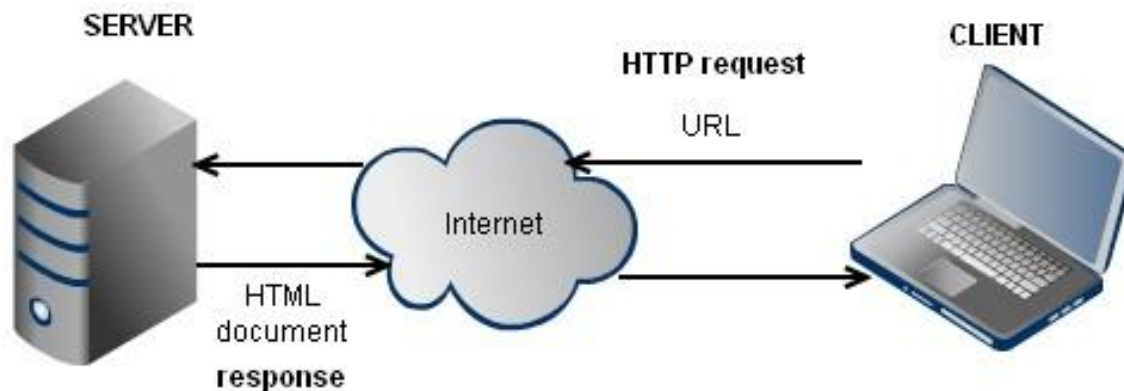


Лекция 1.
Введение в веб-разработку.
Основы HTML

Веб-разработка

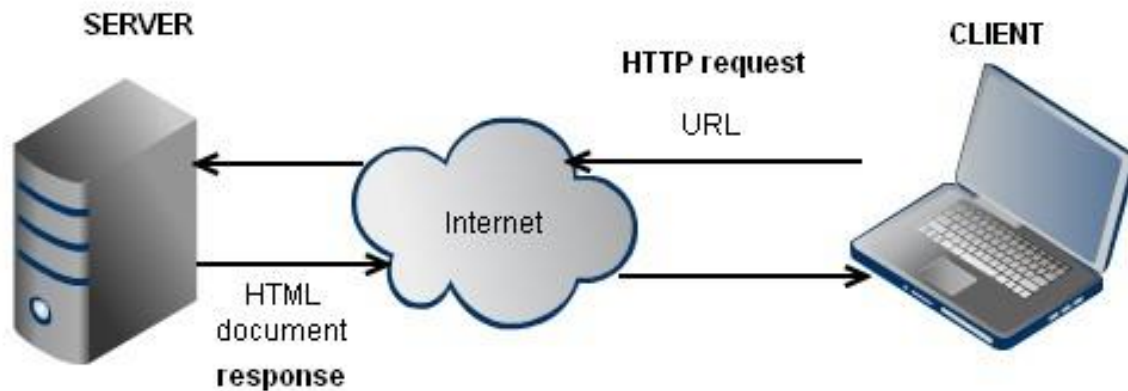
- При веб-разработке приложение состоит из двух частей – **клиентская** и **серверная**
- Есть две взаимодействующие стороны – **клиент** и **сервер**, которые взаимодействуют друг с другом по сети при помощи протокола, например, HTTP
- Сообщения протокола HTTP представляют из себя текст определенного формата. По сути, клиент и сервер обмениваются текстом



Клиентской программой является браузер

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**



Ограничения браузеров

- Возможности браузеров сильно ограничены:
 - Нельзя работать с файловой системой. Но браузеры могут хранить небольшие объемы данных в **cookies** и **local storage**
 - Поэтому все данные хранятся на сервере
 - По сути, нельзя работать ни с чем, кроме самой страницы. Можно работать с разметкой страницы, отправлять запросы на сервер, получать данные. Но никак нельзя повлиять на другие приложения и вкладки

Кросс-браузерность

- Разные браузеры ведут себя по-разному. Как в плане отображения страниц, так и в плане исполнения JavaScript кода
- Поэтому клиентскую часть приложения обязательно нужно проверять на всех целевых браузерах
- **Основные браузеры:**
 - Google Chrome / Yandex Browser
 - Mozilla Firefox
 - Safari
 - IE 10+
 - IE 9-

Проблемы совместимости IE

- Самые большие проблемы совместимости есть у старых IE (IE версии ниже 9.0)
- Старые IE сильно отходили от стандарта, поэтому не поддерживали многие распространенные технологии
- Либо поддерживали, но особым образом. Например, в JavaScript какие-то классы или методы имеют другое имя или ведут себя немного не так, как в других браузерах
- Начиная с IE 9, браузер уже хорошо совместим с остальными
- Но если есть необходимость поддерживать более старые IE, то это обязательно нужно учитывать при разработке
- Сейчас уже обычно поддерживают IE10+

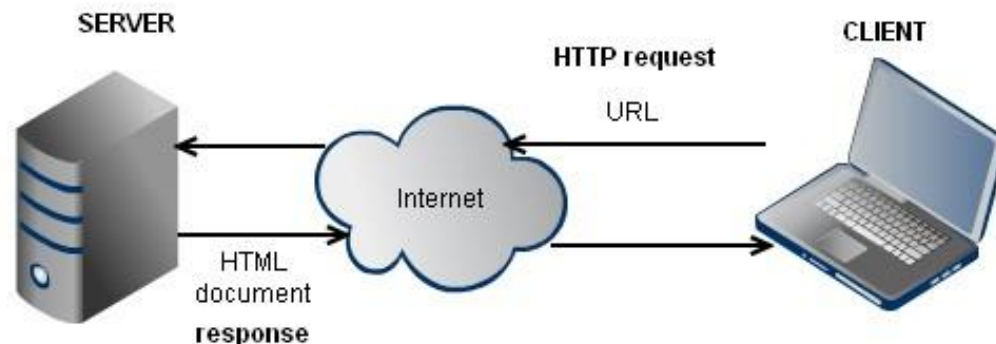


Как в целом все работает?

- На компьютере-сервере устанавливают **серверную часть** приложения
- Серверная часть может быть написана практически на любом языке – Java, C#, PHP, Python, Ruby и т.д.
- Этот код запускается в рамках **веб-сервера** – специальной программы, которая умеет принимать HTTP запросы и отдавать HTTP-ответы

Как в целом все работает?

- Серверная часть умеет "слушать" запросы от клиентов. То есть когда клиент запрашивает какой-то адрес на сервере, то на сервере запускается одна из функций серверной части
- Эта функция получает параметры, которые послал клиент, затем выполняет свой код, который формирует HTTP ответ. По сути – это просто текст. Этот текст, например, может содержать текст HTML страницы
- Браузер получает ответ и отрисовывает переданную страницу



Пример HTTP запроса

- **Пример запроса от браузера:**
- GET / HTTP/1.1 // строка запроса – адрес и версия HTTP
Host: ya.ru // заголовки – пары ключ-значение
Connection: keep-alive
// тело сообщения (тут его нет)
// если тело есть, то оно отделяется одной пустой строкой
- Кстати, в качестве перевода строки используется /r/n

Пример HTTP ответа

- **Пример запроса от браузера:**
- HTTP/1.1 200 OK // версия HTTP, код ответа
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
Cache-Control: no-cache,no-store,max-age=0,must-revalidate
Content-Length: 11369

<!DOCTYPE html><html>...</html> // тело – текст страницы

Клиентская разработка

- Начнем с изучения базовых технологий клиентской разработки: HTML, CSS, JavaScript
- А затем уже будем изучать серверную часть и то, как сделать цельное приложение

- **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

Рекомендация среды разработки

- Для Java рекомендую IDEA Ultimate или WebStorm
- Для C# - Visual Studio

Структура HTML документа

- `<!DOCTYPE html>`

Первая строка – DOCTYPE.
Указывает тип документа

`<html>`

`<head>`

`<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.

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. Про это подробно изучите сами
- 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
 - **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

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`

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

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
```



The image shows three radio buttons arranged horizontally. The first button is labeled '1' in blue, the second is labeled '2' in red and is selected (filled with black), and the third is labeled '3' in blue.
- 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

# 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
- 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

Задача на дом "IntroHtml"

- Пока что можно делать без CSS, сконцентрируйтесь на изучении 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

Задача на дом "TutorialHtml"

- Пройти курс: <https://htmlacademy.ru/courses/4>
- Дома пройдите ещё курсы по HTML:
 - Структура HTML-документа
 - Разметка текста
 - Ссылки и изображения
 - Знакомство с таблицами
 - Знакомство с формами
- **Карта курсов:** https://htmlacademy.ru/learn_map
- Если будет время, проходите оставшиеся HTML курсы от html academy (про HTML 5)