирование абзацев» главы 6) определяет пространство между строками в абзацах (параметр, также известный как *интерлиньяж*).

Если вы просмотрите страницу в браузере сейчас, то увидите, что цвет текста верхнего абзаца стал белым, шрифт крупнее, а между строками увеличилось расстояние (рис. 3.14). Если вы удалите этот абзац в HTML-коде, то заметите, что оставшийся абзац также станет белого цвета и с более крупным шрифтом, поскольку теперь он будет смежным родственником элемента `h1`.

Итак, мы ознакомились с различными типами селекторов. Более подробно вы изучите их (и не только их) в практикумах следующих глав этой книги, но на текущий момент вы уже должны понимать, для чего нужны различные селекторы и почему одни должны находиться над другими.

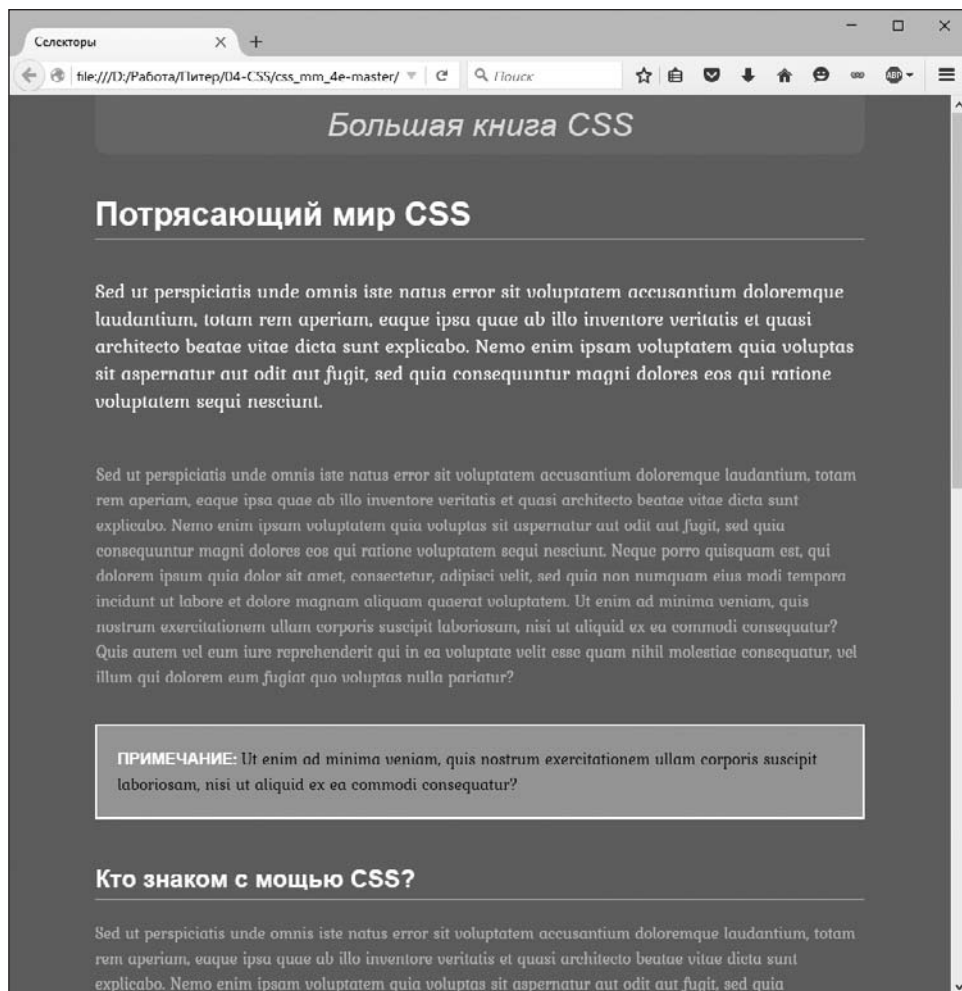
---

#### ПРИМЕЧАНИЕ

Окончательную версию созданной в этой главе веб-страницы вы можете найти в папке `03_finished` загруженного архива с примерами.

---





**Рис. 3.14.** Теперь веб-страница действительно имеет законченный вид. Настройка ширины контента и приемы типографики улучшили аскетичный вид HTML-страницы, показанной в начале практикума этой главы