

Лекция 6. Основы CSS

При первоначальном знакомстве с лекцией пункты 6.13 – 6.20 могут быть пропущены.

Оглавление

Лекция 6. Основы CSS	1
6.1. Основные понятия	2
6.2. Атрибут style.....	2
6.3. Элемент Style.....	3
6.4. Файл CSS.....	3
6.5. Совмещение способов определения стиля и разрешение конфликтов	4
6.6. Устаревшие (deprecated) атрибуты HTML и стили CSS.....	5
6.7. Валидация кода CSS.....	5
6.8. Селекторы.....	5
6.9. Классы.....	5
6.10. Идентификаторы	6
6.11. Универсальный селектор	7
6.12. Стилизация группы селекторов	7
6.13. Селекторы потомков.....	7
6.14. Селекторы дочерних элементов	9
6.15. Селекторы элементов одного уровня	9
6.16. Псевдоклассы.....	11
6.17. Псевдоклассы дочерних элементов	13
6.18. Псевдоклассы форм	18
6.19. Псевдоэлементы	22
6.20. Селекторы атрибутов	24
6.21. Наследование стилей.....	26
6.22. Каскадность стилей	28
6.23. Единицы измерения в CSS.....	31
Задание к лабораторной работе 6:.....	33

6.1. Основные понятия

Каскадные таблицы стилей CSS (Cascading Style Sheets) определяют представление документа, его внешний вид.

Стиль в CSS представляет правило, которое указывает браузеру, как надо форматировать элемент. Форматирование может включать установку цветов фона элемента, настройку шрифта, обрамления и т. д.

Определение стиля состоит из двух частей: **селектор**, который указывает на элемент, и **блок объявления** стиля - набор команд, которые устанавливают правила форматирования. Например:

```
div {  
    background-color:red;  
    width: 100px;  
    height: 60px;  
}
```

В данном случае селектором является `div`. Этот селектор указывает, что этот стиль будет применяться ко всем элементам `div`.

После селектора в фигурных скобках идет блок объявления стиля. Между открывающей и закрывающей фигурными скобками определяются команды, указывающие, как форматировать элемент.

Каждая команда состоит из **свойства** и **значения**. Так, в следующем выражении:

```
background-color: white;
```

`background-color` представляет собой свойство, а `red` - значение.

Свойство определяет конкретный стиль. Свойств CSS существует множество.

Так, свойство `background-color` определяет цвет фона. После двоеточия следует значение для этого свойства: `white`. То есть, для фона элемента устанавливается цвет "white" (белый).

После каждой команды ставится точка с запятой, которая отделяет данную команду от других.

Наборы таких стилей часто называют **таблицами стилей** или CSS (Cascading Style Sheets или каскадные таблицы стилей). Существует 3 основных способа определения стилей.

6.2. Атрибутом style

Первый способ заключается во встраивании стилей непосредственно в элемент с помощью атрибута `style`:

```
<h2 style="color:blue;">Стили</h2>
```

```
<div
```

```
    style="width: 100px; height: 100px; background-color: red;">
```

```
</div>
```

Здесь определены два элемента - заголовок h2 и блок div. У заголовка определен синий цвет текста с помощью свойства color. У блока div определены свойства ширины (width), высоты (height), а также цвета фона (background-color).

6.3. Элемент Style

Второй способ состоит в использования элемента style в документе HTML. Этот элемент сообщает браузеру, что данные внутри являются кодом CSS, а не разметкой HTML:

```
<head>
  <style>
    h2{
      color:blue;
    }
    div{
      width: 100px;
      height: 100px;
      background-color: red;
    }
  </style>
</head>
```

Результат в данном случае будет тем же, что и в предыдущем случае, но стилевые указания будут действовать на *все* теги h2 и div в документе, а не только на те, к которым применён стиль в п. 3.2.

Часто элемент style определяется внутри элемента head, однако может также использоваться в других частях HTML-документа. Элемент style содержит наборы стилей. У каждого стиля указывается вначале селектор, после чего в фигурных скобках идет все те же определения свойств CSS и их значения, что были использованы в предыдущем примере.

Второй способ делает код HTML чище за счет вынесения стилей в элемент style. Но также есть и третий способ, который заключается в вынесении стилей во внешний файл.

6.4. Файл CSS

Создадим в одной папке с HTML странице текстовый файл, который переименуем в styles.css и определим в нем следующее содержимое:

```
h2 {
  color:blue;
}
div {
  width: 100px;
  height: 100px;
  background-color: red;
}
```

Это те же стили, что были внутри элемента `style`. Также изменим код web-страницы:

```
<!DOCTYPE html>
<html lang="ru">
  <head>
    <meta charset="utf-8">
    <title>Стили</title>
    <link rel="stylesheet"
      type="text/css" href="styles.css"/>
  </head>
  <body>
    <h2>Стили</h2>
    <div></div>
  </body>
</html>
```

Здесь уже нет элемента `style`, зато есть элемент `link`, который подключает выше созданный файл `styles.css` из текущей папки.

При таком определении стили легче модифицировать, чем встроенные, способ является **предпочтительным** в HTML5.

Использование стилей во внешних файлах позволяет уменьшить нагрузку на web-сервер с помощью механизма кэширования. К тому же, можно применить один стиль к любому количеству документов HTML.

6.5. Совмещение способов определения стиля и разрешение конфликтов

Все три способа могут сочетаться в одном документе

```
<!DOCTYPE html>
<html lang="ru">
  <head>
    <link rel="stylesheet"
      type="text/css" href="styles.css"/>
    <style>
      div{
        width:200px;
      }
    </style>
  </head>
  <body>
    <div style="width:120px;"></div>
  </body>
</html>
```

Различные стилевые правила, указанные для элемента, могут **конфликтовать** между собой, в этом случае применение конкретного стиля решают **правила приоритета**. Действует следующая система приоритетов:

- Если у элемента определены встроенные стили (inline-стили), то они имеют высший приоритет, то есть в примере выше итоговой шириной будет 120 пикселей;
- Далее в порядке приоритета идут стили, которые определены в элементе `style`;
- Наименее приоритетными стилями являются те, которые определены во внешнем файле.

6.6. Устаревшие (*deprecated*) атрибуты HTML и стили CSS

Многие элементы, совместимые с HTML4, позволяют устанавливать стили отображения с помощью атрибутов. Например, у ряда элементов мы можем применять атрибуты `width` и `height` для установки ширины и высоты элемента соответственно. Однако подобного подхода следует избегать и вместо встроенных атрибутов следует применять стили CSS. Важно чётко понимать, что разметка **HTML** должна предоставлять только **структуру** HTML-документа, а весь его **внешний вид**, стилизацию должны определять стили **CSS**.

6.7. Валидация кода CSS

В процессе написания стилей CSS могут возникать вопросы, а правильно ли так определять стили, корректны ли они. В этом случае мы можем воспользоваться валидатором CSS, который доступен по адресу <https://jigsaw.w3.org/css-validator/>

6.8. Селекторы

Селекторы по тегам содержат имя тега без символов `<` и `>` и применяются ко всем подходящим тегам.

```
div{
    width:50px; /* ширина */
    height:50px; /* высота */
    background-color:red; /* цвет фона */
    margin: 10px; /* отступ от других элементов */
}
```

6.9. Классы

Иногда для одних и тех же элементов требуется различная стилизация. И в этом случае мы можем использовать **классы**.

Для определения селектора класса в CSS перед названием класса ставится **точка**:

```
.redBlock{
    background-color: red;
}
```

Название класса может быть произвольным. Например, в данном случае название класса - "redBlock". Однако при этом в имени класса разрешается использовать

буквы, числа, дефисы и знаки подчеркивания, причём, начинаться название класса должно с буквы.

Также нужно учитывать регистр имен: названия "article" и "ARTICLE" будут представлять разные классы.

После определения класса мы можем его применить к элементу с помощью атрибута class. Например:

```
<style>
    div{
        width: 50px;
        height: 50px;
        margin: 10px;
    }
    .redBlock{
        background-color: red;
    }
    .blueBlock{
        background-color: blue;
    }
</style>
<div class="redBlock"></div>
<div class="redBlock"></div>
<div class="blueBlock"></div>
<div class="redBlock"></div>
```

6.10. Идентификаторы

Для идентификации уникальных на web-странице элементов используются **идентификаторы**, которые определяются с помощью атрибутов id. Например, на странице может быть головной блок или шапка:

```
<div id="header"></div>
```

Определение стилей для идентификаторов аналогично определению классов, только вместо точки ставится символ **решётки #**:

```
<style>
    div{
        margin: 10px;
        border: 1px solid #222;
    }
    #header{
        height: 80px;
        background-color: #ccc;
    }
    #content{
        height: 180px;
        background-color: #eee;
    }
    #footer{
        height: 80px;
```

```

        background-color: #ccc;
    }
</style>
<div id="header">Шапка сайта</div>
<div id="content">Основное содержимое</div>
<div id="footer">Футер</div>

```

Идентификаторы в большей степени относятся к структуре web-страницы и в меньшей степени к стилизации. Для стилизации преимущественно используются классы, нежели идентификаторы.

6.11. Универсальный селектор

Кроме селекторов тегов, классов и идентификаторов в CSS также есть так называемый универсальный селектор, который представляет знак звездочки (*). Он применяет стили ко всем элементам на HTML-странице:

```

* {
    background-color: red;
}

```

Зачастую применение универсального селектора можно заменить установкой стилей для тега html или body.

6.12. Стилизация группы селекторов

Иногда определенные стили применяются к целому ряду селекторов. Например, мы хотим применить ко всем заголовкам подчеркивание. В этом случае мы можем перечислить селекторы всех элементов через запятую:

```

<style>
h1, h2, h3, h4 {
    color: red;
}
</style>
<h1>CSS3</h1>
<h2>Селекторы</h2>
<h3>Группа селекторов</h3>
<p>Некоторый текст...</p>

```

Группа селекторов может содержать как селекторы тегов, так и селекторы классов и идентификаторов, например:

```

h1, #header, .redBlock {
    color: red;
}

```

6.13. Селекторы потомков

Web-страница может иметь сложную организацию, одни элементы внутри себя могут определять другие элементы. Вложенные элементы иначе можно назвать потомками, а контейнер, содержащий эти элементы - родителем.

Например, пусть элемент body на web-странице имеет следующее содержимое:

```

<body>
    <h2>Заголовок</h2>

```

```
<div>
  <p>Текст</p>
</div>
</body>
```

Внутри элемента `body` определено три вложенных элемента: `h2`, `div`, `p`. Все эти элементы являются потомками элемента `body`.

А внутри элемента `div` определен только один вложенный элемент - `p`, поэтому элемент `div` имеет только одного потомка.

Используя специальные селекторы, мы можем стилизовать вложенные элементы или потомков внутри строго определенных элементов. Например, у нас на странице могут быть параграфы внутри блока с основным содержимым и внутри блока футера. Но для параграфов внутри блока основного содержимого мы захотим установить один шрифт, а для параграфов футера другой.

```
<style>
  #main p {
    font-size: 16px;
  }
  #footer p {
    font-size: 13px;
  }
</style>
<div id="main">
  <p>Первый абзац</p>
  <p>Второй абзац</p>
</div>
<div id="footer">
  <p>Текст футера</p>
</div>
```

Для применения стиля к вложенному элементу селектор должен содержать вначале родительский элемент и затем вложенный:

```
#main p{
  font-size: 16px;
}
```

Данный стиль будет применяться только к тем элементам `p`, которые находятся внутри элемента с идентификатором `main`.

```
<style>
li .redLink{
  color: red;
}
</style>
<ul>
<li>Просто элемент</li>
<li><span class="redlink">Красный элемент</span></li>
</ul>
```


Здесь стиль применяется к элементам с классом "redLink", которые находятся внутри элемента ``.

6.14. Селекторы дочерних элементов

Селекторы дочерних элементов отличаются от селекторов потомков тем, что позволяют выбрать элементы только первого уровня вложенности. Например:

```
<body>
  <h2>Заголовок</h2>
  <div>
    <p>Текст</p>
  </div>
</body>
```

Хотя вложенными в элемент `body` элементами являются три элемента - `h2`, `div`, `p`, но дочерними для `body` являются только два - `div` и `h2`, так как они находятся в первом уровне вложенности. А элемент `p` находится на втором уровне вложенности, так как вложен внутрь элемента `div`, а не просто элемента `body`.

Для обращения к дочерним элементам используется знак `>`:

```
<!DOCTYPE html>
<html lang="ru">
  <head>
    <meta charset="utf-8">
    <title>Селекторы атрибутов в CSS3</title>
    <style>
      .article > p{
        color: red;
      }
    </style>
  </head>
  <body>
    <div class="article">
      <p>Аннотация к статье</p>
      <div class="content">
        <p>Текст статьи</p>
      </div>
    </div>
  </body>
</html>
```

В блоке с классом `article` есть два параграфа. Селектор `.article > p` выбирает только те параграфы, который находятся непосредственно в блоке `article`.

6.15. Селекторы элементов одного уровня

Селекторы элементов одного уровня или смежных элементов позволяют выбрать элементы, которые находятся на одном уровне вложенности. Иногда такие элементы еще называют **сiblingи** (siblings) или сестринскими элементами. Например:

```
<body>
```

```
<h2>Заголовок</h2>
<div>
  <p>Текст первого блока</p>
</div>
<div>
  <p>Текст второго блока</p>
</div>
</body>
```

Здесь элементы h2 и оба блока div являются смежными, так как находятся на одном уровне. А элементы параграфов и заголовок h2 не являются смежными, так как параграфы вложены в блоки div.

Чтобы стилизовать **первый смежный элемент** по отношению к определенному элементу, используется знак плюса "+":

```
<!DOCTYPE html>
<html lang="ru">
  <head>
    <meta charset="utf-8">
    <title>Селекторы в CSS3</title>
    <style>
      h2+div { color: red; }
    </style>
  </head>
  <body>
    <h2>Заголовок</h2>
    <div>
      <p>Текст первого блока</p>
</div>
    <div>
      <p>Текст второго блока</p>
    </div>
  </body>
</html>
```

Селектор h2+div позволяет определить стиль (в данном случае красный цвет текста) для блока div, который идет непосредственно после заголовка h2.

Итак, этот селектор будет стилизовать блок div, если он будет идти непосредственно после заголовка. Если же между заголовком и блоком div будет находиться еще какой-либо элемент, то к нему не будет применяться стиль, например:

```
<h2>Заголовок</h2>
<p>Элемент между заголовком и блоком div</p>
<div>
  <p>Текст первого блока</p>
</div>
```

Если нам надо стилизовать вообще **все смежные элементы одного уровня**, независимо непосредственно идут они после определенного элемента или нет, то в этом случае вместо знака плюса необходимо использовать знак тильды "~":

```
<!DOCTYPE html>
```

```

<html lang="ru">
  <head>
    <meta charset="utf-8">
    <title>Селекторы в CSS3</title>
    <style>
      h2~div { color: red; }
    </style>
  </head>
  <body>
    <h2>Заголовок</h2>
    <p>Аннотация</p>
    <div>
      <p>Текст первого блока</p>
    </div>
    <div>
      <p>Текст второго блока</p>
    </div>
  </body>
</html>

```

6.16. Псевдоклассы

В дополнение к селекторам тегов, классов и идентификаторов нам доступны селекторы **псевдоклассов**, которые несут дополнительные возможности по выбору нужных элементов.

Список доступных псевдоклассов:

- **:root:** позволяет выбрать корневой элемент web-страницы, обычно корневым элементом является элемент `<html>`;
- **:link:** применяется к ссылкам и представляет ссылку в обычном состоянии, по которой еще не совершен переход;
- **:visited:** применяется к ссылкам и представляет ссылку, по которой пользователь уже переходил;
- **:active:** применяется к ссылкам и представляет ссылку в тот момент, когда пользователь осуществляет по ней переход;
- **:hover:** представляет элемент, на который пользователь навел указатель мыши. Применяется преимущественно к ссылкам, однако может также применяться и к другим элементам, например, к параграфам;
- **:focus:** представляет элемент, который получает фокус, то есть когда пользователь нажимает клавишу табуляции или нажимает кнопкой мыши на поле ввода (например, текстовое поле) ;
- **:not:** позволяет исключить элементы из списка элементов, к которым применяется стиль;
- **:lang:** стилизует элементы на основании значения атрибута `lang`;
- **:empty:** выбирает элементы, которые не имеют вложенных элементов, то есть являются пустыми.

При применении псевдоклассов перед ними всегда ставится двоеточие. Например, стилизуем ссылки, используя псевдоклассы:

```
<!DOCTYPE html>
<html lang="ru">
  <head>
    <meta charset="utf-8">
    <title>Псевдоклассы в CSS3</title>
  <style>
    a:link      {color:blue; text-decoration:none}
    a:visited   {color:pink; text-decoration:none}
    a:hover     {color:red; text-decoration:underline}
    a:active    {color:yellow; text-decoration:underline}
    input:hover {border:2px solid red;}
  </style>
</head>
  <body>
    <a href="index.html">Учебник по CSS3</a>
    <input type="text" />
  </body>
</html>
```

Селектор `:not()` позволяет выбрать все элементы кроме определенных, то есть исключить некоторые элементы из выбора.

```
<!DOCTYPE html>
<html lang="ru">
  <head>
    <meta charset="utf-8">
    <title>Селекторы в CSS3</title>
    <style>
      a:not(.blueLink) { color: red; }
    </style>
  </head>
  <body>
    <a>Первая ссылка</a><br/>
    <a class="blueLink">Вторая ссылка</a><br/>
    <a>Третья ссылка</a>
  </body>
</html>
```

Селектор `a:not(.blueLink)` применяет стиль ко всем ссылкам за исключением тех, которые имеют класс `"blueLink"`. В скобки псевдоклассу `not` передается селектор элементов, которые надо исключить.

Селектор `:lang` выбирает элементы на основании атрибута `lang`:

```
<!DOCTYPE html>
<html lang="ru">
  <head>
```

```

<meta charset="utf-8">
<title>Селекторы в CSS3</title>
<style>
    :lang(ru) {
        color: red;
    }
</style>
</head>
<body>
    <form>
        <p lang="ru-RU">Я изучаю CSS3</p>
        <p lang="en-US">I study CSS3</p>
        <p lang="de-DE">Ich lerne CSS3</p>
    </form>
</body>
</html>

```

6.17. Псевдоклассы дочерних элементов

Особую группу псевдоклассов образуют псевдоклассы, которые позволяют выбрать определенные дочерние элементы:

- `:first-child`: представляет элемент, который является первым дочерним элементом;
- `:last-child`: представляет элемент, который является последним дочерним элементом;
- `:only-child`: представляет элемент, который является единственным дочерним элементом в каком-нибудь контейнере;
- `:only-of-type`: выбирает элемент, который является единственным элементом определенного типа (тега) в каком-нибудь контейнере;
- `:nth-child(n)`: представляет дочерний элемент, который имеет определенный номер *n*, например, второй дочерний элемент;
- `:nth-last-child(n)`: представляет дочерний элемент, который имеет определенный номер *n*, начиная с конца;
- `:nth-of-type(n)`: выбирает дочерний элемент определенного типа, который имеет определенный номер;
- `:nth-last-of-type(n)`: выбирает дочерний элемент определенного типа, который имеет определенный номер, начиная с конца.

Используем псевдокласс **first-child** для выбора первых ссылок в блоках:

```

<!DOCTYPE html>
<html lang="ru">
    <head>
        <meta charset="utf-8">
        <title>Селекторы в CSS3</title>
        <style>

```

```

        a:first-child{
            color: red;
        }
    </style>
</head>
<body>
    <h3>Планшеты</h3>
    <div>
        <a>Microsoft Surface Pro 4</a><br/>
        <a>Apple iPad Pro</a><br/>
        <a>ASUS ZenPad Z380KL</a>
    </div>
    <h3>Смартфоны</h3>
    <div>
        <p>Топ-смартфоны 2016</p>
        <a>Samsung Galaxy S7</a><br/>
        <a>Apple iPhone SE</a><br/>
        <a>Huawei P9</a>
    </div>
</body>
</html>

```

Стиль по селектору `a:first-child` применяется к ссылке, если она является первым дочерним элементом любого элемента.

В первом блоке элемент ссылки является первым дочерним элементом, поэтому к нему применяется определенный стиль.

А во втором блоке первым элементом является параграф, поэтому ни к одной ссылке не применяется стиль.

Используем псевдокласс `last-child`:

```

<!DOCTYPE html>
<html lang="ru">
    <head>
        <meta charset="utf-8">
        <title>Селекторы в CSS3</title>
        <style>
            a:last-child{
                color: blue;
            }
        </style>
    </head>
    <body>
        <h3>Смартфоны</h3>
        <div>
            <a>Samsung Galaxy S7</a><br/>
            <a>Apple iPhone SE</a><br/>
            <a>Huawei P9</a>
        </div>
    </body>
</html>

```

```

</div>
<h3>Планшеты</h3>
<div>
    <a>Microsoft Surface Pro 4</a><br/>
    <a>Apple iPad Pro</a><br/>
    <a>ASUS ZenPad Z380KL</a>
    <p>Данные за 2016</p>
</div>
</body>
</html>

```

Селектор `a:last-child` определяет стиль для ссылок, которые являются последними дочерними элементами.

В первом блоке как раз последним дочерним элементом является ссылка. А вот во втором последним дочерним элементом является параграф, поэтому во втором блоке стиль не применяется ни к одной из ссылок.

Селектор `:only-child` выбирает элементы, которые являются единственными дочерними элементами в контейнерах:

```

<!DOCTYPE html>
<html lang="ru">
  <head>
    <meta charset="utf-8">
    <title>Селекторы в CSS3</title>
    <style>
      p:only-child{
        color:red;
      }
    </style>
  </head>
  <body>
    <h2>Заголовок</h2>
    <div>
      <p>Текст1</p>
    </div>
    <div>
      <p>Текст2</p>
      <p>Текст3</p>
    </div>
    <div>
      <p>Текст4</p>
    </div>
  </body>
</html>

```

Параграфы с текстами "Текст1" и "Текст4" являются единственными дочерними элементами в своих внешних контейнерах, поэтому к ним применяется стиль - красный цвет шрифта.

Псевдокласс `only-of-type` выбирает элемент, который является единственным элементом определенного типа в контейнере. Например, единственный элемент `div`, при этом элементов других типов в этом же контейнере может быть сколько угодно.

```
<!DOCTYPE html>
<html lang="ru">
  <head>
    <meta charset="utf-8">
    <title>Селекторы в CSS3</title>
    <style>
      span:only-of-type{
        color: green; /* зеленый цвет */
      }
      p:only-of-type{
        color: red; /* красный цвет */
      }
      div:only-of-type{
        color: blue; /* синий цвет */
      }
    </style>
  </head>
  <body>
    <div> Header </div>
    <p>Единственный параграф и
      <span>элемент span</span></p>
    <div> Footer </div>
  </body>
</html>
```

Хотя для элементов `div` определен стиль, он не будет применяться, так как в контейнере `body` находится два элемента `div`, а не один. Зато в `body` есть только один элемент `p`, поэтому он получит стилизацию. И также в контейнере `p` есть только один элемент `span`, поэтому он также будет стилизован.

Псевдокласс `nth-child` позволяет стилизовать каждый второй, третий элемент, только четные или только нечетные элементы и т.д.

Например, стилизуем четные и нечетные строки таблицы:

```
<!DOCTYPE html>
<html lang="ru">
  <head>
    <meta charset="utf-8">
    <title>Селекторы в CSS3</title>
    <style>
      tr:nth-child(odd) { background-color: #bbb; }
      tr:nth-child(even) { background-color: #fff; }
    </style>
  </head>
```



```

<body>
  <h3>Смартфоны</h3>
<table>
<tr><td>Samsung</td><td>Galaxy S7 Edge</td><td>60000</td></tr>
<tr><td>Apple</td><td>iPhone SE</td><td>39000</td></tr>
<tr><td>Microsoft</td><td>Lumia 650</td><td>13500</td></tr>
<tr><td>Alcatel</td><td>Idol S4</td><td>30000</td></tr>
<tr><td>Huawei</td><td>P9</td><td>60000</td></tr>
<tr><td>HTC</td><td>HTC 10</td><td>50000</td></tr>
<tr><td>Meizu</td><td>Pro 6</td><td>40000</td></tr>
<tr><td>Xiaomi</td><td>Mi5</td><td>35000</td></tr>
</table>
</body>
</html>

```

Чтобы определить стиль для нечетных элементов, в селектор передается значение "odd":

```
tr:nth-child(odd) {}
```

Для стилизации четных элементов в селектор передается значение "even":

```
tr:nth-child(even) {}
```

Также в этот селектор мы можем передать номер стилизуемого элемента:

```
tr:nth-child(3) { background-color: #bbb; }
```

В данном случае стилизуется третья строка.

Еще одну возможность представляет использование заменителя для номера, который выражается буквой n:

```
tr:nth-child(2n+1) { background-color: #bbb; }
```

Здесь стиль применяется к каждой второй нечетной строке.

Число перед n (в данном случае 2) представляет тот дочерний элемент, который будет выделен следующим. Число, которое идет после знака плюс, показывают, с какого элемента нужно начинать выделение, то есть, +1 означает, что нужно начинать с первого дочернего элемента.

Таким образом, в данном случае выделение начинается с 1-го элемента, а следующим выделяется $2 * 1 + 1 = 3$ -й элемент, далее $2 * 2 + 1 = 5$ -й элемент и так далее.

К примеру, если мы хотим выделить каждый третий элемент, начиная со второго, то мы могли бы написать:

```
tr:nth-child(3n+2) { background-color: #bbb; }
```

Псевдокласс :nth-last-child по сути предоставляет ту же самую функциональность, только отсчет элементов идет не с начала, а с конца:

```

tr:nth-last-child(2) {
  background-color: #bbb;
  /* 2 строка с конца, то есть предпоследняя */
}
tr:nth-last-child(2n+1) {
  background-color: #eee;
  /* нечетные строки, начиная с конца */
}

```

```
}
```

Псевдокласс `:nth-of-type` позволяет выбрать дочерний элемент определенного типа по определенному номеру:

```
tr:nth-of-type(2) {  
    background-color: #bbb;  
}
```

Аналогично работает псевдокласс **`nth-last-of-type`**, только теперь отсчет элементов идет с конца:

```
tr:nth-last-of-type(2n) {  
    background-color: #bbb;  
}
```

6.18. Псевдоклассы форм

Ряд псевдоклассов используется для работы с элементами форм (см. лекцию 5):

- **`:enabled`**: выбирает элемент, если он доступен для выбора (то есть у него не установлен атрибут `disabled`)
- **`:disabled`**: выбирает элемент, если он не доступен для выбора (то есть у него установлен атрибут `disabled`)
- **`:checked`**: выбирает элемент, если у него установлен атрибут `checked` (для флажков и радиокнопок)
- **`:default`**: выбирает элементы по умолчанию
- **`:valid`**: выбирает элемент, если его значение проходит валидацию HTML5
- **`:invalid`**: выбирает элемент, если его значение не проходит валидацию
- **`:in-range`**: выбирает элемент, если его значение находится в определенном диапазоне (для элементов типа ползунка)
- **`:out-of-range`**: выбирает элемент, если его значение не находится в определенном диапазоне
- **`:required`**: выбирает элемент, если у него установлен атрибут `required`
- **`:optional`**: выбирает элемент, если у него не установлен атрибут `required`

Псевдоклассы `enabled` и `disabled` выбирают элементы форм в зависимости от того, установлен ли у них атрибут `disabled`:

```
<!DOCTYPE html>  
<html lang="ru">  
  <head>  
    <meta charset="utf-8">  
    <title>Селекторы в CSS3</title>  
    <style>  
      :enabled {  
        border: 2px black solid;  
        color: red;  
      }
```

```

    }
    </style>
</head>
<body>
    <p><input type="text" value="Enabled" /></p>
    <p><input type="text" disabled value="Disabled" /></p>
</body>
</html>

```

Псевдокласс `checked` стилизует элементы формы, у которых установлен атрибут `checked`:

```

<!DOCTYPE html>
<html lang="ru">
    <head>
        <meta charset="utf-8">
        <title>Селекторы в CSS3</title>
        <style>
            :checked + span {
                color: red;
                font-weight: bold; /* выделение жирным */
            }
        </style>
    </head>
    <body>
        <h2>Выберите технологию</h2>
        <p>
            <input type="checkbox" checked
                name="html5"/><span>HTML5</span>
        </p>
        <p>
            <input type="checkbox" name="dotnet"/><span>.NET</span>
        </p>
        <p>
            <input type="checkbox" name="java"/><span>Java</span>
        </p>
        <h2>Укажите пол</h2>
        <p>
            <input type="radio" value="man"
                checked name="gender"/><span>мужской</span>
        </p>
        <p>
            <input type="radio" value="woman"
                name="gender"/><span>женский</span>
        </p>
    </body>
</html>

```

Селектор `:checked + span` позволяет выбрать элемент, соседний с отмеченным элементом формы, в нашем случае – подписи к элементам, заключённые в тег `span`. На практике для формирования подписей к флажкам и радиокнопкам следует использовать тег `label` с атрибутом `for`.

Псевдокласс `:default` выбирает стандартный элемент на форме. Как правило, в роли такого элемента выступает кнопка отправки:

```
<!DOCTYPE html>
<html lang="ru">
  <head>
    <meta charset="utf-8">
    <title>Селекторы в CSS3</title>
    <style>
      :default {
        border: 2px solid red;
      }
    </style>
  </head>
  <body>
    <form>
      <input name="login"/>
      <input type="submit" value="Войти" />
    </form>
  </body>
</html>
```

Псевдоклассы `:valid` и `:invalid` стилизуют элементы формы в зависимости от того, проходят они валидацию или нет.

```
<!DOCTYPE html>
<html lang="ru">
  <head>
    <meta charset="utf-8">
    <title>Селекторы в CSS3</title>
    <style>
      input:invalid {
        border: 2px solid red;
      }
      input:valid {
        border: 2px solid green;
      }
    </style>
  </head>
  <body>
    <form>
      <p><input type="text" name="login"
        placeholder="Введите логин" required /></p>
```

```

<p><input type="password" name="password"
  placeholder="Введите пароль" required /></p>
<input type="submit" value="Войти" />
</form>
  </body>
</html>

```

В нашем случае обрамление полей ввода логина и пароля станет зелёным, когда в поля формы будет что-либо введено.

Псевдоклассы `:in-range` и `:out-of-range` стилизуют элементы формы в зависимости от того, попадает ли их значение в определенный диапазон. Это в первую очередь относится к элементу `<input type="number" >`

```

<!DOCTYPE html>
<html lang="ru">
  <head>
    <meta charset="utf-8">
    <title>Селекторы в CSS3</title>
    <style>
      :in-range {
        border: 2px solid green;
      }
      :out-of-range {
        border: 2px solid red;
      }
    </style>
  </head>
  <body>
<form>
  <p>
    <label for="age">Ваш возраст:</label>
    <input type="number" min="1" max="100" value="10"
      id="age" name="age"/>
  </p>
  <input type="submit" value="Отправить" />
</form>
</body>
</html>

```

Здесь атрибуты `min` и `max` задают диапазон, в которое должно попадать вводимое в поле значение. В нашем случае рамка поля ввода станет зелёной, если в него будет введено числовое значение от 10 до 100 включительно.

Псевдоклассы `:required` и `:optional` стилизуют элемент в зависимости от того, установлен ли у него атрибут `required`:

```

<!DOCTYPE html>
<html lang="ru">
  <head>

```

```

<meta charset="utf-8">
<title>Селекторы в CSS3</title>
<style>
    :optional {
        border: 2px solid blue;
    }
    :required {
        border: 2px solid red;
    }
</style>
</head>
<body>
<form>
<p>
    <label for="login">Логин:</label>
    <input type="text" id="login" name="login" required />
</p>
<p>
    <label for="password">Пароль:</label>
    <input type="password" id="password"
        name="password" required />
</p>
<p>
    <label for="name">Имя:</label>
    <input type="text" id="name" name="name" />
</p>
    <input type="submit" value="Регистрация" />
</form>
</body>
</html>

```

В нашем случае первые два поля ввода будут стилизованы красным обрамлением, а третье – синим.

6.19. Псевдоэлементы

Псевдоэлементы обладают рядом дополнительных возможностей по выбору элементов web-страницы и отчасти похожи на псевдоклассы. Список доступных псевдоэлементов:

- `::first-letter`: позволяет выбрать первую букву из текста;
- `::first-line`: стилизует первую строку текста;
- `::before`: добавляет сообщение до определенного элемента;
- `::after`: добавляет сообщение после определенного элемента;
- `::selection`: выбирает выбранные пользователем элементы;

В CSS2 перед псевдоэлементами, как и перед псевдоклассами, ставилось одно двоеточие. В CSS3 для отличия их от псевдоклассов псевдоэлементы стали предваряться двумя двоеточиями. Однако для совместимости с более старыми

браузерами, которые не поддерживают CSS3, допустимо использование одного двоеточия: `:before`.

Стилизуем текст, используя псевдоэлементы **first-letter** и **first-line**:

```
<!DOCTYPE html>
<html lang="ru">
  <head>
    <meta charset="utf-8">
    <title>Псевдоклассы в CSS3</title>
    <style>
      ::first-letter { color:red; font-size: 25px; }
      ::first-line { font-size: 20px; }
    </style>
  </head>
  <body>
    <p>Но он ничего не видал. Над ним не было ничего уже,
кроме неба, — высокого неба.</p>
  </body>
</html>
```

Используем псевдоэлементы **before** и **after**:

```
<!DOCTYPE html>
<html lang="ru">
  <head>
    <meta charset="utf-8">
    <title>Псевдоклассы в CSS3</title>
  <style>
    .warning::before{ content: "Важно! "; font-weight: bold; }
    .warning::after { content: " Будьте осторожны!";
      font-weight: bold;}
  </style>
</head>
<body>
  <p><span class="warning">Не пытайтесь засунуть палец в
электрическую розетку.</span></p>
</body>
</html>
```

Здесь псевдоэлементы применяются к элементу с классом `warning`. Оба псевдоэлемента принимают свойство `content`, которое хранит вставляемый текст. И также для повышения внимания псевдоэлементы используют выделение текста жирным с помощью свойства `font-weight: bold;`.

Используем псевдоэлемент **selection** для стилизации выбранных элементов:

```
<!DOCTYPE html>
<html lang="ru">
  <head>
```

```

<meta charset="utf-8">
<title>Псевдоэлементы в CSS3</title>
<style>
    ::selection {
        color: white;
        background-color: black;
    }
</style>
</head>
<body>
<p>Псевдоэлементы в CSS3 позволяют форматировать текст.</p>
</body>
</html>

```

Результат будет виден при выделении части текста на странице.

6.20. Селекторы атрибутов

Кроме селекторов элементов мы также можем использовать селекторы их **атрибутов**. Например, у нас есть на web-странице несколько полей `input`, а нам надо окрасить в красный цвет только текстовые поля. В этом случае мы как раз можем проверять значение атрибута `type`: если оно имеет значение `text`, то это текстовое поле, и соответственно его надо окрасить в красный цвет. Определение стиля в этом случае выглядело бы так:

```

input[type="text"]{
    border: 2px solid red;
}

```

После элемента в квадратных скобках идет атрибут и его значение. То есть в данном случае мы говорим, что для текстового поля надо установить границу красного цвета 2 пикселя толщиной сплошной линией.

Например:

```

<!DOCTYPE html>
<html lang="ru">
    <head>
        <meta charset="utf-8">
        <title>Селекторы атрибутов в CSS3</title>
        <style>
            input[type="text"]{
                border: 2px solid red;
            }
        </style>
    </head>
    <body>
        <p><input type="text" id="login" /></p>
        <p><input type="password" id="password" /></p>
        <input type="submit" value="Send" />
    </body>

```



```
</html>
```

Селекторы атрибутов можно применять не только к элементам, но и классам и идентификаторам. Например:

```
<!DOCTYPE html>
<html lang="ru">
  <head>
    <meta charset="utf-8">
    <title>Селекторы атрибутов в CSS3</title>
    <style>
      .link[href="http://apple.com"]{
        color: red;
      }
    </style>
  </head>
  <body>
    <a class="link" href="http://microsoft.com">Microsoft</a> |
    <a class="link" href="https://google.com">Google</a> |
    <a class="link" href="http://apple.com">Apple</a>
  </body>
</html>
```

Специальные символы позволяют конкретизировать значение атрибутов. Например символ ^ позволяет выбрать все атрибуты, которые начинаются на определенный текст. Например, нам надо выбрать все ссылки, которые используют протокол https, то есть ссылка должна начинаться на "https://". В этом случае можно применить следующий селектор:

```
a[href^="https://"]{
  color: red;
}
```

Если значение атрибута должно иметь в конце определенный текст, то для проверки используется символ \$. Например, нам надо выбрать все изображения в формате jpg. В этом случае мы можем проверить, оканчивается ли значение атрибута src на текст ".jpg":

```
img[src$=".jpg"]{
  width: 100px;
}
```

Символ "*" (звездочка) позволяет выбрать все элементы с атрибутами, которые в своем значении имеют определенный текст (неважно где - в начале, в середине или в конце):

```
a[href*="microsoft"]{
  color: red;
}
```

Данный атрибут выберет все ссылки, которые в своем адресе имеют текст "microsoft".

6.21. Наследование стилей

Для упрощения определения стилей в CSS применяется механизм **наследования стилей**. Этот механизм предполагает, что вложенные элементы могут наследовать стили своих элементов-контейнеров. Например, пусть на web-странице имеются заголовок и параграф, которые должны иметь текст красного цвета. Мы можем отдельно к параграфу и заголовку применить соответствующий стиль, который установит нужный цвет шрифта:

```
<!DOCTYPE html>
<html lang="ru">
  <head>
    <meta charset="utf-8">
    <title>Наследование стилей в CSS3</title>
    <style>
      p {color: red;}
      h2 {color: red;}
    </style>
  </head>
  <body>
    <h2>Наследование стилей</h2>
    <p>Текст про наследование стилей в CSS 3</p>
  </body>
</html>
```

Однако поскольку и элемент p, и элемент h2 находятся в элементе body, то они наследуют от этого контейнера - элемента body многие стили. И чтобы не дублировать определение стиля, мы могли бы написать так:

```
<!DOCTYPE html>
<html lang="ru">
  <head>
    <meta charset="utf-8">
    <title>Наследование стилей в CSS3</title>
    <style>
      body {color: red;}
    </style>
  </head>
  <body>
    <h2>Наследование стилей</h2>
    <p>Текст про наследование стилей в CSS 3</p>
  </body>
</html>
```

В итоге определение стилей стало проще, а результат остался тем же.

Если нам нежелателен унаследованный стиль, то мы его можем переопределить для определенных элементов:

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

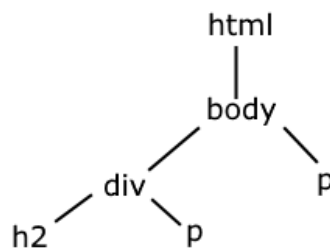
При нескольких уровнях вложенности элементы наследуют стили только ближайшего контейнера:

```

<!DOCTYPE html>
<html lang="ru">
  <head>
    <meta charset="utf-8">
    <title>Наследование стилей в CSS3</title>
    <style>
      body {color: red;}
      div {color:black;}
    </style>
  </head>
  <body>
    <div>
      <h2>Наследование стилей</h2>
      <p>Текст про наследование стилей в CSS 3</p>
    </div>
    <p>Copyright © MyCorp.com</p>
  </body>
</html>

```

Здесь web-страница имеет следующую структуру:



Для элемента `div` переопределяется цвет текста. И так как элемент `h2` и один из параграфов находятся в элементе `div`, то они наследуют стиль именно элемента `div`. Второй параграф находится непосредственно в элементе `body` и поэтому наследует стиль элемента `body`. В результате заголовок будет чёрным, а текст абзаца красным.

К наследуемым относятся в основном свойства, определяющие параметры отображения текста: `font-size`, `font-family`, `font-style`, `font-weight`, `color`, `text-align`, `text-transform`, `text-indent`, `line-height`, `letter-spacing`, `word-spacing`, `white-space`, `direction` и другие.

Также к наследуемым свойствам относятся `list-style`, `cursor`, `visibility`, `border-collapse` и некоторые другие. Но они используются значительно реже.

Наследуемые свойства можно и нужно задавать через предков, следуя структуре документа.

Например, параметры текста зачастую не меняются в пределах крупных блоков страницы: меню, основного содержания, информационных панелей. Поэтому общие параметры текста (цвет, размер, гарнитура) обычно указывают в стилях этих крупных блоков.

С другой стороны, не ко всем свойствам CSS применяется наследование. Например, свойства, которые представляют отступы (margin, padding) и границы (border) элементов, не наследуются.

Кроме того, браузеры по умолчанию также применяют ряд предустановленных стилей к элементам. Например, заголовки имеют определенную высоту и т.д.

6.22. Каскадность стилей

Если же к одному и тому же элементу применяется несколько различных стилей, то возникает вопрос, какой же из этих стилей будет в реальности действовать?

В CSS действует **механизм каскадности**, которую можно определить как набор правил, определяющих последовательность применения множества стилей к одному и тому же элементу.

К примеру, у нас определена следующая web-страница:

```
<!DOCTYPE html>
<html lang="ru">
  <head>
    <meta charset="utf-8">
    <title>Каскадность стилей в CSS3</title>
  <style>
    .redLink {color: red;} /* красный цвет текста */
    .footer a {font-weight: bold;} /* выделение жирным */
    a {text-decoration: none;} /* отмена подчеркивания ссылки */
  </style>
</head>
<body>
  <p class="footer">Для просмотра подробной информации
    пройдите по ссылке:
    <a class="redLink" href="index.php">Основы CSS 3</a>
  </p>
</body>
</html>
```

Здесь определено три стиля и все они применяются к ссылке.

Если к элементу web-страницы применяется несколько стилей, которые не конфликтуют между собой, то браузер объединяет их в один стиль.

Так, в данном случае, все три стиля не конфликтуют между собой, поэтому все эти стили будут применяться к ссылке.

Если же стили конфликтуют между собой, например, определяют разный цвет текста, то в этом случае применяется сложная система правил для вычисления значимости каждого стиля. Все эти правила описаны в спецификации CSS: [Calculating a selectors specificity](#).

Вкратце разберем их.

Для определения стиля к элементу могут применяться различные селекторы, и важность каждого селектора оценивается в баллах. Чем больше у селектора пунктов, тем он важнее, и тем больший приоритет его стили имеют над стилями других селекторов.

- Селекторы тегов имеют важность, оцениваемую в 1 балл;
- Селекторы классов, атрибутов и псевдоклассов оцениваются в 10 баллов;
- Селекторы идентификаторов оцениваются в 100 баллов;
- Встроенные inline-стили (задаваемые через атрибут style) оцениваются в 1000 баллов.

Например:

```
<!DOCTYPE html>
<html lang="ru">
<head>
<meta charset="utf-8">
<title>Каскадность стилей в CSS3</title>
<style>
#index {color: navy;} /* темно-синий текст */
.redLink {color: red; font-size: 20px;} /* красный текст и шрифт 20 пикселей */
a {color: black; font-weight: bold;} /* черный цвет и выделение жирным */
</style>
</head>
<body>
  <a id="index" class="redLink" href="index.php">
    Основы CSS 3</a>
</body>
</html>
```

Здесь к ссылке применяется сразу три стиля. Эти стили содержат два не конфликтующих правила:

```
font-size: 20px;
font-weight: bold;
```

которые устанавливают высоту шрифта 20 пикселей и выделение ссылки жирным. Так как каждое из этих правил определено только в одном стиле, то в итоге они будут суммироваться и применяться к ссылке без проблем.

Кроме того, все три стиля содержат определение цвета текста, но каждый стиль определяет свой цвет текста. Так как селекторы идентификаторов имеют больший удельный вес, то будет применяться темно-синий цвет, задаваемый селектором:

```
#index {color: navy;}
```

Если селектор является **составным**, то происходит сложение баллов всех входящих в селектор подселекторов. Так, рассмотрим следующий пример:

```
<!DOCTYPE html>
<html lang="ru">
<head>
<meta charset="utf-8">
<title>Каскадность стилей в CSS3</title>
<style>
  a {font-size: 18px;}
  .nav li a {color: red;} /* красный текст */
```

```

#menu a {color: navy;}    /* темно-синий текст */
.nav .menuItem {color: green;} /* зеленый текст */
a.menuItem:not(.newsLink) {color: orange;}
/* оранжевый текст*/
div ul li a {color: gray; } /* серый текст */
</style>
</head>
<body>
    <div id="menu">
        <ul class="nav">
            <li><a class="menuItem">Главная</a></li>
            <li><a class="menuItem">Форум</a></li>
            <li><a class="menuItem">Блог</a></li>
            <li><a class="menuItem">О сайте</a></li>
        </ul>
    </div>
</body>
</html>

```

В стиле определено пять различных селекторов, которые устанавливают цвет ссылок. В итоге браузер выберет селектор #menu a и окрасит ссылки в **тёмно-синий** цвет. Но почему, на каком основании браузер выберет этот селектор?

Рассмотрим, как у нас будут суммироваться баллы по каждому из пяти селекторов:

Селектор	Идентификаторы	Классы	Теги	Сумма
.nav li a	0	1	2	12
#menu a	1	0	1	101
.nav .menuItem	0	2	0	20
a.menuItem:not(.newsLink)	0	2	1	21
div ul li a	0	0	4	4

Итак, мы видим, что для селектора #menu a в колонке "сумма" оказалось больше всего баллов - 101. То есть в нем 1 идентификатор (100 баллов) и один тег(1 балл), которые в сумме дают 101 балл.

К примеру, в селекторе .nav .menuItem два селектора класса, каждый из которых дает 10 баллов, то есть в сумме 20 баллов.

При этом псевдокласс :not в отличие от других псевдоклассов не учитывается, однако учитывается тот селектор, который передается в псевдокласс not.

Правило !important

CSS предоставляет возможность полностью отменить значимость стилей. Для этого в конце стиля указывается значение !important:

```

a {font-size: 18px; color: red !important;}
#menu a {color: navy;}

```

В этом случае вне зависимости от наличия других селекторов с большим количеством баллов, к ссылкам будет применяться красный цвет, определяемый первым стилем.

6.23. Единицы измерения в CSS

Все единицы измерения в CSS делятся на *абсолютные* и *относительные*.

Абсолютные единицы измерения привязаны к настоящим физическим размерам и связаны между собой жёсткими пропорциями. Примеры абсолютных единиц измерения:

```
font-size: 1cm; /* 1 сантиметр */  
font-size: 10mm; /* В 1 сантиметре 10 миллиметров */  
font-size: 38px; /* В 1 сантиметре 38 пикселей */
```

Пиксели, px, используют чаще всего, остальные абсолютные единицы почти не применяют.




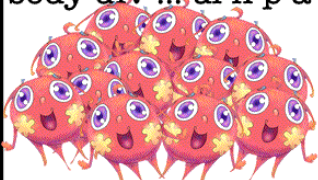
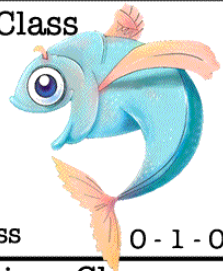
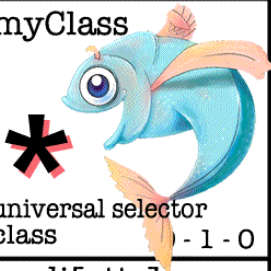
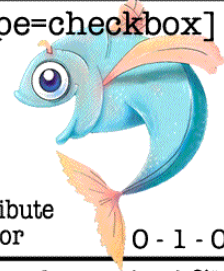
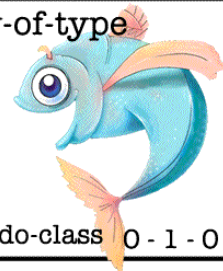
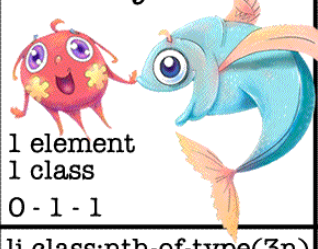
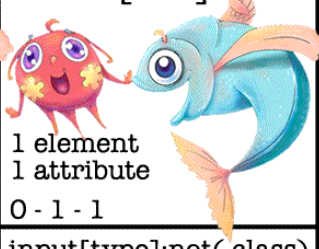
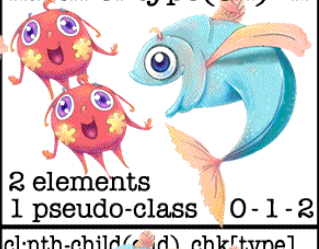
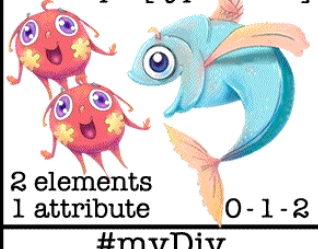
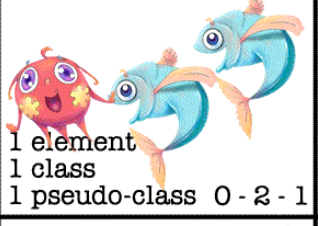
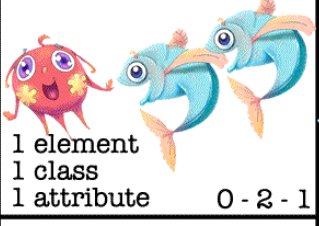

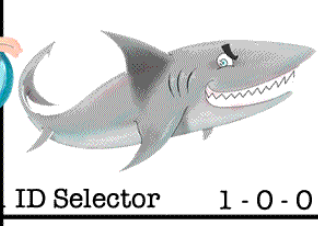
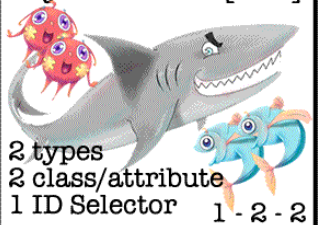
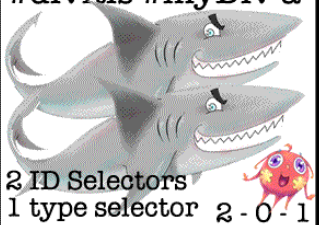
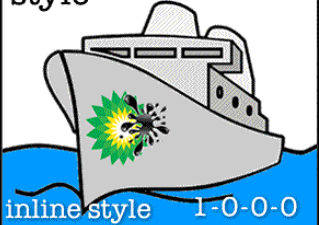

Относительные единицы измерения описывают значения, которые зависят от других значений. Например, ширина элемента в процентах зависит от ширины родительского элемента, а ширина элемента в em зависит от размера шрифта самого элемента.

К относительным единицам относятся em, rem, vh, vw и некоторые другие, ну и, конечно же, проценты. Каждая из таких единиц решает свой круг задач. Например, проценты используют для "резиновой" вёрстки, а em применяют в вёрстке государственных сайтов с особыми дополнительными требованиями к масштабированию текста.

Перечень всех [абсолютных единиц измерения](#) и их соотношений есть в спецификации. Там же, в спецификации, есть перечень всех [относительных единиц измерения](#) и описание правил расчёта.

CSS SPECIFISHITY

WITH PLANKTON, FISH AND SHARKS

*  universal selector 0-0-0	div  1 element 0-0-1	li > ul  2 elements 0-0-2	body div ...ul li p a  12 elements 0-0-12
.myClass  1 class 0-1-0	*.myClass  1 universal selector 1 class 0-1-0	[type=checkbox]  1 attribute selector 0-1-0	:only-of-type  1 pseudo-class 0-1-0
li.myClass  1 element 1 class 0-1-1	li[attr]  1 element 1 attribute 0-1-1	li:nth-of-type(3n)~li  2 elements 1 pseudo-class 0-1-2	form input[type=email]  2 elements 1 attribute 0-1-2
li.class:nth-of-type(3n)  1 element 1 class 1 pseudo-class 0-2-1	input[type]:not(.class)  1 element 1 class 1 attribute 0-2-1	cl:nth-child(4n)chk[type]...  10 class/attribute/ pseudo-classes 0-10-0	#myDiv  ID Selector 1-0-0
#myDiv li.class a[href]  2 types 2 class/attribute 1 ID Selector 1-2-2	#divitis #myDiv a  2 ID Selectors 1 type selector 2-0-1	style=""  inline style 1-0-0-0	!important  !important 1-0-0-0-0

X-0-0: The number of ID selectors

0-Y-0: The number of class selectors, attributes selectors, and pseudo-classes

0-0-Z: The number of type selectors and pseudo-elements

*, +, >, ~ : Universal selector and combinators do not increase specificity

:not(x): Negation selector has no value. Argument increases specificity

ESTELLE WEYL *@ESTELLEW *WWW.STANDARDISTA.COM * 2104



Шпаргалка по основным селекторам

Селектор	CSS	HTML
тега (элемента)	p { color: red; }	<p>Текст</p> применится ко всем абзацам
класса	.red { color: red; }	<p class="red">Текст</p> применится к абзацам с указанным классом
идентификатора	#red { color: red; }	<p id="red">Текст</p> применится к абзацу с указанным идентификатором
псевдокласса	p:hover { color: red; }	<p>Текст</p> применится к абзацам при наведении курсора мыши на содержимое
псевдоэлемента	p::first-letter { color: red; }	<p>Текст</p> применится к первым буквам абзацев
атрибута	p[align="center"] { color: red; }	<p align="center">Текст</p> применится к центрированным атрибутом align абзацам

Задание к лабораторной работе 6:

Разработать валидный стилевой файл для сайта и подключить его к страницам.
Предусмотреть в коде селекторы элементов, классы, идентификаторы.