Английский язык
Библиотека в школе
Биология
География
Дошкольное образование
Здоровье детей
Искусство

Информатика

№28

История
Классное руководство
Литература
Математика
Начальная школа
Немецкий язык
Педагогика
Русский язык
Спорт в школе
Управление школой
Физика
Французский язык
Химия
Школьный психолог

Библиотечка «Первого сентября»

А.А. ДУВАНОВ

CSS: кратко о самом главном

БИБЛИОТЕЧКА «ПЕРВОГО СЕНТЯБРЯ»
Серия «Информатика»
Выпуск 28

А.А. Дуванов

CSS: КРАТКО О САМОМ ГЛАВНОМ

Москва
Чистые пруды
2009

При подготовке брошюры использовались следующие источники:

• Описание стандарта CSS2.1 от W3C (*www.w3.org/TR/2007/CR-CSS21-20070719/*).

• Книга Эрика Мейера "CSS — каскадные таблицы стилей. Подробное руководство, 3-е издание" (Символ, 2008).

Эрик Мейер — широко известный эксперт в области стандартов HTML и CSS, работающий в области веб-технологий с 1993 года. Работал в группе W3C, принимал участие в многочисленных конференциях, посвященных веб-стандартам, применению CSS и веб-дизайну.

HTML и CSS

Объекты гипертекстового документа кодируются при помощи "структурных" элементов HTML, таких, как **H1**, **P**, **A**, **BLOCKQUOTE**, **UL**, **OL**, **TABLE**, **IMG**.

Представление объектов на экране можно задать на HTML "визуальными" элементами (такими, как **FONT**, **CENTER**, **B**, **I** **S**, **U**) и "визуальными" атрибутами в тегах (такими, как `face`, `size`, `color`).

Таким образом, с помощью HTML можно решить две задачи:

1. Разметить структуру документа (указать деление на главы, параграфы, абзацы…; задать ссылки, цитаты, списки, таблицы, иллюстрации…).

2. Объяснить браузеру, как отображать элементы на экране.

Вторую задачу лучше выполнять с помощью каскадных таблиц стилей (CSS), отделяя тем самым структурную разметку документа (на языке HTML) от описаний визуальных свойств объектов (на языке CSS).

CSS (от англ. ***C**ascading **S**tyle **S**heets* — каскадные таблицы стилей) — технология задания свойств объектов гипертекстового документа.

Стили для отдельного элемента

CSS-свойства можно указывать прямо в открывающем теге элемента HTML при помощи атрибута `style`. Ниже для второго абзаца определен увеличенный размер шрифта (значение `large` свойства `font-size`) и красный цвет (значение `red` свойства `color`):

```
<P>Первый абзац.
<P style="color:red;font-size:large;">Второй абзац.
<P>Третий абзац.
```

Такой способ использования стилевых определений не приводит к отделению визуальной разметки от структурной, но позволяет задавать свойства, недоступные для управления атрибутами элементов HTML.

Стили для отдельного файла

CSS-код можно записывать в головной части HTML-файла (внутри элемента **STYLE**) в виде:

ЭЛЕМЕНТ
{
  свойство:значение;
  …
  свойство:значение;
}
	Заголовочная часть определения (в данном случае **ЭЛЕМЕНТ**) называется *селектором*. Селектор предназначен для связи стилевого определения с элементами HTML. *Тело определения* в фигурных скобках состоит из набора *стилевых правил*, отделяемых друг от друга точкой с запятой. Стилевое правило указывает допустимое *значение* для стилевого *свойства* (ключевое слово языка).

Такое определение будет работать для всех элементов документа с именем **ЭЛЕМЕНТ** в текущем файле.

Ниже, в примере, для элемента **P** определен увеличенный размер шрифта и красный цвет. Это указание действует на все абзацы, расположенные в текущем файле HTML:

```
<HTML>
  <HEAD>
    <META http-equiv="Content-Type"
          content="text/html; charset=windows-1251">
    <TITLE>CSS-свойства заданы в головной части</TITLE>
    <STYLE type="text/css">
      P
      {
        color:red;
        font-size:large;
      }
    </STYLE>
  </HEAD>
  <BODY>
    <H1>CSS-свойства заданы в головной части</H1>
    <HR>
    <P>Первый абзац.
    <P>Второй абзац.
    <P>Третий абзац.
  </BODY>
</HTML>
```

Стили для сайта

Самым эффективным способом отделения визуальной разметки от структурной является размещение стилевых определений в отдельном стилевом файле (с расширением `css`). Ниже, в примере, в файле `style.css` для элемента **`P`** определен увеличенный размер шрифта и красный цвет. Это указание действует на все HTML-файлы, в которых в головной части указан путь к файлу `style.css` (при помощи элемента **`LINK`**).

Файл `style.css`

```
P
{
  color:red;
  font-size:large;
}
```

Файл `01.htm`

```
<HTML>
  <HEAD>
    <META http-equiv="Content-Type"
          content="text/html; charset=windows-1251">
    <LINK rel="stylesheet" type="text/css"
          href="style.css" media="all">
    <TITLE>Документ 01</TITLE>
  </HEAD>
  <BODY>
    <H1>Документ 01</H1>
    <HR>
    <P>Первый абзац.
    <P>Второй абзац.
    <P>Третий абзац.
  </BODY>
</HTML>
```

Файл `02.htm`

```
<HTML>
  <HEAD>
    <META http-equiv="Content-Type"
          content="text/html; charset=windows-1251">
    <LINK rel="stylesheet" type="text/css"
          href="style.css" media="all">
    <TITLE>Документ 02</TITLE>
  </HEAD>
  <BODY>
    <H1>Документ 02</H1>
    <HR>
    <P>Первый абзац.
    <P>Второй абзац.
```

```
    <P>Третий абзац.
  </BODY>
</HTML>
```

Хранение стилевых описаний в отдельном файле является обычной практикой. Такой способ расположения стилевых определений очень удобен. На один и тот же стилевой файл могут ссылаться много HTML-документов. Изменения в этом единственном файле скажутся на внешнем виде сотен документов, составляющих большой сайт.

Элемент LINK и его атрибуты

Элемент **LINK**, как мы видели, используется для связи стилевого файла с файлами HTML.

С файлом HTML может быть связан не один, а несколько стилевых файлов:

```
<LINK rel="stylesheet" type="text/css"
      href="style.css" media="all">
<LINK rel="stylesheet" type="text/css"
      href="add.css" media="all">
```

Браузер загрузит обе таблицы и будет работать с ними так, как если бы вторая таблица была продолжением первой.

Атрибут `rel="stylesheet"` информирует браузер о том, что будет подгружаться файл стилевых описаний. Атрибут `type="text/css"` указывает на язык (в данном случае CSS). Атрибут `href="style.css"` (и `href="add.css"`) задает адрес файла.

Наконец, атрибут `media` указывает среду представления, для которой записаны стилевые правила. В нашем случае указано значение `all` — для всех сред. Другие наиболее распространенные значения: `screen` (для экрана), `print` (для печати). Можно задавать не одно, а несколько значений этого атрибута, перечисляя их через запятые: `media="screen, print"`.

Элемент STYLE и директива @import

Связать CSS-файл с файлами HTML можно и при помощи директивы `@import`:

```
<HTML>
  <HEAD>
    <META http-equiv="Content-Type"
          content="text/html; charset=windows-1251">
    <STYLE type="text/css">@import(main.css);</STYLE>
    <TITLE>Документ</TITLE>
  </HEAD>
```

```
  <BODY>
    …
  </BODY>
</HTML>
```

Как и в случае с **LINK**, можно указывать несколько файлов со стилями:

```
<STYLE type="text/css">@import(main.css);@import(add.css);
</STYLE>
```

Директиву `@import` понимают только современные браузеры, поэтому старые загрузить стили не смогут. И это хорошо! Например, Netscape Navigator 4 "думает", что он понимает CSS, и превращает страницу в кашу. Пусть лучше он показывает сайт без всяких стилей. Красот не будет, но функциональность останется.

Можно совмещать загрузку стилевого файла и стилевые определения в элементе **STYLE**:

```
<HTML>
  <HEAD>
    <META http-equiv="Content-Type"
          content="text/html; charset=windows-1251">
    <STYLE type="text/css">
      @import(main.css);
      P
      {
        color:red;
        font-size:large;
      }
    </STYLE>
    <TITLE>Документ</TITLE>
  </HEAD>
  <BODY>
  …
  </BODY>
</HTML>
```

Директивы `@import` должны идти в элементе **STYLE** самыми первыми (иначе они не будут работать).

Селекторы элементов, классов и идентификаторов

Стилевое определение, размещаемое в головной части HTML-кода или в отдельном файле CSS, имеет вид:

```
селектор
{
  набор правил
}
```

Селектор предназначен для связи стилевого определения с элементами HTML. В качестве селектора часто (но не всегда) используется название элемента:

```
P
{
  color:red;
  font-size:large;
}
```

В таком случае говорят, что задано определение с *селектором элемента*.

Стилевые определения можно строить без явной связи с конкретными элементами. В этом случае с определением связывается имя *класса*, которое можно использовать для сопоставления заданного стиля с конкретными элементами (входящими в этот класс). Такие стилевые определения называют определениями с *селектором класса*.

Определение с селектором класса записывается следующим образом:

```
.имя_класса
{
  набор правил
}
```

Заметим, что перед именем класса ставится точка. Сопоставить такое определение с элементом можно при помощи атрибута `class`, задав в качестве значения `имя_класса`.

Так, если в стилевом файле записано определение с именем `mark` (задан красный цвет и рубленый шрифт):

```
.mark
{
  color:red;
  font-family:sans-serif;
}
```

— то для применения его к конкретным элементам нужно в открывающих тегах этих элементов указать атрибут `class` со значением `mark`:

```
<H1 class=mark>Заголовок</H1>
<P>
Текст первого абзаца.
<P class=mark>
Текст второго абзаца.
```

Стилевое определение `mark` будет применяться к заголовку и второму абзацу.

Селектор класса можно записывать с привязкой к элементу:

```
P.mark
{
  color:red;
  font-size:large;
}
```

Такое определение будет работать для тех элементов **P**, открывающий тег которых содержит атрибут `class=mark`, и не будет работать для других элементов, даже с атрибутом `class=mark` в открывающем теге.

Связать стилевое определение с элементом можно и при помощи *селектора идентификатора*.

Селектор идентификатора аналогичен селектору класса, но перед именем ставится не точка, а знак "решетки" (#):

```
#идентификатор
{
  набор правил
}
```

Сопоставить такое определение с элементом можно при помощи атрибута `id`, задав в качестве значения `идентификатор`.

Так, если в стилевом файле записано определение:

```
#mark
{
  color:red;
  font-size:large;
}
```

— то для применения его к конкретному элементу нужно записать в HTML-коде:

```
<P id=mark>
Текст абзаца.
```

Важно помнить, что элементов с одним и тем же классом на странице может быть *много*, а каждый идентификатор — *уникален*, то есть на странице не должно быть двух элементов с одним и тем же идентификатором.

Идентификаторы удобны, когда с элементом планируется связать не только CSS, но и JavaScript (идентификатор используется для доступа к элементу в скриптах).

Универсальный селектор

Этот селектор представлен символом звездочка (*) и соответствует любому элементу.

Чтобы все элементы сделать красными, достаточно написать такое определение:

```
*{color:red;}
```

Группировка селекторов

Одно определение можно сопоставить нескольким селекторам, перечисляя их через запятые (*группировка селекторов*). Следующее определение (красный цвет и рубленый шрифт) относится к заголовкам всех уровней:

```
H1,H2,H3,H4,H5,H6
{
  color:red;
  font-family:sans-serif;
}
```

Контекстные определения

Можно написать стилевое определение, которое будет работать только при определенной комбинации вложенности элементов (*контекстное* определение).

Для задания контекстного определения можно в его заголовке перечислить через пробелы (*символ-комбинатор селектора потомка*) имена тегов в порядке предполагаемой вложенности. Например, можно установить цвет для **EM** красным только для случая, когда этот элемент расположен внутри элемента **P**:

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

Это определение будет применяться ко всем элементам **EM**, расположенным в следующем коде:

```
<P>
В этом абзаце <EM>элемент EM</EM> является прямым
потомком элемента P.
<P>
В этом абзаце <STRONG><EM>элемент EM</EM></STRONG>
не является прямым потомком элемента P.
```

В некоторых случаях требуется указывать не любые, а только прямые потомки. Для этой цели используют *символ-комбинатор селектора дочерних элементов* (знак “больше”):

```
P>EM {color:red;}
```

Это определение будет применяться к элементу **EM**, расположенному в первом абзаце, и не будет работать для элемента **EM**, расположенного во втором абзаце приведенного ниже кода:

```
<P>
В этом абзаце <EM>элемент EM</EM> является прямым
потомком элемента P.
<P>
В этом абзаце <STRONG><EM>элемент EM</EM></STRONG>
не является прямым потомком элемента P.
```

Иногда требуется написать стилевое определение, которое относилось бы к элементу, непосредственно следующему за другим элементом в HTML-коде и имеющему того же родителя.

Пусть два элемента — **ЭЛЕМЕНТ1** и **ЭЛЕМЕНТ2** — имеют общего родителя и следуют непосредственно друг за другом в HTML-коде. Стилевые определения для второго элемента можно записать, используя *символ-комбинатор селектора сестринского элемента* (знак плюс):

```
ЭЛЕМЕНТ1+ЭЛЕМЕНТ2 {набор правил}
```

Чтобы удалить верхний отступ абзаца, непосредственно следующего за заголовком первого уровня, можно записать:

```
H1+P {margin-top:0;}
```

Рассмотрим фрагмент кода:

```
<DIV>
  <OL>
    <LI>Первый элемент
    <LI>Второй элемент
    <LI>Третий элемент
  </OL>
Неразмеченный текст
  <UL>
    <LI>Первый элемент
    <LI>Второй элемент
    <LI>Третий элемент
  </UL>
</DIV>
```

Селектор `OL+UL` указывает на маркированный список (списки идут друг за другом и имеют общего родителя — элемент **DIV**). Неразмеченный текст между двумя списками не мешает работать комбинатору сестринского элемента, но если его поместить внутрь элемента **P**, то для доступа к маркированному списку потребовался бы селектор `OL+P+UL` или просто `P+UL`.

Селектор `LI+LI` указывает на второй и третий элементы каждого списка.

Селектор `UL+LI` не связан ни с одним элементом кода, ведь у элементов **`UL`** и **`LI`** разные родители.

Internet Explorer для Windows не понимает селекторы дочерних и сестринских элементов вплоть до версии IE6. IE7 поддерживает оба типа селекторов.

Селекторы атрибутов

Селекторы классов и селекторы идентификаторов позволяют выбирать элементы по значению их атрибутов `class` и `id`.

Так, селектор `.mark` соответствует элементам с атрибутом `class=mark`, а селектор `#header` — элементу (единственному) с атрибутом `id=header`.

Специальные *селекторы атрибутов* служат для выбора элементов на основании *любых* их атрибутов и значений этих атрибутов.

Существует четыре типа селекторов атрибутов. Продемонстрируем их работу на конкретных примерах.

Internet Explorer для Windows не понимает селекторы атрибутов вплоть до версии IE6. IE7 поддерживает этот тип селекторов.

Простой выбор атрибута

Для выбора элементов, в которых присутствует определенный атрибут (независимо от его значения), название атрибута помещается в квадратные скобки:

```
IMG[alt]{border: 1px solid black;}
```

Приведенное выше определение обеспечит рамкой те картинки на странице, у которых в **`IMG`** задан атрибут `alt` (независимо от значения этого атрибута).

Следующее определение позволит выделить рамочкой все элементы, для которых предусмотрена всплывающая подсказка:

```
*[title]{border: 1px solid black;}
```

Можно кодировать выбор на основании наличия не одного, а нескольких атрибутов. В следующем примере рамочкой будут оформлены только те картинки, у которых в элементе **`IMG`** присутствует и атрибут `alt`, и атрибут `title`:

```
IMG[alt][title]{border: 1px solid black;}
```

Выбор по значению атрибута

Для выбора элементов с учетом значения атрибута используется конструкция, в которой явно указывается имя атрибута и его значение:

```
A[href="http://www.botik.ru/~robot/ru/index.htm"]
{font-style:italic;}
```

Все ссылки на страницу Роботландского университета будут выделяться курсивом.

Выбор по частичному значению атрибута

Выбор элементов на основании частей значений атрибутов назначается при помощи ключевого символа, расположенного перед знаком равенства:

Ключ	Пример	Что означает	Что подходит
~	[attr~="val"]	Выбирает любой элемент, значение атрибута attr которого содержит "val" в виде отделенного пробелами слова	<**ELEMENT** attr="area val">
^	[attr^="val"]	Выбирает любой элемент, значение атрибута attr которого начинается с "val"	<**ELEMENT** attr="value">
$	[attr$="val"]	Выбирает любой элемент, значение атрибута attr которого заканчивается "val"	<**ELEMENT** attr="interval">
*	[attr*="val"]	Выбирает любой элемент, значение атрибута attr которого содержит подстроку "val"	<**ELEMENT** attr="equivalent">

Приведенное ниже определение обеспечит выделение курсивом всех внешних ссылок:

```
A[href^="http://"]{font-style:italic;}
```

Специальный выбор атрибута

Этот тип селектора использует ключевой символ “|”:

```
*[lang|="en"]{color:white;}
```

Будут выбраны все элементы, чьи атрибуты `lang` совпадают с `"en-"` или начинаются с `"en-"`. В следующем коде первые три элемента будут выбраны, а последние два — нет:

```
<H1 lang="en">…</H1>
<P lang="en-us">…</P>
<DIV lang="en-au">…</DIV>
<BLOCKQUOTE lang="ru">…</BLOCKQUOTE>
<H4 lang="cy-en">…</H4>
```

Псевдоклассы и псевдоэлементы

Интерфейс гипертекстового документа выигрывает, если вид ссылки меняется в зависимости от разных событий:

- расположен над ссылкой курсор или нет;
- выполнялся ли щелчок на ссылке ранее;
- выполняется ли на ссылке щелчок в данный момент.

Обычными средствами задать такие свойства невозможно. Если мы присвоим, например, ссылке класс `link`:

```
<A class=link href="…">…</A>
```

— то после щелчка класс `link` не может сам измениться на другой класс `visited`!

Именно по этой причине были придуманы "виртуальные" классы (псевдоклассы), которые позволяют выбирать элементы в зависимости от их текущего состояния.

Кроме того, в CSS используются "виртуальные" элементы (псевдоэлементы), которые позволяют ссылаться на элементы, которых нет в HTML-разметке, например, на первую букву или первую строку.

Ключевые слова псевдоклассов и псевдоэлементов начинаются с двоеточия.

Селекторы псевдоклассов

Для управления стилем гипертекстовых ссылок CSS предлагает *псевдоклассы*, представленные в таблице. Предпоследняя колонка содержит эквивалентные атрибуты элемента **`BODY`**.

Атрибуты элемента **`BODY`** позволяют задавать *только* цвета, в то время как псевдоклассы можно использовать для определения *любых* свойств.

Порядок, в котором идут определения стилей ссылок, важен. Он должен подчиняться правилу LVHA: `link`, `visited`, `hover`, `active`.

Почему так? Потому что эти правила работают вместе и могут конфликтовать. При щелчке на непосещенной ссылке, например, последняя сопо-

Псевдокласс	Пример определения	Атрибут BODY	Комментарий
A:link Задает стиль неотработанной ссылки	A:link{color:blue}	link=blue	Цвет неотработанной ссылки
A:active Задает стиль активной ссылки	A:active{color:red}	alink=red	Цвет активной ссылки
A:visited Задает стиль отработанной ссылки	A:visited{color:purple}	vlink=purple	Цвет отработанной ссылки
A:hover Задает стиль ссылки, над которой расположен курсор	A:hover{color:red}	нет	Цвет ссылки, над которой расположен курсор

ставляется с тремя правилами — A:link, A:hover и A:active. Побеждает то правило, которое идет последним.

Пусть, например, определения записаны в другом порядке:

```
A:active  { ... }
A:hover   { ... }
A:link    { ... }
A:visited { ... }
```

При таком расположении определений ни одна из ссылок никогда не продемонстрирует стили A:hover и A:active, потому что всегда будут побеждать правила A:link и A:visited, которые идут последними.

Еще один псевдокласс :focus применяют обычно к элементам форм. Он позволяет ссылаться на элемент, которому в настоящий момент принадлежит фокус ввода.

В соответствии со стандартом псевдоклассы :focus, :hover и :active можно применять к любым элементам HTML.

Приведенное ниже определение заставит любой элемент в блоке **BODY** менять цвет фона на желтый, если над ним в данный момент расположен курсор мыши.

```
BODY *:hover {background:yellow;}
```

Internet Explorer для Windows вплоть до версии IE6 не позволяет псевдоклассам выбирать какие-либо элементы, кроме гиперссылок. В IE7 добавлена поддержка :hover для всех элементов.

Псевдокласс :first-child предназначен для выбора элементов, являющихся первыми дочерними элементами других элементов.

Рассмотрим фрагмент кода:

```
<DIV>
  <P>Для перетаскивания объекта:</P>
  <UL>
    <LI>расположите над объектом курсор;
    <LI>нажмите и <STRONG>не отпускайте</STRONG>
        левую кнопку мыши;
    <LI>перемещайте объект на новое место.
  </UL>
  <P>При перемещении важно <EM>не</EM> отпускать
     кнопку мыши.</P>
</DIV>
```

В данном примере первые дочерние элементы — это первый **P**, первый **LI** и элементы **STRONG** и **EM**.

Зададим стилевые определения:

```
P:first-child {background:yellow;}
LI:first-child {color:red;}
```

Первое правило изменит фон на желтый у любого элемента **P**, который является первым дочерним элементом другого элемента (в нашем случае это первый **P**). Второе правило будет соотнесено с первым LI приведенного HTML-фрагмента.

Internet Explorer для Windows вплоть до версии IE6 не поддерживает :first-child. В IE7 этот псевдокласс работает.

Селекторы псевдоэлементов

Псевдоэлементы вводят в документ фиктивные элементы, что позволяет связывать с ними стилевые правила.

Первая буква

```
P:first-letter {color:red;}
```

Согласно этому правилу первая буква каждого абзаца будет окрашена в красный цвет.

Первая строка

```
P:first-line {color:red;}
```

Согласно этому правилу первая строка каждого абзаца будет записана красными буквами.

Ограничения

В приведенной ниже таблице приведены свойства, которые можно использовать при написании стилевых правил для псевдоэлементов :first-letter и :first-line:

:first-letter	:first-line
Все свойства font	Все свойства font
color	color
Все свойства background	Все свойства background
Все свойства margin	
Все свойства padding	
Все свойства border	
text-decoration	text-decoration
vertical-align (но не для плавающих блоков)	vertical-align
text-transform	text-transform
line-height	line-height
float	
letter-spacing	letter-spacing
word-spacing	word-spacing

Псевдоэлементы должны размещаться в самом конце селектора. Например, запись P:first-line EM неверная, а запись BLOCKQUOTE P:first-line — правильная.

Комбинирование селекторов

Селекторы можно комбинировать, образуя сложные выражения для отбора нужных элементов.

Рассмотрим несколько примеров.

```
HTML>BODY .area
{
  width:400px;
  height: 200px;
}
```

Это определение относится ко всем элементам, в открывающем теге которых задано class=area, которые являются любыми потомками элемента **BODY**, который, в свою очередь, является прямым потомком элемента **HTML**. Казалось бы, очевидное указание HTML>BODY является лишним. Так оно и есть. Смысл этого селектора — закрыть определение от браузеров IE, версий меньше 7 (они не понимают селектор потомка).

Следующее определение, напротив, поймут только браузеры IE:

```
* HTML .area
{
  width:450px;
  height: 220px;
}
```

По стандарту в корне иерархии элементов страницы расположен элемент **HTML**. Именно так строят объектную модель страницы все браузеры, кроме IE. Браузеры IE располагают в корне некий безымянный элемент, а **HTML** содержится в нем уже как потомок.

Селектор `* HTML .area` определяет элементы с классом `area`, которые являются потомками элемента **HTML**, который, в свою очередь, является потомком какого-либо другого элемента (селектор `*`).

Для любого браузера, кроме IE, селектор `* HTML .area` не указывает ни на один объект, так как нет элементов, у которых потомком был бы **HTML**. Таким образом, этот селектор будет работать только в IE.

Последний пример:

```
.area>BLOCKQUOTE P STRONG+EM
{
  color:red;
}
```

Красным будут покрашены все элементы **EM**, которые идут непосредственно за элементом **STRONG** в абзацах, которые вложены в блок **BLOCKQUOTE**, который является прямым потомком любого элемента с классом `area`.

Наследование стилей

Стили обладают свойством *наследования* — элемент сохраняет (наследует) стили своего родителя. Если записать:

```
BODY {color:red}
```

— то все потомки **BODY** (то есть все элементы гипертекстовой страницы) будут записываться красным цветом.

Не все свойства наследуются, например, не наследуется свойство `border`. В справочниках по свойствам CSS факт наследования всегда отмечается.

Каскадирование стилей

С элементом может быть связано несколько стилевых определений (в том числе в отдельном стилевом файле, в головной части кода и в отдельных тегах).

Принцип обработки множественных определений, заданных для одного элемента, называется *каскадированием*.

Одно из правил каскадирования: конкретное определение главнее общего.

Можно сформулировать и так: стилевое определение потомка отменяет аналогичное определение родителя и передается по наследству.

Сформулируем правила каскадирования более детально.

1. Если для родительского элемента (например, **BODY**) задано стилевое определение, и оно не конфликтует со стилевыми определениями, заданными для потомка (например, для **P**), то стилевое определение родителя наследуется потомком.

2. Стилевое определение потомка (например, **P**) главнее аналогичных определений родителя (например, **BODY**).

3. Стилевое определение в теге главнее стилевого определения в головной части HTML-кода, а последнее главнее определений, заданных в отдельном стилевом файле.

4. Если задано несколько конфликтующих стилевых определений равного каскадного веса, то действует последнее из них. Например, текст в абзаце `<P style="color:red;color:blue">` будет синего цвета.

Следует отметить, что сформулированные выше правила являются упрощенным изложением алгоритма каскада в CSS; ими можно пользоваться в большинстве случаев на практике, но в реальной "игре" участвуют более тонкие алгоритмы, связанные с вычислением *специфичности* селекторов.

Не вдаваясь в подробности, покажем принцип учета специфичности при определении правила-победителя в каскаде правил.

Прежде всего отметим, что специфичность сложного селектора определяется специфичностью входящих в его состав простых селекторов.

Значение специфичности состоит из четырех частей: 0, 0, 0, 0 — и определяется следующим образом:

- Если стилевое определение задано в самом элементе при помощи атрибута `style`, то первый ноль в специфичности заменяется единицей: 1, 0, 0, 0.
- Для каждого селектора идентификатора к специфичности добавляется 0, 1, 0, 0.
- Для каждого селектора класса, псевдокласса или атрибута к специфичности добавляется 0, 0, 1, 0.
- Для каждого селектора элемента или псевдоэлемента к специфичности добавляется 0, 0, 0, 1.
- Комбинаторы и универсальный селектор не учитываются.

Пусть с одним и тем же элементом сопоставлены два определения. Ниже показан способ вычисления победителя в каскадном соревновании правил.

```
H1 {color:red;}         /* 0, 0, 0, 1 */
BODY H1 {color:green;} /* 0, 0, 0, 2 (победитель)*/

H2.mark {color:red;} /* 0, 0, 1, 1 * (победитель)/
H2 {color:silver;}   /* 0, 0, 0, 1 */

HTML>BODY TABLE TR[id="x"] TD UL>LI {color:red;} /* 0, 0, 1, 7 */
LI#answer {color:blue;} /* 0, 1, 0, 1 (победитель)*/
```

Правило-победитель определяется в каждой паре по более высокому показателю специфичности.

Вес компонентов специфичности растет слева направо. Так, специфичность 1, 0, 0, 0 возьмет верх над любой специфичностью, которая начинается с нуля.

Если специфичность селекторов совпадает, то побеждает правило, которое идет позже.

Из приведенных выше правил вычисления специфичности следует, что наибольший каскадный вес имеют правила, записанные в самом элементе при помощи атрибута `style`.

Селектор идентификатора весомее селектора класса, псевдокласса или атрибута. Последние весомее селектора элемента или псевдоэлемента.

Важность

Если в значение свойства включить ключевое слово `!important` прямо перед закрывающей точкой с запятой, то важность правила превысит все остальные факторы.

```
BODY H1 {color:green;}      /* 0, 0, 0, 2 */
H1 {color:red !important;} /* 0, 0, 0, 1 (победитель)*/
```

Ключевое слово `!important` не дает вклад в вычисление специфичности, такие правила рассматриваются отдельно от остальных. Фактически все правила с `!important` группируются вместе, и конфликты (если они есть) разрешаются на уровне вычисления специфичностей членов созданной группы.

Комментарии в языке CSS

Комментарии в языке CSS записываются при помощи конструкции:

```
/* Это комментарий CSS */
```

Пример стилевого определения с комментариями:

```
/* Выделенная область
   ------------------ */
.def
```

```
{
  font-size: 130%;        /* Повышенный размер шрифта    */
  border: 2px solid red; /* Рамка красного цвета        */
  background: #FFFFCC;    /* Желтый фон                  */
  padding: 10px;          /* Отступ содержимого от рамки */
}
```

Комментарии можно записывать не только в стилевом файле или головной части HTML-кода, но и внутри значения атрибута `style`:

```
<P style="font-famile:sans-serif; /*рубленый шрифт*/">
```

Значения и единицы измерения

Числа

В CSS используются два вида чисел: целые и вещественные. Вещественное число определяется как целое, за которым следуют десятичная *точка* и дробная часть.

И целые, и вещественные числа могут быть как положительными, так и отрицательными.

Примеры целых чисел: 0, 22, –100.

Примеры вещественных чисел: 0.28, 3.14, –100.15.

Свойства могут ограничивать диапазон принимаемых ими чисел.

Проценты

Число, за которым следует знак процента (%).

Проценты вычисляются от значений, указанных в свойствах (например, от размера родительского элемента).

Цвет

В спецификации CSS 2.1 зафиксировано 17 цветов, которые можно задавать ключевыми словами. Это 16 цветов, описанных в HTML 4.01, плюс оранжевый цвет (orange):

aqua	gray	navy	red
black	green	olive	silver
blue	lime	orange	teal
fuchsia	maroon	purple	white
			yellow

В CSS3 определен список ключевых слов для 140 цветов.

Цвет можно задавать при помощи RGB-кода в следующих форматах:

```
rgb(число, число, число)
rgb(процент, процент, процент)
```

Допустимый диапазон процентных значений — от 0 до 100%, а допустимый диапазон чисел — от 0 до 255.

Так белый и черный цвета можно закодировать следующим образом:

```
rgb(255, 255, 255)
rgb(0, 0, 0)
```

Или так:

```
rgb(100%, 100%, 100%)
rgb(0%, 0%, 0%)
```

RGB-цвет можно задавать шестнадцатеричной записью в формате:

```
#RRGGBB
```

или

```
#RGB
```

— если для каждой RGB-составляющей используются одинаковые цифры. Так, например, запись `#77FF00` эквивалентна записи `#7F0`.

Примеры:

Ключевое слово	Проценты	Числа	16-ричная запись	Краткая 16-ричная запись
red	rgb(100%, 0%, 0%)	rgb(255, 0, 0)	#FF0000	#F00
orange	rgb(100%, 40%, 0%)	rgb(255, 102, 0)	#FF6600	#F60
green	rgb(0%, 50%, 0%)	rgb(0, 128, 0)	#008000	
gray	rgb(50%, 50%, 50%)	rgb(128, 128, 128)	#808080	

Размер

Абсолютные единицы

Эти единицы размера практически не используются в разработке веб-страниц.

Обозначение	Название	Описание
in	Дюймы	Задает размер в дюймах. Один дюйм равен 25,4 миллиметра. Для буквы Ш указано правило `font-size:1in`. Ш
cm	Сантиметры	Задает размер в сантиметрах. Один сантиметр равен 0,394 дюйма. Для буквы Ш указано правило `font-size:1cm`. Ш

mm	Милли-метры	Задает размер в миллиметрах. Для буквы Ш указано правило `font-size:20mm`. 
pt	Пункты	Задает размер в типографских пунктах. В одном дюйме 72 пункта. Для буквы Ш указано правило `font-size:72pt`. 
pc	Пики	Задает размер в типографских пиках. Одна пика — 12 пунктов, то есть в одном дюйме 6 пик. Для буквы Ш указано правило `font-size:6pc`. 

Относительные единицы

Измеряемый в этих единицах фактический размер меняется под действием внешних факторов, таких, как разрешение экрана, ширина окна браузера, размер шрифта.

Единицы измерения em

В CSS один “em” совпадает со значением свойства `font-size` текущего шрифта.

Пусть, например, для элементов **`H1`**, **`H2`** и **`P`** задан шрифт, соответственно, в 24, 18 и 12 пикселей:

```
H1 {font-size:24px;}
H2 {font-size:18px;}
P  {font-size:12px;}
```

Определим теперь для этих элементов внешний отступ слева в `1em`:

```
H1, H2, P {margin-left:1em;}
```

Получится, что отступ для **`H1`** будет равен 24 пикселям, для **`H2`** —18 пикселям, а для **`P`** —12 пикселям.

При вычислении размера шрифта значение `em` вычисляется относительно размера родительского элемента.

Пусть CSS-определение:

```
BODY {font-size:1em;}
H1 {font-size:1.5em;}
EM {font-size:0.8em;}
```

применяется к HTML-коду:

```
<BODY>
  <H1>Заголовок</H1>
  <P>
Текст абзаца, в котором есть выделенный <EM>фрагмент</EM>.
</BODY>
```

Для **BODY** задан шрифт размером в `1em`. Это означает, что на странице будет использоваться размер шрифта, установленный в браузере по умолчанию. Этот размер наследует шрифт в элементе **P**, но для потомка **EM** шрифт будет иметь размер на 20% меньше. Напротив, элемент **H1** будет записываться шрифтом, размер которого на 50% больше размера шрифта, заданного для **BODY** (то есть установленного в браузере по умолчанию). Если пользователь теперь изменит размер шрифта в браузере, на экране все перестроится пропорционально.

Единицы измерения ex

Величина `ex` опирается на высоту английской буквы "x" нижнего регистра текущего шрифта.

Проблема в том, что высота буквы "x" в разных шрифтах разная, даже при одном шрифтовом размере. Кроме того, браузеры вычисляют эту величину тоже по-разному, поэтому единицу `ex` разработчики используют крайне редко.

Измерение в пикселях

Пиксели отнесены к относительным единицам, потому что размер пикселя зависит от экранного разрешения и размеров монитора.

На одном мониторе при фиксированном разрешении пиксели ведут себя так же, как сантиметры, то есть представляют собой абсолютную единицу.

Если размер шрифта задан в пикселях, пользователи IE вплоть до версии IE6 не смогут менять размер текста в браузере, и это очень плохо с точки зрения добропорядочного интерфейса. В других современных браузерах (IE7 и остальных) такой проблемы нет.

Измерения в пикселях идеальны для растровых картинок.

Ключевые слова

Ключевые слова определяются при описании стилевых свойств. Например, популярно ключевое слово `none`. Чтобы удалить подчеркивание ссылок, можно написать:

```
A {text-decoration:none;}
```

Одно и то же ключевое слово может обозначать разные сущности у разных свойств. Так слово `normal` для свойства `letter-spacing` обозначает совершенно другое по сравнению с этим же ключевым словом для свойства `font-size`.

Ключевое слово `inherit` используется всеми свойствами одинаково: оно делает значение свойства таким же, как у родительского элемента.

Стилевые свойства

На практике разработчику необходимо иметь под рукой хороший справочник по свойствам CSS, лучше в электронном виде, чтобы можно было копировать свойства и их значения в свои коды.

Можно порекомендовать онлайн-справочник Влада Мержевича на странице: *http://htmlbook.ru/css*.

Здесь же можно купить этот справочник в формате CHM, HTML или PDF (по цене 52 руб.).

Проверка (валидация) стилевых определений

Проверить правильность построенных стилевых определений можно при помощи онлайн-валидатора от W3C: *http://jigsaw.w3.org/css-validator*.

Предельно краткий справочник свойств CSS

В этом кратком справочнике свойства CSS сгруппированы по категориям. Для каждого свойства указаны:

- название;
- множество допустимых значений;
- значение по умолчанию (если есть; выделено полужирным или указано отдельно);
- элементы документа (теги), к которым данное свойство применимо (если не указано иное, считается, что свойство применимо ко всем элементам).

Символ «**¥**» указывает на то, что данное свойство элемент *не наследует* от родительского.

Обозначения

`a b` — *пробел используется для перечисления значений стилевых свойств;*

`[a b]` — *квадратные скобки используются для группировки значений стилевых свойств;*

`a | b` — *символ "|" имеет (как обычно) смысл "или";*

`a || b` — *символ "||" имеет значение "a или b, или оба данных значения, перечисленные в любом порядке";*

`a?` — *значение a может быть указано, а может и отсутствовать;*

`a*` — *ни одного или несколько значений a;*

`a+` — *одно или несколько значений a;*

`a{1, 4}` — *от одного до четырех значений a;*

`a{n}` — *n-е значение a.*

Границы, позиционирование, поля и отступы

`margin-top` (¥)

Значение: <абсолютное значение> | <значение в процентах> | `auto`

Значение по умолчанию: 0

Если указано значение в процентах, то проценты берутся от ширины родительского элемента

`margin-right` (¥)

Значение: <абсолютное значение> | <значение в процентах> | `auto`

Значение по умолчанию: 0

Если указано значение в процентах, то проценты берутся от ширины родительского элемента

`margin-bottom` (¥)

Значение: <абсолютное значение> | <значение в процентах> | `auto`

Значение по умолчанию: 0

Если указано значение в процентах, то проценты берутся от ширины родительского элемента

`margin-left` (¥)

Значение: <абсолютное значение> | <значение в процентах> | `auto`

Значение по умолчанию: 0

Если указано значение в процентах, то проценты берутся от ширины родительского элемента

margin (¥)

Значение: <абсолютное значение> | <значение в процентах> | auto {1, 4}

Значение по умолчанию: не определено

Если указано значение в процентах, то проценты берутся от ширины родительского элемента

padding-top (¥)

Значение: <абсолютное значение> | <значение в процентах>

Значение по умолчанию: 0

Если указано значение в процентах, то проценты берутся от ширины ближайшего блочного родительского элемента

padding-right (¥)

Значение: <абсолютное значение> | <значение в процентах>

Значение по умолчанию: 0

Если указано значение в процентах, то проценты берутся от ширины ближайшего блочного родительского элемента

padding-bottom (¥)

Значение: <абсолютное значение> | <значение в процентах>

Значение по умолчанию: 0

Если указано значение в процентах, то проценты берутся от ширины ближайшего блочного родительского элемента

padding-left (¥)

Значение: <абсолютное значение> | <значение в процентах>

Значение по умолчанию: 0

Если указано значение в процентах, то проценты берутся от ширины ближайшего блочного родительского элемента

padding (¥)

Значение: <абсолютное значение> | <значение в процентах> {1, 4}

Значение по умолчанию: не определено

Если указано значение в процентах, то проценты берутся от ширины ближайшего блочного родительского элемента

border-top-width (¥)

Значение: thin | **medium** | thick | <абсолютное значение>

border-right-width (¥)

Значение: thin | **medium** | thick | < абсолютное значение >

border-bottom-width (¥)

Значение: thin | **medium** | thick | < абсолютное значение >

border-left-width (¥)

Значение: thin | **medium** | thick | < абсолютное значение >

border-width (¥)

Значение: thin | medium | thick | < абсолютное значение > {1, 4}

Значение по умолчанию: не определено

border-color (¥)
Значение: <color>{1,4}
Значение по умолчанию: берется значение свойства color данного элемента

border-style (¥)
Значение: **none** | dotted | dashed | solid | double | groove | ridge | inset | outset

border-top (¥)
Значение: <border-top-width> || <border-style> || <color>
Значение по умолчанию: не определено

border-right (¥)
Значение: <border-right-width> || <border-style> || <color>
Значение по умолчанию: не определено

border-bottom (¥)
Значение: <border-bottom-width> || <border-style> || <color>
Значение по умолчанию: не определено

border-left (¥)
Значение: <border-left-width> || <border-style> || <color>
Значение по умолчанию: не определено

border (¥)
Значение: <border-width> || <border-style> || <color>
Значение по умолчанию: не определено

width (¥)
Значение: <абсолютное значение> | <значение в процентах> | **auto**
Применяется к блочным и замещаемым элементам
Если указано значение в процентах, то проценты берутся от ширины ближайшего блочного родительского элемента

height (¥)
Значение: <абсолютное значение> | **auto**
Применяется к блочным и замещаемым элементам

float (¥)
Значение: left | right | **none**

clear (¥)
Значение: **none** | left | right | both

visibility (¥)
Значение: visible | hidden | collapse | **inherit**

position (¥)
Значение: static | relative | absolute | fixed | **inherit**
*Применяется ко всем элементам, за исключением генерируемых**

* Генерируемыми называются визуализируемые браузером элементы, которым непосредственно не соответствует ни какой-либо тег, ни элемент содержания страницы. Хороший пример: маркер списка.

top (¥)

Значение: <абсолютное значение> | <значение в процентах> | **auto** | inherit

Применяется к позиционируемым элементам

Если указано значение в процентах, то проценты берутся от высоты родительского элемента

right (¥)

Значение: <абсолютное значение> | <значение в процентах> | **auto** | inherit

Применяется к позиционируемым элементам

Если указано значение в процентах, то проценты берутся от ширины родительского элемента

bottom (¥)

Значение: <абсолютное значение> | <значение в процентах> | **auto** | inherit

Применяется к позиционируемым элементам

Если указано значение в процентах, то проценты берутся от высоты родительского элемента

left (¥)

Значение: <абсолютное значение> | <значение в процентах> | **auto** | inherit

Применяется к позиционируемым элементам

Если указано значение в процентах, то проценты берутся от ширины родительского элемента

z-index (¥)

Значение: **auto** | < абсолютное значение > | **inherit**

Применяется к позиционируемым элементам

overflow (¥)

Значение: **visible** | hidden | scroll | auto | inherit

Применяется к блочным и замещаемым элементам

Форматирование, списки

display (¥)

Значение: **block** | inline | list-item | none

white-space

Значение: **normal** | pre | nowrap

list-style-type

Значение: **disc** | circle | square | decimal | lower-roman | upper-roman | lower-alpha | upper-alpha | none

Применяется к элементам, для которых свойство display *имеет значение* list-item

list-style-image

Значение: <url> | **none**

Применяется к элементам, для которых свойство display *имеет значение* list-item

list-style-position

Значение: inside | **outside**

Применяется к элементам, для которых свойство display *имеет значение* list-item

list-style

Значение: <list-style-type> || <list-style-position> || <list-style-image>

Значение по умолчанию: не определено

Применяется к элементам, для которых свойство display *имеет значение* list-item

Цвет и фон

background-color (¥)

Значение: <color> | **transparent**

background-image (¥)

Значение: <url> | **none**

background-repeat (¥)

Значение: **repeat** | repeat-x | repeat-y | no-repeat

background-attachment (¥)

Значение: **scroll** | fixed

background-position (¥)

Значение: [<значение в процентах> | <абсолютное значение>]{1,2} | [top | center] || [left | center | right]

Значение по умолчанию: 0% 0%

Применяется к блочным и замещаемым элементам

Если указано значение в процентах, то проценты берутся от размеров самого элемента

background (¥)

Значение: <background-color> || <background-image> <background-repeat> || <background-attachment> <background-position>

Значение по умолчанию: не определено

color

Значение: <цвет>

Значение по умолчанию: определяется браузером

Свойства текста

font-family

Значение: [[<имя шрифта> | <род шрифта>],]* [<имя шрифта > | <род шрифта>]

Значение по умолчанию: определяется браузером

<род шрифта>

serif | sans-serif | cursive | fantasy | monospace

font-style
Значение: **normal** | italic | oblique

font-variant
Значение: **normal** | small-caps

font-weight
Значение: **normal** | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900

font-size
Значение: <абсолютное значение> | <относительное значение>
Если указано относительное значение, то оно берется от соответствующего свойства родительского элемента
<абсолютное значение>
xx-small | x-small | small | **medium** | large | x-large | xx-large
<относительное значение>
larger | smaller

font
Значение: [<font-style> || <font-variant> || <font-weight>]? <font-size> [/ <line-height>]? <font-family>
Значение по умолчанию: не определено

word-spacing
Значение: **normal** | <абсолютное значение>

letter-spacing
Значение: **normal** | <абсолютное значение>

text-decoration (¥)
Значение: **none** | [underline || overline || line-through || blink]

vertical-align (¥)
Значение: **baseline** | sub | super | top | text-top | middle | bottom | text-bottom | <значение в процентах>
Применяется к встроенным элементам
Если указано значение в процентах, то проценты берутся от значения свойства line-height самого элемента

text-transform
Значение: capitalize | uppercase | lowercase | **none**

text-align
Значение: left | right | center | justify
Значение по умолчанию: определяется браузером
Применяется к блочным элементам

text-indent
Значение: <абсолютное значение> | <значение в процентах>
Значение по умолчанию: 0
Применяется к блочным элементам
Если указано значение в процентах, то проценты берутся от ширины родительского элемента

line-height

Значение: **normal** | <абсолютное значение> | <значение в процентах>

Если указано значение в процентах, то проценты берутся от значения свойства font-size самого элемента

Псевдо-классы и псевдо-элементы

Ссылочные псевдо-классы

```
A:link /* не посещенные ссылки */
A:visited /* посещенные ссылки */
A:active /* активные ссылки */
```

Особые элементы параграфов

```
P:first-line /* первая строка параграфа*/
P:first-letter /* первая буква параграфа */
```

Величины

Абсолютные значения

<абсолютное значение>

(+ | -)? <число> <единица измерения>

<число>

<цифра>+[. <цифра>*]?

<единица измерения>

<абсолютная единица измерения> | <относительная единица измерения>

<абсолютная единица измерения>

mm | cm | in | pt | pc

<относительная единица измерения>

em | ex | px

Значения в процентах

<значение в процентах>

<число>%

Кодирование цвета

<цвет>

<имя> | <RGB-код>

<имя>

aqua | black | blue | fuchsia | gray | green | lime | maroon | navy | olive | purple | red | silver | teal | white | yellow

<RGB-код>

#<шестн. цифра>{3}| #<шестн. цифра>{6}|rgb(<число 0–255>, <число 0–255>, <число 0–255>) | rgb(<процент красного> <процент зеленого>, <процент синего>)

URL

<url>

url(путь)

УДК 372.800.2
ББК 74.263.2
Д79

Общая редакция серии “Информатика”: *С.Л. Островский*

Дуванов А.А.

Д79 CSS: кратко о самом главном / А.А. Дуванов. – М. : Чистые пруды, 2009. – 32 с. – (Библиотечка “Первого сентября”, серия “Информатика”. Вып. 28).

ISBN 978-5-9667-0591-6

В брошюре в кратком, но систематическом виде представлена информация о языке CSS. Сегодня “чистый” HTML — уже большая редкость. Вне зависимости от масштабов проекта — идет ли речь о целом школьном сайте или одной страничке, использование возможностей CSS будет как минимум полезно, как максимум — необходимо.

УДК 372.800.2
ББК 74.263.2

Учебное издание

ДУВАНОВ Александр Александрович

CSS: кратко о самом главном

Редактор *С.Л. Островский*
Корректор *Е.Л. Володина*
Компьютерная верстка *Н.И. Пронская*

Свидетельство о регистрации СМИ ПИ № ФС77–19078 от 08.12.2004 г.

Подписано в печать 26.05.2009.
Формат 60x90$^{1}/_{16}$. Гарнитура “Таймс”. Печать офсетная. Печ. л. 2,0.

Тираж экз. Заказ №

ООО “Чистые пруды”, ул. Киевская, 24, Москва, 121165
Тел. (499) 249-28-77, http://www.1september.ru

Отпечатано с готовых диапозитивов в филиале ГУП МО «КТ» «Раменская типография»
Сафоновский пр., д. 1, г. Раменское, МО, 140100
Тел. (495) 377-07-83. E-mail: ramentip@mail.ru

ISBN 978-5-9667-0591-6