HIGH TECH

РЕЙЧЕЛ ЭНДРЮ

3 издание

CSS

100 и 1 совет

СИМВОЛ®

The CSS Anthology

101 Essential Tips, Tricks & Hacks

Third Edition

Rachel Andrew

sitepoint

HIGH TECH

CSS

100 и 1 совет

Третье издание

Рейчел Эндрю

Символ®

Санкт-Петербург — Москва
2010

Серия «High tech»

Рейчел Эндрю

CSS: 100 и 1 совет, 3-е издание

Перевод А. Минаевой

Главный редактор	*А. Галунов*
Зав. редакцией	*Н. Макарова*
Выпускающий редактор	*П. Щеголев*
Научный редактор	*В. Коршунов*
Редактор	*Ю. Бочина*
Корректор	*Е. Кирюхина*
Верстка	*К. Чубаров*

Эндрю Р.
CSS: 100 и 1 совет, 3-е издание. – Пер. с англ. – СПб: Символ-Плюс, 2010. – 336 с., ил.

ISBN 978-5-93286-182-0

«CSS: 100 и 1 совет» представляет интерес для веб-дизайнеров и разработчиков, у которых нет времени на штудирование огромного количества теоретических материалов при создании собственного сайта. Это сборник готовых решений наиболее распространенных проблем, которые можно сразу применить на практике; более того, они могут послужить основой для разработки собственных методов. Здесь представлены ответы на сложные вопросы и практические методы использования CSS, примеры создания сложных макетов страниц, элементов навигации и форм, решение проблем, связанных с особенностями различных броузеров.

Чтение книги не требует специальной подготовки: первая глава содержит обзор основных особенностей CSS, но в дальнейшем сложность приводимых методов постепенно возрастает, так что наличие у читателя базовых знаний CSS существенно облегчит восприятие материала.

Третье издание полностью пересмотрено и обновлено с целью охвата новейших технологий и особенностей самых современных броузеров, включая Firefox 3 и Internet Explorer 8. Данную книгу можно использовать как справочник для поиска подходящего решения при создании сайта и тем самым выиграть время и сдать проект в срок.

ISBN 978-5-93286-182-0
ISBN 978-0-9805768-0-1 (англ)

Издательство «Символ-Плюс». 199034, Санкт-Петербург, 16 линия, 7,
тел. (812) 324-5353, www.symbol.ru. Лицензия ЛП N 000054 от 25.12.98.

Подписано в печать 22.04.2010. Формат 70×100 $^{1}/_{16}$. Печать офсетная.
Объем 21 печ. л. Тираж 1500 экз. Заказ №
Отпечатано с готовых диапозитивов в ГУП «Типография «Наука»
199034, Санкт-Петербург, 9 линия, 12.

Посвящается Бетани

Оглавление

Предисловие

Кроме написания книг, подобных той, что вы держите в руках, я пишу код. Я зарабатываю на жизнь разработкой веб-сайтов и приложений, надо полагать, как и многие мои читатели. Я использую CSS в повседневной работе, и мне прекрасно известно, чего стоят отчаянные попытки найти ошибки в коде, когда проект должен быть сдан на следующее утро.

Многие дизайнеры и разработчики, предпочитающие использовать каскадные таблицы стилей лишь для простого форматирования текста или вовсе обходиться без их использования, объясняют это нехваткой времени на изучение новой для них технологии с нуля. В конечном счете применение таблиц и изображений в формате GIF позволяет достичь поставленной цели, а значит, работа выполнена и оплачивается.

Мне повезло: я начала изучение технологии CSS с момента ее появления и настолько увлеклась, что мне захотелось постоянно совершенствовать свои навыки. Именно благодаря рано появившемуся интересу к CSS мои знания развивались вместе с самой технологией, и теперь за моими плечами девятилетний опыт проектирования веб-сайтов с помощью этой технологии.

В данной книге я намерена представить полезные приемы и методы использования CSS, упрощающие процесс и сокращающие время разработки веб-сайтов и приложений.

Эта книга не предлагает вам многостраничных теоретических рассуждений. В ней вы найдете решения, которые можно сразу же применить на практике; более того, они могут послужить основой для разработки собственных методов. По опыту могу сказать, что легче чему-то научиться делая, чем просто читая, поэтому вполне можете полагаться на данную книгу как на справочник для поиска подходящего решения при создании клиентского веб-сайта и тем самым выиграть время и сдать проект в срок. При этом хочется надеяться, что экспериментирование с предлагаемыми примерами поможет освоению новых технологий.

Книга составлена таким образом, чтобы вы смогли использовать представленные материалы для решения текущих задач. Ее необязательно читать от корки до корки – достаточно изучить интересующий вас в данный момент раздел. Каждый пример сопровождается комментарием, объясняющим механизм работы данного метода. Это позволит вам использовать эти примеры в дальнейшем, модифицируя их в соответствии с решаемыми в данный момент проблемами.

Надеюсь, что вам понравится эта книга. Я получила большое удовольствие от самого процесса ее написания и надеюсь, что она станет для вас полезным ежедневным справочником, а также помощником в освоении новых методов использования CSS.

Для кого предназначена эта книга

Данная книга будет интересна всем, кому приходится работать с CSS, а в особенности веб-дизайнерам и разработчикам, которые, конечно, видели много красивых веб-сайтов, основанных на применении CSS, но у которых нет времени на то, чтобы проштудировать огромное количество теоретических и накопленных практических материалов при создании собственного сайта. В этой книге вы найдете готовые решения различных проблем, которые могут быть использованы непосредственно в представленном здесь виде или взяты за основу при создании собственных решений.

В целом данную книгу нельзя назвать учебником; так как первая глава содержит обзор только основных особенностей CSS и сложность приводимых методов возрастает при движении от начальных глав к заключительным, то наличие у вас базовых знаний CSS существенно облегчило бы восприятие материала книги.

О чем написано в этой книге

Глава 1. CSS: основы основ

Данная глава по форме сильно отличается от прочих частей книги. По сути, она является кратким учебником по базовым возможностям CSS, который позволит также освежить имеющиеся знания. Если вы активно используете CSS в своих проектах, можете пропустить эту главу и возвращаться к ней по мере необходимости – если нужно более глубоко изучить основополагающие принципы технологии.

Глава 2. Форматирование текста и другие базовые возможности

Данная глава содержит информацию о различных методах форматирования и задания стиля текста в ваших документах: настройки размера шрифта, цвета и избавления от раздражающего пустого пространства, обрамляющего элементы страницы. Вы наверняка найдете для себя интересные приемы, даже если давно пользуетесь CSS для задания стилевого оформления текста.

Глава 3. CSS и графика

В данной главе я предлагаю различные приемы совместного использования CSS и графических элементов, позволяющие добиться впечатляющих результатов. Среди рассматриваемых тем – управление фоновым изображением различных элементов, позиционирование текста и изображений, а также многие другие.

Глава 4. Навигация

Без навигации не обойдется ни один проект, и в данной главе мы рассмотрим методы ее создания с помощью CSS. Представленные приемы позволят использовать CSS вместо графического меню, реализовать навигацию с помощью *вкладок*, создавать удобные и привлекательные панели навигации на основе CSS и фоновых изображений, а также использовать списки для структурирования системы навигации.

Глава 5. Табличные данные

Следует избегать использования таблиц для позиционирования элементов на странице: их необходимо применять по прямому назначению – для представления табличных данных, как в электронных таблицах. В данной главе мы познакомимся с приемами оформления табличных данных в наиболее привлекательной и удобной для использования форме.

Глава 6. Формы и пользовательские интерфейсы

Если вы работаете в качестве дизайнера или разработчика, то, несомненно, тратите уйму времени на создание форм для ввода данных. Из данной главы вы узнаете, как с помощью CSS сделать ваши формы удобными, привлекательными и доступными для различных категорий пользователей.

Глава 7. Кросс-броузерные решения

Как быть со старыми версиями броузеров, броузерами, неправильно интерпретирующими определенные свойства CSS, и альтернативными устройствами? Решением этих проблем мы займемся в главе 7. Кроме того, мы рассмотрим способы поиска «багов» и тестирования веб-страниц в максимально возможном количестве броузеров.

Глава 8. Доступность для людей с ограниченными возможностями и альтернативные устройства

Ваши страницы прекрасно выглядят для большинства посетителей сайта? Замечательно, однако как же быть людям, вынужденным использовать специальные возможности, такие как экранная лупа и экранный диктор? А пользователям, по тем или иным причинам предпочитающим при просмотре веб-сайтов пользоваться клавиатурой вместо мыши? В данной главе вы узнаете о том, как сделать ваши страницы одинаково доступными для *всех* пользователей, а не только для тех, кто не имеет никаких ограничений при доступе к сайту.

Глава 9. Позиционирование элементов с помощью CSS

В данной главе мы рассмотрим различные способы позиционирования элементов страницы и множество приемов, которые можно сочетать произвольным образом или взять за основу для своих собственных методов при создании оригинальной разметки страницы.

Веб-сайт книги

Сайт данной книги, доступный по адресу *http://www.sitepoint.com/books/cssant3/*, предлагает следующие возможности.

Архив кода

По ходу чтения книги вы заметите расположенные над листингами названия файлов. Они соответствуют именам файлов в архиве кода, который можно загрузить в виде файла в формате ZIP, щелкнув по ссылке Code Archive на сайте книги. В архиве содержатся все завершенные примеры, представленные в данной книге[1].

Опечатки и обновления

Ни одна книга не безупречна, и внимательный читатель наверняка заметит пару-тройку ошибок. Страница Corrections and Typos на сайте книги содержит актуальную информацию о найденных типографских ошибках и опечатках в коде, а также обновления, связанные с выходом новых версий броузеров и появлением новых стандартов.[2]

Форумы SitePoint

В сообществе дизайнеров SitePoint вы можете обсудить данную книгу.[3] В частности, на форуме, посвященном CSS, вы найдете огромное количество информации, выходящей далеко за пределы охвата данной книги. Здесь общаются многие опытные веб-разработчики и дизайнеры, делясь новыми приемами и трюками, отвечая на вопросы, – вы не пожалеете о затраченном времени.[4]

Электронные рассылки SitePoint

Помимо книг, подобных этой, SitePoint составляет бесплатные электронные рассылки, среди которых *The SitePoint Tribune*, *The SitePoint Tech Times* и *The SitePoint Design View*. В них содержится самая свежая информация, охватывающая различные аспекты разработки вебсайтов: последние новости, выпуск новых продуктов, актуальные тенденции и полезные приемы. Возможно, вы захотите читать только новости, касающиеся CSS, но если вас интересуют и другие технологии, то вы сможете найти для себя множество полезных тем. Оформить бесплатную подписку можно по адресу *http://www.sitepoint.com/newsletter*.

---

[1] Архив примеров кода можно скачать с сайта издательства по адресу *www.symbol.ru/library/css-101-tips*

[2] *http://www.sitepoint.com/books/cssant3/errata.php*

[3] *http://www.sitepoint.com/forums/*

[4] *http://www.sitepoint.com/launch/cssforum/*

Подкаст SitePoint

Подкасты SitePoint содержат актуальные новости и интервью; веб-разработчики и дизайнеры делятся своим мнением по различным вопросам. Они обсуждают последние тенденции в области разработки вебсайтов, приглашают интересных гостей и проводят интервью с самыми влиятельными людьми в отрасли. Предыдущие и текущие подскасты можно послушать по адресу *http://www.sitepoint.com/blogs/category/podcast/*; кроме того, вы можете оформить подписку через iTunes.

Обратная связь

Если вы не смогли найти ответ на свой вопрос на форумах или хотите связаться с нами по какой-то иной причине, это можно сделать по адресу *books@sitepoint.com*. Ваши письма будут рассмотрены службой поддержки, сотрудники которой готовы ответить на ваши вопросы. Кроме того, мы с благодарностью примем ваши предложения с уточнениями и сообщения о найденных ошибках.

Благодарности

Прежде всего, мне хотелось бы поблагодарить всех сотрудников SitePoint – благодаря им эта книга увидела свет. С вами было приятно работать, несмотря на различия часовых поясов – я уже засыпала, когда ваш рабочий день только начинался.

Спасибо тем, кто развивает новые идеи в мире CSS, многие из которых будут рассмотрены на страницах книги, и тем, кто делится ими с сообществом разработчиков.

Спасибо Дрю за одобрение и поддержку, за то, что обсуждал со мной методы использования CSS, когда я работала над примерами для книги, за то, что заставлял меня улыбнуться, когда я начинала раздражаться от усталости, за готовность примириться с тем, что мы практически совершенно выпали из светской жизни. И наконец, я хочу поблагодарить свою дочь Бетани за понимание, когда я проводила все время за компьютером, и за то, что каждый день напоминала мне о том, что в жизни действительно важно. Благодаря вам двоим многое становится возможным, спасибо.

Обозначения, принятые в книге

Любая разметка – HTML или CSS – выводится моноширинным шрифтом, как в следующем примере:

```
<h1>A perfect summer's day</h1>
<p>It was a lovely day for a walk in the park. The birds
were singing and the kids were all back at school.</p>
```

Если данный код представлен в размещенном на сайте архиве, в верхней части листинга будет отображено имя соответствующего файла, например:

example.css

```
.footer {
  background-color: #CCC;
  border-top: 1px solid #333;
}
```

Если выводится только часть кода, к названию будет добавлено слово *фрагмент*:

example.css (фрагмент)

```
border-top: 1px solid #333;
```

Если к рабочему примеру добавляется новый код, он выделяется жирным шрифтом:

```
function animate() {
  new_variable = "Hello";
}
```

Если для контекста необходим существующий код, вместо повторения будет использовано вертикальное многоточие:

```
function animate() {
  ⋮
  return new_variable;
}
```

Некоторые части кода не умещаются в одну строчку из-за ограниченной ширины печатной страницы. Для обозначения разрыва строки используется знак ➥. Он выполняет исключительно функцию форматирования.

```
URL.open("http://www.sitepoint.com/blogs/2007/05/28/user-style-she
➥ets-come-of-age/");
```

Примечания, заметки, советы и предупреждения оформлены следующим образом:

Совет

Так будут представлены полезные советы и подсказки.

Примечание

Заметки содержат дополнительную информацию, связанную с рассматриваемой в данный момент темой.

Внимание

На эти важные аспекты следует обратить внимание.

Предупреждение

Предупреждения указывают на подводные камни, с которыми вы можете столкнуться в ходе работы.

С сайта издательства (*www.symbol.ru/library/css-101-tips)* можно скачать цветные версии тех иллюстраций, где наличие цвета может помочь лучше разобраться в предлагаемом материале.

1

CSS: основы основ

Каскадные таблицы стилей... Звучит несколько устрашающе. Такое название способно вызвать в воображении нагромождение таинственных символов кода, едва ли доступных для понимания новичку. Однако на самом деле CSS – один из самых простых и удобных в использовании инструментов, имеющихся в распоряжении веб-разработчиков. В первой главе, отличающейся по формату от последующих частей, мы познакомимся с основами CSS и использованием этой технологии для облегчения задачи управления единообразно оформленным сайтом. Если у вас уже есть некоторый опыт работы с CSS для форматирования текста на веб-страницах, можете пропустить данную главу и приступить к чтению второй главы, где мы начнем рассмотрение практических решений.

Определение стиля с помощью CSS

CSS используется главным образом для создания **описаний** (или **деклараций**) **стилей** (к примеру, шрифта, размера или цвета) и их применения к выбранным частям HTML-кода посредством **селекторов** – ссылок на элемент или группу элементов, по отношению к которым нужно применить указанный стиль.

Решение

Рассмотрим данный основополагающий механизм на примере следующего HTML-документа:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
  <head>
    <title>A Simple Page</title>
```

```
    <meta http-equiv="content-type"
      content="text/html; charset=utf-8" />
  </head>
 <body>
    <h1>First Title</h1>
    <p>A paragraph of interesting content.</p>

    <h2>Second Title</h2>
    <p>A paragraph of interesting content.</p>

    <h2>Third title</h2>
    <p>A paragraph of interesting content.</p>
  </body>
</html>
```

Он содержит три заголовка (выделенных жирным шрифтом), созданных с помощью тегов <h1> и <h2>. Если разработчик не определил собственные стили, заголовки будут оформлены в соответствии со стандартной таблицей стилей броузера, т. е. заголовок h1 будет отображаться крупным шрифтом, h2 – мельче, чем h1, но крупнее обычного текста абзаца. Документ, оформленный стилями по умолчанию, будет вполне *читаемым*, если он достаточно прост. Для изменения внешнего вида этих элементов будет достаточно добавить несколько строчек CSS-кода:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
  <head>
    <title>A Simple Page</title>
    <meta http-equiv="content-type"
      content="text/html; charset=utf-8" />
    <style type="text/css">
      h1, h2 {
        font-family: sans-serif;
        color: #3366CC;
      }
    </style>
  </head>
  <body>
    <h1>First Title</h1>
    <p>A paragraph of interesting content.</p>

    <h2>Second Title</h2>
    <p>A paragraph of interesting content.</p>

    <h2>Third title</h2>
    <p>A paragraph of interesting content.</p>
  </body>
</html>
```

Все преобразования разместились между тегами <style>, расположенными в заголовке страницы (элемент head). Мы определили, что заго-

ловки h1 и h2 должны отображаться шрифтом sans-serif светло-голубого цвета. Особенности синтаксиса мы рассмотрим чуть позднее. Нам ничего не потребуется менять в разметке страницы – изменения в описании стиля, размещенного в верхней части страницы, распространяются на все три имеющихся на странице заголовка и будут также применены ко всем заголовкам, добавленным позднее с помощью тегов <h1> или <h2>.

Примечание

HTML или XHTML? На протяжении всей книги при упоминании веб-страниц, разметки и примеров кода я буду использовать термин HTML. Это можно интерпретировать как HTML и/или XHTML, если специально не оговорено иначе.

Теперь, когда вы имеете представление о назначении стилей CSS, настало время рассмотреть способы их использования в HTML-документах.

Внутритекстовые стили

Проще всего изменять оформление элементов страницы с помощью **внутритекстовых стилей**. Этот способ заключается в присваивании HTML-элементу определенного стиля посредством атрибута style, как в следующем примере:

```
<p style="font-family: sans-serif; color: #3366CC;">
  Amazingly few discotheques provide jukeboxes.
</p>
```

Селекторы при этом не используются – описание стиля применяется непосредственно к элементу, в тег которого оно добавлено (в вышеприведенном примере – в тег <p>).

Основным недостатком данного способа является отсутствие возможности повторного использования внутритекстовых стилей. Например, чтобы применить тот же самый стиль к другому элементу p, нужно снова указать его в качестве значения атрибута style. Если в дальнейшем появится необходимость в изменении определения стиля, придется искать и редактировать все теги, в которых оно было использовано. Кроме того, размещение внутритекстовых стилей в разметке страницы затрудняет чтение и дальнейшее управление кодом.

Встраиваемые стили

CSS-стили можно применять к веб-страницам с помощью элемента style, как в приведенном выше примере. Таким образом, у вас появляется возможность создать произвольное количество деклараций стилей, размещаемых между открывающим и закрывающим тегами <style>:

```
<style type="text/css">
  ⋮ CSS Styles…
</style>
```

Теги <style> расположены внутри элемента head веб-страницы. Атрибут type используется для указания языка, на котором написана таблица стилей. Единственным широко распространенным языком для создания стилей является CSS, чему соответствует значение text/css.

Этот простой и понятный метод размещения стилей между тегами <style> обладает одним недостатком: чтобы использовать определенный набор стилей на всем сайте, придется скопировать его и разместить в заголовке каждой страницы сайта.

Гораздо разумнее вынести все определения стилей в отдельный текстовый файл, а в самом документе создать ссылку на него. Такой отдельный файл называют внешней таблицей стилей.

Внешние таблицы стилей

Внешняя таблица стилей – это отдельный файл (как правило, с расширением **.css**), содержащий все описания CSS-стилей для данного сайта. На один и тот же файл могут ссылаться множество страниц, и любые изменения описаний стиля будут распространяться на каждую из них. Такой метод позволяет создавать набор деклараций для целого сайта, о чем уже говорилось выше.

Для создания в документе ссылки на внешнюю таблицу стилей (к примеру, **styles.css**) достаточно разместить в его заголовке элемент link:

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

Помните наш первый пример, в котором для трех заголовков был определен одинаковый стиль? Реализуем то же самое с помощью ссылки на внешнюю таблицу стилей с именем **styles.css**:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
<head>
<title>A Simple Page</title>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<link rel="stylesheet" type="text/css" href="styles.css" />
</head>
<body>

<h1>First Title</h1>
<p>…</p>

<h2>Second Title</h2>
<p>…</p>

<h2>Third Title</h2>
<p>…</p>
</body>
</html>
```

Атрибуту `rel` необходимо присвоить значение `stylesheet`, а `type` — `text/css`. Атрибут `href` указывает на размещение и имя файла с таблицами стилей.

Файл **styles.css** содержит следующие описания стилей:

```
h1, h2 {
  font-family: sans-serif;
  color: #3366CC;
}
```

Подобно изображениям, файл **styles.css** можно использовать многократно. Поэтому вы избавляетесь от необходимости повторно вводить описания стилей, а также можете быть уверены, что все заголовки на сайте будут оформлены одинаково.

Синтаксис CSS

Таблица стилей состоит из набора определений стилей. Двумя основными компонентами определения (или правила) стиля являются:

- Список из одного или более **селекторов**, разделенных запятыми. Они указывают на элемент, к которому будет применен данный стиль.
- **Блок описаний**, заключенный в фигурные скобки, в котором содержится информация о том, как данный стиль повлияет на отображение элемента.

Блок описаний состоит из одного или нескольких **описаний стиля**, каждый из которых устанавливает значение определенного **свойства**. Если описаний несколько, они отделяются друг от друга точкой с запятой (;). Описание свойства состоит из его имени и значения, разделенных двоеточием (:). Схематично все описанные элементы изображены на рис. 1.1.

Рис. 1.1. *Составные части правила CSS: список селекторов и блок описаний*

Декларация свойства определяет значения параметров шрифта, цветов и других настроек, которые будут присвоены элементу после применения стиля. Предлагаемые в данной книге решения в большинстве своем основаны на сочетании различных свойств и их значений.

На рис. 1.1 также видно, что правило стиля можно записать в одну строчку. Одни разработчики предпочитают переносить каждое новое

свойство на следующую строку – такой код легче читать; другие пишут правила в одну строчку, чтобы сэкономить место. Ниже приведены два варианта записи одного и того же стиля:

```
h1, h2 {
  font-family: sans-serif;
  color: #3366CC;
}

h1, h2 { font-family: sans-serif; color: #3366CC; }
```

Форматирование не влияет на интерпретацию стилей, поэтому вы можете выбрать более близкий вам способ.

Что такое селекторы и как их правильно использовать

В следующем примере используются два селектора – `h1` и `h2`. Это означает, что указанные стили будут применяться ко всем элементам `h1` и `h2`:

```
h1, h2 {
  font-family: sans-serif;
  color: #3366CC;
}
```

Решение

В этом разделе представлено описание широко используемых в настоящее время селекторов CSS2.1 с примерами их практического использования.

Селекторы типа

Самыми простыми селекторами являются **селекторы типа**, с которыми мы уже встречались в приведенных выше примерах. Адресация по имени HTML-элемента обеспечивает применение правила стиля ко всем элементам с данным именем, присутствующим в документе. Селекторы типа часто используются для определения основных стилей для всех страниц сайта. К примеру, следующее правило стиля задает шрифт абзаца, используемый на сайте по умолчанию:

```
p {
  font-family: Tahoma, Verdana, Arial, Helvetica, sans-serif;
  font-size: 1em;
  color: #000000;
}
```

Таким образом устанавливается шрифт, размер и цвет для всех абзацев (элементов `p`) в документе.

Селекторы класса

Задавать стили для элемента, конечно, удобно, однако что делать, если возникает необходимость определения различных стилей для нескольких элементов одного типа, размещенных в различных частях страницы? На помощь придут **классы**.

Рассмотрим следующий пример, в котором задается синий цвет шрифта для всех абзацев:

```
p {
  color: #0000FF;
}
```

Отлично! А теперь представьте, что на вашей странице будет расположена боковая панель с голубым фоном. При этом голубой текст на панели сольется с ним, и разобрать что-либо станет просто невозможно. Эту проблему можно решить, определив класс для текста, выводимого на боковой панели, а затем задав для этого класса необходимый стиль с помощью CSS.

Для создания абзаца класса `sidebar` необходимо вначале добавить атрибут `class` с соответствующим значением в открывающий тег:

```
<p class="sidebar">This text will be white, as specified by the CSS style
   definition below.</p>
```

Затем можно приступить к описанию стиля для данного класса:

```
p {
  color: #0000FF;
}
  .sidebar {
  color: #FFFFFF;
}
```

Во втором правиле используется селектор класса, определяющий стиль для любого элемента со значением `sidebar` атрибута `class`. Перед именем ставится точка, указывающая, что это класс, а не HTML-элемент.

А если на панели также присутствуют ссылки? По умолчанию все ссылки в вашем документе будут оформлены одинаковым образом. Однако, если добавить в тег ссылки атрибут `class="sidebar"`, они также станут белыми:

```
<p class="sidebar">This text will be white, <a class="sidebar"
href="link.html">and so will this link</a>.</p>
```

Прекрасно. Но, возможно, вам хотелось бы, чтобы ссылки отличались от основного текста? К примеру, им можно было бы придать жирное начертание. Добавление описания стиля для жирного начертания текста к набору стилей для класса `sidebar` приведет к тому, что весь текст

на панели будет выведен жирным шрифтом, что противоречит поставленной нами цели. Нужен CSS-селектор для выбора ссылок, размещенных внутри элемента класса `sidebar`. Далее путем комбинации селектора типа и селектора класса можно добиться желаемого эффекта:

```
p {
  color: #0000FF;
}
.sidebar {
  color: #FFFFFF;
}
a.sidebar:link, a.sidebar:visited {
  font-weight: bold;
}
```

Обратите внимание, что в данном примере используются псевдоклассы `:link` и `:visited`. Мы поговорим о псевдоклассах чуть позже в этой главе.

Если вы добавите приведенные выше стили в таблицу стилей вашего сайта и обновите страницу в броузере, то увидите, что ссылки будут отображены жирным шрифтом белого цвета, поскольку к ним были применены оба определенных для класса `sidebar` правила стилей. Однако если бы мы добавили в третье правило стиль изменения цвета шрифта, цвет ссылок также был бы изменен, поскольку третий селектор более конкретен, а правила CSS применяются в порядке увеличения конкретности селекторов.

Кстати, процесс применения различных правил стилей к одному и тому же элементу называется **каскадированием**; отсюда и термин «каскадные таблицы стилей». Ближе к концу главы мы узнаем о степени конкретности различных селекторов и правилах каскадирования.

Селекторы ID

В отличие от селекторов класса, определяющих стиль для группы элементов, **селекторы ID** используются для присваивания стиля одному конкретному элементу. Чтобы воспользоваться таким селектором, необходимо прежде всего добавить атрибут `id` элементу, стиль которого должен быть изменен. Важно помнить, что идентификатор (ID) в HTML-документах уникален:

```
<p id="tagline">This paragraph is uniquely identified by the ID “tagline”.</p>
```

В качестве ссылки на данный элемент (посредством селектора ID) необходимо указать идентификатор с предшествующим ему знаком «решетка» (#). К примеру, следующее правило стиля задает белый цвет шрифта для приведенного выше абзаца:

```
#tagline {
  color: #FFFFFF;
}
```

Селекторы ID можно использовать совместно с селекторами других типов. В следующем примере правило стиля применяется ко всем элементам класса `new`, расположенным внутри абзаца со значением `tagline` атрибута `id`:

```
#tagline .new {
  font-weight: bold;
  color: #FFFFFF;
}
```

Использованный в данном примере селектор называется контекстным; селекторам этого типа посвящен следующий раздел.

Контекстные селекторы

Если на панели расположено несколько абзацев, *можно* добавить атрибут класса каждому открывающему тегу <p>. Однако гораздо лучше указать атрибут `class` со значением `sidebar` для элемента, служащего контейнером, а затем изменить цвет каждого расположенного внутри контейнера элемента `p` на белый. Для этого достаточно написать всего одно правило стиля с использованием **контекстного селектора (селектора-потомка).**

Ниже приведено правило с новым селектором:

```
p {
  color: #0000FF;
}
.sidebar p {
  color: #FFFFFF;
}
```

А так выглядит обновленный HTML-код:

```
<div class="sidebar">
  <p>This paragraph will be displayed in white.</p>
  <p>So will this one.</p>
</div>
```

Как видите, контекстный селектор состоит из списка селекторов, разделенных пробелами, соответствующих элементам страницы от *внешних к внутренним*. В данном случае на странице имеется элемент `div` класса `sidebar`, и контекстный селектор `.sidebar p` ссылается на все элементы `p`, расположенные внутри этого `div`.

Дочерние селекторы

Контекстный селектор ссылается на всех потомков родительского элемента, в том числе и *непрямых* потомков.

Рассмотрим следующий код:

```
<div class="sidebar">
  <p>This paragraph will be displayed in white.</p>
```

```
    <p>So will this one.</p>
    <div class="tagline">
      <p>If we use a descendant selector, this will be white too. But if we
         use a child selector, it will be blue.</p>
    </div>
  </div>
```

В данном примере используется контекстный селектор из предыдущего раздела – `.sidebar p`. Он ссылается на все элементы `p`, вложенные в `div` класса `.sidebar p`, *включая* и абзацы, расположенные внутри `div` класса `tagline`. Чтобы указанное оформление распространялось исключительно на *прямых* потомков элемента `div` класса `sidebar`, следует использовать **дочерний селектор**. Для указания прямого потомка в селекторе используется знак >.

В приведенном ниже коде используется новый селектор для задания белого цвета текста абзацев, расположенных непосредственно внутри элемента `div` класса `sidebar` (но не внутри элемента `div` класса `tagline`):

```
p {
  color: #0000FF;
}
.sidebar>p {
  color: #FFFFFF;
}
```

Предупреждение

Internet Explorer 6 не поддерживает дочерние селекторы. Поэтому в том случае, если задаваемый с их помощью стиль важен для форматирования страницы, необходимо продумать альтернативные способы обращения к требуемому элементу.

Смежные селекторы

Смежные селекторы ссылаются на элемент, расположенный непосредственно после другого обозначенного элемента. Таким образом, если на странице имеется следующий HTML-код:

```
<h2>This is a title</h2>
<p>This paragraph will be displayed in white.</p>
<p>Ths paragraph will be displayed in black.</p>
```

А затем мы используем следующий селектор:

```
p {
  color: #000000;
}
h2+p {
  color: #FFFFFF;
}
```

При этом только первый абзац отображается в белом цвете. Второй элемент p не является смежным с h2 и потому его текст выводится черным шрифтом, что указано в первом правиле для элементов p.

Предупреждение

Internet Explorer 6 не поддерживает смежные селекторы. Потому, если задаваемый с их помощью стиль важен для форматирования страницы, следует найти альтернативный способ обращения к требуемому элементу.

Селекторы псевдоклассов для ссылок

HTML предлагает гораздо более широкие возможности для форматирования ссылок (создаваемых с помощью элемента a или **якоря**) по сравнению с остальными элементами. С помощью CSS можно задавать стиль отображения ссылок в зависимости от их состояния – посещенных или непосещенных – при наведении на них указателя мыши или при щелчке по ней. Рассмотрим следующий пример:

```
a:link { color: #0000FF; }
a:visited { color: #FF00FF; }
a:hover { color: #00CCFF; }
a:active { color: #FF0000; }
```

Приведенный выше код содержит четыре определения стилей CSS. При этом с каждым из селекторов используется так называемый **псевдокласс** элемента a. Псевдокласс по сути является ярлыком из набора, который может быть добавлен к селектору для указания на состояние соответствующего элемента. Между селектором и псевдоклассом ставится двоеточие (:). Ниже представлены широко используемые псевдоклассы для ссылок:

- :link распространяет свое действие исключительно на *непосещенные* ссылки и задает для них синий цвет шрифта.
- :visited действует на *посещенные* ссылки и выводит их в фиолетовом цвете.
- :hover окрашивает ссылки в светло-голубой цвет при наведении на них указателя мыши вне зависимости от того, были они посещены или нет.
- :active изменяет цвет ссылки на красный после щелчка по ней.

Важное значение имеет порядок следования селекторов псевдоклассов в таблице стилей. В данном случае :active стоит на последнем месте; это означает, что его действие имеет более высокий приоритет над перечисленными ранее тремя описаниями, и потому его применение не зависит от того, были ли ссылки посещены или нет, навел ли пользователь на них указатель мыши или нет.

Состояния :hover и :active формально называют **динамическими селекторами псевдоклассов**, поскольку они появляются только при взаимодействии пользователя с соответствующими элементами путем наведения указателя мыши и щелчка по ссылке соответственно.

Примечание

Применение :hover с другими элементами. Динамический селектор псевдокласса :hover, помимо ссылок, можно использовать с другими элементами, что позволяет создавать эффекты вроде подсветки ряда таблицы при наведении на него указателя мыши. Однако Internet Explorer 6 и более ранние версии поддерживают использование данного псевдокласса исключительно с элементами ссылок.

Псевдокласс :first-child

Еще одним примером псевдокласса является :first-child. В то время как смежный селектор ссылается на элемент, *следующий за* указанным элементом в исходном коде документа, селектор псевдокласса :first-child ссылается на первый по порядку дочерний элемент обозначенного родителя. Таким образом, единственным отличием в использовании данных элементов является то, что в последнем случае поставленному условию удовлетворяет только первый дочерний элемент:

```
<div class="article">
  <p>
    This is an intro paragraph to be
    displayed with a larger font size.
  </p>
  <p>
    Here is a second paragraph of text
    displayed at normal text size.
  </p>
</div>
```

CSS-код будет выглядеть следующим образом:

```
p {
  font-size: 100%
}
.article p:first-child {
  font-size: 160%;
}
```

Поскольку первый абзац является первым дочерним элементом родительского блока div класса article, его текст будет отображаться более крупным шрифтом. Размер шрифта второго абзаца соответствует определенному для всех элементов p в документе.

Предупреждение

Internet Explorer 6 не поддерживает псевдокласс :first-child. Как оказалось, броузер IE6 не поддерживает и селектор псевдокласса :first-child, поэтому, если задаваемый с их помощью стиль важен для форматирования страницы, следует найти альтернативный способ адресации.

Каким образом броузер определяет, какие стили нужно использовать

Как броузер может «понять» намерения разработчика? Если к одному и тому же элементу могут быть применены несколько стилей, то определение, какие из них будут использованы, происходит по принципу **каскадирования**.

Принцип каскадирования чрезвычайно важен для понимания CSS, поскольку большинство ошибок разработки, связанных с таблицами стилей, происходят из-за того, что применение стиля не соответствует намерениям разработчика. В данной главе мы уже рассмотрели пример, в котором было определено общее правило стиля для элементов типа абзац, а также более конкретное правило для одного или нескольких определенных абзацев. Оба правила предназначены для изменения оформления абзацев, *однако более конкретные правила имеют приоритет над более общими*.

Решение

На выбор применяемого броузером стиля влияют четыре фактора: вес, источник, конкретность и порядок следования.

Вес определенного описания стиля можно определить с помощью ключевого слова `!important`, добавляемого после значения свойства. При этом свойство приобретает приоритетное значение над аналогичными свойствами, указанными в других правилах стиля, за исключением редких случаев. Применение ключевого свойства `!important` сильно усложняет процесс поддержки кода; кроме того, необходимость в его использовании возникает не слишком часто. Ввиду приведенных причин рекомендуется по возможности избегать его, как в примерах данной книги. Дополнительную информацию о данном ключевом слове можно найти в документации SitePoint CSS Reference.[1]

Существуют три возможных **источника** стилей – они могут быть определены броузером, разработчиком или пользователем. В данной книге нас интересуют **таблицы стилей, написанные разработчиком** веб-страницы, т. е. вами. Мы уже говорили, что у броузера есть своя внутренняя таблица стилей, определяющая оформление всех элементов по умолча-

[1] *http://reference.sitepoint.com/css/importantdeclarations/*

нию, однако стили, написанные разработчиком, всегда имеют более высокий приоритет и переопределяют настройки по умолчанию. Кроме того, возможно применение **пользовательской таблицы стилей** – набора стилей, созданного пользователями, однако и они, за исключением редких случаев, переопределяются таблицей стилей разработчика. Источникам стилей посвящен целый раздел документации SitePoint CSS Reference.[1]

Помимо этого, влияние каскадирования на поведение стилей CSS зависит от степени конкретности свойств и порядка их следования.

Правило стиля с более конкретным селектором переопределяет стили, задаваемые правилом с более общим селектором. Рассмотрим этот механизм на практическом примере со следующим несложным HTML-кодом:

```
<div id="content">
  <p class="message">
    This is an important message.
  </p>
</div>
```

А теперь обратите внимание на правила стилей, применяемых к приведенной выше разметке:

```
p { color: #000000; }

.message { color: #CCCCCC; }

p.message { color: #0000FF; }

#content p.message { color: #FF0000; }
```

В данном примере действие всех четырех селекторов, задающих цвет шрифта, направлено на элемент `p`. Какой же цвет будет использован? Совершенно верно, красный (`#FF0000`). Селекторы `p` (любой элемент `p`) и `.message` (любой элемент класса `message`) имеют одинаковую степень значимости; приоритет селектора `p.message` (любой элемент `p` класса `message`) выше, но самым высоким приоритетом обладает селектор `#content p.message` (любой элемент `p` класса `message`, являющийся дочерним для элемента со значением `content` атрибута `id`).

Однако длину селекторов не стоит рассматривать как показатель степени конкретности. К примеру, селектор ID всегда будет иметь более высокий приоритет, чем селектор типа или класса. Чем сложнее используемые вами селекторы, тем труднее разобраться в степени их важности, при этом приведенные в данной книге примеры достаточно про-

[1] *http://reference.sitepoint.com/css/cascade/*

сты. Точную формулу измерения степени конкретности вы можете найти в документации SitePoint CSS Reference.[1]

Если, несмотря на вышеперечисленные факторы, остается несколько стилей, которые могут быть применены по отношению к одному и тому же элементу, в расчет принимается **порядок следования** правил – в этом случае будет использовано последнее из них. В приведенном выше примере степень значимости селекторов `p` и `.message` одинакова, поэтому при отсутствии иных регулирующих принципов используется правило стиля, записанное последним, – в данном случае с селектором `.message`. Так же дело обстоит и при объявлении в таблице стилей нескольких правил с одним и тем же селектором – например, `.message`. В этом случае будет применено правило, определенное вторым. В последующих главах мы увидим, что такой механизм работы можно эффективно использовать для достижения своих целей.

Заключение

В данной главе вы познакомились с основными принципами и способами использования CSS. Мы даже затронули такую сложную тему, как правила каскадирования. Даже если вы никогда раньше не работали с CSS, но разобрались с понятиями, представленными в данной главе, этого будет достаточно для понимания приводимых далее примеров.

Так как примеры, представленные в начале книги, проще, чем примеры в конце книги, то, если вы новичок в области применения CSS, рекомендую вам начать чтение с первых глав. Таким образом вы сможете пополнить знания, приобретенные в первой главе, и приступить к практике, погрузившись в увлекательный (надеюсь) мир каскадных таблиц стилей.

[1] *http://reference.sitepoint.com/css/specificity*

2

Оформление текста и другие базовые возможности

В данной главе мы рассмотрим основные способы использования CSS для задания стилевого оформления текста, а также прочие основные возможности; вы найдете ответы на наиболее часто задаваемые вопросы об их применении. Если вы новичок в области CSS, то с помощью приведенных примеров вы узнаете о существовании многих свойств и о методах их использования, что создаст необходимую базу для написания собственного кода. Если вы уже хорошо знакомы с технологией CSS, то эта глава сможет подсказать вам необходимые приемы, если в процессе разработки вы вдруг обнаружите, что забыли, как достичь того или иного эффекта.

Приведенные примеры поддерживаются подавляющим большинством броузеров, однако следует помнить о важности тестирования написанного кода в различных версиях броузеров. Хотя некоторые несоответствия или отсутствие поддержки предлагаемых методов могут встречаться в более ранних версиях броузеров, в целом представленные решения не должны вызывать каких-то серьезных проблем.

Задание определенного шрифта для текста

Решение

Задать определенный шрифт для оформления текста можно с помощью свойства `font-family`, как показано в следующем примере:

```
p {
  font-family: Verdana;
}
```