

**HTML, CSS,
JavaScript**

- **HTML (HyperText Markup Language)** – это язык разметки, на котором пишутся веб-страницы
- HTML предназначен для того, чтобы задавать структуру и содержимое страницы
- HTML не является языком программирования, а используется только для разметки веб-страниц (задания структуры и содержимого)
- Последняя на данный момент версия – HTML 5
- Файлы HTML – просто текстовые файлы с расширением .html, поэтому работать с ними их можно даже в блокноте
- Но лучше в среде разработки

Среды разработки – от Microsoft

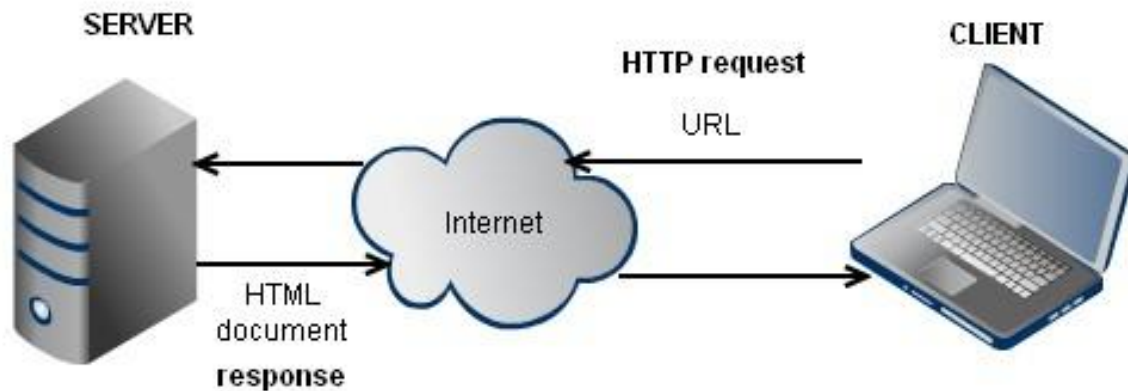
- **Visual Studio 2017 Community**
 - Бесплатна
 - Поддержка HTML, CSS, JavaScript
 - Желательно поставить плагин ReSharper (платный)
- **Visual Studio Code**
 - Бесплатна
 - Очень легковесна и быстро работает
 - Поддержка HTML, CSS, JavaScript
 - Функциональность добавляется плагинами
 - Минус – надо искать и подбирать плагины

Среды разработки – от JetBrains

- **IDEA Ultimate**
 - Community IDEA не имеет поддержки CSS и JS
- **PHPStorm** (та же IDEA, только для PHP и веба, триал)
- **WebStorm** (IDEA для клиентского веба, триал)
- Все эти среды платные, но у них 30-дневный триал. После истечения триала эти среды продолжают работать, просто отключаются каждые полчаса

Браузер

- **Браузер** – приложение, предназначенное для просмотра и работы с веб-страницами
- Браузер умеет:
 - Отображать **HTML** страницы
 - Применять к ним стили **CSS**
 - Исполнять для страницы код на языке **JavaScript**



Структура HTML документа

- `<!DOCTYPE html>`

Первая строка – DOCTYPE.
Указывает тип документа

`<html>`

`<head>`

`<meta charset="UTF-8">`

`<title>Мой первый сайт</title>`

`</head>`

`<body>`

Основное содержимое идет
внутри тега html

В нем есть два тега – head и body

`</body>`

`</html>`

В head находятся общие вещи –
заголовок вкладки, ссылки на
скрипты и стили и др.

meta нужно чтобы
указать кодировку
страницы

В body находится содержимое
страницы

Doctype

- Doctype ставится в начале документа и определяет тип документа: HTML 5, HTML 4.01, XHTML 1.0 и др.
- Для HTML 5: `<!DOCTYPE html>`
- Для HTML 4.01: `<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">`
- В зависимости от типа документа, браузер может отображать его немного по-разному
- Если забыть doctype, то страница в некоторых браузерах будет отображаться неверно. В основном это касается IE
- В этом случае браузер переходит в режим совместимости (**Quirks mode**). В этом случае надо поправлять страницу – добавить забытый doctype

Теги, атрибуты

- HTML разметка состоит из **элементов**
- Каждый элемент обозначается при помощи **тегов**, тег указывает тип элемента
- Например, рассмотрим элемент-абзац:
- `<p>Текст</p>`
- Содержимое абзаца находится между тегами `<p>` и `</p>`
- Первый из этих тегов называется **открывающим тегом**, а второй (со слэшем) – **закрывающим тегом**
- Такие теги называются **парными**. Они ограничивают некоторую часть документа

Теги, атрибуты

- Часть элементов не требует закрывающего тега и не может иметь вложенного содержимого. Это **одиночные теги**
- На их место в документ вставляется некоторый объект
- Например, это теги:
 - Перевод строки: `
`
 - Горизонтальная линия: `<hr />`
 - Изображение: ``
- В нижнем примере у тега `img` есть атрибуты `src` и `alt` – пары ключ-значение. Тег `src` указывает адрес картинки
- Синтаксис атрибута: `name="value"`
- Допускаются одинарные кавычки, но лучше ставить двойные

Нестрогость синтаксиса HTML

- Вообще, браузеры сделаны неприхотливыми
- Если ошибиться в HTML разметке, то браузер, скорее всего, всё равно сможет показать все правильно
- Если, например, забыть закрыть тег или написать теги в верхнем регистре, то браузер поймет
- Но всё же рекомендуется писать правильно:
 - Всегда закрывать теги
 - Писать теги в нижнем регистре
 - Ставить кавычки вокруг значений атрибутов
- Потому что разные браузеры могут вести себя по-разному при ошибках

Структура HTML документа

- Демонстрация – открыть какой-нибудь сайт, например, сайт курсов и исследовать структуру
- Заметим, что HTML не отвечает за то, как именно должен отображаться документ. HTML отвечает только за структуру и содержимое документа
- За отображение документа отвечает другая технология – **CSS (Cascading Style Sheet)**
- Хотя, в HTML есть теги и атрибуты, которые отвечают за отображение элементов, но они являются deprecated – вместо них следует использовать CSS

Теги, отвечающие за стиль

- Рассмотрим некоторые из этих deprecated тегов
 - `Жирный текст`
 - `<i>Курсив</i>`
 - `Текст`
 - И др.
- Вместо всего этого нужно использовать CSS

Инструменты разработчика в браузере

- Все браузеры имеют встроенные средства анализа структуры страницы и отладки JavaScript кода
- Рекомендую пользоваться средствами отладки Google Chrome, т.к. они наиболее удобны
- Чтобы активировать их, надо нажать F12
- Еще неплохие средства отладки в FireFox. Для FireFox еще рекомендуется плагин FireBug

Еще про HTML

- Теги могут вкладываться друг в друга
- В некоторые теги нельзя ничего вкладывать, в некоторые можно
- Все элементы делятся на 2 группы – **блочные** и **строчные**:
 - **Блочные элементы** начинаются с новой строки и следующий за ними элемент начинается с новой строки. Пример – абзац, div
 - **Строчные элементы** начинаются с этой же строки и следующий за ними элемент также будет на этой же строке (естественно, если соседи этого элемента тоже строчные элементы). Пример – ссылка, span
- Многострочный комментарий: `<!-- -->`

Основные теги

- Абзац: `<p>Содержимое</p>`
- Блок: `<div>Содержимое</div>`
- Строчный элемент: `Содержимое`
- Картинка: ``
- Перевод строки: `
`
- Горизонтальная линия: `<hr />`
- Ссылка: `Ссылка`
- Заголовки h1-h6: `<h1>Заголовок 1</h1>`
 - Чем больше номер, тем меньше размер заголовка
 - Обычно на странице один заголовок h1

Ссылки

- Ссылки обозначаются тегом `a`
- Обязательно нужно задать атрибут `href`, который содержит адрес, куда нужно перейти
- `Ссылка на Google`
- Ссылка будет открываться в текущей вкладке. Можно кликнуть колесом мыши, тогда откроется в новой вкладке
- Можно сделать чтобы ссылка всегда открывалась в новой вкладке, надо задать значение атрибута `target="_blank"`
- ``
Ссылка на Google откроется в новой вкладке
``
- У атрибута `target` есть и другие значения, см. документацию

Ссылка на элемент страницы

- Любому элементу можно задать атрибут `id`
- Это уникальный идентификатор элемента, он должен быть уникальным по всей странице
- `<p id="myElement">Первый параграф</p>`
- Ссылке в качестве адреса можно задавать идентификатор элемента в таком виде (решетка # обязательна):
- `К первому параграфу`
- Тогда при клике по этой ссылке браузер перескочит к элементу **myElement**. При этом в адресной строке в конце добавится `#myElement`
- Этот прием используется для создания навигации

Особые ссылки

- Есть возможность делать ссылки на почтовые адреса
- При клике по ним открывается mail клиент
- Для этого надо в адресе указать mailto:
- `test@demo.com`
- Также можно сделать ссылку на телефон:
- `Позвоните нам`

Списки

- Списки бывают **упорядоченные ol** и **неупорядоченные ul**
- Элементами списков являются элементы **li** (list item)

- ****

Coffee

Tea

Milk

• Coffee

• Tea

• Milk

- ****

Coffee

Tea

Milk

1. Coffee

2. Tea

3. Milk

Таблица

- `<table border="1">`

`<tr>`

`<th>Столбец 1</th>`

`<th>Столбец 2</th>`

`</tr>`

`<tr>`

`<td>Ячейка 1</td>`

`<td>Ячейка 2</td>`

`</tr>`

`</table>`

Столбец 1	Столбец 2
Ячейка 1	Ячейка 2

- Внутри тега `table` могут быть только строки `tr` и некоторые другие теги
- Внутри `tr` – только теги ячеек `th` и `td`.
`th` – это ячейка - заголовок

Таблица

- ```
<table border="1">
 <thead>
 <tr>
 <th>Столбец 1</th>
 <th>Столбец 2</th>
 </tr>
 </thead>
 <tbody>
 <tr>...</tr>
 <tr>...</tr>
 </tbody>
</table>
```

Столбец 1	Столбец 2
Ячейка 1	Ячейка 2

- Часто заголовочную строку таблицы оборачивают в **thead**, а обычные строки пишут внутри **tbody**

# Таблица

- ```
<table border="1">  
  <tr>  
    <th colspan="2">Столбец 1</th>  
  </tr>  
  <tr>  
    <td>Ячейка 1</td>  
    <td>Ячейка 2</td>  
  </tr>  
</table>
```

Столбец 1	
Ячейка 1	Ячейка 2

- Можно объединять ячейки при помощи атрибутов ячеек **colspan** и **rowspan**
- **colspan** – объединяет ячейки по горизонтали, а **rowspan** – по вертикали

Практика

- Создать страницу следующего вида:

Виды браузеров

В таблице перечислены основные браузеры и их движки для отрисовки содержимого и исполнения JavaScript:

Движок	Разработчики	Браузер
Blink	Google	Google Chrome
	Opera	
	Samsung	Opera 15+
	Intel	
Gecko	Netscape/Mozilla Foundation	Mozilla Firefox
EdgeHTML	Microsoft	Microsoft Edge

[Больше информации на сайте Wiki](#)

Формы

- Любое сложное приложение должно уметь передавать данные на сервер
- Для этого есть 2 варианта: **формы** и **AJAX (асинхронный JavaScript)**
- Формы позволяют ввести данные, а потом отправить их на сервер по нажатию кнопки
- А сервер уже может сохранить или обработать эти данные

Формы

- Все элементы формы должны находиться внутри тега

`<form>`

У формы есть 2 важных атрибута:

`action` – url к обработчику и `method` – `get/post`

- `<form action="/handler.php" method="post">`
 `<label for="clientName">Ваше имя</label>`
 `<input type="text" id="clientName" name="clientName" />`

`<button type="submit">Отправить заявку</button>`
`</form>`

Ваше имя

Отправить заявку

Для каждого элемента важно заполнить атрибут `name` – по этому `name` сервер сможет получить переданные данные

Элементы input

- Для большинства элементов формы используется тег `<input>` с разным атрибутом `type`
 - Текстовое поле: `<input type="text" />`
 - Радио-баттон: `<input type="radio" />`
 - Чекбокс: `<input type="checkbox" />`
 - Кнопка отправки формы: `<input type="submit" />`
 - Загрузка файлов: `<input type="file" />`
 - И др.
- <http://htmlbook.ru/html/input/type>

Textarea, select, button

- Для многострочного поля ввода используется `<textarea>`
- <http://htmlbook.ru/html/textarea>
- Для выпадающего списка используется `<select>`
- <http://htmlbook.ru/html/select>
- Для кнопок можно использовать `<button>`, у которого есть атрибут `type`
- <http://htmlbook.ru/html/button>

Label и атрибут for

- Чтобы сделать подпись к элементу управления, лучше использовать элемент `<label>`
- Каждому элементу можно присвоить атрибут `id` – это должно быть имя, уникальное для всей страницы
- Атрибут `id` широко используется в CSS и JavaScript
- Другое его применение – привязка `<label>` и элемента управления – тогда по клику на `label`, фокус ввода перейдет на элемент, это очень удобно

Label и атрибут for

- Другое его применение – привязка `<label>` и элемента управления – тогда по клику на `label`, фокус ввода перейдет на элемент, это очень удобно
- Чтобы сделать привязку, надо задать `id` элементу управления, а элементу `label` добавить атрибут `for` и вписать туда `id` элемента

Ваше имя

Отправить заявку

- `<label for="clientName">Ваше имя</label>`
`<input type="text" id="clientName" name="clientName" />`

Другой вариант привязки


- Другой вариант – поместить элемент управления внутри `label`
- Это часто более удобно, потому что не надо придумывать `id` для элемента
- **Вариант с вложенностью:**
`<label>`
 Ваше имя `<input type="text" name="clientName" />`
`</label>`
- **Вариант с `for`:**
`<label for="clientName">`Ваше имя`</label>`
`<input type="text" id="clientName" name="clientName" />`

Radio button'ы

- Чтобы radio button'ы нормально работали в группе, им надо задать одинаковое значение атрибута **name**
- ```
<input type="radio" value="1" name="myRadio"> 1
```

```
<input type="radio" value="2" name="myRadio"> 2
```

```
<input type="radio" value="3" name="myRadio"> 3
```


- Также каждому radio button'у надо задать атрибут **value**, чтобы можно было понять какой именно radio button выбран. Там можно указать любую строку, у каждой кнопки она должна быть своя
- По умолчанию все кнопки будут не выбраны. Чтобы выбрать вариант по умолчанию, надо задать атрибут **checked="checked"**
- ```
<input checked="checked" type="radio" value="1" name="myRadio">
```
- Этот атрибут также можно использовать для checkbox'ов

Формы

- Демонстрация в каком виде уходят данные с формы для методов GET и POST

AJAX

- **AJAX (асинхронный JavaScript)** – технология для запросов к серверу без перезагрузки страницы
- Очень часто применяется, т.к. полная перезагрузка страницы – это дольше по времени и больше по трафику
- Данные с формы собираются кодом на JS и делается запрос (GET/POST)
- AJAX запросы можно отслеживать во вкладке **Network**, фильтр **XHR**
- Чаще всего данные передаются и приходят в формате JSON

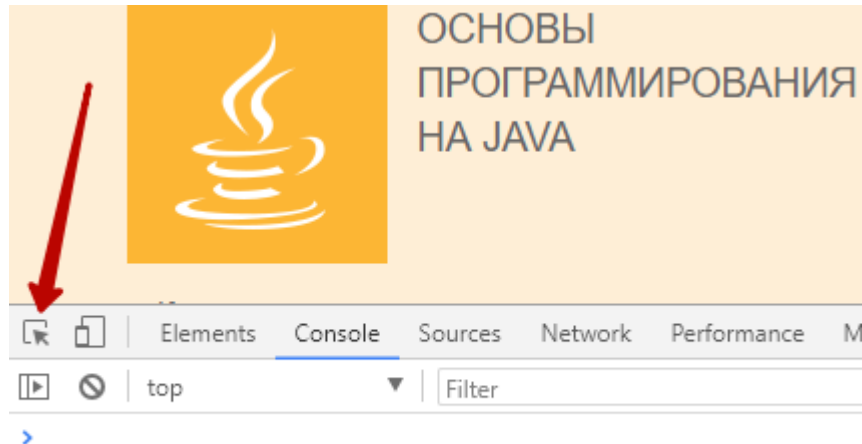
Meta

- Внутри тега **head** можно писать теги **meta**
- Это метаданные страницы, они позволяют задать информацию о странице
- Такая meta используется, чтобы задать кодировку странице, без нее могут быть проблемы с кодировкой:
 - ```
<head>
 <meta charset="utf-8">
</head>
```
- Некоторые **meta** используются для SEO – keywords, description

# Chrome Dev Tools

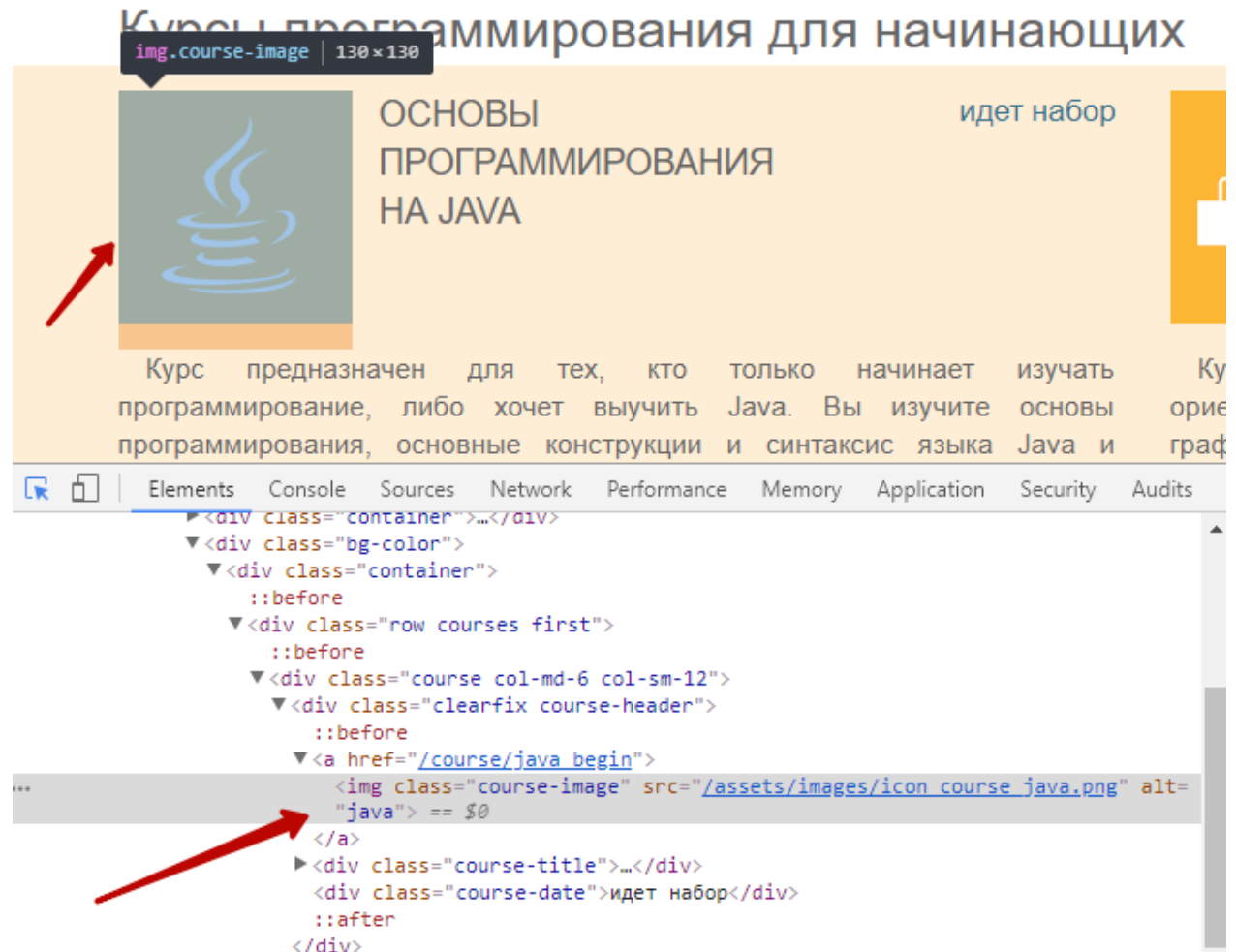
# Выбор элемента

- Одна из важнейших вещей – **выбор элемента**
- Активировать его можно здесь:



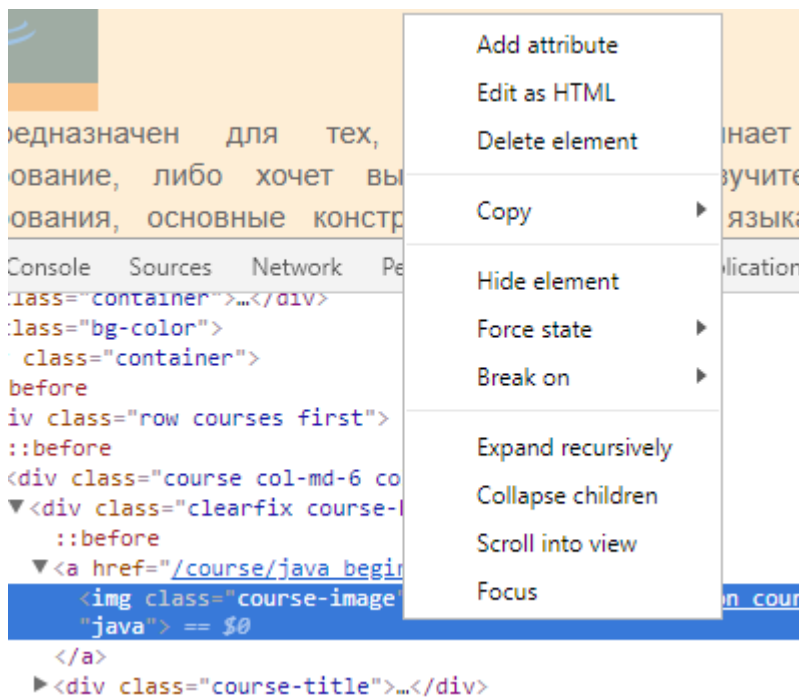
# Выбор элемента

- После этого наводим мышь на нужный элемент, при этом он также подсветится в дереве. Можно кликнуть и выбрать элемент



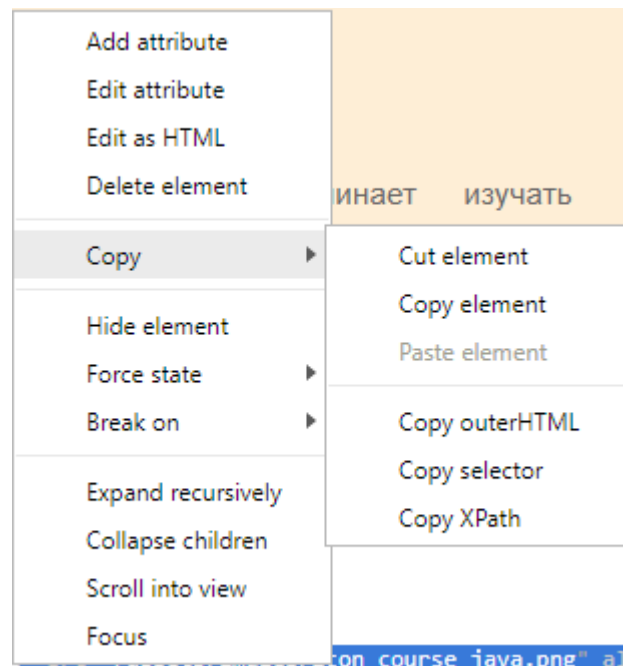
# Выбор элемента

- Можно кликнуть в дереве на элемент правой кнопкой мыши, и вызвать меню



# Пункты меню для элемента

- Самые полезные пункты:
  - **Add attribute** – добавить атрибут
  - **Edit attribute** – изменить атрибут
  - **Edit as HTML** – редактировать разметку
  - **Delete element** – удалить элемент
  - **Copy -> Copy element** – элемент копируется, потом его можно вставить через **Copy -> Paste Element**
  - **Copy -> Copy selector** – скопировать CSS селектор
  - **Copy XPath** – скопировать XPath



# Пункты меню для элемента

- **Copy -> Copy selector** – скопировать CSS селектор
- **Copy XPath** – скопировать Xpath
- Эти пункты (особенно **Copy XPath**) пригодятся вам при автоматизации через **Selenium**





- **CSS (Cascading Style Sheets)** – технология для управления отображением документа (веб-страницы)
- Т.е. позволяет задавать **стили** элементов – цвет, цвет фона, размер шрифта, положение элемента на странице и т.д.
- До появления CSS, для этих целей использовались специальные HTML теги и атрибуты, сейчас они deprecated (нежелательны к использованию)

# Синтаксис CSS

- Файл стилей CSS представляет собой набор правил следующего вида:

- селектор {  
    свойство1: значение1;  
    свойство2: значение2;  
}

- Например,  
p {  
    color: red;  
    font-size: 16px;  
}

**Селектор – это выражение, указывающее, к каким элементам страницы применять правила**

**Правила задаются парами ключ-значение.**

**Ключ отделяется от значения при помощи :  
В конце пары ставится ;**

**Пример задает всем параграфам размер шрифта 16px и красный цвет**

# Как применить CSS

- Есть 3 варианта:
  - **Inline-стили (встроенные стили)**
    - для конкретного элемента конкретной страницы (много дублирования)
  - **HTML-элемент <style>**
    - для всех элементов конкретной страницы (меньше дублирования, но есть)
  - **Подключаемый CSS файл**
    - для всех страниц (нет дублирования)
- Чаще всего используется третий вариант

# Подключаемый CSS

- Для CSS стилей пишется отдельный файл, который подключается на всех страницах, где он нужен
- Подключить CSS-файл можно при помощи элемента `link`, который должен быть помещен в `head`

- **index.html**

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

**style.css**

```
.header {
 font-size: 16px;
 font-weight: bold;
}
```

# Селекторы

- Чтобы применить стили, надо:
  1. При помощи **селектора** указать, к каким элементам их применять
  2. Указать сами стили
- Язык селекторов очень гибкий и позволяет легко выбрать нужные нам элементы
- **Полезная шпаргалка по селекторам CSS:**
- <http://code.tutsplus.com/ru/tutorials/the-30-css-selectors-you-must-memorize--net-16048>

# Селектор по типу элемента

- Самый простой селектор – по типу элемента (по тегу)

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

Сделать текст всех абзацев красным

HTML-атрибуты class и id

- Для «точечного» выбора элементов, CSS может пользоваться двумя стандартными HTML атрибутами, которые могут быть применены почти к любому элементу: **id** и **class**
- **id** – уникальный идентификатор элемента. Обязательно должен быть уникален во всем документе, иначе будут ошибки
- Пример:
- `<h1 id="page-header"></h1>`
- При работе с **Selenium** ключевым элементам страницы разработчики задают атрибут **id** (или **data-test-id**), чтобы было легко найти эти элементы

HTML-атрибуты class и id

- **class** – уже не обязан быть уникальным. Одним классом следует помечать элементы, у которых должны быть одинаковые стили
- **Важное отличие:** в одном атрибуте **class** можно через пробел указывать много классов
- Примеры:
- ```
<h1 class="header"></h1>
<h2 class="header"></h2>
<h3 class="header small"></h3>
```
- Элемент может иметь и **id**, и **class** одновременно
- Оба атрибута - **регистрозависимы**

К h3 тут применено 2 класса  
– header и small

# Селектор по id

- `<h1 id="page-header"></h1>`
- Селектор по id: **# ставится без пробела**
- ```
#page-header {  
    font-size: 20px;  
}
```
- Селектору соответствует только 1 элемент, с этим id

Селектор по class

- `<h2 class="header"></h2>`
`<h3 class="header small"></h3>`

- Селектор по `class`:

. ставится без пробела

- `.header {`
 `font-weight: bold;`
`}`

Результат – оба элемента, у них есть класс `header`

`.small {`
 `font-size: 10px;`
`}`

Результат – 1 элемент, у которого есть класс `small`

Селекторы по атрибуту

- Есть возможность выбирать элементы, у которых есть заданный атрибут
- `<div myAttr="myValue">Text</div>`
- ```
[myAttr] {
 color: red;
}
```
- Но есть и другие варианты этого селектора, которые позволяют проверять значение атрибута

# Селекторы по атрибуту

- Проверка, что у элемента есть такой атрибут, и его значение совпадает с указанным
- `[myAttr="myValue"] {`  
    `color: red;`  
    `}`
- Проверка, что у элемента есть такой атрибут, и его значение содержит указанную строку
- `[myAttr*="myValue"] {`  
    `color: red;`  
    `}`

# Селекторы по атрибуту

- Проверка, что у элемента есть такой атрибут, и его значение начинается с указанной строки
- `[myAttr^="myValue"] {  
 color: red;  
}`
- Проверка, что у элемента есть такой атрибут, и его значение заканчивается на указанную строку
- `[myAttr$="myValue"] {  
 color: red;  
}`

# Селекторы по атрибуту

- Проверка, что у элемента есть такой атрибут, и его значение содержит указанное слово
- Предполагается, что атрибут имеет значение, состоящее из одного или нескольких слов, разделенных пробелами
- ``
- ```
[source~="external"] {  
    color: red;  
}  
[source~="image"] {  
    border: 1px solid black;  
}
```

Комбинирование селекторов

- Все селекторы можно комбинировать (если это имеет смысл)
- Для этого надо написать условия **подряд, без пробела**

Если поставить пробел, то
смысл селектора изменится

- Например, можно делать так:
- `p.wide {`
 ... `// выбрать параграфы с классом wide`
 `}`
- `p[myAttr="value"].wide {`
 ... `// выбрать параграфы с классом wide,`
 `// у которых при этом есть атрибут myAttr, и его`
 `// значение равно "value"`
 `}`

Комбинирование селекторов

- `#some-id.wide {`
 ... `// выбрать элемент с id some-id и классом wide`
 `}`
- `.wide#some-id {`
 ... `// выбрать элемент с id some-id и классом wide`
 `}`
- Эти селекторы равносильны, порядок можно менять
- Порядок нельзя поменять для случаев, когда мы используем селектор по элементу – CSS просто не поймет где кончается класс и начинается тип элемента
- `a.link` – хорошо, `.linka` – искать элементы с классом `linka`
- Обычно, нет смысла комбинировать селектор по id с другими



Селекторы с учетом вложенности

- Можно применять селекторы, основываясь на вложенности элементов друг в друга
- Для этого нужно поставить пробел между селекторами
- ```
<div class="wide">
 <div><p>Параграф 1</p></div>
 <p>Параграф 2</p>
</div>
```
- ```
.wide p {  
  ... // выбрать параграфы, которые лежат на любом  
  // уровне вложенности внутри элементов с классом wide  
}
```
- Селектор выберет оба параграфа

Селектор выбора детей

- Есть селектор, позволяющий выбрать только непосредственных детей
- ```
<div class="wide">
 <div><p>Параграф 1</p></div>
 <p>Параграф 2</p>
</div>
```
- ```
.wide > p {  
  ... // выбрать параграфы, которые лежат  
  // непосредственно внутри элементов с классом wide  
}
```
- Селектор выберет только параграф 2

Примеры

- Еще примеры:
- `.wide > p > a` 
- `.wide p a` 
- `.long.high [src]`
- `#someId p.wide`
- `form.form input[type="text"]`

Селектор +

- `<div>Div</div>`
`<p>Paragraph 1</p>`
`<p>Paragraph 2</p>`
- `div + p {`

`}`
- Такой селектор выберет все параграфы, для которых предыдущий элемент на этом же уровне является div'ом
- Т.е. этот селектор – выбор непосредственных соседей
- В данном примере будет выбран только первый параграф, потому что для второго предыдущим элементом является первый параграф, а не div

Селектор ~

- `<div>Div</div>`
`<p>Paragraph 1</p>`
`<p>Paragraph 2</p>`

Селекторы + и ~ используются редко, но в некоторых случаях очень выручают

- `div ~ p {`

`}`

- Такой селектор выберет все параграфы, для которых некоторый предыдущий элемент на этом же уровне является div'ом
- Т.е. этот селектор – выбор всех соседей, которые идут после указанного элемента
- В данном примере будут выбраны оба параграфа

Copy selector

- Функция **Copy Selector** как раз копирует CSS селектор для элемента

Emmet

- Во многих средах разработки есть специальный синтаксис для быстрого создания HTML элементов – **Emmet (Zen Coding)**
- <https://docs.emmet.io/abbreviations/syntax/>
- По факту Emmet использует язык CSS селекторов + некоторые дополнительные возможности
- Пример: `ul>li*3`
- Потом нажмите Tab, и это выражение превратится в

```
<ul>
  <li></li>
  <li></li>
  <li></li>
</ul>
```


XPath

XPath

- **XPath** – это язык запросов к XML документу
- Он позволяет находить нужные элементы, удовлетворяющие запросу
- Его можно применять к XML и HTML (т.к. синтаксис очень похож на XML)
- <https://ru.wikipedia.org/wiki/XPath>
- <https://msiter.ru/tutorials/xpath>
- <https://msdn.microsoft.com/ru-ru/library/ms256086%28v=vs.120%29.aspx?f=255&MSPPError=-2147217396>

Основные конструкции XPath

- **имяУзла** – выбрать все узлы с именем «имяУзла» (имя тега)
- **/** - выбор непосредственных детей текущего узла
- **//** - выбор потомков любого уровня текущего узла
- **@имяАтрибута** - выбор атрибута
- ***** - любое имя узла
- **[целоеЧисло]** – выбор ребенка по индексу (отсчитываются от 1)

Примеры

- `//*[@id="nav-search"]/form`
 - В любом месте документа (т.к. `//`) найти любые элементы (т.к. `*`) с атрибутом `id`, равным `nav-search`, взять их дочерние элементы `form`
- `//div[@id='main-container']/div[2]/div[2]/div`
 - В любом месте документа найти `div` с атрибутом `id` равным `main-container`, взять у него второй дочерний `div`, у него тоже второй дочерний `div`, у него `div`
- Как видите, если есть `id`, то XPath получается более простой и устойчивый к изменениям разметки

Типичные ошибки при веб-разработке

HTML-инъекция, длина поля

- **HTML-инъекция**
 - В поля ввода можно ввести HTML, отправить данные на сервер, и потом когда эти данные придут с сервера, то этот HTML встроится в страницу
 - <http://testphp.vulnweb.com/search.php>
- **Нет ограничения на длину в полях ввода**
 - Нет ограничения на ввод, и сохранение данных на сервере упадёт, если длина данных превысит некоторую величину

Некроссбраузерность

- Верстка и JavaScript в разных браузерах ведут себя по-разному
- Поэтому надо тестировать на отличающихся браузерах:
 - Chrome
 - Firefox
 - Safari
 - Safari iOS
 - Android
 - IE 8, IE 9, IE 10, Edge

Падение JS

- **Падение JavaScript на странице**
 - Можно открыть средства разработчика (F12), выбрать вкладку Console
 - Красным текстом там идут ошибки, возникающие в JS
 - Некоторые вещи ошибками не являются, но можно обращать на эти вещи внимание
- <https://petrovich.ru/>
- Ошибки в консоли

Частые ошибки

- Не предусмотрен случай, когда нет данных
- Нет лоадера / индикатора загрузки во время долгих операций (при поиске авиабилетов, например)

Задача на дом "HTML"

1. Сделать страницу с формой, в которой будут следующие виды элементов: текстовые поля ввода, многострочные поля ввода, чекбоксы, радио-баттоны, выпадающие списки с единственным и множественным выбором, кнопка
 2. Сделать страницу, в которой есть: картинки, таблица с `colspan` и `rowspan`, абзацы, заголовки, ссылки (пусть одна из них ведет на страницу с формой из п.1), горизонтальные линии и переводы строки и что еще угодно
- Связать эти две страницы с помощью ссылок
 - Подключить файл CSS и сделать минимальное оформление

Задание для саморазвития

- Пройти курс: <https://htmlacademy.ru/courses/4>
- После этого можете проходить другие курсы по HTML и CSS, они бесплатные
- Самое главное – закреплять на практике в своих проектах, иначе ничего не запоминается