

Вступление

Данный самоучитель предназначен в первую очередь для начинающих, только постигающих азы создания сайтов. Множество примеров, иллюстраций, вопросов для проверки и заданий для самостоятельной работы помогут быстрее изучить CSS.

Обозначения

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

**** — тег.

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

right — значение свойства или параметра тега.

HTML — термин.

class — имя класса, идентификатора или переменной.

Tab — клавиша на клавиатуре.

File > Open — пункт меню указанной программы.

Примеры

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

IE — Internet Explorer.

Cr — Google Chrome.

Op — Opera.

Sa — Apple Safari.

Fx — Mozilla Firefox.

Работоспособность примера в браузере помечается цветом, зелёный — работает, красный — нет.

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.

Введение в CSS

После знакомства с HTML разработчики сайтов разделяются на две основные категории. Одна часть считает, что с помощью HTML на сайте можно создавать всё или практически всё, другая же понимает, что в целом средств разметки недостаточно для оформления веб-документов. Действительно, HTML лишь первый этап в процессе обучения созданию сайтов. Следующим шагом является изучение стилей или CSS (Cascading Style Sheets, каскадные таблицы стилей).

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

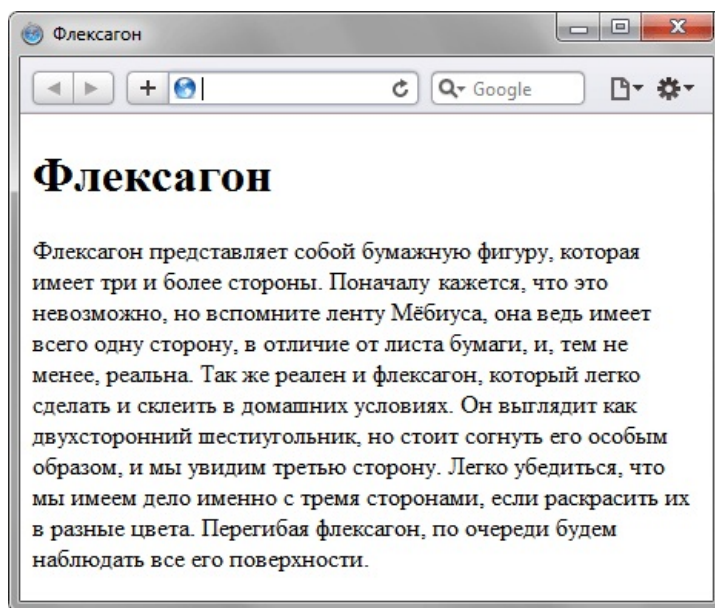


Рис. 1.1. Веб-страница, созданная только на HTML

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

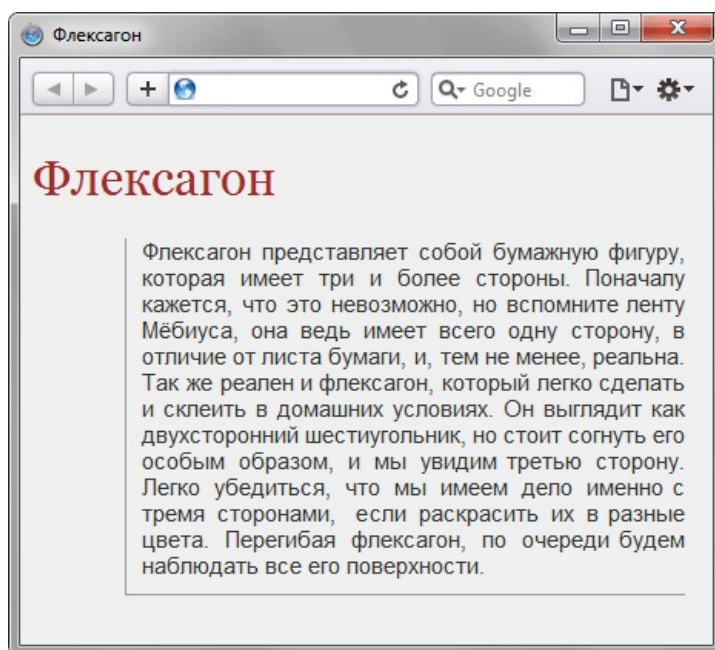


Рис. 1.2. Веб-страница, созданная на HTML и CSS

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

Пример 1.1. Исходный код документа

HTML5 CSS 2.1 IE Cr Op Sa Fx

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Флексагон</title>
    <meta charset="utf-8">
    <link rel="stylesheet" href="style.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">`. 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.

CSS представляет собой свой собственный язык, который совпадает с HTML только некоторыми значениями, например способом определения цвета.

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 «голового» 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 1.2 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.

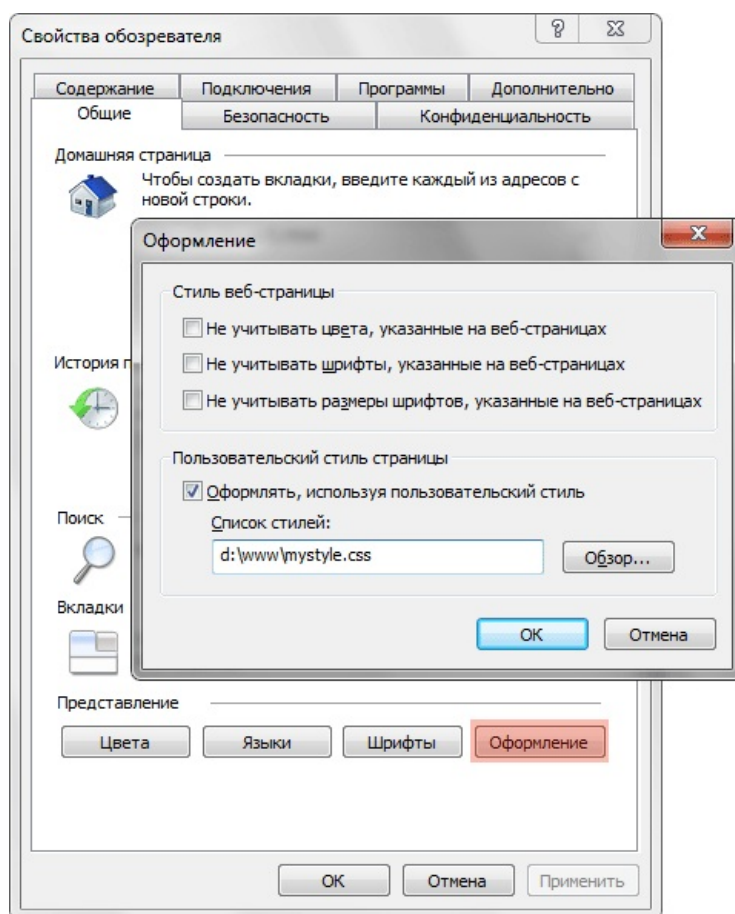


Рис. 1.3. Подключение стиля пользователя в браузере Internet Explorer

В браузере Opera аналогичное действие происходит через команду **Инструменты > Общие**

настройки > Вкладка «Расширенные» > Содержимое > Кнопка «Параметры стиля» (рис. 1.4).

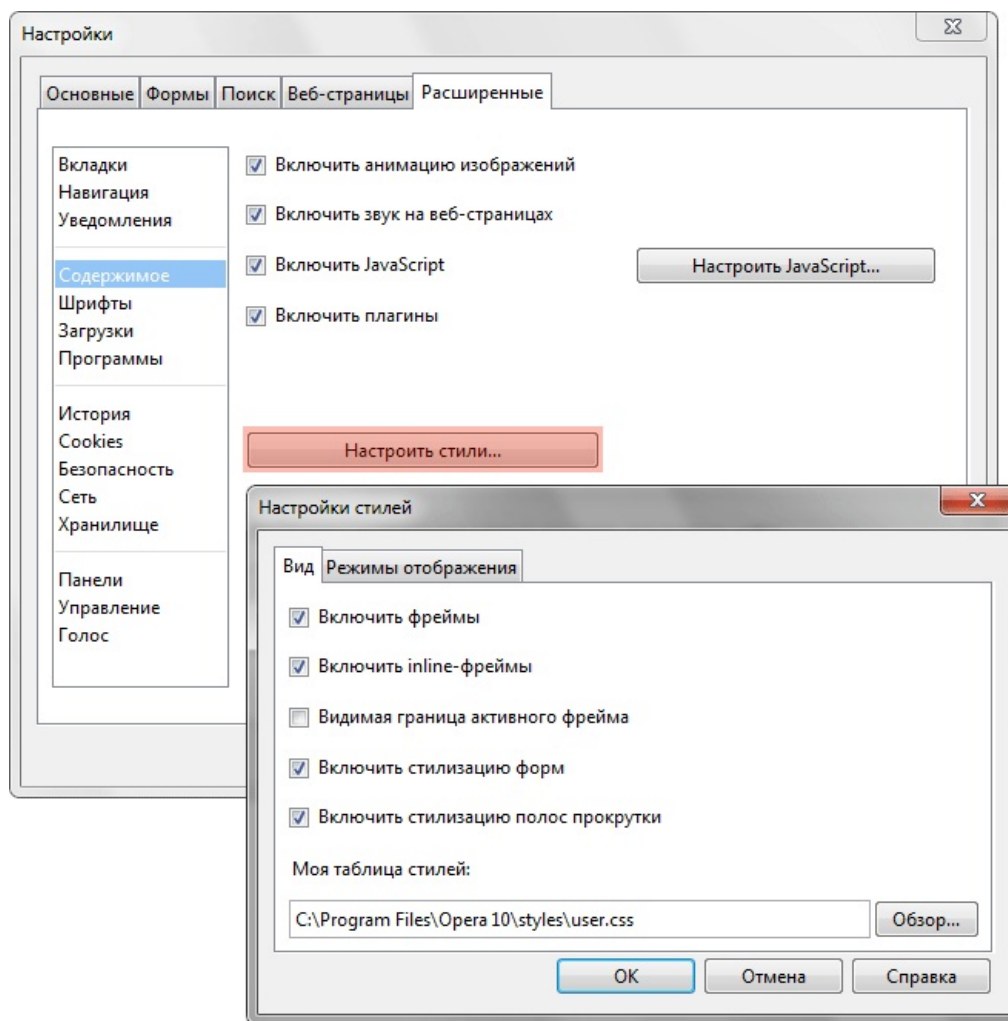


Рис. 1.4. Подключение стиля пользователя в браузере Opera

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 идёт стиль браузера.

Вопросы для проверки

1. Требуется задать цвет заголовка зелёным. Какое стилевое свойство подойдёт для этой цели?

1. font-color
2. color
3. font-family
4. text
5. font-size

2. Что такое стиль?

1. Способ сокращения HTML-кода за счёт переноса части данных в другой файл.
2. Язык разметки гипертекстовых документов.
3. Набор правил форматирования элементов веб-страницы.
4. Метод преобразований текстовых документов в HTML.

5. Технология, представляющая собой разные приёмы для вёрстки HTML-кода.

3. Как расшифровывается аббревиатура CSS?

1. Colorful Style Sheets
2. Cascading Style Sheets
3. Computer Style Sheets
4. Creative Style Sheets
5. Common Style Sheets

Ответы

1. color
2. Набор правил форматирования элементов веб-страницы.
3. Cascading Style Sheets

Преимущества стилей

Стили являются удобным, практичным и эффективным инструментом при вёрстке веб-страниц и оформления текста, ссылок, изображений и других элементов. Несмотря на явные плюсы применения стилей, рассмотрим все преимущества CSS, в том числе и незаметные на первый взгляд.

Разграничение кода и оформления

Идея о том, чтобы код HTML был свободен от элементов оформления вроде установки цвета, размера шрифта и других параметров, стара как мир. В идеале, веб-страница должна содержать только теги логического форматирования, а вид элементов задаётся через стили. При подобном разделении работа над дизайном и версткой сайта может вестись параллельно.

Разное оформление для разных устройств

С помощью стилей можно определить вид веб-страницы для разных устройств вывода: монитора, принтера, смартфона, планшета и др. Например, на экране монитора отображать страницу в одном оформлении, а при её печати — в другом. Эта возможность также позволяет скрывать или показывать некоторые элементы документа при отображении на разных устройствах.

Расширенные по сравнению с HTML способы оформления элементов

В отличие от HTML стили имеют гораздо больше возможностей по оформлению элементов веб-страниц. Простыми средствами можно изменить цвет фона элемента, добавить рамку, установить шрифт, определить размеры, положение и многое другое.

Ускорение загрузки сайта

При хранении стилей в отдельном файле, он кэшируется и при повторном обращении к нему извлекается из кэша браузера. За счёт кэширования и того, что стили хранятся в отдельном файле, уменьшается код веб-страниц и снижается время загрузки документов.

Кэшем называется специальное место на локальном компьютере пользователя, куда браузер сохраняет файлы при первом обращении к сайту. При следующем обращении к сайту эти файлы уже не скачиваются по сети, а берутся с локального диска. Такой подход позволяет существенно повысить скорость загрузки веб-страниц.

Единое стилевое оформление множества документов

Сайт это не просто набор связанных между собой документов, но и одинаковое расположение основных блоков, и их вид. Применение единообразного оформления заголовков, основного текста и других элементов создает преемственность между страницами и облегчает пользователям работу с сайтом и его восприятие в целом. Разработчикам же использование стилей существенно упрощает проектирование дизайна.

Централизованное хранение

Стили, как правило, хранятся в одном или нескольких специальных файлах, ссылка на которые указывается во всех документах сайта. Благодаря этому удобно править стиль в одном месте, при этом оформление элементов автоматически меняется на всех страницах, которые связаны с указанным файлом. Вместо того чтобы модифицировать десятки HTML-файлов, достаточно отредактировать один файл со стилем и оформление нужных документов сразу же поменяется.

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 3.1.

Пример 3.1. Подключение связанных стилей

HTML5 CSS 2.1 IE Cr Op Sa Fx

```
<!DOCTYPE HTML>
<html>
  <head>
    <meta charset="utf-8">
    <title>Стили</title>
    <link rel="stylesheet" href="mysite.css">
    <link rel="stylesheet" href="http://www.htmlbook.ru/main.css">
  </head>
  <body>
    <h1>Заголовок</h1>
    <p>Текст</p>
  </body>
</html>
```

Значение атрибута тега `<link>` — `rel` остаётся неизменным независимо от кода, как приведено в данном примере. Значение `href` задаёт путь к CSS-файлу, он может быть задан как относительно, так и абсолютно. Заметьте, что таким образом можно подключать таблицу стилей, которая находится на другом сайте.

Содержимое файла `mysite.css` подключаемого посредством тега `<link>` приведено в примере 3.2.

Пример 3.2. Файл со стилем

```
H1 {
  color: #000080;
  font-size: 200%;
  font-family: Arial, Verdana, sans-serif;
  text-align: center;
}
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 той же странице с помощью контейнера `<style>`, как показано в примере 3.3.

Пример 3.3. Использование глобального стиля

HTML5 CSS 2.1 IE Cr Op Sa Fx

```
<!DOCTYPE HTML>
<html>
  <head>
    <meta charset="utf-8">
    <title>Глобальные стили</title>
    <style>
      H1 {
        font-size: 120%;
        font-family: Verdana, Arial, Helvetica, sans-serif;
        color: #333366;
      }
    </style>
  </head>
  <body>
    <h1>Hello, world!</h1>
  </body>
</html>
```

В данном примере задан стиль тега `<h1>`, который затем можно повсеместно использовать на данной веб-странице (рис. 3.1).

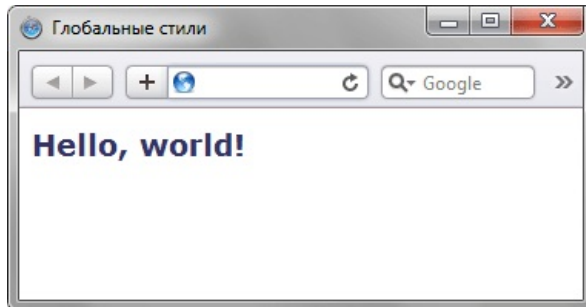


Рис. 3.1. Вид заголовка, оформленного с помощью стилей

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 3.4).

Пример 3.4. Использование внутреннего стиля

HTML5 CSS 2.1 IE Cr Op Sa Fx

```
<!DOCTYPE HTML>
<html>
  <head>
    <meta 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>` задаётся с помощью атрибута `style`, в котором через точку с запятой перечисляются стилевые свойства (рис. 3.2).

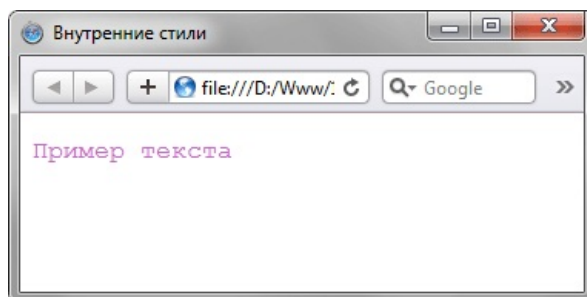


Рис. 3.2. Использование внутренних стилей для изменения вида текста

Внутренние стили рекомендуется применять на сайте ограниченно или вообще отказаться от их использования. Дело в том, что добавление таких стилей увеличивает общий объём 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 имеет приоритет внутренний стиль, затем глобальный стиль и в последнюю очередь связанный стиль. В примере 3.5 применяется сразу два метода добавления стиля в документ.

Пример 3.5. Сочетание разных методов

HTML5	CSS 2.1	IE	Cr	Op	Sa	Fx
-------	---------	----	----	----	----	----

```
<!DOCTYPE HTML>
<html>
<head>
  <meta charset="utf-8">
  <title>Подключение стиля</title>
  <style>
    H1 {
      font-size: 120%;
      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 задаётся красным цветом размером 36 пикселей с помощью внутреннего стиля, а следующий — зелёным цветом через таблицу глобальных стилей (рис. 3.3).

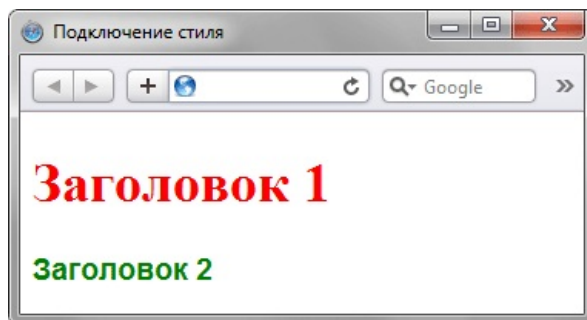


Рис. 3.3. Результат применения стилей

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 с внутренними 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 3.6 показано, как можно импортировать стиль из внешнего файла в таблицу глобальных стилей.

Пример 3.6. Импорт CSS

HTML5 CSS 2.1 IE Cr Op Sa Fx

```
<!DOCTYPE HTML>  
<html>  
  <head>  
    <meta charset="utf-8">  
    <title>Импорт</title>  
    <style>  
      @import url("style/header.css");  
      h1 {  
        font-size: 120%;  
        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 **header.css**, который расположен в папке **style**.

Аналогично происходит импорт и в файле со стилем, который затем подключается к документу (пример 3.7).

Пример 3.7. 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 содержимое файла `mysite.css`, который добавляется к нужным документам способом, показанным в примере 3.1, а именно с помощью тега `<link>`.

Вопросы для проверки

1. Сайт имеет более ста HTML-документов, имеющих одинаковое стилевое оформление. Какой способ подключения CSS подходит лучше всего?

1. Связанные стили.
2. Глобальные стили.
3. Блочные стили.
4. Внутренние стили.
5. Экспорт стиля.

2. В данном примере, какой цвет будет у текста на веб-странице?

```
<!DOCTYPE HTML>
<html>
<head>
  <meta charset="utf-8">
  <title>Цвет текста</title>
  <style>
    HTML { color: black; }
    BODY { color: red; }
    P { color: green; }
  </style>
</head>
<body>
  <p style="color: blue;"><span style="color: olive;">Текст</span></p>
</body>
</html>
```

1. Чёрный
2. Красный
3. Зелёный
4. Синий
5. Оливковый

3. Какой HTML-код применяется для подключения внешнего CSS-файла?

1. `<style>mystyle.css</style>`
2. `<style>@mystyle.css</style>`
3. `<link rel="stylesheet" href="mystyle.css">`
4. `<link>@import url(mystyle.css)</link>`
5. `<stylesheet>mystyle.css</stylesheet>`

4. Какой атрибут используется для определения внутреннего стиля?

1. `style`
2. `class`
3. `styles`
4. `font`

5. link

ОТВЕТЫ

1. Связанные стили.
2. Оливковый.
3. `<link rel="stylesheet" href="mystyle.css">`
4. style

Типы носителей

Широкое развитие различных платформ и устройств заставляет разработчиков делать под них специальные версии сайтов, что достаточно трудоёмко 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 определённого устройства. В табл. 4.1 перечислены некоторые типы носителей.

Табл. 4.1. 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 4.1.

Пример 4.1. Импорт стилевого файла

HTML5 CSS 2.1 IE Cr Op Sa Fx

```
<!DOCTYPE HTML>
<html>
<head>
  <meta charset="utf-8">
  <title>Импорт стиля</title>
  <style>
    /* Стиль для вывода результата на монитор */
    @import "/style/main.css" screen;
    /* Стиль для печати и смартфона */
    @import "/style/smart.css" print, handheld;
  </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 `smart.css` — при печати страницы и отображении на смартфоне.

Браузер Internet Explorer до седьмой версии включительно не поддерживает типы носителей при импорте стилевого файла.

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** идёт один или несколько типов носителя, перечисленных в табл. 4.1, если их больше одного, то они разделяются между собой запятой. После чего следуют обязательные фигурные скобки, внутри которых идёт обычное описание стилевых правил. В примере 4.2 показано, как задать разный стиль для печати и отображения на мониторе.

Пример 4.2. Стили для разных типов носителей

HTML5	CSS 2.1	IE	Cr	Op	Sa	Fx
-------	---------	----	----	----	----	----

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>Типы носителей</title>
<style>
@media screen { /* Стил ь для отображения в браузере */
    BODY {
        font-family: Arial, Verdana, sans-serif; /* Рубленый шрифт */
        font-size: 90%; /* Размер шрифта */
        color: #000080; /* Цвет текста */
    }
    H1 {
        background: #faf0e6; /* Цвет фона */
        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 ь для печати */
    BODY {
        font-family: Times, 'Times New Roman', serif; /* Шрифт с засечками */
    }
    H1, H2, P {
```



```

    color: black; /* Чёрный 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. 4.1 и рис. 4.2.

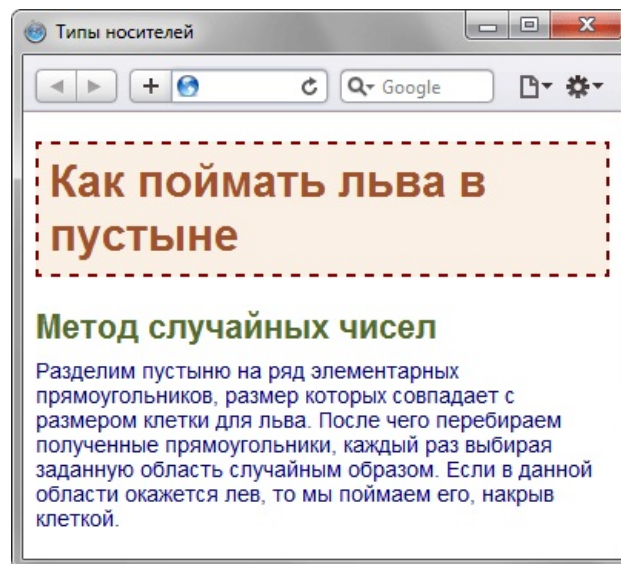


Рис. 4.1. Страница для отображения в окне браузера

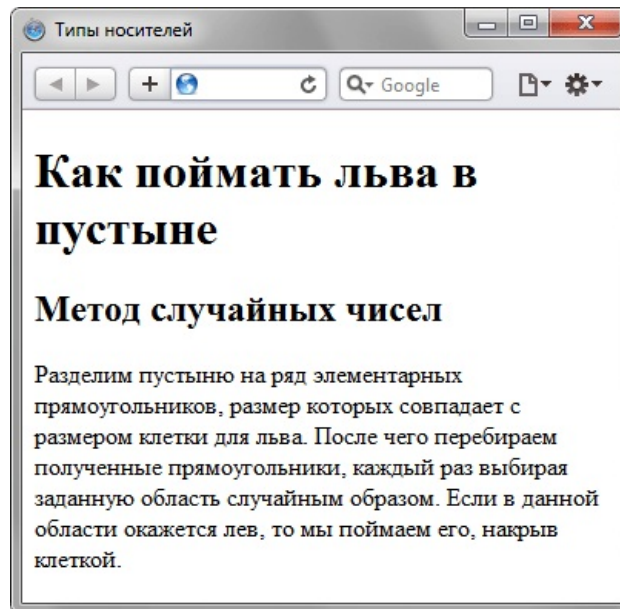


Рис. 4.2. Страница, предназначенная для печати

Просмотреть документ, у которого CSS установлен как тип **print** можно, если распечатать определенную страницу или воспользовавшись предварительным просмотром в браузере (Файл > Предварительный просмотр). Или пойти на хитрость и временно заменить **print** на **screen**, чтобы отобразить итог в браузере. Именно так был получен рис. 4.2.

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. 4.1.

В примере 4.3 показано, как создавать ссылки на CSS-файлы, которые предназначены для разных типов носителей.

Пример 4.3. Подключение стилей для разных носителей

HTML5 CSS 2.1 IE Cr Op Sa Fx

```
<!DOCTYPE HTML>
<html>
<head>
  <meta charset="utf-8">
  <title>Разные носители</title>
  <link media="print, handheld" rel="stylesheet" href="print.css">
  <link media="screen" rel="stylesheet" href="main.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 определённых устройств.

Аналогично можно использовать и глобальные стили, добавляя атрибут **media** к тегу **<style>** (пример 4.4).

```
<!DOCTYPE HTML>
<html>
  <head>
    <meta charset="utf-8">
    <title>Разные носители</title>
    <style media="handheld">
      BODY {
        width: 320px; /* Ширина страницы */
      }
    </style>
  </head>
  <body>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam
    nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat
    volutpat. </p>
  </body>
</html>
```

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

Вопросы для проверки

1. Паша решил для своего сайта сделать версию для печати. Какую строку ему следует использовать в коде?

1. <link media="printer" rel="stylesheet" href="print.css">
2. @import "palm.css" print;
3. @import url("printer.css") printer;
4. @media "palm.css" print;
5. <style media="print">

2. В какой момент подключается стиль для принтера?

1. Во время печати документа.
2. Сразу после загрузки страницы.
3. Как только браузер найдёт в коде подходящий стиль или ссылку на стилевой файл.
4. После обнаружения компьютером принтера.
5. После того, как принтер сообщит браузеру о своем наличии.

3. В какой строке кода содержится ошибка?

```
@media hanheld {
BODY {
color: #080;
background: #ffe;
}
```

1. @media hanheld {
2. BODY {
3. color: #080;
4. background: #ffe;
5. }

Ответы

1. `@import "palm.css" print;`
2. Во время печати документа.
3. `}`

Базовый синтаксис CSS

Как уже было отмечено ранее, стилевые правила записываются в своём 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 параметры будут применяться к тегу `<table>`, затем идут фигурные скобки, в которых записывается стилевое свойство, а его значение указывается после двоеточия. Силевые 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 5.1 показаны две разновидности оформления селекторов и их правил.

Пример 5.1. Использование стилей

HTML5 CSS 2.1 IE Cr Op Sa Fx

```
<!DOCTYPE HTML>
<html>
  <head>
    <meta charset="utf-8">
    <title>Заголовки</title>
    <style>
      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 счёт активного использования пробелов и переносов строк. Так что в любом случае способ оформления стилевых параметров 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 5.2.

Пример 5.2. 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 5.3).

Пример 5.3. 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 задаётся 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 5.4).

Пример 5.4. Разные значения у одного свойства

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

В данном примере для селектора **p** цвет текста вначале установлен зелёным, 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 случайно, 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 объём 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 5.5).

Пример 5.5. Комментарии в CSS-файле

```
/*
  Стиль для сайта htmlbook.ru
*/
```

```

    Сделан для ознакомительных целей
*/

div {
    width: 200px; /* Ширина блока */
    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.

Вопросы для проверки

1. Люба подключила к HTML-документу одновременно два стилевых файла — style1.css и style2.css. Причём в файле style2.css первой строкой импортируется еще один файл с именем style3.css. В файле style1.css цвет текста задается красным, в style2.css — синим, а в style3.css — зелёным. Какой цвет текста будет на странице?

1. красный.
2. синий.
3. зелёный.
4. чёрный.
5. установленный в браузере по умолчанию.

2. В какой строке кода содержится ошибка?

1. p { text-align: center; color: #000000 }
2. div { color: red; font-size: 11pt; }
3. title { color: #fc0; margin: 10px; }
4. p { color: green; color; }
5. html { float: left; }

3. Какая ошибка содержится в следующем коде?

```

/* -----
div {
color: #fc0; /* Цвет текста */
margin: 10px; /* Поля вокруг элемента */
float: left /* Обтекание по правому краю */
}
----- */

```

1. Опечатка в тексте комментария.
2. Вложенные комментарии.
3. Нет точки с запятой.
4. Недопустимые значения у стилевых свойств.
5. Лишние переносы в коде.

4. В какой строке содержится корректный синтаксис?

1. body:color=black
2. body{color:black}
3. {body;color:black}
4. {body:color=black}
5. body{color=black}

5. Как правильно вставить комментарий в CSS-файл?

1. ' комментарий
2. // комментарий
3. // комментарий //
4. /* комментарий */
5. <!-- комментарий -->

Ответы

1. синий.
2. p { color: green; color; }
3. Вложенные комментарии.
4. body{color:black}
5. /* комментарий */

Значения стилевых свойств

Всё многообразие значений стилевых свойств может быть сведено к определённому 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 6.1).

Пример 6.1. 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 всё 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 6.2).

Пример 6.2. Числа в качестве значений

HTML5 CSS 2.1 IE Cr Op Sa Fx

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>Числа</title>
<style>
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 её 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 6.3).

Пример 6.3. Процентная запись

HTML5 CSS 2.1 IE Cr Op Sa Fx

```
<!DOCTYPE HTML>
```

```

<html>
<head>
  <meta charset="utf-8">
  <title>Ширина в процентах</title>
  <style>
    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%.

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. 6.1 перечислены основные относительные единицы.

Табл. 6.1. 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 6.4 показано применение пикселей и **em** для задания размера шрифта.

Пример 6.4. Использование относительных единиц

HTML5 CSS 2.1 IE Cr Op Sa Fx

```

<!DOCTYPE HTML>
<html>
<head>
  <meta charset="utf-8">
  <title>Относительные единицы</title>
  <style>
    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. 6.1).

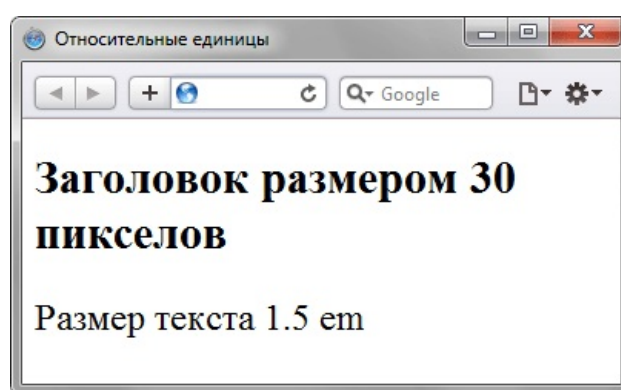


Рис. 6.1. Размер текста при различных единицах

Абсолютные единицы

Абсолютные единицы применяются реже, чем относительные и обычно при работе с текстом. В табл. 6.2 перечислены основные абсолютные единицы.

Табл. 6.2. 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 6.5 показано использование пунктов и миллиметров.

Пример 6.5. Использование абсолютных единиц

HTML5 CSS 2.1 IE Cr Op Sa Fx

```

<!DOCTYPE HTML>
<html>

```

```

<head>
  <meta charset="utf-8">
  <title>Абсолютные единицы</title>
  <style>
    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. 6.2).

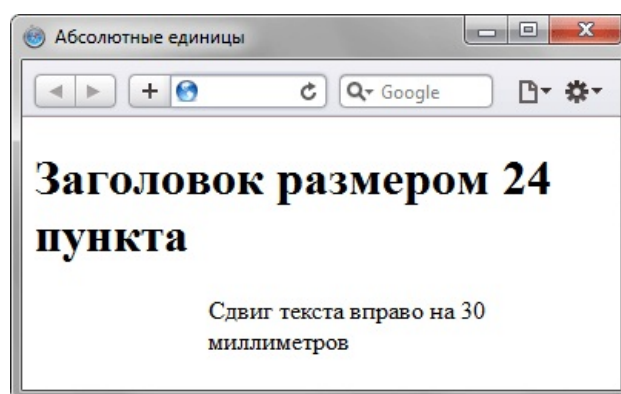


Рис. 6.2. Размер текста при различных единицах

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 тремя способами: по шестнадцатеричному значению, по названию и в формате RGB.

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 — зелёную, а два последних — синюю. Допускается использовать сокращенную форму вида #rgb, где каждый символ следует удваивать (#rrggbb). К примеру, запись #fe0 расценивается как #ffee00.

По названию

Браузеры поддерживают некоторые цвета по их названию. В табл. 6.3 приведены названия,

шестнадцатеричный код и описание.

Табл. 6.3. Названия цветов

Имя	Цвет	Код	Описание
white		#ffffff или #fff	Белый
silver		#c0c0c0	Серый
gray		#808080	Тёмно-серый
black		#000000 или #000	Чёрный
maroon		#800000	Тёмно-красный
red		#ff0000 или #f00	Красный
orange		#ffa500	Оранжевый
yellow		#ffff00 или #ff0	Жёлтый
olive		#808000	Оливковый
lime		#00ff00 или #0f0	Светло-зелёный
green		#008000	Зелёный
aqua		#00ffff или #0ff	Голубой
blue		#0000ff или #00f	Синий
navy		#000080	Тёмно-синий
teal		#008080	Сине-зелёный
fuchsia		#ff00ff или #f0f	Розовый
purple		#800080	Фиолетовый

С помощью RGB

Можно определить цвет, используя значения красной, зелёной и синей составляющей в десятичном исчислении. Значение каждого из трех цветов может принимать значения от 0 до 255. Также можно задавать цвет в процентном отношении. Вначале указывается ключевое слово **rgb**, а затем в скобках, через запятую указываются компоненты цвета, например **rgb(255, 0, 0)** или **rgb(100%, 20%, 20%)**.

В примере 6.6 представлены различные способы задания цветов элементов веб-страниц.

Пример 6.6. Представление цвета

HTML5 CSS 2.1 IE Cr Op Sa Fx

```
<!DOCTYPE HTML>
<html>
<head>
  <meta charset="utf-8">
  <title>Цвета</title>
  <style>
    BODY {
      background-color: #3366CC; /* Цвет фона веб-страницы */
    }
    H1 {
```

```

background-color: RGB(249, 201, 16); /* Цвет фона под заголовком */
}
P {
background-color: maroon; /* Цвет фона под текстом абзаца */
color: white; /* Цвет текста */
}
</style>
</head>
<body>
<h1>Lorem ipsum dolor sit amet</h1>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit,
sed diam nonummy nibh euismod tincidunt ut laoreet dolore
magna aliquam erat volutpat.</p>
</body>
</html>

```

Результат данного примера показан на рис. 6.3.



Рис. 6.3. Цвета на веб-странице

Адреса

Адреса (URI, Uniform Resource Identifiers, унифицированный идентификатор ресурсов) 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 6.7).

Пример 6.7. Адрес графического файла

HTML5 CSS 2.1 IE Cr Op Sa Fx

```

<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>Добавление фона</title>
<style>
body {
background: url('http://webimg.ru/images/156_1.png') no-repeat;
}
div {
background: url(images/warning.png) no-repeat;
padding-left: 20px;
margin-left: 200px;
}
</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"; }
```

Вопросы для проверки

1. В какой строке содержится ошибка?

1. font-size: 20px
2. font-size: 0
3. font-size: 1,5em
4. font-size: 5mm
5. font-size: 300ex

2. Какое выражение написано корректно?

1. color: #fco
2. width: "auto"
3. font-size: blue
4. bakground: red
5. border: none

3. Какой размер в пунктах будет у текста `<p>Пример текста</p>`, если на странице задан следующий стиль?

```
BODY { font-size: 24pt; }
P { font-size: 50%; }
SPAN { font-size: 1.5em; }
```

1. 48pt
2. 36pt
3. 24pt
4. 18pt
5. 12pt

4. Ане хочется установить сиреневый цвет фона веб-страницы. Какое значение свойства **background** подойдёт лучше всего?

1. #cbd1e8
2. rgb(121, 232, 47)
3. #33f
4. #728574
5. rgb(205%, 85%, 53%)

5. Какая ошибка содержится в следующем стиле?

```
img { float: left; border-width: 3; display: block }
```

1. Не хватает точки с запятой в конце записи.
2. Не указаны единицы измерения свойства border-width.
3. Значения left и block написаны без кавычек.
4. Свойства border-width не существует.
5. Значение block свойства display не допускается применять к изображениям.

Ответы

1. font-size: 1,5em
2. border: none
3. 18pt
4. #cbd1e8
5. Не указаны единицы измерения свойства border-width.

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 7.1).

Пример 7.1. Изменение стиля тега абзаца

HTML5	CSS 2.1	IE	Cr	Op	Sa	Fx
-------	---------	----	----	----	----	----

```
<!DOCTYPE HTML>
<html>
  <head>
    <meta charset="utf-8">
    <title>Селекторы тегов</title>
    <style>
      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 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>`.

Вопросы для проверки

1. В какой строке содержится ошибка?

1. H1 { margin-left: 20px; }
2. p { margin-left: 20px; padding-left: 20px; }
3. h2 { margin-right: 20px; }
4. head { color: #rob; }
5. body { font-size: 11pt; color: #aaa; }

2. Таня для фона веб-страницы и цвета текста выбрала цвета `#ffe9f2` и `#6e143b` и в стилях использовала следующий код, однако нужные цвета не проявились. В чем причина?

```
body {
background-color: #ffe9f2
color: #6e143b
```

```
}
```

1. body написан строчными буквами.
2. Свойство background-color неверное, следует писать background.
3. Значения цветов указаны неправильно.
4. В качестве селектора применять body некорректно.
5. Не хватает точки с запятой.

3. Какая строка написана правильно?

1. <P> { color: #333; }
2. P { color: #333; }
3. P: { color: #333; }
4. P { color: 333; }
5. P { color: #3333; }

4. К какому селектору следует применить свойство margin, чтобы изменить отступы на веб-странице?

1. !DOCTYPE
2. A
3. HEAD
4. TITLE
5. BODY

5. Как добавить фоновый цвет ко всем элементам <H1>?

1. h1 { background-color: white }
2. h1.all { background-color: white }
3. h1:all { background-color: white }
4. h1[all] { background-color: white }
5. h1#all { background-color: white }

Ответы

1. head { color: #rob; }
2. Не хватает точки с запятой.
3. P { color: #333; }
4. BODY
5. h1 { background-color: white }

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 определённым классом, к тегу добавляется атрибут `class="Имя класса"` (пример 8.1).

Пример 8.1. Использование классов

HTML5	CSS 2.1	IE	Cr	Op	Sa	Fx
-------	---------	----	----	----	----	----

```
<!DOCTYPE HTML>
<html>
<head>
  <meta charset="utf-8">
  <title>Классы</title>
  <style>
    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 спектральному 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. 8.1.

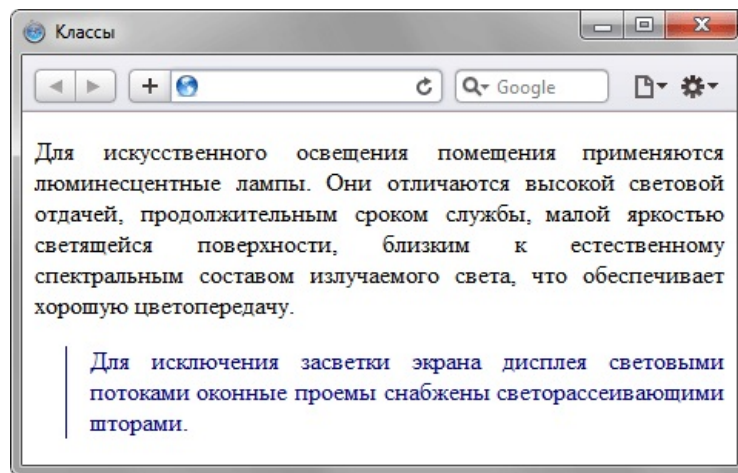


Рис. 8.1. Вид текста, оформленного с помощью стилевых классов

Первый абзац выровнен по ширине с текстом чёрного цвета (этот цвет задаётся 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 8.2).

Пример 8.2. Использование классов

HTML5 CSS 2.1 IE Cr Op Sa Fx

```
<!DOCTYPE HTML>
<html>
<head>
  <meta charset="utf-8">
  <title>Классы</title>
  <style>
    .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">предельным 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. 8.2.

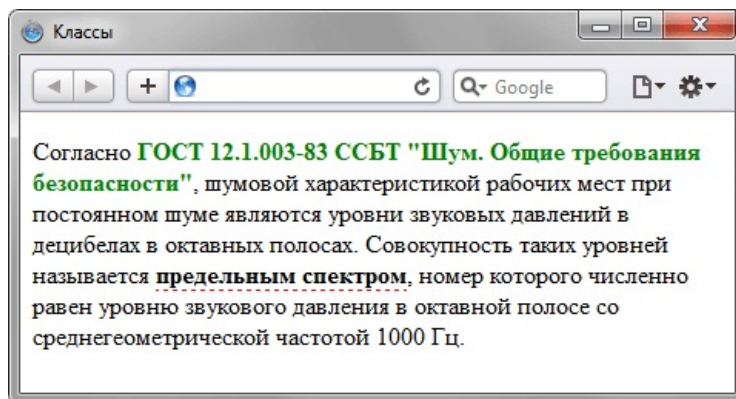


Рис. 8.2. Вид тегов, оформленных с помощью классов

Классы удобно использовать, когда нужно применить стиль к разным элементам веб-страницы: ячейкам таблицы, ссылкам, абзацам и др. В примере 8.3 показано изменение цвета фона строк таблицы для создания «зебры».

Пример 8.3. Использование классов

HTML5 CSS 2.1 IE Cr Op Sa Fx

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>Камни</title>
<style>
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>
```

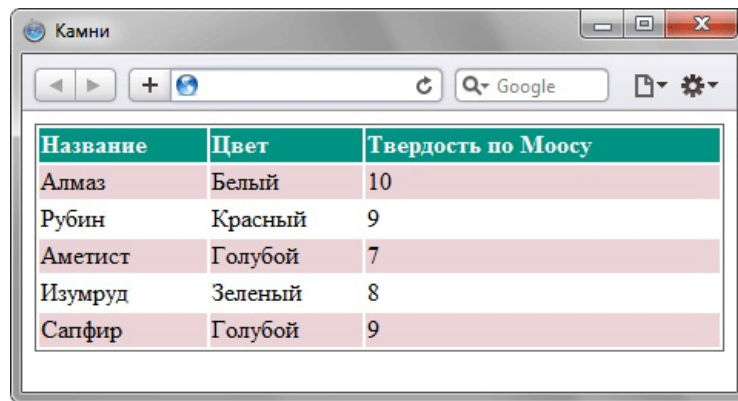


```

    </tr>
  </table>
</body>
</html>

```

Результат данного примера показан на рис. 8.3. В примере класс с именем `odd` используется для изменения цвета фона строки таблицы. За счёт 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. 8.3. Результат применения классов

Одновременное использование разных классов

К любому тегу одновременно можно добавить несколько классов, перечисляя их в атрибуте `class` через пробел. 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 берётся значение у класса, который описан в коде ниже.

В примере 8.4 показано использование разных классов для создания облака тегов.

Пример 8.4. Сочетание разных классов

HTML5 CSS 2.1 IE Cr Op Sa Fx

```

<!DOCTYPE HTML>
<html>
  <head>
    <meta 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>
    </div>
  </body>
</html>

```

```

<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. 8.4.

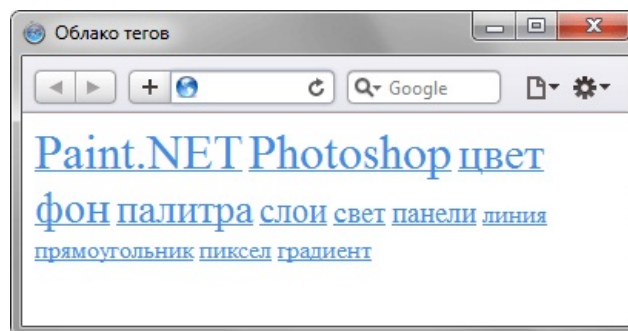


Рис. 8.4. Облако тегов

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`.

Вопросы для проверки

1. Какое имя класса написано правильно?

1. 2layer1
2. 1layer
3. Яndex
4. pink-floyd
5. 28_days_later

2. Какой цвет будет у слова «потока» в коде?

```

<p class="c1">Коэффициент использования излучаемого светильниками <span
class="c2 c3">потока</span>, на расчетной плоскости.</p>

```

При использовании следующего стиля?

```

BODY { color: red; }
P { color: green; }
.c1 {color: blue; }
.c2 { color: yellow; }
.c3 { color: orange; }
.c2.c3 { color: black; }

```

1. Зелёный.
2. Синий.
3. Жёлтый.

4. Оранжевый.
5. Чёрный.

3. Как задать стиль у тега `<div class="iddqd">DOOM</div>`?

1. `div[iddqd] { color: red; }`
2. `div.iddqd { color: red; }`
3. `iddqd.div { color: red; }`
4. `div#iddqd { color: red; }`
5. `div=iddqd { color: red; }`

4. Какое имя класса следует добавить к тегу `<P>`, чтобы текст был одновременно жирным и красного цвета, если имеется следующий стиль?

```
s1 { color: red; font-weight: bold; }  
.s2 { color: red; }  
.s3 { background-color: red; font-weight: bold; }  
.s4 { font-weight: bold; }  
.s5 { font: red bold; }
```

1. s1
2. s2
3. s3
4. s2 s4
5. s5

Ответы

1. pink-floyd
2. Чёрный.
3. `div.iddqd { color: red; }`
4. s2 s4

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 решётки (#), 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 ключевого слова у тега используется атрибут **id**, значением которого выступает имя идентификатора (пример 9.1). Символ решётки при этом уже не указывается.

Пример 9.1. Использование идентификатора

HTML5 CSS 2.1 IE Cr Op Sa Fx

```
<!DOCTYPE HTML>
<html>
<head>
  <meta charset="utf-8">
  <title>Идентификаторы</title>
  <style>
    #help {
      position: absolute; /* Абсолютное позиционирование */
      left: 160px; /* Положение элемента от левого края */
      top: 50px; /* Положение от верхнего края */
      width: 225px; /* Ширина блока */
      padding: 5px; /* Поля вокруг текста */
      background: #f0f0f0; /* Цвет фона */
    }
  </style>
</head>
<body>
  <div id="help">
    Этот элемент 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** (рис. 9.1).

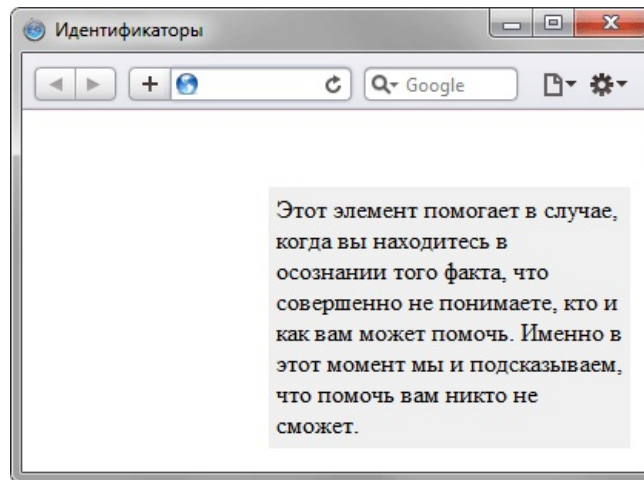


Рис. 9.1. Результат применения идентификатора

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 решётки и название идентификатора. В примере 9.2 показано использование идентификатора применительно к тегу `<p>`.

Пример 9.2. Идентификатор совместно с тегом

HTML5 CSS 2.1 IE Cr Op Sa Fx

```
<!DOCTYPE HTML>
<html>
  <head>
    <meta charset="utf-8">
    <title>Идентификаторы</title>
    <style>
      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>
  </head>
  <body>
    <p>Обычный параграф</p>
    <p id="opa">Параграф необычный</p>
  </body>
</html>
```

Результат данного примера показан на рис. 9.2.

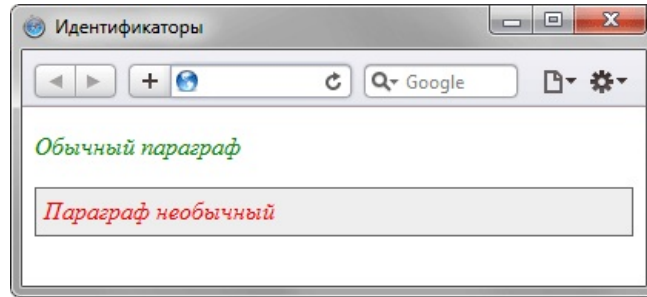


Рис. 9.2. Вид текста после применения стиля

В данном примере вводится стиль для тега `<p>` и для такого же тега, но с указанием идентификатора `ора`.

Вопросы для проверки

1. В каких ситуациях имена идентификаторов и классов можно называть одинаково?

1. Никогда, это недопустимо.
2. В любом случае.
3. Только, если они применяются к одному элементу.
4. Только, если они применяются к разным элементам.
5. Только, если в коде они встречаются один раз.

2. Какое имя идентификатора написано неправильно?

1. id_1id1
2. a-a-a-1-1-1
3. L0g0
4. bla-bla
5. krevedko

3. Какая ошибка содержится в следующем коде?

```
<div class="frame1">
<div id="_nav"><a href="209.html">Подключение к MySQL через PHP</a></div>
<div id="_nav"><a href="213.html">Создание таблиц в phpMyAdmin</a></div>
<div id="_nav"><a href="211.html">Структура базы данных</a></div>
</div>
```

1. Имя класса написано неверно.
2. Имена идентификаторов написаны неверно.
3. Неправильное вложение тегов.
4. Повторяющиеся идентификаторы.
5. Разные идентификаторы для однотипных элементов.

4. Как корректно задать стиль для тега `<div>` с идентификатором `loom`?

1. `loom { font-size: bold; }`
2. `div { font-size: bold; }`
3. `.loom { font-size: bold; }`
4. `#loom# { font-size: bold; }`
5. `#loom { font-size: bold; }`

Ответы

1. В любом случае.
2. krevedko
3. Повторяющиеся идентификаторы.
4. #loom { font-size: bold; }

Контекстные селекторы

При создании веб-страницы часто приходится вкладывать одни теги внутрь других. Чтобы стили для этих тегов использовались корректно, помогут селекторы, которые работают только в определённом 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 10.1.

Пример 10.1. Контекстные селекторы

HTML5 CSS 2.1 IE Cr Op Sa Fx

```
<!DOCTYPE HTML>
<html>
<head>
  <meta charset="utf-8">
  <title>Контекстные селекторы</title>
  <style>
    P B {
      font-family: Times, serif; /* Семейство шрифта */
      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. 10.1.

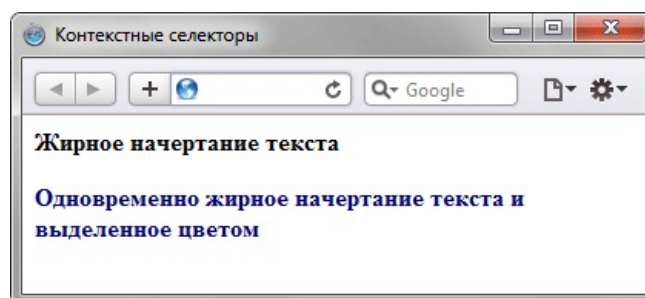


Рис. 10.1. Оформление текста в зависимости от вложенности тегов

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 определённого класса, как показано в примере 10.2.

Пример 10.2. Использование классов

HTML5 CSS 2.1 IE Cr Op Sa Fx

```
<!DOCTYPE HTML>
<html>
<head>
  <meta charset="utf-8">
  <title>Контекстные селекторы</title>
  <style>
    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. 10.2.

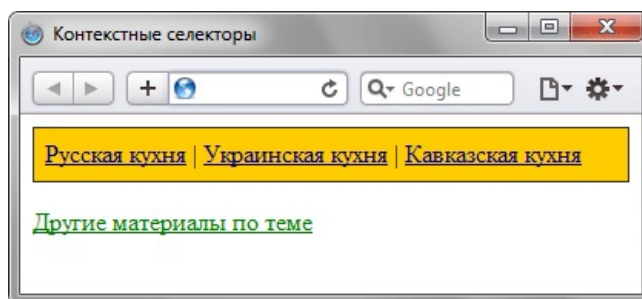


Рис. 10.2. Ссылки разных цветов

В данном примере используется два типа ссылок. Первая ссылка, стиль которой задаётся 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.

Вопросы для проверки

1. Какой цвет будет у текста списка в следующем коде?

```
<!DOCTYPE HTML>
<html>
<head>
  <meta charset="utf-8">
  <title>Контекстные селекторы</title>
  <style>
    UL LI UL { color: green; }
    UL UL { color: red; }
    LI SPAN { color: blue; }
    LI LI { color: fuchsia; }
    UL SPAN { color: orange; }
  </style>
</head>
<body>
<ul>
  <li>
    <ul>
      <li><span>Первый</span></li>
      <li><span>Второй</span></li>
      <li><span>Третий</span></li>
    </ul>
  </li>
</ul>
</body>
</html>
```

1. Зелёный.
2. Красный.
3. Синий.
4. Розовый.
5. Оранжевый.

2. В коде выше какого цвета будут маркеры перед текстом?

1. Зелёного.
2. Красного.
3. Синего.
4. Розового.
5. Оранжевого.

Ответы

1. Оранжевый.
2. Розового.

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 11.1 показана структура взаимодействия тегов между собой.

Пример 11.1. Использование соседних селекторов

HTML5 CSS 2.1 IE Cr Op Sa Fx

```
<!DOCTYPE HTML>
<html>
  <head>
    <meta charset="utf-8">
    <title>Соседние селекторы</title>
    <style>
      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. 11.1.

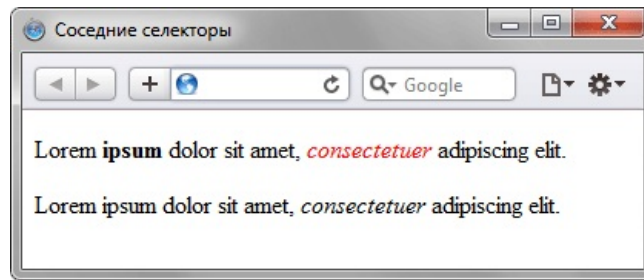


Рис. 11.1. Выделение текста цветом при помощи соседних селекторов

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 стилевой класс и применяют его к абзацу, таким способом можно легко изменить вид текста. Но мы пойдем другим путём и воспользуемся соседними селекторами. Для выделения замечаний создадим новый класс, назовём его `sic`, и станем применять его к тегу `<h2>`. Первый абзац после такого заголовка выделяется цветом фона и отступом (пример 11.2). Вид остальных абзацев останется неизменным.

Пример 11.2. Изменение стиля абзаца

HTML5 CSS 2.1 IE Cr Op Sa Fx

```
<!DOCTYPE HTML>
<html>
<head>
  <meta charset="utf-8">
  <title>Изменение стиля абзаца</title>
  <style>
    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. Учитывая, 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
завершенной.</p>
</body>
</html>

```

Результат данного примера показан на рис. 11.2.

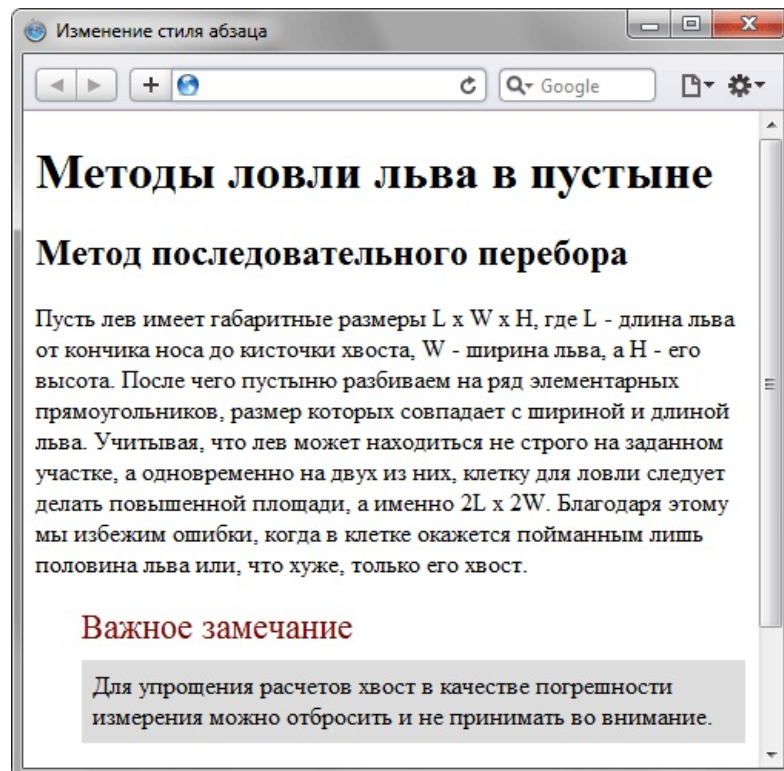


Рис. 11.2. Изменение вида абзаца за счёт использования соседних селекторов

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 с именем `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 11.3 таким манером изменяется величина отступов между указанными тегами.

Пример 11.3. Отступы между заголовками и текстом

HTML5 CSS 2.1 IE Cr Op Sa Fx

```

<!DOCTYPE HTML>
<html>
<head>
  <meta charset="utf-8">
  <title>Соседние селекторы</title>

```

```

<style>
  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 счёт включения отрицательного значения у свойства **margin-top**. При этом текст поднимается вверх, ближе к предыдущему элементу.

Вопросы для проверки

1. Какие теги в данном коде являются соседними?

```

<p><b>Формула серной кислоты:</b><i>H<sub><small>2</small></sub>SO<sub><small>4</small></sub></i></p>

```

1. <P> и <I>
2. и <I>
3. <I> и <SUB>
4. <SUB> и <SMALL>
5. <I> и <SMALL>

2. Имеется следующий код HTML:

```

<p><b>Великая теорема Ферма</b></p>
<p><i>X <sup><small>n</small></sup> + Y <sup><small>n</small></sup> = Z <sup><small>n</small></sup></i></p>
<p>где n - целое число > 2</p>

```

Какой текст выделится красным цветом с помощью стиля **SUP + SUP { color: red; }** ?

1. «X»
2. «Y»
3. «Z»
4. Вторая «n»
5. Вторая и третья «n».

Ответы

1. и <I>
2. Вторая и третья «n».

Дочерние селекторы

Дочерним называется элемент, который непосредственно располагается внутри родительского элемента. Чтобы лучше понять отношения между элементами документа, разберём небольшой код (пример 12.1).

Пример 12.1. Вложенность элементов в документе

HTML5 CSS 2.1 IE Cr Op Sa Fx

```
<!DOCTYPE HTML>
<html>
  <head>
    <meta charset="utf-8">
    <title>Lorem ipsum</title>
  </head>
  <body>
    <div class="main">
      <p><em>Lorem ipsum dolor sit amet</em>, consectetur adipiscing
elit, sed diem 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. 12.1).

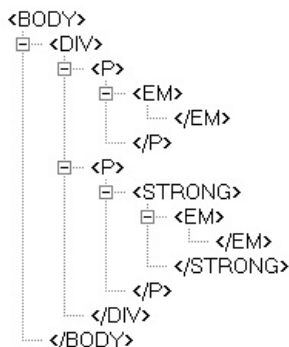


Рис. 12.1. Дерево элементов для примера

На рис. 12.1 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>`.

Вернёмся 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 12.1, 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, разберём следующий код (пример 12.2).

Пример 12.2. Контекстные и дочерние селекторы

HTML5 CSS 2.1 IE Cr Op Sa Fx

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>Дочерние селекторы</title>
<style>
DIV I { /* Контекстный селектор */
  color: green; /* Зеленый цвет текста */
}
P > I { /* Дочерний селектор */
  color: red; /* Красный цвет текста */
}
</style>
</head>
<body>
<div>
<p><i>Lorem ipsum dolor sit amet</i>, consectetur adipiscing
elit, sed diem nonummy nibh euismod tincidunt ut laoreet <i>dolore
magna</i> aliquam erat volutpat.</p>
</div>
</body>
</html>
```

Результат данного примера показан на рис. 12.2.

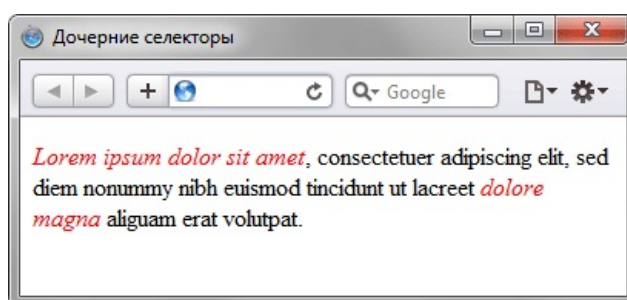


Рис. 12.2. Цвет текста, заданный с помощью дочернего селектора

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 12.3 показано изменение вида списка с помощью стилей. За счёт вложения одного списка в другой получаем разновидность меню. Заголовки при этом располагаются горизонтально, а набор ссылок — вертикально под заголовками (рис. 12.3).

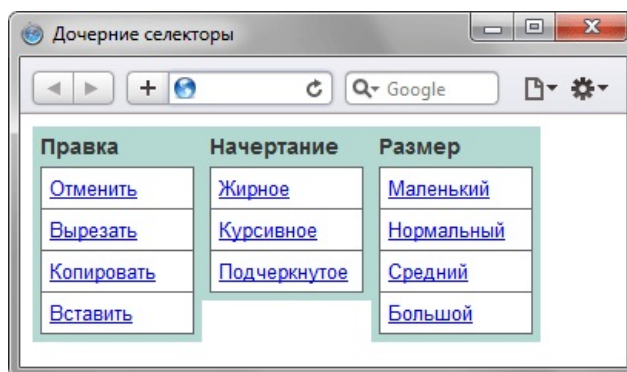


Рис. 12.3. Список в виде меню

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 12.3).

Пример 12.3. Использование дочерних селекторов

HTML5 CSS 2.1 IE Cr Op Sa Fx

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>Дочерние селекторы</title>
<style>
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.

Вопросы для проверки

1. Какой цвет будет у жирного курсивного текста в коде

```

<p>Нормы освещённости построены на <b><i>основе классификации зрительных работ</i></b> по определенным количественным признакам.</p>

```

При использовании следующего стиля?

```

P { color: green; }
B {color: blue; }
I {color: orange; }
B > I { color: olive; }
P > I { color: yellow; }

```

1. Зелёный.
2. Синий.
3. Оранжевый.
4. Оливковый.

5. Жёлтый.

2. Какой элемент является родительским для тега <TITLE>?

1. <HEAD>
2. <BODY>
3. <HTML>
4. <META>
5. <!DOCTYPE>

3. Для какого тега элемент <!DOCTYPE> выступает родителем?

1. <HTML>
2. <TITLE>
3. <BODY>
4. <HEAD>
5. Ни для одного тега.

Ответы

1. Оливковый.
2. <HEAD>
3. Ни для одного тега.

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 счёт 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 определённого 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 13.1 показано изменение стиля тега `<q>`, в том случае, если к нему добавлен атрибут `title`.

Пример 13.1. Вид элемента в зависимости от его атрибута

HTML5	CSS 2.1	IE	Cr	Op	Sa	Fx
-------	---------	----	----	----	----	----

```
<!DOCTYPE HTML>
<html>
  <head>
    <meta charset="utf-8">
    <title>Селекторы атрибутов</title>
    <style>
      Q {
        font-style: italic; /* Курсивное начертание */
        quotes: "\00AB" "\00BB"; /* 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. 13.1.

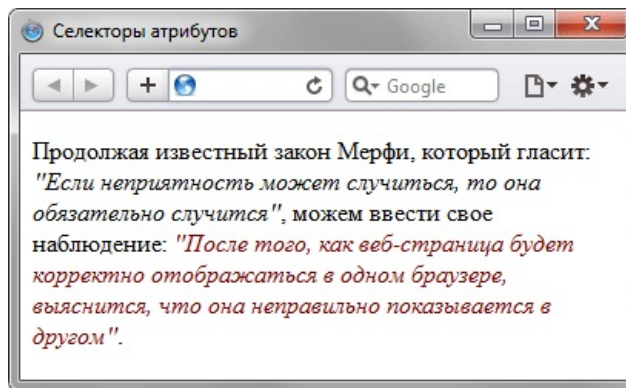


Рис. 13.1. Изменение стиля элемента в зависимости от применения атрибута `title`

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 определённое 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 определённым селекторам.

В примере 13.2 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 13.2. Стиль для открытия ссылок в новом окне

HTML5 CSS 2.1 IE Cr Op Sa Fx

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>Селекторы атрибутов</title>
<style>
A[target="_blank"] {
/* Параметры фонового рисунка */
background: url(images/blank.png) 0 6px no-repeat;
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. 13.2).

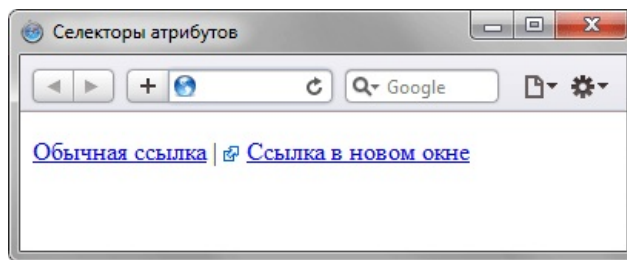


Рис. 13.2. Изменение стиля элемента в зависимости от значения *target*

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 определённого 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 начинаются с указанного текста. А во втором — только к определённым селекторам. Использование кавычек не обязательно, но только если значение содержит латинские буквы и без пробелов.

Предположим, что на сайте требуется разделить стиль обычных и внешних ссылок — ссылки, которые ведут на другие сайты. Чтобы не вводить в тег `<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 13.3.

Пример 13.3. Изменение стиля внешней ссылки

HTML5 CSS 2.1 IE Cr Op Sa Fx

```
<!DOCTYPE HTML>
<html>
<head>
  <meta charset="utf-8">
  <title>Селекторы атрибутов</title>
  <style>
    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. 13.3).

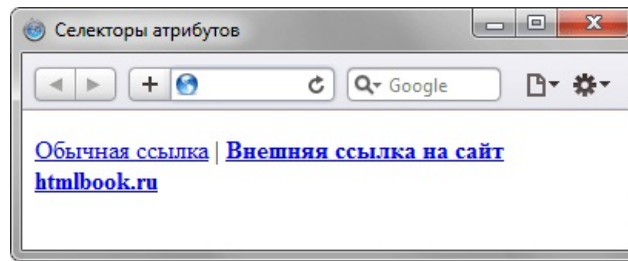


Рис. 13.3. Изменение стиля для внешних ссылок

В данном примере внешние ссылки выделяются жирным начертанием. Также можно воспользоваться показанным в примере 13.2 приёмом и добавлять к ссылке небольшое изображение, которое будет сообщать, что ссылка ведёт на другой сайт.

Значение атрибута оканчивается определённым 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 определённым селекторам.

Таким способом можно автоматически разделять стиль для ссылок на сайты домена ru и для ссылок на сайты других доменов вроде com, как показано в примере 13.4.

Пример 13.4. Стиль для разных доменов

HTML5 CSS 2.1 IE Cr Op Sa Fx

```
<!DOCTYPE HTML>
<html>
<head>
  <meta charset="utf-8">
  <title>Селекторы атрибутов</title>
  <style>
    A[href$=".ru"] { /* Если ссылка заканчивается на .ru */
      /* Добавляем фоновый рисунок */
      background: url(images/ru.png) no-repeat 0 6px;
      padding-left: 12px; /* Смещаем текст вправо */
    }
    A[href$=".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. 13.4). 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 селектором, у которого заданный текст встречается в любом месте значения атрибута.

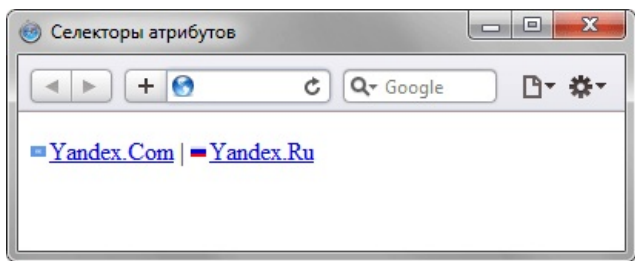


Рис. 13.4. Добавление картинки к ссылкам

Значение атрибута содержит указанный текст

Возможны варианты, когда стиль следует применить к тегу с определённым 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 }
```

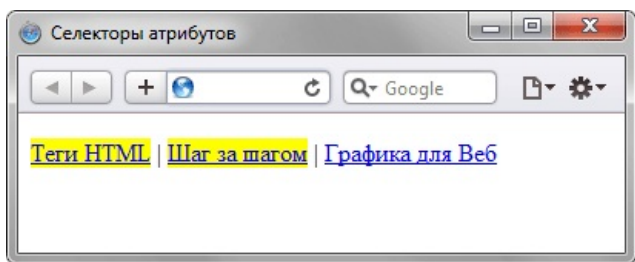
В примере 13.5 показано изменение стиля ссылок, в атрибуте **href** которых встречается слово «htmlbook».

Пример 13.5. Стиль для разных сайтов

HTML5 CSS 2.1 IE Cr Op Sa Fx

```
<!DOCTYPE HTML>  
<html>  
  <head>  
    <meta charset="utf-8">  
    <title>Селекторы атрибутов</title>  
    <style>  
      [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. 13.5.



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 13.6).

Пример 13.6. Стиль в зависимости от имени класса

HTML5 CSS 2.1 IE Cr Op Sa 5 Fx

```
<!DOCTYPE HTML>  
<html>  
  <head>  
    <meta charset="utf-8">  
    <title>Блок</title>  
    <style>  
      [class~="block"] h3 { color: green; }  
    </style>  
  </head>  
  <body>  
    <div class="block tag">  
      <h3>Заголовок</h3>  
    </div>  
  </body>  
</html>
```

В данном примере зелёный 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 идёт дефис (пример 13.7).

Пример 13.7. Дефисы в значениях

HTML5 CSS 2.1 IE Cr Op Sa Fx

```
<!DOCTYPE HTML>  
<html>  
  <head>  
    <meta charset="utf-8">  
    <title>Блок</title>  
    <style>  
      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 }

```

Вопросы для проверки

1. Необходимо задать цвет фона у текстового поля. Какой стиль для этой цели подойдет?

1. `INPUT[type="text"] { background: #acdacc; }`
2. `INPUT[type="textinput"] { background: #acdacc; }`
3. `INPUT[type="textfield"] { background: #acdacc; }`
4. `INPUT[type="textarea"] { background: #acdacc; }`
5. `INPUT[type="texts"] { background: #acdacc; }`

2. Какой стиль необходимо использовать, чтобы изменить цвет текста только у второго абзаца?

```

<p class="text text1-count1-text">Первый абзац</p>
<p class="text text2-count2-text">Второй абзац</p>
<p class="text text3-count3-text">Третий абзац</p>

```

1. `P[class="text2"] { color: red; }`
2. `P[class^="text2"] { color: red; }`
3. `P[class~="text2"] { color: red; }`

4. P[class*="text2"] { color: red; }
5. P[class\$="text2"] { color: red; }

3. К какому элементу будет применяться следующий стиль?

```
[class~="lorem"] { background: #666; }
```

1. <p class="lorem-ipsum">Lorem ipsum dolor sit amet</p>
2. <div class="lorem-ipsum dolor">Lorem ipsum dolor sit amet</div>
3. <p class="ipsum-lorem">Lorem ipsum dolor sit amet</p>
4. <div class="lorem ipsum">Lorem ipsum dolor sit amet</div>
5. <p>Lorem ipsum dolor sit amet</p>

Ответы

1. INPUT[type="text"] { background: #acdacc; }
2. P[class*="text2"] { color: red; }
3. <div class="lorem ipsum">Lorem ipsum dolor sit amet</div>

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 звёздочки (*) 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 14.1 показано одно из возможных приложений универсального селектора — выбор шрифта и размера текста для всех элементов документа.

Пример 14.1. Использование универсального селектора

HTML5 CSS 2.1 IE Cr Op Sa Fx

```
<!DOCTYPE HTML>
<html>
<head>
  <meta charset="utf-8">
  <title>Универсальный селектор</title>
  <style>
    * {
      font-family: Arial, Verdana, sans-serif; /* Шрифт для текста */
      font-size: 96%; /* Размер текста */
    }
  </style>
</head>
<body>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam
  nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam
  erat volutpat.</p>
</body>
</html>
```

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

Вопросы для проверки

1. Какой стиль установит красный цвет текста в абзаце?

1. * HTML P { color: red; }
2. HTML * P { color: red; }
3. P * { color: red; }
4. BODY * P { color: red; }
5. BODY P * { color: red; }

2. Что означает следующая запись в стилях?

```
* DIV * { background: green; }
```

1. Установить фоновый цвет для всех элементов веб-страницы.

2. Задать цвет для всех текстовых элементов документа.
3. Установить фоновый цвет для всех элементов внутри контейнера <DIV>.
4. Установить фоновый цвет только для тегов <DIV>, вложенных в другие контейнеры.
5. Зеленый цвет фона для всех тегов <DIV> в коде.

3. К какому слову применяется стиль селектора div * em * в следующем фрагменте кода?

```
<div>
  <h1><em>Lorem</em> ipsum</h1>
  <p>Lorem ipsum dolor sit amet, <strong>consectetuer</strong>
    adipiscing elit.</p>
  <ul>
    <li><em>Ut</em> wisis enim ad</li>
    <li>Quis <em><span>nostrud</span></em> exerci</li>
    <li>Tution ullamcorper suscipit</li>
  </ul>
  <em>Nisl</em> ut aliquip exea commodo consequat.
</div>
```

1. Lorem
2. consectetuer
3. Ut
4. nostrud
5. Nisl

Ответы

1. HTML * P { color: red; }
2. Установить фоновый цвет для всех элементов внутри контейнера <DIV>.
3. nostrud

Псевдоклассы

Псевдоклассы определяют динамическое состояние элементов, которое изменяется с помощью действий пользователя, а также положение в дереве документа. Примером такого состояния служит текстовая ссылка, которая меняет свой цвет при наведении на неё 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 идёт 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.

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 неё курсор и щёлкнуть мышкой. Несмотря на то, что активным может стать практически любой элемент веб-страницы, псевдокласс `:active` используется преимущественно для ссылок.

:link

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

Запись `A {...}` и `A:link {...}` по своему результату равноценна, поскольку в браузере даёт 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 15.1).

Пример 15.1. Применение псевдокласса :focus

HTML5	CSS 2.1	IE	Cr	Op	Sa	Fx
-------	---------	----	----	----	----	----

```
<!DOCTYPE HTML>
<html>
  <head>
```

```

<meta charset="utf-8">
<title>Псевдоклассы</title>
<style>
  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. 15.1). Во второй строке находится курсор, поэтому текстовое поле получило фокус.

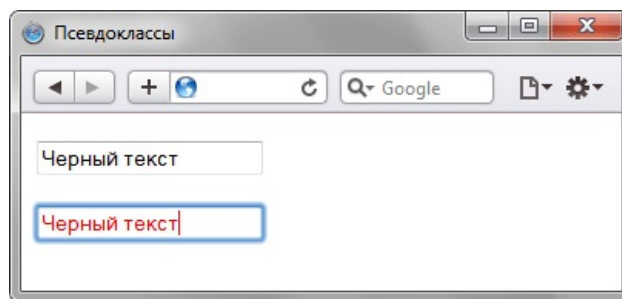


Рис. 15.1. Изменение стиля текста при получении фокуса

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 щёлкнуть в любом месте страницы (кроме текстового поля, естественно), как произойдет потеря фокуса и текст вернётся к первоначальному чёрному 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 посещённым ссылкам. Обычно такая ссылка меняет свой цвет по умолчанию на фиолетовый, но с помощью стилей цвет и другие параметры можно задать самостоятельно (пример 15.2).

Пример 15.2. Изменение цвета ссылок

HTML5 CSS 2.1 IE Cr Op Sa Fx

```

<!DOCTYPE HTML>
<html>
<head>
  <meta charset="utf-8">
  <title>Псевдоклассы</title>

```

```

<style>
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. 15.2.

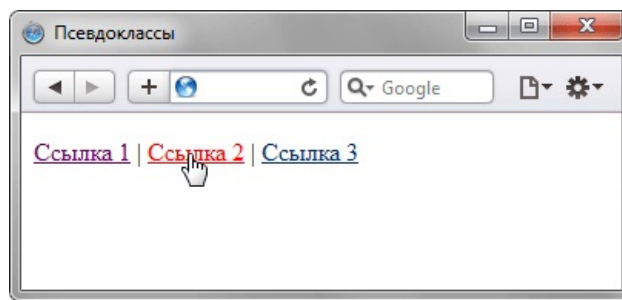


Рис. 15.2. Вид ссылки при наведении на неё курсора мыши

В данном примере показано использование псевдоклассов совместно со ссылками. При этом имеет значение порядок следования псевдоклассов. Вначале указывается **:visited**, а затем идёт **:hover**, в противном случае посещённые 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 и младше 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 15.3 показана таблица, строки которой меняют свой цвет при наведении на них курсора мыши. Это достигается за счёт добавления псевдокласса **:hover** к селектору **TR**.

Пример 15.3. Выделение строк таблицы

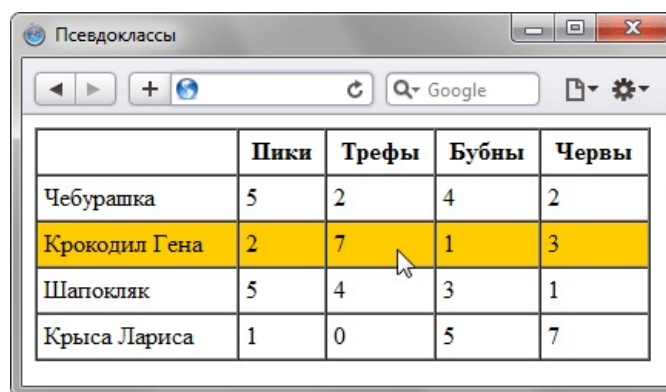
HTML5 CSS 2.1 IE Cr Op Sa Fx


```

<!DOCTYPE HTML>
<html>
<head>
  <meta charset="utf-8">
  <title>Псевдоклассы</title>
  <style>
    table { border-spacing: 0; }
    td { padding: 4px; }
    tr:hover {
      background: #fc0; /* Меняем цвет фона строки таблицы */
    }
  </style>
</head>
<body>
  <table width="400" border="1">
    <tr>
      <th></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. 15.3).



	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. 15.3. 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, разберём небольшой код (пример 15.4).

Пример 15.4. Использование псевдокласса :first-child

HTML5 CSS 2.1 IE Cr Op Sa Fx

```
<!DOCTYPE HTML>
<html>
  <head>
    <meta charset="utf-8">
    <title>Псевдоклассы</title>
    <style type="text/css">
      B:first-child {
        color: red; /* Красный цвет текста */
      }
    </style>
  </head>
  <body>

    <p><b>Lorem ipsum</b> dolor sit amet, <b>consectetuer</b>
    adipiscing <b>elit</b>, sed diem nonummy nibh euismod tincidunt
    ut lacreet dolore magna aliquam erat volutpat.</p>

    <p><b>Ut wisis enim</b> ad minim veniam, <b>quis nostrud</b>
    exerci tution ullamcorper suscipit lobortis nisl ut aliquip
    ex ea <b>commodo consequat</b>.</p>

  </body>
</html>
```

Результат примера показан ниже (рис. 15.4).

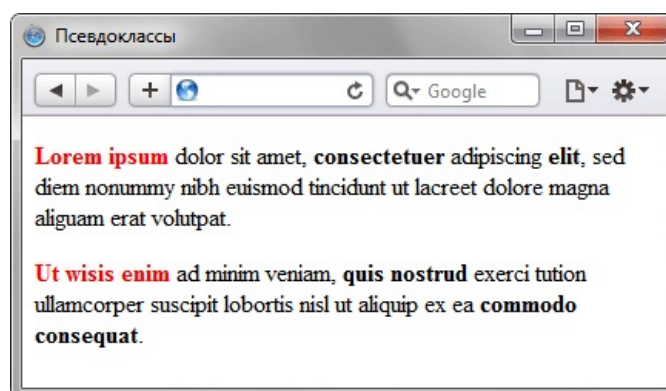


Рис. 15.4. Использование псевдокласса :first-child

В данном примере псевдокласс **:first-child** добавляется к селектору **B** и устанавливает для него красный цвет текста. Хотя контейнер **** встречается в первом абзаце три раза, красным цветом будет выделено лишь первое упоминание, т. е. текст «Lorem ipsum». В остальных случаях содержимое контейнера **** отображается чёрным цветом. Со следующим абзацем всё начинается снова, поскольку родительский элемент поменялся. Поэтому фраза «Ut wisis enim» также будет выделена красным цветом.

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, в некоторых случаях 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 15.5).

Пример 15.5. Отступы для абзаца

HTML5 CSS 2.1 IE Cr Op Sa Fx

```
<!DOCTYPE HTML>
<html>
<head>
  <meta charset="utf-8">
  <title>Псевдоклассы</title>
  <style>
    P {
      text-indent: 1em; /* Отступ первой строки */
    }
    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. 15.5).

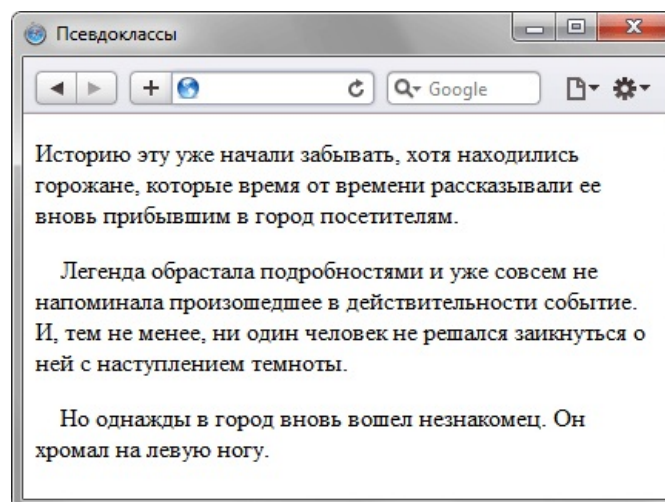


Рис. 15.5. Изменение стиля первого абзаца

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

Псевдоклассы, задающие язык текста

Для документов, одновременно содержащих 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 атрибут **lang**, он обычно добавляется к тегу `<html>`. С помощью псевдокласса **:lang** можно задавать определённые 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 15.6. Вид кавычек в зависимости от языка

HTML5	CSS 2.1	IE	Cr	Op	Sa	Fx
-------	---------	----	----	----	----	----

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>lang</title>
<style>
P {
font-size: 150%; /* Размер текста */
}
q:lang(de) {
/* Вид кавычек для немецкого языка */
quotes: "\201E" "\201C";
}
q:lang(en) {
/* Вид кавычек для английского языка */
quotes: "\201C" "\201D";
}
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. 15.6. 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>`.

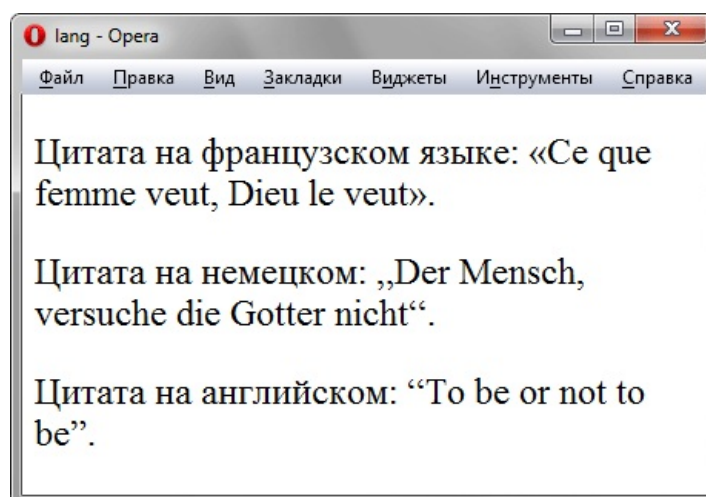


Рис. 15.6. Разные кавычки для разных языков

Вопросы для проверки

1. Олег сделал ссылки, которые меняют цвет фона при наведении на них курсора мыши. Для этого он использовал следующий стиль.

```
A { color: blue; background: orange; }
A:hover { background: yellow; color: black; }
A:visited { color: white; }
A:active { color: red; }
```

Однако некоторые ссылки при наведении на них курсора не меняли свой цвет на чёрный. Почему?

1. Код CSS не валидный.
2. К селектору A не добавлен псевдокласс `:link`.
3. Псевдокласс `:visited` стоит после `:hover`.
4. Псевдокласс `:active` стоит после `:visited`.
5. Неверное значение свойства `color` у `A:hover`.

2. Требуется выделить фоновым цветом первую строку таблицы. Какой псевдокласс для этой цели подойдёт?

1. `:active`
2. `:first-child`
3. `:focus`
4. `:hover`
5. `:lang`

3. К каким элементам добавляет стиль следующая конструкция — `A:link:visited:hover` ?

1. К непосещённым ссылкам.
2. К посещённым ссылкам.
3. К любым ссылкам при наведении на них курсора мыши.
4. К посещённым ссылкам при наведении на них курсора мыши.
5. Ни к одному элементу.

Ответы

1. Псевдокласс `:visited` стоит после `:hover`.
2. `:first-child`
3. Ни к одному элементу.

Псевдоэлементы

Псевдоэлементы позволяют задать стиль элементов не определённых 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 идёт 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}
```

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

Далее перечислены все псевдоэлементы, их описание и свойства.

:after

Применяется для вставки назначенного контента после содержимого элемента. Этот псевдоэлемент работает совместно со стилевым свойством **content**, которое определяет содержимое для вставки. В примере 16.1 показано использование псевдоэлемента **:after** для добавления текста в конец абзаца.

Пример 16.1. Применение :after

HTML5	CSS 2.1	IE 7	IE 8+	Cr	Op	Sa	Fx
-------	---------	------	-------	----	----	----	----

```
<!DOCTYPE HTML>  
<html>  
  <head>  
    <meta charset="utf-8">  
    <title>Псевдоэлементы</title>  
    <style>  
      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. 16.1.

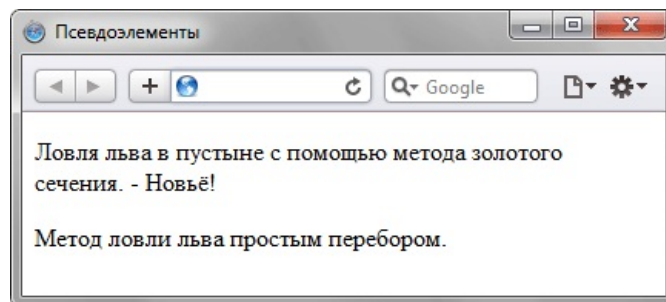


Рис. 16.1. Добавление текста к абзацу с помощью `:after`

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

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

`:before`

По своему действию `:before` аналогичен псевдоэлементу `:after`, но вставляет контент до содержимого элемента. В примере 16.2 показано добавление маркеров своего типа к элементам списка посредством скрытия стандартных маркеров и применения псевдоэлемента `:before`.

Пример 16.2. Использование `:before`

HTML5 CSS 2.1 IE 7 IE 8+ Cr Op Sa Fx 4

```
<!DOCTYPE HTML>
<html>
<head>
  <meta charset="utf-8">
  <title>Псевдоэлементы</title>
  <style>
    UL {
      padding-left: 0; /* Убираем отступ слева */
      list-style-type: none; /* Прячем маркеры списка */
    }
    LI:before {
      /* Добавляем перед элементом списка символ в юникоде */
      content: "\20aa ";
    }
  </style>
</head>
<body>
  <ul>
    <li>Чебурашка</li>
    <li>Крокодил Гена</li>
    <li>Шапокляк</li>
    <li>Крыса Лариса</li>
  </ul>
</body>
</html>
```

Результат примера показан ниже (рис. 16.2).

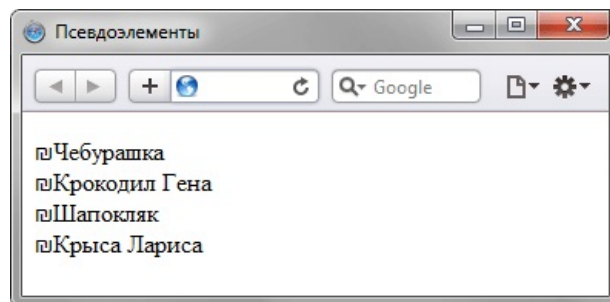


Рис. 16.2. Изменение вида маркеров с помощью `:before`

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

И `:after` и `:before` дают результат только для тех элементов, у которых имеется содержимое, поэтому добавление к селектору `img` или `image` ничего не выведет.

`: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 16.3).

Пример 16.3. Использование `:first-letter`

HTML5 CSS 2.1 IE Cr Op Sa Fx

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>Псевдоэлементы</title>
<style>
P {
/* Гарнитура шрифта основного текста */
font-family: Arial, Helvetica, sans-serif;
font-size: 90%; /* Размер шрифта */
color: black; /* Черный цвет текста */
}
P:first-letter {
/* Гарнитура шрифта первой буквы */
font-family: 'Times New Roman', Times, serif;
font-size: 200%; /* Размер шрифта первого символа */
color: red; /* Красный цвет текста */
}
</style>
</head>
<body>
```

```

<p>Луч фонарика осветил старые скрипучие ступени, по которым
не далее как пять минут назад в дом поднялся Паша. Оля осторожно
приоткрыла дверь и осветила внутрь дома. Луч света, словно
нехотя, пробивался сквозь тугую завесу из мрака и пыли. </p>
<p>Взгляд Оли опустился на пол, и она вскрикнула. В пустом
помещении никого не было, и лишь на полу валялась порванная
туфля Паши.</p>
</body>
</html>

```

Результат примера показан ниже (рис. 16.3).

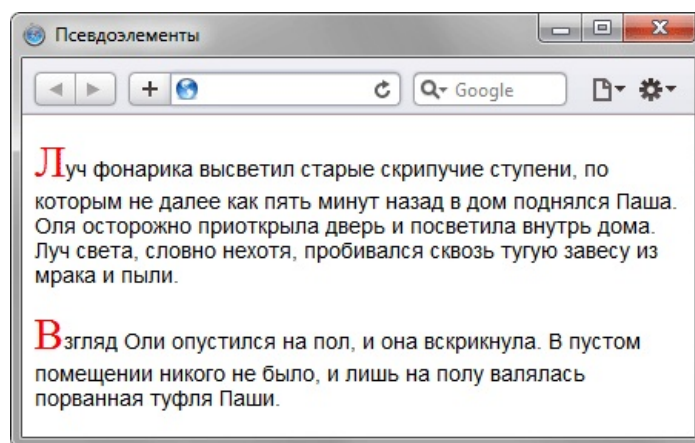


Рис. 16.3. Создание выступающего инициала

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 16.4 показано использование псевдоэлемента **:first-line** применительно к абзацу текста.

Пример 16.4. Выделение первой строки текста

HTML5 CSS 2.1 IE Cr Op Sa Fx

```

<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>Псевдоэлементы</title>
<style>
P:first-line {
color: red; /* Красный цвет текста */
font-style: italic; /* Курсивное начертание */
}
</style>
</head>
<body>
<p>Интересно, а существует ли способ действительно практичного

```

```
применения свойства 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. 16.4.

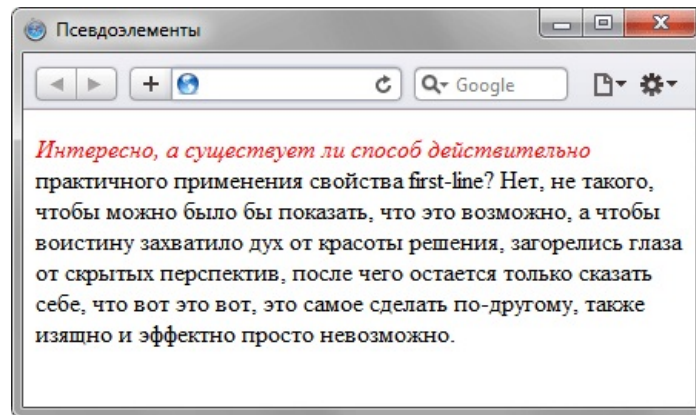


Рис. 16.4. Результат применения псевдоэлемента :first-line

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.

Вопросы для проверки

1. Какой псевдоэлемент позволяет добавить текст в начало предложения?

1. :after
2. :before
3. :first-line
4. :first-text
5. :first-letter

2. Что делает следующий стиль?

```
OL LI:first-letter {  
color: red;  
}
```

1. Изменяет цвет первой буквы элемента маркированного списка.
2. Изменяет цвет первой буквы элемента нумерованного списка.
3. Изменяет цвет первой строки в маркированном списке.
4. Изменяет цвет первой строки в нумерованном списке.
5. Изменяет цвет текста всего списка.

3. Какой селектор написан с ошибкой?

1. p.new:before
2. abbr:first-line
3. p.new.back:after

4. `div:before:first-letter`
5. `a:hover:before`

Ответы

1. `:before`
2. Изменяет цвет первой буквы элемента нумерованного списка.
3. `div:before:first-letter`

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 17.1 показана обычная запись, здесь для каждого селектора приводится свой набор стилевых свойств.

Пример 17.1. 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 17.2.

Пример 17.2. 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.

Вопросы для проверки

1. Какой цвет фона будет у элемента с классом button при включении приведённого стиля ?

```
.bgzapas, .button, h1 {  
  font-size: 1.2em;  
  padding: 10px;  
  background-color: #fcfaed;  
}  
.bgzapas {  
  background-color: #e7f2d7;  
}  
.button, h2 {  
  width: 95px;  
  font-size: 11px;  
  color: #f3fced;  
  background-color: #5ca22e;  
}  
.bgzapas, .button {  
  background-color: #d9d7f2;  
}
```

1. #fcfaed
2. #e7f2d7
3. #f3fced
4. #5ca22e
5. #d9d7f2

Ответы

5. #d9d7f2

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.

Разберём наследование на примере таблицы. Особенностью таблиц можно считать строгую иерархическую структуру тегов. Вначале следует контейнер `<table>` внутри которого добавляются теги `<tr>`, а затем идёт тег `<td>`. Если в стилях для селектора `TABLE` задать цвет текста, то он автоматически устанавливается для содержимого ячеек, как показано в примере 18.1.

Пример 18.1. Наследование параметров цвета

HTML5 CSS 2.1 IE Cr Op Sa Fx

```
<!DOCTYPE HTML>
<html>
<head>
  <meta charset="utf-8">
  <title>Наследование</title>
  <style>
    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` задаёт 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 тёмный, 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 18.2).

Пример 18.2. Параметры текста для всей веб-страницы

HTML5 CSS 2.1 IE Cr Op Sa Fx

```
<!DOCTYPE HTML>
<html>
  <head>
    <meta charset="utf-8">
    <title>Наследование</title>
    <style>
      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 всё-таки изменить цвет для отдельного контейнера. В этом случае придётся переопределять нужные параметры явно, как показано в примере 18.3.

Пример 18.3. Изменение свойств наследуемого элемента

HTML5 CSS 2.1 IE Cr Op Sa Fx

```
<!DOCTYPE HTML>
<html>
  <head>
    <meta charset="utf-8">
    <title>Наследование</title>
    <style>
      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**.

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 некоторые 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 автора.
3. Стиль пользователя.
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.

Как задавать пользовательский стиль рассказывалось в [главе 1](#) (см. рис. 1.3 и 1.4).

!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 стиле пользователя — будет применяться стиль пользователя.
- **!important** содержится в авторском стиле и 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 а) начисляется 100, за каждый класс и псевдокласс (b) начисляется 10, за каждый селектор тега и псевдоэлемент (с) начисляется 1. Складывая указанные значения в определённом порядке, 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 указан в коде ниже.

В примере 19.1 показано, как влияет специфичность на стиль элементов списка.

Пример 19.1. Цвет списка

HTML5 CSS 2.1 IE Cr Op Sa Fx

```
<!DOCTYPE HTML>
<html>
<head>
  <meta charset="utf-8">
  <title>Список</title>
  <style>
    #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 зелёным, 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 10, что явно меньше. Поэтому текст окрашиваться красным цветом не будет. Чтобы исправить ситуацию, необходимо либо понизить специфичность первого селектора,

либо повысить специфичность второго (пример 19.2).

Пример 19.2. 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 счёт убирания идентификатора применяется редко, в основном, повышается специфичность нужного селектора.

Вопросы для проверки

1. Какая специфичность будет у селектора `table.forum tr: hover p`?

1. 14
2. 22
3. 23
4. 32
5. 41

2. Какая специфичность будет у селектора `#catalog .col3 .height div`?

1. 301
2. 203
3. 121
4. 40
5. 31

Ответы

1. 23
2. 121

Валидация CSS

Валидацией называется проверка CSS-кода на соответствие спецификации CSS2.1 или CSS3. Соответственно, корректный код, не содержащий ошибок, называется валидный, 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. 20.1).

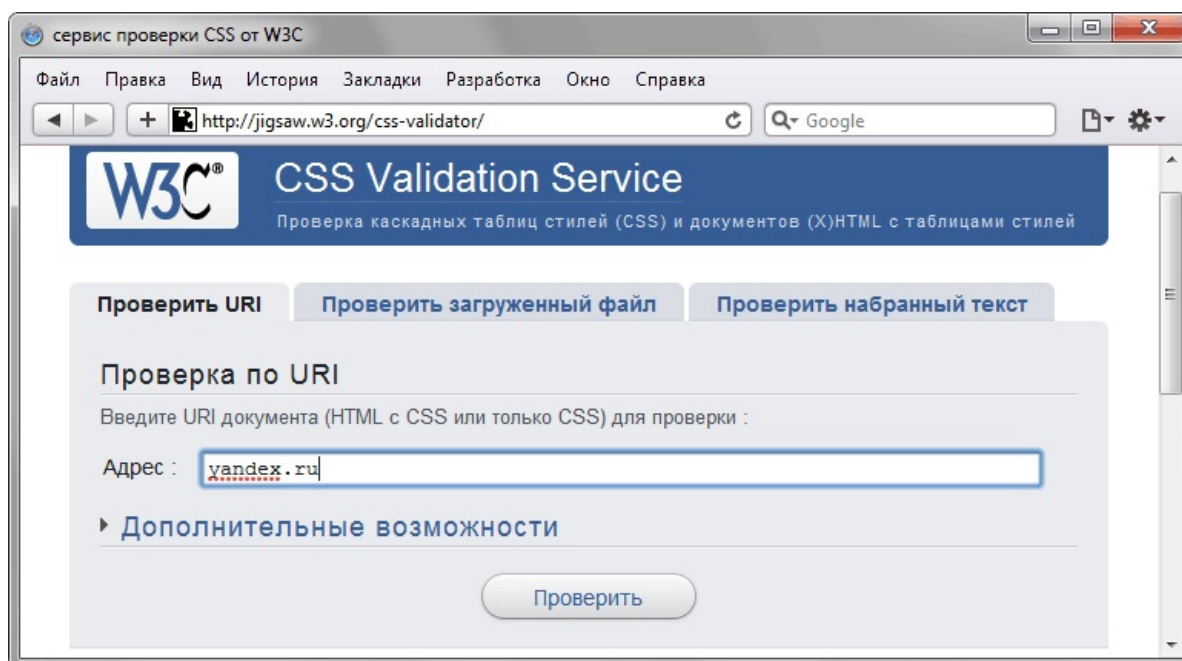


Рис. 20.1. Проверка документа по адресу

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. 20.2).

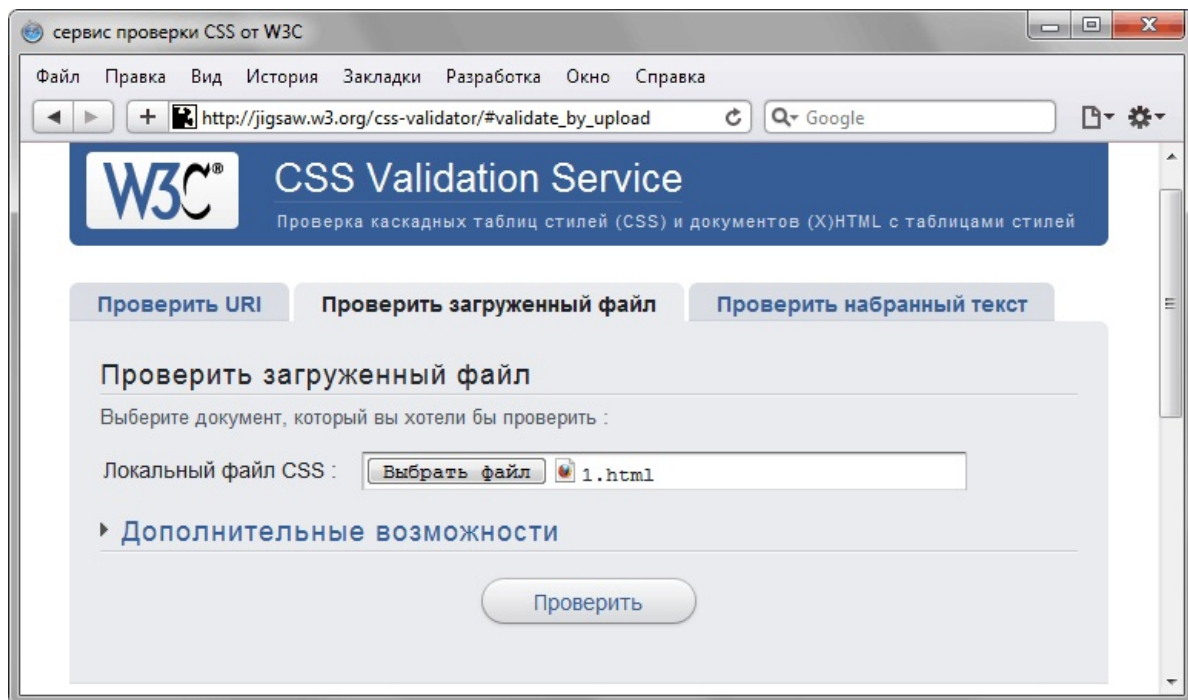


Рис. 20.2. Проверка файла при его загрузке

Сервис автоматически распознает тип файла и если указан HTML-документ, вычленяет 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. 20.3).

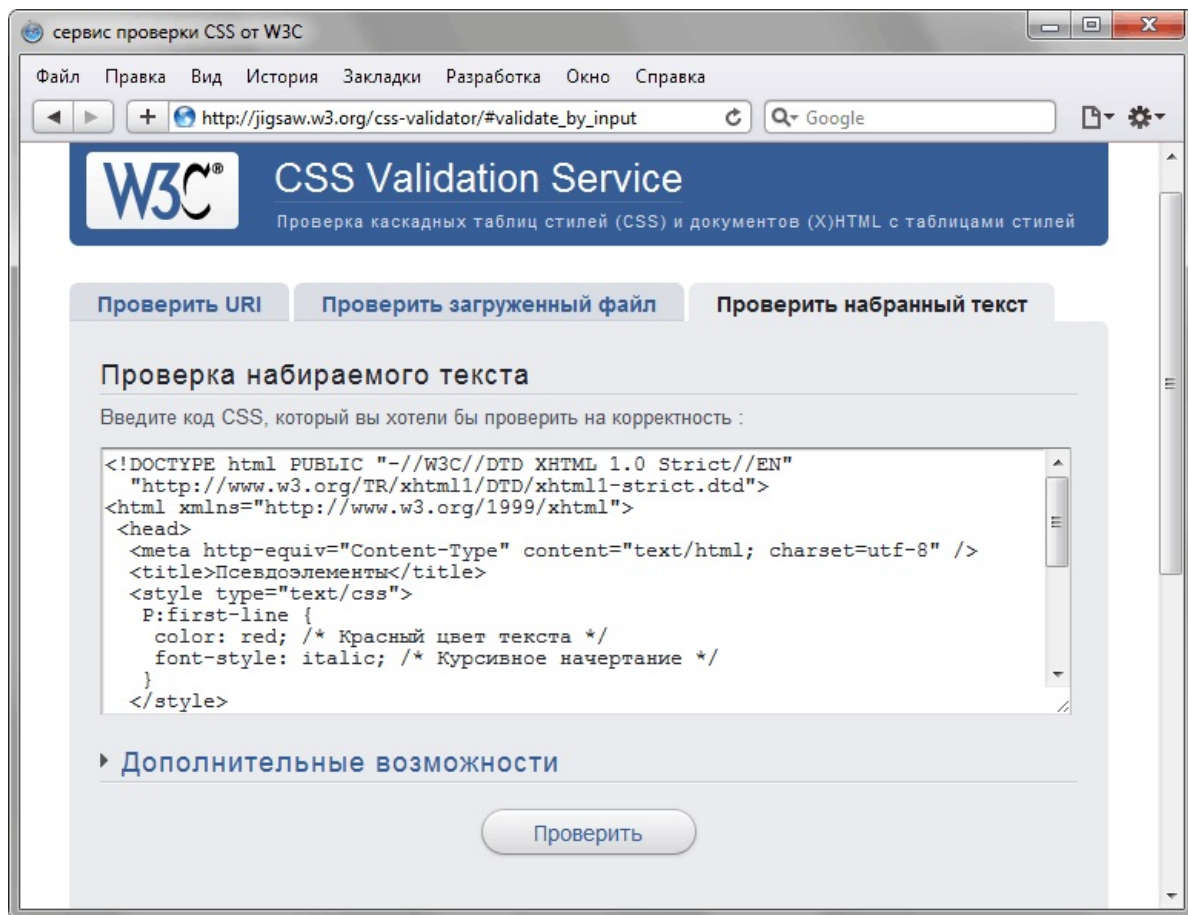


Рис. 20.3. Проверка введённого кода

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 учётом версии. По умолчанию в сервисе указан CSS3, так что если вы хотите проверить код на соответствие CSS2.1, это следует указать явно. Для этого щелкните по тексту «Дополнительные возможности» и в открывшемся блоке из списка «Профиль» выберите CSS2.1 (рис. 20.4).

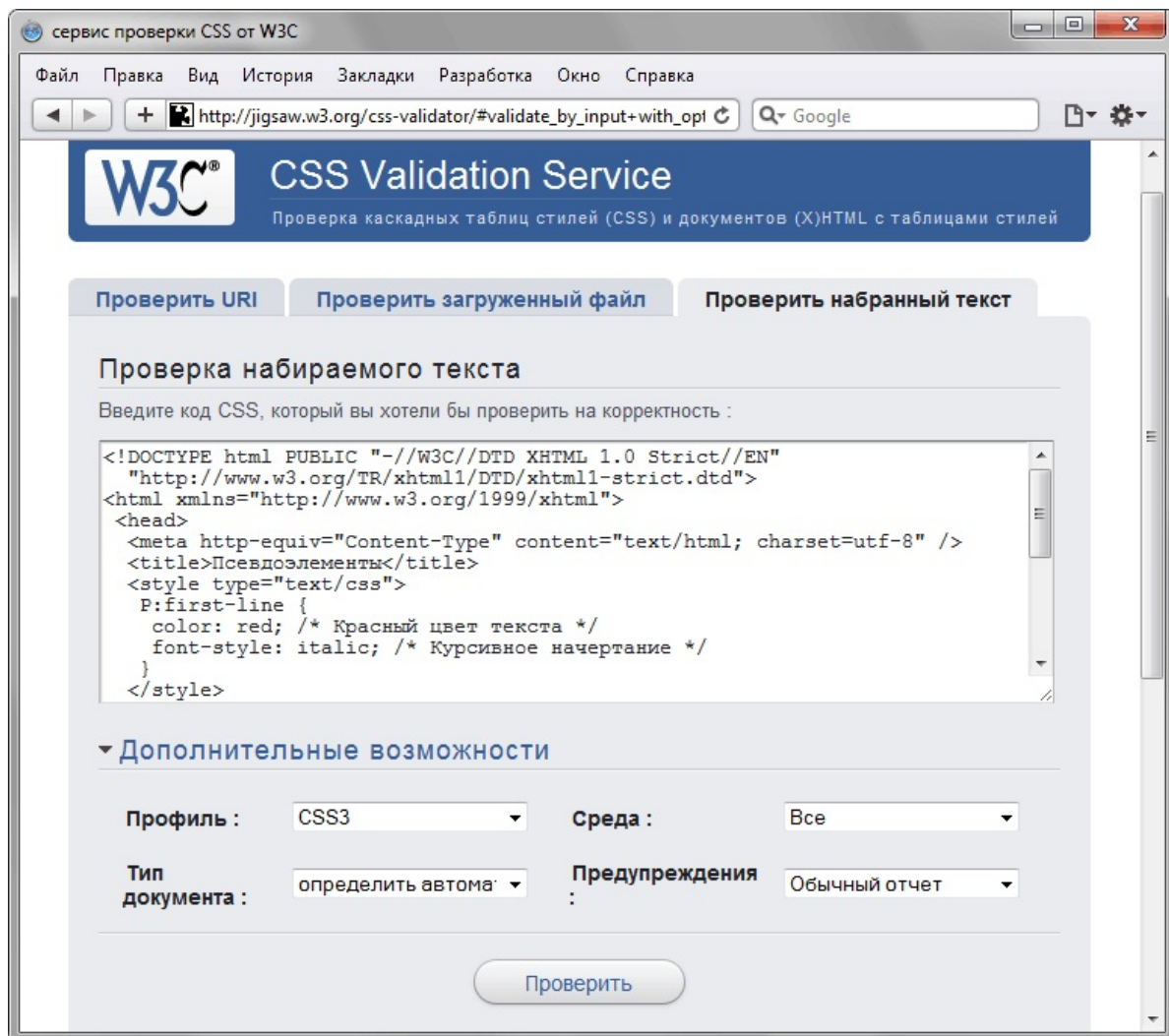


Рис. 20.4. Указание версии CSS для проверки

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.
- Стил 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 21.1 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 21.1. Проверка данных формы

HTML5 IE Cr Op Sa Fx

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Проверка формы</title>
    <script>
      function validForm(f) {
        user = document.getElementById("userName");
        if(user.value == "")
          document.getElementById("msg").innerHTML = '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 21.2.

Пример 21.2. Сочетание стилей

HTML5 CSS 2.1 IE Cr Op Sa Fx

```

<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Идентификаторы</title>
  <style>
    #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. 21.1 показан результат применения стилей.

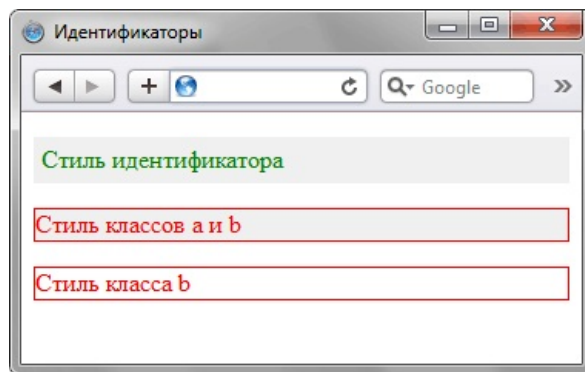


Рис. 21.1. Использование стилей для текста

Специфичность в каскадировании начинает играть роль при разрастании стилевого файла за счёт 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, причём некоторые блоки ещё имеют красную рамку, а некоторые нет. В этом случае можем написать такой стиль (пример 21.3).

Пример 21.3. 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 скруглёнными или прямыми уголками. Это несколько похоже на группирование селекторов, но обладает большей гибкостью.

Написание эффективного кода

В процессе написания CSS следует придерживаться некоторых принципов, которые позволяют сократить код CSS, сделать его более удобным, наглядным и читабельным. Читабельность в данном случае означает, что разработчик спустя какое-то время может легко понять и модифицировать стиль или что в коде разберётся даже сторонний человек.

Размещайте каскадные таблицы стилей в отдельном файле

Размещение стилей в отдельном файле позволяет ускорить загрузку веб-страниц за счёт уменьшения их кода, а также кэширования файла с описанием стиля.

Удаляйте неиспользуемые селекторы

Большое количество селекторов создаёт путаницу в вопросе о том, кто из них за что отвечает, да и просто увеличивает объем документа. Чтобы этого не произошло, удаляйте селекторы, которые никак не применяются на сайте. К сожалению, определить точно, какой селектор используется, а какой нет, довольно сложно, поэтому добавляйте комментарий в код. Это поможет хотя бы не запутаться в большом объёме текста.

Применяйте группирование

Достоинство и удобство группирования состоит в описании одинаковых свойств в одном месте. Тем самым, значение свойства пишется только один раз, а не повторяется многократно.

Используйте универсальные свойства

Вместо того чтобы указывать значения отступа на каждой стороне элемента через свойства `margin-left`, `margin-right`, `margin-top` и `margin-bottom`, это можно одновременно задать через универсальное свойство `margin`. Перечисление значений через пробел позволяет установить индивидуальные отступы для каждой стороны. Кроме `margin` к универсальным свойствам относятся `background`, `border`, `font`, `padding`. Применение этих свойств сокращает объём кода и повышает его читабельность.

Форматирование кода

Существует множество разных подходов как же писать CSS-код. Кто-то упорядочивает селекторы по блокам, другой согласно структуре документа, третий по алфавиту, в общем, сколько людей, столько и мнений. Вы можете воспользоваться онлайн-инструментом, который форматирует CSS-код сразу четырьмя разными способами. А там уже сами решите, какой из способов вам симпатичнее.

Ссылка на сайт

<http://www.cssportal.com/format-css/>

Принцип работы очень простой, вводите в текстовое поле свой код, нажимаете на кнопку «Format Code» и получаете четыре разных вида первоначального кода. Для примера возьмём следующий небольшой фрагмент.

```
body {
  font: 0.9em Arial, Verdana, Helvetica, sans-serif;
  color: #000;
  background: #fff;
  margin: 0;
}
.top {
  margin-bottom: 10px;
  padding-left: 3%;
```

```
border-bottom: 1px solid #acacac;
}
```

В результате его форматирования получатся такие варианты.

Форматированный CSS (Formatted CSS)

```
body {
    font: 0.9em Arial, Verdana, Helvetica, sans-serif;
    color: #000;
    background: #fff;
    margin: 0;
}

.top {
    margin-bottom: 10px;
    padding-left: 3%;
    border-bottom: 1px solid #acacac;
}
```

Порядок свойств не меняется, строки со свойствами сдвигаются вправо на четыре пробела, селекторы разделяются между собой пустой строкой.

Свойства в алфавитном порядке (Properties in Alphabetical Order)

```
body {
    background: #fff;
    color: #000;
    font: 0.9em Arial, Verdana, Helvetica, sans-serif;
    margin: 0;
}

.top {
    border-bottom: 1px solid #acacac;
    margin-bottom: 10px;
    padding-left: 3%;
}
```

Строки со свойствами сдвигаются вправо на четыре пробела, селекторы разделяются между собой пустой строкой, стилевые свойства упорядочиваются по алфавиту.

Лесенкой (Longest Property to Shortest)

```
body {
    font: 0.9em Arial, Verdana, Helvetica, sans-serif;
    background: #fff;
    color: #000;
    margin: 0;
}

.top {
    border-bottom: 1px solid #acacac;
    margin-bottom: 10px;
    padding-left: 3%;
}
```

Строки со свойствами сдвигаются вправо на четыре пробела, селекторы разделяются между собой пустой строкой, строки со свойствами упорядочиваются по длине. Вначале идут самые длинные строки, в конце самые короткие.

Компактно (Compact)

```
body {font: 0.9em Arial, Verdana, Helvetica, sans-serif;color: #000;background-color: #fff;}.top {margin-bottom: 10px;padding-left: 3%;border-bottom: 1px solid #acacac}
```

Селекторы и свойства записываются в одну строку, пустые строки удаляются.

Приведённый инструмент, конечно, не претендует на полноту, в нём нельзя задать величину отступа между селекторами, количество пробелов перед свойством. Также не сокращаются лишние пробелы перед значениями свойств. Тем не менее, главное, что процесс форматирования кода прост и удобен.

Минимизация кода

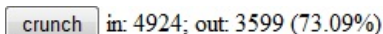
При редактировании CSS-файла возникает противоречивая задача. С одной стороны код должен быть удобным для восприятия и редактирования, быстрого отыскания нужного селектора, для чего активно применяются отбивки, комментарии, пробелы и символы табуляции. С другой стороны, объём кода должен быть компактным и не содержать в себе ничего лишнего. Компактность позволяет несколько ускорить загрузку сайта и повысить его производительность.

Данное противоречие решается наличием двух версий CSS-файла: один файл для редактирования, а второй для загрузки на сервер. Сам же процесс сокращения кода называется минимизацией и вполне автоматизирован с помощью сетевых сервисов, которые и рассмотрим далее.

CSSMin

<http://tools.w3clubs.com/cssmin/>

Простой, даже можно сказать, примитивный сервис, построенный на JavaScript и библиотеке YUI Compressor. Вводите в поле «Source» код CSS, нажимаете кнопку «Crunch» и получаете готовый результат в соседнем поле. Также даётся оценка входного и выходного объёма и соотношение в процентах между ними (рис. 22.1).



crunch in: 4924; out: 3599 (73.09%)

Рис. 22.1. Итог минимизации кода

CSS compressor

<http://www.csscompressor.com>

Этот сервис удобен тем, что комментирует все свои действия, так что вы будете в курсе изменений вашего стиля. Работает он следующим образом. В поле CSS Input вставляете код CSS, выбираете желаемые настройки и нажимаете кнопку «Compress» (рис. 22.2).

Compression Mode:
Highest

Compression Options:

☐ Sort Properties ☐ Lowercase selectors

☒ Compress colors ☒ Remove unnecessary backslashes

☒ Compress font-weight ☒ Remove unnecessary semi-colons

CSS Input:

```
body {
  margin: 20px;
  padding: 0;
  font: 0.92em Arial, sans-serif;
  min-width: 890px;
  line-height: 1.3;
  background: #ffffff;
  color: #000000;
}
```

Compress Reset

Рис. 22.2. Интерфейс

Настройки следующие.

- Compression Mode — режим сжатия. Различается четырьмя видами.
 - Highest — все правила записываются в одну строку.
 - High — каждое правило записывается на своей строке.
 - Standard — каждое свойство пишется на отдельной строке,
 - Low — каждое свойство пишется на отдельной строке и отбивается табуляцией.
- Sort Properties — сортировка стилевых свойств в алфавитном порядке.
- Compress colors — цвета вида #ffffff заменяются сокращённой формой #fff.
- Compress font-weight — оптимизируется насыщенность шрифта. Такое значение **font-weight** как **normal** заменяется на 400, а **bold** на 700.
- Lowercase selectors — все селекторы записываются в нижнем регистре.
- Remove unnecessary backslashes — ненужные слэши (\) удаляются.
- Remove unnecessary semi-colons — удалить необязательную точку с запятой в последнем свойстве.

После сжатия выводятся два поля: список сделанных изменений в свойствах и сжатый CSS (рис. 22.3).

Messages:

```
4 Optimised number: Changed "0.92em" to ".92em"
7 Optimised color: Changed "#ffffff" to "#fff"
24 Optimised number: Changed "0.5em" to ".5em"
29 Optimised font-weight: Changed "normal" to "400"
34 Optimised number: Changed "0.6em" to ".6em"
52 Optimised font-weight: Changed "bold" to "700"
```

Compressed CSS:

```
body{font:.92em Arial, sans-serif;min-width:890px;line-height:1.3;background:#fff;color:#000000;margin:20px;padding:0}form{margin:0}.clear{clear:both}.clearleft{clear:left!important}.clearright{clear:right}a{color:#1d67a4}a:visited{color:#90278E}a:hover{color:#BD2026!important}.error,.ok{color:inherit;padding:3px 3px 3px 30px}.error{background:#F7ECDC url(images/error.png) no-repeat 5px 50%}.ok{background:#E3F7E7 url(images/ok.png) no-repeat 5px 50%}li{margin-bottom:.5em}h1,h2{font-size:1.8em;font-weight:400}h1{text-align:center;font:1.8em Georgia, Times, serif;margin:0 0
```

Compression ratio: 3.607KB/4.926KB = 26.8% (-1319 Bytes) Select All

Рис. 22.3. Результат использования

<http://www.generateit.net/css-optimize>

Сервис построен на том же движке, что и предыдущий, поэтому имеет ряд схожих настроек. Из приятных плюсов можно отметить подсветку синтаксиса кода, сохранение в файл, а также ввод сетевого адреса CSS-файла.

На рис. 22.4 показано окно настроек.

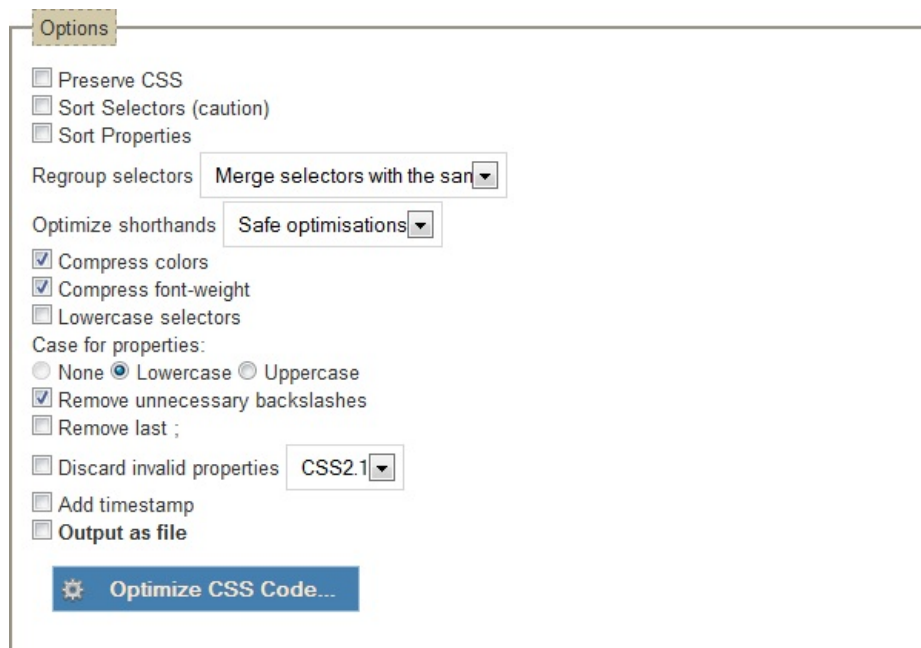


Рис. 22.4. Настройки

- Preserve CSS — сохраняет все комментарии, хаки и др. При включении этой настройки некоторые опции становятся недоступными.
- Sort Selectors (caution) — сортировать селекторы по алфавиту.
- Sort Properties — сортировать свойства по алфавиту.
- Regroup selectors — позволяет перегруппировать селекторы, например, разделить или объединить их.
- Optimize shorthands — оптимизирует универсальные свойства вроде **margin**.
- Compress colors — цвета вида #ffffff заменяются сокращённой формой #fff.
- Compress font-weight — оптимизируется насыщенность шрифта. Такое значение **font-weight** как **normal** заменяется на 400, а **bold** на 700.
- Lowercase selectors — все селекторы записываются в нижнем регистре.
- Case for properties — стилевые свойства пишутся в нижнем или верхнем регистре.
- Remove unnecessary backslashes — удалить ненужные слэши (\).
- Remove last ; — удалить необязательную точку с запятой в последнем свойстве.
- Discard invalid properties — удалить свойства, которых нет в указанной спецификации.
- Add timestamp — включить в код текущую дату и время.
- Output as file — сохранить результат в виде файла.

Библиотека minify

Если приходится часто вносить изменения в CSS-файл, то процесс минимизации становится неудобным. Сами посудите, вначале надо отредактировать файл, затем его минимизировать и полученный код сохранить в файл, который нужно залить на сервер. Слишком много действий приходится совершать ради одного изменения. Логичнее было бы возложить задачу минимизации на сайт. Загрузили файл на сервер, и вот он уже в компактном виде отдаётся

посетителям. Одно из таких универсальных решений называется minify, это библиотека на PHP5. Она минимизирует, объединяет и кэширует CSS-файлы, а также JavaScript.

Ссылка на проект minify

<http://code.google.com/p/minify/>

Библиотека minify существует как отдельно, так и в виде плагина для WordPress.

Процесс использования библиотеки следующий. Скачиваете архив, внутри него лежит каталог min, который необходимо переписать на сервер. Сжатие CSS-файла происходит довольно просто, вместо обычного пути к стилевому файлу теперь указываем:

```
http://example.ru/min/?f=themes/default/style.css
```

В параметре f указывается путь к CSS-файлу относительно корня сайта. Два и более файла пишутся через запятую:

```
http://example.ru/min/?f=themes/default/style.css,themes/default/cms.css
```

Также процесс получения адреса можно автоматизировать, перейдя по адресу <http://example.ru/min>, откроется страница, где предлагается указать путь к файлам, которые вы желаете минимизировать (рис. 22.5).

Minify URI Builder

Create a list of Javascript or CSS files (or 1 is fine) you'd like to combine and click [Update].

1. <http://htmlbook.lc/themes/hb/style.css> [x] [↑] [↓] ✓

2. <http://htmlbook.lc/themes/hb/shadow.css> [x] [↑] [↓] 404! recheck

[Add file +]

[Update]

Рис. 22.5. Страница для управления файлами

Кнопки со стрелками вверх и вниз нужны для изменения порядка файлов, а кнопка «x» для удаления файла из списка. Проверка правильности пути осуществляется автоматически, в случае ошибки появится кнопка с надписью «404!», как показано на рисунке выше. Для добавления еще одного файла в список служит кнопка «Add file». После того, как все файлы указаны, пути к ним заданы корректно, что подтверждается наличием галочки напротив каждого файла, можно нажать кнопку «Update». Ниже на странице появится ссылка на новый комбинированный файл (текст с надписью URI) и тег `<link>` (текст с надписью HTML), который требуется вставить себе на страницу взамен старого (рис. 22.6).

Minify URI

Place this URI in your HTML to serve the files above combined, minified, compressed and with cache headers.

URI </min/b=themes/hb&f=style.css,shadow.css> (opens in new window)

HTML `<link type="text/css" rel="stylesheet" href="/min/b=themes/hb&f=style.css,shadow.css" />`

Рис. 22.6. Результат минимизации

Согласно тестам над WordPress использование библиотеки позволяет сократить количество HTTP-запросов и уменьшить объем CSS и JavaScript-файлов до 70%.

Кроме приведенных методов минимизации CSS-файлов имеются также решения, позволяющие архивировать файлы утилитой gzip прямо на сервере и отдавать браузеру упакованную версию. Современные браузеры прекрасно понимают gzip и распаковывают его на месте. Весь процесс происходит автоматически и приводит к существенному сокращению объема передаваемых по сети файлов. Вопросы настройки gzip выходят за рамки самоучителя, поэтому я не буду на них останавливаться. Всем заинтересованным рекомендую статьи по этим ссылкам.

- [Модуль mod_deflate](#)
- [Сжатие ответов Веб-сервера Apache средствами модуля mod_deflate](#)
- [Настройка mod_gzip – сжатие трафика в Apache](#)