# **Основы CSS**

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

* 1. **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.

Зачем нужен CSS?

Все очень просто. В основе большинства веб-ресурсов лежит язык гипертекстовой разметки HTML. Ранее он отвечал как за содержимое веб-страниц, так и за их оформление. Однако сайт, сделанный только на HTML, выглядит, мягко говоря, не очень красиво. И если захотите добавить впоследствии на свой ресурс какой-то новый элемент или изменить уже имеющийся, придется делать много исправлений в коде. К примеру, чтобы изменить размер и цвет шрифта, придется переверстать все страницы сайта. А если сайт большой, на нем сотни страниц? Представляете, сколько понадобится времени и усилий?

Тогда в помощь языку гипертекстовой разметки был создан язык стилевой разметки (так иногда называют CSS). HTML стал отвечать только за содержание веб-документа. Задачи по оформлению сайтов выполняют таблицы стилей CSS.

За что конкретно отвечают таблицы стилей CSS?

1. Цветовое оформление веб-страниц.

2. Шрифты (размер, отступы, цвета, оформление заголовков и пр.).

3. Размещение разных блоков и элементов дизайна.

4. Внешний вид таблиц и др.

Итак, с целью облегчить жизнь веб-мастеру и улучшить дизайн сайтов появились CSS таблицы, отвечающие за оформление веб-страниц. Появление CSS позволило разделить содержание и оформление веб-документов. Это ускорило процесс создания сайтов благодаря наличию единых стилей оформления для большого количества страниц.

Кроме того, каскадные таблицы стилей дают возможность быстро изменять оформление веб-ресурса или вносить в него правки. Так что с помощью одного файла CSS теперь можно быстро управлять оформлением огромного количества веб-страниц.

Важное преимущество таблиц стилей – оформление веб-страницы для разных устройств. Стоит иметь ввиду, что для разных устройств существуют разные таблицы стилей. То есть сразу создается несколько дизайнов (для печати, просмотра на экране компьютера, мобильных устройствах). В зависимости от того, на каких устройствах планируется вывод сайта, CSS пишется для каждого устройства один раз при создании сайта.

Без CSS оформление пришлось бы прописывать для каждой отдельной веб-страницы. Теперь же каждая страница сайта просто ссылается на CSS файл. Чтобы изменить что-либо в оформлении нашего сайта, надо внести изменения в код файла CSS.

* 1. **Синтаксис СSS**

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

Селектор {

Свойство1: значение1;

Свойство2: значение2;

}

Свойств и значений может быть сколько угодно, хоть 50. CSS в этом плане ничуть не сложнее HTML.

Пример:

p {

color: blue;

}

Этот код добавит всем параграфам на странице синий цвет.

Как оформлять: можно все свойства и значения селектора указать в одной строке, это улучшит быстродействие, но ухудшит читаемость кода. Для редактирования это будет очень неудобно. Так никто не делает. Гораздо удобнее указывать каждое свойство и его значение на отдельной строке.

* 1. **Комментарии в CSS**

/\* Внешний вид\*/

p {

color : blue;

}

/\*

Стили для параграфа

\*/

В CSS можно также указывать комментарии для комментирования того, свойства каких элементов будут описываться, или для комментирования самих стилей при редактировании документа. Также комментариями можно закрывать блоки кода, если они не нужны в данный момент *(сезонное оформление сайта, к примеру).*

* 1. **Способы объявления CSS**

Есть три способа объявления:

* Inline стили
* Стили в разделе head
* Подключение внешнего файла

**Inline**: прямо в теле html указываем атрибут style и его значения, например, чтобы изменить цвет заголовка на красный, надо в теге заголовка прописать:

<H3 style="color: red;" >Свойства товара</H3>

Чтобы остальные заголовки тоже были такого же стиля, необходимо в каждом прописать этот атрибут с такими же свойствами. Кстати, html позволяет редактировать одновременно в нескольких местах, надо зажать клавишу ctrl. Стиль можно применять также и к ссылкам.

Плюсы inline стилей: можно записать стили достаточно быстро, подключил для нужного элемента и готово.

Минусы: необходимость объявления для каждого элемента отдельно. Минус серьезный, особенно для больших сайтов, поэтому inline стили не популярны.

**Объявление стиля в разделе head.**

Также существует HTML тег style. Его указываем сразу в заголовке, для него в атрибуте type указываем тип данных, в данном случае это text/css. Внутри этого тега мы уже прописываем стили, которые будут действовать для всей данной страницы. Так немного быстрее, чем предыдущим способом, но придется так делать на каждой странице.

**Подключение внешнего файла**

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

**style.css**

body {

color : blue;

background : #0f0;

}

h1 {

text - align : center;

}

А в нужном HTML файле его подключаем:

**index.html**

<head>

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

</head>

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

**Какой способ подключения стилей выбрать?**

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

* 1. **Селекторы в CSS**

**Селекторы тегов**

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

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

**Селекторы идентификаторов (id)**

|  |  |
| --- | --- |
| **html** | **css** |
| <p id = ”first” > Цвет фона данного параграфа будет серым </p> | #first {  background: #ccc;  } |

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

**Селекторы классов (class)**

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

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

**Селекторы атрибутов**

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

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

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

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

* 1. **Свойства стилей**

**Единицы измерения в CSS**

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

Относительные единицы измерения:

● % - процент;

● em – высота текущего шрифта.

Абсолютные единицы измерения

● cm - сантиметр;

● mm – миллиметр;

● in - дюйм;

● px - пиксел;

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

**Пиксели**

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

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

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

Плюсы этой единицы: четкость и понятность, минусы: другие единицы могут устанавливать соотношения между различными размерами

**Проценты**

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

Относительно шрифта: em

1em – текущий размер шрифта. Можно брать любые пропорции от текущего шрифта: 2em, 0.5em и т.п. Размеры в em – относительные, они определяются по текущему контексту.

* 1. **Способы задания цветов**

Цвета в CSS можно задавать 3 различными способами. Первый способ - задавать цвета, используя названия цветов на английском языке, например: red, green, blue, black, yellow, white и т.д. Но при таком подходе есть ограничения в выборе цвета, невозможно получить различные оттенки цветов.

Для того, чтобы можно было выбрать один из более, чем 16 млн. цветов, нужно использовать способ выбора цвета: либо как функциональный RGB, либо шестнадцатеричный RGB. RGB – это аббревиатура, и расшифровывается она как Red-Green-Blue, то есть Красный-Зеленый-Синий. Таким образом, любой цвет можно получить, смешав эти три цвета.

**Функциональный RGB**

RGB(255, 130, 0) 0 - 255

RGB(100%, 70%, 0%) 0 – 100%

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

**Шестнадцатеричный RGB**

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

#FFAA00 => #FA0

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

* 1. **Свойства стилей CSS**

**Ширина и высота: width и height**

p {

height : 200px;

width : 300px;

}

Можно задавать ширину и высоту в любых единицах измерения CSS. Если содержимое блока превышает указанную высоту, то высота элемента останется неизменной, а содержимое будет отображаться поверх него. Из-за этой особенности может получиться наложение содержимого элементов друг на друга, когда элементы в коде HTML идут последовательно. Чтобы этого не произошло, добавьте overflow: auto к стилю элемента.

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

**Фон элемента - background**

background - color : #ff0;

background - image : url ( img / photo . jpg );

background - position : top ; ( bottom | left | right)

background - repeat : repeat - x ; ( repeat - y | no - repeat)

background - attachment : fixed;

● background-color - задает цвет фона, который можно задавать любым из трех способов задания цветов.

● background-image используется для того, чтобы в качестве фона можно было установить изображение. Для этого необходимо в значении свойства указать путь к изображению в скобках url.

● background-position - указывает где будет располагаться фоновое изображение. Может иметь значения: top, bottom, left, right. background-repeat определяет, нужно ли повторять фоновое изображение. repeat-x - изображение повторяется по горизонтали, repeat-y - по вертикали, no-repeat - изображение не повторяется. По умолчанию у этого свойства установлено значение repeat, что означает, что изображение будет повторяться по горизонтали и по вертикали.

● background-attachment - определяет, будет ли изображение прокручиваться вместе с содержимым элемента. По умолчанию, оно установлено как scroll, что означает, что изображение будет прокручиваться, а при значении fixed, изображение будет оставаться неподвижным.

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

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

**border рамка – вокруг элемента**

border - color : red ; ( #f00 | RGB(255, 0, 0))

border - style : solid ; ( dotted | dashed | groove | ridge | solid | double | inset | outset)

border - width : 2px;

border тоже имеет различные свойства:

● border-color - цвет рамки.

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

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

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

border : 1px solid black;

Есть возможность каждую границу задавать отдельно, когда необходима, к примеру, только одна граница:

border - top : 2px dotted green ;

border - bottom : 3px double blue ;

border - left : 1px solid red ;

border - right : 4px inset #000;

**Цвет текста - color**

Цвет текста также можно задавать любым из 3 способов:

color: red;

color: #78fa2e;

color: RGB ( 34 , 21 , 56 );

**Шрифт - font**

font-family - устанавливает шрифт текста.

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

● serif — шрифты с засечками;

● sans-serif — рубленные шрифты, без засечек;

● cursive — курсивные шрифты;

● fantasy — декоративные шрифты;

● monospace — моноширинные шрифты.

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

font - style: italic; ( oblique | normal)

font - variant: small - caps;

font - weight: bold; ( bolder | lighter | 100 | 200 );

font - size: 20px; ( small | medium | large );

● font-style – стиль шрифта. По умолчанию установлен шрифт в значении normal, italic – это курсивное начертание, которое имитирует рукописный текст, а oblique – наклонное начертание, которое получается путем наклона знаков вправо.

● font-variant имеет только 2 значения, по умолчанию установлено значение normal и small-caps, которое у строчных букв имитирует заглавные буквы, только уменьшенного размера.

● font-weight задает насыщенность шрифта. Можно указывать значения предопределенными словами, например, bold - полужирный, bolder - жирный, lighter - светлый. Ещё есть возможность указывать насыщенность цифрами от 100 до 900.

● font-size определяет размер шрифта. Можно указывать в любых единицах измерения или предопределенными словами. Указывать стиль шрифта можно при помощи сокращенной записи. В данном случае важен порядок следования значений.

font: font - style

font - variant

font - weight

font - size

font - family;

font: bold 24px Arial, Verdana;

**Оформление списков - list-style**

list - style - type: circle; ( disc | square | armenian | decimal)

list - style - position: inside;

list - style - image: url ( img / list. png );

Свойство list-style определяет стиль маркера у списков.

● list-style-type - тип маркера, который может быть разных видов, в примере приведены только некоторые из них. Остальные виды маркеров можно найти в справочнике.

● list-style-position определяет то, где располагается маркер, по умолчанию у него значение outside. В этом случае маркеры будут располагаться за пределами текстового блока. При значении inside, наоборот, внутри текстовых блоков.

● list-style-image позволяет вместо маркера установить изображение, для этого нужно указать к нему путь в скобках url.

Для определения стиля маркеров также существует сокращенная запись.

list - style: list - style - type

list - style - position

list - style - image;

list - style: circle inside;

text - align: center; ( justify | left | right)

text - decoration: none; ( line - through | overline | underline | none)

text - transform: capitalize; ( lowercase | uppercase)

И еще некоторые полезные свойства.

● text-align - выравнивание содержимого блока по горизонтали. Принимает 4 значения: left, right, center и justify (выравнивание происходит по ширине, т.е. одновременно по левому и по правому краю).

● text-decoration применяется для следующего оформления текста: line-through – перечеркивает текст, overline - задает линию над текстом, underline - задает линию под текстом (подчеркивает текст), none (по умолчанию) - отменяет все эффекты.

● text-transform используется для изменения регистра символов. При значении capitalize каждое слово в предложении будет начинаться с заглавной буквы, при значении lowercase все символы будут строчными, а при значении uppercase все символы будут заглавными.

* 1. **Вложенность**

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

**Контекстные селекторы**

|  |  |
| --- | --- |
| **html** | **css** |
| <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 задаем определенный стиль. Обратимся к этой ссылке при помощи контекстного селектора. Для этого в качестве селектора сначала указывается тег параграфа с классом main, затем ставится пробел и следующим указывается тег strong, после этого обращаемся к тегу нужной нам ссылки. Таким образом, заданный стиль будет применяться ТОЛЬКО к первой ссылке в параграфе с классом main.

Если убрать тег strong, ВСЕ ссылки в параграфе main станут красного цвета с размером шрифта в 18 px.

**Дочерние селекторы**

|  |  |
| --- | --- |
| **html** | **css** |
| <p class= ”main” >  <a href=“#”> ссылка 1 </a>  </p>  <p class= ”main” >  <i><a href=“#”> ссылка 2 </a></i>  </p> | .main > a {  font-size: 18px ;  color: red;  } |

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

**Соседние селекторы**

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

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

**Наследование**

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

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

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

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

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

**Группировка свойств**

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

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

|  |  |
| --- | --- |
| **Без группировки свойств** | **С группировкой** |
| 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;  } |

* 1. **Приоритеты стилей в css**

Вы можете столкнуться с ситуацией, когда при разработке сайтов, вы задаёте определенное свойство какому-нибудь элементу, а это свойство не работает, т.е. элемент не приобретает заданный стиль. Это происходит потому, что где-то уже был установлен определенный стиль этому элементу. Чтобы решить эту проблему и задать нужный стиль, нужно знать приоритеты применения стилей.

Существует такое понятие, как каскадирование, которое применяется тогда, когда одному и тому же элементу пытаются присвоить одни и те же стили. Например, мы всем параграфам пытаемся присвоить сначала черный цвет, а потом зеленый. И какое правило должно тогда примениться?

h1 {

color : black;

}

h1 {

color : green;

}

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

**Приоритеты источников стилей**

1. Стиль автора документа обладает самым высоким приоритетом. Этот стиль задает сам разработчик сайта.

2. Стиль, заданный пользователем в настройках браузера. Стиль CSS может задать конечный пользователь этого сайта, если подключит свой собственный файл стилей. Этот источник является менее приоритетным.

3. Стиль самого браузера, т.е. тот стиль, который определен в настройках самого браузера. Это источник обладает самым низким приоритетом.

**Приоритеты стилей автора**

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

h1 {

color : black ! important;

}

h1 {

color : green;

}

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

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

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

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

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

● Селекторы тегов и псевдоэлементы — по 1 баллу (0, 0, 0, 1);

● Селекторы атрибутов, классы и псевдоклассы — по 10 баллов (0, 0, 1, 0);

● Идентификаторы — по 100 баллов (0, 1, 0, 0);

● Атрибут style – 1000 баллов (1, 0, 0, 0).

Пример начисления баллов за специфичность:

● p { } - 1 балл (селектор тегов );

● p:first-letter { } - 2 балла (1 - селектор тегов и 1 - псевдоэлемент);

● input[type=“submit"] { } - 11 баллов (по 1 селектору тегов и атрибутов);

● div.head .new { } - 21 балл (2 класса и 1 селектор тегов);

● #header a:hover { } - 111 баллов (идентификатор, селектор тегов и псевдокласс).

Следующие по приоритетности стили указанны в порядке убывания:

● Стили, заданные в разделе <head>;

● Стили, подключаемые из внешних файлов;

● Наследуемые стили от предков.

# **Псевдоклассы, табличная вёрстка**

*Приоритеты применения стилей. Псевдоклассы и псевдоэлементы. Создание таблиц. Объединение ячеек. Вложенные таблицы. Стилевое оформление таблиц.*

* 1. **Псевдоклассы и псевдоэлементы**

**Псевдоклассы**

Псевдоклассы позволяют добавлять особые классы к элементам, выбирая объекты, которых нет в структуре веб-страницы, либо которые нельзя выбрать с помощью обычных селекторов, например, таким как, первая буква или первая строка одного абзаца.

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

Рассмотрим синтаксис псевдоклассов:

Селектор:псевдокласс {

свойство 1: значение 1;

}

a: hover {

color: #ccc;

}

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

**Псевдоклассы, определяющие состояние элементов**

a:link — ссылается на непосещенную ссылку.

a:visited — ссылается на уже посещенную ссылку.

a:hover — ссылается на любой элемент, по которому проводят курсором мыши.

a:focus — ссылается на любой элемент, над которым находится курсор мыши.

a:active — ссылается на элемент, который был активизирован пользователем.

:valid — выберет поля формы, содержимое которых прошло проверку в браузере на соответствие указанному типу.

:invalid — выберет поля формы, содержимое которых не соответствует указанному типу.

:enabled — выберет все доступные (активные) поля форм.

:disabled — выберет заблокированные поля форм, т.е. находящиеся в неактивном состоянии.

:in-range — выберет поля формы, значения которых находятся в заданном диапазоне.

:out-of-range — выберет поля формы, значения которых не входят в установленный диапазон.

:lang() — выбирает абзацы на указанном языке. :not(селектор) — выберет элементы, которые не содержат указанный селектор, например, класс, идентификатор или селектор элемента

:not([type="submit"]). :target — выбирает элемент с символом #, на который ссылаются в документе.

:checked — выбирает выделенные (выбранные пользователем) элементы.

**Псевдоклассы, структурные**

:nth-child(odd) — выбирает нечетные дочерние элементы.

:nth-child(even) — выбирает четные дочерние элементы.

:nth-child(3n) — выбирает каждый третий элемент среди дочерних.

:nth-child(3n+2) — выбирает каждый третий элемент, начиная со второго дочернего элемента (+2).

:nth-child(n+2) — выбирает все элементы, начиная со второго.

:nth-child(3) — выбирает третий дочерний элемент.

:nth-last-child() — в списке дочерних элементов выбирает элемент с указанным местоположением, аналогично с :nth-child(), но начиная с последнего, в обратную сторону.

:first-child — позволяет оформить только самый первый дочерний элемент тега.

:last-child — позволяет форматировать последний дочерний элемент тега.

:only-child — выбирает элемент, являющийся единственным дочерним элементом.

:empty — выбирает элементы, у которых нет дочерних элементов.

:root — выбирает элемент, являющийся корневым в документе (элемент html).

**Псевдоклассы по типу дочернего элемента**

:nth-of-type() — выбирает элементы по аналогии с

:nth-child(), при этом берет во внимание только тип элемента.

:first-of-type — позволяет выбрать первый дочерний элемент.

:last-of-type — выбирает последний тег конкретного типа.

:nth-last-of-type() — выбирает элемент заданного типа в списке элементов в соответствии с указанным местоположением, начиная с конца.

:only-of-type — выбирает единственный элемент указанного типа среди дочерних элементов родительского элемента.

**Пример использования псевдоклассов**

|  |  |
| --- | --- |
| **html** | **css** |
| <ul>  <li> Первый элемент </li>  <li> Второй элемент </li>  <li> Третий элемент </li>  </ul> | li: first-child {  font-size: 24px;  color: #F23401;  } |

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

**Комбинирование псевдоклассов**

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

/\* При наведении на не посещенную ссылку цвет текста будет зеленым \*/

a: link: hover {

color: #0F0;

}

/\* При наведении на посещенную ссылку цвет текста будет красным\*/

a: visited: hover {

color: #F00;

}

**Псевдоэлементы**

Псевдоэлементы - практически то же самое, что и псевдоклассы, только они позволяют ввести несуществующие элементы в веб-документ, и придать им определенные стили. В CSS существует 4 псевдоэлемента:

:first-letter — выбирает первую букву каждого абзаца, применяется только к блочным элементам.

:first-line — выбирает первую строку текста элемента, применяется только к блочным элементам.

:before — вставляет генерируемое содержимое перед элементом.

:after — добавляет генерируемое содержимое после элемента.

**Пример использования псевдоэлементов**

|  |  |
| --- | --- |
| **html** | **css** |
| <ul>  <li> Первый элемент </li>  <li> Второй элемент </li>  <li> Третий элемент </li>  </ul> | li: after {  content: “new”;  color: #F00;  } |

После элемента списка li появится текст new красного цвета.

* 1. **Для чего нужны таблицы в HTML**

Основное назначение таблиц в HTML - это представление табличных данных, таких как: просмотр информации о пользователях, просмотр заказанных товаров в интернет-магазине, просмотр отчётов о продажах и многое другое. *Второе назначение таблиц в html таково, что при помощи их можно верстать веб-страницы. А верстать - это значит разбивать нашу страницу на элементы, т.е. формировать различные блоки нашего сайта, такие как: шапка, меню, контент, подвал и другие элементы. Табличный способ верстки в настоящее время считается неправильным и устаревшим, потому что таблицы используются не по назначению, но многие сайты либо уже сверстаны при помощи таблиц, либо продолжают верстаться этим способом. Поэтому разработчику сайтов важно уметь пользоваться как табличным способом верстки, так и версткой слоями. И оба способа верстки мы рассмотрим в данном курсе.*

**Структура таблицы в html**

<table>

<tr>

<td> Столбец 1 </td>

<td> Столбец 2 </td>

<td> Столбец 3 </td>

</tr>

</table>

Любая таблица в html помещается в контейнер <table>, после чего в контейнер строки <tr> помещаются уже столбцы таблицы, которые помещаются в тег <td>. В данном примере таблица будет состоять из 3-х столбцов и одной строки, т.е. в таблице будет 3 ячейки. Для того, чтобы добавить еще одну строку в данную таблицу, нужно вложить в контейнер <table> еще один контейнер строки <tr>, и в него поместить то же количество столбцов <td>, т.е. 3 столбца.

<table>

<tr>

<td> Столбец 1 </td>

<td> Столбец 2 </td>

<td> Столбец 3 </td>

</tr>

<tr>

<td> Столбец 1 </td>

<td> Столбец 2 </td>

<td> Столбец 3 </td>

</tr>

</table>

**Создание ячеек таблицы**

Элемент <td> создает ячейки таблицы с данными, добавляя их в строку таблицы. Парные теги <td>…</td>, расположенные между тегами соответствующей строки, определяют количество ячеек в пределах одной строки. Количество пар ячеек таблицы должно быть равным количеству пар ячеек заголовка.

Элемент <th> создает заголовок — специальную ячейку, текст в которой выделяется полужирным шрифтом. Количество ячеек заголовка определяется количеством пар тегов <th>…</th>.

**Теги группирования строк html таблицы**

Для создания более сложных таблиц можно использовать теги:

● <caption>...</caption> - обрамление заголовка таблицы (текст, размещенный между этими тегами, выравнивается по центру). В исходном коде тег <caption> ставится сразу после тега <table>;

● <thead>...</thead> - обрамление шапки таблицы (в одной таблице допускается не более одного тега <thead>);

● <tbody>...</tbody> - группирует строки таблицы в один табличный блок (таких блоков в таблице может быть несколько);

● <tfoot>...</tfoot> - обрамление футера, т.е. нижней, завершающей части таблицы (в одной таблице допускается не более одного тега <tfoot>). В исходном коде тег <tfoot> ставится до тега <tbody>.

**Группировка строк и столбцов таблицы**

Элемент <colgroup> создает структурную группу столбцов, выделяя логически однородные ячейки. Группирует один или более столбцов для форматирования, позволяя применить стили к столбцам, вместо того, чтобы повторять стили для каждой ячейки и для каждой строки. Добавляется непосредственно после тегов <table> и <caption>. Элемент <col> формирует неструктурные группы столбцов, которые делят таблицу на разделы, не относящиеся к общей структуре, т.е. не содержащие информацию одного типа. Позволяет задавать свойства столбцов для каждого столбца в пределах элемента <colgroup>. С помощью атрибута <style> можно изменить основной цвет фона ячеек. Для элемента <col> доступен атрибут span, задающий количество столбцов для объединения.

**Объединение ячеек таблицы**

Для того, чтобы в полной мере начать использовать таблицы в html, необходимо научиться объединять ячейки. Для этого у тега <td> или <th> существуют атрибуты colspan и rowspan.

● colspan - объединяет ячейки по горизонтали

● rowspan - объединяет ячейки по вертикали

|  |  |  |
| --- | --- | --- |
| rowspan –  объединение по  вертикали (строк) | colspan – объединение по горизонтали  (столбцов) | |
|  |  |
|  |  |  |

В данном примере, при помощи атрибута rowspan в первом столбце объединяются 2 строки. При помощи атрибута colspan в первой строке объединяются уже два столбца.

Посмотрим, как это выглядит в html.

<table>

<tr>

<td rowspan = "2" > Столбец 1 </td>

<td colspan = "2" > Столбец 2 </td>

</tr>

<tr>

<td> Столбец 5 </td>

<td> Столбец 6 </td>

</tr>

<tr>

<td> Столбец 7 </td>

<td> Столбец 8 </td>

<td> Столбец 9 </td>

</tr>

</table>

В первой строке данной таблицы будет всего два столбца, в первом столбце первая строка объединяется со второй строкой, поэтому у первой ячейки указываем атрибут rowspan со значением 2, что означает, что нужно объединить две ячейки. В этой же строке, во втором и третьем столбце объединяем две ячейки по горизонтали при помощи атрибута colspan, также со значением 2. Далее во второй строке нужно учесть, что не надо указывать для неё первый столбец, т.к. он уже объединен с первой строкой. Поэтому во второй строке указываются два столбца, начиная со второго. В третьей строке присутствуют все три столбца по порядку.

**Вложенные таблицы**

При помощи объединения ячеек можно сделать каркас будущего сайта, т.е. макет сайта будет состоять из одной большой таблицы. А уже в эту таблицу можно вкладывать другие таблицы, например, вложить галерею изображений, которую тоже можно сделать в виде таблицы, т.е. каждое изображение будет находиться в отдельной ячейке. Или использовать таблицу по ее прямому назначению, т.е. для представления табличных данных. Для этого можно вложить в определенную ячейку еще одну таблицу, соблюдая ту же структуру.

<table>

<tr>

<td>

<table>

<tr>

<td> Столбец 1 </td>

<td> Столбец 2 </td>

<td> Столбец 3 </td>

</tr>

</table>

</td>

<td></td>

<td></td>

</tr>

</table>

* 1. **Стилевое оформление таблиц**

По умолчанию таблица и ячейки таблицы не имеют видимых границ и фона, при этом ячейки внутри таблицы не прилегают вплотную друг к другу. Ширина ячеек таблицы определяется шириной их содержимого, поэтому ширина столбцов таблицы может быть разной. Высота всех ячеек сетки строки по умолчанию одинаковая.

Границу таблице можно задавать аналогично любому другому элементу, это мы рассматривали на прошлом занятии. Единственное, если задать свойство border только тегу <table>, то граница будет задана только внешним границам таблицы. Это происходит потому, что свойство border не наследуется. Чтобы задать границы для ячеек, нужно задать их тегу <td>. Здесь уже можно задавать границы ячейкам любого стиля.

/\*Внешняя граница таблицы\*/

table {

border : 1px solid #000;

}

/\*Границы для ячеек таблицы\*/

td {

border : 1px solid #000;

}

Ширину и высоту таблицы задаем при помощи свойств CSS width и height.

table {

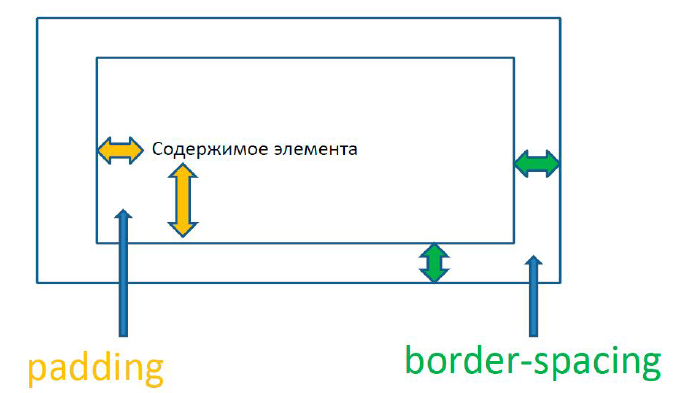
width : 400px;

height : 200px;

}

По умолчанию ширина и высота таблицы определяется содержимым её ячеек. Если не задать ширину, то она будет равна ширине самого широкого ряда Ширину таблицы и её столбцов можно задать с помощью свойства width. Если для таблицы задана ширина table {width: 100%;}, то она будет равна ширине блока-контейнера, а ширина столбцов установится в соответствии с шириной содержимого ячеек. Чаще всего ширину таблицы и столбцов задают в px или %.

Высота таблицы обычно не задается. Высотой строк таблицы можно управлять, добавив верхний и нижний padding, или же задав высоту с помощью свойства height. Для задания отступов элементу, в CSS существуют свойства padding и border-spacing.



● border-spacing – внешние отступы

● padding – внутренние отступы

Внешние отступы (border-spacing) можно задать либо всем 4 сторонам, либо отдельно. В этом случае первое значение будет действовать для левой и правой сторон, а второе - для верхней и нижней.

Внутренние отступы (padding) можно задавать и для каждой границы 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

Для выравнивания по горизонтали внутри ячейки существует устаревший атрибут align, который заменяется на свойство CSS text-align.

td {

text - align : center ; ( right | justify | left)

}

Для вертикального выравнивания внутри ячейки у тега <td> существует устаревший атрибут valign. Для его замены в CSS существует свойство vertical-align.

td {

vertical - align : top ; ( bottom | baseline | middle)

}

Это свойство в отношении ячеек таблицы может принимать четыре значения:

● top - выравнивание по верхнему краю;

● bottom - выравнивание по нижнему краю;

● baseline - базовая линия ячейки выравнивается по базовой линии первой текстовой строки;

● middle – выравнивание по середине (по умолчанию).

Есть ещё одно полезное свойство для оформления таблиц. Если для всех ячеек задать атрибут cellspacing=”0”, или CSS свойство border-spacing=”0”, то границы соседних ячеек будут двойными. Для того, чтобы убрать двойные границы, в CSS существует свойство border-collapse:

table {

border - collapse : collapse;

}

Свойство border-collapse может принимать два значения. По умолчанию установлено separate, при котором рамка двойная, а при значении collapse рамка становится одинарной.

* 1. **Основные теги для верстки**

При вёрстке сайта с помощью слоёв самым часто используемым html-тегом является <div>, который как раз и формирует слой на веб-странице. Он является блочным тегом. Второй тег, который используется при верстке, это строчный тег <span>. Сами по себе эти теги ничего на экране не отображают, и оформляются они стилями css.

<div> Блочный элемент </div>

<span> Строчный элемент </span>

**Блочный элемент** создает разрыв строки перед тегом и после него. Он образуют прямоугольную область, по ширине занимающую всю ширину веб-страницы или блока-родителя, если для него не задано значение width.

Блочные элементы могут содержать внутри себя элементы любого типа. Нельзя размещать блочные элементы внутри строчных, за исключением элемента <img>. Для блочных элементов можно задавать margin и padding. Свойства width и height устанавливают ширину и высоту области содержимого элемента. Фактическая ширина элемента складывается из ширины полей (внутренних отступов), границ и внешних отступов.

Строчные элементы не создают блоки, они отображаются на одной строке с содержимым рядом стоящих тегов. Строчные элементы являются потомками блочных элементов. Они игнорируют верхние и нижние margin и padding, но, если для элемента задан фон, он будет распространяться на верхний и нижний padding, заходя на соседние строки текста.

Ширина и высота строчного элемента зависит только от его содержания, задать его размеры с помощью CSS нельзя. Можно увеличить расстояние между соседними элементами по горизонтали с помощью горизонтальных полей и отступов.

Давайте вспомним, какие основные элементы html относятся к блочным, а какие к строчным.

**Блочные элементы**

● <h1>...</h1>

● <h2>...</h2>

● <h3>...</h3>

● <h4>...</h4>

● <h5>...</h5>

● <h6>...</h6>

● <li>…</li>

● <legend>…</legend>

● <nav>…</nav>

● <ol>…</ol>,

● <article>…</article>

● <aside>…</aside>

● <body>…</body>

● <blockquote>…</blockquote>

● <div>…</div>

● <fieldset>…</fieldset>

● <footer>…</footer>

● <form>...</form>

● <header>…</header>

● <output>…</output>

● <p>…</p>

● <pre>…</pre>

● <section>…</section>

● <ul>…</ul>

**Строчные элементы**

● <a href=”#”>...</a>

● <i>...</i>

● <b>...</b>

● <em>...</em>

● <strong>...</strong>

● <font>...</font>

● <img>

● <input>

В этом списке тег картинки <img> является замещаемым строчным элементом, т.е. при помощи замещаемых элементов указывается, что в данном месте должен быть какой-то сторонний объект, в данном случае - картинка. Замещаемому элементу можно задать и ширину и высоту, но все равно замещаемый элемент будет являться строчным.