# УРОК №3. Основы CSS

## Содержание урока

- Что такое CSS
- Синтаксис CSS
- Способы объявления CSS
- Селекторы (id, class, tag)
- Селекторы атрибутов
- Основные свойства стилей
- Вложенность, наследование и группирование свойств
- Приоритеты применения стилей
- Псевдоклассы и псевдоэлементы

## *Что такое CSS*

CSS (Cascading Style Sheets) – это каскадные листы стилей, которые применяются для описания внешнего вида веб-документа, написанного при помощи языка разметки HTML.
Другими словами, с помощью HTML появляется структура документа, а CSS - это уже его оформление. При помощи CSS можно менять цвета, шрифты у текста, изменять положение элементов на странице, их размеры, задавать элементам рамки, границы и отступы.
Раньше, когда не было CSS, документы оформляли при помощи атрибутов, но у такого способа очень ограниченные возможности, поэтому сайты в то время были скучные и однотипные. Дальше приводится пример оформления веб-документа без использования CSS.

```
<body bgcolor="green">
    <h1 align="center">
        <font color="red">Заголовок</font>
    </h1>
</body>
```

Видно, что в этом примере, при помощи атрибута bgcolor тега body, задается фон у всего документа. Заголовок первого уровня выравнивается посередине документа при помощи атрибута align. Для того чтобы заголовок был красного цвет, необходимо вложить еще один тег font, и у него в атрибуте color указать красный цвет. С появлением CSS данным способом веб-документы уже никто не оформляет, и все приведенные атрибуты тегов считаются устаревшими.

## *Синтаксис CSS*

```
Селектор {
    свойство1: значение1;
    свойство2: значение2;
}
```

Сначала обязательно указывается селектор, в роли которого могут выступать теги, идентификаторы, классы, атрибуты тегов. В фигурных скобках указываются свойства данного селектора в виде пары название свойства и через двоеточие его значение. После каждой пары ставится точка с запятой, которая свидетельствует о разделении свойств. Если после последней пары свойство-значение, либо если эта пара одна, не поставить точку с запятой, то ошибки не будет, но возьмите в привычку всегда ставить этот знак, так вы просто не будете про него забывать.

### Оформление CSS

*1-й способ*

```
Селектор { свойство1: значение1; свойство2: значение2; }
```

*2-й способ*

```
Селектор
{
    свойство1: значение1;
    свойство2: значение2;
}
```

*3-й способ*

```
Селектор {
    свойство1: значение1;
    свойство2: значение2;
}
```

Первый способ, записывать все свойства в одну строку, но этот способ не очень удобен, т.к. свойств у селектора может быть много, они не уместятся на экран редактора, соответственно появиться горизонтальная полоса прокрутки, и ваш лист стилей неудобно будет читать. Лучше использовать второй или третий способ оформления.

### Комментарии в CSS

```
/* здесь пишем комментарий */

/*
и здесь можно
тоже
его указать
*/
```

В CSS можно также указывать комментарии, для комментирования того, свойства, каких элементов будут описываться, или можно комментировать сами стили при редактировании документа.

## *Способы объявления ссс*

Для того, чтобы использовать стили CSS в веб-документе, необходимо их сначала подключить. Для этого существует три способа.

### Inline-стили

```
<div style="свойство1: значение1; свойство2: значение2;">

<body style="background: #0f0;" >
    <h1 style="text-align: center; color: #f00;">
        Заголовок
    </h1>
</body>
```

Для подключения CSS этим способом в HTML существует тег style, который можно указывать практически у любого HTML-тега. В значении атрибута style перечисляются стили в том же формате, свойство и его значение.

## Стили в разделе head

```
<head>
<style type="text/css">
    body{
        свойство1: значение1;
        свойство2: значение2;
  }
</style>
</head>
<body>
    …
</body>
```

Для того, чтобы подключить стили этим способом существует HTML тег style. У него в атрибуте type указываем mime тип данных, в данном случае это text/css. Внутри этого тега мы уже прописываем стили, которые будут действовать для всей данной страницы.

## Внешний CSS файл

Создаем файл с расширением .css. Обычно, так же как и в случае с картинками, все css файлы размещают в отдельной папке.

**css/style.css**

```
body{
    свойство1: значение1;
    свойство2: значение2;
}
```

А в нужном HTML файле этот файл подключаем.

**index.html**

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

Для подключения CSS файла используется тег link, который помещается в раздел head нужного HTML файла. И для того чтобы правильно подключить файл стилей, у тега link нужно указать несколько атрибутов. В атрибуте rel указывается значение stylesheet, т.е. лист стилей, это нужно для того, чтобы браузел понимал, что подключается файл стилей CSS. В атрибуте href указывается путь к CSS файлу, причем так же как и в случае с гиперссылками, этот путь может быть как относительным, так и абсолютным. Указывается атрибут type со значением text/css.

### Какой способ подключения стилей выбрать?

Плюс inline-стилей в том, что можно быстро прописать какой-нибудь простой стиль для элемента. Например, какому-нибудь слову в тексте задать простой стиль, например, выделить красным цветом. Недостатки такого подхода в том, что для каждого тега необходимо прописывать стили, например, у нас несколько параграфов, и все они должны быть определенного стиля, и тогда каждому параграфу нужно прописывать этот стиль. И еще один существенный недостаток в том, что при таком подходе, стили сложно редактировать. Что делать, если нужно будет в проекте поменять размер шрифта во всех параграфах?
При втором способе уже можно прописывать стили для нескольких элементов, только все стили будут применяться в пределах одного документа. В этом и состоит главный минус этого подхода. Получается, что если на сайте большое количество страниц, и у нас задача, поменять тот же цвет или размер шрифта всех параграфов, соответственно нужно открывать каждую страницу.
Если подключить отдельный файл, то он будет действовать на все страницы, где мы подключим данный файл. И тогда, для того, чтобы изменить цвет или размер шрифта всех параграфов, нужно будет изменить его один раз в одном месте. И еще одно преимущество в том, что браузер кэширует файл стилей, т.е. другими словами, сохраняет его у себя в памяти, чтобы не обращаться при каждом запросе на сервер.

## *Селекторы в CSS*

### Селекторы тегов

```
h1 {
    background: #ccc;
}


<h1>
    Для всех заголовков первого уровня цвет фона будет серым
</h1>
```

При использовании селекторов тегов стиль будет применяться ко всем указанным тегам. В качестве селектора указывается название любого HTML тега.

## Селекторы идентификаторов

```
#first {
    color: red;
}


<p id="first">
    цвет этого абзаца будет красным
</p>
```

В качестве селекторов, можно использовать идентификаторы. Определенному тегу в значении атрибута id указывается название, которое придумываем сами, а в селекторе, ставиться знак #, а затем это название. Очень важно запомнить следующее, идентификатор должен быть уникальным, т.е. нельзя задавать одно и тоже имя двум и более элементам.

## Селекторы классов

```
.is_border {
    border: 1px solid #000;
}
<h1 class="is_border">
    Заголовок с рамкой
</h1>
<p class="is_border">
    параграф с рамкой
</p>
```

Классы используются аналогично id, только вместо атрибута id, указывается атрибут class, а в селекторе вместо решетки, точка. Классы отличаются от идентификаторов тем, что применять один и тот же стиль к разным элементам.

## Селекторы атрибутов

В качестве селекторов можно указывать атрибуты HTML тегов. Существует множество различных способов указывать селекторы атрибутов, но для начала мы не будем их все рассматривать, рассмотрим 2 примера. Остальные способы вы сможете найти в справочнике, если они вам понадобятся.

```
img[title]{
    width: 200px;
}
input[type="text"]{
    font-size: 18px;
}
<img src="picture.png" title="картинка" />
<input type="text" />
```

В данном примере, сначала указывается стиль для всех картинок, у которых присутствует атрибут title, для этого название атрибута указывается в квадратных скобках, сразу после названия тега.
Второй пример. Стиль будет применяться для всех тегов <input />, в значении атрибута type которого, присутствует значение text, т.е. для всех обычных текстовых полей ввода.

# *Свойства стилей*

## Единицы измерения в CSS

В CSS существует достаточное количество единиц измерения, с помощью которых, можно определять длину, ширину элементов, а также размеры шрифтов. Но не все они используются в повседневной верстке, но вам нужно иметь представления о них. Единицы измерения подразделяются на относительные и абсолютные. Относительными единицами измерения называются единицы которые могут изменятся в зависимости от различных факторов. К таким факторам относятся: разрешение монитора пользователя, ширина области просмотра (окна браузера), различные настройки пользователя. Относительные единицы измерения наиболее часто используются на веб-страницах.

**Относительные единицы измерения**

- ✓ px - пиксел
- ✓ % - процент
- ✓ em – высота текущего шрифта
- ✓ ex – высота буквы x

К абсолютным единицам относятся единицы измерения, которые используются в обычной жизни. Но они в веб-страницах используются достаточно редко, поэтому их использовать крайне нежелательно.

**Абсолютные единицы измерения**

- ✓ cm - сантиметр
- ✓ mm – миллиметр
- ✓ in - дюйм
- ✓ pt - пункт
- ✓ pc - пика

## Способы задания цветов

Цвета в CSS можно задавать 3 различными способами. Первый способ, это задавать цвета используя названия цветов на английском языке, например: red, green, blue, black, yellow, white и т.д. Но при таком подходе есть ограничения в выборе цвета, невозможно получить различные оттенки цветов. Для того, чтобы можно было выбрать один из более, чем 16 млн. цветов нужно использовать способ выбора цвета либо как функциональный RGB, либо шестнадцатиричный RGB. RGB – это аббревиатура, и расшифровывается она как Red-Green-Blue, то есть Красный-Зеленый-Синий. Таким образом, любой цвет можно получить, смешав эти три цвета.

**Функциональный RGB**

RGB(255, 130, 0)  0 - 255
RGB(100%, 70%, 0%)  0 – 100%

В этом случае, выбирается насыщенность любого из трех цветов в диапазоне от 0 - 255, или в процентах от 0 - 100%, и используется значение RGB, где в скобках, через запятую, перечисляются насыщенность каждого из 3-х цветов.

**Шестнадцатиричный RGB**

#FA96CF;  0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
#FFAA00 => #FA0

Если использовать шестнадцатиричный RGB, то здесь каждый цвет можно представить в виде пары значений, т.е. первый и второй символ - это красный цвет, третий и четвертый - это зеленый, пятый и шестой - синий. А каждый символ может быть представлен одним из шестнадцати знаков, от 0 до буквы F латинского алфавита. При шестнадцатиричном rgb перед кодом цвета ставится символ решетки, а дальше уже записывается сам код цвета. Если совпадают две буквы или цифры одного и того же цвета, то можно использовать сокращенную форму записи, т.е. каждый цвет будет состоять не из пары значений, а из одного значения, и в коде цвета будет 3 символа после знака решетки.
При подборе цвета на первое время лучше использовать любой графический редактор, там можно выбрать любой нужный вам цвет из палитры, и редактор покажет код выбранного цвета, который можно скопировать и вставить на страницу.

А теперь, рассмотрим сами свойства стилей CSS.

## Фон элемента - background

```
background-color: #ff0;
background-image: url(img/foto.jpg);
background-position: top; (bottom | left | right)
background-repeat: repeat-x; (repeat-y | no-repeat)
background-attachment: fixed;
```

*background-color* - задает цвет фона, который можно задавать любым из трех способов задания цветов.
*background-image* используется для того, чтобы в качестве фона можно было установит изображение. Для этого необходимо в значении свойства указать путь к изображению в скобках url.
*background-position* - указывает где будет располагаться фоновое изображение. Может иметь значения: top, bottom, left, right.
*background-repeat* определяет, нужно ли повторять фоновое изображение. repeat-x - изображение повторяется по горизонтали, repeat-y - по вертикали, no-repeat - изображение не повторяется. По умолчанию у этого свойства установлено значение repeat, что означает, что изображение будет повторятся по горизонтали и по вертикали.
*background-attachment* - определяет, будет ли изображение прокручиваться вместе с содержимым элемента. По умолчанию, оно установлено как scroll, что означает, что изображение будет прокручиваться, а при значении fixed, изображение будет оставаться неподвижным.
Существует короткая форма записи, можно записать все перечисленные значения в одну строку, разделяя значения пробелом, порядок следования значений не важен. Если пропускать какие-либо значения, то тогда будут подставляться значения по умолчанию.

```
background: #ff0 url(img/foto.jpg) top repeat-x;
```

## border – рамка вокруг элемента

```
border-color: red; (#f00 | RGB(255, 0, 0))
border-style: solid; (dotted | solid | double | inset | outset)
border-width: 2px;
```

border тоже подразделяется на различные свойства.
*border-color* - цвет рамки.

*border-style* - стиль рамки, которая может быть разных значений, такие как: dotted, dashed, solid, double, groove, ridge, inset, outset.
*border-width* задает толщину рамки, причем ее можно задать для каждой из 4 сторон отдельно:

(1px 2px) - 1px: верхняя и нижняя, 2px: левая и правая
(1px 2px 3px) - 1px: верхняя, 2px: левая и правая, 3 нижняя
(1px 2px 3px 4px) - 1px: верхняя, 2px: правая, 3px: нижняя, 4px: левая

Также возможно перечислять свойства в одну строчку, разделяя пробелом. В этом случае тоже не важен порядок следования свойств.

```
border: 1px solid black;
```

Есть возможность каждую границу задавать отдельно, когда необходимо, к примеру только одна граница.

```
border-top: 2px dotted green;
border-bottom: 3px double blue;
border-left: 1px solid red;
border-right: 4px inset #000;
```

## Цвет текста - color

```
color: red;
color: #78fa2e;
color: RGB(34, 21, 56);
```

Цвет текста также можно задавать любым из 3 способов.

## Ширина и высота: width и height

```
width: 200px;
height: 300px;
```

Можно задавать ширину и высоту в любых единицах измерения CSS. Подробнее на этих свойствах остановимся, когда будем верстать макет при помощи слоев.

## Шрифт - font

```
font-family: "Times New Roman", serif, Verdana;
```

- serif — шрифты с засечками
- sans-serif — рубленные шрифты, без засечек
- cursive — курсивные шрифты
- fantasy — декоративные шрифты
- monospace — моноширинные шрифты

*font-family* - устанавливает шрифт текста. Существуют 5 основных семейств шрифтов. У каждого семейства существуют несколько видов шрифтов. Какие шрифты относятся к какому семейству, можно узнать из справочников. Можно через запятую указывать несколько шрифтов. Первым будет использоваться шрифт "Times New Roman", если по каким либо причинам данный шрифт не установлен на компьютер, то тогда будет отображаться следующий шрифт. Если название  шрифта состоит из нескольких слов, то его заключают в кавычки.

```
font-style: italic; (oblique | normal)
font-variant: small-caps;
font-weight: bold; (bolder | lighter| 100 | 200)
font-size: 20px; (small | medium | large)
```

*font-style* – стиль шрифта. По умолчанию установлен шрифт в значении normal, italic - это курсивное начертание, который имитирует рукописный текст, а oblique - наклонное начертание, который получается путем наклона знаков вправо.
*font-variant* имеет только 2 значения, по умолчанию установлено значение normal, и small-caps, которое у строчных букв имитирует заглавные буквы, только уменьшенного размера.
*font-weight* задает насыщенность шрифта, можно указывать значения предопределенными словами, например bold -полужирный, bolder - жирный, lighter - светлый. Еще есть возможность указывать насыщенность цифрами от 100 до 900.
*font-size* определяет размер шрифта, можно указывать в любых единицах измерения, или предопределенными словами.
Указывать стиль шрифта можно при помощи сокращенной записи. В данном случае важен порядок следования значений.

```
font: font-style
      font-variant
      font-weight
      font-size
      font-family;

font: bold 24px Arial, Verdana;
```

## Оформление списков - list-style

```
list-style-type: circle; (disc | square | armenian | decimal)
list-style-position: inside;
list-style-image: url(img/list.png);
```

Свойство list-style определяет стиль маркера у списков.
*list-style-type* - тип маркера, который может быть разных видов, в примере приведены только некоторые из них. Остальные виды маркеров можно найти в справочнике.
*list-style-position* определяет то, где располагается маркер, по умолчанию у него значение outside. В этом случае маркеры будут располагаться за пределами текстового блока. При значении inside, наоборот, внутри текстовых блоков.
*list-style-image* позволяет вместо маркера установить изображение, для этого нужно просто указать к нему путь в скобках url.
Для определения стиля маркеров также существует сокращенная запись.

```
list-style: list-style-type
            list-style-position
            list-style-image;


list-style: circle inside;
```

```
text-align: center; (justify | left | right)
text-decoration: none; (blink | line-through | overline | underline | none)
text-transform: capitalize;  (lowercase | uppercase)
```

И еще некоторые полезные свойства.
*text-align* - выравнивание содержимого блока по горизонтали. Принимает 4 значения: left, right, center и justify (выравнивание происходит по ширине, т.е. одновременно по левому и по правому краю).
*text-decoration* применяется для следующего оформления текста: blink - добавляет мигание текста, line-through - перечеркивает текст, overline - задает линию над текстом, underline - задает линию под текстом (подчеркивает текст), none (по умолчанию) - отменяет все эффекты.
*text-transform* используется для изменения регистра символов. При значении capitalize, каждое слово в предложении будет начинаться с заглавной буквы, при значении lowercase все символы будут прописными, а при значении uppercase,

все символы будут строчными.

## *Вложенность.*

При изучении тегов HTML мы рассматривали, что можно вкладывать одни HTML теги в другие. А при помощи CSS есть возможность управлять различными вложенными конструкциями. Для управления вложенностью в CSS существуют несколько специальных селекторов. Рассмотрим эти селекторы на примерах.

### Контекстные селекторы

```
p.main strong a {
    font-size: 28px;
    color: red;
}


<p class="main">В этом параграфе
    <strong><a href="#">эта ссылка</a></strong> будет размером 28 px и
красного цвета,
    <a href="#">а эта будет обычной</a>.
</p>
<p>А в этом параграфе
    <a href="#">ссылка</a> тоже будет обычной.
</p>
```

В этом примере можно увидеть два параграфа, первый из которых, с классом main. В обоих параграфах вложены ссылки, причем в первом параграфе две ссылки, и первая ссылка вложена в тег strong. Этой ссылке задаем определенный стиль. Обратимся к этой ссылке при помощи контекстного селектора. Для этого в в качестве селектора сначала указывается тег параграфа с классом main, затем ставится пробел, и следующим указывается тег strong, после этого обращаемся к тегу нужной нам ссылки. Таким образом заданный стиль будет применяться ТОЛЬКО к первой ссылке в параграфе с классом main.

```
p.main a {
      font-size: 28px;
      color: red;
 }
 <p class="main">В этом параграфе
      <strong><a href="#">эта ссылка</a></strong> будет размером 28 px и
красного цвета,
      <a href="#">и эта тоже</a>.
 </p>
 <p>А в этом параграфе
      <a href="#">ссылка</a> тоже будет обычной.
 </p>
```

Теперь из контекстного селектора убираем тег strong, и в этом случае обе ссылки из параграфа с классом main приобретут заданный стиль, т.е. станут красного цвета с размером шрифта в 28 px. Запись данного контекстного селектора означает, что нужно применить стиль ко всем тегам ссылки, которые находятся внутри параграфов с классом main.

## Дочерние селекторы

```
p. main > a {
      font-size: 20px;
      color: green;
      background: yellow;
}
<p class="main">
      <a href="#">Ссылка 1</a>
</p>
<p class="main">
      <em><a href="#">Ссылка 2</a></em>
</p>
```

При помощи дочерних селекторов можно выбрать только те теги, которые являются прямыми потомками определенного элемента. В этом примере, в первом параграфе, ссылка является дочерним элементом для этого параграфа, т.к. вложена только в параграф. А во втором параграфе, ссылка для параграфа не будет являться дочерним элементом, т.к. она вложена в тег em, и соответственно будет являться дочерним элементом для тега em. А тег em для этой ссылки будет родительским элементом. В разделе CSS кода после обращения к селектору параграфа p с классом main ставится знак больше, а затем тег a, это означает, что необходимо обратиться к дочернему тегу ссылки параграфа с классом main. В этом примере такому запросу соответствует ссылка в первом параграфе, и ТОЛЬКО данной ссылке будет применен заданный стиль.

## Соседние селекторы

```
p. main em + a {
      font-size: 20px;
      color: green;
       background: yellow;
}
<p class="main">
      <a href="#">Ссылка 1</a>
</p>
<p class="main">
      <em>Соседним элементом будет</em> 
      <a href="#">Ссылка 2</a>
</p>
```

Для демонстрации работы соседних селекторов изменим предыдущий пример, ссылка 2 теперь будет не вложена в тег em, а находиться рядом, т.е. будет являться для тега em соседним элементом. Для того, чтобы обратиться к этой ссылке и задать ей определенный стиль, в разделе CSS кода после тега em надо поставить знак плюс, а затем указать тег ссылки. В этом случает ТОЛЬКО ссылка 2 приобретет заданный стиль.

## Наследование

Наследование - это перенос стилей от элемента к вложенным в него тегам.

```
p {
      font-size: 18px;
      color: red;
}


<p>В этом случае весь текст в этом параграфе <b>будет</b> красного цвета и шрифтом <b><i>размером 18 px</i><b>
</p>
```

В данном примере в тег <p> вложены теги два тега <b>, в один из которых также вложен тег <i>. В этом случае весь текст в этом параграфе будет заданного стиля. То есть, все теги, вложенные в параграф, унаследовали заданный стиль. Но, не все свойства CSS наследуются.

```
p {
    font-size: 18px;
    color: red;
}


 <p>В этом случае весь текст в этом параграфе <b>будет</b> красного цвета и шрифтом <b><i>размером 18 px</i><b>, только
    <a href="#">эта ссылка</a> не будет красной
 </p>
```

Если добавить в этот пример еще и ссылку, то эта ссылка унаследует только свойство font-size, но свойство color не унаследует. Узнать, наследуется ли определенное свойство CSS или нет, можно узнать только из справочников. В данном примере, чтобы применить к ссылке свойство color, так же как у всех остальных элементов, можно для этой ссылки задать значение inherit для свойства color. inherit означает, что элементу необходимо наследовать данное свойство от родителя.

## Группирование свойств

Группирование свойств необходимо использовать тогда, когда для разных элементов заданны одинаковые стили. И в этом случае надо стараться избегать повторения кода.

```
h1 {
    color: blue;
    text-align: center;
    font-family: Verdana;
}
h3 {
    color: blue;
    text-align: center;
    font-family: Arial;
}
p {
    color: blue;
    text-align: center;
    font-size: 12px;
}
```

Для того, чтобы свойства сгруппировать, надо через запятую перечислить те селекторы, для которых будем нужно сгруппировать свойства, затем перечислить повторяющиеся стили. А дальше уже для каждого элемента задать уникальные стили.

```
h1, h3, p {
     color: blue;
     text-align: center;
}
h1 {
      font-family: Verdana;
}
h3 {
     font-family: Arial;
}
p {
     font-size: 12px;
}
```

## *Приоритеты стилей в css*

Вы можете столкнуться с ситуацией, когда при разработке сайтов, вы задаете определенное свойство какому-нибудь элементу, а это свойство не работает, т.е. элемент не приобретает заданный стиль. Это происходит потому, что где-то уже был установлен определенный стиль этому элементу. Чтобы решить эту проблему и задать нужный стиль, нужно знать приоритеты применения стилей. Существует такое понятие, как каскадирование, которое применяется тогда, когда одному и тому же элементу пытаются присвоить одни и те же стили. Например, мы всем параграфам пытаемся присвоить сначала красный цвет, а потом зеленый. И какое правило должно тогда примениться?

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

В данном случае все параграфы будут зелеными, потому что по правилам, если одинаковому селектору присваивать одинаковые свойства, то применится тот стиль, который стоит ниже.

## Приоритеты источников стилей

1. Стиль автора документа обладает самым высоким приоритетом. Этот стиль задает сам разработчик сайта.
2. Стиль, заданный пользователем в настройках браузера. Стиль CSS может задать конечный пользователь этого сайта, если подключит свой собственный файл стилей. Этот источник является менее приоритетным.
3. Стиль самого браузера, т.е. тот стиль, который определен в настройках самого браузера. Это источник обладает самым низким приоритетом.

## Приоритеты стилей автора

Рассмотрим приоритеты стилей автора проекта. Самым важным свойством является то, у которого после значения свойства установлена директива !important.

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

Добавим свойству первого параграфа из предыдущего примера директиву !important, то теперь в этом случае у всех параграфов цвет текста будет красным, несмотря на то, что это объявление свойства стоит первым. Если эту директиву применит пользователь с своем собственном файле стилей, то тогда этот стиль становиться важнее стиля автора. Это нужно для того, чтобы люди с ограниченными возможностями смогли устанавливать свои стили, которые будут важнее всех.

Вторым по приоритетности является стиль, объявленный в атрибуте style любого тега.

```
h1 {
    color: green;
}

<h1 style="color: red;">Цвет будет красным</h1>
```

В данном примере, цвет текста заголовка первого уровня будет красным, так так этот стиль переопределен в атрибуте style.

Следующий уровень, это уровень приоритета селекторов. Существует такое понятие как специфичность. Смысл в том, что браузер будет начислять определенное кол-во баллов за разные типы селекторов, а так же их количество. И больший приоритет получают те стили, которые набирают большее кол-во баллов.

- *Селекторы тегов* и *псевдоэлементы* — по 1 баллу (0, 0, 0, 1)
- *Селекторы атрибутов*, *классы* и *псевдоклассы* — по 10 баллов (0, 0, 1, 0)
- *Идентификаторы* — по 100 баллов (0, 1, 0, 0)
- Атрибут style – 1000 баллов (1, 0, 0, 0)

Пример начисления баллов за специфичность

- p { }  - 1 балл (селектор тегов )
- p:first-letter { } - 2 балла (1 - селектор тегов и 1 - псевдоэлемент)
- input[type="submit"] { } - 11 баллов (по 1 селектору тегов и атрибутов)
- div.head .new { } - 21 балл (2 класса и 1 селектор тегов)
- #header a:hover { } - 111 баллов (идентификатор, селектор тегов и псевдокласс)

Следующие по приоритетности стили указанны в порядке убывания

4. Стили, заданные в разделе <head>
5. Стили, подключаемы из внешних файлах
6. Наследуемые стили от предков

# *Псевдоклассы и псевдоэлементы*

Псевдоклассы и псевдоэлементы позволяют назначить CSS стили структурам, существование которых на веб-странице не обязательно, т.е. стили применяются к частям документа не на основании той информации, которая содержится в его структуре, и даже изучив эту структуру, невозможно понять, что к этим элементам применяются какие-либо стили.

## Псевдоклассы

Вы наверняка замечали на многих сайтах, когда вы наводите мышкой на какой либо пункт меню, и этот пункт меняет свой вид, у него может изменяться цвет фона, цвет ссылки, может поменяться даже шрифт, или его размер. Так вот, это происходит как раз благодаря псевдоклассам. Рассмотрим синтаксис псевдоклассов.

```
Селектор:псевдокласс {
    свойство:значение;
}

a:hover{
    color: #ccc;
}
```

После селектора ставится двоеточие, и сразу после него без пробела, указывается название псевдокласса.

## Псевдоклассы, определяющие состояние элементов

```
:hover {} – курсор мыши в пределах элемента
:active {} – при активации элемента
:focus {} – при получении фокуса элемента
:link {} – непосещенные ссылки
:visited {} – посещенные ссылки
```

hover - это псевдокласс, который при наведении на элемент курсора мышки меняет его состояние.
active активизируется в тот момент, когда пользователь активирует какой-нибудь элемент, например, нажимает на гиперссылку, которая в момент нажатия, будет иметь тот стиль, который будет задан при помощи псевдокласса active.
focus активизируется в момент фокусирования в данный элемент, к примеру, когда в поле ввода попадает указатель.
Псевдоклассы link и visited применяются к непосещенным и посещенным ссылкам соответственно. По умолчанию непосещенные ссылки заданны синего цвета, а посещенные фиолетовые, а при помощи псевдоклассов link и hover можно менять их стили.

Псевдокласс first-child позволяет выбрать первый дочерний элемент какого-либо тега.

```
li:first-child{
    font-size: 24px;
    color: #F23401;
}

<ul>
    <li>Первый элемент</li>
    <li>Второй элемент</li>
    <li>Третий элемент</li>
</ul>
```

Для того, чтобы понять, как работает данный псевдокласс, рассмотрим простой пример. Задаем элементу списку  <li> псевдокласс first-child, и у него прописываем определенный стиль. Как видно, маркированный список, состоит из трех элементов. Заданный стиль будет применен ТОЛЬКО к первому элементу данного списка. Это происходит потому, что первый элемент списка <li> будет именно первым дочерним у тега <ul>.

Псевдокласс lang, позволяет управлять стилями, когда у в тексте могут встречаться разные языки.

```
strong:lang(en){
    color: #f00;
}

<p><strong>Меня зовут</strong> -
    <strong lang="en">My name is</strong>
</p>
```

Этот псевдокласс принимает в скобках после названия, т.е. после lang, название языка. В том теге, где необходимо использовать данный псевдокласс указывается атрибут, который называется тоже lang, и в его значении - название того языка, который будет принимать псевдокласс lang.

## Комбинирование псевдоклассов

Псевдоклассы можно комбинировать в одном селекторе, просто перечисляя их через двоеточие, как показано на следующем примере.

```
a:link:hover{
    color: #FD56SE;
}
a:visited:hover{
    color: #FD5;
}
```

## Псевдоэлементы

Псевдоэлементы - практически то же самое, что и псевдоклассы, только они позволяют ввести несуществующие элементы в веб-документ, и придать им определенные стили. Синтаксис у них абсолютно такой же, как и у псевдоклассов, т.е. после селектора ставится двоеточие, а затем название псевдоэлемента.
В CSS существует 4 псевдоэлемента:

- :first-letter - первая буква в тексте элемента
- :first-line - первая строка
- :after - применение стилей после элемента
- :before - применение стилей до элемента

## *Домашнее задание*

В этом домашнем задании вы будете дорабатывать задание второго урока, придавая ему стили CSS. Задание со звездочками делать не обязательно, но желательно.

1. Создайте файл style.css, в котором будут храниться все стили вашей работы. Не забывайте о том, что стили, так же как и изображения хранятся в отдельной папке. Подключите этот файл ко всем страницам.
2. В качестве фона, на всех страницах, установите любое изображение в верхнем правом углу.
3. Для всех ссылок меню задайте определенный стиль. Измените у них шрифт, цвет, фон, ну и т.д.
4. На станице «Статьи» заголовки должны быть определенного шрифта, определенного цвета, отличного от стандартного цвета ссылок, и эти ссылки не должны быть подчеркнутыми (вспомните, что в предыдущем задании было условие, что заголовок является одновременно ссылкой на конкретную статью). У заголовка должна быть только нижняя граница (border). Краткому описанию статьи установите какой-нибудь фоновый цвет. При наведении на ссылку «читать далее», она перестает быть подчеркнутой и меняет цвет. Ссылку «перейти к комментариям» покрасьте в другой цвет, например, в серый.
5. На странице просмотра конкретной статьи заголовок выровняйте по центу и в каждом параграфе сделайте первую букву первого слова другого шрифта, размера и цвета. Например так: *П*ервая буква любого параграфа…
6. На странице «Каталог» в списке установите в качестве маркеров произвольные изображения, для категорий одни, для подкатегорий – другие. Различные иконки можно найти в интернете, например, здесь: http://findicons.com/. У категорий должен быть размер шрифта больше, чем у

подкатегорий. Установите разные фоновые цвета у списка с категориями и списка с подкатегориями. И задайте списку определенную ширину.
* После некоторых элементов в списке категорий, сделайте лейбл, что это категория новая, например, так:

- Цветы
  - Живые цветы new
  - Искусственные цветы
- Игрушки
  - Мягкие игрушки
  - Конструкторы new
- Книги
  - Аудиокниги new
  - Книги в мягком переплете

7. На странице «галерея изображений» установите в качестве фона текстуру (небольшое изображение, размноженное по экрану) и каждому изображению задайте толстую рамку в несколько пикселей.
   * При наведении мышкой на любую картинку в галерее, рамка должна исчезнуть, но сама картинка не должна при этом смещаться, т.е. должна оставаться неподвижной.
8. На странице «регистрация», установите всем стандартным текстовым полям ввода одну ширину при помощи CSS, полю для ввода пароля другую. При вводе адреса электронной почты в соответствующее поле, фон и цвет шрифта должны изменяться. В момент нажатия на кнопку «зарегистрироваться» она должна окрашиваться в другой цвет, то же самое должно происходить и с кнопкой «очистить», только цвет должен отличаться.
   * В форме, при наведении курсора на поля checkbox и radio, должна вокруг этих полей появляться прерывистая рамка.
   * При вводе текста в поля с ФИО, каждое слово должно начинаться с большой буквы, даже если пользователь будет вводить ФИО с маленькой буквы. То же самое касается и полей на странице с контактами.
9. Приветствуются самостоятельные дополнения ваших работ в пределах пройденной темы.