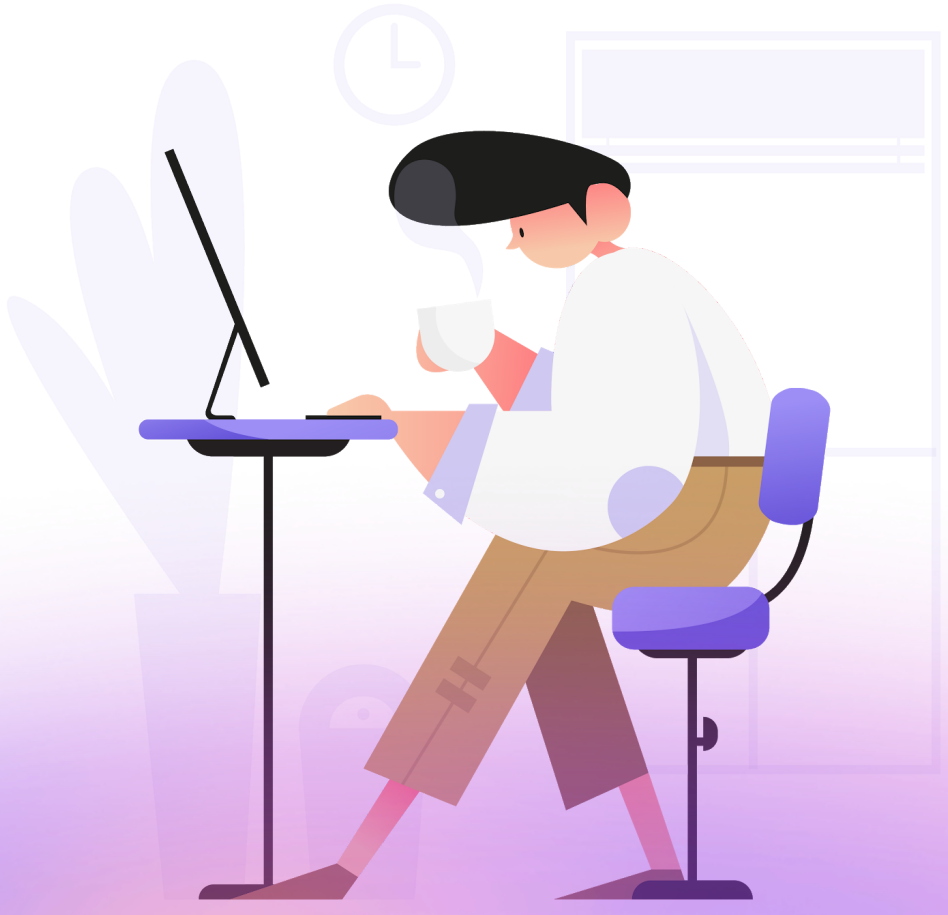
GeekBrains

HTML/CSS. Интерактивный курс

Основы CSS

[CSS3]



На этом уроке

1. Узнаем, что такое css и его синтаксис.
2. Рассмотрим, какие способы объявления css бывают.
3. Познакомимся с основными свойствами стилей.
4. Узнаем о приоритетах 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.

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.

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.

Комментарии используются в ситуациях, когда нужны стили только в определённое время года. Например, у вас есть зимняя тема сайта. Чтобы её не создавать каждый год, можно в марте закомментировать часть, отвечающую за зимнее оформление. А в декабре её раскомментировать.

Важно! *Если вы вы хотите быстро закомментировать несколько строк css, нужно выделить их и нажать ctrl + /.*

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

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

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

Стили в разделе head

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

Чтобы подключить стили этим способом, применяют HTML-тег style. Внутри этого тега прописываем стили, которые будут действовать для всей страницы.

Внешний CSS-файл

Создаём файл с расширением .css.

style.css

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

В нужном HTML-файле этот файл подключаем.b

index.html

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

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 указывается значение stylesheet, то есть лист стилей. Это нужно, чтобы браузер понимал, что подключается файл стилей CSS. В атрибуте href указывается путь к CSS-файлу. Причём как и в случае с гиперссылками, этот путь может быть относительным и абсолютным.

Указывается атрибут type 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.

Какой способ подключения стилей выбрать?

Плюс inline-стилей в том, что можно быстро прописать какой-нибудь простой стиль для элемента. Например, какое-то слово в тексте выделить красным цветом. Недостатки такого подхода в том, что для каждого тега нужно прописывать стили.

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?

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 много страниц, у нас задача поменять цвет или размер шрифта всех параграфов, нужно будет открывать каждую страницу.

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. Поэтому рассмотрим другие способы.

Селекторы идентификаторов (id)

html	css

Цвет фона параграфа будет серым	#paragraph {
	  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>
 	
 Параграф с рамкой
	.border {
	  border: 1px solid black;
	}

Классы используются аналогично id, только вместо атрибута id указывается class, а в селекторе вместо решётки — точка. Он (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.

Селекторы атрибутов

В качестве селекторов можно указывать атрибуты HTML-тегов. Есть разные способы указывать селекторы атрибутов. Для начала рассмотрим 2 примера. Остальные способы вы сможете найти в справочнике, если они вам понадобятся.

html	css

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

В этом примере сначала указывается стиль для всех картинок, у которых присутствует атрибут alt. Для этого название атрибута выдаётся в квадратных скобках, сразу после названия тега.

Второй пример. Стиль будет применяться для всех тегов <input>, в значении атрибута type которого присутствует значение text, то есть. для всех обычных текстовых полей ввода.

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

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.

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

\+ Чёткость и понятность.

\- Другие единицы могут устанавливать соотношения между различными размерами.

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

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 задавать 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 сокращённую форму записи. То есть каждый цвет будет состоять не из пары значений, а из одного, и в коде цвета окажутся 3 символа после знака решётки.

При подборе цвета первое время лучше использовать любой графический редактор. Там можно выбрать любой нужный вам цвет из палитры. Редактор покажет код выбранного цвета, который можете скопировать и вставить на страницу.

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.

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-size: cover; (contain)
```

1. background-color задаёт 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.
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-attachment определяет, будет ли изображение прокручиваться вместе с содержимым элемента. По умолчанию оно установлено как scroll. Это означает, что изображение будет прокручиваться, а при значении fixed изображение останется неподвижным.
6. background-size — размер изображения. Возможно, вы не хотите, чтобы изображение занимало стандартные параметры. Поэтому можно выставить процентное значение или универсальное cover, которое будет занимать всё 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 задаёт 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;
```

Не нужно определять значение цвета текста самостоятельно. Вам всего лишь потребуется нажать на нужный текст в макете, и значение цвета будет представлено в любом представленном значении.

Шрифт — font

font-family устанавливает шрифт текста.

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

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 Times New Roman. 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 путём 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 задаёт насыщенность шрифта. Можно указывать значения предопределёнными 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 предопределёнными словами. Определять стиль шрифта можно сокращённой 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.

```
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.

1. list-style-type — тип маркера, который может быть у разных видов. В примере приводятся только некоторые из них. Остальные виды маркеров можно найти в справочнике.
2. List-style-position определяет, 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.
3. list-style-image позволяет вместо маркера установить изображение, для этого нужно указать к нему путь в скобках url.

Для определения стиля маркеров также есть сокращённая запись.

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

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 — перечёркивает текст, overline — задаёт линию над текстом, underline — под текстом (подчёркивает 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="main">`
`  В этом параграфе`
`  <strong><a href="#">эта ссылка</a></strong>`
`будет размером 18px и красного цвета,`
`  <a href="#">а эта будет обычной</a>.`
`</p>`
	`.main strong a {`
	`  font-size: 18px;`
	`  color: red;`
	`}`

В этом примере можно увидеть параграф с классом main. В параграф вложена ссылка. Этой ссылке задаём определённый стиль. Обратимся к 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
эта ссылка
будет размером 18px и красного цвета,
и эта красного цвета.
 	
	.main a {
	font-size: 18px;
	color: red;
	}

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

X **Важно!** *Вложенность всегда будет работать в несколько раз медленнее, поэтому рекомендуется использовать обращение по классам.*

Чтобы добавить стили к любому элементу, нужно задать класс и к нему стиль. Не стоит применять разные методы, если есть один максимально простой и удобный.

Наследование

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

html	css

В этом параграфе весь текст
будеткрасного цвета и шрифтом
размером 18 px
 	
	p {
	font-size: 18px;
	color: red;
	}

Здесь в тег вложены два тега , в один из которых также вложен тег . В этом случае весь текст параграфа 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 будеткрасного цвета и шрифтом размером 18 px, только эта ссылка не будет красной	p { font-size: 18px; color: red; }

Если добавить в пример ещё и ссылку, то она унаследует только свойство font-size, свойство color не унаследует. Узнать, наследуется ли определённое свойство CSS, можно только из справочников. В этом примере, чтобы применить к ссылке свойство color, можно задать значение inherit для свойства color. inherit означает, что элемент должен наследовать это свойство от родителя.

Группирование свойств

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

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

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

Можно столкнуться с ситуацией, когда при разработке сайтов задаётся определённое 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?

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

В этом случае все параграфы будут зелёными. 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.

Второй по приоритетности — стиль, объявленный в атрибуте style любого тега.

html	css

`<h1 style="color: red;">`
`  Заголовок первого уровня`
`</h1>`
	`h1 {`
	`  color: green;`
	`}`

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

Следующий уровень — это уровень приоритета селекторов. Есть такое понятие, как специфичность. Смысл его в том, что браузеру будет начисляться определенное количество баллов за разные типы селекторов, а также их количество. И приоритет получают те стили, которые набирают больше баллов.

1. Селекторы тегов и псевдоэлементы — по 1 баллу (0, 0, 0, 1).
2. Селекторы атрибутов, классы и псевдоклассы — по 10 баллов (0, 0, 1, 0).
3. Идентификаторы — по 100 баллов (0, 1, 0, 0).
4. Атрибут style — 1000 баллов (1, 0, 0, 0).

Пример начисления баллов за специфичность:

Следующие по приоритетности стили указаны в порядке убывания:

- стили, заданные в разделе <head>;
- стили, подключаемы из внешних файлов;
- наследуемые стили от предков.

Дополнительные материалы

1. Статья про добавление стилей на страницу.
2. Интересная статья про селекторы.
3. 30 CSS-селекторов, о которых полезно помнить.
4. http://htmlbook.ru/ — справочник по всем свойствам css.

Используемые источники

1. Способы добавления стилей — ссылка.
2. Дочерние селекторы — ссылка.
3. Селекторы в CSS — ссылка.
4. Единицы измерения — ссылка.