CSS3

Инструктор: Максим

Содержание

1. Общая информация. Типы стилей

CSS (Cascading Style Sheets, каскадные таблицы стилей) – представляют собой набор параметров, управляющих видом и положением элементов Web-страницы

Различают несколько типов стилей, которые могут совместно применяться к одному документу:

- **стиль браузера** – оформление, которое по умолчанию применяется к элементам Web-страницы браузером. Это оформление можно увидеть в случае "голого" HTML, когда к документу не добавляется никаких стилей
- **стиль автора** – стиль, который добавляет к документу его разработчик
- **стиль пользователя** – это стиль, который может включить пользователь сайта через настройки браузера. Такой стиль имеет более высокий приоритет и переопределяет исходное оформление документа

1. Общая информация. Преимущества

Разграничение кода и оформления

Идея в том, чтобы код HTML был свободен от элементов оформления вроде установки цвета, размера шрифта и других параметров. В идеале, Web-страница должна содержать только элементы логического форматирования, а вид элементов задаётся через стили

Разное оформление для разных устройств

С помощью стилей можно определить вид Web-страницы для разных устройств вывода: монитора, принтера, смартфона, планшета и др.

Ускорение загрузки сайта

При хранении стилей в отдельном файле, он кэшируется и при повторном обращении к нему извлекается из **кэша** браузера. За счёт кэширования и того, что стили хранятся в отдельном файле, уменьшается код Web-страниц и снижается время загрузки документов

1. Общая информация. Преимущества

Единое стилевое оформление множества документов

Сайт это не просто набор связанных между собой документов, но и одинаковое расположение основных блоков, и их вид. Применение единообразного оформления заголовков, основного текста и других элементов создает преемственность между страницами и облегчает пользователям работу с сайтом и его восприятие в целом. Разработчикам же использование стилей существенно упрощает проектирование дизайна

Централизованное хранение

Стили, как правило, хранятся в одном или нескольких специальных файлах, ссылка на которые указывается во всех документах сайта. Благодаря этому удобно править стиль в одном месте, при этом оформление элементов автоматически меняется на всех страницах, которые связаны с указанным файлом. Вместо того чтобы модифицировать десятки HTML-файлов, достаточно отредактировать один файл со стилем и оформление нужных документов сразу же поменяется

2. Подключение CSS. Связанные стили

Для добавления стилей на Web-страницу существует несколько способов, которые различаются своими возможностями и назначением

Связанные стили

При использовании связанных стилей, описание стилей располагается в отдельном файле, как правило, с расширением `.css`, а для связывания документа с этим файлом применяется элемент **link**. Данный элемент помещается в head

Значение атрибута элемента link – **rel** остаётся неизменным независимо от кода. Значение атрибута **href** задаёт путь к CSS-файлу, он может быть задан как относительно, так и абсолютно. Обратите внимание, что таким образом можно подключать таблицу стилей, которая находится на другом сайте

Файл со стилем не хранит никаких данных, кроме синтаксиса CSS. В свою очередь и HTML-документ содержит только ссылку на файл со стилем, т.е. таким способом реализуется принцип разделения кода и оформления сайта. Поэтому использование связанных стилей является наиболее универсальным и удобным методом добавления стиля на сайт. Ведь стили хранятся в одном файле, а в HTML-документах указывается только ссылка на него

2. Подключение CSS. Связанные стили

Содержимое HTML файла:

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

Содержимое CSS файла:

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

Текст

2. Подключение CSS. Глобальные стили

Глобальные стили

При использовании глобальных стилей свойства CSS описываются в самом документе и располагаются в заголовке Web-страницы. По своей гибкости и возможностям этот способ добавления стиля уступает предыдущему, но также позволяет хранить стили в одном месте, в данном случае прямо на той же странице с помощью элемента **style**:

```
<head>
    ...
    <style>
        p {
            color: green;
            font-size: 18px;
        }
    </style>
</head>
<body>
    <p>Текст</p>
</body>
```

Текст

2. Подключение CSS. Внутренние стили. Приоритет

Внутренние стили

Внутренний или встроенный стиль является по существу расширением для одиночного элемента используемого на текущей Web-странице. Для определения стиля используется атрибут style, а его значением выступает набор стилевых правил:

```
<body>
    <p style="color: green; font-size: 18px;">Текст</p>
</body>
```

Обратите внимание, все описанные методы использования CSS могут применяться как самостоятельно, так и в сочетании друг с другом. В этом случае необходимо помнить об их иерархии. Первым имеет приоритет **внутренний** стиль, затем **глобальный** стиль и в последнюю очередь **связанный** стиль

2. Подключение CSS. Приоритет

HTML файл:

```
<head>
    ...
    <link rel="stylesheet"
href="style.css" />
    <style>
        h2 {
            color: green;
        }
        h3 {
            color: green;
        }
    </style>
</head>
<body>
    <h1>Заголовок 1</h1>
    <h2>Заголовок 2</h2>
    <h3 style="color: blue;">Заголовок
3</h3>
</body>
```

CSS файл:

```
h1 {
    color: red;
}
h2 {
    color: red;
}
h3 {
    color: red;
}
```

2. Подключение CSS. Типы носителей

Широкое развитие различных платформ и устройств заставляет разработчиков делать под них специальные версии сайтов, что достаточно трудоёмко и проблематично. Вместе с тем, времена и потребности меняются, и создание сайта для различных устройств является неизбежным и необходимым звеном его развития. С учетом этого в CSS введено понятие типа носителя, когда стиль применяется только для определённого устройства

Тип	Описание
all	Все типы. Используется по умолчанию
aural	Речевые синтезаторы, а также программы для воспроизведения текста вслух, например, речевые браузеры
braille	Устройства, основанные на системе Брайля, которые предназначены для слепых людей
handheld	Наладонные компьютеры и аналогичные им аппараты, например, смартфоны
print	Печатающие устройства вроде принтера
projection	Проектор
screen	Экран монитора
tv	Телевизор

2. Подключение CSS. Типы носителей

Команда **@media** позволяет указать тип носителя для **глобальных** или **связанных** стилей и имеет следующий синтаксис:

```
<style>
    @media screen {
        body {
            color: blue;
        }
    }
    @media print {
        body {
            color: green;
        }
    }
</style>
```

2. Подключение CSS. Типы носителей

Команда @media применяется в основном для формирования одного стилевого файла, который разбит на блоки по типу устройств. Иногда же имеет смысл создать несколько разных CSS-файлов – один для печати, другой для отображения в браузере – и подключать их к документу по мере необходимости. В подобном случае следует воспользоваться элементом **link** с атрибутом **media**:

```
<link rel="stylesheet" href="main.css" media="screen" />
<link rel="stylesheet" href="print.css" media="print,
handheld" />
```

2. Подключение CSS. Полезные советы

Какой способ подключения CSS лучше использовать и когда?

- Используйте атрибут style для какого либо элемента если этот элемент с отличным от других элементов стилем один единственный на всём сайте
- Используйте элемент style со стилевым описанием, в том случае, если страница должна иметь индивидуальный дизайн в корне отличный от других страниц сайта
- В большинстве случаев разумно выносить каскадную таблицу стилей в отдельный CSS-файл

3. Значения стилевых свойств. Строки

Всё многообразие значений стилевых свойств может быть сведено к определённому типу: **строка**, **число**, **проценты**, **размер**, **цвет**, **адрес** или **ключевое слово**

Строки

Любые строки необходимо брать в двойные или одинарные кавычки. Если внутри строки требуется оставить одну или несколько кавычек, то можно комбинировать типы кавычек или добавить перед кавычкой слэш:

```
'Гостиница "Турист"'
"Гостиница 'Турист'"
"Гостиница \"Турист\""
```

В данном примере в первой строке применяются одинарные кавычки, а слово "Турист" взято в двойные кавычки. Во второй строке всё с точностью до наоборот, в третьей же строке используются только двойные кавычки, но внутренние экранированы с помощью слэша

```
div::after {
    content: ".";
}
```

3. Значения стилевых свойств. Числа

Числа

Значением может выступать целое число, содержащее цифры от 0 до 9 и десятичная дробь, в которой целая и десятичная часть разделяются точкой:

```
p {
    font-weight: 600; /* Насыщенность шрифта(жирность) */
    line-height: 1.2; /* Межстрочный интервал */
}
```

Обратите внимание, если в десятичной дроби целая часть равна нулю, то её разрешается не писать. Запись .7 и 0.7 равнозначна

3. Значения стилевых свойств. Проценты

Проценты

Процентная запись обычно применяется в тех случаях, когда надо изменить значение относительно родительского элемента или когда размеры зависят от внешних условий:

```
table {
    width: 100%;
}
```

Обратите внимание, проценты не обязательно должны быть целым числом, допускается использовать десятичные дроби, вроде значения 56.8%

3. Значения стилевых свойств. Размеры

Размеры

Для задания размеров различных элементов, в CSS используются **относительные** и **абсолютные** единицы измерения

Относительные единицы

Относительные единицы обычно используют для работы с текстом

Единица **em** это изменяемое значение, которое зависит от размера шрифта текущего элемента (свойство font-size). В каждом браузере заложен размер текста, применяемый в том случае, когда этот размер явно не задан. Поэтому изначально 1em равен размеру шрифта, заданного в браузере по умолчанию или размеру шрифта родительского элемента. Процентная запись идентична em, в том смысле, что значения 1em и 100% равны

```
.cl1 { font-size: 1.5em; }
```

Единица **rem** задаёт размер относительно размера шрифта элемента html

```
.cl2 { font-size: 2rem; }
```

3. Значения стилевых свойств. Размеры

Абсолютные единицы

Абсолютные единицы представляют собой физические размеры – дюймы, сантиметры, миллиметры, пункты, пики, а также пиксели. Для устройств с низким dpi (количество точек приходящихся на один дюйм, определяет плотность точек) привязка идёт к пикселю. В этом случае один дюйм равен 96 пикселям. Очевидно, что реальный дюйм не будет совпадать с дюймом на таком устройстве. На устройствах с высоким dpi реальный дюйм совпадает с дюймом на экране, поэтому размер пикселя вычисляется как 1/96 от дюйма

Самой распространенной абсолютной единицей является **пиксель**

```
.cl3 { height: 20px; }
```

Обратите внимание, при установке размеров обязательно указывайте единицы измерения, например `width: 30px`. В противном случае браузер не сможет показать желаемый результат, поскольку не понимает, какой размер вам требуется. Единицы не добавляются только при нулевом значении (`margin: 0`)

3. Значения стилевых свойств. Цвет

Цвет

Цвет в стилях можно задавать следующими способами: по **шестнадцатеричному значению**, по **названию**, в **формате RGB**, в **формате HSL** и в **формате RGB/HSL с альфа каналом**

По шестнадцатеричному значению

Для задания цветов используются числа в шестнадцатеричном коде. Цифры будут: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Цифры от 10 до 15 заменены латинскими буквами. Числа больше 15 в шестнадцатеричной системе образуются объединением двух чисел в одно. Например, числу 255 в десятичной системе соответствует число FF в шестнадцатеричной системе. Чтобы не возникало путаницы в определении системы счисления, перед шестнадцатеричным числом ставят символ решетки **#**, например #666999. Каждый из трех цветов – красный, зеленый и синий – может принимать значения от 00 до FF. Таким образом, обозначение цвета разбивается на три составляющие #rrggbb, где первые два символа отмечают красную компоненту цвета, два средних – зелёную, а два последних – синюю. Допускается использовать сокращенную форму вида #rgb, где каждый символ следует удваивать (#rrggbb). К примеру, запись #fe0 расценивается как #ffee00:

```
p { color: #FF0000; }
```

3. Значения стилевых свойств. Цвет

По названию

Браузеры поддерживают некоторые цвета по их названию:

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

Имя	Цвет	Код	Описание
white		#ffffff или #fff	Белый
silver		#c0c0c0	Серый
gray		#808080	Тёмно-серый
black		#000000 или #000	Чёрный
maroon		#800000	Тёмно-красный
red		#ff0000 или #f00	Красный
orange		#ffa500	Оранжевый
yellow		#ffff00 или #ff0	Жёлтый
olive		#808000	Оливковый

Имя	Цвет	Код	Описание
lime		#00ff00 или #0f0	Светло-зелёный
green		#008000	Зелёный
aqua		#00ffff или #0ff	Голубой
blue		#0000ff или #00f	Синий
navy		#000080	Тёмно-синий
teal		#008080	Сине-зелёный
fuchsia		#ff00ff или #f0f	Розовый
purple		#800080	Фиолетовый

3. Значения стилевых свойств. Цвет

С помощью RGB

Можно определить цвет, используя значения красной (Red), зелёной (Green) и синей (Blue) составляющей в десятичном исчислении. Значение каждого из трех цветов может принимать значения от 0 до 255. Также можно задавать цвет в процентном отношении. Вначале указывается ключевое слово **rgb**, а затем в скобках, через запятую указываются компоненты цвета:

```
p {
    color: rgb(255, 0, 0);
}
h1 {
    color: rgb(100%, 20%, 20%);
}
```

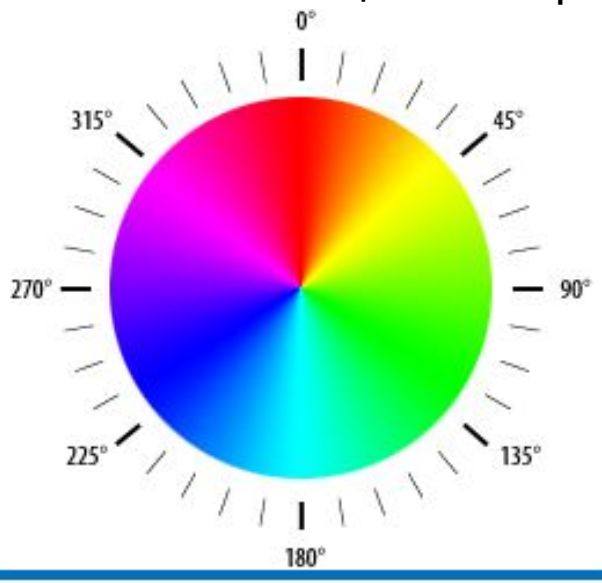
3. Значения стилевых свойств. Цвет

С помощью HSL

HSL – это модель, в которой цвет определяется тремя параметрами: **оттенком** (**тоном**), **насыщенностью** и **светлотой**

Оттенок

Для того чтобы определить тон (Hue) нужно указать градус поворота (от 0° до 359°) цветового спектра замкнутого в цветовой круг. На рисунке наглядно показано откуда берётся этот угол. Радуга замкнутая в круг в котором красный всегда ориентирован на север и равен 0 градусам (ну и 360° тоже), 120 градусов это зелёный, 240° синий – это основные три цвета, смешиваясь, образуют ещё три дополнительных 60° жёлтый 180° голубой и 300° фиолетовый. Между этими шестью основными и дополнительными цветами расположены все остальные оттенки цветового спектра



3. Значения стилевых свойств. Цвет

Насыщенность

Второе значение (Saturation) цветовой модели HSL определяет **насыщенность** выбранного оттенка и указывается в процентах в диапазоне от 0% до 100%. Чем ближе данное значение к 100% тем цвет выглядят более чисто и "сочно" и наоборот если насыщенность стремится к 0% то цвет "линяет" и становится серым

Светлота

Светлота или **яркость** (Lightness) это **третий** параметр HSL. Точно так же как и насыщенность указывается в процентах, чем выше процент, тем ярче становится цвет. Крайние значения 0% и 100% будут обозначать соответственно чёрный (отсутствие света) и белый (засвеченный) цвета, причём неважно, какой оттенок из цветового круга был выбран изначально. Оптимальное значение яркости цвета равняется 50%

```
.t1 { color: hsl(0, 100%, 50%); }
.t2 { color: hsl(0, 50%, 50%); }
.t3 { color: hsl(0, 0%, 50%); }
.t4 { color: hsl(0, 100%, 100%); }
.t5 { color: hsl(0, 100%, 0%); }
```

Текст
Текст
Текст

Текст

3. Значения стилевых свойств. Цвет

С помощью RGB/HSL с альфа каналом

Альфа канал – позволяет сделать цвет прозрачным. Степень прозрачности, определяется значением от 0 до 1, где 0 – полностью прозрачный, а 1 – полностью непрозрачный

Пример RGB с альфа каналом:

```
p {
    color: rgba(255, 0, 255, 0.5);
}
```

Текст

Пример HSL с альфа каналом:

```
p {
    color: hsla(120, 100%, 50%, 0.9);
}
```

Текст

3. Значения стилевых свойств. Адреса. Ключевые слова

Адреса

Адреса (URI) применяются для указания пути к файлу, например, для установки фоновой картинки на странице. Для этого применяется ключевое слово **url**, внутри скобок пишется относительный или абсолютный адрес файла. При этом адрес можно задавать в необязательных одинарных или двойных кавычках:

```
body {
    background: url(fon.png) no-repeat;
}
```

Ключевые слова

В качестве значений активно применяются ключевые слова, которые определяют желаемый результат действия стилевых свойств. Ключевые слова пишутся без кавычек

Правильно: `p { text-align: right; }`
Неправильно: `p { text-align: "right"; }`

4. Селекторы

Стилевые правила записываются в своём формате, отличном от HTML. Основным понятием выступает **селектор** – это некоторое имя стиля, для которого добавляются параметры форматирования. В качестве селектора выступают элементы, классы и идентификаторы

Вначале пишется имя селектора, например, body, это означает, что все стилевые параметры будут применяться к элементу body, затем идут фигурные скобки, в которых записывается стилевое свойство, а его значение указывается после двоеточия. Стилевые свойства разделяются между собой точкой с запятой, в конце этот символ можно опустить

Обратите внимание, CSS не чувствителен к регистру, переносу строк, пробелам и символам табуляции, поэтому форма записи зависит от желания разработчика

4. Селекторы. Правила применения стилей

Существуют правила, которые необходимо знать при описании стиля

Форма записи

Для селектора допускается добавлять каждое стилевое свойство и его значение по отдельности:

```
td { background: olive; }
td { color: white; }
td { border: 1px solid black; }
```

Однако такая запись не очень удобна. Приходится повторять несколько раз один и тот же селектор, да и легко запутаться в их количестве. Поэтому все свойства для каждого селектора лучше писать вместе:

```
td {
   background: olive;
   color: white;
   border: 1px solid black;
}
```

Эта форма записи более наглядная и удобная в использовании

4. Селекторы. Правила применения стилей

Чем ниже значение, тем выше приоритет

Если для селектора вначале задаётся свойство с одним значением, а затем то же свойство, но уже с другим значением, то применяться будет то значение, которое в коде установлено ниже:

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

Для селектора p цвет текста вначале установлен зелёным, а затем красным. Поскольку значение red расположено ниже, то оно в итоге и будет применяться к тексту

Обратите внимание, на самом деле такой записи лучше вообще избегать и удалять повторяющиеся значения. Но подобное может произойти случайно, например, в случае подключения разных стилевых файлов, в которых содержатся одинаковые селекторы

4. Селекторы. Правила применения стилей

Значения

У каждого свойства может быть только соответствующее его функции значение. Например, для color, который устанавливает цвет текста, в качестве значений недопустимо использовать одно число

Комментарии

Комментарии нужны, чтобы делать пояснения по поводу использования того или иного стилевого свойства, выделять разделы или писать свои заметки. Комментарии позволяют легко вспоминать логику и структуру селекторов, и повышают разборчивость кода. Обратите внимание, вместе с тем, добавление текста увеличивает объём документов, что отрицательно сказывается на времени их загрузки. Поэтому комментарии обычно применяют в отладочных или учебных целях, а при выкладывании сайта в сеть их стирают

Чтобы пометить, что текст является комментарием, применяют следующую конструкцию `/* Текст комментария */`

4. Селекторы. Классы

Классы применяют, когда необходимо определить стиль для индивидуального элемента Web-страницы или задать разные стили для одного элемента:

```
p.c1 {
    color: red;
}
<p class="c1">Текст</p>
```

Текст

Внутри стиля вначале пишется желаемый элемент, а затем, через точку пользовательское имя класса. Имена классов должны начинаться с латинского символа и могут содержать в себе символ дефиса (-) и подчеркивания (_). Использование русских букв в именах классов недопустимо. Чтобы указать в коде HTML, что элемент используется с определённым классом, к элементу добавляется атрибут class=" имя_класса“

4. Селекторы. Классы

Можно, также, использовать классы и без указания элемента:

```
.c1 {
    color: red;
}
<h1 class="c1">Заголовок</h1>
<p class="c1">Текст</p>
```

При такой записи, стиль применится ко всем элементам с заданным классом

4. Селекторы. Использование разных классов

К любому элементу одновременно можно добавить несколько классов, перечисляя их в атрибуте class через пробел. В этом случае к элементу применяется стиль, описанный в правилах для каждого класса:

```
.c1 {
    color: red;
}
.c2 {
    background-color: blue;
}
<h1 class="c1 c2">Заголовок</h1>
<p class="c1 c2">Текст</p>
```

Обратите внимание, в стилях также допускается использовать запись вида `.layer1.layer2`, где layer1 и layer2 представляют собой имена классов. Стиль применяется только для элементов, у которых одновременно заданы классы layer1 и layer2

4. Селекторы. Идентификаторы

Идентификатор определяет уникальное имя элемента, которое используется для изменения его стиля и обращения к нему через скрипты:

```
#p1 {
    color: blue;
}
<p id="p1">Текст</p>
```

Текст

При описании идентификатора вначале указывается символ решётки **#**, затем идет имя идентификатора. Оно должно начинаться с латинского символа и может содержать в себе символ дефиса (-) и подчеркивания (_). Использование русских букв в именах идентификатора недопустимо. В отличие от классов идентификаторы должны быть уникальны, иными словами, встречаться в коде документа только один раз

Обращение к идентификатору происходит аналогично классам, но в качестве ключевого слова у элемента используется атрибут id, значением которого выступает имя идентификатора. Символ решётки при этом уже не указывается

4. Селекторы. Вложенные

При создании Web-страницы часто приходится вкладывать одни элементы внутрь других. Чтобы стили для этих элементов использовались корректно, помогут селекторы, которые работают только в определённом контексте. Таким образом можно одновременно установить стиль для отдельного элемента, а также для элемента, который находится внутри другого. **Вложенный** селектор состоит из простых селекторов разделенных пробелом. Синтаксис следующий:

```
селектор1 селектор2 { Описание правил стиля }
```

В этом случае стиль будет применяться к селектор2 когда он размещается внутри селектор1:

```
p strong {
    color: blue;
}
<p><strong>Текст</strong> <em><strong>Текст</strong></em></p>
```

Текст ***Текст***

Обратите внимание, не обязательно контекстные селекторы содержат только один вложенный селектор. В зависимости от ситуации допустимо применять два и более последовательно вложенных друг в друга селекторов:

```
p em strong {
    color: blue;
}
```

Текст ***Текст***

4. Селекторы. Дочерние

Дочерним селектором считается такой, который в дереве элементов находится прямо внутри родительского элемента. Синтаксис следующий:

```
селектор1 > селектор2 { Описание правил стиля }
```

Стиль применяется к селектор2, но только в том случае, если он является дочерним для селектор1:

```
.c1 > p {
    color: red;
}
<div class="c1">
    <p>Текст</p>
    <div>
        <p>Текст</p>
    </div>
</div>
```

Текст

Текст

По своей логике дочерние селекторы похожи на селекторы контекстные. Разница между ними следующая. Стиль к дочернему селектору применяется только в том случае, когда он является прямым потомком, иными словами, непосредственно располагается внутри родительского элемента. Для контекстного селектора же допустим любой уровень вложенности

4. Селекторы. Соседние

Соседними называются элементы Web-страницы, когда они следуют непосредственно друг за другом в коде документа. Для управления стилем соседних элементов используется символ плюса **+**, который устанавливается между двумя селекторами. Синтаксис следующий:

```
селектор1 + селектор2 { Описание правил стиля }
```

Стиль при такой записи применяется к селектор2, но только в том случае, если он является соседним для селектор1 и следует сразу после него:

```
.c1 + .c2 {
    color: blue;
}
<p>
    <em class="c1">Текст</em>
    <em class="c2">Текст</em>
    <em class="c2">Текст</em>
</p>
```

Текст Текст Текст

4. Селекторы. Смежные

Смежный селектор позволяет выбрать элементы, которые будут отобраны на основе их родственных элементов, т.е. те, у которых один и тот же общий родитель. Он создаются с помощью символа тильды ~ между двумя элементами внутри селектора. Первый элемент определяет, что второй элемент должен быть родственным с ним, и у обоих должен быть один и тот же родитель:

```
селектор1 ~ селектор2 { Описание правил стиля }
```

Стиль при такой записи применяется к селектор2, которые следуют после любых элементов селектор1:

```
#id1 ~ em {
    color: red;
}
<p>
    <b id="id1">Текст</b>
    <em>Текст</em>
    <span>Текст</span>
    <em>Текст</em>
</p>
```

Текст *Текст* Текст *Текст*

4. Селекторы. На основе атрибутов

Многие элемент различаются по своему действию в зависимости от того, какие в них используются атрибуты. Чтобы гибко управлять стилем подобных элементов, в CSS введены селекторы атрибутов. Они позволяют установить стиль по присутствию определённого атрибута элемента или его значения

Простой селектор атрибута

Устанавливает стиль для элемента, если задан специфичный атрибут элемента. Его значение в данном случае не важно. Синтаксис следующий:

```
[атрибут] { Описание правил стиля }
Селектор[атрибут] { Описание правил стиля }
```

Пример:

```
[href] { color: red; }
p[title] { color: blue; }
<a href="#">Ссылка</a>
<p title="text">Текст</p>
```

4. Селекторы. На основе атрибутов

Атрибут со значением

Устанавливает стиль для элемента в том случае, если задано определённое значение специфичного атрибута. Синтаксис следующий:

```
[атрибут="значение"] { Описание правил стиля }
Селектор[атрибут="значение"] { Описание правил стиля }
```

Пример:

```
[hreflang="en-US"] { color: green; }
p[title="text"] { color: blue; }
<a href="#" hreflang="en-US">Ссылка</a>
<p title="text">Текст</p>
```

4. Селекторы. На основе атрибутов

Значение атрибута начинается с определённого текста

Устанавливает стиль для элемента в том случае, если значение атрибута элемента начинается с указанного текста. Синтаксис следующий:

```
[атрибут^="значение"] { Описание правил стиля }
Селектор[атрибут^="значение"] { Описание правил стиля }
```

Пример:

```
[title^="te"] { color: red; }
a[href^="http://"] { color: green; }
<p title="text">Текст</p>
<a href="http://example.com">Ссылка</a>
```

4. Селекторы. На основе атрибутов

Значение атрибута оканчивается определённым текстом

Устанавливает стиль для элемента в том случае, если значение атрибута оканчивается указанным текстом. Синтаксис следующий:

```
[атрибут$="значение"] { Описание правил стиля }
Селектор[атрибут$="значение"] { Описание правил стиля }
```

Пример:

```
[title$="xt"] { color: green; }
a[href$=".com"] { color: red; }
<p title="text">Текст</p>
<a href="http://example.com">Ссылка</a>
```

4. Селекторы. На основе атрибутов

Значение атрибута содержит указанный текст

Возможны варианты, когда стиль следует применить к элементу с определённым атрибутом, при этом частью его значения является некоторый текст. При этом точно не известно, в каком месте значения включен данный текст – в начале, середине или конце. В подобном случае следует использовать синтаксис:

```
[атрибут*="значение"] { Описание правил стиля }
Селектор[атрибут*="значение"] { Описание правил стиля }
```

Пример:

```
[title*="ex"] { color: blue; }
a[href*="example"] { color: red; }
<p title="text">Текст</p>
<a href="http://example.com">Ссылка</a>
```

4. Селекторы. На основе атрибутов

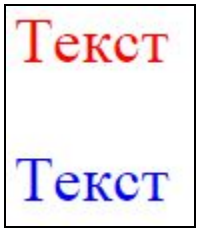
Одно из нескольких значений атрибута

Некоторые значения атрибутов могут перечисляться через пробел, например имена классов. Чтобы задать стиль при наличии в списке требуемого значения применяется следующий синтаксис:

```
[атрибут~="значение"] { Описание правил стиля }
Селектор[атрибут~="значение"] { Описание правил стиля }
```

Пример:

```
[class~="c1"] { color: red; }
p[class~="c3"] { color: blue; }
<p class="c1 c2">Текст</p>
<p class="c2 c3">Текст</p>
```



4. Селекторы. На основе атрибутов

Дефис в значении атрибута

В именах идентификаторов и классов разрешено использовать символ дефиса (-), что позволяет создавать значащие значения атрибутов id и class. Для изменения стиля элементов, в значении которых применяется дефис, следует воспользоваться следующим синтаксисом:

```
[атрибут|="значение"] { Описание правил стиля }
Селектор[атрибут|="значение"] { Описание правил стиля }
```

Пример:

```
[class|="my1"] { color: blue; }
p[class|="my2"] { color: green; }
<p class="my1-class">Текст</p>
<p class="my2-class">Текст</p>
```

4. Селекторы. На основе атрибутов

Все перечисленные методы можно комбинировать между собой, определяя стиль для элементов, которые содержат два и более атрибута. В подобных случаях квадратные скобки идут подряд. Синтаксис следующий:

```
[атрибут1="значение1"][атрибут2="значение2"] { Описание правил стиля }
Селектор[атрибут1="значение1"][атрибут2="значение2"] { Описание правил стиля }
```

Пример:

```
[class="c1"][title] { color: green; }
p[class="c2"][title="text"] { color: red; }
<p class="c1" title="myTitle">Текст</p>
<p class="c2" title="text">Текст</p>
```

4. Селекторы. Универсальный

Иногда требуется установить одновременно **один стиль для всех элементов** Web-страницы, например, задать шрифт или начертание текста. В этом случае поможет универсальный селектор, который соответствует любому элементу Web-страницы. Для обозначения универсального селектора применяется символ звёздочки * и в общем случае синтаксис будет следующий:

```
* { Описание правил стиля }
```

Пример:

```
* { color: blue; }
<p>Текст</p>
```

Текст

Обратите внимание, в некоторых случаях указывать универсальный селектор не обязательно. Так, например, записи `*.class` и `.class` являются идентичными по своему результату

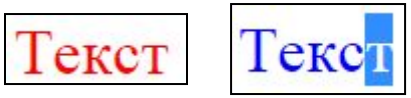
4. Селекторы. Псевдоклассы

Псевдоклассы определяют динамическое состояние элементов, которое изменяется с помощью действий пользователя, а также положение в дереве документа. Примером такого состояния служит текстовая ссылка, которая меняет свой цвет при наведении на неё курсора мыши. При использовании псевдоклассов браузер не перегружает текущий документ, поэтому с помощью псевдоклассов можно получить разные динамические эффекты на странице. Синтаксис следующий:

```
Селектор:Псевдокласс { Описание правил стиля }
:Псевдокласс { Описание правил стиля }
```

Пример:

```
p { color: red; }
p:hover { color: blue; }
<p>Текст</p>
```



Вначале указывается селектор, к которому добавляется псевдокласс, затем следует двоеточие, после которого идёт имя псевдокласса. Допускается применять псевдоклассы к именам идентификаторов или классов, а также к контекстным селекторам. Обратите внимание, если псевдокласс указывается без селектора впереди, то он будет применяться ко всем элементам документа

4. Селекторы. Псевдоклассы

Некоторые псевдоклассы:

- **:link** – отвечает за стили не посещенной ссылки
- **:hover** – состояние объекта (не обязательно ссылки) при наведении на него мышкой
- **:active** – состояние активного объекта (например, для ссылки и зажатие ее мышкой)
- **:visited** – состояние посещенной ссылки
- **:focus** – когда используется какой-то объект на страницы, то на нем устанавливается фокус (в случае и текстовым поле это постановка курсора в это поле)
- **:first-child** – первый дочерний элемент текущего элемента
- **:last-child** – последний дочерний элемент текущего элемента

4. Селекторы. Псевдоэлементы

Псевдоэлементы – это особый вид свойств CSS, которые позволяют работать не над самим элементом, а над его отдельной частью. В CSS3 псевдоэлементы начинаются с двух двоеточий `::`

Синтаксис следующий:

```
Селектор::Псевдоэлемент { Описание правил стиля }
```

Пример:

```
p::first-letter { color: red; }
<p>Текст</p>
```

Текст

Вначале следует имя селектора, затем пишется двоеточие, после которого идёт имя псевдоэлемента. Каждый псевдоэлемент может применяться только к одному селектору, если требуется установить сразу несколько псевдоэлементов для одного селектора, правила стиля должны добавляться к ним по отдельности

4. Селекторы. Псевдоэлементы

Список псевдоэлементов:

- **::after** – используется для вывода желаемого контента после элемента, к которому он добавляется
- **::before** – применяется для отображения желаемого контента до элемента, к которому он добавляется
- **::first-letter** – определяет стиль первого символа в тексте элемента, к которому добавляется
- **::first-line** – задает стиль первой строки форматированного текста
- **::selection** – применяет стиль к выделенному пользователем фрагменту текста

4. Селекторы. Группирование

При создании стиля для сайта, когда одновременно используется множество селекторов, возможно появление повторяющихся стилевых правил. Чтобы не повторять дважды одни и те же элементы, их можно сгруппировать для удобства представления и сокращения кода

Селекторы группируются в виде списка селекторов, разделенных между собой запятыми. Синтаксис следующий:

```
Селектор 1, Селектор 2, ... Селектор N { Описание правил стиля }
```

Пример:

```
h1, h2, h3 { font-family: Arial; }
h1 { color: red; }
h2 { color: green; }
h3 { color: blue; }
```

При такой записи правила стиля применяются ко всем селекторам, перечисленным в одной группе

4. Селекторы. Полезные советы

При построении CSS будьте логичны, соблюдайте "значимость" элементов и их порядок, так же как они вложены друг в друга в HTML коде

Например:

```
body { сначала опишите стиль страницы в целом }
div { потом её отдельных частей – блоков }
a { затем ссылок }
h1 – h6 { далее заголовков }
p { и в конце параграфов }
```

Для чего это нужно?

- Просто для удобного чтения и "навигации" по CSS описанию. Когда потребуется найти какой-нибудь элемент, уже изначально будет представление где он приблизительно находится, в начале, середине, или конце
- Загрузка страницы происходит не моментально и не всегда приятно наблюдать как содержание данной страницы при загрузке "прыгает" и всячески "шевелится" так как сначала прописываются "малозначимые" стили элементов, например шрифт параграфов, а в конце "значительные" например размеры блоков, с помощью которых свёрстан весь сайт. К тому же загрузка, по каким либо причинам, вообще может пройти не до конца

4. Селекторы. Полезные советы

При использовании классов и идентификаторов придумывайте им осмысленные информативные имена

Варианты `.aaa .a12 #abc #cba` приведут к путанице, а также возможно в Вашем коде будет разбираться посторонний человек

Придумайте свою "систему" названий и не нарушайте её, так сэкономите собственное время и затраченные усилия

5. Наследование

При указании стиля для элемента часть свойств может быть унаследована его дочерними элементами и потомками, этот эффект называется **наследованием**

Например, все элементы расположенные внутри элемента body являются его дочерними элементами и потомками. Если в стиле для body задать с помощью свойства color красный цвет текста, то цвет текста всех его дочерних элементов и потомков тоже станет красным

Наследуемые свойства можно переопределить, применив индивидуальное правило для нужного элемента

Чтобы узнать, какие CSS-свойства наследуются, а какие нет, нужно смотреть описание конкретного свойства в CSS-справочнике (например: https://webref.ru/css)

6. Специфичность

Специфичность селекторов определяет их приоритетность в таблице стилей. Чем специфичнее селектор, тем выше его приоритет. Для вычисления специфичности селектора используются три группы чисел (a, b, c), расчёт производится следующим образом:

- Считается число идентификаторов в селекторе (=a)
- Считается число селекторов классов, атрибутов и псевдоклассов в селекторе (=b)
- Считается число селекторов типа и псевдоэлементов в селекторе (=c)
- Селектор внутри псевдокласса отрицания (:not) считается как любой другой селектор, но сам псевдокласс отрицания не участвует в вычислении селектора
- Универсальный селектор (*) и комбинаторы не участвуют в вычислении веса селектора

6. Специфичность

В примере селекторы расположены в порядке увеличения их специфичности:

```
1. *                /* a=0 b=0 c=0 -> специфичность = 0 0 0 */
2. li               /* a=0 b=0 c=1 -> специфичность = 0 0 1 */
3. ul li            /* a=0 b=0 c=2 -> специфичность = 0 0 2 */
4. ul ol+li         /* a=0 b=0 c=3 -> специфичность = 0 0 3 */
5. h1 + *[rel=up]   /* a=0 b=1 c=1 -> специфичность = 0 1 1 */
6. ul ol li.red     /* a=0 b=1 c=3 -> специфичность = 0 1 3 */
7. li.red.level     /* a=0 b=2 c=1 -> специфичность = 0 2 1 */
8. #x34y            /* a=1 b=0 c=0 -> специфичность = 1 0 0 */
9. #s12:not(p)      /* a=1 b=0 c=1 -> специфичность = 1 0 1 */
```

Стиль для элемента, определённый внутри атрибута style, имеет больший приоритет, чем любой селектор, определённый в таблице стилей. Однако, если для конкретного свойства в таблице стилей указать специальное объявление **!important**, то оно будет иметь больший приоритет, чем значение аналогичного свойства, указанного в атрибуте style

6. Специфичность. !important

Синтаксис применения !important следующий:

```
Свойство: значение !important
```

Пример:

```
* {
    color: blue !important;
}
```

Вначале пишется желаемое стилевое свойство, затем через двоеточие его значение и в конце после пробела указывается ключевое слово !important

7. Каскадность

Каскадность это фундаментальная особенность CSS, с помощью которой браузер определяет значения каких свойств будут применены к элементу при возникновении конфликта свойств. Конфликт свойств возникает, когда для одного элемента определено более одного правила с одинаковым приоритетом и они содержат одинаковые свойства, но с разными значениями

Каскадность работает следующим образом: если в таблице стилей для одного элемента определено несколько правил, селекторы которых имеют одинаковую специфичность и они содержат конфликтующие свойства то, для элемента устанавливаются значения тех свойств, которые расположены ниже в таблице стилей

Если разные правила для одного элемента содержат свойства, которые не конфликтуют, то они объединяются в один стиль, т.е. каждое новое правило добавляет новую информацию о стиле к тому правилу, которое находится перед ним

8. Кроссбраузерная верстка

Кроссбраузерная верстка – это верстка сайта, которая позволяет отображать его во всех браузерах одинаково

Причины, почему разные браузеры воспринимают написанных код по-разному:

- Есть **стандарты HTML**, которым может не соответствовать как верстка так и определенные версии браузеров, которыми все равно пользуются
- **Прогресс движется вперед**, появляются новые технологии, которые позволяют делать более стильные и функциональные сайты. Соответственно появляются новые библиотеки, а также правила CSS, которые будут хорошо восприниматься только обновленными версиями браузеров
- **Internet Explorer**, который играет по своим каким-то правилам. Этот браузер половину правил не понимает. Вот и приходилось и приходится верстальщикам использовать различные не стандартные решения, чтобы решить проблемы IE
- **Стили браузера**. В разных браузерах могут быть разные стилевые значения для элементов по умолчанию

8. Кроссбраузерная верстка. Полезные советы

Как решить все эти проблемы и научиться кроссбраузерной верстки? Практиковаться и искать в сети решения, почему определенный код не хочет понимать определенный браузер

Полезные советы:

- Для того чтобы **решить проблему со стилями браузеров**, можно переопределить стили браузера по умолчанию ("сброс" CSS), что позволит построить стили CSS на единой основе, примеры сбросов:

```
* { padding: 0; margin: 0; } /* Простой сброс */

* { /* Сброс всего необходимого */
   vertical-align: baseline;
   font-weight: inherit;
   font-family: inherit;
   font-style: inherit;
   font-size: 100%;
   border: 0 none;
   outline: 0;
   padding: 0;
   margin: 0;
}
```

8. Кроссбраузерная верстка. Полезные советы

- Проверяйте работу **CSS в разных браузерах** (желательно не забывать за IE 8 версии)

- Определите **стратегию верстки**. Можно сразу сверстать сайт под один из браузеров, а потом подгонять к остальным, либо же верстать постепенно и каждый элемент проверять во всех браузерах, при необходимости корректируя

- Чтобы быстрее находить ошибки в верстке для того или иного браузера, **используйте инструменты разработчика**, которые предоставляют браузеры

Источники

http://webref.ru/
http://professorweb.ru/

Спасибо за внимание!