Глава 12

Структура и синтаксис CSS

В этой главе...

- Использование селекторов и объявлений
- Создание правил стилей
- Исследование свойств CSS
- Наследование и каскад стилей

В этой главе будут рассмотрены правила, или синтаксис, CSS. Когда вы поймете, как CSS находит элементы и применяет к ним стили, мы поговорим о другой важной теме: каскадировании стилей. Это ключевая концепция, которая играет важнейшую роль в понимании принципов использования CSS.

Исследуем структуру и синтаксис CSS

Таблица стилей состоит из *правил стиля*. В свою очередь, каждое правило состоит из следующих двух частей.

- ✓ **Селектор**. Определяет элемент разметки, к которому применяется правило стиля.
- ✓ **Объявление**. Указывает, как должен выглядеть размечаемый контент.

Правило стиля определяется с помощью набора символов пунктуации и специальных символов. При этом используется следующий синтаксис:

```
селектор {объявление;}
```

За каждым объявлением следует точка с запятой, которая облегчает идентификацию объявлений компьютером. Как будет показано далее, одиночный селектор может включать одно или несколько объявлений. Более того, каждое объявление разбивается на следующие два подэлемента.

- ✓ **Свойства** идентифицируют параметры отображения текста и графики компьютером (например, размер шрифта или цвет фона).
- ✓ **Значения** указывают, как конкретно текст и изображения будут форматироваться на странице (например, размер шрифта 24 пункта или желтый фон).

Свойство отделяется от значения в объявлении с помощью двоеточия, а каждое объявление завершается точкой с запятой:

```
селектор {свойство: значение;}
```

В спецификации CSS указаны свойства, поддерживаемые в правилах стиля, а также различные значения, которые они могут принимать. Большинство свойств не требуют дополнительных объяснений (например, `color` и `border`). Обратитесь к главе 11, в которой приводится краткий обзор свойств CSS.

Таблицы стилей переопределяют внутренние правила отображения, заданные по умолчанию в браузере. Объявления стиля влияют на итоговое представление страницы в окне пользовательского браузера. Это означает, что можно контролировать внешний вид контента, а также создавать более согласованные и наглядные веб-страницы.

На рис. 12.1 показана простая HTML-страница, отображающая подзаголовки трех уровней (а также основной текст страницы) без использования таблицы стилей. Браузер применяет настройки, заданные по умолчанию, для отображения заголовков с использованием шрифтов разного размера.

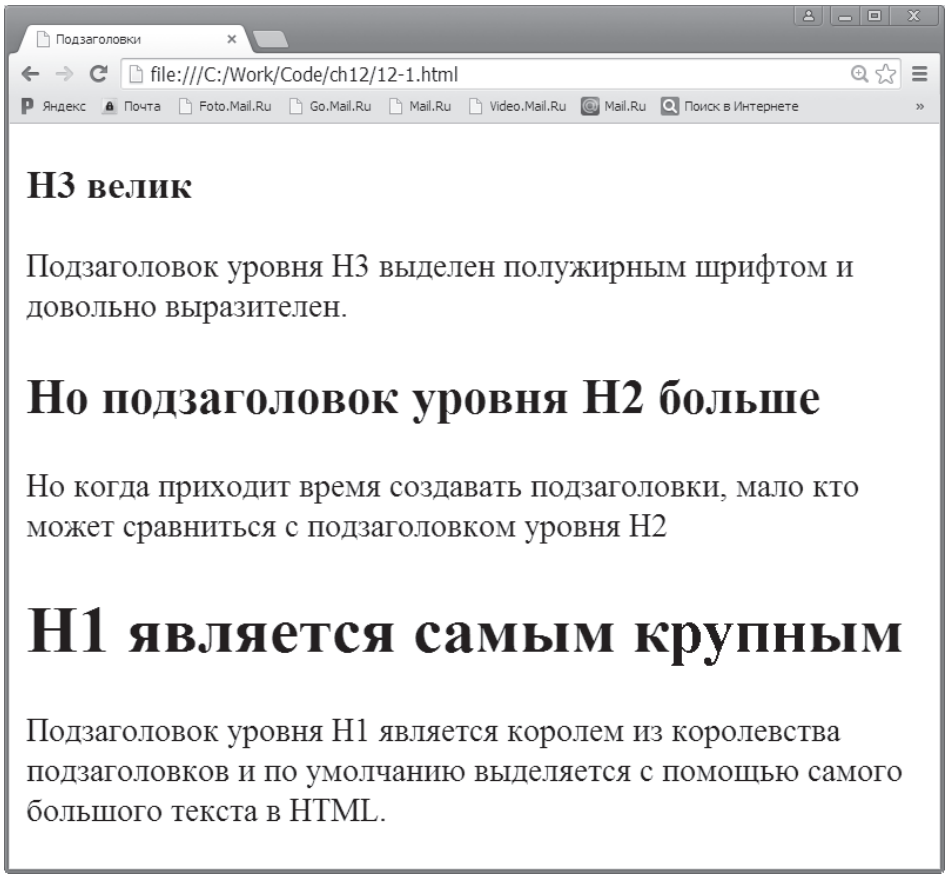

Рис. 12.1. HTML-страница, отображаемая без применения спецификаций стиля

Чтобы изменить формат страницы, примените к ней следующие правила стиля.

```
body {font-family: Arial;}

h1 {color: teal;
   font-size: 3em;}

h2 {color: maroon;
   font-size: 2em;}

h3 {color: black;
   font-size: 1.5em;}

p {font-style: italic;}
```

На рис. 12.2 показана та же веб-страница после применения стилей. Формат веб-страницы довольно сильно изменился, а именно: для основного текста веб-страницы выбран шрифт без засечек, для подзаголовков выбраны разные цвета, текст абзаца выделен курсивом, а размеры самих подзаголовков увеличены.

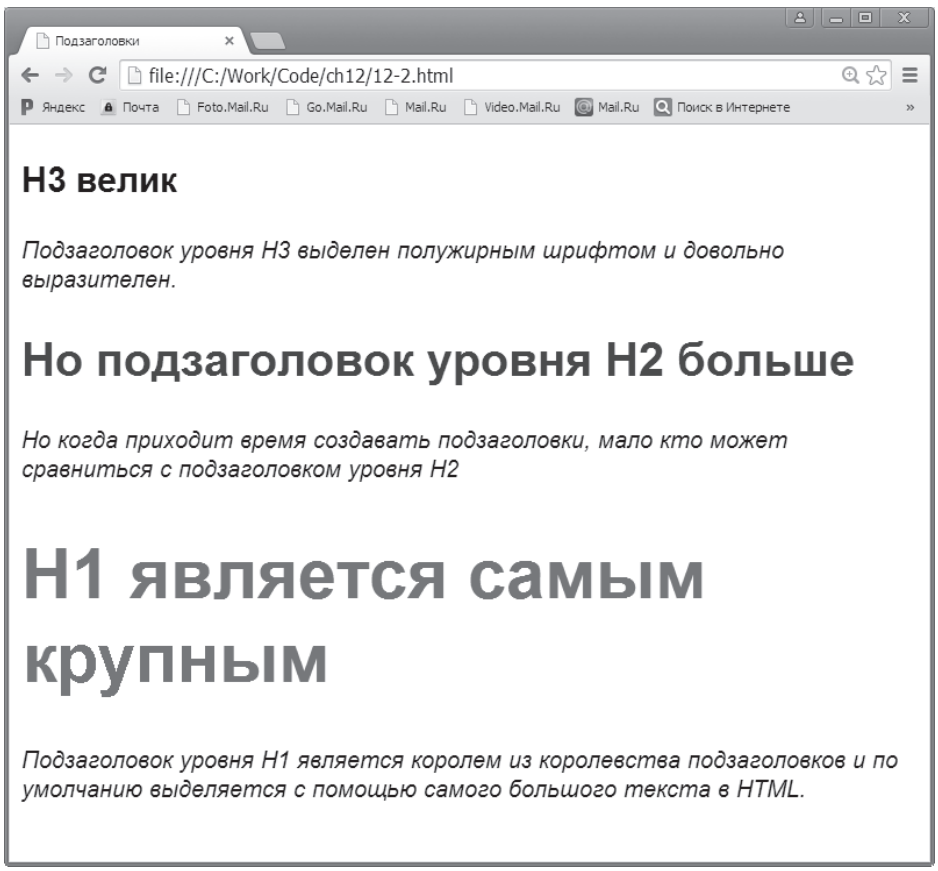

Рис. 12.2. Так выглядит HTML-страница после применения пользовательской таблицы стилей

Пользователи могут изменять настройки своих браузеров, что приведет к игнорированию применяемых таблиц стилей. Например, если пользователь испытывает трудности при чтении мелкого текста, то он может переопределить используемую таблицу стилей, чтобы отображать текст более крупным шрифтом. Если вы не предвидели возможность подобных изменений, то увеличение размера шрифта может привести к ухудшению внешнего вида сайта либо даже нарушить его работу. Протестируйте вебстраницы с отключенными таблицами стилей, чтобы убедиться в том, что они сохранили приемлемый внешний вид.

Чтобы получить подробные сведения об отключении или изменении таблиц стилей, посмотрите дискуссию “Reading Web Pages without CSS”, поддерживаемую Джимом Тетчером и доступную по следующему адресу:

`http://www.jimthatcher.com/webcourseb.htm`

Инструкции могут изменяться в зависимости от применяемого браузера. Чтобы управлять таблицами стилей или отключать их, воспользуйтесь вспомогательными подключаемыми модулями.

Семейство шрифтов

Для присваивания значений свойству `font-family` можно воспользоваться названиями шрифтов, разделенными запятыми. Эти названия должны соответствовать шрифтам, доступным для браузера пользователя. Если название шрифта включает пробелы, например Times New Roman, то оно заключается в кавычки.

```
h1 {font-family: Verdana, "Times New Roman", serif;}
```

Это правило сначала задает использование шрифта Verdana. Если этот шрифт недоступен, то браузер будет искать шрифт Times New Roman, а в случае недоступности этого шрифта будет использован обобщенный шрифт с засечками. Использование шрифтов рассмотрено в главе 17.

Селекторы и объявления

Иногда требуются правила стиля, которые влияют на отображение нескольких свойств для произвольного выбранного селектора. Можно создать несколько правил стиля для единственного селектора, для каждого из которых используется отдельное объявление.

```
h1 {color: teal;}
h1 {font-family: Arial;}
h1 {font-size: 3em;}
```

Но при наличии огромной коллекции правил стилей затрудняется управление ими. В CSS допускается комбинирование нескольких объявлений в виде единственного правила стиля, которое оказывает влияние на несколько характеристик, определяющих отображение единственного селектора, как показано в следующем примере кода.

```
h1 {color: teal;
   font-family: Arial;
   font-size: 3em;}
```

Все объявления, относящиеся к селектору `h1`, находятся внутри одного и того же блока фигурных скобок (`{}`) и разделены точками с запятой (`;`). В правило стиля можно включить произвольное количество объявлений, а каждое объявление завершается точкой с запятой.

С технической точки зрения пробелы в таблицах стилей игнорируются (как и в HTML), но все равно желательно выравнивать строки пробелами, чтобы облегчить чтение и редактирование таблиц стилей. Исключение из этого правила имеет место в том случае, когда вы объявляете несколько названий шрифтов в свойстве `font-family`. Чтобы получить дополнительные сведения по этой теме, обратитесь к врезке “Семейство шрифтов”.

Можно создать один и тот же набор объявлений, применяемый к коллекции селекторов, разделяемых запятыми. Следующие правила стиля применяют объявления для цвета текста, семейства шрифтов и размера шрифта к селекторам `h1`, `h2` и `h3`.

```
h1, h2, h3 {color: teal;
            font-family: Arial;
            font-size: 2.5em;}
```

Поскольку синтаксис таблиц стилей основан на использовании знаков пунктуации, следите за появлением двоеточий и запятых, иначе правила стиля могут работать не так, как вы ожидаете. Во избежание недоразумений убедитесь, что вместо двоеточия не используется точка с запятой, а вместо фигурных скобок — круглые скобки. Следите также за точками с запятой. Используйте инструментальные средства верификации, которые помогут отслеживать наличие подобных проблем. В частности, можно воспользоваться сервисом верификации W3C CSS, доступным по следующему адресу:

`http://jigsaw.w3.org/css-validator`

Селекторы

Прежде чем применить стиль к элементу, нужно указать браузеру, о каком элементе идет речь. В этом случае используются селекторы. В CSS доступно несколько способов выбора элементов, которые ранжируются от очень широкого (выбор всех элементов) до самого узкого (выбор лишь одного экземпляра элемента).

Универсальный селектор

Среди всех селекторов наиболее простым является *универсальный селектор*. Он соответствует любому типу элемента. Для обозначения универсального селектора используется символ звездочки (`*`). Ниже приводится универсальный селектор, с помощью которого устанавливаются поля нулевой ширины для каждого элемента в документе:

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

Универсальный селектор может использоваться в качестве замены сложных селекторов, которые иногда требуются для получения доступа к элементам, вложенным глубоко в иерархии документа. Например, предположим, что вы готовитесь к Дню Святого Патрика и хотите окрасить каждый элемент в зеленый цвет. Для выполнения этой задачи можно переопределить базовый цвет каждого элемента, используя универсальный селектор:

```
* {color: green;}
```

Универсальный селектор может применяться для решения очень сложных задач. Чтобы убедиться в широких возможностях универсального селектора, выполните следующие действия.

1. **Откройте окно редактора Aptana Studio.**
2. **Откройте файл `12-3.html`, который можно загрузить на сайте книги:**

   `http://www.dialektika.com/Books/978-5-8459-2035-5.html`

   Перед вами появится начальная страница сайта "Кафе HTML", для формирования которой используется разметка, приведенная в листинге 12.1.

Листинг 12.1. HTML-разметка для начальной страницы сайта "Кафе HTML5"

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8">
      <title>Кафе HTML5: начальная страница</title>
      <meta name="description" content=
         "Пример сайта">
      <meta name="viewport" content="width=device-width">
      <link rel="stylesheet" href="css/normalize.css">
      <link rel="stylesheet" href="css/main.css">
   </head>
   <body>
      <div id="container">
         <nav id="topnav">
            <a href="index.html">ДОМОЙ</a> |
               <a href="about.html">О НАС</a> |
               <a href="menu.html">МЕНЮ</a> |
               <a href="contact.html">КОНТАКТЫ</a>
         </nav>
         <div id="content">
            <h1>Добро пожаловать в кафе HTML5</h1>
            <p>Здесь вы найдете все сорта кофе, приготовленного на
               основе HTML5 и CSS3.
            </p>
            <figure id="home-image">
            <img src="img/pitr_Coffee_cup_icon.png"
               width="400" height="400" alt="ароматный кофе">
            <figcaption class="warning">
               Сила в кофе
```

```
            </figcaption>
         </figure>
         </div>
         <footer>
            p class="copyright">Все права защищены &copy;
               dummieshtml.com</p>
         </footer>
      </div>
   </body>
</html>
```

Если на этом этапе открыть начальную страницу в окне браузера, то она примет вид, показанный на рис. 12.3.

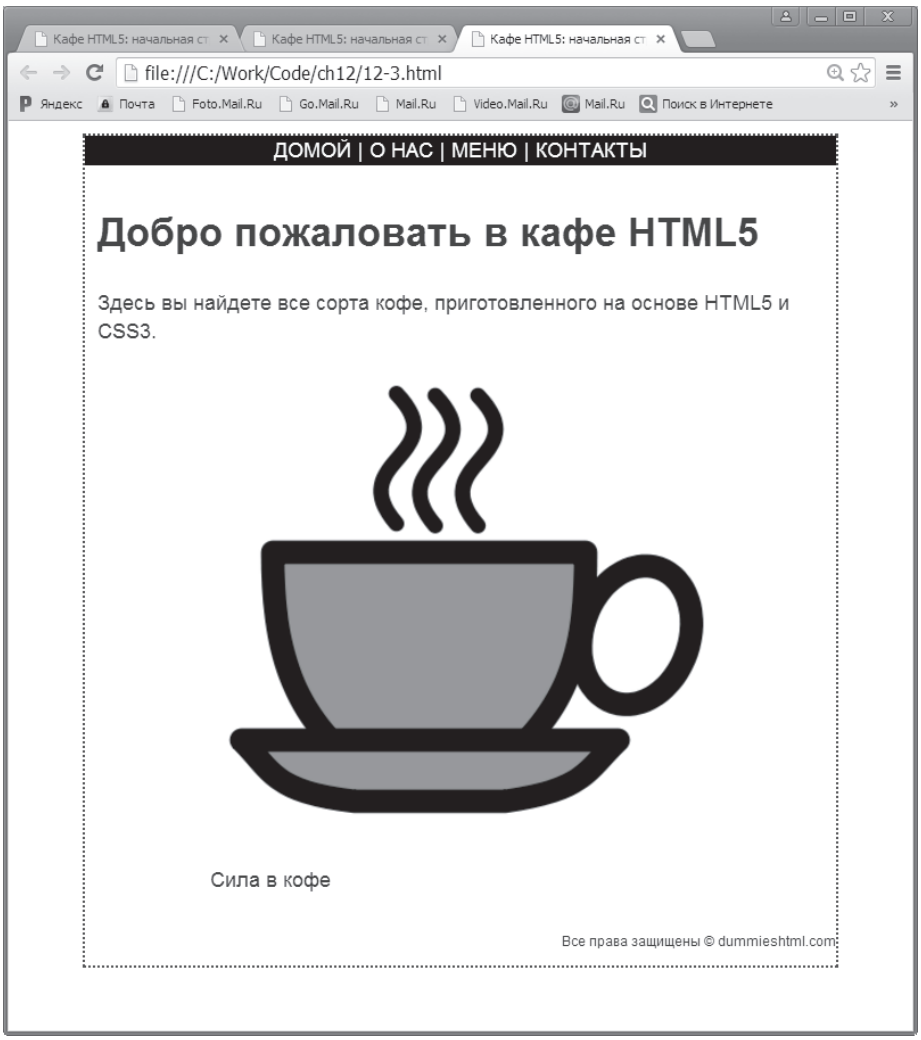

Рис. 12.3. Начальная страница сайта "Кафе HTML5"

3. **Вставьте новую строку непосредственно перед тегом `</head>`.**
4. **Введите следующий код.**

```
<style>
   * {color: green;}
</style>
```

5. **Сохраните изменения в файле под названием `12-4.html` и откройте его в окне браузера.**

   Вы увидите изображение, подобное показанному на рис. 12.4.

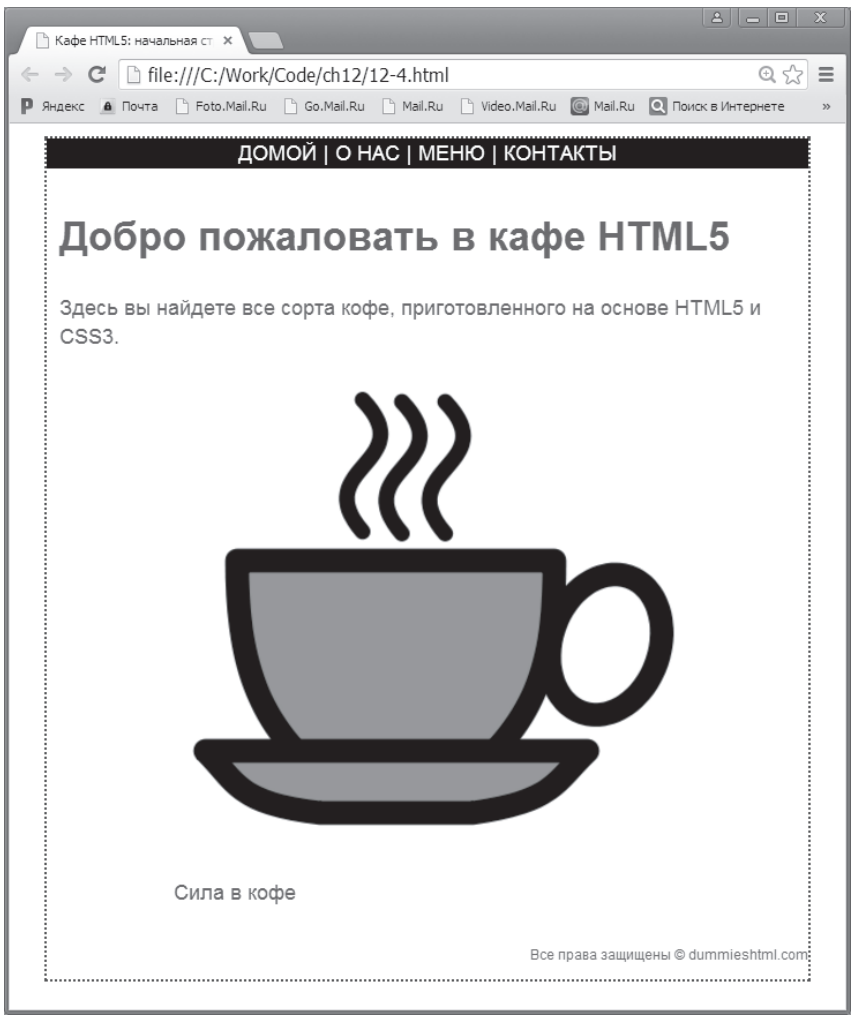

Рис. 12.4. Теперь текст выделяется зеленым цветом

На самом деле на веб-странице, показанной на рис. 12.4, появилось не так уж и много окрашенных в зеленый цвет областей. Например, граница вокруг страницы и

навигационные ссылки сохранили прежние цвета. Причина подобного положения дел заключается в следующем определении универсального селектора:

Универсальный селектор выбирает произвольный тип элемента.

Другими словами, правила стиля вместе с универсальным селектором применяются к каждому элементу, а произвольный элемент в документе может быть стилизован по-разному, если применяются другие, более специфичные селекторы. Именно это и произошло в данном случае.

А теперь рассмотрим более интересный пример универсального селектора. При выполнении этого примера мы добавим эффект трансформации CSS в каждый элемент документа.

1. **Откройте файл `12_4.html` в окне редактора Aptana Studio и удалите ранее созданное правило CSS (но сохраните открывающий и закрывающий теги `<style>` и `</style>` соответственно).**
2. **Между тегами `<style>` и `</style>` введите следующее правило.**

```
* {-webkit-transform: rotate(100deg);
   -moz-transition: rotate(100deg);
   -ms-transition: rotate(100deg);
   -o-transition: rotate(100deg);}
```

   Это правило приводит к повороту каждого элемента в документе на 100º. При просмотре документа, сохраненного под именем `12-5.html`, в окне браузера вы увидите картинку, подобную показанной на рис. 12.5.

Универсальный селектор является наиболее грубым инструментом из нашего арсенала. Для выполнения более точной работы с помощью CSS используются дополнительные селекторы.

Иногда встречаются объявления CSS, содержащие пары *`свойство`*`:` *`значение`*, которые идентичны, за исключением нескольких символов, находящихся после начального дефиса, как показано в следующем примере.

```
-webkit-transform: rotate(100deg);
-moz-transition: rotate(100deg);
-ms-transition: rotate(100deg);
-o-transition: rotate(100deg);
```

Эти символы, такие как `webkit` и `moz`, находящиеся в начале имени свойства, называются *префиксами браузера*. Поскольку стандартизация некоторых свойств CSS3 до сих пор не завершена, разработчики браузеров создали собственные версии свойств CSS3 и присвоили им названия с помощью префиксов браузера, указав на то, что используются нестандартные версии свойств.

Каждое объявление, использованное в предыдущем правиле, предназначено для отдельного браузера. Первый префикс, `-webkit`, используется при работе с браузерами Google Chrome и Apple Safari (и некоторыми другими). Второй префикс, `-moz`, используется браузером Mozilla Firefox. Префикс `-o` относится к браузеру Opera. Ну а префикс `-ms` предназначен для Internet Explorer.

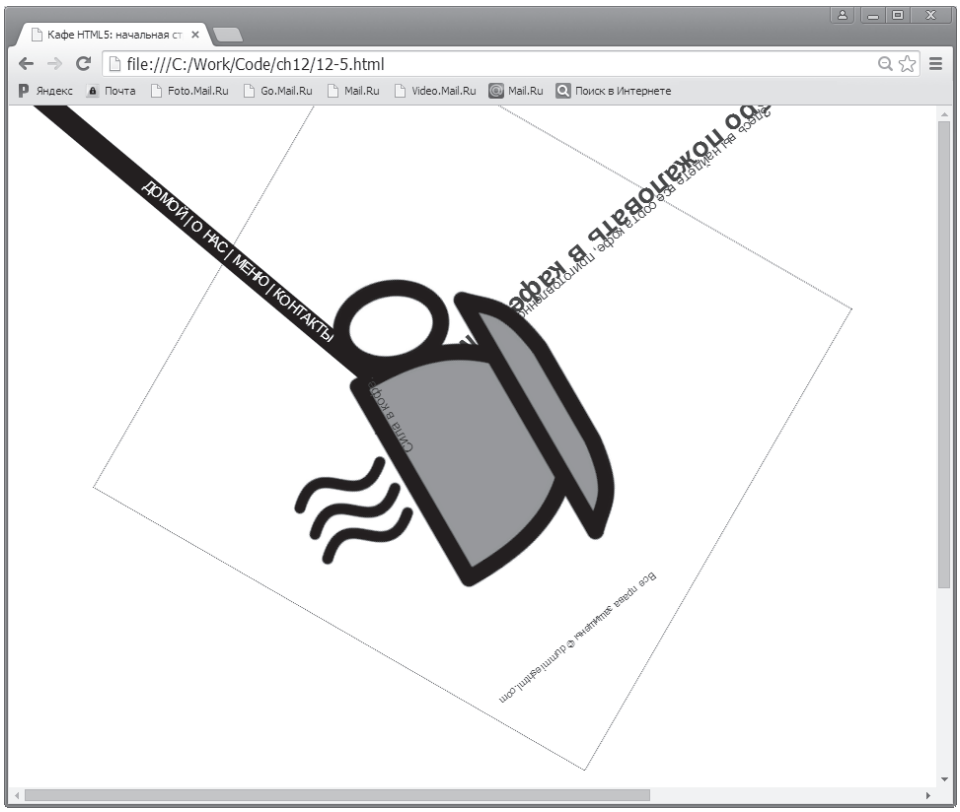

Рис. 12.5. Результат поворота веб-страницы в окне браузера

Обычно свойства, специфичные для браузеров, работают схожим образом. После того как свойства CSS3 получат статус стандарта, префиксы браузера станут ненужными. В настоящее же время настоятельно рекомендуется использовать префиксы браузера для определенных свойств CSS3.

Селекторы типа элемента

Селекторы типа элемента применяются для выбора элемента определенного типа. Например, может понадобиться изменить размер каждого элемента `h1` в документе либо изменить стиль шрифта в абзацах. Ниже приводится пример селектора типа элемента в составе правила, изменяющий размер шрифта элемента `h1` на `3em`.

```
h1 {font-size: 3em;}
```

Селектор типа элемента работает так же, как универсальный селектор, но применяется только к одному типу элемента. Например, если вы хотите просто повернуть изображение кофейной чашки на начальной странице сайта “Кафе HTML5”, то измените универсальный селектор из предыдущего примера на `img`. Ниже приведено новое объявление.

```
img {-webkit-transform: rotate(100deg);
     -moz-transition: rotate(100deg);
     -ms-transition: rotate(100deg);
     -o-transition: rotate(100deg);}
```

Новый вариант начальной страницы сайта “Кафе HTML5” показан на рис. 12.6.

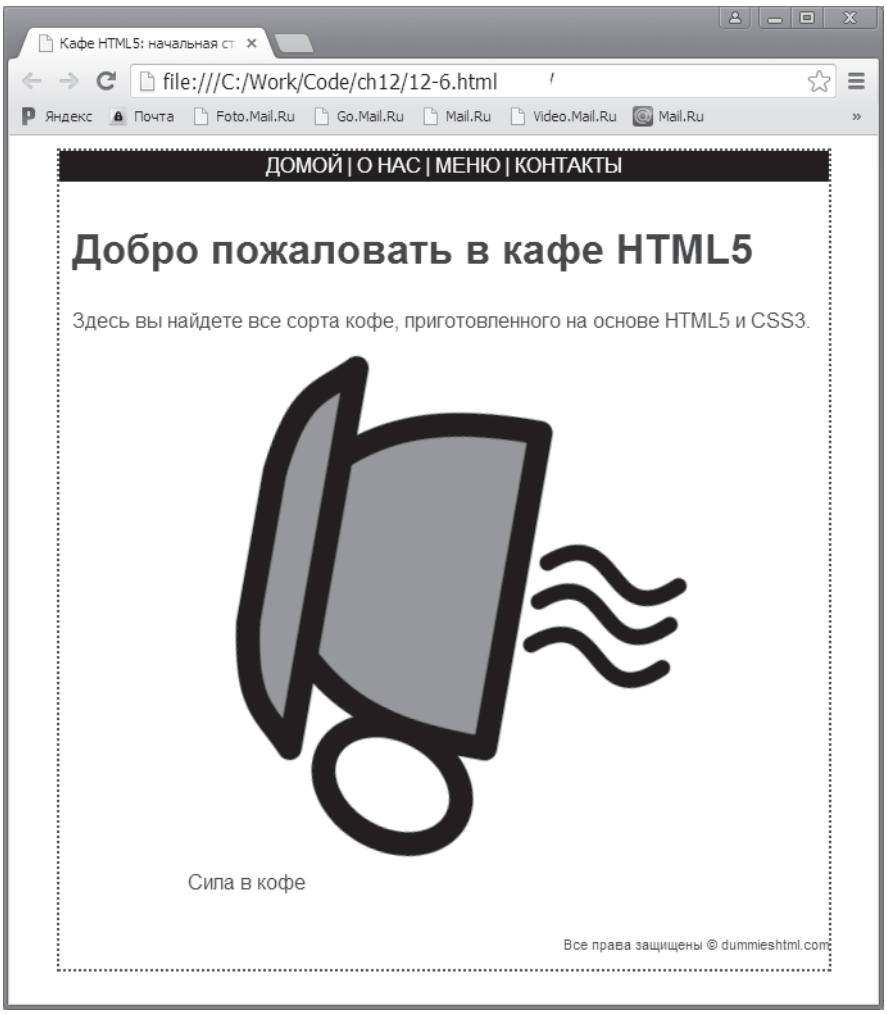

Рис. 12.6. Новый вариант вращающейся чашки кофе

Но что делать в том случае, если на странице есть несколько изображений и вы хотите повернуть лишь одно из них? Для выполнения этой задачи нужен специфический селектор.

Селекторы идентификаторов

Селекторы идентификаторов применяются для выбора элемента в соответствии с его идентификатором. Поскольку атрибут `ID` является уникальным в пределах HTML-документа, селектор идентификатора идеален для выбора нужного элемента.

Изучив HTML-разметку начальной страницы "Кафе HTML5", вы увидите, что она включает два элемента `div`. Один из этих элементов охватывает весь контент элемента `body` и имеет идентификатор `container`. Другой элемент окружает лишь область центрального контента (кроме области навигации и нижнего колонтитула) и имеет идентификатор `content`.

Чтобы использовать селектор идентификатора, перед значением выбираемого идентификатора укажите символ `#`. Например, если нужно изменить фоновый цвет в разделе `div` с идентификатором `content`, то используйте следующее объявление:

```
#content {background-color: aquamarine;}
```

Селекторы классов

Иногда требуются правила стиля, применяемые только к выбранным экземплярам элемента HTML-разметки. Например, если необходимо стилевое правило, применяемое к абзацам, содержащим сведения об авторском праве, то нужно указать браузеру, что правило имеет ограниченную область применения.

Чтобы точнее "нацелить" правило стиля, вместе с элементом разметки используйте атрибут `class`. Следующие примеры демонстрируют код HTML, используемый для формирования двух разновидностей абзацев.

- ✓ Обычный абзац (без атрибута `class`).

```
<p>Обычный абзац.</p>
```

- ✓ Атрибут `class`, принимающий значение `copyright`.

```
<p class="copyright">Абзац, заданный с помощью атрибута class,
          принимающего значение &copy; 2013.</p>
```

Чтобы создать правило стиля, которое применяется только к абзацу, включающему символ авторского права, дополните селектор абзаца в правиле стиля, поставив после него:

- ✓ точку (`.`);
- ✓ значение атрибута `class`, такое как `copyright`.

Результирующее правило выглядит так.

```
p.copyright {font-family: Arial;
             font-size: 12px;
             color: white;
             background: teal;}
```

Комбинируя селектор типа элемента (`p`) с селектором класса (`.copyright`), можно задать применение следующих правил только к тем элементам `p`, которые включают

этот атрибут класса. Такое правило стиля означает, что все абзацы с классом `copyright` отображают белый текст на зеленовато-голубом фоне. Кроме того, выбран шрифт Arial размером 12px.

Чтобы протестировать только что созданный класс `copyright`, добавьте тег абзаца с классом `copyright` в область сведений об авторском праве, находящуюся в нижнем колонтитуле страницы "Кафе HTML5" в файле `12-6.html`, а затем сохраните файл под именем `12-7.html`.

```
<footer>
   <p class="copyright">Все права защищены &copy; dummieshtml.com</p>
</footer>
```

Как показано на рис. 12.7, браузер применяет этот стиль только к абзацу, в котором `class` принимает значение `copyright`. К другим абзацам на этой странице стиль не применяется.

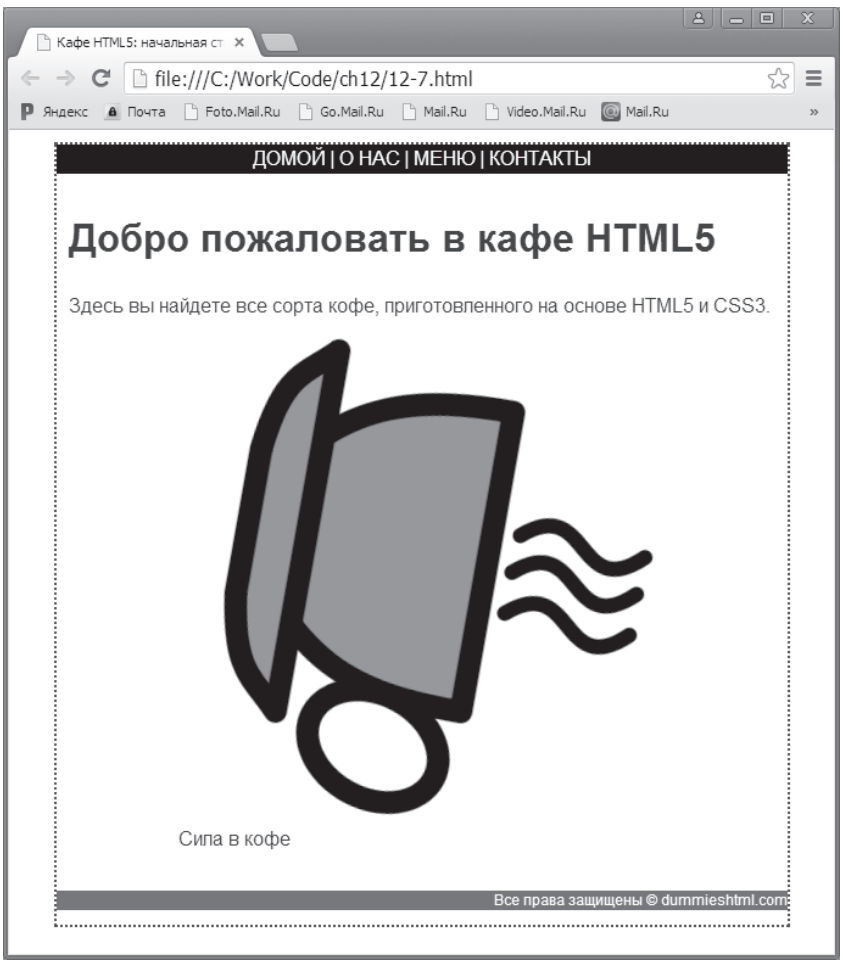

Рис. 12.7. С помощью классов можно более точно применять стили

Можно также создавать классы “стиль-правило”, которые не связаны с каким-либо элементом, как в следующем примере.

```
.warning {color: red;
          font-weight: bold;
          font-size: 1.5em;}
```

Вы можете использовать этот стилевой класс вместе с произвольным элементом, добавив к такому элементу конструкцию `class="warning"`. На рис. 12.8 показано, каким образом браузер применяет стиль предупреждения к абзацу и заголовку, но не к цитате. Ниже приводится соответствующая разметка HTML.

```
<p>Абзац, к которому не применялся класс warning.</p>
<blockquote>Цитата без определенного класса.</blockquote>
<h1 class="warning">Предупреждения</h1>
<p class="warning">Абзац, к которому применен класс
   warning.</p>
```

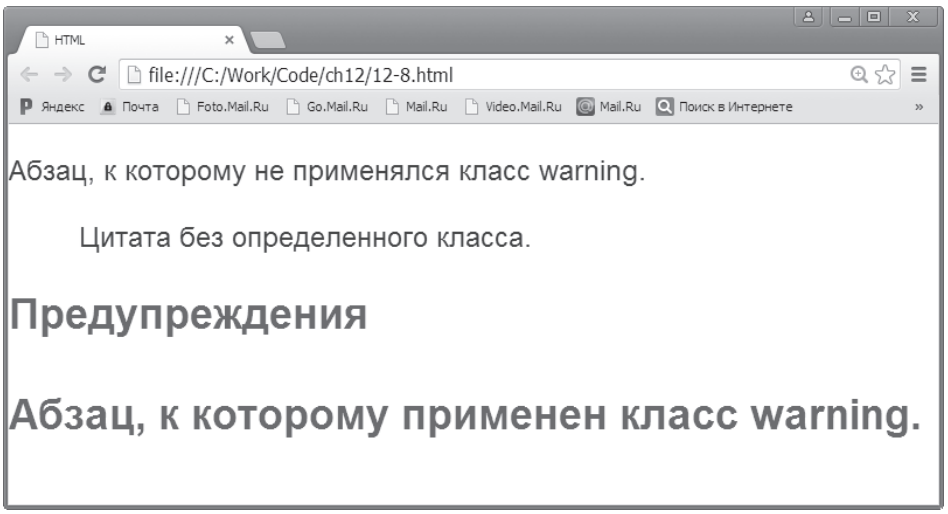

Рис. 12.8. Воспользуйтесь селекторами класса для создания правил стиля, которые могут применяться к любому элементу

Можно также использовать элемент `span` для избирательного применения пользовательских стилей к строчному контенту (либо создать контейнеры для произвольного контента, которые расположены от открывающего тега `<span>` до закрывающего тега `</span>`).

```
<p>Абзац без <span class="warning">класса warning
   </span>, примененного к словам "класс warning".</p>
```

Чтобы увидеть это объявление в действии, добавьте его в тег `<style>` в файле `index.html` и поместите конструкцию `class="warning"` в открывающий тег одного из элементов, содержащих текст в файле `index.html`.

```
<figcaption class="warning">Сила в кофе</figcaption>
```

При просмотре документа в окне браузера текст, находящийся внутри тега <figcaption>, окрашивается в красный цвет и выделяется крупным полужирным шрифтом (рис. 12.9).

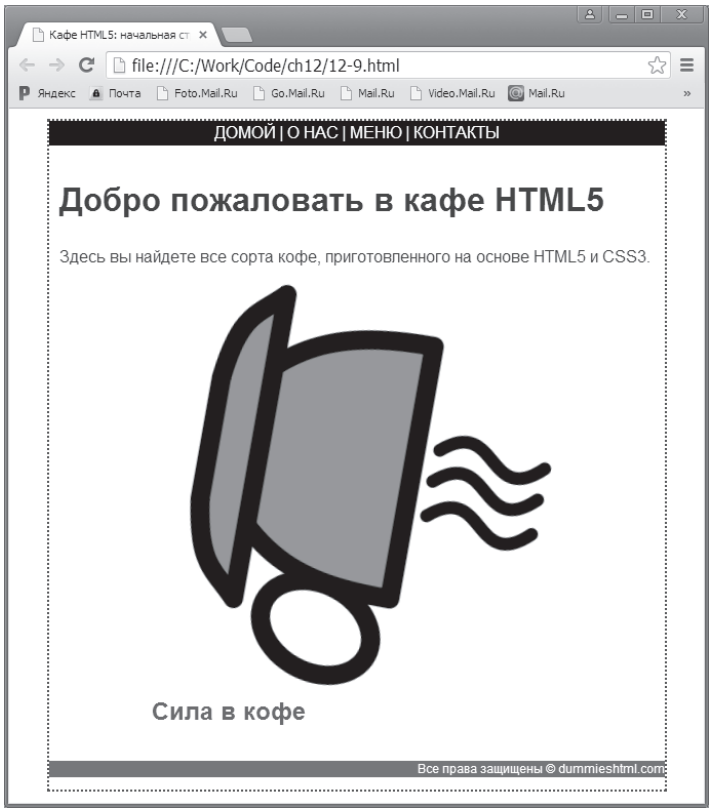

Рис. 12.9. Использование селектора класса для применения стиля к элементу

Другие селекторы

В состав CSS включены другие типы селекторов, которые немного более сложные и реже используемые, чем четыре селектора, рассмотренные в предыдущих разделах. Эти селекторы перечислены ниже.

- ✓ **Селектор соседей.** Выбирает элементы только в том случае, если они следуют за другим выбранным элементом:

  ```
  h1 + p {font-style: italic; }
  ```

  Предыдущее правило применяется к произвольному элементу `p`, который непосредственно следует за элементом `h1`.
- ✓ **Селектор атрибутов.** Выбирает элементы на основе значений их атрибутов:

  ```
  img[src="chris.jpg"] {border: 6px solid purple;}
  ```

Это правило создает толстую пурпурную границу вокруг любого элемента `img` с атрибутом `src`, значение которого равно `chris.jpg`.

- ✓ **Селектор потомков.** Выбирает элементы, которые являются непосредственными потомками другого элемента. Например, в следующем коде HTML элемент `em` является непосредственным потомком элемента `<p>`, но не элемента `<a>`.

```
<p><em>Важно</em> запомнить это <a
        href="picture.jpg">Рисунок</a>.</p>
```

Чтобы использовать селектор потомков, введите `>` между предком и непосредственным потомком, который вы хотите использовать в качестве цели:

```
p>em {color:red;}
```

- ✓ **Селектор наследников.** Выделяет элементы, которые в дереве документа находятся ниже текущего элемента. Например, следующее правило выбирает любой элемент `p`, который вложен в элемент `footer`:

```
footer p {font-size: 50%;}
```

Обратите внимание на то, что селекторы наследников применяются ко всем наследникам, которые соответствуют селектору, а не только к первому наследнику. Поэтому предыдущий пример соответствует элементу `p` в следующем коде HTML:

```
<footer><p>copyright 2013</p></footer>
```

а также элементу `p` в данном коде HTML:

```
<footer><div><p>copyright 2013</p></div></footer>
```

- ✓ **Псевдоклассы.** Выбирают элементы, основываясь на свойствах, отсутствующих в документе дерева. Например, с помощью псевдокласса можно применить стиль к элементу после установки указателя мыши над ним либо изменить стиль ссылки, которая была предварительно посещена:

```
a:hover{text-decoration:none;}
```

В этом примере подчеркивание было удалено из ссылок после того, как пользователь выбрал ссылку с помощью указателя мыши.

Ниже приводится список возможных псевдоклассов.

- `:first-child`
- `:link`
- `:visited`
- `:hover`
- `:active`
- `:focus`
- `:lang(n)`

Синтаксис, применяемый для определения значений CSS

В спецификации CSS, разработанной консорциумом World Wide Web (W3C), определяются синтаксис и возможные значения для всех свойств CSS. Чтобы создать наиболее точные определения, консорциум W3C использует формальную схему, которая называется *синтаксис определения значений CSS*.

Воспринимать подобный синтаксис не так уж и сложно. Например, вот синтаксис для свойства `margin`, задающего поля элемента:

```
[ <длина> | <процент> | auto ]{1,4}
```

Ниже приводится пример корректного использования свойства `margin`.

```
margin: 0px;
margin: 1px 10%;
margin: 1px 0px auto;
margin: 1px 0px 1px 0px;
```

Ниже приводится краткое руководство по чтению синтаксиса определений.

- ✓ Если ключевое слово отображается без кавычек, значит, оно может использоваться в исходном виде, как слово `auto` в приведенном выше примере.
- ✓ Базовые типы данных CSS находятся между угловыми скобками < и >. Например, <*время*>, <*цвет*> и <*uri*> — это базовые типы, которые определены повсеместно.
- ✓ Два амперсанда (`&&`) означают, что значения, разделенные этим символом, являются обязательными, но могут отображаться в произвольном порядке.
- ✓ Квадратные скобки (`[]`) группируют значения в одно целое.
- ✓ Значения, разделяемые символами `||`, необязательны. Одно из этих значений должно отображаться, причем значения могут выводиться в произвольном порядке.
- ✓ Символ `|` означает, что параметры являются взаимоисключающими — лишь один из них должен отображаться.
- ✓ Символ звездочки (`*`) означает, что значение может не отображаться, отображается один раз или несколько раз.
- ✓ Знак плюс (+) означает, что значение должно появляться один раз или несколько раз.
- ✓ Вопросительный знак (`?`) указывает на то, что значение является необязательным и может отображаться лишь один раз.
- ✓ Если вы увидите элемент <*длина*>, значит, допустимо значение с фиксированной длиной, например `10px`.
- ✓ Если вы увидите элемент <*процент*>, значит, допустимо процентное значение, такое как `10%`.

- ✓ Числа, заключенные в фигурные скобки, означают, что минимальное количество появлений свойства задается первым числом, а максимальное количество — вторым числом. Например, запись `{1, 4}` означает, что свойство должно появляться как минимум один раз и как максимум четыре раза.
- ✓ Символ # означает, что значение должно повторяться один или несколько раз. Каждое значение должно отделяться запятой.

Наследование стилей

Базовая концепция, используемая в HTML (и в языках разметки в целом), — вложение тегов.

- ✓ Каждый корректный HTML-документ вкладывается в теги `<html>` и `</html>`.
- ✓ Контент, отображаемый в окне браузера, вкладывается в теги `<body>` и `</body>`.

В спецификации CSS подчеркивается, что в пользовательской разметке элементы часто оказываются вложенными, поэтому необходимо гарантировать, что стили, связанные с родительским элементом, будут применены и к элементу-потомку. Такой механизм называется *наследованием*.

После присваивания стиля элементу этот же стиль применяется и ко всем вложенным элементам. Например, правило стиля для элемента `body`, которое устанавливает фон страницы, цвет текста, размер шрифта, семейство шрифтов и поля, выглядит следующим образом.

```
body {background: teal;
      color: white;
      font-size: 18px;
      font-family: Garamond;
      margin-left: 72px;
      margin-right: 72px;
      margin-top: 72px;}
```

Чтобы установить правила стилей для всего документа, сначала установите их для элемента `body`. При использовании такого способа изменение шрифтов для всей страницы не составит особого труда, поскольку все элементы изменяются одновременно.

Если подключить к следующему фрагменту HTML-кода показанное выше правило, которое применяется к элементу `body`, то заданное форматирование будет унаследовано всеми подчиненными элементами.

```
<body>
   <p>Этот абзац наследует стили страницы</p>
   <h1>Как и этот заголовок</h1>
   <ul>
```

```
      <li>Как и элементы данного списка</li>
      <li>Элемент</li>
      <li>Элемент</li>
   </ul>
</body>
```

Уделите внимание наследованию

При создании сложных таблиц стилей, управляющих отображением каждого аспекта страницы, учитывайте особенности наследования. Например, если в правиле стиля для элемента `body` установить поля, то на основе этой настройки будут заданы поля для всех элементов страницы. Если вы знаете, как функционируют правила стиля, то благодаря наследованию можно минимизировать повторения правил и получить согласованный внешний вид страницы.

В этой главе рассматривается базовый синтаксис CSS, но существует множество профессиональных приемов настройки стилевых правил. Полный обзор синтаксических правил CSS приведен в руководстве “CSS Structure and Rules”, написанном организацией Web Design Group и доступном по следующему адресу:

`http://www.htmlhelp.com/reference/css/structure.html`

Понятие о каскаде

На элементы страницы могут влиять несколько таблиц стилей, созданных одна поверх другой. Все это напоминает наследование стилей веб-страницы и называется *каскадированием* в CSS.

Каскадирование определяет, каким образом CSS обрабатывает ситуации, когда два или больше правил стиля имеют объявления, применяемые к одному и тому же элементу или свойству. Следует учитывать три принципа каскадирования.

- ✓ **Происхождение**. Указал ли пользователь одно из правил стилей? Если да, то это наиболее важное качество, которое поможет выиграть в конфликте.
- ✓ **Специфичность**. Селекторы идентификаторов “побеждают” селекторы класса. В свою очередь, селекторы класса “побеждают” селекторы типа элементов.
- ✓ **Близость**. Насколько близок стиль к элементу, по отношению к которому будет применяться? В следующей главе будет показано, что есть три места, в которых могут определяться правила стилей. Выбирается стиль, который ближе всего к элементу либо который читается браузером в последнюю очередь.

Например, ниже представлен фрагмент простой HTML-разметки, включающей пару элементов.

```
<body>
   <div id="content">
      <p class="information">Это контент.</p>
```

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

А вот часть таблицы стилей, применяемая к этой разметке.

```
body {font-family: Arial;}
div {color: green;}
p {font-size: 18px;
   color: blue;}
.information {font-size: 16px;}
```

Обратите внимание на то, что эти правила применяют два разных размера шрифта и два разных цвета к элементу `p`. Браузеру приходится разрешать этот конфликт, используя принципы происхождения, специфичности и близости. В данном случае после разрешения всех конфликтов к тексту, находящемуся внутри элемента `p`, применялись следующие правила.

```
font-family: Arial;
color: blue;
font-size: 16px;
```

Даже если в этом примере применяются два разных размера шрифта и два разных цвета, текст может быть окрашен только в один цвет и набран шрифтом единственного размера. Применяя три принципа каскадирования, браузер разрешает конфликт, выбирая “победившие” стили.