

Тема 6: Знакомство с CSS

Каскадные (многоуровневые) таблицы стилей - cascading style sheets (CSS) - это мощный стандарт на основе текстового формата, определяющий представление данных в браузере.

Ранее при работе с таблицами мы уже применяли CSS, но теперь нужно определить понятия, которыми нам нужно оперировать, для примера возьмем такой код:

```
/*Это комментарий, который игнорируется при обработке*/
```

```
p  
  
{  
  
    width: 80%;  
  
    margin: 0 auto;  
  
    background: red;  
  
}
```

Где:

p – это *селектор* (подробнее будут рассмотрены в главе 7), позволяют осуществлять выборку необходимых нам элементов HTML.

width и margin – это *свойство*, которое определяет какой именно параметр мы хотим изменить.

80% и 0 auto – это *значения* свойств. Некоторые свойства требуют на ввод несколько значений.

Практическая работа №6.1

Попробуем посмотреть, что получится при применении этих свойств. Для этого запишем код в html файл в таком виде. Текст будет иметь боковые внешние отступы, располагаться по центру с красным фоном.

```
<!DOCTYPE html>
<html>
<head>
  <title>Знакомство с CSS</title>
  <meta charset="utf-8">
  <script></script>
  <style>
    p {
      width: 70%;
      margin: 0 auto;
      background-color: red;
    }
  </style>
</head>
<body>
  <p>
    Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
    tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
    quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
    consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse
    cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
    proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
  </p>
</body>
</html>
```

Теперь пора познакомиться со способами применения CSS свойств:

- 1) Вам уже знакомый способ с помощью тега стилей - `<style>`.

Пример смотрите выше.

- 2) С помощью прямого написания свойств к тегу html.

Осуществляется атрибутом style.

```
<p style="width: 70%; margin: 0 auto; background-color: red;"> ... </p>
```

Внимание!!! данный способ не эффективен при множественной выборке элементов, так как приходится дублировать атрибут в каждый из тегов, в связи с этим этот способ считается не верным.

3) Подключение CSS внешним файлом с помощью тега <link>. В самой первой теме была рассмотрена структура папок. Теперь в папке css создайте файл с расширением *.css (чаще всего файл называют style.css). Теперь подключим созданный файл.

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

Теперь ваша таблица стилей будет храниться отдельно и далее вы будете видеть два кода: HTML и CSS.

Практическая работа №6.2

Не изменяя текст HTML перенесите селектор со значениями в отдельный файл.

```
<!DOCTYPE html>
<html>
<head>
  <title>Знакомство с CSS</title>
  <meta charset="utf-8">
  <script></script>
  <link rel="stylesheet" type="text/css" href="css/style.css">
</head>
<body>
  <p>
    Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
    tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
    quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
    consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse
    cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
    proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
  </p>
</body>
</html>

p {
  width: 70%;
  margin: 0 auto;
  background-color: red;
}
```

Проверка знаний

Приведите пример, где подключение через атрибут style напрямую покажет свою неэффективность.

Тема 7: Селекторы, псевдоклассы и псевдоэлементы

*** - селектор всех элементов.

section – селектор всех тегов с таким именем.

#id – селектор элемента с данным id.

.class – селектор элементов с таким классом.

[attribute="value"] – селектор по атрибуту (значения атрибута).

:visited – это псевдокласс, куда могут быть записаны разные условия элемента.

Селекторы можно комбинировать, записывая последовательно, без пробела.

.class1.class2 – селектор для элементов с одновременно двумя классами.

a#id:visited – элемент тега `<a>` с данным id и псевдоклассом *:visited*.

Также важно для селекторов важно знать отношения:

section p – элементы `<p>`, являющиеся потомками *section*.

section > p – только непосредственные потомки *section*.

section ~ p – все `<p>` на том же уровне вложенности, которые идут после *section*.

section + p – первый `<p>` на том же уровне вложенности, который идет сразу после *section*.

Практическая работа №7.1

Давайте разберемся на примере с селекторами, которые мы рассмотрели.

```
<!DOCTYPE html>
<html>
<head>
  <title>Селекторы CSS</title>
  <meta charset="utf-8">
  <link rel="stylesheet" type="text/css" href="css/style.css">
</head>
<body>
  <h3>Балтославянские языки</h3>
  <ol id="languages">
    <li>Славянские языки
      <ol>
        <li>Славянские микроязыки</li>
        <li>Праславянский язык</li>
        <li id="e-slavic">Восточнославянские языки <code>#e-slavic</code></li>
        <li>Западнославянские языки <code>#e-slavic ~ li</code></li>
        <li>Южнославянские языки <code>#e-slavic ~ li</code></li>
        <li>... <code>#e-slavic ~ li</code></li>
      </ol>
    </li>
    <li>Балтийские языки
      <ol>
        <li>Литовский язык</li>
        <li id="latvian">Латышский язык <code>#latvian</code>
          <ol>
            <li>Латгальский язык <code>#latvian *</code></li>
          </ol>
        </li>
        <li>Прусский язык <code>#latvian + li</code></li>
        <li>... (следующий элемент уже не <code>#latvian + li</code>)</li>
      </ol>
    </li>
  </ol>
</body>
</html>
```

```
#languages li {
  color: brown;
}

#languages > li {
  color: black;
}

#e-slavic {
  font-style: italic;
}

#e-slavic ~ li {
  color: red;
}

#latvian {
  font-style: italic;
}

#latvian * {
  font-style: normal
}

#latvian + li {
  color: green;
}

code {
  border: 1px solid black;
}
```

Существуют псевдоклассы для указания его места среди соседей.

:first-child – первый потомок своего родителя.

:last-child – последний потомок своего родителя.

:only-child – единственный потомок своего родителя, соседних элементов нет.

:nth-child(a) – потомок номер a своего родителя, например *:nth-child(2)* – второй потомок. Нумерация начинается с 1.

:nth-child(an+b) – расширение предыдущего селектора через указание номера потомка формулой, где a, b – константы, а под n подразумевается любое целое число. Этот псевдокласс будет фильтровать все элементы, которые попадают под формулу при каком-либо n . Например: *:nth-child(2n)* даст элементы номер 2, 4, 6..., то есть чётные.

:nth-child(2n+1) даст элементы номер 1, 3..., то есть нечётные.

:nth-child(3n+2) даст элементы номер 2, 5, 8 и так далее.

:nth-last-child(a), *:nth-last-child(an+b)* – то же самое, но отсчёт начинается с конца, например *:nth-last-child(2)* – второй элемент с конца.

Практическая работа №7.2

```
<!DOCTYPE html>
<html>
<head>
  <title>Селекторы CSS</title>
  <meta charset="utf-8">
  <link rel="stylesheet" type="text/css" href="css/style.css">
</head>
<body>
  <ul>
    <li>Древнерусский язык <code>li:first-child</code></li>
    <li>Древненовгородский диалект <code>li:nth-child(2n)</code></li>
    <li>Западнорусский письменный язык <code>li:nth-child(3)</code></li>
    <li>Украинский язык <code>li:nth-child(2n)</code></li>
    <li>Белорусский язык</li>
    <li>Славянский язык <code>li:nth-child(2n)</code></li>
    <li>Другие языки <code>li:last-child</code></li>
  </ul>
</body>
</html>

li:nth-child(2n) {
  background: #eee;
}

li:nth-child(3) {
  color: red;
}
li:first-child {
  background: rgba(255, 0, 0, .2);
}
li:last-child {
  background: rgba(0, 0, 255, .2);
}
code {
  border: 1px solid black;
}
```

Есть аналогичные псевдоклассы, которые учитывают не всех соседей, а только с тем же тегом:

:first-of-type

:last-of-type

:only-of-type

:nth-of-type

:nth-last-of-type

Они имеют в точности тот же смысл, что и обычные :first-child, :last-child и так далее, но во время подсчёта игнорируют элементы с другими тегами, чем тот, к которому применяется фильтр.

Практическая работа №7.3

Пример использования для раскраски списка DT «через один» и предпоследнего DD:

```
<!DOCTYPE html>
<html>
<head>
  <title>Селекторы CSS</title>
  <meta charset="utf-8">
  <link rel="stylesheet" type="text/css" href="css/style.css">
</head>
<body>
  <dl>
    <dt>Первый dt</dt>
    <dd>Описание dd</dd>
    <dt>Второй dt <code>dt:nth-of-type(2n)</code></dt>
    <dd>Описание dd</dd>
    <dt>Третий dt</dt>
    <dd>Описание dd <code>dd:nth-last-of-type(2)</code></dd>
    <dt>Четвёртый dt <code>dt:nth-of-type(2n)</code></dt>
    <dd>Описание dd</dd>
  </dl>
</body>
</html>

dt:nth-of-type(2n) {
  background: #eee;
}
dd:nth-last-of-type(2) {
  color: red;
}
code {
  border: 1px solid black;
}
```

Теперь рассмотрим селекторы атрибутов.

На атрибут целиком:

[attr] – атрибут установлен,

[attr="val"] – атрибут равен val.

На начало атрибута:

$[attr^="val"]$ – атрибут начинается с val, например "value".

$[attr/="val"]$ – атрибут равен val или начинается с val-, например равен "val-1".

На содержание:

$[attr*="val"]$ – атрибут содержит подстроку val, например равен "myvalue".

$[attr\sim="val"]$ – атрибут содержит val как одно из значений через пробел.

Например: $[attr\sim="delete"]$ верно для "edit delete" и неверно для "undelete" или "no-delete".

На конец атрибута:

$[attr\$="val"]$ – атрибут заканчивается на val, например равен "myval".

Практическая работа №7.4

```
<ul>
  <li class="focus">Селектор <code>li[class] и li[class*='fo']</code></li>
  <li class="nefocus">По <code>li[class] и li[class*='fo']</code></li>
  <li class="focus-pocus">Атрибутам <code>li[class] и li[class*='fo']</code></li>
  <li>Не</li>
  <li class="nefocus-pocus">Встречается <code>li[class] и li[class*='fo']</code></li>
  <li class="fo">Так <code>li[class] и li[class|='fo']</code></li>
  <li class="triton_spring">Часто <code>li[class]</code></li>
  <li>Но</li>
  <li class="rofl">Его <code>li[class] и li[class='rofl']</code></li>
  <li class="say_no_spring">Нужно <code>li[class] и li[class~='spring']</code></li>
  <li class="spring">Знать <code>li[class] и li[class~='spring']</code></li>
</ul>

li[class] {
  list-style-type: cambodian;
}
li[class='rofl'] {
  color: red;
}
li[class*='fo'] {
  background: rgba(255, 0, 0, .2);
}
li[class|='fo'] {
  background: rgba(0, 255, 0, .2);
}
li[class~='spring'] {
  background: rgba(0, 0, 255, .2);
}
code {
  border: 1px solid black;
}
```

Еще существуют такие псевдоклассы:

:not(селектор) – все, кроме подходящих под селектор.

:focus – в фокусе.

:hover – под мышью.

:empty – без детей (даже без текстовых).

:checked, :disabled, :enabled – состояния INPUT.

:target – этот фильтр сработает для элемента, ID которого совпадает с анкором #... текущего URL.

Например, если на странице есть элемент с `id="intro"`, то правило `:target { color: red }` подсветит его в том случае, если текущий URL имеет вид `http://...#intro`

«Псевдоэлементы» – различные вспомогательные элементы, которые браузер записывает или может записать в документ.

При помощи псевдоэлементов *::before* и *::after* можно добавлять содержимое в начало и конец элемента.

Практическая работа №7.5

Теперь попробуем посмотреть на практике как работают изученные выше псевдоклассы и псевдоэлементы.

```
<!DOCTYPE html>
<html>
<head>
  <title>Селекторы CSS</title>
  <meta charset="utf-8">
  <link rel="stylesheet" type="text/css" href="css/style.css">
</head>
<body>
  <nav>
    <a href="#">Это</a>
    <a href="#">Очень</a>
    <a href="#">Просто</a>
  </nav>
</body>
</html>
```

```
nav > a {
  margin-right: 1vw;
  font-size: 24pt;
  text-decoration: none;
  color: black;
  transition: text-shadow 0.5s;
}
nav > a:hover {
  text-shadow: -1px 1px 2px gray;
}
nav::before {
  content: ">>> ";
  font-size: 24pt;
}
nav::after {
  content: " <<<";
  font-size: 24pt;
}
```

Проверка знаний

Вам дан исходный код HTML и CSS (без селекторов)

```
<body>
  <ul>
    <li class="green_bg">Зеленый фон, синие буквы</li>
    <li>Без изменений
      <ul>
        <li>Синие буквы
          <ul>
            <li>Красные буквы</li>
            <li>Синие буквы
              <ul>
                <li>Синие буквы</li>
              </ul>
            </li>
            <li>Синие буквы
              <ul>
                <li class="green_bg">Зеленый фон, синие буквы</li>
              </ul>
            </li>
          </ul>
        </li>
        <li>Без изменений
          <ul>
            <li>Синие буквы</li>
          </ul>
        </li>
        <li>Синие буквы</li>
      </ul>
    </li>
    <li>Синие буквы
      <ul>
        <li>Синие буквы</li>
      </ul>
    </li>
    <li id="italic_style">Курсив</li>
    <li>Синие буквы
      <ul>
        <li>Синие буквы</li>
        <li>Синие буквы
          <ul>
            <li>Синие буквы, красный фон</li>
          </ul>
        </li>
        <li>Синие буквы</li>
        <li>Синие буквы
          <ul>
            <li>Синие буквы</li>
            <li>Синие буквы
              <ul>
                <li>Синие буквы</li>
              </ul>
            </li>
            <li>Синие буквы</li>
          </ul>
        </li>
        <li>Синие буквы</li>
      </ul>
    </li>
    <li>Синий фон</li>
  </ul>
</body>
```

```

{
    background-color: rgba(0, 255, 0, .3);
}
{
    font-style: italic;
}
{
    color: red;
}
{
    color: blue;
}
{
    background-color: rgba(255, 0, 0, .3);
}
{
    background-color: rgba(0, 0, 255, .3);
}

```

Приведите страницу в соответствие со скриншотом, написав нужные селекторы.

- Зеленый фон, синие буквы
- Без изменений
 - Синие буквы
 - Красные буквы
 - Синие буквы
 - Синие буквы
 - Синие буквы
 - Зеленый фон, синие буквы
 - Без изменений
 - Синие буквы
 - Синие буквы
- Синие буквы
 - Синие буквы
- Курсив
- Синие буквы
 - Синие буквы
 - Синие буквы
 - Синие буквы, красный фон
 - Синие буквы
 - Синие буквы
 - Синие буквы
 - Синие буквы
 - Синие буквы
 - Синие буквы
 - Синие буквы
- Синий фон