Основы CSS

Что такое CSS

Давайте вспомним что же 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.

Давайте сразу представим что мы создали простую страницу с помощью html, какую проблему мы сможем заметить? Все заголовки или параграфы что мы добавили, расположены друг под другом и цвет текста черный на белом фоне, никто не любит такие сайты, именно для решения этой проблемы и был создан CSS, так что вы сможете задать и размер шрифта и цвет шрифта и даже расположение любых html элементов на странице.

Синтаксис CSS

```
селектор {
    свойство: значение;
    свойство2: значение2;
}
```

Рассмотрим пример кода.

Первым всегда указывается селектор. Он сообщает браузеру, к какому элементу или элементам веб-страницы будет применяться стиль. Это могут быть теги, идентификаторы, классы, атрибуты тегов, которые мы рассмотрим чуть позже.

В фигурных скобках указываются свойства селектора в виде пары: название свойства и через двоеточие — его значение. После каждой пары ставится точка с запятой (;), которая свидетельствует о разделении свойств.

Если этого не сделать, ошибки не будет. Но возьмите в привычку всегда ставить этот знак. Так вы не будете про него забывать.

Способы оформления CSS

1-й способ

```
Селектор{свойство: значение;свойство2: значение2;}
```

В нём нужно записывать все свойства в одну строку. Этот способ не очень удобен, так как 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.

2-й способ

```
Селектор
{
    свойство: значение;
    свойство2: значение2;
}
```

Второй способ используется не так часто, как третий, так как наглядность не становится больше.

3-й способ

```
Селектор {
    свойство: значение;
    свойство2: значение2;
}
```

Самый популярный и удобный способ написания стилей — третий. В нём можно увидеть наглядность и одновременно с этим, компактность. Поэтому его выбирает большая часть программистов.

Вам нужно помнить что способов можете быть сколько угодно, вы можете увидеть какой-то уникальный синтаксис, это не будет ошибкой, чаще всего это выбор компании.

Важно: при работе с CSS мы будем использовать автоформат кода, поэтому мы будем использовать синтаксис из 3-его способа

Комментарии в CSS

```
/* Внешний вид комментария */
p {
    color: blue;
}
```

```
/*
Стили
для
параграфа
*/
```

В CSS можно оставлять комментарии для описания свойств элементов или для комментирования самих стилей при редактировании документа.

Также вы можете оставить любые подсказки для себя, если вы хотите указать что это стили для определенного раздела или это стили которые необходимо будет поменять.

Важно! Если вы вы хотите быстро закомментировать несколько строк css, нужно выделить их и нажать ctrl + /. Если вы используете операционную систему MacOs то горячие клавиши будут command + / . (Данный набор горячих клавиш будет работать в редакторе кода Visual Studio Code)

Способы объявления CSS

Чтобы использовать стили CSS в веб-документе, нужно их сначала подключить. Для этого есть три способа.

1. Inline-стили.
2. Стили в разделе head.
3. Внешний CSS-файл.

Inline-стили

```
<body style="background: green;">
      <h1 style="color: blue; text-align: center;">
            Заголовок
      </h1>
</body>
```

Для подключения CSS этим способом в HTML есть атрибут style. Его можно указывать практически у любого HTML-тега. В значении атрибута style перечисляются стили свойств и их значений в том же формате.

Плюсы

Можно быстро прописать какой-нибудь простой стиль для элемента. Например, какое-то слово в тексте выделить красным цветом.

Минусы

- Нужно прописывать стиль для каждого файла
- Сложность редактирования
- Нагромождения в html

Допустим, у нас несколько параграфов, и все они должны быть определённого стиля. Тогда каждому параграфу нужно прописывать этот стиль. Ещё один существенный недостаток — при таком подходе стили сложно редактировать. Что делать, если в проекте нужно поменять размер шрифта во всех параграфах?

```
<h2 style="color: blue; text-align: center;">Заголовок 1</h2>
<h2 style="color: blue; text-align: center;">Заголовок 2</h2>
<h2 style="color: blue; text-align: center;">Заголовок 3</h2>
<h2 style="color: blue; text-align: center;">Заголовок 4</h2>
```

При просмотре данного примера мы можем заметить что всем параграфам придётся дописать один и тот же стиль, что сделает код html еще более наглядным и возможно мы забудем добавить этот стиль.

Этот способ применяется редко, поэтому рекомендуется использовать Внешний CSS-файл.

Стили в разделе head

```
<head>
      <style>
          body {
             background: green;
          }
          h1 {
             color: blue;
             text-align: center;
          }
      </style>
</head>
```

Чтобы подключить стили этим способом, применяют HTML-тег style. Внутри этого тега прописываем стили, которые будут действовать для всей страницы. Тут нужно запомнить что эти стили работают в границах одной страницы, это очень важно знать.

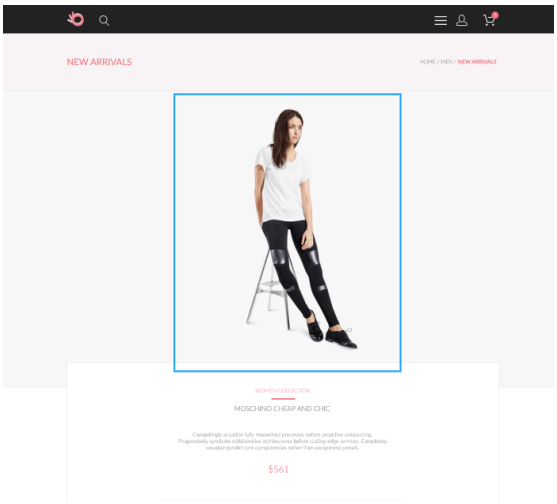
Плюсы

- Группирование стилей одной страницы
- Можно задать один стиль, для нескольких элементов

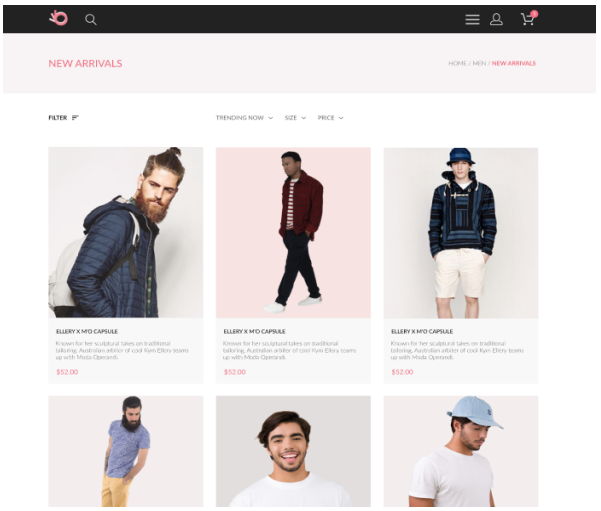
Минусы

Если вам необходимо поменять стили на второй странице, то вам автоматически нужно переходить на все остальные страницы и там также добавлять данную стилистику, это будет очень неудобно, при большем количестве страниц.

Страница товара



Страница каталога



Допустим у нас есть страница товара и страница каталога, нам необходимо поменять цвет шапки сайта, что нам придётся сделать, зайти на страницу товара, поменять значение цвета, зайти на страницу каталога и поменять значение цвета, но на главной странице такая же шапка сайта, нам необходимо открыть все страницы, чтобы поменять значение цвета шапки сайта, что является совсем не рациональным подходом.

Внешний CSS-файл

Создаём файл с расширением .css.

style.css

```
body {
    background: green;
}
h1 {
    color: blue;
    text-align: center;
}
```

В нужном HTML-файле этот файл подключаем.

index.html

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

Важно: если вы хотите быстро подключить файл стилей, вам необходимо внутри тега <head> добавить текст link:css и нажать клавишу tab, так вы получите ссылку подключение.

Для подключения CSS-файла используется тег link, который помещается в раздел head нужного HTML-файла. Чтобы правильно подключить файл стилей, у тега link нужно указать несколько атрибутов.

В атрибуте rel (обязательный атрибут) указывается значение stylesheet, то есть лист стилей. Это нужно, чтобы браузер понимал, что подключается файл стилей CSS. В атрибуте href (обязательный атрибут) указывается путь к CSS-файлу. Указывается атрибут type (необязательный атрибут) со значением text/css, В html 5 он уже необязателен. Поэтому если вы видите такой атрибут, знайте, этот синтаксис использовался в ранней версии html 4.

Плюсы

Если подключить отдельный файл, он будет действовать на все страницы, где подключим файл. И тогда, чтобы изменить цвет или размер шрифта всех параграфов, нужно изменить его один раз в одном месте.

Ещё одно преимущество в том, что браузер кеширует файл стилей. Он. сохраняет его у себя в памяти, чтобы не обращаться при каждом запросе на сервер.

Селекторы в CSS

Селекторы тегов

html	css

Для всех заголовков первого уровня цвет текста будет синим	h1 {
	  color: blue;
	}

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

Этот способ обращения к html используется крайне редко. Вы никогда не знаете, когда ещё потребуется создать заголовок или параграф.

Например, вы создали стили для всего проекта. В дальнейшем большая часть заголовков будет в другой стилистике. В этом случае придётся каждый раз менять стили, которые вы уже задали заранее.

Давайте рассмотрим пример, чтобы точно понять, что использовать селекторы тегов будет не самым верным решением

html	css

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Placeat, ab.
 Lorem ipsum dolor sit amet, consectetur adipisicing elit. Placeat, ab.
 Lorem ipsum dolor sit amet, consectetur adipisicing elit. Placeat, ab.
	p {
	background: #ccc;
	color: green;
	font-size: 20px;
	font-weight: 600;
	line-height: 20px;
	}

В примере у вас есть три параграфа и стиль для них задан в виде селектора тегов, а теперь представим что нужно создать еще 4 параграф, у которого будет только размер шрифта в 24 пикселя, какое значение color нам нужно выставить, а каким будет line-height этой части мы просто не знаем и даже если узнаем в интернете, то зачем нам ее исправлять, поэтому давайте рассмотрим какие есть виды селекторов

Селекторы идентификаторов (id)

html	css

Цвет фона параграфа будет серым	#text {
	background: #ccc;
	}

В качестве селекторов можно использовать идентификаторы. Определённому тегу в значении атрибута id указываем название, которое придумываем сами. В селекторе ставится знак #, а затем это название. Помните, что идентификатор должен быть уникальным, то есть нельзя задавать одно и то же имя двум и более элементам.

Важно! Этот способ для создания стилей не используется, так как, возможно, над проектом работает несколько разработчиков. Мы не сможем сделать так, чтобы они придумывали разные названия, обязательно будут пересечения, которых быть не должно.

Селекторы классов (class)

html	css

<h1 class="border">Заголовок с рамкой</h1>	.border {
	border: 1px solid black;
	}

Параграф с рамкой	

Классы используются аналогично id, только вместо атрибута id указывается class, а в селекторе вместо решётки — точка. Они (классы) отличаются от идентификаторов тем, что можно применять один и тот же стиль к разным элементам.

Этот способ часто используется для верстки, так как у него нет минусов, которые присутствуют при обращении к тегам и id.

Давайте рассмотрим пример https://jsfiddle.net/0qeyxn41/

Уже знакомый нам пример, но с использованием классов, получается что мы сгруппировали три параграфа и если стили поменяются, то они изменяется только для данных параграфов и если нам нужно будет добавить еще какой-то новый параграф, то он не будет содержать стилей и вы всегда сможете задать ему уникальные стили.

Свойства стилей

Единицы измерения в CSS

В CSS есть единицы измерения, с помощью которых можно определять длину, ширину элементов, а также размеры шрифтов. Не все они используются в повседневной вёрстке, но нужно иметь представления о них. Единицы измерения делятся на относительные и абсолютные.

Относительными 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 часто используются на веб-страницах.

Относительные единицы измерения

- px — пиксель;
- % — процент;
- em — высота текущего шрифта.

Пиксели (px)

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 — появляются 33.333...px. При окончательном отображении дробные пиксели округляются и становятся целыми.

Проценты (%)

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

EM (em)

При создании параграфа или заголовка мы не указываем этим текстовым блокам размер шрифта. Он задан автоматически. Мы можем заметить, что заголовок h1 больше, чем параграф p. В итоге получается лучше выставить коэффициент, который и будет определять, насколько заголовок больше или меньше параграфа.

1em — текущий размер шрифта (стандартное значение 1em = 16px).

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 ним относятся единицы измерения, которые используются в обычной жизни. Но в веб-страницах они применяются редко, поэтому использовать их нежелательно.

Как поменять значение цвета?

Цвета в CSS можно задавать различными способами. Первый способ — задавать цвета, используя их названия на английском языке, например: red, green, blue, black, yellow, white и т. д. Но при таком подходе есть ограничения в выборе цвета, невозможно получить разные оттенки цветов.

Чтобы можно было выбрать один из более, чем 16 млн цветов, нужно применить 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.

Шестнадцатеричный RGB

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

Есть сокращенный внешний вид #FFAA00 => #FA0.

Если использовать шестнадцатеричный RGB, каждый цвет можно представить в виде пары значений. То есть первый и второй символ — это красный цвет, третий и четвёртый — зелёный, пятый и шестой — синий.

Каждый символ может представляться одним из шестнадцати знаков, от 0 до буквы F латинского алфавита.

При шестнадцатеричном RGB перед кодом цвета ставится символ решётки, а дальше уже записывается сам код.

Если совпадают две буквы или цифры одного и того же цвета, можно использовать сокращенную форму записи. То есть каждый цвет будет состоять не из пары значений, а из одного, и в коде цвета окажутся 3 символа после знака решетки.

Важно уточнение, что это будет работать только если все цвета состоят из пар, если же у вас совпали только два или один из трех, то такое сокращение нельзя использовать

На начальном этапе звучит достаточно сложно, но как только мы начнём работать с макетом, сразу станет понятно что вы никогда не записываете значение цвета самостоятельно, вам просто нужно будет нажать на интересующий вас элемент, скопировать значение цвета и добавить себе в код css, так что стилизация любых элементов это проще чем кажется изначально.

Свойства стилей CSS

Ширина и высота: width и height

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

Можно задавать ширину и высоту в любых единицах измерения CSS. Если содержимое блока превышает указанную высоту, 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.

Допускается использовать значения в пикселях или процентах. Если установлена процентная запись, размеры изображения вычисляются относительно родительского элемента — контейнера, где находится тег <img>.

При отсутствии родительского контейнера в его качестве выступает окно браузера. Иными словами, width="100%" означает, что рисунок будет растянут на всю ширину веб-страницы.

Важно: 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 | center)
background-repeat: repeat-x; (repeat-y | no-repeat)
background-size: cover; (contain)
```

1. background-color задает цвет фона, который мы всегда сможем определить в макете.
2. background-image используется, чтобы в качестве фона можно было установить изображение. Для этого в значении свойства нужно указать путь к изображению в скобках url.
3. background-position указывает, где будет располагаться фоновое изображение. Может иметь значения: top, bottom, left, right, center.
4. 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. Это означает, что изображение будет повторяться по горизонтали и по вертикали.
5. background-size — размер изображения. Возможно, вы не хотите, чтобы изображение занимало стандартные параметры. Поэтому можно выставить процентное значение или универсальное cover, которое будет занимать все доступное пространство, contain — доступное пространство по высоте или ширине, чтобы фоновое изображение поместилось полностью.

Существует короткая форма записи, в которой можно записать все перечисленные значения в одну строку, разделяя значения пробелом. Если пропускать какие-либо значения, они будут подставляться 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:

1. border-color — цвет рамки.
2. border-style — стиль рамки, которая может быть разных значений: dotted, dashed, solid, double, groove, ridge, inset, outset.
3. border-width задает толщину рамки, причём её 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.

Можно перечислять свойства в одну строку, разделяя их пробелом. В этом случае не важен порядок следования свойств.

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

Каждую границу можно задавать отдельно, когда это 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;
```

Значение цвета текста, одно из самых простых и понятных свойств

Шрифт — font

font-family устанавливает шрифт текста.

```
font-family: "Gill Sans", serif;
```

1. serif — шрифты с засечками.
2. sans-serif — рубленые шрифты, без засечек.
3. cursive — курсивные шрифты.
4. fantasy — декоративные шрифты.
5. monospace — моноширинные шрифты.

Существует 5 основных семейств шрифтов. У каждого семейства есть несколько видов шрифтов. Узнать о типах шрифтов и их семействах можете узнать из справочников . Можно через запятую указывать несколько шрифтов. Первым будет использоваться шрифт Gill Sans. Если он не установлен на компьютер, то будет отображаться следующий шрифт. Название шрифта заключают в кавычки, если он состоит из нескольких слов.

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

1. font-style — стиль шрифта. Предустановлен шрифт в значении normal. italic — это курсивное начертание, которое имитирует рукописный текст. oblique — наклонное начертание. Оно получается путем наклона знаков вправо.
2. font-variant имеет только 2 значения. По умолчанию установлено значение normal и small-caps, которое у строчных букв имитирует заглавные буквы, только уменьшенного размера.
3. font-weight задает насыщенность шрифта. Можно указывать значения предопределенными словами, например, bold — полужирный, bolder — жирный, lighter — светлый. Насыщенность определяется цифрами от 100 до 900.
4. font-size определяет размер шрифта. Можно указывать в любых единицах измерения или предопределенными словами. Определять стиль шрифта можно сокращенной записью. В этом случае важен порядок следования значений.

Есть общее свойство font. На начальном этапе его не рекомендуется использовать, так как его будет сложно запомнить. Можно допустить ошибку в написании.

Работа с текстом

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

1. text-align — выравнивание содержимого блока по горизонтали. Принимает 4 значения: left, right, center и justify. Выравнивание происходит по ширине, то есть одновременно по левому и по правому краю.
2. text-decoration применяется для следующего оформления текста: line-through — перечеркивает текст, overline — задает линию над текстом, underline — под текстом (подчеркивает текст), none (по умолчанию) — отменяет все эффекты.
3. text-transform используется для изменения регистра символов. capitalize — каждое слово в предложении будет начинаться с заглавной буквы. При значении lowercase все символы будут строчными, а при uppercase — 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 есть несколько специальных селекторов. Рассмотрим их на примерах.

Контекстные селекторы

html	css

`<p class="text">`
`  В этом параграфе <a href="#">эта ссылка</a> будет размером 18px и красного цвета,`
`  <a href="#">а эта будет обычной</a>.`
`</p>`
	`.text a {`
	`  font-size: 18px;`
	`  color: red;`
	`}`

Сразу хочется отметить что данный подход является достаточно старым и мы его рассмотрим для того, чтобы вы могли ориентироваться при поддержке старых сайтов, но применять данную технологию я настоятельно не рекомендую, так как она работает намного дольше чем обращение по классам.

В данном примере представлен класс text внутри которого находится ссылка и чтобы не задавать ей отдельный класс прибегли к вложенности **.text a** но чем же данный подход плох? всё дело в том, что css будет искать все ссылки на сайте и только после этого найдет все классы текст, что естественно является не рациональным для стилизации проекта.

Группирование свойств

Группировку свойств нужно использовать, когда для разных элементов заданы одинаковые стили. Старайтесь избегать повторения кода.

html (без группировки свойств)	**html**

`.title {`
`  text-align: center;`
`  color: blue;`
`  font-family: sans-serif;`
`}`
`.heading {`
`  text-align: center;`
`  color: blue;`
`  font-family: Arial;`
`}`
`.text {`
`  text-align: center;`
`  color: blue;`
`  font-size: 12px;`
`}`
	`.title, .heading, .text {`
	`  text-align: center;`
	`  color: blue;`
	`}`
	`.title {`
	`  font-family: sans-serif;`
	`}`
	`.heading {`
	`  font-family: Arial;`
	`}`
	`.text {`
	`  font-size: 12px;`
	`}`

Чтобы свойства группировать, необходимо записать их через запятую и присвоить одинаковые стили. Дальше уже для каждого элемента задать уникальные стили.

Приоритеты стилей в CSS

Можно столкнуться с ситуацией, когда при разработке сайтов задается определенное свойство какому-нибудь элементу. Это свойство не работает, то есть элемент не приобретает заданный стиль. Это происходит потому, что где-то уже был установлен конкретный стиль 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 разные стили. Например, мы всем параграфам хотим присвоить сначала чёрный цвет, а потом зелёный. Какое правило должно тогда примениться?

```
.title {
     color: black;
}
.title {
     color: green;
}
```

В этом случае все элементы с классом title будут зелеными. Потому что по правилам, если одинаковому селектору присваивать одинаковые свойства, применяется тот стиль, который стоит ниже.

Приоритеты источников стилей

1. Стиль автора документа обладает самым высоким приоритетом. Этот стиль задаёт сам разработчик сайта.
2. Стиль, заданный пользователем в настройках браузера. Его может задать конечный пользователь этого сайта, если подключит собственный файл стилей. Этот источник будет менее приоритетным.
3. Стиль браузера определяется в его настройках. Этот источник обладает самым низким приоритетом.

Приоритеты стилей автора

Рассмотрим приоритеты стилей автора проекта. Самое важное свойство — то, у которого после значения свойства установлена директива !important.

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

h1 {
     color: green;
}
```

Добавим свойству первого заголовка из предыдущего примера директиву !important. Теперь у всех заголовков цвет текста будет чёрным, хотя это объявление свойства стоит первым.

Если эту директиву применит пользователь в собственном файле стилей, то этот стиль станет важнее стиля автора. Это нужно, чтобы люди с ограниченными возможностями могли устанавливать свои стили, которые будут важнее всех.