Тема 4-2.
Технология CSS

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 показано изменение стиля тега **<q>**, в том случае, если к нему добавлен атрибут title.

Вид элемента в зависимости от его атрибута

```
<!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>
```

Селекторы атрибутов

Продолжая известный закон Мерфи, который гласит: *"Если неприятность может случиться, то она обязательно случится"*, можем ввести свое наблюдение: *"После того, как веб-страница будет корректно отображаться в одном браузере, выяснится, что она неправильно показывается в другом"*.

Атрибут со значением

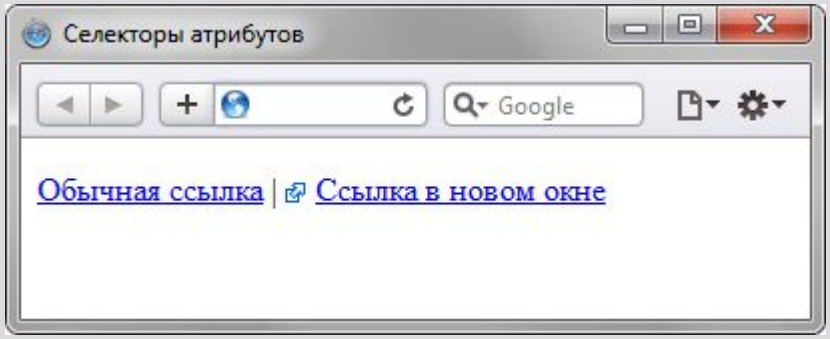
- Устанавливает стиль для элемента в том случае, если задано определённое значение специфичного атрибута. Синтаксис применения следующий.

[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 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.

Стиль для открытия ссылок в новом окне




```
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="utf-8">
  <title>Селекторы атрибутов</title>
  <style>
   A[target="_blank"] {
    background: url(images/blank.png) 0 6px no-repeat; /*
Параметры фонового рисунка */
    padding-left: 15px; /* Смещаем текст вправо */
   }
  </style>
 </head>
 <body>
   <p><a href="1.html">Обычная ссылка</a> |
   <a href="link2" target="_blank">Ссылка в новом окне</a></p>
 </body>
</html>
```

Значение атрибута начинается с определённого текста

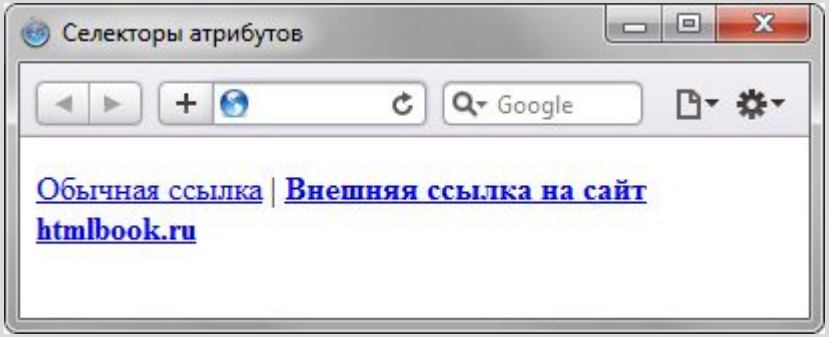
- Устанавливает стиль для элемента в том случае, если значение атрибута тега начинается с указанного текста. Синтаксис применения следующий.

[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

Изменение стиля внешней ссылки




```
<!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>
```

Значение атрибута заканчивается определённым текстом

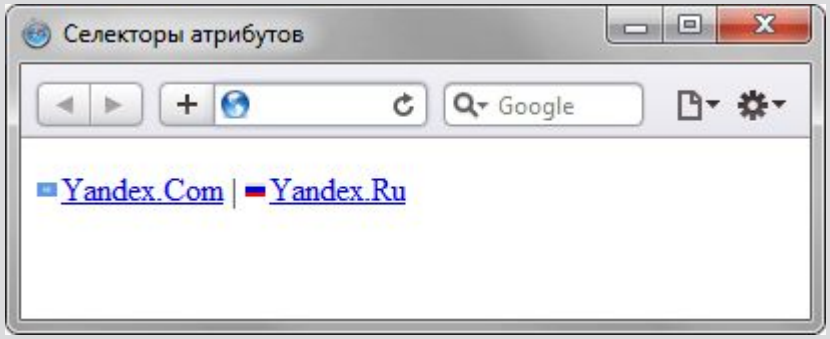
- Устанавливает стиль для элемента в том случае, если значение атрибута заканчивается указанным текстом. Синтаксис применения следующий.

[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

Стиль для разных доменов




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

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

 </body>
</html>
```

Значение атрибута содержит указанный текст

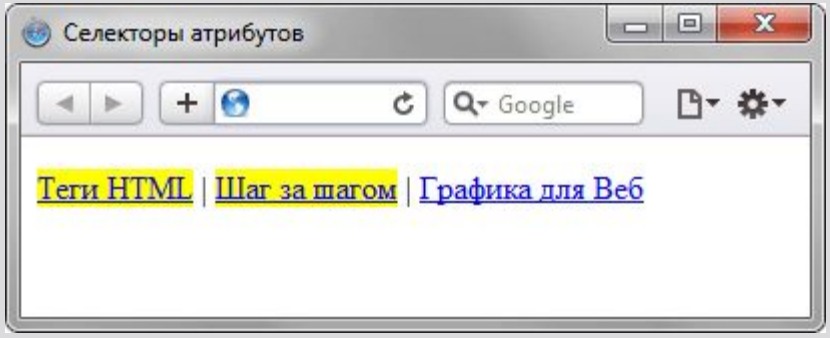
- Возможны варианты, когда стиль следует применить к тегу с определённым атрибутом, при этом частью его значения является некоторый текст.
- При этом точно не известно, в каком месте значения включен данный текст — в начале, середине или конце.
- В подобном случае следует использовать такой синтаксис.

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

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

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

Стиль для разных сайтов




```
<!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>
```

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

Стиль в зависимости от имени класса

```
<!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

Дефисы в значениях

```
<!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.

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

Использование универсального селектора

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

Аналогичный результат можно получить, если в данном примере поменять селектор * на **BODY**.

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.

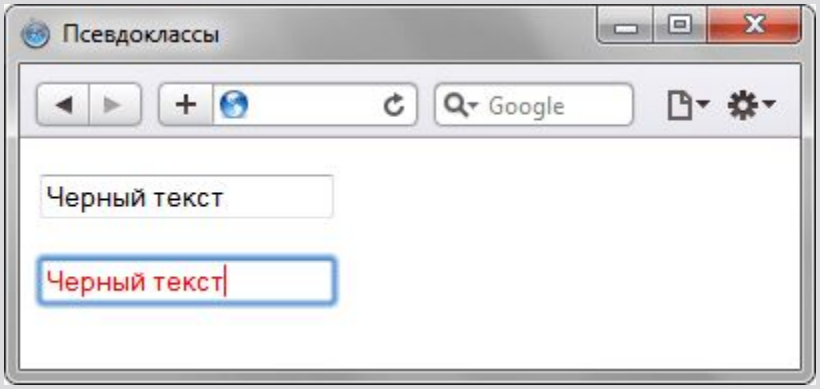
:link

- Применяется к непосещенным ссылкам, т. е. таким ссылкам, на которые пользователь ещё не нажимал. Браузер некоторое время сохраняет историю посещений, поэтому ссылка может быть помечена как посещенная 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

Применение псевдокласса :focus




```
<!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>
```

Псевдоклассы

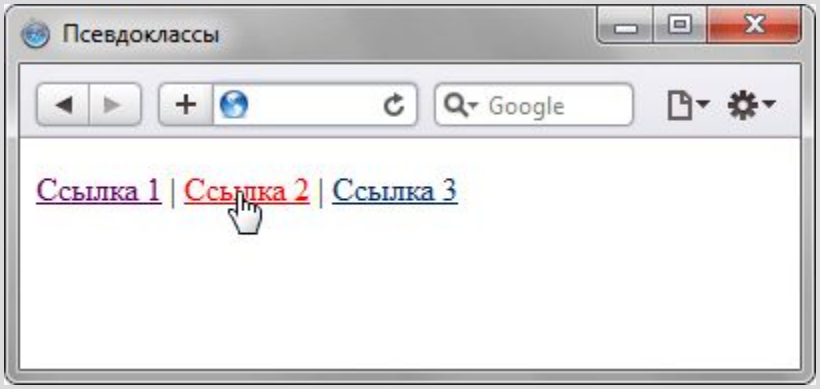
:hover

- 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

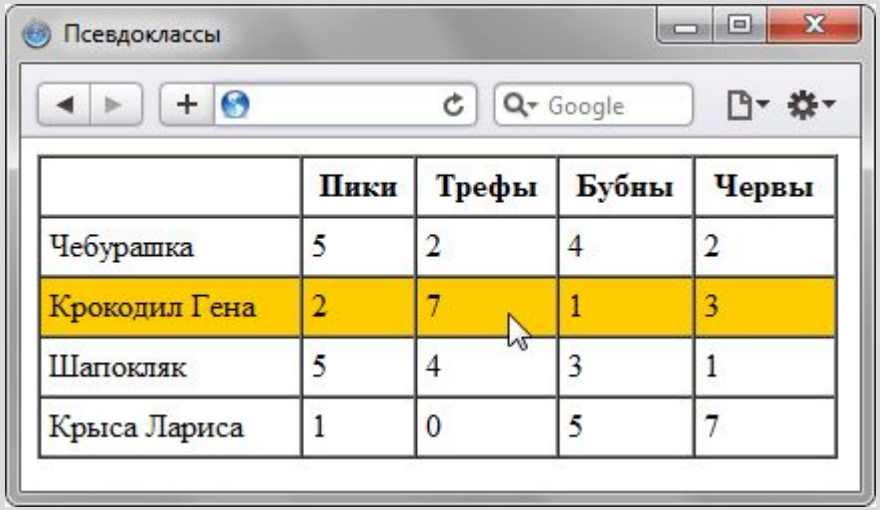
Изменение цвета ссылок




```
<!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>
```

Выделение строк таблицы

```
<!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>
```



	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

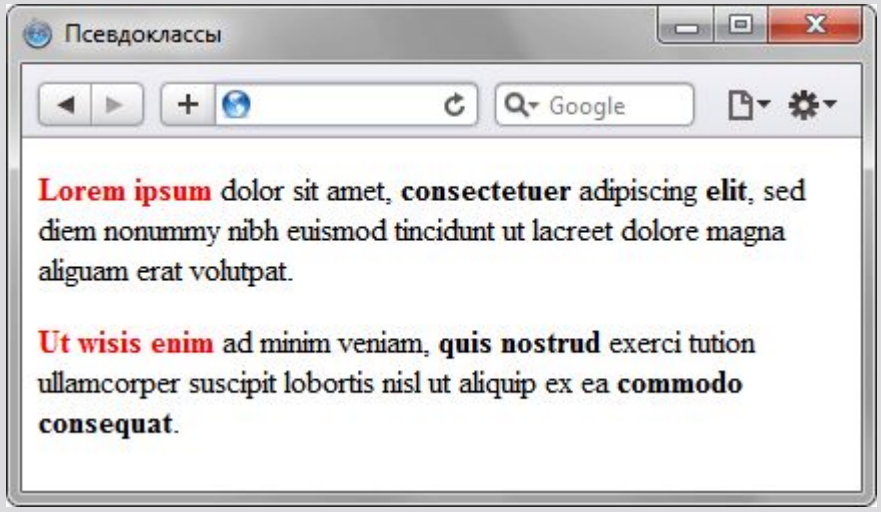
Псевдоклассы, имеющие отношение к дереву документа

- 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

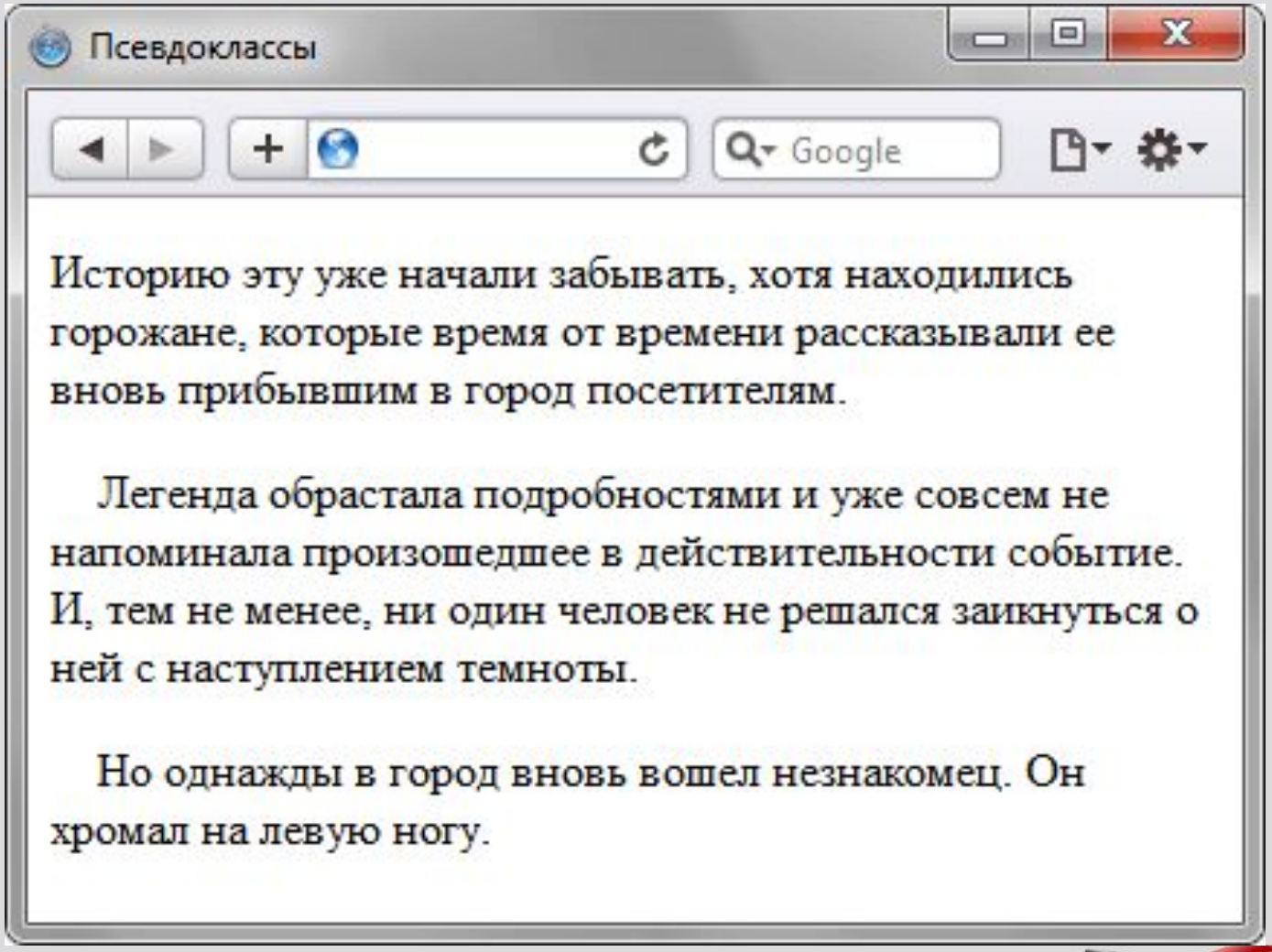
Использование псевдокласса :first-child




```
<!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 aliguam 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>
```

Отступы для абзаца

```
<!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>
```


Псевдоклассы
Google
Историю эту уже начали забывать, хотя находились горожане, которые время от времени рассказывали ее вновь прибывшим в город посетителям.
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.
Но однажды в город вновь вошел незнакомец. Он хромал на левую ногу.

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.

Вид кавычек в зависимости от языка

```
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="utf-8">
  <title>lang</title>
  <style>
   P {
    font-size: 150%; /* Размер текста */
   }
   q:lang(de) {
    quotes: "\201E" "\201C"; /* Вид кавычек для немецкого языка */
   }
   q:lang(en) {
    quotes: "\201C" "\201D"; /* Вид кавычек для английского языка */
   }
   q:lang(fr), q:lang(ru) { /* Вид кавычек для русского и французского языка */
    quotes: "\00AB" "\00BB";
   }
  </style>
 </head>
 <body>
  <p>Цитата на французском языке: <q lang="fr">Ce que femme veut, Dieu le veut</q>.</p>
  <p>Цитата на немецком: <q lang="de">Der Mensch, versuche die Gotter nicht</q>.</p>
  <p>Цитата на английском: <q lang="en">To be or not to be</q>.</p>
 </body>
</html>
```

lang - Opera

Файл Правка Вид Закладки Виджеты Инструменты Справка

Цитата на французском языке: «Ce que femme veut, Dieu le veut».

Цитата на немецком: „Der Mensch, versuche die Gotter nicht“.

Цитата на английском: “To be or not to be”.

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.

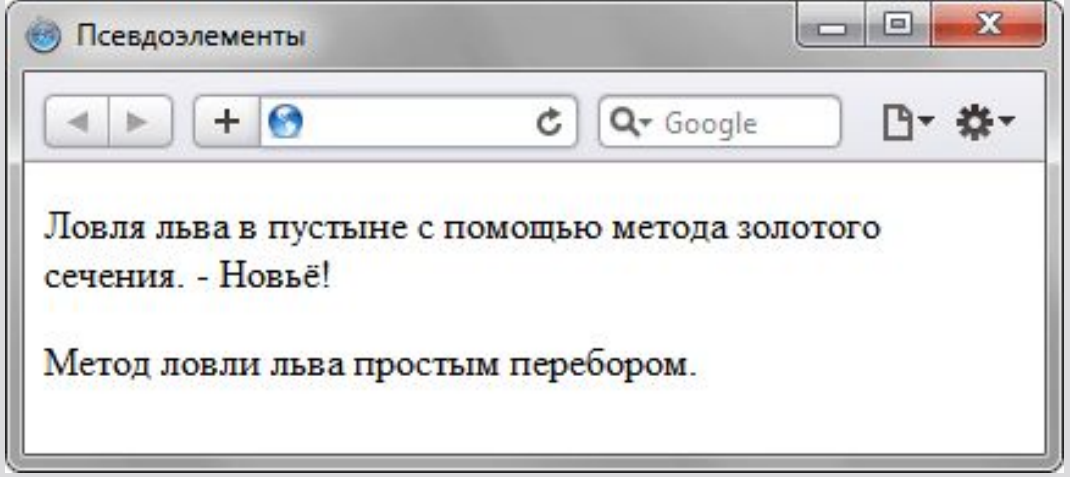
```
.foo:first-letter { color: red }
.foo:first-line {font-style: italic}
```

- 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 показано использование псевдоэлемента :after для добавления текста в конец абзаца.

Применение :after

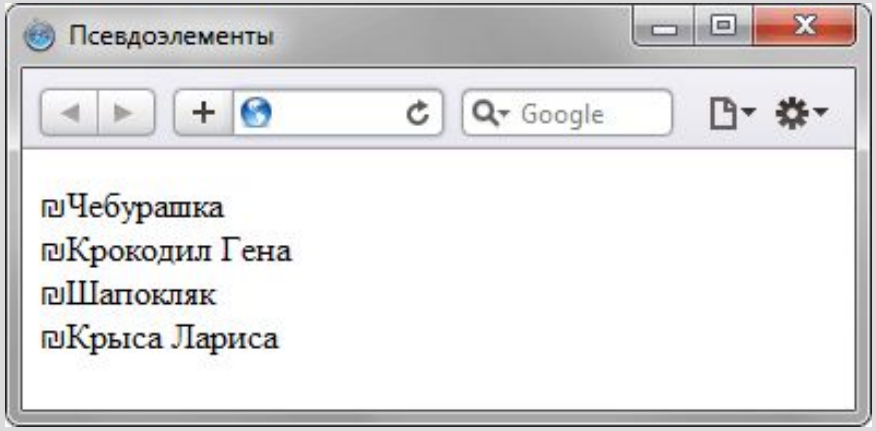



```
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="utf-8">
  <title>Псевдоэлементы</title>
  <style>
   P.new:after {
    content: " - Новьё!"; /* Добавляем после текста абзаца */
   }
  </style>
 </head>
 <body>
  <p class="new">Ловля льва в пустыне с помощью метода
золотого сечения.</p>
  <p>Метод ловли льва простым перебором.</p>
 </body>
</html>
```

:before

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

Использование :before



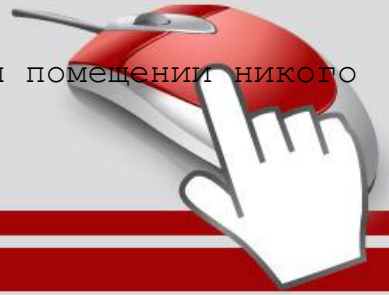

```
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="utf-8">
  <title>Псевдоэлементы</title>
  <style>
   UL {
    padding-left: 0; /* Убираем отступ слева */
    list-style-type: none; /* Прячем маркеры списка */
   }
   LI:before {
    content: "\20aa "; /* Добавляем перед элементом списка символ в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

: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.

Использование :first-letter

```
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="utf-8">
  <title>Псевдоэлементы</title>
  <style>
   P {
    font-family: Arial, Helvetica, sans-serif; /* Гарнитура шрифта основного
текста */
    font-size: 90%; /* Размер шрифта */
    color: black; /* Черный цвет текста */
   }
   P:first-letter {
    font-family: 'Times New Roman', Times, serif; /* Гарнитура шрифта первой
буквы */
    font-size: 200%; /* Размер шрифта первого символа */
    color: red; /* Красный цвет текста */
   }
  </style>
 </head>
 <body>
    <p>Луч фонарика высветил старые скрипучие ступени, по которым не далее как
пять минут назад в дом поднялся Паша. Оля осторожно приоткрыла дверь и посветила
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>
```



Псевдоэлементы

Google

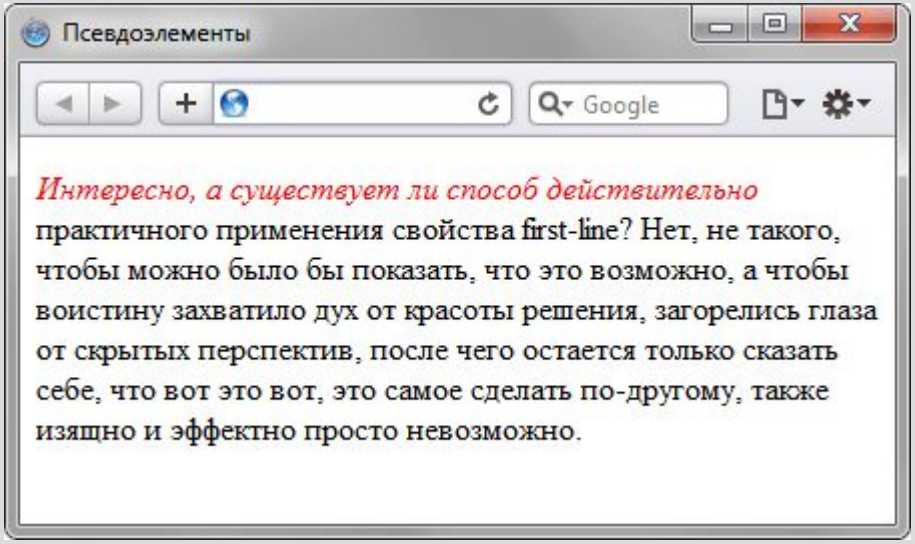
Луч фонарика высветил старые скрипучие ступени, по которым не далее как пять минут назад в дом поднялся Паша. Оля осторожно приоткрыла дверь и посветила внутрь дома. Луч света, словно нехотя, пробивался сквозь тугую завесу из мрака и пыли.

Взгляд Оли опустился на пол, и она вскрикнула. В пустом помещении никого не было, и лишь на полу валялась порванная туфля Паши.

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

Выделение первой строки текста




```
<!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>
```

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

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.

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.

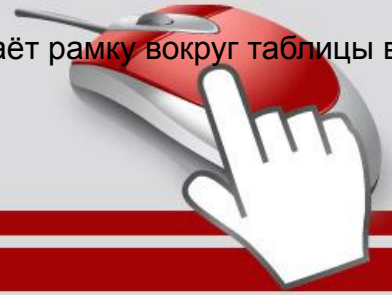
НАСЛЕДОВАНИЕ

- 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.

Наследование параметров цвета

```
<!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.



Параметры текста для всей веб-страницы

```
<!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.

Изменение свойств наследуемого элемента

```
<!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.
  - Стиль браузера.
  - Стиль автора.
  - Стиль пользователя.
  - Стиль автора с добавлением !important.
  - Стиль пользователя с добавлением !important.
- Самым низким приоритетом обладает стиль браузера — оформление, которое по умолчанию применяется к элементам веб-страницы браузером.
- Это оформление можно увидеть в случае «голого» HTML, когда к документу не добавляется никаких стилей.

!important

- Ключевое слово !important играет роль в том случае, когда пользователи подключают свою собственную таблицу стилей.
- Если возникает противоречие, когда стиль автора страницы и пользователя для одного и того же элемента не совпадает, то !important позволяет повысить приоритет стиля или его важность, иными словами.
- При использовании пользовательской таблицы стилей или одновременном применении разного стиля автора и пользователя к одному и тому же селектору, браузер руководствуется следующим алгоритмом.

!important добавлен в авторский стиль — будет применяться стиль автора.

!important добавлен в пользовательский стиль — будет применяться стиль пользователя.

!important нет как в авторском стиле, так и 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 a) начисляется 100, за каждый класс и псевдокласс (b) начисляется 10, за каждый селектор тега и псевдоэлемент (c) 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.

Цвет списка

```
<!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.

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.

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 онлайновым 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.
- В результате его форматирования получатся такие варианты.

```
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;
}
```

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: #fff;margin: 0;}
.top {margin-bottom: 10px;padding-left: 3%;border-bottom:
1px solid #acacac;}
```

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

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.

CSSMin

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

- Простой, даже можно сказать, примитивный сервис, построенный на JavaScript и библиотеке YUI Compressor.

CSSMin

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

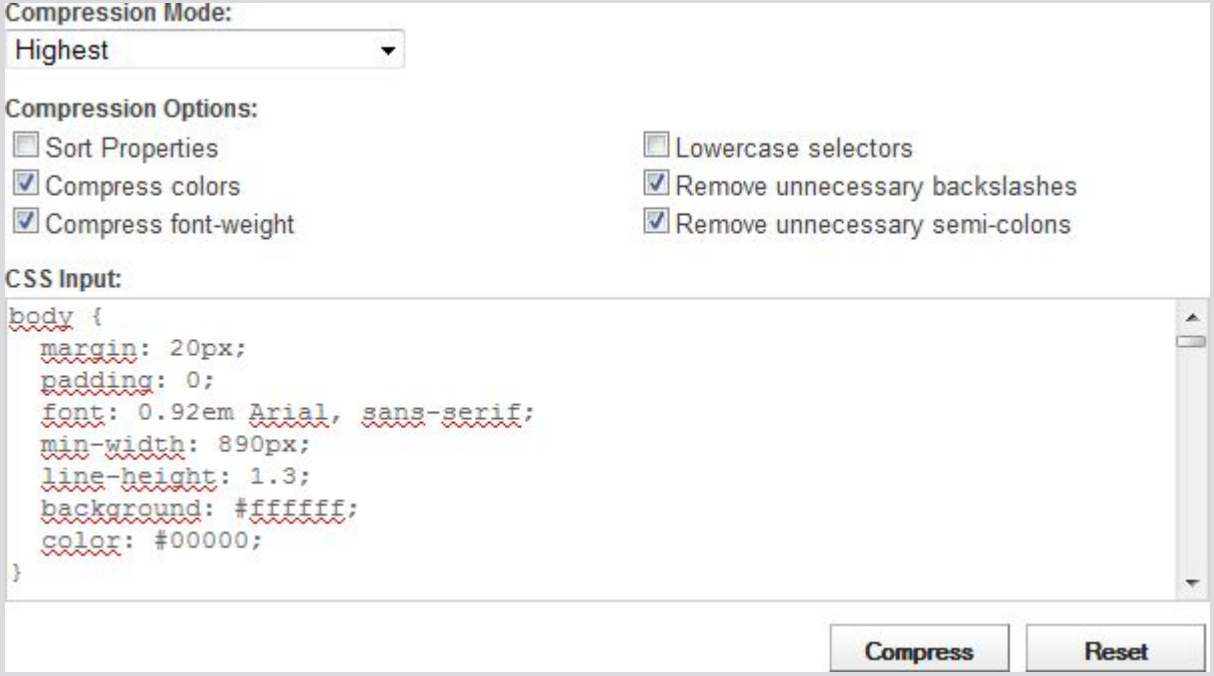
- Вводите в поле «Source» код CSS, нажимаете кнопку «Crunch» и получаете готовый результат в соседнем поле.
- Также даётся оценка входного и выходного объёма и соотношение в процентах между ними

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

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»



CSS compressor

http://www.csscompressor.com

- 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

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:#00000;margin:20px;padding:0}form{margi
n:0}.clear{clear:both}.clearleft{clear:left!important}.clearright{clear:
right}a{color:#1d67a4}a:visited{color:#90278E}a:hover{color:#BD2026!impo
rtant}.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)

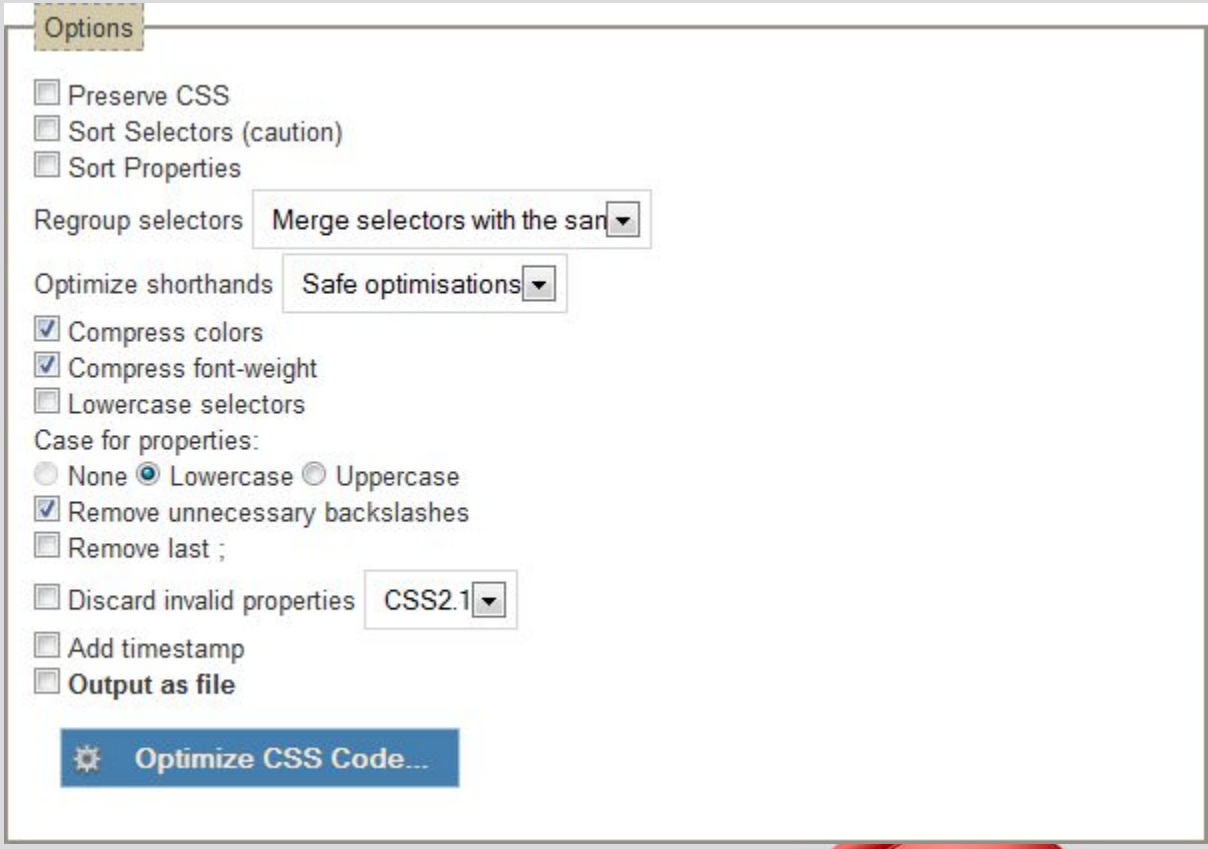
Select All

Результат использования

CSS Code Formatter and Optimizer

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 рисунке показано окно настроек.



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.

Спасибо за внимание!