

WEB технологии

О чем этот курс ?

- Общая архитектура WWW
- Протокол HTTP
- Архитектура backend
- Актуальные технологии и ПО
- «Best practices»

Какие знания потребуются ?

- Общие навыки программирование
- Знание языка Python на базовом уровне
- Использование Linux - на уровне пользователя
- Желательно: опыт HTML верстки

Что останется за рамками курса ?

- Сетевые технологии
- Все используемые языки программирования
- Frontend разработка

Internet vs WWW

Internet

Internet - глобальная сеть передачи данных

Протоколы

- **HTTP**, SSH, P2P - прикладные протоколы
- DNS - система имен
- TCP - надежная последовательная передача данных
- IP - глобальная адресация, передача в гетерогенной среде

World Wide Web

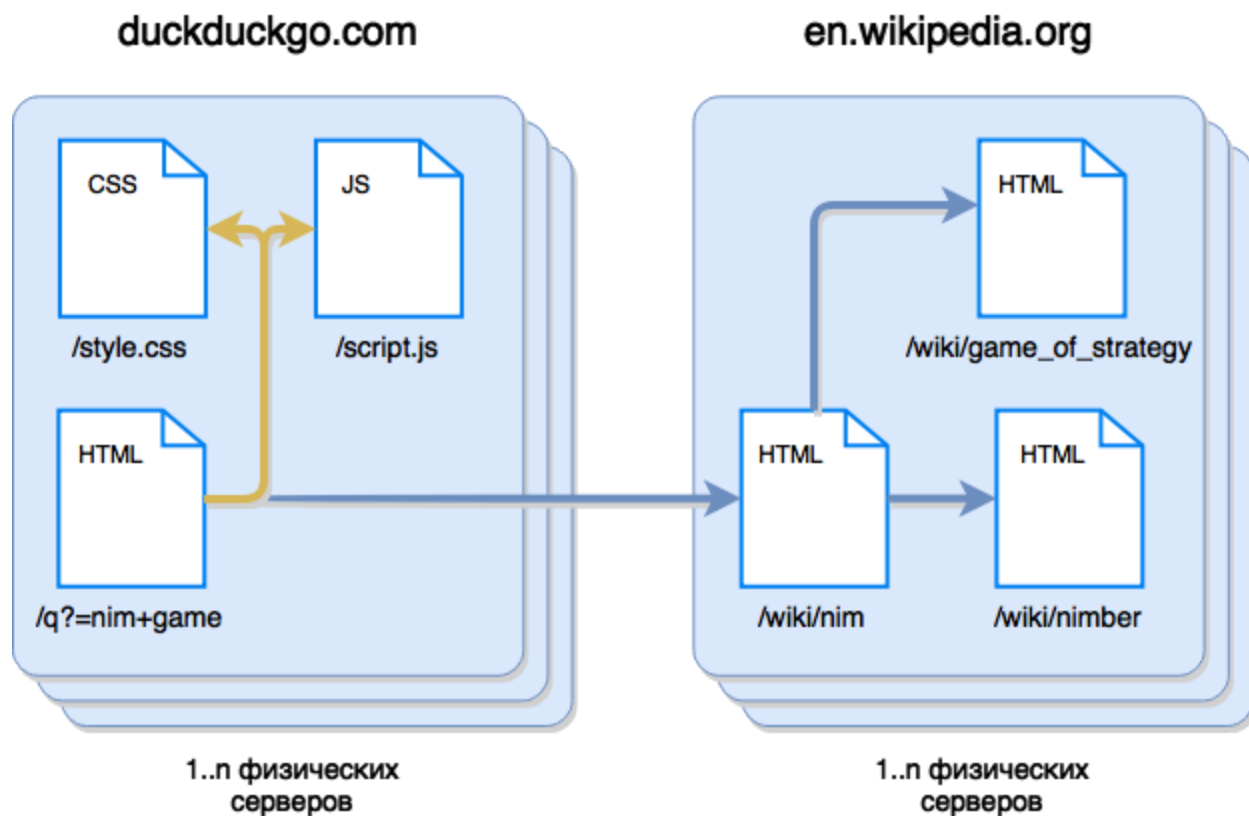
WWW - множество **взаимосвязанных документов**,
располагающихся на машинах подключенных к Internet

WWW - набор протоколов, серверного и клиентского ПО,
позволяющих получать доступ к документам

World Wide Web

WWW - множество **взаимосвязанных документов**,
располагающихся на машинах подключенных к Internet

WWW - набор протоколов, серверного и клиентского ПО,
позволяющих получать доступ к документам



Гиперссылка



Ресурс



Документы

Типы документов (MIME-типы)

- text/html
- text/css
- text/javascript
- image/png
- video/mp4
- text/xml
- application/json
- [Полный список MIME типов](#)

Расширения файлов играют второстепенную роль

text/html

```
<html>
<body>
  <link rel="stylesheet" href="/css/style.css">
  <script src="http://code.jquery.com/jquery-2.1.4.js">
  </script>
  <p>Some text with 
    and <a href="#yes">hyperlinks</a>
  </p>
</body>
</html>
```

text/css

```
.hljs-subst,  
.hljs-title,  
.json .hljs-value {  
  font-weight: normal;  
  color: #000;  
}
```

text/xml

```
<response status="ok">  
  <friends>  
    <friend id="1" name="v.pupkin"/>  
    <friend id="2" name="a.pushkin"/>  
    <friend id="3" name="n.tesla"/>  
  </friends>  
</response>
```

application/json

```
{  
  "status": "ok",  
  "friends": [  
    { "id": 1, "name": "v.pupkin" },  
    { "id": 2, "name": "a.pushkin" },  
    { "id": 3, "name": "n.tesla" }  
  ]  
}
```

Документы могут быть

- **Статические**
 - Это файлы на дисках сервера
 - Как правило, обладают постоянным адресом
- **Динамические**
 - Создаются на каждый запрос
 - Содержимое зависит от времени и пользователя
 - Адрес может быть постоянным или меняться

URL

URL - unified resource locator

```
http://server.org:8080/path/doc.html?a=1&b=2#part1
```

- http - протокол
- server.org - DNS имя сервера
- 8080 - TCP порт
- /path/doc.html - путь к файлу
- a=1&b=2 - опции запроса
- part1 - якорь, положение на странице

Абсолютные и относительные URL

- `http://server.org/1.html` - абсолютный
- `//server.org/1.html` - абсолютный (schemeless)
- `/another/page.html?a=1` - относительный (в пределах домена)
- `pictures/1.png` - относительный (от URL текущего документа)
- `?a=1&b=2` - относительный (от URL текущего документа)
- `#part2` - относительный (в пределах текущего документа)

Правила разрешения URL

`https://site.com/path/page.html` - основной документ

+ `http://wikipedia.org` = `http://wikipedia.org`

+ `//cdn.org/jquery.js` = `https://cdn.org/jquery.js`

+ `/admin/index.html` = `https://site.com/admin/index.html`

+ `another.html` = `https://site.com/path/another.html`

+ `?full=1` = `https://site.com/path/page.html?full=1`

+ `#chapter2` = `https://site.com/path/page.html#chapter2`

Как документы
могут ссылаться
друг на друга?

HTML - гиперссылки

Список товаров в `корзине`

Список товаров в [корзине](#)

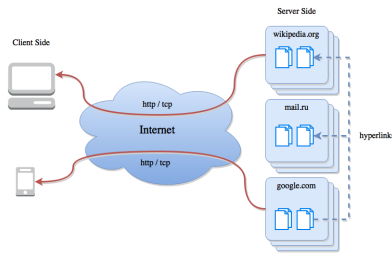
HTML - формы

```
<form action="https://duckduckgo.com/">  
  <input type="text" name="q" value="">  
  <input type="hidden" name="ia" value="images">  
  <button type="submit">Найти</a>  
</form>
```

HTML - ресурсы

```
<link rel="stylesheet" href="/css/index.css">  
<script src="http://code.jquery.com/jquery-2.1.4.js">  
</script>  

```



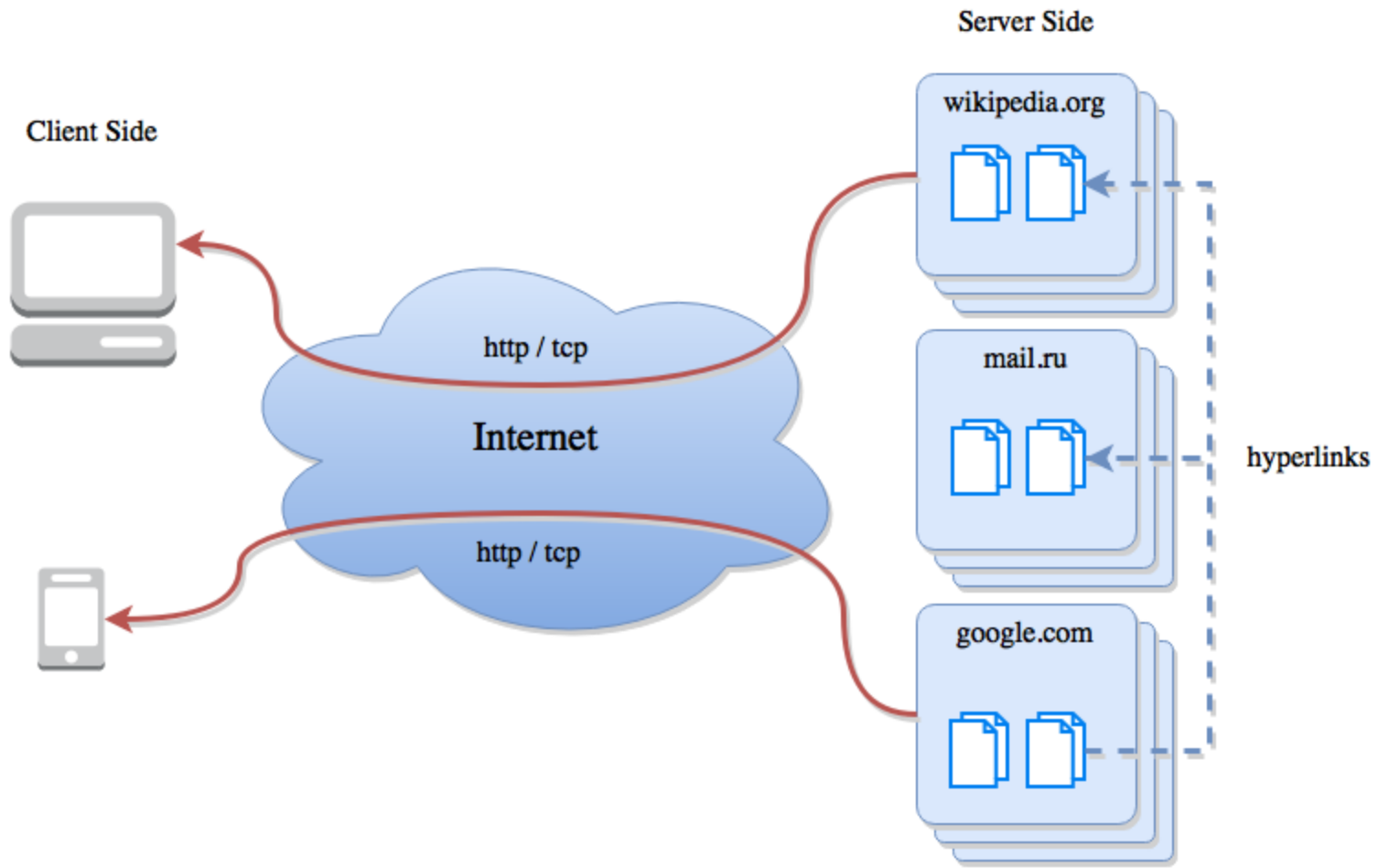
CSS - ресурсы

```
.slide {  
    background-image: url(../pictures/network.png)  
}  
@font-face {  
    font-family: Terminus;  
    src: url(fonts/terminus.ttf);  
}
```

JavaScript - прямое указание URL

```
var saveApiUrl = '/items/save/';  
var newTitle = 'Duck tales';  
$.ajax({  
    type: 'POST',  
    url: saveApiUrl,  
    data: { id: 10, title: newTitle }  
});
```

Клиент
серверная
архитектура



Клиент-серверная архитектура

Web-клиенты работают на компьютерах конечных пользователей.

Задача Web-клиентов состоит в получении и отображении документов.

Web-сервера работают (как правило) на серверах в датацентрах.

Их задача заключается в хранении (или генерации) и отдаче документов.

Преимущества подхода

- Открытый протокол
- Стандартный клиент
- Прозрачный способ взаимодействия приложений
- Распределенная и масштабируемая система