

Самоучитель CSS



Дизайн інтер'єров
3D-визуалізація



arch.design@jungle.in.ua

Авторские права

Самоучитель CSS (далее произведение) распространяется на условиях следующей лицензии Creative Commons: «Указание авторства — Некоммерческое использование — Без производных» (Attribution-NonCommercial-NoDerivs 2.5 Generic (CC BY-NC-ND 2.5)).

Вы можете свободно



— копировать, распространять и передавать данное произведение.

На следующих условиях



— обязательно указывать автора произведения (Влад Мержевич).



— нельзя использовать произведение для заработка.



— запрещается изменять произведение или создавать другие с его помощью.

Обозначения

Для удобства и наглядного представления материала используется разделение по цветам различных элементов.

`` — тег.

`align` — атрибут тега, стилевое свойство, ключевое слово или выделение.

`right` — значение.

`layer` — имя класса или идентификатора.

`File > Open` — пункт меню указанной программы.

`Tab` — клавиша на клавиатуре.



Этот символ призван привлечь внимание к некоторому замечанию по тексту.

Браузеры

В таблице с браузерами встречаются такие изображения:



— свойство полностью поддерживается браузером со всеми допустимыми значениями;



— свойство браузером не воспринимается и игнорируется;



— свойство поддерживается лишь частично, например, не все допустимые значения действуют или свойство применяется не ко всем элементам, которые указаны в спецификации;



— свойство понимается браузером, но при его работе возможно появление различных ошибок. В примечании обычно указывается, какого рода ошибки обнаруживаются в браузере.

Примеры

Как правило, примеры содержат список браузеров, в которых проверялся код. Для сокращения места браузеры обозначаются двумя буквами с номером версии.

- IE — Internet Explorer.
- Cr — Google Chrome.
- Op — Opera.
- Sa — Apple Safari.
- Fx — Mozilla Firefox.

Работоспособность примера в браузере помечается цветом, совпадающим с цветами в таблице с браузерами.

HTML задает основную структуру веб-страницы, а также указывает, какие элементы на ней присутствуют. Само оформление веб-страницы, положение и вид элементов возложен на стили или CSS (Cascading Style Sheets, каскадные таблицы стилей). Когда говорят о верстке веб-страниц, подразумевается синергия HTML и CSS. Что такое синергия? Сам HTML не представляет отдельного интереса, в силу своей простоты и ограниченности. Также и CSS не играет отдельной роли, поскольку привязывается к определенным элементам кода и задает их оформление. Поэтому работая вместе в одной связке, они превращают скромную страницу в тот документ, который придумал и нарисовал дизайнер. Такое взаимное усиление свойств, суммирующий эффект и является синергией.

Любая веб-страница это, по сути, комбинация HTML-кода и CSS-кода. Без основных знаний этих технологий не получится грамотно сверстать ни один документ. Поэтому первая глава посвящена основам CSS и его применению на практике. Если вы считаете, что это уже вам известно, можете пропустить эту главу и перейти к следующей.

Что такое стили?

Стили представляют собой набор параметров, управляющих видом и положением элементов веб-страницы. Чтобы стало понятно, о чем идет речь, посмотрим на рис. 1.1.

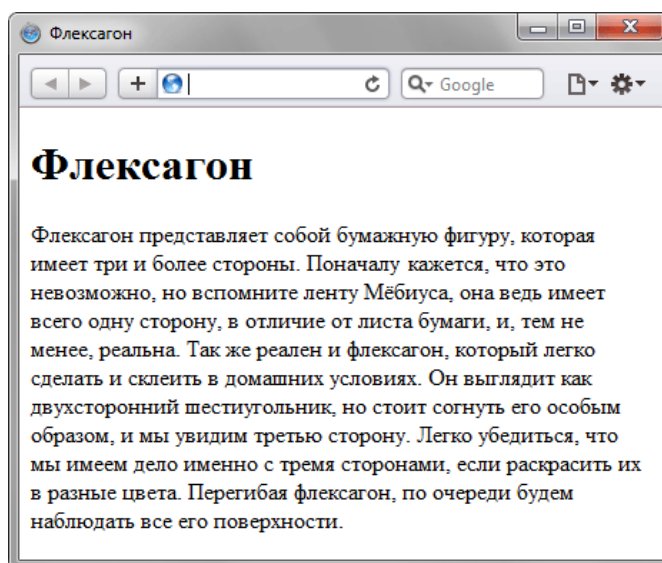


Рис. 1.1. Веб-страница, созданная только на HTML

Это обычная веб-страница, оформленная без всяких изысков. Тот же самый документ, но уже с добавлением стилей приобретает совершенно иной вид (рис. 1.2).

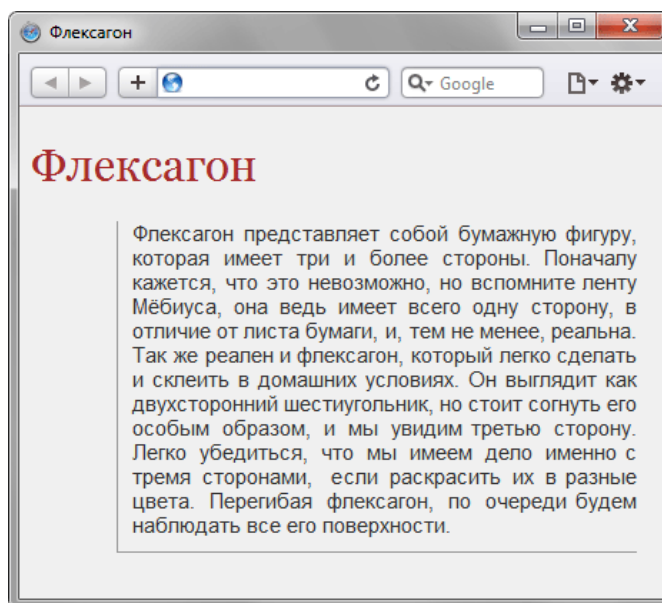


Рис. 1.2. Веб-страница, созданная на HTML и CSS

Перемена разительна, поэтому заглянем в код, чтобы понять, в чем же разница (пример 1.1).

```

<!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">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Флексагон</title>
  <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
  <h1>Флексагон</h1>
  <p>Флексагон представляет собой бумажную фигуру, которая имеет три и более стороны.
  Поначалу кажется, что это невозможно, но вспомните ленту Мёбиуса, она ведь имеет всего
  одну сторону, в отличие от листа бумаги, и, тем не менее, реальна. Так же реален и
  флексагон, который легко сделать и склеить в домашних условиях. Он выглядит как
  двухсторонний шестиугольник, но стоит согнуть его особым образом, и мы увидим
  третью сторону. Легко убедиться, что мы имеем дело именно с тремя сторонами, если
  раскрасить их в разные цвета. Перегибая флексагон, по очереди будем наблюдать
  все его поверхности.</p>
</body>
</html>

```

Сам код HTML никаких изменений не претерпел и единственное добавление — это строка `<link rel="stylesheet" href="style.css" type="text/css" />`. Она ссылается на внешний файл с описанием стилей под именем `style.css`. Содержимое этого файла показано в примере 1.2.

Пример 1.2. Содержимое стилевого файла `style.css`

```

body {
  font-family: Arial, Verdana, sans-serif; /* Семейство шрифтов */
  font-size: 11pt; /* Размер основного шрифта в пунктах */
  background-color: #f0f0f0; /* Цвет фона веб-страницы */
  color: #333; /* Цвет основного текста */
}
h1 {
  color: #a52a2a; /* Цвет заголовка */
  font-size: 24pt; /* Размер шрифта в пунктах */
  font-family: Georgia, Times, serif; /* Семейство шрифтов */
  font-weight: normal; /* Нормальное начертание текста */
}
p {
  text-align: justify; /* Выравнивание по ширине */
  margin-left: 60px; /* Отступ слева в пикселах */
  margin-right: 10px; /* Отступ справа в пикселах */
  border-left: 1px solid #999; /* Параметры линии слева */
  border-bottom: 1px solid #999; /* Параметры линии снизу */
  padding-left: 10px; /* Отступ от линии слева до текста */
  padding-bottom: 10px; /* Отступ от линии снизу до текста */
}

```

В файле `style.css` как раз и описаны все параметры оформления таких тегов как `<body>`, `<h1>` и `<p>`. Заметьте, что сами теги в коде HTML пишутся как обычно.

Поскольку на файл со стилем можно ссылаться из любого веб-документа, это приводит в итоге к сокращению объема повторяющихся данных. А благодаря разделению кода и оформления повышается гибкость управления видом документа и скорость работы над сайтом.

Типы стилей

Различают несколько типов стилей, которые могут совместно применяться к одному документу. Это стиль браузера, стиль автора и стиль пользователя.

Стиль браузера

Оформление, которое по умолчанию применяется к элементам веб-страницы браузером. Это оформление можно увидеть в случае «голого» HTML, когда к документу не добавляется никаких стилей. Например, заголовок страницы, формируемый тегом `<h1>`, в большинстве браузеров выводится шрифтом с засечками размером 24 пункта.

Стиль автора

Стиль, который добавляет к документу его разработчик. В примере 1.1 показан один из возможных способов подключения авторского стиля.

Стиль пользователя

Это стиль, который может включить пользователь сайта через настройки браузера. Такой стиль имеет более высокий приоритет и переопределяет исходное оформление документа. В браузере Internet Explorer подключение стиля пользователя делается через меню `Сервис > Свойство обозревателя > Кнопка «Оформление»`, как показано на рис. 1.3.

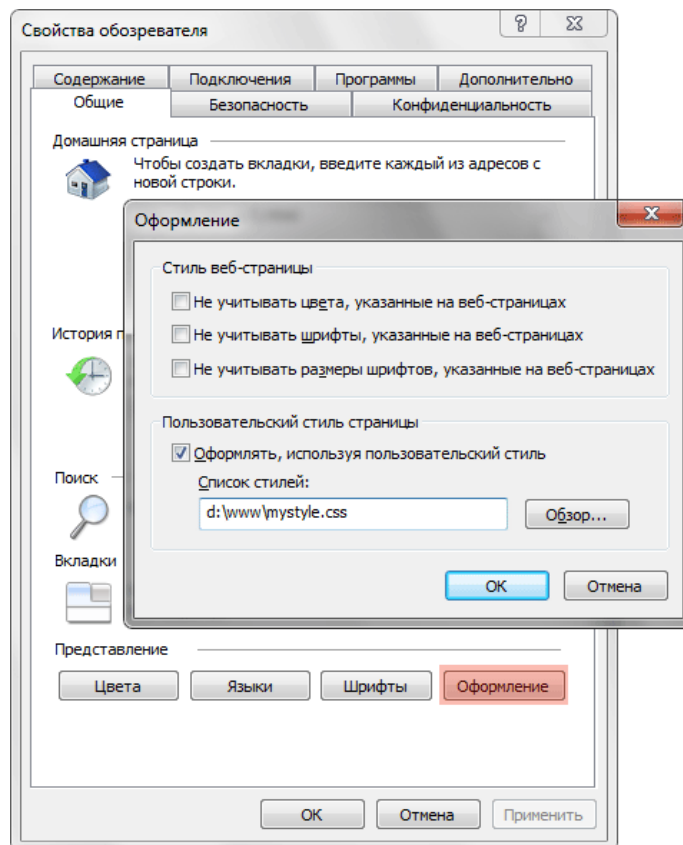


Рис. 1.3. Подключение стиля пользователя в браузере Internet Explorer

В браузере Opera аналогичное действие происходит через команду Инструменты > Общие настройки > Вкладка «Расширенные» > Содержимое > Кнопка «Параметры стиля» (рис. 1.4).

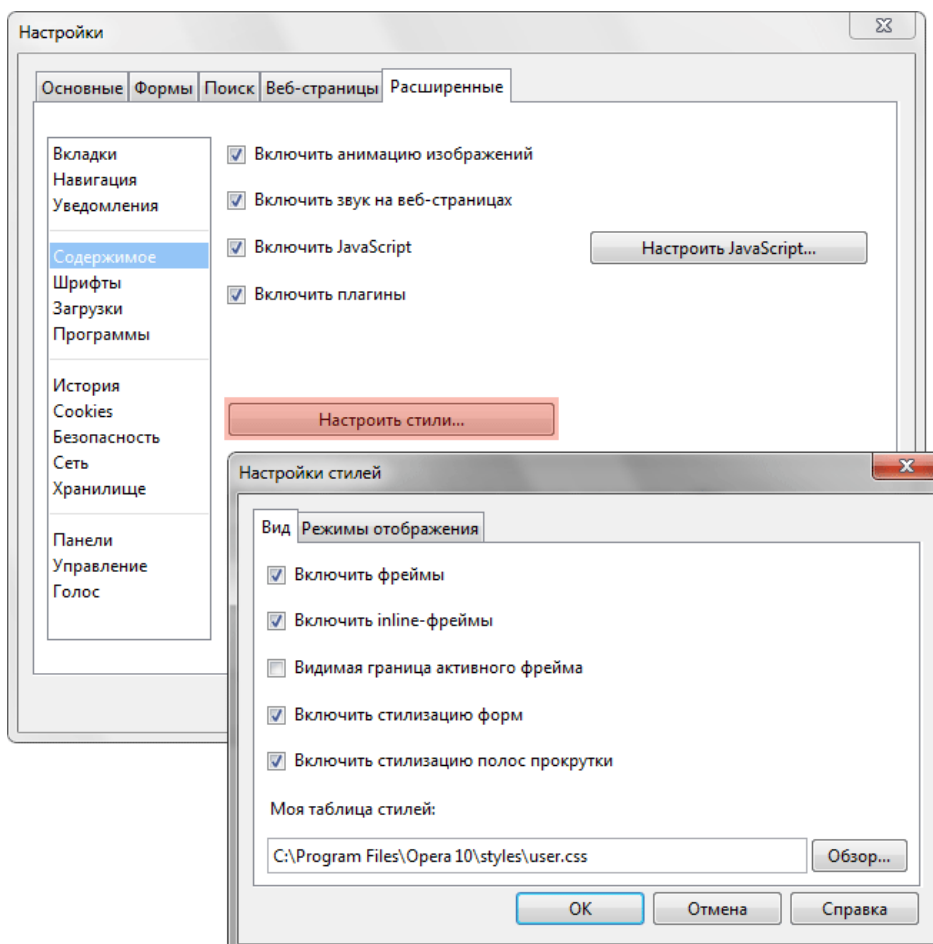


Рис. 1.4. Подключение стиля пользователя в браузере Opera

Указанные типы стилей могут спокойно существовать друг с другом, если они не пытаются изменить вид одного элемента. В случае возникновения противоречия вначале имеет приоритет стиль пользователя, затем стиль автора и последним идет стиль браузера.

Комплексные решения по созданию эффективных IT-инфраструктур в полном соответствии с бизнес задачами вашей компании



СЕРВЕРНЫЕ И СЕТЕВЫЕ РЕШЕНИЯ

Поставки серверного оборудования и программного обеспечения, а также создание комплексных решений на базе серверных технологий для построения информационных систем различного масштаба: от серверных решений начального уровня, до корпоративных систем крупных предприятий.

РЕШЕНИЯ ДЛЯ ПЕЧАТИ

Оптимизация и упрощение сетей устройств печати и копирования за счет выравнивания производительности работы конечных пользователей, улучшения технической поддержки и экономии затрат. Мы улучшим Вашу среду печати сэкономив время, деньги и технические ресурсы

КОМПЬЮТЕРНЫЕ РЕШЕНИЯ

Реализация компьютерных систем, а также разработка и производство персональных компьютеров под собственной торговой маркой "RestR". Мы можем предложить компьютеры любого уровня - от офисных и недорогих, до самых современных игровых и профессиональных рабочих станций.

РЕШЕНИЯ В ОБЛАСТИ ЭЛЕКТРОПИТАНИЯ

Конфигурирование и внедрение максимально эффективных комплексных инфраструктурных проектов на базе интегрированных решений APC, обеспечивающих защиту от перегрева и проблем с электропитанием, обеспечивая постоянную готовность оборудования.

Мы постоянно следим за развитием IT-отрасли. Установленные отношения с мировыми производителями - служат крепкой основой для предложения нашим партнерам выгодных и привлекательных решений, которые отличаются высоким качеством и безотказной работой.



Подробнее на нашем сайте WWW.RESTR.COM

ООО Рестрком 125480, г. Москва, ул. Героев Панфиловцев, д.10, кор.1. м.Планерная

Телефон: (495) 649-62-03 (многоканальный)

E-mail: info@restr.com

Способы добавления стилей на страницу

Для добавления стилей на веб-страницу существует несколько способов, которые различаются своими возможностями и назначением. Далее рассмотрим их подробнее.

Связанные стили

При использовании связанных стилей описание селекторов и их значений располагается в отдельном файле, как правило, с расширением `css`, а для связывания документа с этим файлом применяется тег `<link>`. Данный тег помещается в контейнер `<head>`, как показано в примере 1.3.

Пример 1.3. Подключение связанных стилей

XHTML 1.0 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Стили</title>
<link rel="stylesheet" type="text/css" href="style/mysite.css" />
<link rel="stylesheet" type="text/css" href="http://www.htmlbook.ru/main.css" />
</head>
<body>
<h1>Заголовок</h1>
<p>Текст</p>
</body>
</html>
```

Значения атрибутов тега `<link>` — `rel` и `type` остаются неизменными независимо от кода, как приведено в данном примере. Значение `href` задает путь к CSS-файлу, он может быть задан как относительно, так и абсолютно. Заметьте, что таким образом можно подключать таблицы стилей, которая находится на другом сайте.

Содержимое файла `mysite.css` подключаемого посредством тега `<link>` приведено в примере 1.4.

Пример 1.4. Файл со стилем

```
H1 {
  color: #000080;
  font-size: 2em;
  font-family: Arial, Verdana, sans-serif;
  text-align: center; /* Выравнивание по центру */
}
P {
  padding-left: 20px;
}
```

Как видно из данного примера, файл со стилем не хранит никаких данных, кроме синтаксиса CSS. В свою очередь и HTML-документ содержит только ссылку на файл со стилем, т.е. таким способом в полной мере реализуется принцип разделения кода и оформления сайта. Поэтому использование связанных стилей является наиболее универсальным и удобным методом добавления стиля на сайт. Ведь стили хранятся в одном файле, а в HTML-документах указывается только ссылка на него.

Глобальные стили

При использовании глобальных стилей свойства CSS описываются в самом документе и располагаются в заголовке веб-страницы. По своей гибкости и возможностям этот способ добавления стиля уступает предыдущему, но также позволяет хранить стили в одном месте, в данном случае прямо на странице с помощью контейнера `<style>`, как показано в примере 1.5.

Пример 1.5. Использование глобального стиля

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Глобальные стили</title>
<style type="text/css">
H1 {
  font-size: 1.2em;
  font-family: Verdana, Arial, Helvetica, sans-serif;
  color: #333366;
}
</style>
</head>
<body>
<h1>Hello, world!</h1>
</body>
</html>
```

В данном примере определен стиль тега `<h1>`, который затем можно повсеместно использовать на данной веб-

странице (рис. 1.5).

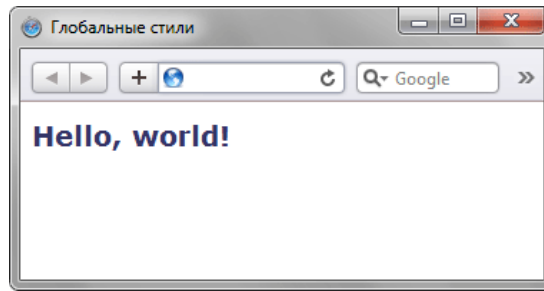


Рис. 1.5. Вид заголовка, оформленного с помощью стилей

Внутренние стили

Внутренний или встроенный стиль является по существу расширением для одиночного тега используемого на текущей веб-странице. Для определения стиля используется атрибут `style`, а его значением выступает набор стилевых правил (пример 1.6).

Пример 1.6. Использование внутреннего стиля

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>Внутренние стили</title>
  </head>
  <body>
    <p style="font-size: 120%; font-family: monospace; color: #cd66cc">Пример текста</p>
  </body>
</html>
```

В данном примере стиль тега `<p>` задается с помощью атрибута `style`, в котором через точку с запятой перечисляются стилевые свойства (рис. 1.6).

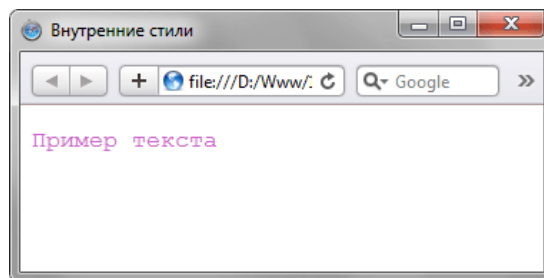


Рис. 1.6. Использование внутренних стилей для изменения вида текста



Внутренние стили рекомендуется применять на сайте ограниченно или вообще отказаться от их использования. Дело в том, что добавление таких стилей увеличивает общий объем файлов, что ведет к повышению времени их загрузки в браузере, и усложняет редактирование документов для разработчиков.

Все описанные методы использования CSS могут применяться как самостоятельно, так и в сочетании друг с другом. В этом случае необходимо помнить об их иерархии. Первым всегда применяется внутренний стиль, затем глобальный стиль и в последнюю очередь связанный стиль. В примере 1.7 применяется сразу два метода добавления стиля в документ.

Пример 1.7. Сочетание разных методов подключения стилей

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>Подключение стиля</title>
    <style type="text/css">
      h1 {
        font-size: 1.2em;
        font-family: Arial, Helvetica, sans-serif;
        color: green;
      }
    </style>
  </head>
  <body>
    <h1 style="font-size: 36px; font-family: Times, serif; color: red">Заголовок 1</h1>
    <h1>Заголовок 2</h1>
  </body>
```

```
</html>
```

В данном примере первый заголовок задается красным цветом размером 36 пикселей с помощью внутреннего стиля, а следующий — зеленым цветом через таблицу глобальных стилей (рис. 1.7).

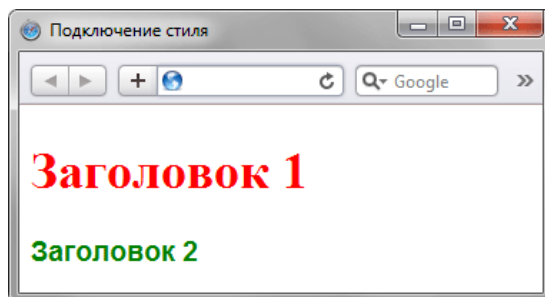


Рис. 1.7. Результат применения стилей

Импорт CSS

В текущую стилевую таблицу можно импортировать содержимое CSS-файла с помощью команды `@import`. Этот метод допускается использовать совместно со связанными или глобальными стилями, но никак не со встроенными стилями. Общий синтаксис следующий.

```
@import url("имя файла") типы носителей;  
@import "имя файла" типы носителей;
```

После ключевого слова `@import` указывается путь к стилевому файлу одним из двух приведенных способов — с помощью `url` или без него. В примере 1.8 показано, как можно импортировать стиль из внешнего файла в таблицу глобальных стилей.

Пример 1.8. Импорт CSS в глобальную таблицу стилей

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">  
<head>  
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  
<title>Импорт</title>  
<style type="text/css">  
  @import url("style/mysite.css");  
  H2 {  
    font-size: 1.2em;  
    font-family: Arial, Helvetica, sans-serif;  
    color: green;  
  }  
</style>  
</head>  
<body>  
<h1>Заголовок 1</h1>  
<h2>Заголовок 2</h2>  
</body>  
</html>
```

В данном примере показано подключение файла `mysite.css`, который расположен в папке `style`.

Аналогично происходит импорт и в файле со стилем, который затем подключается к документу (пример 1.9).

Пример 1.9. Импорт в таблице связанных стилей

```
@import "/style/print.css";  
@import "/style/palm.css";  
BODY {  
  font-family: Arial, Verdana, Helvetica, sans-serif;  
  font-size: 90%;  
  background: white;  
  color: black;  
}
```

В данном примере показано сдержимое файла `mysite.css`, который добавляется к нужным документам способом, показанным в примере 1.3, а именно с помощью тега `<link>`.

Типы носителей

Широкое развитие различных платформ и устройств заставляет разработчиков делать под них специальные версии сайтов, что достаточно трудоемко и проблематично. Вместе с тем, времена и потребности меняются, и создание сайта для различных устройств является неизбежным и необходимым звеном его развития. С учетом этого в CSS введено понятие типа носителя, когда стиль применяется только для определенного устройства. В табл. 1.1 перечислены некоторые типы носителей.

Табл. 1.1. Типы носителей и их описание

Тип	Описание
all	Все типы. Это значение используется по умолчанию.
aural	Речевые синтезаторы, а также программы для воспроизведения текста вслух. Сюда, например, можно отнести речевые браузеры.
braille	Устройства, основанные на системе Брайля, которые предназначены для слепых людей.
handheld	Наладонные компьютеры и аналогичные им аппараты.
print	Печатающие устройства вроде принтера.
projection	Проектор.
screen	Экран монитора.
tv	Телевизор.

В CSS для указания типа носителей применяются команды `@import` и `@media`, с помощью которых можно определить стиль для элементов в зависимости от того, выводится документ на экран или на принтер.



Ключевые слова `@media` и `@import` относятся к эт-правилам. Такое название произошло от наименования символа `@` — «эт», с которого и начинаются эти ключевые слова. В рунете для обозначения символа `@` применяется устоявшийся термин «собака». Только вот использовать выражение «собачье правило» язык не поворачивается.

При импортировании стиля через команду `@import` тип носителя указывается после адреса файла. При этом допускается задавать сразу несколько типов, упоминая их через запятую, как показано в примере 1.10.

Пример 1.10. Импорт стилового файла

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>Импорт стиля</title>
    <style type="text/css">
      @import "style/main.css" screen; /* Стиль для вывода результата на монитор */
      @import "style/print.css" print, handheld; /* Стиль для печати и смартфона */
    </style>
  </head>
  <body>
    <p>...</p>
  </body>
</html>
```

В данном примере импортируется два файла — `main.css` предназначен для изменения вида документа при его просмотре на экране монитора, и `print.css` — при печати страницы и отображении на смартфоне.



Браузер Internet Explorer до седьмой версии включительно не поддерживает типы носителей при импорте стилового файла. Более того, при добавлении типа носителя стилового файл вообще не загружается.

Команда `@media` позволяет указать тип носителя для глобальных или связанных стилей и в общем случае имеет следующий синтаксис.

```
@media тип носителя 1 {
  Описание стиля для типа носителя 1
}
@media тип носителя 2 {
  Описание стиля для типа носителя 2
}
```

После ключевого слова `@media` идет один или несколько типов носителя, перечисленных в табл. 1.1, если их больше

одного, то они разделяются между собой запятой. После чего следуют обязательные фигурные скобки, внутри которых идет обычное описание стилевых правил. В примере 1.11 показано, как задать разный стиль для печати и отображения на мониторе.

Пример 1.11. Стили для разных типов носителей

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Типы носителей</title>
<style type="text/css">
@media screen { /* Стил для отображения в браузере */
BODY {
font-family: Arial, Verdana, sans-serif; /* Рубленый шрифт */
font-size: 0.9em; /* Размер шрифта */
color: #000080; /* Цвет текста */
}
H1 {
background: #faf0e6; /* Цвет текста */
border: 2px dashed maroon; /* Рамка вокруг заголовка */
color: #a0522d; /* Цвет текста */
padding: 7px; /* Поля вокруг текста */
}
H2 {
color: #556b2f; /* Цвет текста */
margin: 0; /* Убираем отступы */
}
P {
margin-top: 0.5em; /* Отступ сверху */
}
}
@media print { /* Стил для печати */
BODY {
font-family: Times, 'Times New Roman', serif; /* Шрифт с засечками */
}
H1, H2, P {
color: black; /* Черный цвет текста */
}
}
</style>
</head>
<body>
<h1>Как поймать льва в пустыне</h1>
<h2>Метод случайных чисел</h2>
<p>Разделим пустыню на ряд элементарных прямоугольников, размер
которых совпадает с размером клетки для льва. После чего перебираем
полученные прямоугольники, каждый раз выбирая заданную
область случайным образом. Если в данной области окажется лев,
то мы поймаем его, накрыв клеткой.</p>
</body>
</html>
```

В данном примере вводится два стиля — один для изменения вида элементов при их обычном отображении в браузере, а второй — при выводе страницы на печать. При этом облик документа для разных носителей может сильно различаться между собой, например, как это показано на рис. 1.8 и рис. 1.9.

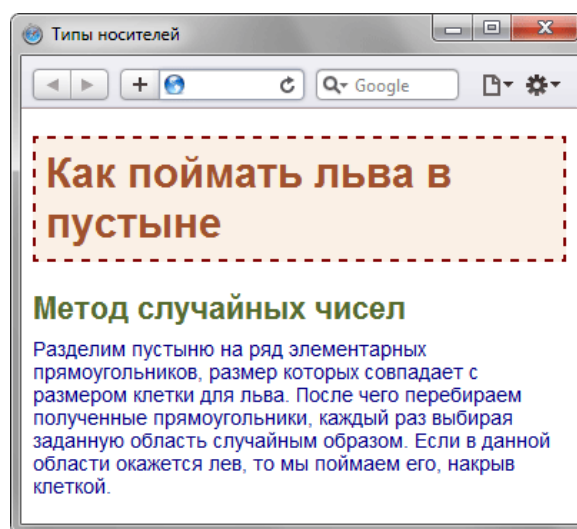


Рис. 1.8. Страница для отображения в окне браузера

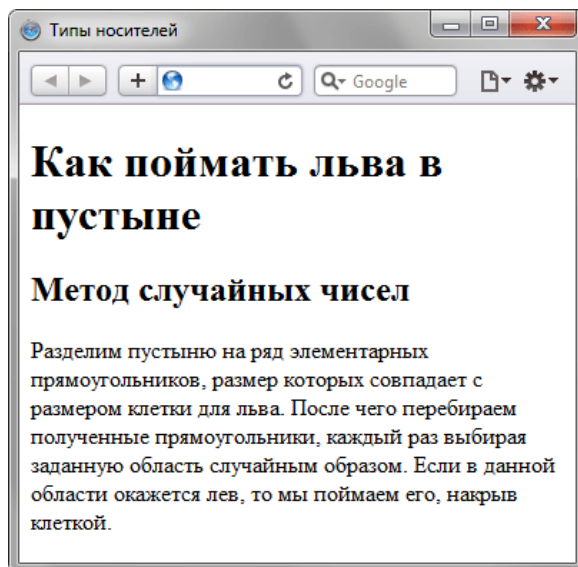


Рис. 1.9. Страница, предназначенная для печати

Просмотреть документ, у которого CSS установлен как тип `print` можно, если распечатать определенную страницу. Или пойти на хитрость и временно заменить `print` на `screen`, чтобы отобразить итог в браузере. Именно так был получен рис. 1.9.

Команда `@media` применяется в основном для формирования одного стилевого файла, который разбит на блоки по типу устройств. Иногда же имеет смысл создать несколько разных CSS-файлов — один для печати, другой для отображения в браузере — и подключать их к документу по мере необходимости. В подобном случае следует воспользоваться тегом `<link>` с атрибутом `media`, значением которого выступают все те же типы, перечисленные в табл. 1.1.

В примере 1.12 показано, как создавать ссылки на CSS-файлы, которые предназначены для разных типов носителей.

Пример 1.12. Подключение стилей для разных носителей

XHTML 1.0 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Разные носители</title>
<link media="print, handheld" rel="stylesheet" href="style/print.css" type="text/css" />
<link media="screen" rel="stylesheet" href="style/main.css" type="text/css" />
</head>
<body>
<p>...</p>
</body>
</html>
```

В данном примере используются две таблицы связанных стилей, одна для отображения в браузере, а вторая — для печати документа и его просмотра на смартфоне. Хотя на страницу загружаются одновременно два разных стиля, применяются они только для определенных устройств.

Аналогично можно использовать и глобальные стили, добавляя параметр `media` к тегу `<style>` (пример 1.13).

Пример 1.13. Стиль для смартфона

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Разные носители</title>
<style type="text/css" media="handheld">
BODY {
width: 320px; /* Ширина страницы */
}
</style>
</head>
<body>
<p>Разделим пустыню на ряд элементарных прямоугольников, размер
которых совпадает с размером клетки для льва. После чего
перебираем полученные прямоугольники, каждый раз выбирая заданную
область случайным образом. Если в данной области окажется лев,
то мы поймем его, накрыв клеткой.</p>
</body>
</html>
```

В данном примере ширина для устройств типа `handheld` ограничена размером 320 пикселей.

Базовый синтаксис CSS

Как уже было отмечено ранее, стилевые правила записываются в своем формате, отличном от HTML. Основным понятием выступает селектор — это некоторое имя стиля, для которого добавляются параметры форматирования. В качестве селектора выступают теги, классы и идентификаторы. Общий способ записи имеет следующий вид.

селектор свойство значение

```
body { background: #ffc910; }
```

Вначале пишется имя селектора, например, **TABLE**, это означает, что все стилевые правила будут применяться к тегу `<table>`, затем идут фигурные скобки, в которых записывается стилевое свойство, а его значение указывается после двоеточия. Стилевые свойства разделяются между собой точкой с запятой, в конце этот символ можно опустить.

CSS не чувствителен к регистру, переносу строк, пробелам и символам табуляции, поэтому форма записи зависит от желания разработчика. Так, в примере 1.14 показаны две разновидности оформления селекторов и их правил.

Пример 1.14. Использование стилей

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Заголовки</title>
<style type="text/css">
H1 { color: #a6780a; font-weight: normal; }
H2 {
color: olive;
border-bottom: 2px solid black;
}
</style>
</head>
<body>
<h1>Заголовок 1</h1>
<h2>Заголовок 2</h2>
</body>
</html>
```

В данном примере свойства селектора **H1** записаны в одну строку, а для селектора **H2** каждое свойство находится на отдельной строке. Во втором случае легче отыскивать нужные свойства и править их по необходимости, но при этом незначительно возрастает объем данных за счет активного использования пробелов и переносов строк. Так что в любом случае способ оформления стилевых правил зависит от разработчика.

Правила применения стилей

Далее приведены некоторые правила, которые необходимо знать при описании стиля.

Форма записи

Для селектора допускается добавлять каждое стилевое свойство и его значение по отдельности, как это показано в примере 1.15.

Пример 1.15. Расширенная форма записи

```
td { background: olive; }
td { color: white; }
td { border: 1px solid black; }
```

Однако такая запись не очень удобна. Приходится повторять несколько раз один и тот же селектор, да и легко запутаться в их количестве. Поэтому пишите все свойства для каждого селектора вместе. Указанный набор записей в таком случае получит следующий вид (пример 1.16).

Пример 1.16. Компактная форма записи

```
td {
background: olive;
color: white;
border: 1px solid black;
}
```

Эта форма записи более наглядная и удобная в использовании.

Имеет приоритет значение, указанное в коде ниже

Если для селектора вначале задается свойство с одним значением, а затем то же свойство, но уже с другим

значением, то применяться будет то значение, которое в коде установлено ниже (пример 1.17).

Пример 1.17. Разные значения у одного свойства

```
P { color: green; }  
P { color: red; }
```

В данном примере для селектора `P` цвет текста вначале задается зеленым, а затем красным. Поскольку значение `red` расположено ниже, то оно в итоге и будет применяться к тексту.

На самом деле такой записи лучше вообще избегать и удалять повторяющиеся значения. Но подобное может произойти не явно, например, в случае подключения разных стилевых файлов, в которых содержатся одинаковые селекторы.

Значения

У каждого свойства может быть только соответствующее его функции значение. Например, для `color`, который устанавливает цвет текста, в качестве значений недопустимо использовать числа.

Комментарии

Комментарии нужны, чтобы делать пояснения по поводу использования того или иного стилевого свойства, выделять разделы или писать свои заметки. Комментарии позволяют легко вспоминать логику и структуру селекторов, и повышают разборчивость кода. Вместе с тем, добавление текста увеличивает объем документов, что отрицательно сказывается на времени их загрузки. Поэтому комментарии обычно применяют в отладочных или учебных целях, а при выкладывании сайта в сеть их стирают.

Чтобы пометить, что текст является комментарием, применяют следующую конструкцию `/* ... */` (пример 1.18).

Пример 1.18. Комментарии в CSS-файле

```
/*  
  Стилль для сайта htmlbook.ru  
  Сделан для ознакомительных целей  
*/  
  
div {  
  width: 200px; /* Ширина контента */  
  margin: 10px; /* Поля вокруг элемента */  
  float: left; /* Обтекание по правому краю */  
}
```

Как следует из данного примера, комментарии можно добавлять в любое место CSS-документа, а также писать текст комментария в несколько строк. Вложенные комментарии недопустимы.

Значения стилевых свойств

Все многообразие значений стилевых свойств может быть сведено к определенному типу: строка, число, проценты, размер, цвет, адрес или ключевое слово.

Строки

Любые строки необходимо брать в двойные или одинарные кавычки. Если внутри строки требуется оставить одну или несколько кавычек, то можно комбинировать типы кавычек или добавить перед кавычкой слэш (пример 1.19).

Пример 1.19. Допустимые строки

```
'Гостиница "Турист"'
"Гостиница 'Турист'"
"Гостиница \"Турист\""
```

В данном примере в первой строке применяются одинарные кавычки, а слово «Турист» взято в двойные кавычки. Во второй строке все с точностью до наоборот, в третьей же строке используются только двойные кавычки, но внутренние экранированы с помощью слэша.

Числа

Значением может выступать целое число, содержащее цифры от 0 до 9 и десятичная дробь, в которой целая и десятичная часть разделяются точкой (пример 1.20).

Пример 1.20. Числа в качестве значений

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Числа</title>
<style type="text/css">
P {
font-weight: 600; /* Жирное начертание */
line-height: 1.2; /* Межстрочный интервал */
}
</style>
</head>
<body>
<p>Пример текста</p>
</body>
</html>
```

Если в десятичной дроби целая часть равна нулю, то ее разрешается не писать. Запись `.7` и `0.7` равнозначна.

Проценты

Процентная запись обычно применяется в тех случаях, когда надо изменить значение относительно родительского элемента или когда размеры зависят от внешних условий. Так, ширина таблицы 100% означает, что она будет подстраиваться под размеры окна браузера и меняться вместе с шириной окна (пример 1.21).

Пример 1.21. Процентная запись

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Ширина в процентах</title>
<style type="text/css">
TABLE {
width: 100%; /* Ширина таблицы в процентах */
background: #f0f0f0; /* Цвет фона */
}
</style>
</head>
<body>
<table>
<tr><td>Содержимое таблицы</td></tr>
</table>
</body>
</html>
```

Проценты не обязательно должны быть целым числом, допускается использовать десятичные дроби, вроде значения `56.8%`, но не всегда.

Размеры

Для задания размеров различных элементов, в CSS используются абсолютные и относительные единицы измерения.

Абсолютные единицы не зависят от устройства вывода, а относительные единицы определяют размер элемента относительно значения другого размера.

Относительные единицы

Относительные единицы обычно используют для работы с текстом, либо когда надо вычислить процентное соотношение между элементами. В табл. 1.2 перечислены основные относительные единицы.

Табл. 1.2. Относительные единицы измерения

Единица	Описание
em	Размер шрифта текущего элемента
ex	Высота символа x
px	Пиксел
%	Процент

Единица **em** это изменяемое значение, которое зависит от размера шрифта текущего элемента (размер устанавливается через стилевое свойство **font-size**). В каждом браузере заложен размер текста, применяемый в том случае, когда этот размер явно не задан. Поэтому изначально **1em** равен размеру шрифта, заданного в браузере по умолчанию или размеру шрифта родительского элемента. Процентная запись идентична **em**, в том смысле, что значения **1em** и **100%** равны.

Единица **ex** определяется как высота символа «x» в нижнем регистре. На **ex** распространяются те же правила, что и для **em**, а именно, он привязан к размеру шрифта, заданного в браузере по умолчанию, или к размеру шрифта родительского элемента.

Пиксел это элементарная точка, отображаемая монитором или другим подобным устройством, например, смартфоном. Размер пиксела зависит от разрешения устройства и его технических характеристик. В примере 1.22 показано применение пикселей и **em** для задания размера шрифта.

Пример 1.22. Использование относительных единиц

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>Относительные единицы</title>
    <style type="text/css">
      h1 { font-size: 30px; }
      P { font-size: 1.5em; }
    </style>
  </head>
  <body>
    <h1>Заголовок размером 30 пикселей</h1>
    <p>Размер текста 1.5 em</p>
  </body>
</html>
```

Результат данного примера показан ниже (рис. 1.10).

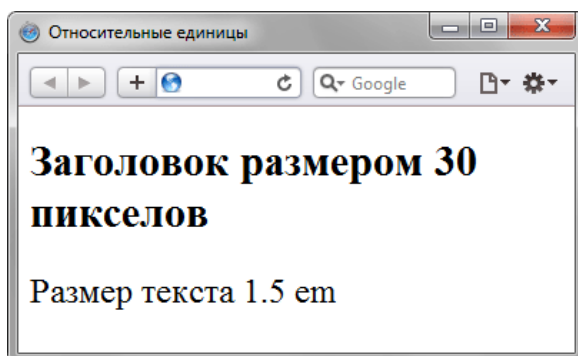


Рис. 1.10. Размер текста при различных единицах

Абсолютные единицы

Абсолютные единицы применяются реже, чем относительные и обычно при работе с текстом. В табл. 1.3 перечислены основные абсолютные единицы.

Табл. 1.3. Абсолютные единицы измерения

Единица	Описание
---------	----------

in	Дюйм (1 дюйм равен 2,54 см)
cm	Сантиметр
mm	Миллиметр
pt	Пункт (1 пункт равен 1/72 дюйма)
pc	Пика (1 пика равна 12 пунктам)

Самой, пожалуй, распространенной единицей является пункт, который используется для указания размера шрифта. Хотя мы привыкли измерять все в миллиметрах и подобных единицах, пункт, пожалуй, единственная величина из не метрической системы измерения, которая используется у нас повсеместно. И все благодаря текстовым редакторам и издательским системам. В примере 1.23 показано использование пунктов и миллиметров.

Пример 1.23. Использование абсолютных единиц

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Абсолютные единицы</title>
<style type="text/css">
H1 { font-size: 24pt; }
P { margin-left: 30mm; }
</style>
</head>
<body>
<h1>Заголовок размером 24 пункта</h1>
<p>Сдвиг текста вправо на 30 миллиметров</p>
</body>
</html>
```

Результат использования абсолютных единиц измерения показан ниже (рис. 1.11).

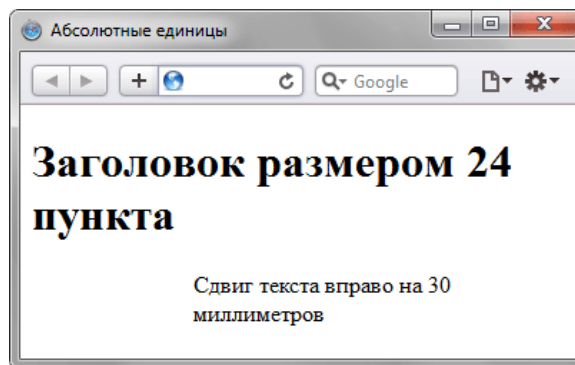


Рис. 1.11. Размер текста при различных единицах

При установке размеров обязательно указывайте единицы измерения, например `width: 30px`. В противном случае браузер не сможет показать желаемый результат, поскольку не понимает, какой размер вам требуется. Единицы не добавляются только при нулевом значении (`margin: 0`).

Цвет

Цвет в стилях можно задавать разными способами: по шестнадцатеричному значению, по названию, в формате RGB, RGBA, HSL, HSLA.

По шестнадцатеричному значению

Для задания цветов используются числа в шестнадцатеричном коде. Шестнадцатеричная система, в отличие от десятичной системы, базируется, как следует из ее названия, на числе 16. Цифры будут следующие: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Цифры от 10 до 15 заменены латинскими буквами. Числа больше 15 в шестнадцатеричной системе образуются объединением двух чисел в одно. Например, числу 255 в десятичной системе соответствует число FF в шестнадцатеричной системе. Чтобы не возникало путаницы в определении системы счисления, перед шестнадцатеричным числом ставят символ решетки #, например #666999. Каждый из трех цветов — красный, зеленый и синий — может принимать значения от 00 до FF. Таким образом, обозначение цвета разбивается на три составляющие #rrggbb, где первые два символа отмечают красную компоненту цвета, два средних — зеленую, а два последних — синюю. Допускается использовать сокращенную форму вида #rgb, где каждый символ следует удваивать. Так, запись #fe0 следует расценивать как #fee00.

По названию

Браузеры поддерживают некоторые цвета по их названию. В табл. 1.4 приведены названия, шестнадцатеричный код,

значения в формате RGB, HSL и описание.

Табл. 1.4. Названия цветов

Имя	Цвет	Код	RGB	HSL	Описание
white		#ffffff или #fff	rgb(255,255,255)	hsl(0,0%,100%)	Белый
silver		#c0c0c0	rgb(192,192,192)	hsl(0,0%,75%)	Серый
gray		#808080	rgb(128,128,128)	hsl(0,0%,50%)	Темно-серый
black		#000000 или #000	rgb(0,0,0)	hsl(0,0%,0%)	Черный
maroon		#800000	rgb(128,0,0)	hsl(0,100%,25%)	Темно-красный
red		#ff0000 или #f00	rgb(255,0,0)	hsl(0,100%,50%)	Красный
orange		#ffa500	rgb(255,165,0)	hsl(38.8,100%,50%)	Оранжевый
yellow		#ffff00 или #ff0	rgb(255,255,0)	hsl(60,100%,50%)	Желтый
olive		#808000	rgb(128,128,0)	hsl(60,100%,25%)	Оливковый
lime		#00ff00 или #0f0	rgb(0,255,0)	hsl(120,100%,50%)	Светло-зеленый
green		#008000	rgb(0,128,0)	hsl(120,100%,25%)	Зеленый
aqua		#00ffff или #0ff	rgb(0,255,255)	hsl(180,100%,50%)	Голубой
blue		#0000ff или #00f	rgb(0,0,255)	hsl(240,100%,50%)	Синий
navy		#000080	rgb(0,0,128)	hsl(240,100%,25%)	Темно-синий
teal		#008080	rgb(0,128,128)	hsl(180,100%,25%)	Сине-зеленый
fuchsia		#ff00ff или #f0f	rgb(255,0,255)	hsl(300,100%,50%)	Розовый
purple		#800080	rgb(128,0,128)	hsl(300,100%,25%)	Фиолетовый

С помощью RGB

Можно определить цвет, используя значения красной, зеленой и синей составляющей в десятичном исчислении. Каждая из трех компонент цвета принимает значение от 0 до 255. Также допустимо задавать цвет в процентном отношении, при этом 100% будет соответствовать числу 255. Вначале указывается ключевое слово **rgb**, а затем в скобках, через запятую указываются компоненты цвета, например `rgb(255, 128, 128)` или `rgb(100%, 50%, 50%)`.

RGBA

Формат RGBA похож по синтаксису на RGB, но включает в себя альфа-канал, задающий прозрачность элемента. Значение 0 соответствует полной прозрачности, 1 — непрозрачности, а промежуточное значение вроде 0.5 — полупрозрачности.

RGBA добавлен в CSS3, поэтому валидацию CSS-кода надо проводить именно по этой версии. Следует отметить, что стандарт CSS3 еще находится в разработке и некоторые возможности в нем могут поменяться. К примеру, цвет в формате RGB добавленный к свойству `background-color` проходит валидацию, а добавленный к свойству `background` уже нет. При этом браузеры вполне корректно понимают цвет для того и другого свойства. В табл. 1.4 приведен список браузеров, которые поддерживают RGBA; в браузерах, где этот формат не работает, значение цвета будет игнорироваться.

Табл. 1.4. Поддержка браузерами формата RGBA

Internet Explorer				Chrome				Opera				Safari			Firefox			
6.0	7.0	8.0	9.0	5.0	6.0	7.0	8.0	9.2	9.6	10	11	3.1	4.0	5.0	2.0	3.0	3.6	4.0
✘	✘	✘	✘	✔	✔	✔	✔	✘	✘	✔	✔	✔	✔	✔	✘	✔	✔	✔

HSL

Название формата HSL образовано от сочетания первых букв Hue (оттенок), Saturate (насыщенность) и Lightness (светлота). Оттенок это значение цвета на цветовом круге (рис. 1.12) и задается в градусах. 0° соответствует красному цвету, 120° — зеленому, а 240° — синему. Значение оттенка может изменяться от 0 до 359.



Рис. 1.12. Цветовой круг

Насыщенностью называется интенсивность цвета, измеряется в процентах от 0% до 100%. Значение 0% обозначает отсутствие цвета и оттенок серого, 100% максимальное значение насыщенности.

Светлота задает, насколько цвет яркий и указывается в процентах от 0% до 100%. Малые значения делают цвет темнее, а высокие светлее, крайние значения 0% и 100% соответствуют чёрному и белому цвету.

Примеры использования HSL приведены в табл. 1.4. За исключением Internet Explorer все современные браузеры корректно работают с этим форматом (табл. 1.5).

Табл. 1.5. Поддержка браузерами формата HSL

Internet Explorer				Chrome				Opera				Safari			Firefox			
6.0	7.0	8.0	9.0	5.0	6.0	7.0	8.0	9.2	9.5	10	11	3.1	4.0	5.0	2.0	3.0	3.6	4.0
✘	✘	✘	✘	✔	✔	✔	✔	✘	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔

HSLA

Формат HSLA похож по синтаксису на HSL, но включает в себя альфа-канал, задающий прозрачность элемента. Значение 0 соответствует полной прозрачности, 1 — непрозрачности, а промежуточное значение вроде 0.5 — полупрозрачности. По сравнению с HSL поддержка браузерами немного другая (табл. 1.6).

Табл. 1.6. Поддержка браузерами формата HSLA

Internet Explorer				Chrome				Opera				Safari			Firefox			
6.0	7.0	8.0	9.0	5.0	6.0	7.0	8.0	9.2	9.6	10	11	3.1	4.0	5.0	2.0	3.0	3.6	4.0
✘	✘	✘	✘	✔	✔	✔	✔	✘	✘	✔	✔	✔	✔	✔	✘	✔	✔	✔

Значения цвета в форматах RGBA, HSL и HSLA добавлены в CSS3, поэтому при использовании этих форматов проверяйте код на валидность с учётом версии.

В примере 1.24 представлены различные способы задания цветов элементов веб-страниц.

Пример 1.24. Представление цвета

XHTML 1.0 | CSS 2.1 | CSS 3 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```

<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Цвета</title>
<style type="text/css">
BODY {
background-color: #F9F2E3;
}
H2 {
background-color: rgb(214, 86, 43);
color: rgba(255, 255, 255, .9);
padding: 10px;
}
P {
color: green;
}
DIV {
background-color: hsl(60, 100%, 25%);
color: hsla(120, 100%, 50%, 0.1);
}
</style>
</head>
<body>
<h2>Предупреждение</h2>
<p>Все перечисленные на сайте методы ловли льва являются теоретическими
и базируются на вычислительных методах. Авторы не гарантируют вашей
безопасности при их использовании и снимают с себя всякую
ответственность за результат. Помните, лев это хищник и
опасное животное!</p>
<div>Аpppprx!</div>

```

```
</body>
</html>
```

Результат данного примера показан на рис. 1.13.

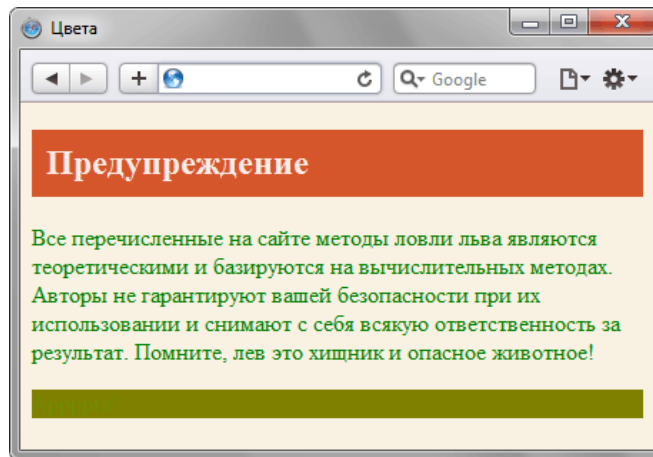


Рис. 1.13. Цвета на веб-странице

Адреса

Адреса применяются для указания пути к файлу, например, для установки фоновой картинки на странице. Для этого применяется ключевое слово `url()`, внутри скобок пишется относительный или абсолютный адрес файла. При этом адрес можно задавать в необязательных одинарных или двойных кавычках (пример 1.25).

Пример 1.25. Адрес графического файла

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Добавление фона</title>
<style type="text/css">
BODY {
background: url('http://webimg.ru/images/156_1.png') no-repeat;
}
DIV {
background: url(images/warning.png) no-repeat;
padding-left: 140px;
}
</style>
</head>
<body>
<div>Внимание, запрашиваемая страница не найдена!</div>
</body>
</html>
```

В данном примере в селекторе `BODY` используется абсолютный адрес к графическому файлу, а в селекторе `DIV` — относительный.

Ключевые слова

В качестве значений активно применяются ключевые слова, которые определяют желаемый результат действия стилевых свойств. Ключевые слова пишутся без кавычек.

```
Правильно: P { text-align: right; }
Неверно: P { text-align: "right"; }
```

inherit

Ключевое слово, которое сообщает, что необходимо наследовать значение свойства у родительского элемента. Естественно, результат будет заметен только в том случае, если у родителя указанное свойство установлено. Браузер Internet Explorer до версии 7.0 включительно не поддерживает значение `inherit`.

initial

Значение `initial` применяется для установки исходного значения свойства. Может пригодиться в нескольких случаях, к примеру, восстановить значения свойств, заданных браузером по умолчанию или задать начальное значение свойства, измененное в результате наследования. Ключевое слово `initial` добавлено в CSS3 и пока плохо поддерживается браузерами (табл. 1.7).

Табл. 1.7. Поддержка браузерами значения `initial`

Браузер	Internet Explorer	Chrome	Opera	Safari	Firefox
Версия	—	2.0+	—	2.0+	1.0+
Значение	—	initial	—	initial	-moz-initial

Значение `-moz-initial` является нестандартным, поэтому его применение приведёт к невалидному коду CSS.

В примере 1.26 показаны некоторые аспекты применения `initial`.

Пример 1.26. Использование `initial`

XHTML 1.0 | CSS 2.1 | CSS 3 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```

<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>initial</title>
<style type="text/css">
H2 {
color: #ffb734;
font-family: Arial, sans-serif;
}
P {
color: green;
}
.initial {
color: initial;
color: -moz-initial;
font-family: initial;
font-family: -moz-initial;
}
</style>
</head>
<body>
<h2>Метод случайных чисел</h2>
<p>Разделим пустыню на <span class="initial">ряд элементарных
прямоугольников</span>, размер которых совпадает с размером
клетки для льва. После чего перебираем полученные прямоугольники,
каждый раз выбирая заданную область случайным образом.
Если в данной области окажется лев, то мы поймаем его, накрыв
клеткой.</p>
<h2 class="initial">Метод Гаусса</h2>
</body>
</html>

```

В данном примере изменяется цвет текста и шрифт заголовка. Цвет текста внутри тега `` явно не задаётся, но он наследует значение цвета своего родителя — тега `<p>`. С помощью класса `initial` цвет фрагмента текста устанавливается исходным, по умолчанию он черный. Аналогично обстоит дело и с заголовком, через стили задается его цвет и шрифт. Чтобы восстановить стиль по умолчанию, к тегу `<h1>` добавляется класс `initial`, в котором используется значение `initial` (рис. 1.14).

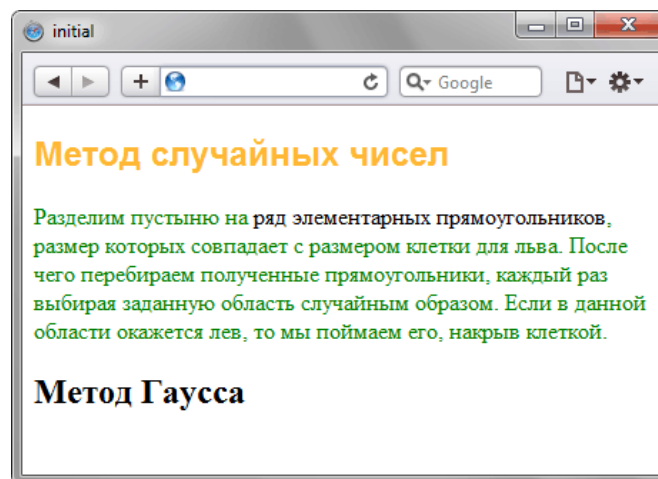


Рис. 1.14. Результат использования `initial`

Селекторы тегов

В качестве селектора может выступать любой тег HTML для которого определяются правила форматирования, такие как: цвет, фон, размер и т.д. Правила задаются в следующем виде.

```
Тег { свойство1: значение; свойство2: значение; ... }
```

Вначале указывается имя тега, оформление которого будет переопределено, заглавными или строчными символами не имеет значения. Внутри фигурных скобок пишется стилевое свойство, а после двоеточия — его значение. Набор свойств разделяется между собой точкой с запятой и может располагаться как в одну строку, так и в несколько (пример 1.27).

Пример 1.27. Изменение стиля тега <p>

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Селекторы тегов</title>
<style type="text/css">
  P {
    text-align: justify; /* Выравнивание по ширине */
    color: green; /* Зеленый цвет текста */
  }
</style>
</head>
<body>
<p>Более эффективным способом ловли льва в пустыне
является метод золотого сечения. При его использовании пустыня делится
на две неравные части, размер которых подчиняется правилу золотого
сечения.</p>
</body>
</html>
```

В данном примере изменяется цвет текста и выравнивание текста абзаца. Стиль будет применяться только к тексту, который располагается внутри контейнера <p>.

Следует понимать, что хотя стиль можно применить к любому тегу, результат будет заметен только для тегов, которые непосредственно отображаются в контейнере <body>.

Классы

Классы применяют, когда необходимо определить стиль для индивидуального элемента веб-страницы или задать разные стили для одного тега. При использовании совместно с тегами синтаксис для классов будет следующий.

```
Тег.Имя класса { свойство1: значение; свойство2: значение; ... }
```

Внутри стиля вначале пишется желаемый тег, а затем, через точку пользовательское имя класса. Имена классов должны начинаться с латинского символа и могут содержать в себе символ дефиса (-) и подчеркивания (_). Использование русских букв в именах классов недопустимо. Чтобы указать в коде HTML, что тег используется с определенным классом, к тегу добавляется атрибут **class** значением которого выступает имя класса (пример 1.28).

Пример 1.28. Использование классов

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Классы</title>
<style type="text/css">
P { /* Обычный абзац */
text-align: justify; /* Выравнивание текста по ширине */
}
P.cite { /* Абзац с классом cite */
color: navy; /* Цвет текста */
margin-left: 20px; /* Отступ слева */
border-left: 1px solid navy; /* Граница слева от текста */
padding-left: 15px; /* Расстояние от линии до текста */
}
</style>
</head>
<body>
<p>Для искусственного освещения помещения применяются люминесцентные лампы.
Они отличаются высокой световой отдачей, продолжительным сроком службы,
малой яркостью светящейся поверхности, близким к естественному спектральным
составом излучаемого света, что обеспечивает хорошую цветопередачу.</p>
<p class="cite">Для исключения засветки экрана дисплея световыми
потоками оконные проемы снабжены светорассеивающими шторами.</p>
</body>
</html>
```

Результат данного примера показан на рис. 1.15.

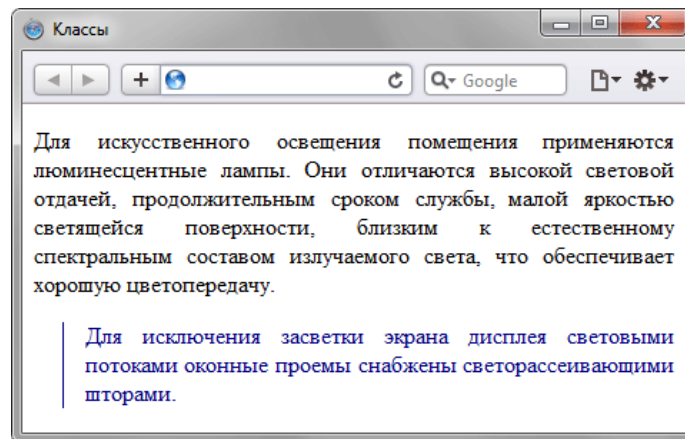


Рис. 1.15. Вид текста, оформленного с помощью стилевых классов

Первый абзац выровнен по ширине с текстом черного цвета (этот цвет задается браузером по умолчанию), а следующий, к которому применен класс с именем `cite` — отображается синим цветом и с линией слева.

Можно, также, использовать классы и без указания тега. Синтаксис в этом случае будет следующий.

```
.Имя класса { свойство1: значение; свойство2: значение; ... }
```

При такой записи, класс можно применять к любому тегу (пример 1.29).

Пример 1.29. Использование классов

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Классы</title>
<style type="text/css">
```

```

.gost {
  color: green; /* Цвет текста */
  font-weight: bold; /* Жирное начертание */
}
.term {
  border-bottom: 1px dashed red; /* Подчеркивание под текстом */
}
</style>
</head>
<body>
<p>Согласно <span class="gost">ГОСТ 12.1.003-83 ССБТ &quot;Шум. Общие
требования безопасности&quot;</span>, шумовой характеристикой рабочих
мест при постоянном шуме являются уровни звуковых давлений в децибелах
в октавных полосах. Совокупность таких уровней называется
<b class="term">пределным спектром</b>, номер которого численно равен
уровню звукового давления в октавной полосе со среднегеометрической
частотой 1000&nbsp;Гц.</p>
</body>
</html>

```

Результат применения классов к тегам `` и `` показан на рис. 1.16.

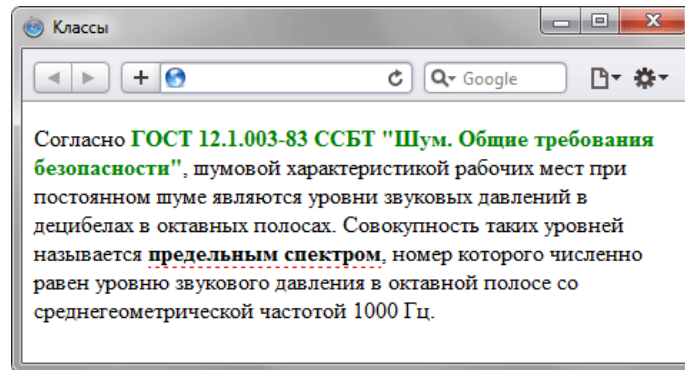


Рис. 1.16. Вид тегов, оформленных с помощью классов

Классы удобно использовать, когда нужно применить стиль к разным элементам веб-страницы: ячейкам таблицы, ссылкам, абзацам и др. В примере 1.30 показано изменение цвета фона строк таблицы для создания «зебры».

Пример 1.30. Использование классов

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

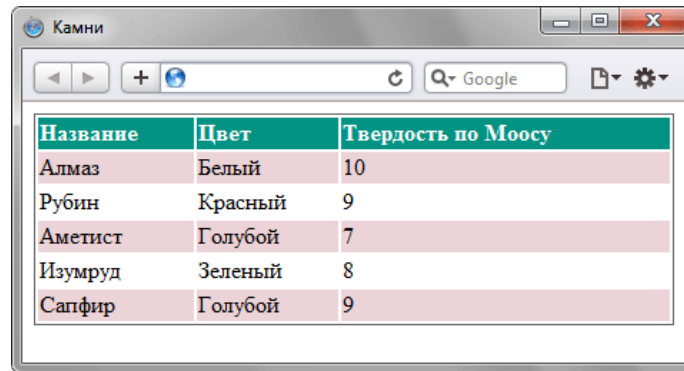
```

<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Камни</title>
<style type="text/css">
table.jewel {
  width: 100%; /* Ширина таблицы */
  border: 1px solid #666; /* Рамка вокруг таблицы */
}
th {
  background: #009383; /* Цвет фона */
  color: #fff; /* Цвет текста */
  text-align: left; /* Выравнивание по левому краю */
}
tr.odd {
  background: #ebd3d7; /* Цвет фона */
}
</style>
</head>
<body>
<table class="jewel">
<tr>
<th>Название</th><th>Цвет</th><th>Твердость по Моосу</th>
</tr>
<tr class="odd">
<td>Алмаз</td><td>Белый</td><td>10</td>
</tr>
<tr>
<td>Рубин</td><td>Красный</td><td>9</td>
</tr>
<tr class="odd">
<td>Аметист</td><td>Голубой</td><td>7</td>
</tr>
<tr>
<td>Изумруд</td><td>Зеленый</td><td>8</td>
</tr>
<tr class="odd">
<td>Сапфир</td><td>Голубой</td><td>9</td>
</tr>
</table>
</body>
</html>

```

Результат данного примера показан на рис. 1.17. В примере класс с именем `odd` используется для изменения цвета фона строки таблицы. За счет того, что этот класс добавляется не ко всем тегам `<tr>` и получается чередование

разных цветов.



Название	Цвет	Твердость по Моосу
Алмаз	Белый	10
Рубин	Красный	9
Аметист	Голубой	7
Изумруд	Зеленый	8
Сапфир	Голубой	9

Рис. 1.17. Результат применения классов

Одновременное использование разных классов

К любому тегу можно одновременно добавить несколько классов, перечисляя их в значении атрибута `class` через пробел, такая форма называется мультиклассы. В этом случае к элементу применяется стиль, описанный в правилах для каждого класса. Поскольку при добавлении нескольких классов они могут содержать одинаковые стилевые свойства, но с разными значениями, то берется значение у класса, который описан в коде ниже. В примере 1.31 показано использование разных классов для создания облака тегов.

Пример 1.31. Сочетание разных классов

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Облако тегов</title>
<style type="text/css">
.level1 { font-size: 1em; }
.level2 { font-size: 1.2em; }
.level3 { font-size: 1.4em; }
.level4 { font-size: 1.6em; }
.level5 { font-size: 1.8em; }
.level6 { font-size: 2em; }
A.tag {
color: #468be1; /* Цвет ссылок */
}
</style>
</head>
<body>
<div>
<a href="/term/2" class="tag level6">Paint.NET</a>
<a href="/term/69" class="tag level6">Photoshop</a>
<a href="/term/3" class="tag level5">цвет</a>
<a href="/term/95" class="tag level5">фон</a>
<a href="/term/11" class="tag level4">палитра</a>
<a href="/term/43" class="tag level3">слои</a>
<a href="/term/97" class="tag level2">свет</a>
<a href="/term/44" class="tag level2">панели</a>
<a href="/term/16" class="tag level1">линия</a>
<a href="/term/33" class="tag level1">прямоугольник</a>
<a href="/term/14" class="tag level1">пиксел</a>
<a href="/term/27" class="tag level1">градиент</a>
</div>
</body>
</html>
```

Результат данного примера показан на рис. 1.18.

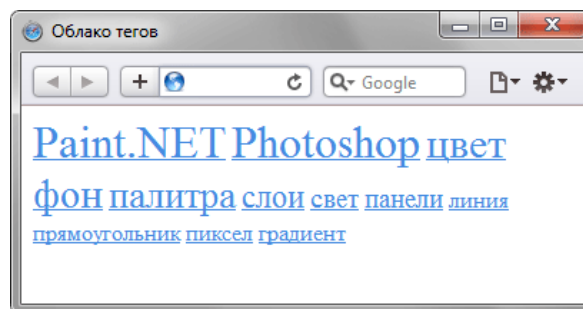


Рис. 1.18. Облако тегов

В стилях также допускается использовать запись вида `.layer1.layer2`, где `layer1` и `layer2` представляют собой имена классов. Стиль применяется только для элементов, у которых одновременно заданы классы `layer1` и `layer2`

(пример 1.32).

Пример 1.32. Использование мультиклассов

XHTML 1.0	CSS 2.1	IE 6	IE 7	IE 8	IE 9	Cr 8	Op 11	Sa 5	Fx 3.6
-----------	---------	------	------	------	------	------	-------	------	--------

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Мультиклассы</title>
<style type="text/css">
.layer1 { color: red; }
.layer2 { color: blue; }
.layer1.layer2 {color: green; }
</style>
</head>
<body>
<p class="layer1">Текст красного цвета</p>
<p class="layer2">Текст синего цвета</p>
<p class="layer1 layer2">Текст зелёного цвета</p>
</body>
</html>
```

Браузер IE6 некорректно работает с мультиклассами и понимает запись `.a.b` как `.b`, т.е. воспринимает только имя последнего класса, что приводит к ошибкам в работе данного примера. Текст с классом `layer2` будет отображаться зеленым цветом, а не синим, как должно быть.

Идентификаторы

Идентификатор (называемый также «ID селектор») определяет уникальное имя элемента, которое используется для изменения его стиля и обращения к нему через скрипты.

Синтаксис применения идентификатора следующий.

```
#Имя идентификатора { свойство1: значение; свойство2: значение; ... }
```

При описании идентификатора вначале указывается символ решетки (#), затем идет имя идентификатора. Оно должно начинаться с латинского символа и может содержать в себе символ дефиса (-) и подчеркивания (_). Использование русских букв в именах идентификатора недопустимо. В отличие от классов идентификаторы должны быть уникальны, иными словами, встречаться в коде документа только один раз.

Обращение к идентификатору происходит аналогично классам, но в качестве атрибута тега используется **id**, значением которого выступает имя идентификатора (пример 1.33). Символ решетки при этом уже не указывается.

Пример 1.33. Использование идентификатора

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Идентификаторы</title>
<style type="text/css">
#help {
position: absolute; /* Абсолютное позиционирование */
right: 20px; /* Положение от правого края */
top: 50px; /* Положение от верхнего края */
width: 225px; /* Ширина блока */
padding: 5px; /* Поля вокруг текста */
background: #f0f0f0; /* Цвет фона */
display: none; /* Скрыть */
}
</style>
</head>
<body>
<p><a href="#" onclick="document.getElementById('help').style.display='block'">Справка</a></p>
<div id="help">
Эта справка помогает в случае, когда вы находитесь в осознании того
факта, что совершенно не понимаете, кто и как вам может помочь. Именно
в этот момент мы и подсказываем, что помочь вам никто не сможет.
</div>
</body>
</html>
```

В данном примере определяется стиль тега **<div>** через идентификатор с именем **help**. Исходно этот слой скрывается от просмотра и делается видимым при нажатии на ссылку, вызывающую небольшой скрипт через событие **onclick**.

Как и при использовании классов, идентификаторы можно применять к конкретному тегу. Синтаксис при этом будет следующий.

```
Тег#Имя идентификатора { свойство1: значение; свойство2: значение; ... }
```

Вначале указывается имя тега, затем без пробелов символ решетки и название идентификатора. В примере 1.34 показано использование идентификатора применительно к тегу **<p>**.

Пример 1.34. Применение идентификатора совместно с тег

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Идентификаторы</title>
<style type="text/css">
P {
color: green; /* Зеленый цвет текста */
font-style: italic; /* Курсивное начертание текста */
}
P#opa {
color: red; /* Красный цвет текста */
border: 1px solid #666; /* Параметры рамки */
background: #eee; /* Цвет фона */
padding: 5px; /* Поля вокруг текста */
}
</style>
<script type="text/javascript">
function newText() {
document.getElementById("opa").innerHTML = 'Необычный текст';
}
</script>
```

```
</head>
<body onload="newText()">
  <p>Обычный текст</p>
  <p id="opa"></p>
</body>
</html>
```

Результат данного примера показан на рис. 1.19.

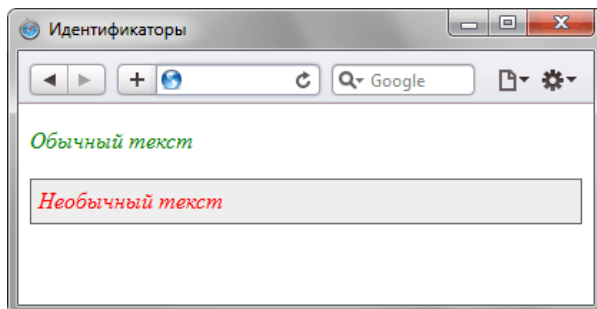


Рис. 1.19. Вид веб-страницы

В данном примере вводится идентификатор с именем opa, который применяется к тегу `<p>`. В функции идентификатора входит изменение стиля абзаца и вывод в него текста через скрипт. Конкретно эта задача возложена на метод `innerHTML`.

Контекстные селекторы

При создании веб-страницы часто приходится вкладывать одни теги внутри других. Чтобы стили для этих тегов использовались корректно, помогут селекторы, которые работают только в определенном контексте. Например, задать стиль для тега `` только когда он располагается внутри контейнера `<p>`. Таким образом можно одновременно установить стиль для отдельного тега, а также для тега, который находится внутри другого.

Контекстный селектор состоит из простых селекторов разделенных пробелом. Так, для селектора тега синтаксис будет следующий.

```
Ter1 Ter2 { ... }
```

В этом случае стиль будет применяться к Тегу2 когда он размещается внутри Тега1, как показано ниже.

```
<Ter1>
  <Ter2> ... </Ter2>
</Ter1>
```

Использование контекстных селекторов продемонстрировано в примере 1.35.

Пример 1.35. Контекстные селекторы

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Контекстные селекторы</title>
<style type="text/css">
  P B {
    font-family: Times, serif; /* Семейство шрифта */
    font-weight: bold; /* Жирное начертание */
    color: navy; /* Синий цвет текста */
  }
</style>
</head>
<body>
<div><b>Жирное начертание текста</b></div>
<p><b>Одновременно жирное начертание текста
и выделенное цветом</b></p>
</body>
</html>
```

В данном примере показано обычное применение тега `` и этого же тега, когда он вложен внутрь абзаца `<p>`. При этом меняется цвет и шрифт текста, как показано на рис. 1.19.

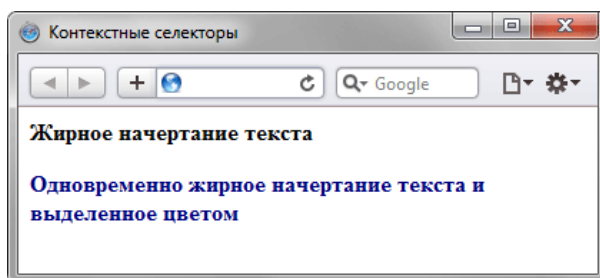


Рис. 1.19. Оформление текста в зависимости от вложенности тегов



Не обязательно контекстные селекторы содержат только один вложенный тег. В зависимости от ситуации допустимо применять два и более последовательно вложенных друг в друга тегов.

Более широкие возможности контекстные селекторы дают при использовании идентификаторов и классов. Это позволяет устанавливать стиль только для того элемента, который располагается внутри определенного класса, как показано в примере 10.2.

Пример 1.36. Использование классов

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Контекстные селекторы</title>
<style type="text/css">
  A {
    color: green; /* Зеленый цвет текста для всех ссылок */
  }
  .menu {
```

```
padding: 7px; /* Поля вокруг текста */
border: 1px solid #333; /* Параметры рамки */
background: #fc0; /* Цвет фона */
}
.menu A {
color: navy; /* Темно-синий цвет ссылок */
}
</style>
</head>
<body>
<div class="menu">
<a href="1.html">Русская кухня</a> |
<a href="2.html">Украинская кухня</a> |
<a href="3.html">Кавказская кухня</a>
</div>
<p><a href="text.html">Другие материалы по теме</a></p>
</body>
</html>
```

Результат данного примера показан на рис. 1.20.

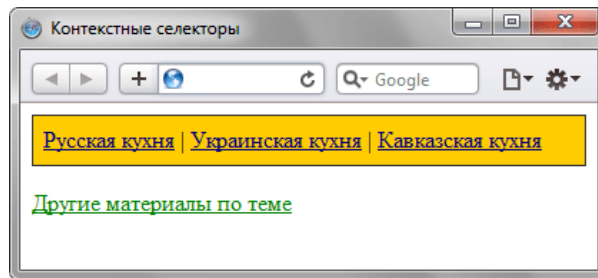


Рис. 1.20. Ссылки разных цветов

В данном примере используется два типа ссылок. Первая ссылка, стиль которой задается с помощью селектора `A`, будет действовать на всей странице, а стиль второй ссылки (`.menu A`) применяется только к ссылкам внутри элемента с классом `menu`.

При таком подходе легко управлять стилем одинаковых элементов, вроде изображений и ссылок, оформление которых должно различаться в разных областях веб-страницы.

Соседние селекторы

Соседними называются элементы веб-страницы, когда они следуют непосредственно друг за другом в коде документа. Рассмотрим несколько примеров отношения элементов.

```
<p>Lorem ipsum <b>dolor</b> sit amet.</p>
```

В этом примере тег `` является дочерним по отношению к тегу `<p>`, поскольку он находится внутри этого контейнера. Соответственно `<p>` выступает в качестве родителя ``.

```
<p>Lorem ipsum <b>dolor</b> <var>sit</var> amet.</p>
```

Здесь теги `<var>` и `` никак не перекрываются и представляют собой соседние элементы. То, что они расположены внутри контейнера `<p>`, никак не влияет на их отношение.

```
<p>Lorem <b>ipsum </b> dolor sit amet, <i>consectetuer</i> adipiscing <tt>elit</tt>.</p>
```

Соседними здесь являются теги `` и `<i>`, а также `<i>` и `<tt>`. При этом `` и `<tt>` к соседним элементам не относятся из-за того, что между ними расположен контейнер `<i>`.

Для управления стилем соседних элементов используется символ плюса (+), который устанавливается между двумя селекторами. Общий синтаксис следующий.

```
Селектор 1 + Селектор 2 { Описание правил стиля }
```

Пробелы вокруг плюса не обязательны, стиль при такой записи применяется к Селектору 2, но только в том случае, если он является соседним для Селектора 1 и следует сразу после него.

В примере 1.37 показана структура взаимодействия тегов между собой.

Пример 1.37. Использование соседних селекторов

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Соседние селекторы</title>
<style type="text/css">
  B + I {
    color: red; /* Красный цвет текста */
  }
</style>
</head>
<body>
<p>Lorem <b>ipsum </b> dolor sit amet, <i>consectetuer</i> adipiscing elit.</p>
<p>Lorem ipsum dolor sit amet, <i>consectetuer</i> adipiscing elit.</p>
</body>
</html>
```

Результат примера показан на рис. 1.22.

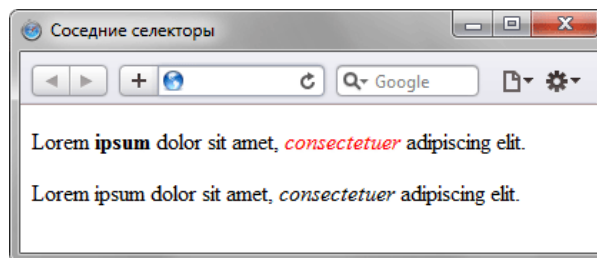


Рис. 1.22. Выделение текста цветом при помощи соседних селекторов

В данном примере происходит изменение цвета текста для содержимого контейнера `<i>`, когда он располагается сразу после контейнера ``. В первом абзаце такая ситуация реализована, поэтому слово «consectetuer» в браузере отображается красным цветом. Во втором абзаце, хотя и присутствует тег `<i>`, но по соседству никакого тега `` нет, так что стиль к этому контейнеру не применяется.

Разберем более практичный пример. Часто возникает необходимость в текст статьи включать различные сноски и примечания. Обычно для этой цели создают новый стилиевой класс и применяют его к абзацу, таким способом можно легко изменить вид текста. Но мы пойдем другим путем и воспользуемся соседними селекторами. Для выделения замечаний создадим новый класс, назовем его `sic`, и станем применять его к тегу `<h2>`. Первый абзац после такого

заголовка выделяется цветом фона и отступом (пример 1.38). Вид остальных абзацев останется неизменным.

Пример 1.38. Изменение стиля абзаца

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Изменение стиля абзаца</title>
<style type="text/css">
  H2.sic {
    font-size: 140%; /* Размер шрифта */
    color: maroon; /* Цвет текста */
    font-weight: normal; /* Нормальное начертание текста */
    margin-left: 30px; /* Отступ слева */
    margin-bottom: 0px; /* Отступ снизу */
  }
  H2.sic + P {
    background: #ddd; /* Цвет фона */
    margin-left: 30px; /* Отступ слева */
    margin-top: 0.5em; /* Отступ сверху */
    padding: 7px; /* Поля вокруг текста */
  }
</style>
</head>
<body>
<h1>Методы ловли льва в пустыне</h1>
<h2>Метод последовательного перебора</h2>
<p>Пусть лев имеет габаритные размеры L x W x H, где L - длина льва от
кончика носа до кисточки хвоста, W - ширина льва, а H - его высота. После чего
пустыню разбиваем на ряд элементарных прямоугольников, размер которых совпадает
с шириной и длиной льва. Учтывая, что лев может находиться не строго на заданном
участке, а одновременно на двух из них, клетку для ловли следует делать повышенной
площади, а именно 2L x 2W. Благодаря этому мы избежим ошибки, когда в клетке
окажется пойманным лишь половина льва или, что хуже, только его хвост.</p>
<h2 class="sic">Важное замечание</h2>
<p>Для упрощения расчетов хвост в качестве погрешности измерения можно
отбросить и не принимать во внимание.</p>
<p>Далее последовательно накрываем каждый из размеченных прямоугольников
пустыни клеткой и проверяем, пойман лев или нет. Как только лев окажется в клетке,
процедура поимки считается завершённой.</p>
</body>
</html>
```

Результат данного примера показан на рис. 1.23.

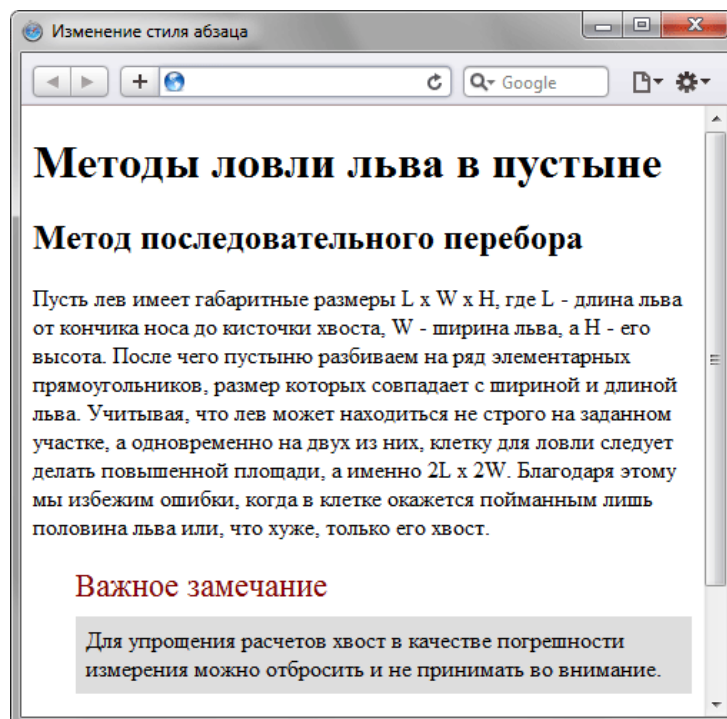


Рис. 11.2. Изменение вида абзаца за счет использования соседних селекторов

В данном примере текст отформатирован с применением абзацев (тег `<p>`), но запись `H2.sic + P` устанавливает стиль только для первого абзаца идущего после тега `<h2>`, у которого добавлен класс `sic`.

Соседние селекторы удобно использовать для тех тегов, к которым автоматически добавляются отступы, чтобы самостоятельно регулировать величину отбивки. Например, если подряд идут теги `<h1>` и `<h2>`, то расстояние между ними легко регулировать как раз с помощью соседних селекторов. Аналогично дело обстоит и для идущих подряд тегов `<h2>` и `<p>`, а также в других подобных случаях. В примере 1.39 таким манером изменяется величина

отступов между указанными тегами.

Пример 1.39. Отступы между заголовками и текстом

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Соседние селекторы</title>
<style type="text/css">
  H1 + H2 {
    margin-top: -10px; /* Смещаем заголовок 2 вверх */
  }
  H2 + P {
    margin-top: -1em; /* Смещаем первый абзац вверх к заголовку */
  }
</style>
</head>
<body>
<h1>Заголовок 1</h1>
<h2>Заголовок 2</h2>
<p>Абзац!</p>
</body>
</html>
```

Поскольку при использовании соседних селекторов стиль применяется только ко второму элементу, то размер отступов уменьшается за счет включения отрицательного значения у свойства **margin-top**. При этом текст поднимается вверх, ближе к предыдущему элементу.

Дочерние селекторы

Дочерним называется элемент, который непосредственно располагается внутри родительского элемента. Чтобы лучше понять отношения между элементами документа, разберем небольшой код (пример 1.40).

Пример 1.40. Вложенность элементов в документе

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Lorem ipsum</title>
</head>
<body>
<div class="main">
<p><em>Lorem ipsum dolor sit amet</em>, consectetur adipiscing
elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam
erat volutpat.</p>
<p><strong><em>Ut wisis enim ad minim veniam</em></strong>,
quis nostrud exerci tution ullamcorper suscipit lobortis nisl ut aliquip ex
ea commodo consequat.</p>
</div>
</body>
</html>
```

В данном примере применяется несколько контейнеров, которые в коде располагаются один в другом. Нагляднее это видно на дереве элементов, так называется структура отношений тегов документа между собой (рис. 1.24).

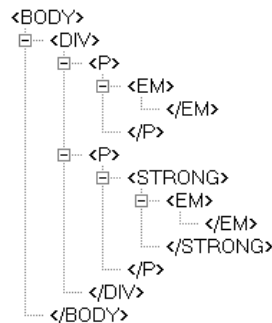


Рис. 1.24. Дерево элементов для примера

На данном рисунке в удобном виде представлена вложенность элементов и их иерархия. Здесь дочерним элементом по отношению к тегу `<div>` выступает тег `<p>`. Вместе с тем тег `` не является дочерним для тега `<div>`, поскольку он расположен в контейнере `<p>`.

Вернемся теперь к селекторам. Дочерним селектором считается такой, который в дереве элементов находится прямо внутри родительского элемента. Синтаксис применения таких селекторов следующий.

```
Селектор 1 > Селектор 2 { Описание правил стиля }
```

Стиль применяется к Селектору 2, но только в том случае, если он является дочерним для Селектора 1.

Если снова обратиться к примеру 1.40, то стиль вида `P > EM { color: red }` будет установлен для первого абзаца документа, поскольку тег `` находится внутри контейнера `<p>`, и не даст никакого результата для второго абзаца. А все из-за того, что тег `` во втором абзаце расположен в контейнере ``, поэтому нарушается условие вложенности.

По своей логике дочерние селекторы похожи на селекторы контекстные. Разница между ними следующая. Стиль к дочернему селектору применяется только в том случае, когда он является прямым потомком, иными словами, непосредственно располагается внутри родительского элемента. Для контекстного селектора же допустим любой уровень вложенности. Чтобы стало понятно, о чем идет речь, разберем следующий код (пример 1.41).

Пример 1.41. Контекстные и дочерние селекторы

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Дочерние селекторы</title>
<style type="text/css">
DIV I { /* Контекстный селектор */
  color: green; /* Зеленый цвет текста */
}
P > I { /* Дочерний селектор */
  color: red; /* Красный цвет текста */
}
```

```

}
</style>
</head>
<body>
<div>
<p>Известно, что пустыня обладает неправильной формой, которую
принимаем прямоугольной. Это достигается одним из двух способов —
<i>включением частей</i>, выходящих за пределы области пустыни
или их <i>отбрасыванием</i>.
</div>
</body>
</html>

```

Результат данного примера показан на рис. 1.25.

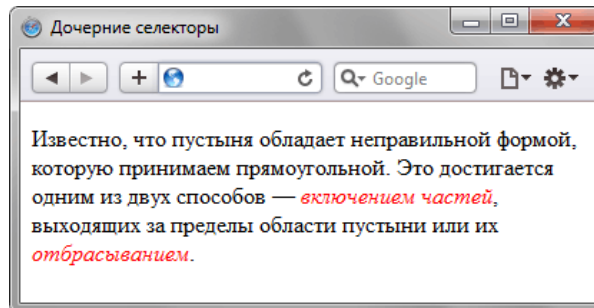


Рис. 1.25. Цвет текста, заданный с помощью дочернего селектора

На тег `<i>` в примере действуют одновременно два правила: контекстный селектор (тег `<i>` расположен внутри `<div>`) и дочерний селектор (тег `<i>` является дочерним по отношению к `<p>`). При этом правила являются равносильными, поскольку все условия для них выполняются и не противоречат друг другу. В подобных случаях применяется стиль, который расположен в коде ниже, поэтому курсивный текст отображается красным цветом. Стоит поменять правила местами и поставить `DIV I` ниже, как цвет текста изменится с красного на зеленый.

Заметим, что в большинстве случаев от добавления дочерних селекторов можно отказаться, заменив их контекстными селекторами. Однако использование дочерних селекторов расширяет возможности по управлению стилями элементов, что в итоге позволяет получить нужный результат, а также простой и наглядный код.

Удобнее всего применять указанные селекторы для элементов, которые обладают иерархической структурой — сюда относятся, например, таблицы и разные списки. В примере 1.42 показано изменение вида списка с помощью стилей. За счет вложения одного списка в другой получаем разновидность меню. Заголовки при этом располагаются горизонтально, а набор ссылок — вертикально под заголовками (рис. 1.26).

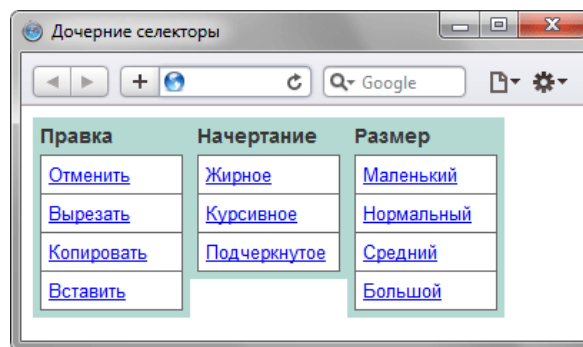


Рис. 1.26. Список в виде меню

Для размещения текста по горизонтали к селектору `LI` добавляется стилевое свойство `float`. Чтобы при этом разделить между собой стиль горизонтального и вертикального списка и применяются дочерние селекторы (пример 1.42).

Пример 1.42. Использование дочерних селекторов

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```

<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Дочерние селекторы</title>
<style type="text/css">
  UL#menu {
    margin: 0; padding: 0; /* Убираем отступы */
  }
  UL#menu > LI {
    list-style: none; /* Убираем маркеры списка */
    width: 100px; /* Ширина элемента в пикселах */

```

```

background: #b3d9d2; /* Цвет фона */
color: #333; /* Цвет текста */
padding: 5px; /* Поля вокруг текста */
font-family: Arial, sans-serif; /* Рубленый шрифт */
font-size: 90%; /* Размер шрифта */
font-weight: bold; /* Жирное начертание */
float: left; /* Располагаем элементы по горизонтали */
}
LI > UL {
list-style: none; /* Убираем маркеры списка */
margin: 0; padding: 0; /* Убираем отступы вокруг элементов списка */
border-bottom: 1px solid #666; /* Граница внизу */
padding-top: 5px; /* Добавляем отступ сверху */
}
LI > A {
display: block; /* Ссылки отображаются в виде блока */
font-weight: normal; /* Нормальное начертание текста */
font-size: 90%; /* Размер шрифта */
background: #fff; /* Цвет фона */
border: 1px solid #666; /* Параметры рамки */
border-bottom: none; /* Убираем границу снизу */
padding: 5px; /* Поля вокруг текста */
}
</style>
</head>
<body>
<ul id="menu">
<li>Правка
<ul>
<li><a href="#">Отменить</a></li>
<li><a href="#">Вырезать</a></li>
<li><a href="#">Копировать</a></li>
<li><a href="#">Вставить</a></li>
</ul>
</li>
<li>Начертание
<ul>
<li><a href="#">Жирное</a></li>
<li><a href="#">Курсивное</a></li>
<li><a href="#">Подчеркнутое</a></li>
</ul>
</li>
<li>Размер
<ul>
<li><a href="#">Маленький</a></li>
<li><a href="#">Нормальный</a></li>
<li><a href="#">Средний</a></li>
<li><a href="#">Большой</a></li>
</ul>
</li>
</ul>
</body>
</html>

```

В данном примере дочерние селекторы требуются, чтобы разделить стиль элементов списка верхнего уровня и вложенные списки, которые выполняют разные задачи, поэтому стиль для них не должен пересекаться.

Верстаю сайты

Занимаюсь (X)HTML/CSS вёрсткой.

Делаю действительно качественную работу.

- любой сложности
- под любые платформы
- под любые браузеры
- валидная начисто!
- уникальная организация (x)html/css кода
- смантически корректный код
- логичность верстки
- табличная, дивная (предпочитаю второе, 49/50 как правило блочные работы)
- индивидуальный подход к каждому клиенту

Цены

От 100\$ за шаблон.

IE6+ доплата, в зависимости от сложности и времени.

Контакты

e-mail: psywalker09@gmail.com

qip: 366905663

skype: walker1071

Отзывы обо мне

<http://forum.htmlbook.ru/index.php?showtopic=20231>

Портфолио и код моих работ демонстрирую непосредственно заказчику

Селекторы атрибутов

Многие теги различаются по своему действию в зависимости от того, какие в них используются атрибуты. Например, тег `<input>` может создавать кнопку, текстовое поле и другие элементы формы всего лишь за счет изменения значения атрибута `type`. При этом добавление правил стиля к селектору `INPUT` применит стиль одновременно ко всем созданным с помощью этого тега элементам. Чтобы гибко управлять стилем подобных элементов, в CSS введены селекторы атрибутов. Они позволяют установить стиль по присутствию определенного атрибута тега или его значения.

Рассмотрим несколько типичных вариантов применения таких селекторов.

Простой селектор атрибута

Устанавливает стиль для элемента, если задан специфичный атрибут тега. Его значение в данном случае не важно. Синтаксис применения такого селектора следующий.

```
[атрибут] { Описание правил стиля }
Селектор[атрибут] { Описание правил стиля }
```

Стиль применяется к тем тегам, внутри которых добавлен указанный атрибут. Пробел между именем селектора и квадратными скобками не допускается.

В примере 1.43 показано изменение стиля тега `<q>`, в том случае, если к нему добавлен атрибут `title`.

Пример 1.43. Вид элемента в зависимости от его атрибута

```
XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Селекторы атрибутов</title>
<style type="text/css">
Q {
font-style: italic; /* Курсивное начертание */
quotes: "«" "»"; /* Меняем вид кавычек в цитате */
}
Q[title] {
color: maroon; /* Цвет текста */
}
</style>
</head>
<body>
<p>Продолжая известный закон Мерфи, который гласит: <q>Если неприятность
может случиться, то она обязательно случится</q>, можем ввести свое наблюдение:
<q title="Из законов Фергюссона-Мержевича">После того, как веб-страница
будет корректно отображаться в одном браузере, выяснится,
что она неправильно показывается в другом</q>.</p>
</body>
</html>
```

Результат примера показан на рис. 1.27. Браузер Safari и Chrome не выводят наши кавычки в тексте, заменяя их на стандартные.

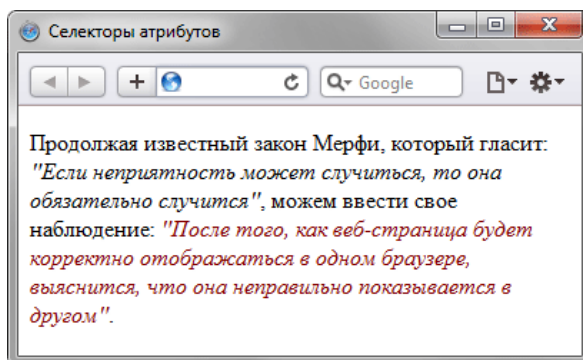


Рис. 1.27. Изменение стиля элемента в зависимости от применения атрибута `title`

В данном примере меняется цвет текста внутри контейнера `<q>`, когда к нему добавляется `title`. Обратите внимание, что для селектора `Q[title]` нет нужды повторять стилевые свойства, поскольку они наследуются от селектора `Q`.

Атрибут со значением

Устанавливает стиль для элемента в том случае, если задано определенное значение специфичного атрибута. Синтаксис применения следующий.

```
[атрибут="значение"] { Описание правил стиля }
Селектор[атрибут="значение"] { Описание правил стиля }
```

В первом случае стиль применяется ко всем тегам, которые содержат указанное значение. А во втором — только к определенным селекторам.

В примере 1.44 показано изменение стиля ссылки в том случае, если тег `<a>` содержит атрибут `target` со значением `_blank`. При этом ссылка будет открываться в новом окне и чтобы показать это, с помощью стилей добавляем небольшой рисунок перед текстом ссылки.

Пример 1.44. Стиль для открытия ссылок в новом окне | XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Селекторы атрибутов</title>
<style type="text/css">
  A[target="_blank"] {
    background: url(images/blank.png) 0 6px no-repeat; /* Параметры фонового рисунка */
    padding-left: 15px; /* Смещаем текст вправо */
  }
</style>
</head>
<body>
<p><a href="1.html">Обычная ссылка</a> |
<a href="link2" target="_blank">Ссылка в новом окне</a></p>
</body>
</html>
```

Результат примера показан ниже (рис. 1.28).

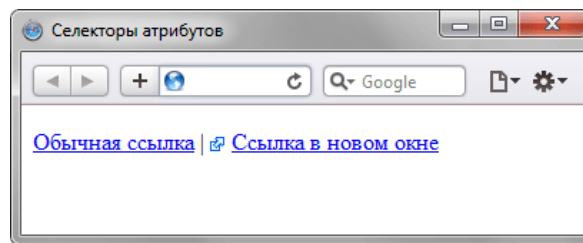


Рис. 1.28. Изменение стиля элемента в зависимости от значения `target`

В данном примере рисунок к ссылке добавляется с помощью свойства `background`. В его функции входит создание повторяющейся фоновой картинки, но повторение фона можно отменить через значение `no-repeat`, что в итоге даст единственное изображение.

Значение атрибута начинается с определенного текста

Устанавливает стиль для элемента в том случае, если значение атрибута тега начинается с указанного текста. Синтаксис применения следующий.

```
[атрибут^="значение"] { Описание правил стиля }
Селектор[атрибут^="значение"] { Описание правил стиля }
```

В первом случае стиль применяется ко всем элементам, у которых значение атрибута начинается с указанного текста. А во втором — только к определенным селекторам. Использование кавычек не обязательно.

Предположим, что на сайте требуется разделить стиль обычных и внешних ссылок — ссылки, которые ведут на другие сайты. Чтобы не добавлять к тегу `<a>` новый класс, воспользуемся селекторами атрибутов. Внешние ссылки характеризуются добавлением к адресу протокола, например, для доступа к гипертекстовым документам используется протокол HTTP. Поэтому внешние ссылки начинаются с ключевого слова `http://`, его и добавляем к селектору `A`, как показано в примере 1.45.

Пример 1.45. Изменение стиля внешней ссылки | XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Селекторы атрибутов</title>
<style type="text/css">
  A[href^="http://"] {
    font-weight: bold /* Жирное начертание */
  }
</style>
</head>
```

```

<body>
<p><a href="1.html">Обычная ссылка</a> |
<a href="http://htmlbook.ru" target="_blank">Внешняя
ссылка на сайт htmlbook.ru</a></p>
</body>
</html>

```

Результат примера показан ниже (рис. 1.29).

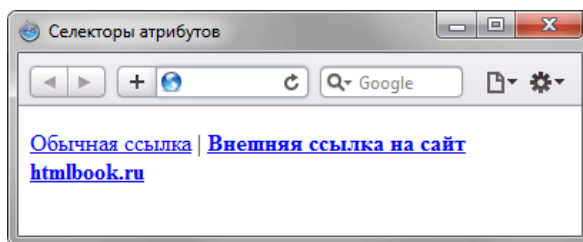


Рис. 1.29. Изменение стиля для внешних ссылок

В данном примере внешние ссылки выделяются жирным начертанием. Также можно воспользоваться показанным в примере 1.44 приемом и добавлять к ссылке небольшое изображение, которое будет сообщать, что ссылка ведет на другой сайт.

Значение атрибута оканчивается определенным текстом

Устанавливает стиль для элемента в том случае, если значение атрибута оканчивается указанным текстом. Синтаксис применения следующий.

```

[атрибут$="значение"] { Описание правил стиля }
Селектор[атрибут$="значение"] { Описание правил стиля }

```

В первом случае стиль применяется ко всем элементам у которых значение атрибута завершается заданным текстом. А во втором — только к определенным селекторам.

Таким способом можно автоматически разделять стиль для сайтов домена ru и для сайтов других доменов вроде com, как показано в примере 1.46.

Пример 1.46. Стиль для разных доменов

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```

<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Селекторы атрибутов</title>
<style type="text/css">
A[href$=".ru"] { /* Если ссылка заканчивается на .ru */
background: url(images/ru.png) no-repeat 0 6px; /* Добавляем фоновый рисунок */
padding-left: 12px; /* Смещаем текст вправо */
}
A[href$=".com"] { /* Если ссылка заканчивается на .com */
background: url(images/com.png) no-repeat 0 6px;
padding-left: 12px;
}
</style>
</head>
<body>

<p><a href="http://www.yandex.com">Yandex.Com</a> |
<a href="http://www.yandex.ru">Yandex.Ru</a></p>

</body>
</html>

```

В данном примере содержатся две ссылки, ведущие на разные домены — com и ru. При этом к каждой такой ссылке с помощью стилей добавляется своя фоновая картинка (рис. 1.30). Стилиевые свойства будут применяться только для тех ссылок, атрибут href которых оканчивается на «.ru» или «.com». Заметьте, что добавив к имени домена слэш (http://www.yandex.ru/) или адрес страницы (http://www.yandex.ru/fun.html), мы изменим тем самым окончание и стиль применяться уже не будет. В этом случае лучше воспользоваться командой ***=**.

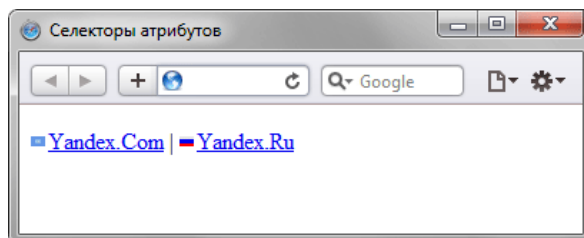


Рис. 1.30. Добавление картинки к ссылкам

Значение атрибута содержит указанный текст

Возможны варианты, когда стиль следует применить к тегу с определенным атрибутом, при этом частью его значения является некоторый текст. При этом точно не известно, в каком месте значения включен данный текст — в начале, середине или конце. В подобном случае следует использовать такой синтаксис.

```
[атрибут*="значение"] { Описание правил стиля }
Селектор[атрибут*="значение"] { Описание правил стиля }
```

В примере 1.47 показано изменение стиля ссылок, в атрибуте `href` которых встречается слово «htmlbook».

Пример 1.47. Стиль для разных сайтов

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Селекторы атрибутов</title>
<style type="text/css">
[ href*="htmlbook" ] {
background: yellow; /* Желтый цвет фона */
}
</style>
</head>
<body>
<p><a href="http://www.htmlbook.ru/html/">Теги HTML</a> |
<a href="http://stepbystep.htmlbook.ru">Шаг за шагом</a> |
<a href="http://webimg.ru">Графика для Веб</a></p>
</body>
</html>
```

Результат данного примера показан на рис. 1.31.

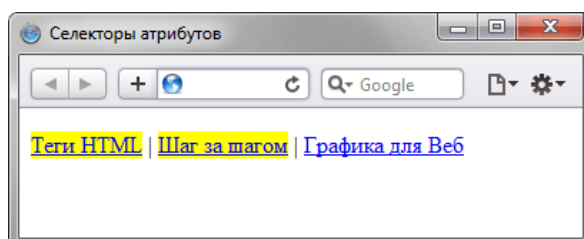


Рис. 1.31. Изменение стиля для ссылок, в адресе которых встречается «htmlbook»

Одно из нескольких значений атрибута

Некоторые значения атрибутов могут перечисляться через пробел, например имена классов. Чтобы задать стиль при наличии в списке требуемого значения применяется следующий синтаксис.

```
[атрибут~="значение"] { Описание правил стиля }
Селектор[атрибут~="значение"] { Описание правил стиля }
```

Стиль применяется в том случае, если у атрибута имеется указанное значение или оно входит в список значений, разделяемых пробелом (пример 1.48).

Пример 1.48. Стиль в зависимости от имени класса

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Блок</title>
<style type="text/css">
[class~="block"] h3 { color: green; }
</style>
</head>
<body>
<div class="block tag">
```

```
<h3>Заголовок</h3>
</div>
</body>
</html>
```

В данном примере зеленый цвет текста применяется к селектору `h3`, если имя класса у слоя задано как `block`. Отметим, что аналогичный результат можно получить, если использовать конструкцию `*=` вместо `~=`.

Дефис в значении атрибута

В именах идентификаторов и классов разрешено использовать символ дефиса (-), что позволяет создавать значащие значения атрибутов `id` и `class`. Для изменения стиля элементов, в значении которых применяется дефис, следует воспользоваться следующим синтаксисом.

```
[атрибут|=значение] { Описание правил стиля }
Селектор[атрибут|=значение] { Описание правил стиля }
```

Стиль применяется к элементам, у которых атрибут начинается с указанного значения или с фрагмента значения, после которого идет дефис (пример 1.49).

Пример 1.49. Дефисы в значениях

XHTML 1.0 | CSS 2.1 | **IE 6** | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Блок</title>
<style type="text/css">
DIV[class|=block] {
background: #306589; /* Цвет фона */
color: #acdb4c; /* Цвет текста */
padding: 5px; /* Поля */
}
DIV[class|=block] A {
color: #fff; /* Цвет ссылок */
}
</style>
</head>
<body>
<div class="block-menu-therm">
<h2>Термины</h2>
<div class="content">
<ul class="menu">
<li><a href="t1.html">Буквица</a></li>
<li><a href="t2.html">Выворотка</a></li>
<li><a href="t3.html">Выключка</a></li>
<li><a href="t4.html">Интерлиньяж</a></li>
<li><a href="t5.html">Капитель</a></li>
<li><a href="t6.html">Начертание</a></li>
<li><a href="t7.html">Отбивка</a></li>
</ul>
</div>
</div>
</body>
</html>
```

В данном примере имя класса задано как `block-menu-therm`, поэтому в стилях используется конструкция `|="block"`, поскольку значение начинается именно с этого слова и в значении встречаются дефисы.

Все перечисленные методы можно комбинировать между собой, определяя стиль для элементов, которые содержат два и более атрибута. В подобных случаях квадратные скобки идут подряд.

```
[атрибут1="значение1"][атрибут2="значение2"] { Описание правил стиля }
Селектор[атрибут1="значение1"][атрибут2="значение2"] { Описание правил стиля }
```

Универсальный селектор

Иногда требуется установить одновременно один стиль для всех элементов веб-страницы, например, задать шрифт или начертание текста. В этом случае поможет универсальный селектор, который соответствует любому элементу веб-страницы.

Для обозначения универсального селектора применяется символ звездочки (*) и в общем случае синтаксис будет следующий.

```
* { Описание правил стиля }
```

В некоторых случаях указывать универсальный селектор не обязательно. Так, например, записи `*.class` и `.class` являются идентичными по своему результату.

В примере 1.50 показано одно из возможных приложений универсального селектора — выбор шрифта и размера текста для всех элементов документа.

Пример 1.50. Использование универсального селектора

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Универсальный селектор</title>
<style type="text/css">
* {
font-family: Arial, Verdana, sans-serif; /* Рубленый шрифт для текста */
font-size: 96%; /* Размер текста */
}
</style>
</head>
<body>
<p>Средний самец льва имеет длину около трех метров и весит
от 180 до 230 килограмм.</p>
</body>
</html>
```

Заметим, что аналогичный результат можно получить, если в данном примере поменять селектор * на **BODY**.

Универсальный селектор часто применяется для обнуления отступов и полей у всех элементов, как показано в примере 1.51.

Пример 1.51. Стиль для обнуления отступов и полей

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

Универсальный селектор часто применяется для обнуления отступов и полей у всех элементов, как показано в примере 1.51.

Несмотря на внешние удобства универсального селектора и его широкое распространение, ни в коем случае не идите на поводу «моды» и не используйте селектор, как показано в примере выше. Доводы следующие.

- Универсальный селектор применяет стиль ко всем элементам веб-страницы, включая невидимые, что приводит к замедлению браузера, поскольку ему требуется некоторое время для построения дерева элементов и добавления к ним стилей. Чем больше элементов в коде, тем сильнее выражено замедление. В некоторых крайних случаях вообще может появиться «зависание» браузера на несколько секунд.
- При неверном использовании универсального селектора результат может оказаться непредсказуемым. Пример ниже является полностью корректным с точки зрения CSS, но приводит страницу к парадоксальному виду.

```
* {
display: block;
border: 1px solid #c00;
}
```

- «Обнуление стилей» (см. пример 1.51) прививает у разработчика дурную манеру верстки. Вместо того чтобы знать, какие значения свойств установлены по умолчанию, разработчик переключает эту работу на браузер, насильно устанавливая все значения в ноль. В результате некоторые значения применяются к свойствам, для которых не могут устанавливаться или применяются к свойствам, у которых данное значение и так нулевое. Это опять же приводит к повышению нагрузки на браузер и замедлению его работы.
- Применение одного стиля сразу ко всем элементам иногда приводит к ошибкам отображения элементов в

отдельных браузеров. В примере ниже к ссылкам добавляется пунктирное подчеркивание, которое не показывается в IE7 из-за заданного обнуления полей у ссылок.

```
* {
  padding: 0;
}
a {
  text-decoration: none;
  border-bottom: 1px dashed red;
}
```

Несмотря на указанные особенности, универсальный селектор можно и нужно включать в стиль, но в комбинации с другими селекторами. Так, если требуется изменить стиль всех элементов в форме, то следует использовать контекстный селектор `FORM *` или селектор, показанный в примере 1.52.

Пример 1.52. Использование универсального селектора

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Универсальный селектор</title>
<style type="text/css">
FORM P * {
  border: 1px solid #333; /* Параметры рамки */
  background: #ffe; /* Цвет фона */
}
</style>
</head>
<body>
<form action="handler.php">
<p><input type="text" /></p>
<p><input type="submit" /></p>
</form>
</body>
</html>
```

В данном примере рамка и фон добавляется ко всем элементам формы, расположенных внутри абзаца (тег `<p>`).

Комплексные решения по созданию эффективных IT-инфраструктур в полном соответствии с бизнес задачами вашей компании



СЕРВЕРНЫЕ И СЕТЕВЫЕ РЕШЕНИЯ

Поставки серверного оборудования и программного обеспечения, а также создание комплексных решений на базе серверных технологий для построения информационных систем различного масштаба: от серверных решений начального уровня, до корпоративных систем крупных предприятий.

РЕШЕНИЯ ДЛЯ ПЕЧАТИ

Оптимизация и упрощение сетей устройств печати и копирования за счет выравнивания производительности работы конечных пользователей, улучшения технической поддержки и экономии затрат. Мы улучшим Вашу среду печати сэкономив время, деньги и технические ресурсы

КОМПЬЮТЕРНЫЕ РЕШЕНИЯ

Реализация компьютерных систем, а также разработка и производство персональных компьютеров под собственной торговой маркой "RestR". Мы можем предложить компьютеры любого уровня - от офисных и недорогих, до самых современных игровых и профессиональных рабочих станций.

РЕШЕНИЯ В ОБЛАСТИ ЭЛЕКТРОПИТАНИЯ

Конфигурирование и внедрение максимально эффективных комплексных инфраструктурных проектов на базе интегрированных решений APC, обеспечивающих защиту от перегрева и проблем с электропитанием, обеспечивая постоянную готовность оборудования.

Мы постоянно следим за развитием IT-отрасли. Установленные отношения с мировыми производителями - служат крепкой основой для предложения нашим партнерам выгодных и привлекательных решений, которые отличаются высоким качеством и безотказной работой.



Подробнее на нашем сайте WWW.RESTR.COM

ООО Рестрком 125480, г. Москва, ул. Героев Панфиловцев, д.10, кор.1. м.Планерная

Телефон: (495) 649-62-03 (многоканальный)

E-mail: info@restr.com

Псевдоклассы

Псевдоклассы определяют динамическое состояние элементов, которое изменяется со временем или с помощью действий пользователя, а также положение в дереве документа. Примером такого состояния служит текстовая ссылка, которая меняет свой цвет при наведении на нее курсора мыши. При использовании псевдоклассов браузер не перегружает текущий документ, поэтому с помощью псевдоклассов можно получить разные динамические эффекты на странице.

Синтаксис применения псевдоклассов следующий.

```
Селектор:Псевдокласс { Описание правил стиля }
```

Вначале указывается селектор, к которому добавляется псевдокласс, затем следует двоеточие, после которого идет имя псевдокласса. Допускается применять псевдоклассы к именам идентификаторов или классов (`A.menu:hover {color: green}`), а также к контекстным селекторам (`.menu A:hover {background: #fc0}`). Если псевдокласс указывается без селектора впереди (`:hover`), то он будет применяться ко всем элементам документа.

Условно все псевдоклассы делятся на три группы:

- определяющие состояние элементов;
- имеющие отношение к дереву элементов;
- указывающие язык текста.



В дальнейшем имена псевдоклассов будем писать с двоеточием впереди, чтобы различать между собой псевдоклассы и стилевые свойства.

Псевдоклассы, определяющие состояние элементов

К этой группе относятся псевдоклассы, которые распознают текущее состояние элемента и применяют стиль только для этого состояния.

:active

Происходит при активации пользователем элемента. Например, ссылка становится активной, если навести на нее курсор и щелкнуть мышкой. Несмотря на то, что активным может стать практически любой элемент веб-страницы, псевдокласс `:active` используется преимущественно для ссылок.

:link

Применяется к непосещенным ссылкам, т.е. таким ссылкам, на которые пользователь еще не нажимал. Браузер некоторое время сохраняет историю посещений, поэтому ссылка может быть помечена как посещенная хотя бы потому, что по ней был зафиксирован переход раньше.



Запись `A {...}` и `A:link {...}` по своему результату равноценна, поскольку в браузере дает один и тот же эффект, поэтому псевдокласс `:link` можно не указывать. Исключением являются якоря, на них действие `:link` не распространяется.

:focus

Применяется к элементу при получении им фокуса. Например, для текстового поля формы получение фокуса означает, что курсор установлен в поле, и с помощью клавиатуры можно вводить в него текст (пример 1.53).

Пример 1.53. Применение псевдокласса focus

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Псевдоклассы</title>
<style type="text/css">
  INPUT:focus {
    color: red; /* Красный цвет текста */
  }
</style>
</head>
<body>
<form action="">
  <p><input type="text" value="Черный текст" /></p>
  <p><input type="text" value="Черный текст" /></p>
</form>
</body>
</html>
```

Результат примера показан ниже (рис. 1.32). Во второй строке находится курсор, поэтому текстовое поле получило фокус.

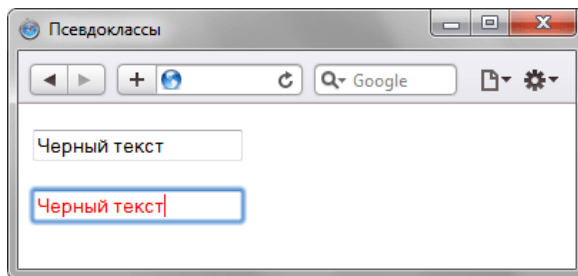



Рис. 1.32. Изменение стиля текста при получении фокуса

В данном примере в текстовом поле содержится предварительный текст, он определяется значением атрибута `value` тега `<input>`. При щелчке по элементу формы происходит получение полем фокуса, и цвет текста меняется на красный. Достаточно щелкнуть в любом месте страницы (кроме текстового поля, естественно), как произойдет потеря фокуса и текст вернется к первоначальному черному цвету.

 Результат будет виден только для тех элементов, которые могут получить фокус. В частности, это теги `<a>`, `<input>`, `<select>` и `<textarea>`.

:hover

Псевдокласс `:hover` активизируется, когда курсор мыши находится в пределах элемента, но щелчка по нему не происходит.

:visited

Данный псевдокласс применяется к посещенным ссылкам. Обычно такая ссылка меняет свой цвет по умолчанию на фиолетовый, но с помощью стилей цвет и другие параметры можно задать самостоятельно (пример 1.54).

Пример 1.54. Изменение цвета ссылок

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Псевдоклассы</title>
<style type="text/css">
A:link {
color: #036; /* Цвет непосещенных ссылок */
}
A:visited {
color: #606; /* Цвет посещенных ссылок */
}
A:hover {
color: #f00; /* Цвет ссылок при наведении на них курсора мыши */
}
A:active {
color: #ff0; /* Цвет активных ссылок */
}
</style>
</head>
<body>
<p><a href="1.html">Ссылка 1</a> |
<a href="2.html">Ссылка 2</a> |
<a href="3.html">Ссылка 3</a></p>
</body>
</html>
```

Результат примера показан на рис. 1.33.

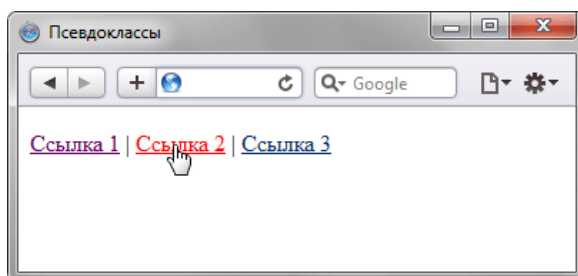


Рис. 1.33. Вид ссылки при наведении на нее курсора мыши

В данном примере показано использование псевдоклассов совместно со ссылками. При этом имеет значение порядок следования псевдоклассов. Вначале указывается `:visited`, а затем идет `:hover`, в противном случае посещенные ссылки не будут изменять свой цвет при наведении на них курсора.

Селекторы могут содержать более одного псевдокласса, которые перечисляются подряд через двоеточие, но только в том случае, если их действия не противоречат друг другу. Так, запись `A:visited:hover` является корректной, а запись `A:link:visited` — нет. Впрочем, если подходить формально, то валидатор CSS считает правильным любое сочетание псевдоклассов.

⚠ Браузер Internet Explorer 6 позволяет использовать псевдоклассы `:active` и `:hover` только для ссылок. Начиная с версии 7.0 псевдоклассы в этом браузере работают и для других элементов.

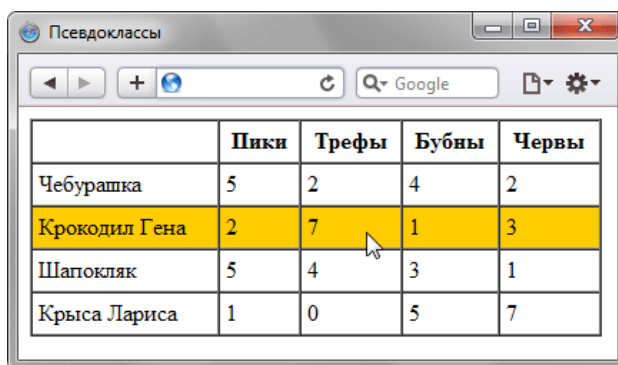
Псевдокласс `:hover` не обязательно должен применяться к ссылкам, его можно добавлять и к другим элементам документа. Так, в примере 1.55 показана таблица, строки которой меняют свой цвет при наведении на них курсора мыши. Это достигается за счет добавления псевдокласса `:hover` к селектору `TR`.

Пример 1.55. Выделение строк таблицы

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Псевдоклассы</title>
<style type="text/css">
  TR:hover {
    background: #fc0; /* Меняем цвет фона строки таблицы */
  }
</style>
</head>
<body>
<table width="400" border="1" cellpadding="4" cellspacing="0">
<tr>
<th>&nbsp;</th>
<th>Пики</th><th>Трефы</th><th>Бубны</th><th>Червы</th>
</tr>
<tr>
<td>Чебурашка</td>
<td>5</td><td>2</td><td>4</td><td>2</td>
</tr>
<tr>
<td>Крокодил Гена</td>
<td>2</td><td>7</td><td>1</td><td>3</td>
</tr>
<tr>
<td>Шапокляк</td>
<td>5</td><td>4</td><td>3</td><td>1</td>
</tr>
<tr>
<td>Крыса Лариса</td>
<td>1</td><td>0</td><td>5</td><td>7</td>
</tr>
</table>
</body>
</html>
```

Результат примера показан ниже (рис. 1.34).



	Пики	Трефы	Бубны	Червы
Чебурашка	5	2	4	2
Крокодил Гена	2	7	1	3
Шапокляк	5	4	3	1
Крыса Лариса	1	0	5	7

Рис. 1.34. Выделение строк таблицы при наведении на них курсора мыши

Псевдоклассы, имеющие отношение к дереву документа

К этой группе относятся псевдоклассы, которые определяют положение элемента в дереве документа и применяют к нему стиль в зависимости от его статуса.

`:first-child`

Применяется к первому дочернему элементу селектора, который расположен в дереве элементов документа. Чтобы

стало понятно, о чем речь, разберем небольшой код (пример 1.56).

Пример 1.56. Использование псевдокласса `first-child`

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Псевдоклассы</title>
<style type="text/css">
  div p:first-child {
    font: 1.2em Arial, sans-serif; /* Шрифт */
    color: #E81E26; /* Цвет текста */
    margin-bottom: -0.5em; /* Отступ снизу */
  }
</style>
</head>
<body>
<div>
<p>Размеры</p>
<p>Желательно, чтобы ширина и длина клетки были кратные значениям А и В.
Таким образом, всю пустыню можно поделить на ряд дискретных областей,
размеры которых совпадают с размерами нашей клетки.</p>
</div>
<div>
<p>Положение льва</p>
<p>Во всех случаях положение льва считается стационарным.
Во время поиска животное остается на своем месте и не перемещается
до его поимки в клетку.</p>
</div>
</body>
</html>
```

Результат примера показан ниже (рис. 1.35).

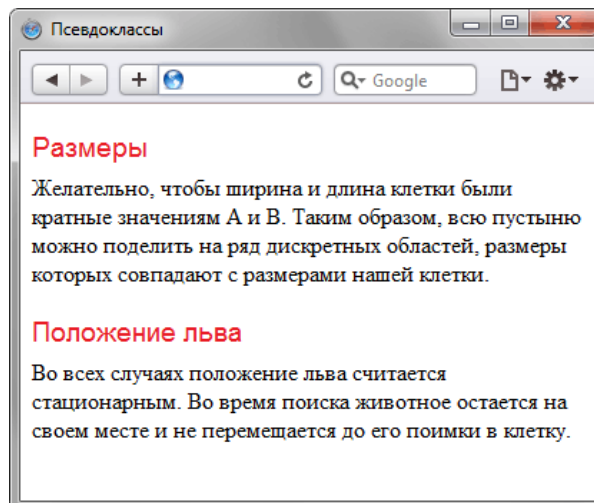


Рис. 1.35. Использование псевдокласса `:first-child`

В данном примере псевдокласс `:first-child` добавляется к селектору `p` и устанавливает для него рубленый шрифт и красный цвет текста. Хотя тег `<p>` внутри `<div>` встречается по два раза, стиль будет применяться только к первым абзацам. В остальных случаях стиль внутри `<p>` останется исходным. Со следующим тегом `<div>` все начинается снова, поскольку родительский элемент поменялся.



Браузер Internet Explorer поддерживает псевдокласс `:first-child` начиная с версии 7.0.

Псевдокласс `:first-child` удобнее всего использовать в тех случаях, когда требуется задать разный стиль для первого и остальных однотипных элементов. Например, по правилам типографики красную строку для первого абзаца текста не устанавливают, а для остальных абзацев добавляют отступ первой строки. С этой целью применяют свойство `text-indent` с нужным значением отступа. Но чтобы изменить стиль первого абзаца и убрать для него отступ потребуется воспользоваться псевдоклассом `:first-child` (пример 1.57).

Пример 1.57. Отступы для абзаца

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Псевдоклассы</title>
<style type="text/css">
  p {
    text-indent: 1em; /* Отступ первой строки */
  }
</style>
</head>
<body>
<p>Желательно, чтобы ширина и длина клетки были кратные значениям А и В.
Таким образом, всю пустыню можно поделить на ряд дискретных областей,
размеры которых совпадают с размерами нашей клетки.</p>
</body>
</html>
```

```

    }
    P:first-child {
      text-indent: 0; /* Для первого абзаца отступ убираем */
    }
  </style>
</head>
<body>
  <p>Историю эту уже начали забывать, хотя находились горожане, которые
  время от времени рассказывали ее вновь прибывшим в город посетителям.</p>
  <p>Легенда обрастала подробностями и уже совсем не напоминала произошедшее
  в действительности событие. И, тем не менее, ни один человек не решался заикнуться
  о ней с наступлением темноты.</p>
  <p>Но однажды в город вновь вошел незнакомец. Он хромал на левую ногу.</p>
</body>
</html>

```

Результат примера показан ниже (рис. 1.36).

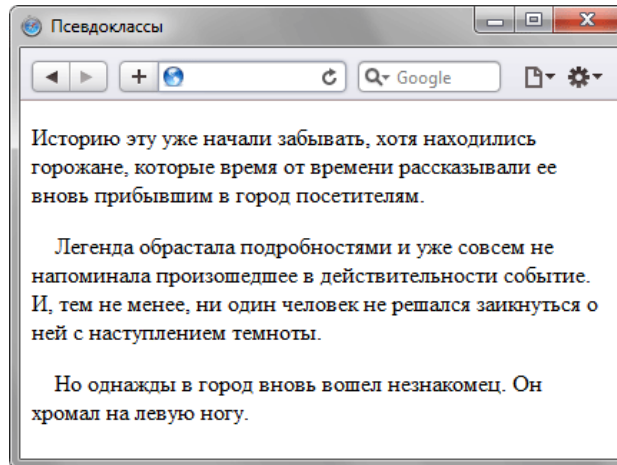


Рис. 1.36. Изменение стиля первого абзаца

В данном примере первый абзац текста не содержит отступа первой строки, а для остальных он установлен.

Псевдоклассы, задающие язык текста

Для документов, одновременно содержащие тексты на нескольких языках имеет значение соблюдение правил синтаксиса, характерные для того или иного языка. С помощью псевдоклассов можно изменять стиль оформления иностранных текстов, а также некоторые настройки.

:lang

Определяет язык, который используется в документе или его фрагменте. В коде HTML язык устанавливается через параметр `charset` тега `<meta>`. С помощью псевдокласса `:lang` можно задавать определенные настройки, характерные для разных языков, например, вид кавычек в цитатах. Синтаксис следующий.

```
Элемент: lang (язык) { ... }
```

В качестве языка могут выступать следующие значения: `ru` — русский; `en` — английский; `de` — немецкий; `fr` — французский; `it` — итальянский.

Пример 1.58. Вид кавычек в зависимости от языка

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```

<!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">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Псевдоклассы</title>
  <style type="text/css">
    P {
      font-size: 1.50em; /* Размер текста */
    }
    q:lang(de) {
      quotes: "\201E" "\201C"; /* Вид кавычек для немецкого языка */
    }
    q:lang(en) {
      quotes: "\201C" "\201D"; /* Вид кавычек для английского языка */
    }
    q:lang(fr), q:lang(ru) { /* Вид кавычек для русского и французского языка */
      quotes: "\00AB" "\00BB";
    }
  </style>
</head>
<body>
  <p>Цитата на французском языке: <q lang="fr">Ce que femme veut, Dieu le veut</q>.</p>
  <p>Цитата на немецком: <q lang="de">Der Mensch, versuche die Gotter nicht</q>.</p>
  <p>Цитата на английском: <q lang="en">To be or not to be</q>.</p>

```

```
</body>  
</html>
```

Результат данного примера показан на рис. 1.37. Для отображения типовых кавычек в примере используется стилевое свойство `quotes`, а само переключение языка и соответствующего вида кавычек происходит через атрибут `lang`, добавляемый к тегу `<q>`.

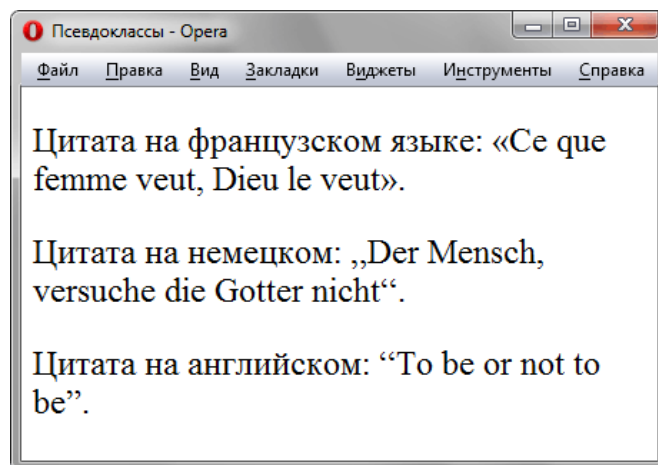


Рис. 1.37. Разные кавычки для разных языков



Дизайн інтер'єров
3D-визуалізація



arch.design@jungle.in.ua

Псевдоэлементы

Псевдоэлементы позволяют задать стиль элементов не определенных в дереве элементов документа, а также генерировать содержимое, которого нет в исходном коде текста.

Синтаксис использования псевдоэлементов следующий.

```
Селектор:Псевдоэлемент { Описание правил стиля }
```

Вначале следует имя селектора, затем пишется двоеточие, после которого идет имя псевдоэлемента. Каждый псевдоэлемент может применяться только к одному селектору, если требуется установить сразу несколько псевдоэлементов для одного селектора, правила стиля должны добавляться к ним по отдельности, как показано ниже.

```
.foo:first-letter { color: red }  
.foo:first-line { font-style: italic }
```

В CSS3 чтобы различать псевдоклассы и псевдоэлементы, перед именем псевдоэлемента ставится два двоеточия (::after). Internet Explorer игнорирует подобную запись, остальные браузеры корректно её понимают.

! Псевдоэлементы не могут применяться к внутренним стилям, только к таблице связанных или глобальных стилей.

Далее перечислены псевдоэлементы, их описание и свойства.

:after

Применяется для вставки назначенного контента после элемента. Этот псевдоэлемент работает совместно со стиливым свойством **content**, которое определяет содержимое для вставки. В примере 1.59 показано использование псевдокласса **:after** для добавления текста в конец абзаца.

Пример 1.59. Применение псевдоэлемента :after

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">  
<html>  
<head>  
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">  
<title>Псевдоэлементы</title>  
<style type="text/css">  
  P.new:after {  
    content: " - Новьё!"; /* Добавляем после текста абзаца */  
  }  
</style>  
</head>  
<body>  
<p class="new">Ловля льва в пустыне с помощью метода золотого сечения.</p>  
<p>Метод ловли льва простым перебором.</p>  
</body>  
</html>
```

Результат примера показан на рис. 1.38.

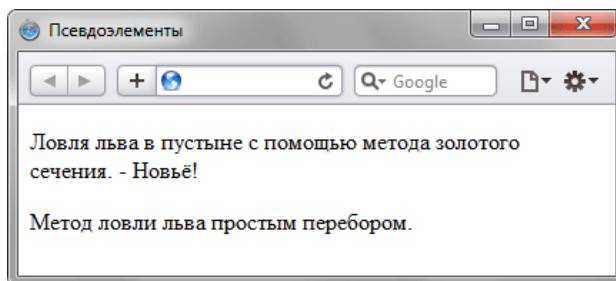


Рис. 1.38. Добавление текста к абзацу с помощью :after

В данном примере к содержимому абзаца с классом **new** добавляется дополнительное слово, которое выступает значением свойства **content**.

! Псевдоэлементы **:after** и **:before**, а также стиливое свойство **content** не поддерживаются браузером Internet Explorer до седьмой версии включительно.

:before

По своему действию `:before` аналогичен псевдоэлементу `:after`, но вставляет контент до элемента. В примере 1.60 показано добавление маркеров своего типа к элементам списка посредством скрытия стандартных маркеров и применения псевдокласса `:before`.

Пример 1.60. Использование псевдоэлемента `:before`

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Псевдоэлементы</title>
<style type="text/css">
  UL {
    padding-left: 0; /* Убираем отступ слева */
    list-style-type: none; /* Прячем маркеры списка */
  }
  LI:before {
    content: "◊ "; /* Добавляем перед элементом списка символ */
  }
</style>
</head>
<body>
<ul>
<li>Метод простых итераций</li>
<li>Метод случайных чисел</li>
<li>Метод дихотомии</li>
<li>Метод золотого сечения</li>
</ul>
</body>
</html>
```

Результат примера показан ниже (рис. 1.39).

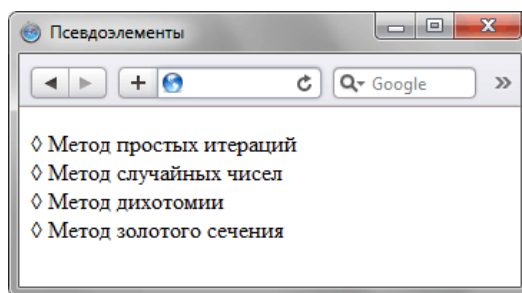



Рис. 1.39. Изменение вида маркеров с помощью `:before`

В данном примере псевдокласс `:before` устанавливается для селектора `LI`, определяющего элементы списка. Добавление желаемых символов происходит путем задания значения свойства `content`. Обратите внимание, что в качестве аргумента не обязательно выступает текст, могут применяться также символы юникода.

`:first-letter`

Определяет стиль первого символа в тексте элемента, к которому добавляется. Это позволяет создавать в тексте буквицу и выступающий инициал.

 Буквица представляет собой увеличенную первую букву, базовая линия которой ниже на одну или несколько строк базовой линии основного текста. Выступающий инициал — увеличенная прописная буква, базовая линия которой совпадает с базовой линией основного текста.

Рассмотрим пример создания выступающего инициала. Для этого требуется добавить к селектору `P` псевдоэлемент `:first-letter` и установить желаемый стиль инициала. В частности, увеличить размер текста и поменять цвет текста (пример 1.61).

Пример 1.61. Использование псевдоэлемента `first-letter`

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Псевдоэлементы</title>
<style type="text/css">
  P {
    font-family: Arial, Helvetica, sans-serif; /* Гарнитура шрифта основного текста */
    font-size: 0.9em; /* Размер шрифта */
    color: black; /* Черный цвет текста */
  }
  P:first-letter {
    font-family: 'Times New Roman', Times, serif; /* Гарнитура шрифта первой буквы */
    font-size: 2em; /* Размер шрифта первого символа */
    color: red; /* Красный цвет текста */
  }
</style>
</head>
<body>
<p>Псевдоэлементы</p>
</body>
</html>
```

```

}
</style>
</head>
<body>
<p>Метод случайных чисел является вариантом метода простых итераций,
в котором происходит перебор участков пустыни для поиска льва.
Разница только в том, что области выбираются не последовательно,
а случайно. Таким образом, в лучшем случае поиск может закончиться
сразу же, а в худшем — превратиться в метод перебора.</p>
<p>Поскольку номер области выбирается случайно, он может выпасть
больше одного раза. Однако переходить второй раз в уже просмотренную
область, необходимости нет.</p>
</body>
</html>

```

Результат примера показан ниже (рис. 1.40).

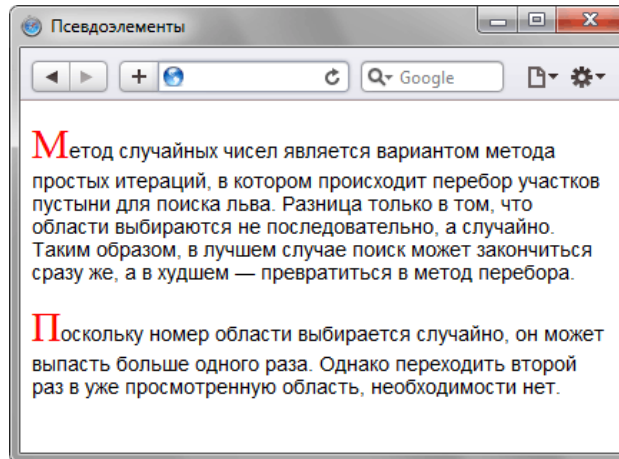


Рис. 1.40. Создание выступающего инициала

В данном примере изменяется шрифт, размер и цвет первой буквы каждого абзаца текста.

:first-line

Определяет стиль первой строки блочного текста. Длина этой строки зависит от многих факторов, таких как используемый шрифт, размер окна браузера, ширина блока, языка и т.д.

⚠ К псевдоэлементу `:first-line` могут применяться не все стилевые свойства. Допустимо использовать свойства, относящиеся к шрифту, изменению цвет текста и фона, а также: `clear`, `line-height`, `letter-spacing`, `text-decoration`, `text-transform`, `vertical-align` и `word-spacing`.

В примере 1.62 показано использование псевдоэлемента `:first-line` применительно к абзацу текста.

Пример 1.62. Выделение первой строки текста

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```

<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Псевдоэлементы</title>
<style type="text/css">
P:first-line {
color: red; /* Красный цвет текста */
font-style: italic; /* Курсивное начертание */
}
</style>
</head>
<body>
<p>Интересно, а существует ли способ действительно практического применения
first-line? Нет, не такого, чтобы можно было бы показать, что это возможно,
а чтобы воистину захватило дух от красоты решения, загорелись глаза от скрытых
перспектив, после чего остается только сказать себе, что вот это вот, это самое
сделать по-другому, также изящно и эффектно просто невозможно.</p>
</body>
</html>

```

Результат примера показан на рис. 1.41.

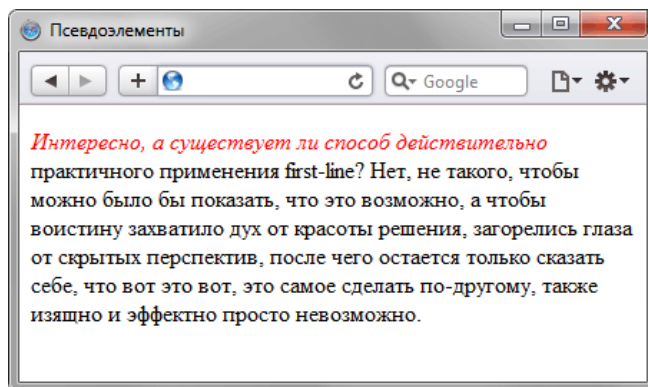


Рис. 1.41. Результат применения псевдоэлемента `:first-line`

В данном примере первая строка выделяется красным цветом и курсивным начертанием. Обратите внимание, что при изменении ширины окна браузера, стиль первой строки остается постоянным, независимо от числа входящих в нее слов.

Группирование

При создании стиля для сайта, когда одновременно используется множество селекторов, возможно появление повторяющихся стилевых правил. Чтобы не повторять дважды одни и те же элементы, их можно сгруппировать для удобства представления и сокращения кода. В примере 1.63 показана обычная запись, здесь для каждого селектора приводится свой набор стилевых свойств.

Пример 1.63. Стиль для каждого селектора

```
H1 {
  font-family: Arial, Helvetica, sans-serif;
  font-size: 160%;
  color: #003;
}
H2 {
  font-family: Arial, Helvetica, sans-serif;
  font-size: 135%;
  color: #333;
}
H3 {
  font-family: Arial, Helvetica, sans-serif;
  font-size: 120%;
  color: #900;
}
P {
  font-family: Times, serif;
}
```

Из данного примера видно, что стиль для тегов заголовков содержит одинаковое значение `font-family`. Группирование как раз и позволяет установить одно свойство сразу для нескольких селекторов, как показано в примере 1.64.

Пример 1.64. Сгруппированные селекторы

```
H1, H2, H3 {
  font-family: Arial, Helvetica, sans-serif;
}
H1 {
  font-size: 160%;
  color: #003;
}
H2 {
  font-size: 135%;
  color: #333;
}
H3 {
  font-size: 120%;
  color: #900;
}
```

В данном примере `font-family` единое для всех селекторов применяется сразу к нескольким тегам, а индивидуальные свойства уже добавляются к каждому селектору отдельно.

Селекторы группируются в виде списка тегов, разделенных между собой запятыми. В группу могут входить не только селекторы тегов, но также идентификаторы и классы. Общий синтаксис следующий.

```
Селектор 1, Селектор 2, ... Селектор N { Описание правил стиля }
```

При такой записи правила стиля применяются ко всем селекторам, перечисленным в одной группе.

Наследование

Наследованием называется перенос правил форматирования для элементов, находящихся внутри других. Такие элементы являются дочерними, и они наследуют некоторые стилевые свойства своих родителей, внутри которых располагаются.

Разберем наследование на примере таблицы. Особенностью таблиц можно считать строгую иерархическую структуру тегов. Вначале следует контейнер `<table>` внутри которого добавляются теги `<tr>`, а затем идет тег `<td>`. Если в стилях для селектора `TABLE` задать цвет текста, то он автоматически устанавливается для содержимого ячеек, как показано в примере 1.65.

Пример 1.65. Наследование параметров цвета

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Наследование</title>
<style type="text/css">
TABLE {
color: red; /* Цвет текста */
background: #333; /* Цвет фона таблицы */
border: 2px solid red; /* Красная рамка вокруг таблицы */
}
</style>
</head>
<body>
<table cellpadding="4" cellspacing="0">
<tr>
<td>Ячейка 1</td><td>Ячейка 2</td>
</tr>
<tr>
<td>Ячейка 3</td><td>Ячейка 4</td>
</tr>
</table>
</body>
</html>
```

В данном примере для всей таблицы установлен красный цвет текста, поэтому в ячейках он также применяется, поскольку тег `<td>` наследует свойства тега `<table>`. При этом следует понимать, что не все стилевые свойства наследуются. Так, `border` задает рамку вокруг таблицы в целом, но никак не вокруг ячеек. Аналогично не наследуется значение свойства `background`. Тогда почему цвет фона у ячеек в данном примере темный, раз он не наследуется? Дело в том, что у свойства `background` в качестве значения по умолчанию выступает `transparent`, т.е. прозрачность. Таким образом цвет фона родительского элемента «проглядывает» сквозь дочерний элемент.

Чтобы определить, наследуется значение стилевого свойства или нет, требуется заглянуть в справочник по свойствам CSS и посмотреть там. Подключать свою интуицию в подобном случае бесполезно, может и подвести.

Наследование позволяет задавать значения некоторых свойств единожды, определяя их для родителей верхнего уровня. Допустим, требуется установить цвет и шрифт для основного текста. Достаточно воспользоваться селектором `BODY`, добавить для него желаемые свойства, и цвет текста внутри абзацев и других текстовых элементов поменяется автоматически (пример 1.66).

Пример 1.66. Параметры текста для всей веб-страницы

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Наследование</title>
<style type="text/css">
BODY {
font-family: Arial, Helvetica, sans-serif; /* Гарнитура шрифта */
color: navy; /* Синий цвет текста */
}
</style>
</head>
<body>
<p>Цвет текста этого абзаца синий.</p>
</body>
</html>
```

В данном примере рубленый шрифт и цвет текста абзацев устанавливается с помощью селектора `BODY`. Благодаря наследованию уже нет нужды задавать цвет для каждого элемента документа в отдельности. Однако бывают варианты, когда требуется все-таки изменить цвет для отдельного контейнера. В этом случае придется переопределять нужные параметры явно, как показано в примере 1.67.

Пример 1.67. Изменение свойств наследуемого элемента

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Наследование</title>
<style type="text/css">
BODY {
font-family: Arial, Helvetica, sans-serif; /* Гарнитура шрифта */
color: navy; /* Синий цвет текста */
}
P.red {
color: maroon; /* Темно-красный цвет текста */
}
</style>
</head>
<body>
<p>Цвет текста этого абзаца синий.</p>
<p class="red">А у этого абзаца цвет текста уже другой.</p>
</body>
</html>
```

В данном примере цвет первого абзаца наследуется от селектора **BODY**, а для второго установлен явно через класс с именем **red**.

Верстаю сайты

Занимаюсь (X)HTML/CSS вёрсткой.

Делаю действительно качественную работу.

- любой сложности
- под любые платформы
- под любые браузеры
- валидная начисто!
- уникальная организация (x)html/css кода
- семантически корректный код
- логичность верстки
- табличная, дивная (предпочитаю второе, 49/50 как правило блочные работы)
- индивидуальный подход к каждому клиенту

Цены

От 100\$ за шаблон.

IE6+ доплата, в зависимости от сложности и времени.

Контакты

e-mail: psywalker09@gmail.com

qip: 366905663

skype: walker1071

Отзывы обо мне

<http://forum.htmlbook.ru/index.php?showtopic=20231>

Портфолио и код моих работ демонстрирую непосредственно заказчику

Каскадирование

Аббревиатура CSS расшифровывается как Cascading Style Sheets (каскадные таблицы стилей), где одним из ключевых слов выступает «каскад». Под каскадом в данном случае понимается одновременное применение разных стилевых правил к элементам документа — с помощью подключения нескольких стилевых файлов, наследования свойств и других методов. Чтобы в подобной ситуации браузер понимал, какое в итоге правило применять к элементу, и не возникало конфликтов в поведении разных браузеров, введены определенные приоритеты.

Ниже приведены приоритеты браузеров, которыми они руководствуются при обработке стилевых правил. Чем выше в списке находится пункт, тем ниже его приоритет, и наоборот.

1. Стиль браузера.
2. Стиль пользователя.
3. Стиль автора.
4. Стиль автора с добавлением !important.
5. Стиль пользователя с добавлением !important.

Самым низким приоритетом обладает стиль браузера — оформление, которое по умолчанию применяется к элементам веб-страницы браузером. Это оформление можно увидеть в случае «голого» HTML, когда к документу не добавляется никаких стилей.

!important

Ключевое слово **!important** играет роль в том случае, когда пользователи подключают свою собственную таблицу стилей. Если возникает противоречие, когда стиль автора страницы и пользователя для одного и того же элемента не совпадает, то **!important** позволяет повысить приоритет стиля или его важность, иными словами.

При использовании пользовательской таблицы стилей или одновременном применении разного стиля автора и пользователя к одному и тому же селектору, браузер руководствуется следующим алгоритмом.

- **!important** добавлен в авторский стиль — будет применяться стиль автора.
- **!important** добавлен в пользовательский стиль — будет применяться стиль пользователя.
- **!important** нет как в авторском стиле, так и в стиле пользователя — будет применяться стиль автора.
- **!important** содержится в авторском стиле и в стиле пользователя — будет применяться стиль пользователя.

Синтаксис применения **!important** следующий.

```
Свойство: значение !important;
```

Вначале пишется желаемое стилевое свойство, затем через двоеточие его значение и в конце после пробела указывается ключевое слово **!important**.

Повышение важности требуется не только для регулирования приоритета между авторской и пользовательской таблицей стилей, но и для повышения специфичности определенного селектора.

Специфичность

Если к одному элементу одновременно применяются противоречивые стилевые правила, то более высокий приоритет имеет правило, у которого значение специфичности селектора больше. Специфичность это некоторая условная величина, вычисляемая следующим образом. За каждый идентификатор (в дальнейшем будем обозначать их количество через a) начисляется 100, за каждый класс и псевдокласс (b) начисляется 10, за каждый селектор тега и псевдоэлемент (c) начисляется 1. Суммируя указанные значения, получим значение специфичности для данного селектора.

```
*      {} /* a=0 b=0 c=0 -> специфичность = 0 */
li     {} /* a=0 b=0 c=1 -> специфичность = 1 */
li:first-line {} /* a=0 b=0 c=2 -> специфичность = 2 */
ul li  {} /* a=0 b=0 c=2 -> специфичность = 2 */
ul ol+li {} /* a=0 b=0 c=3 -> специфичность = 3 */
ul li.red {} /* a=0 b=1 c=2 -> специфичность = 12 */
li.red.level {} /* a=0 b=2 c=1 -> специфичность = 21 */
#t34   {} /* a=1 b=0 c=0 -> специфичность = 100 */
#content #wrap {} /* a=2 b=0 c=0 -> специфичность = 200 */
```

Встроенный стиль, добавляемый к тегу через атрибут **style**, имеет специфичность 1000, поэтому всегда перекрывает связанные и глобальные стили. Однако добавление **!important** перекрывает в том числе и встроенные стили.

Если два селектора имеют одинаковую специфичность, то применяться будет тот стиль, что определен в коде ниже.

В примере 1.68 показано, как влияет специфичность на стиль элементов списка.

Пример 1.68. Цвет списка

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Список</title>
  <style type="text/css">
    #menu ul li {
      color: green;
    }
    .two {
      color: red;
    }
  </style>
</head>
<body>
  <div id="menu">
    <ul>
      <li>Первый</li>
      <li class="two">Второй</li>
      <li>Третий</li>
    </ul>
  </div>
</body>
</html>
```

В данном примере цвет текста списка задан зеленым, а второй пункт списка с помощью класса `two` выделен красным цветом. Вычисляем специфичность селектора `#menu ul li` — один идентификатор (100) и два тега (2) в сумме дают значение 102, а селектор `.two` будет иметь значение специфичности 20, что явно меньше. Поэтому текст окрашиваться красным цветом не будет. Чтобы исправить ситуацию, необходимо либо понизить специфичность первого селектора, либо повысить специфичность второго (пример 1.69).

Пример 1.69. Изменение специфичности

```
/* Понижаем специфичность первого селектора */
ul li {...} /* Убираем идентификатор */
.two {...}

/* Повышаем специфичность второго селектора */
#menu ul li {...}
#menu .two {...} /* Добавляем идентификатор */

#menu ul li {...}
.two { color: red !important; } /* Добавляем !important */
```

Добавление идентификатора используется не только для изменения специфичности селектора, но и для применения стиля только к указанному списку. Поэтому понижение специфичности за счет убирания идентификатора применяется редко, в основном, повышается специфичность нужного селектора.

Валидация CSS

Валидацией называется проверка CSS-кода на соответствие спецификации CSS2.1 или CSS3. Соответственно, корректный код, не содержащий ошибок, называется валидным, а не удовлетворяющий спецификации — невалидный. Наиболее удобно делать проверку кода через сайт <http://jigsaw.w3.org/css-validator/>, с помощью этого сервиса можно указать адрес документа, загрузить файл или проверить набранный текст. Большим плюсом сервиса является поддержка русского и украинского языка.

Проверить URI

Эта вкладка позволяет указывать адрес страницы размещенной в Интернете. Протокол `http://` можно не писать, он будет добавлен автоматически (рис. 1.42).

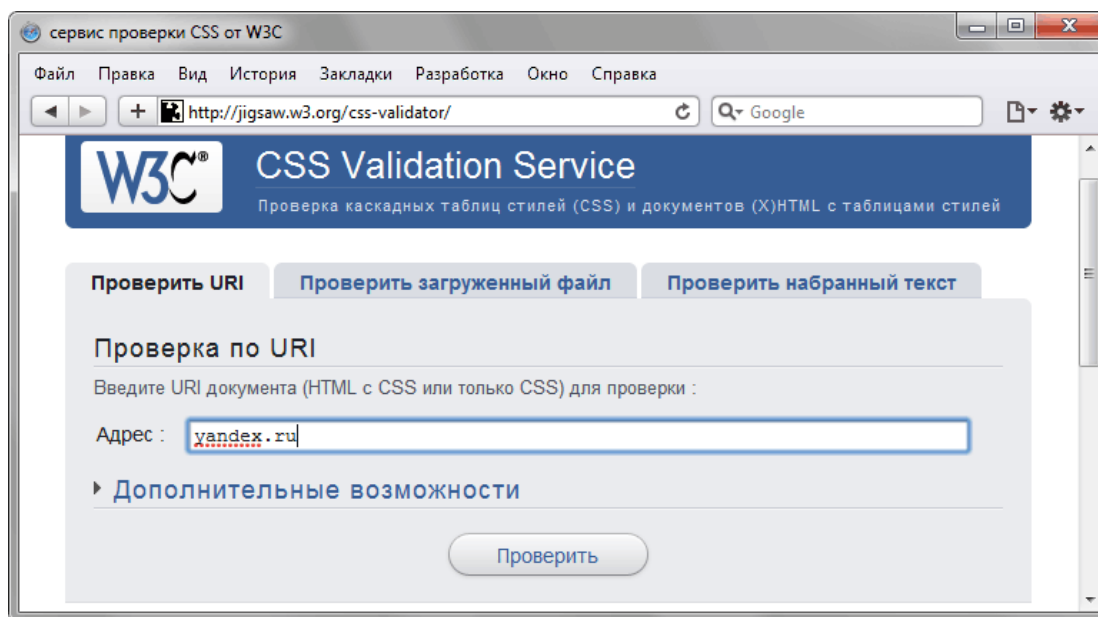


Рис. 1.42. Проверка документа по адресу

После ввода адреса нажмите на кнопку «Проверить» и появится одна из двух надписей: «Поздравляем! Ошибок не обнаружено» в случае успеха или «К сожалению, мы обнаружили следующие ошибки» при невалидном коде. Сообщения об ошибках или предупреждениях содержат номер строки, селектор и описание ошибки.

Проверить загруженный файл

Эта вкладка позволяет загрузить HTML или CSS-файл и проверить его на наличие ошибок (рис. 1.43).

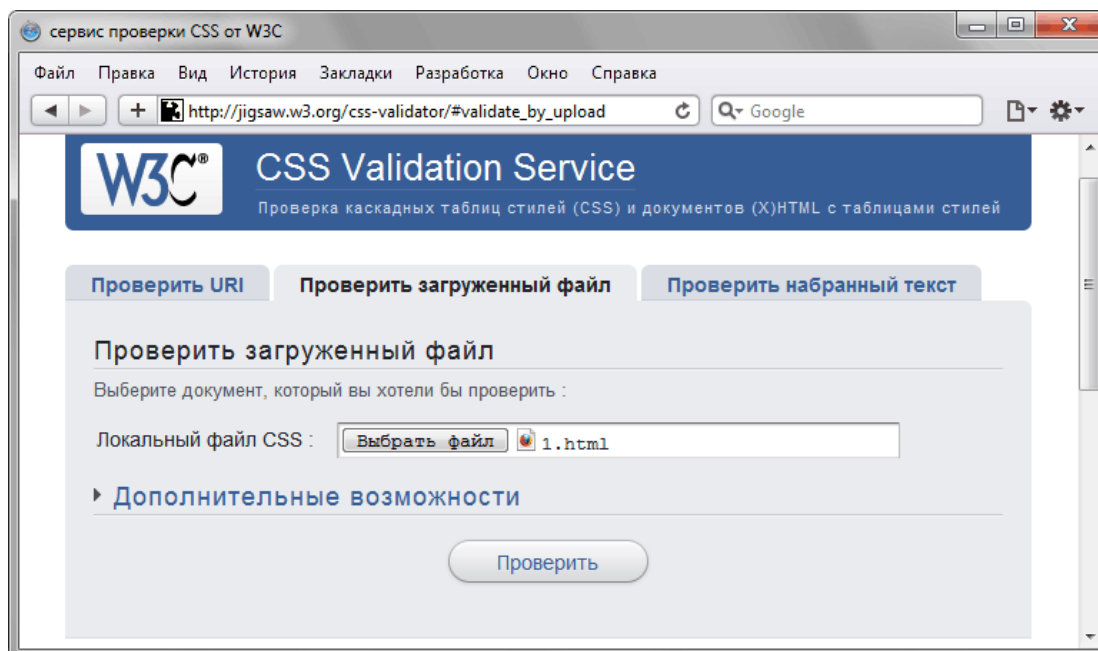


Рис. 1.43. Проверка файла при его загрузке

Сервис автоматически распознает тип файла и если указан HTML-документ, вычлняет из него стиль для валидации.

Проверить набранный текст

Последняя вкладка предназначена для непосредственного ввода HTML или CSS-кода, при этом проверке будет подвергнут только стиль (рис. 1.44).

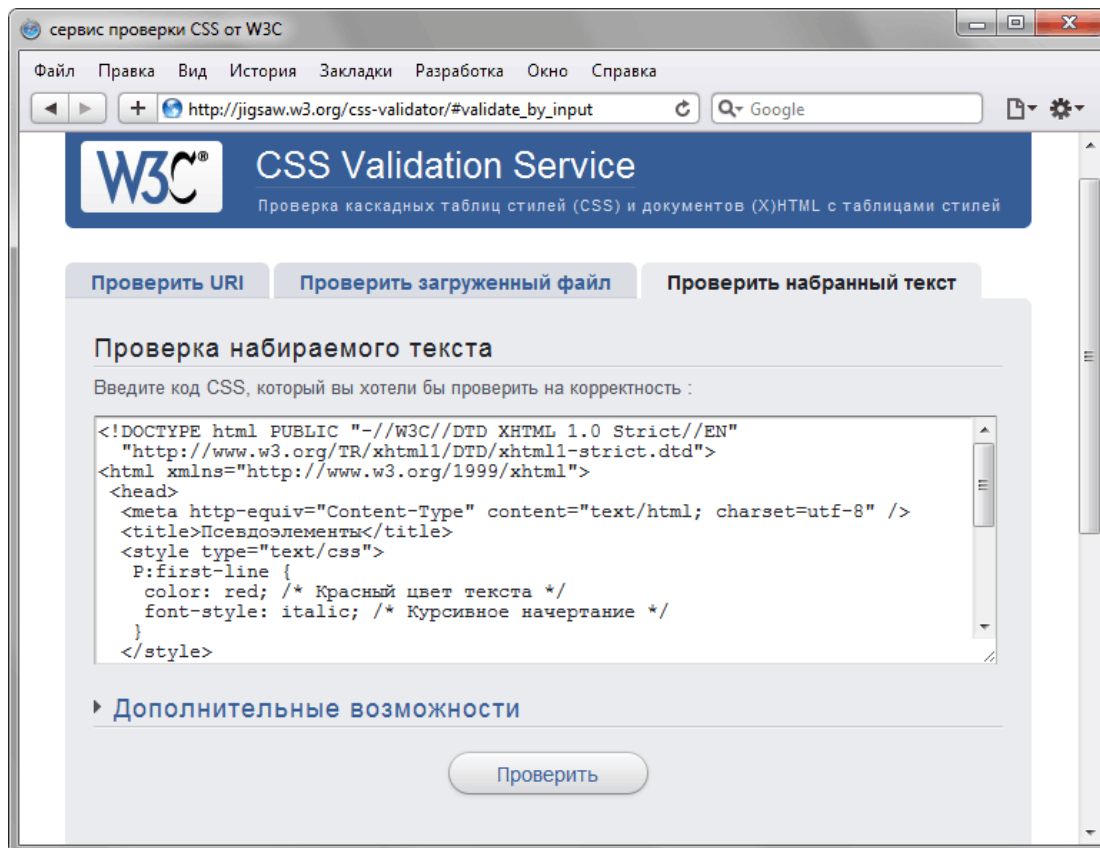


Рис. 1.44. Проверка введенного кода

Этот вариант представляется наиболее удобным для проведения различных экспериментов над кодом или быстрой проверки небольших фрагментов.

Выбор версии CSS

В CSS3 добавлено много новых стилевых свойств по сравнению с предыдущей версией, поэтому проводить проверку кода следует с учетом версии. По умолчанию в сервисе указан CSS2.1, так что если вы хотите проверить код на соответствие CSS3, это следует указать явно. Для этого щелкните по тексту «Дополнительные возможности» и в открывшемся блоке из списка «Профиль» выберите CSS3 (рис. 1.45).

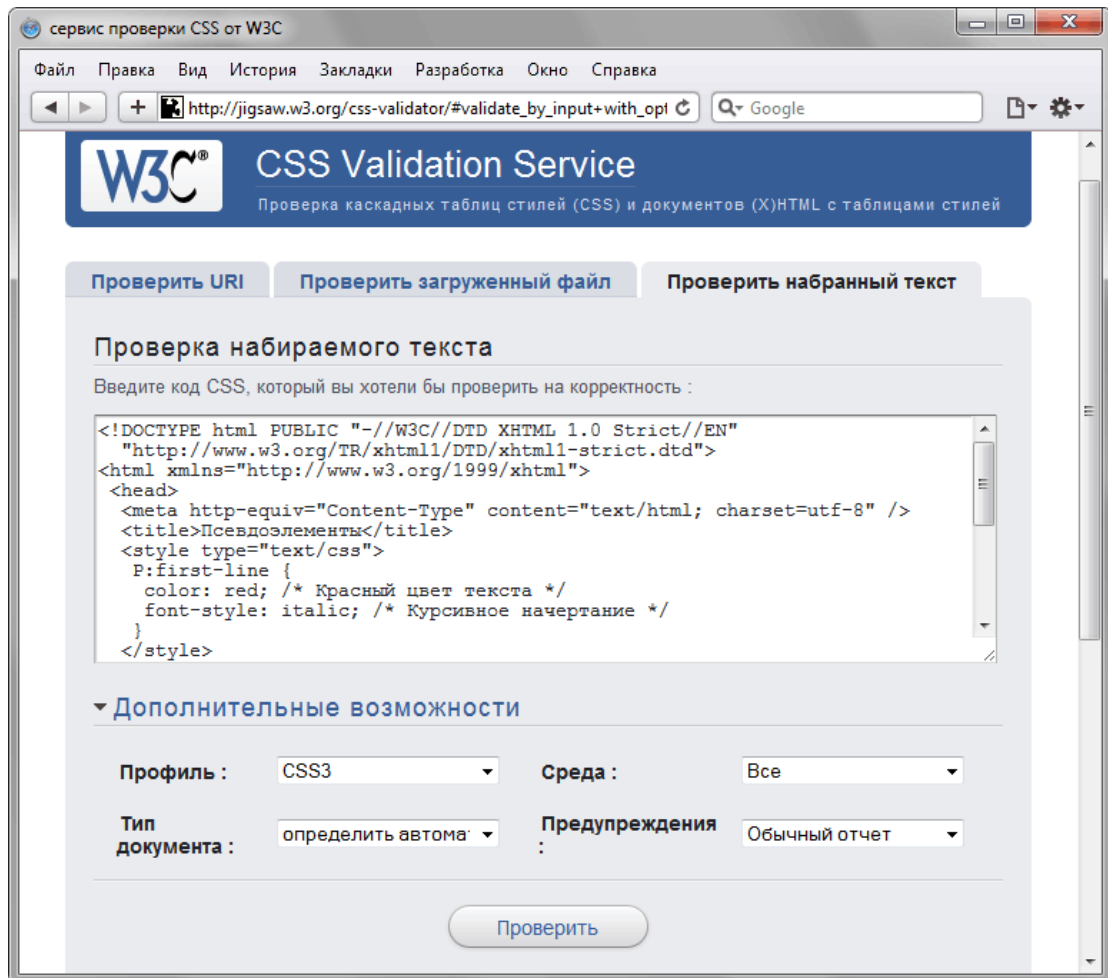


Рис. 1.45. Указание версии CSS для проверки

Идентификаторы и классы

Периодически поднимается спор о целесообразности использования идентификаторов в вёрстке. Основной довод состоит в том, что идентификаторы предназначены для доступа и управления элементами веб-страницы с помощью скриптов, а для изменения стилей элементов должны применяться исключительно классы. В действительности нет разницы, через что задавать стили, но следует помнить о некоторых особенностях идентификаторов и классов, а также подводных камнях, которые могут поджидать разработчиков.

Для начала перечислим характерные признаки этих селекторов.

Идентификаторы

- В коде документа каждый идентификатор уникален и должен быть включён лишь один раз.
- Имя идентификатора чувствительно к регистру.
- Через метод `getElementById` можно получить доступ к элементу по его идентификатору и изменить свойства элемента.
- Стиль для идентификатора имеет приоритет выше, чем у классов.

Классы

- Классы могут использоваться в коде неоднократно.
- Имена классов чувствительны к регистру.
- Классы можно комбинировать между собой, добавляя несколько классов к одному тегу.

Идентификаторы

Если во время работы веб-страницы требуется изменить стиль некоторых элементов «на лету» или выводить внутри них какой-либо текст, с идентификаторами это делается гораздо проще. Обращение к элементу происходит через метод `getElementById`, параметром которого служит имя идентификатора. В примере 1.70 к текстовому полю формы добавляется идентификатор с именем `userName`, затем с помощью функции JavaScript делается проверка на то, что пользователь ввёл в это поле какой-либо текст. Если никакого текста нет, но кнопка Submit нажата, выводится сообщение внутри тега с идентификатором `msg`. Если всё правильно, данные формы отправляются по адресу, указанному атрибутом `action`.

Пример 1.70. Проверка данных формы

XHTML 1.0 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>Проверка формы</title>
    <script type="text/javascript">
      function validForm(f) {
        user = document.getElementById("userName");
        if(user.value == "") document.getElementById("msg").innerHTML = 'Пожалуйста, введите имя.';
        else f.submit();
      }
    </script>
  </head>
  <body>
    <form action="handler.php" onsubmit="validForm(this); return false">
      <p>Введите свое имя:</p>
      <div id="msg"></div>
      <p><input type="text" id="userName" name="user" /></p>
      <p><input type="submit" /></p>
    </form>
  </body>
</html>
```

Поскольку идентификаторы чувствительны к регистру, имеет значение их однотипное написание. Внутри тега `<input>` используется имя `userName`, его же следует указать и в методе `getElementById`. При ошибочном написании, например, `username`, скрипт перестанет работать, как требуется.

В примере выше стили вообще никакого участия не принимали, сами идентификаторы требовались для работы скриптов. При использовании в CSS следует учитывать, что идентификаторы обладают высоким приоритетом по сравнению с классами. Поясним это на примере 1.71.

Пример 1.71. Сочетание стилей

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!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">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

```

<title>Идентификаторы</title>
<style type="text/css">
  #A, .a {
    border: none;
    background: #f0f0f0;
    color: green;
    padding: 5px;
  }
  .b {
    border: 1px solid red;
    color: red;
    padding: 0;
  }
</style>
</head>
<body>
<p id="A" class="b">Стиль идентификатора</p>
<p class="a b">Стиль классов a и b</p>
<p class="b">Стиль класса b</p>
</body>
</html>

```

Для первого абзаца устанавливается стиль от идентификатора `A` и класса `b`, свойства которых противоречат друг другу. При этом стиль класса будет игнорироваться из-за особенностей каскадирования и специфичности. Для второго абзаца стиль задаётся через классы `a` и `b` одновременно. Приоритет у классов одинаковый, значит, в случае противоречия будут задействованы те свойства, которые указаны в стиле ниже. К последнему абзацу применяется стиль только от класса `b`. На рис. 1.46 показан результат применения стилей.

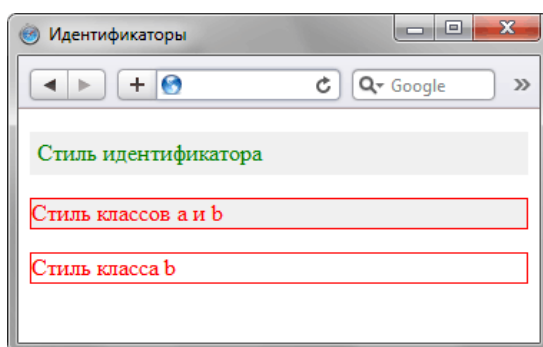


Рис. 1.46. Использование стилей для текста

Специфичность в каскадировании начинает играть роль при разрастании стилевого файла за счет увеличения числа селекторов, что характерно для больших и сложных сайтов. Чтобы стиль применялся корректно, необходимо грамотно управлять специфичностью селекторов путем использования идентификаторов, повышения важности через `!important`, порядком следования свойств.

Классы

Поскольку к элементу одновременно можно добавлять более одного класса, это позволяет завести несколько универсальных классов со стилевыми свойствами на все случаи и включать их к тегам при необходимости. Предположим, что большинство блоков на странице имеют закругленные уголки, причем некоторые блоки еще имеют красную рамку, а некоторые нет. В этом случае можем написать такой стиль (пример 1.72).

Пример 1.72. Использование классов

```

.r, .b {
  padding: 10px;
  background: #FCE3EE;
}
.r {
  border-radius: 8px;
  -webkit-border-radius: 8px;
  -moz-border-radius: 8px
}
.b { border: 1px solid #ED1C24; }
.n { border: none; }

```

Указывая разные классы в атрибуте `class` мы можем комбинировать набор стилевых свойств и получить таким образом блоки с рамкой, блоки без рамки, со скругленными или прямыми углами. Это несколько похоже на группирование селекторов, но обладает большей гибкостью.

Верстаю сайты

Занимаюсь (X)HTML/CSS вёрсткой.

Делаю действительно качественную работу.

- любой сложности
- под любые платформы
- под любые браузеры
- валидная начисто!
- уникальная организация (x)html/css кода
- семантически корректный код
- логичность верстки
- табличная, дивная (предпочитаю второе, 49/50 как правило блочные работы)
- индивидуальный подход к каждому клиенту

Цены

От 100\$ за шаблон.

IE6+ доплата, в зависимости от сложности и времени.

Контакты

e-mail: psywalker09@gmail.com

qip: 366905663

skype: walker1071

Отзывы обо мне

<http://forum.htmlbook.ru/index.php?showtopic=20231>

Портфолио и код моих работ демонстрирую непосредственно заказчику

Комплексные решения по созданию эффективных IT-инфраструктур в полном соответствии с бизнес задачами вашей компании



СЕРВЕРНЫЕ И СЕТЕВЫЕ РЕШЕНИЯ

Поставки серверного оборудования и программного обеспечения, а также создание комплексных решений на базе серверных технологий для построения информационных систем различного масштаба: от серверных решений начального уровня, до корпоративных систем крупных предприятий.

РЕШЕНИЯ ДЛЯ ПЕЧАТИ

Оптимизация и упрощение сетей устройств печати и копирования за счет выравнивания производительности работы конечных пользователей, улучшения технической поддержки и экономии затрат. Мы улучшим Вашу среду печати сэкономив время, деньги и технические ресурсы

КОМПЬЮТЕРНЫЕ РЕШЕНИЯ

Реализация компьютерных систем, а также разработка и производство персональных компьютеров под собственной торговой маркой "RestR". Мы можем предложить компьютеры любого уровня - от офисных и недорогих, до самых современных игровых и профессиональных рабочих станций.

РЕШЕНИЯ В ОБЛАСТИ ЭЛЕКТРОПИТАНИЯ

Конфигурирование и внедрение максимально эффективных комплексных инфраструктурных проектов на базе интегрированных решений APC, обеспечивающих защиту от перегрева и проблем с электропитанием, обеспечивая постоянную готовность оборудования.

Мы постоянно следим за развитием IT-отрасли. Установленные отношения с мировыми производителями - служат крепкой основой для предложения нашим партнерам выгодных и привлекательных решений, которые отличаются высоким качеством и безотказной работой.



Подробности на нашем сайте WWW.RESTR.COM

ООО Рестрком 125480, г. Москва, ул. Героев Панфиловцев, д.10, кор.1. м.Планерная

Телефон: (495) 649-62-03 (многоканальный)

E-mail: info@restr.com



Дизайн інтер'єров
3D-визуалізація



arch.design@jungle.in.ua