
Урок 3

# Основы CSS

Что такое CSS. Синтаксис CSS. Способы объявления CSS. Селекторы (id, class, tag). Селекторы атрибутов. Основные свойства стилей. Вложенность. Наследование и группирование свойств. Проверка подключения файла стилей.

## Оглавление

# 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.

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

## Комментарии в CSS

```
/* Внешний вид*/
p {
color: blue;
}
/*
Стили
для
параграфа
*/
```

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

# Способы объявления css

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

## Inline-стили

```
<body style="background:#0f0;">
      <h1 style="color: blue;text-align: center;">
            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 в том же формате стили, свойств и их значений.

## Стили в разделе head

```
<head>
	<style type="text/css">
	body {
		color: blue;
		background: #0f0;
	}
	h1 {
		text-align: center;
	}
	</style>
</head>
```

Для того, чтобы подключить стили этим способом, существует HTML тег style. У него в атрибуте type указываем 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.

**style.css**

```
body {
	color: blue;
	background: #0f0;
}
h1 {
	text-align: center;
}
```

А в нужном HTML файле этот файл подключаем.

**index.html**

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

| html | css |
| --- | --- |
| <h1>Для всех заголовков первого уровня цвет текста будет синим</h1> | h1 {<br>color: blue;<br>} |

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

## Селекторы идентификаторов (id)

| html | css |
| --- | --- |
| Цвет фона данного параграфа будет серым | #first {<br>background: #ccc;<br>} |

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 то же имя двум и более элементам.

## Селекторы классов (class)

| html | css |
| --- | --- |
| <h1 class=”border”>Заголовок с рамкой</h1><br><br> Параграф с рамкой | .border {<br>border: 1px solid black;<br>} |

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.

| html | css |
| --- | --- |
| <img src="pic.jpg" alt="Фото"><br><br><input type="text"> | img[alt] {<br>width: 100px;<br>}<br>input[type="text"] {<br>font-size: 10px;<br>} |

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. 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 изменяться 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.

### Пиксели

Пиксель px – это самая базовая, абсолютная и окончательная единица измерения. Количество пикселей задаётся в настройках разрешения экрана, один px – это как раз один такой пиксель на экране. Все значения браузер в итоге пересчитает в пиксели.

Пиксели могут быть дробными, например, размер можно задать в 16.5px. Это совершенно нормально, браузер сам использует дробные пиксели для внутренних вычислений. К примеру, есть элемент шириной в 100px, его нужно разделить на три части – волей-неволей появляются 33.333...px. При окончательном отображении дробные пиксели, конечно же, округляются и становятся целыми.

Для мобильных устройств, у которых много пикселей на экране, но сам экран маленький, браузер автоматически применяет масштабирование, чтобы обеспечить читаемость.

\+ Четкость и понятность

\- Другие единицы могут устанавливать соотношения между различными размерами

### Проценты

Проценты %, как и em – относительные единицы. Когда мы говорим «процент», то возникает вопрос – «Процент от чего?» Как правило, процент берётся от значения свойства родителя с тем же названием, но не всегда. Это очень важная особенность процентов, про которую, увы, часто забывают.

**Относительно шрифта: em**

1em – текущий размер шрифта.

Можно брать любые пропорции от текущего шрифта: 2em, 0.5em и т.п.

Размеры в em – относительные, они определяются по текущему контексту.

# Абсолютные единицы измерения

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

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

## Способы задания цветов

Цвета в CSS можно задавать 3 различными способами. Первый способ - 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 шестнадцатеричный 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 перечисляется насыщенность каждого из 3-х цветов.

### Шестнадцатеричный RGB

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

#FFAA00 => #FA0

Если использовать шестнадцатеричный RGB, то 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. 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.

# Свойства стилей CSS.

## Ширина и высота: width и height

```
p {
	height: 200px;
	width: 300px;
}
```

Можно задавать ширину и высоту в любых единицах измерения CSS. Если содержимое блока превышает указанную высоту, то высота элемента останется неизменной, а содержимое будет отображаться поверх него. Из-за этой особенности может получиться наложение содержимого

элементов друг на друга, когда элементы в коде HTML идут последовательно. Чтобы этого не произошло, добавьте overflow: auto к стилю элемента.

Для изменения размеров изображения средствами HTML предусмотрены атрибуты height и width. Допускается использовать значения в пикселах или процентах. Если установлена процентная запись, то размеры изображения вычисляются относительно родительского элемента — контейнера, где находится тег <img>. В случае отсутствия родительского контейнера, в его качестве выступает окно браузера. Иными словами, width="100%" означает, что рисунок будет растянут на всю ширину веб-страницы. Добавление только одного атрибута width или height сохраняет пропорции и соотношение сторон изображения. Браузер при этом ожидает полной загрузки рисунка, чтобы определить его первоначальную высоту и ширину.

## Фон элемента - background

```
background-color: #ff0;
background-image: url(img/photo.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 повторяться 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, в которой можно записать все перечисленные значения в одну строку, разделяя значения пробелом. Если пропускать какие-либо значения, то будут подставляться значения по умолчанию.

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

## border – рамка вокруг элемента

```
border-color: red; (#f00 | RGB(255, 0, 0))
border-style: solid; (dotted | dashed | groove | ridge | 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 необходима, 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.

## 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 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, которое имитирует рукописный текст, а oblique - наклонное начертание, которое 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. Ещё 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 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; (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: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 строчными, а при значении uppercase все символы будут заглавными.

# 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

| html | css |
|---|---|
| В этом параграфе<br><a href="#">эта ссылка</a> будет размером 18px и красного цвета,<br><a href="#">а эта будет обычной</a>.<br> | .main strong a {<br>font-size: 18px;<br>color: red;<br>} |

В этом примере можно увидеть параграф с классом main. В параграф вложена ссылка. Этой ссылке задаем определенный стиль. Обратимся к этой ссылке при помощи контекстного селектора. Для этого 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.

| html | css |
| --- | --- |
| В этом параграфе<br><a href="#">эта ссылка</a> будет размером 18px и красного цвета,<br><a href="#">а эта будет обычной</a>.<br> | .main a {<br>font-size: 18px;<br>color: red;<br>} |

Теперь из контекстного селектора убираем тег strong. В этом случае обе ссылки из параграфа с классом main приобретут заданный стиль, т.е. станут красного цвета с размером шрифта в 18 px. Запись данного контекстного селектора означает, что нужно применить стиль ко всем тегам ссылки, которые находятся внутри параграфов с классом main.

## Дочерние селекторы

| html | css |
| --- | --- |
| <br><a href="#">ссылка 1</a><br> <br> <br><a href="#">ссылка 2</a><br> | .main > a {<br>font-size: 18px;<br>color: red;<br>} |

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 i, и, соответственно, будет являться дочерним элементом для тега i. А тег i 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

| html | css |
| --- | --- |
| <br>  <a href="#">ссылка 1</a><br> <br> <br>  Соседний элемент<br>  <a href="#">ссылка 2</a><br> | .main i + a {<br>  font-size: 18px;<br>  color: red;<br>} |

Для демонстрации работы соседних селекторов изменим предыдущий пример. Ссылка 2 теперь будет не вложена в тег i, а находиться рядом, т.е. будет являться для тега i соседним элементом. Для того, чтобы обратиться к этой ссылке и задать ей определенный стиль, в разделе CSS кода после тега i 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.

| html | css |
| --- | --- |
| В этом параграфе весь текст **будет**красного цвета и шрифтом ***размером 18 px***<br> | p {<br>  font-size: 18px;<br>  color: red;<br>} |

В данном примере в тег  вложены теги два тега , в один из которых также вложен тег . 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.

| html | css |
| --- | --- |
| В этом параграфе весь текст **будет**красного цвета и шрифтом ***размером 18 px***, только <a href="#">эта ссылка</a> не будет красной<br> | p {<br>  font-size: 18px;<br>  color: red;<br>} |

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 также 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.

## Группировка свойств

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

| html (без группировки свойств) | html (правильный вариант) |
|---|---|
| h1 {<br>text-align: center;<br>color: blue;<br>font-family: Verdana;<br>}<br>h3 {<br>text-align: center;<br>color: blue;<br>font-family: Arial;<br>}<br>p {<br>text-align: center;<br>color: blue;<br>font-size: 12px;<br>} | h1, h3, p {<br>text-align: center;<br>color: blue;<br>}<br>h1 {<br>font-family: Verdana;<br>}<br>h3 {<br>font-family: Arial;<br>}<br>p {<br>font-size: 12px;<br>} |

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

# Приоритеты стилей в css

Вы можете столкнуться с ситуацией, когда при разработке сайтов, вы задаёте 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 черный цвет, а потом зеленый. И какое правило должно тогда примениться?

```
h1 {
	color: black;
}
h1 {
	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.

```
h1 {
	color: black!important;
}

h1 {
	color: green;
}
```

Добавим свойству первого параграфа из предыдущего примера директиву !important, теперь в этом случае у всех параграфов цвет текста будет черным, несмотря на то, что это объявление свойства стоит первым. Если эту директиву применит пользователь в своём 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.

| html | css |
| --- | --- |
| <h1 style="color: red;"><br>Заголовок первого уровня<br></h1> | h1 {<br>color: green;<br>} |

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

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

- Стили, заданные в разделе <head>;
- Стили, подключаемы из внешних файлах;
- Наследуемые стили от предков.

# Практика

## Создание стилей для меню сайта

```
<!-- Меню-->
<ul class=”menu”>
    <li>Главная</li>
    <li><a href="catalog.html">Каталог</a></li>
    <li><a href="contacts.html">Контакты</a></li>
</ul>
```

```
body {
    background-color: #f8f8f8;
}
.menu li {
    font-style: italic;
    font-size: 20px;
    list-style-type: none;
}
.menu li a {
    color: #8B00FF;
}
```

# Домашнее задание

В этом домашнем задании вы будете дорабатывать задание второго урока, придавая ему стили CSS.

1. Создать файл style.css, в котором будут храниться все стили вашей работы. Подключить этот файл ко всем страницам.
2. В качестве фона на всех страницах установить цвет #f8f8f8.
3. Меню сайта:
    a. Для всех ссылок меню задать определённый стиль. (Цвет текста, размер шрифта, начертание шрифта и т.д.)
    b. Убрать маркеры списка.
4. Страница “Подробное описание товара”
    a. Заголовки (Краткое описание товара, Характеристики, Подробное описание товара)
        i. Цвет текста черный.
        ii. Размер шрифта 18px;
        iii. Насыщенность шрифта 400. (font-weight);
        iv. установите цвет фона #eaeaea;
    b. Для текста краткого описания товар.
        i. Цвет текста #707070;
        ii. Размер шрифта 14px.
        iii. Начертание шрифта italic. (font-style)
        iv. Высоту текста 16px (line-height);
    c. Для текста подробного описания товара:
        i. Цвет текста #484343.
        ii. Размер шрифта 16px.
        iii. Насыщенность шрифта 400. (font-weight);
        iv. Высоту текста 24px (line-height);
        v. Расположение текста по левому краю (text-align)
    d. Для списка внурти подраздела Характеристики товара
        i. задайте списку стили, отличные от всего остального текста.
        ii. *установите в качестве маркеров произвольные изображения
5. Странице “Контакты”:
    a. Задать значения ширины и высоты для полей ввода.
    b. Задать стили для текста, внутри полей input. ( Цвет текста, размер шрифта и т.д.).
    c. Следите за тем, чтобы на странице всё выглядело гармонично, не выбирайте слишком резких цветов.
6. *Для изображений, размещенных на странице подробного описания товара, задать рамку произвольным цветом.

7. *Приветствуются самостоятельные дополнения ваших работ в пределах пройденной темы.

# Дополнительные материалы

1. Статья про добавление стилей на страницу
2. Интересная статься про селекторы
3. 30 CSS-селекторов, о которых полезно помнить
4. Вложенность селекторов
5. Группировка и вложенность селекторов

# Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:


1. http://htmlbook.ru/samcss/sposoby-dobavleniya-stiley-na-stranitsu
2. http://htmlbook.ru/samcss/dochernie-selektory
3. https://learn.javascript.ru/css-selectors
4. https://learn.javascript.ru/css-units
5. http://technologyweb.org/%D0%B3%D1%80%D1%83%D0%BF%D0%BF%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0-%D0%B8-%D0%B2%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D1%8C/