

Web клиенты

# Разновидности web-клиентов

- Библиотеки в ЯП: libcurl, urllib, requests и т.д.
- Консольные утилиты: wget, curl, telnet!
- Роботы: поисковики, вредоносные скрипты
- Браузеры:
  - Полноценные: firefox, chrome и т.д.
  - Встроенные: web-view, webkit и т.д.

# Особенности библиотек web-клиентов

- Предоставляют максимум опций для работы с HTTP
- Осуществляют кодирование / декодирование данных
- Перенаправления, куки - опционально

Назначение: используются внутри других программ для простой работы с HTTP

# Пример использования requests

```
# pip install requests
import requests
url = 'https://api.site.com/method/'
params = {'argument1': 'value1',
          'argument2': 'value2'}
headers = {'User-Agent': 'python requests'}
response = requests.get(url, params=params, headers=headers)
response.text
# '{"type": "User", "name": "Pupkin" ...'
response.json()
# {'type': 'User', 'name': 'Pupkin', ...}
```

# Назначение консольных клиентов

- Автоматизация в shell-скриптах
- Создание "статической копии сайта"
- Отладка web-приложений

# Telnet

Telnet - это простейшее средство отладки. telnet открывает tcp соединение и связывает его с консолью, позволяя общаться с web-сервером напрямую с клавиатуры.

```
$ telnet www.ru 80
Trying 217.112.35.75...
Connected to www.ru.
Escape character is '^]'.
GET /index.html HTTP/1.1
Host: www.ru
```

```
HTTP/1.1 404 Not Found
Server: nginx
Date: Wed, 06 Mar 2024 09:17:03 GMT
Content-Type: text/html; charset=utf-8
Transfer-Encoding: chunked
Connection: keep-alive
Keep-Alive: timeout=20
```

# Еще примеры отладки

GET запрос к серверу с отображением всех заголовков:

```
curl -vv 'http://api.site.com/method/?arg=1'
```

POST запрос к серверу с авторизацией и передачей доп.  
заголовков:

```
curl -vv -d 'arg=1' -H 'X-Token: 123'
```

```
'http://api.site.com/method/'
```



```
$ curl -vv 'https://api.site.com/method/?arg=1'
*   Trying 1.2.3.4:443...
* Connected to api.vkplay.ru (1.2.3.4) port 443
> GET /method/?arg=1 HTTP/2
> Host: api.site.com
> User-Agent: curl/8.4.0
> Accept: */*
>
< HTTP/2 200
< server: nginx
< date: Wed, 06 Mar 2024 09:17:03 GMT
< content-type: application/json; charset=utf-8
< content-length: 117
< cache-control: no-store
<
{"status":"ok"}
```

Браузер

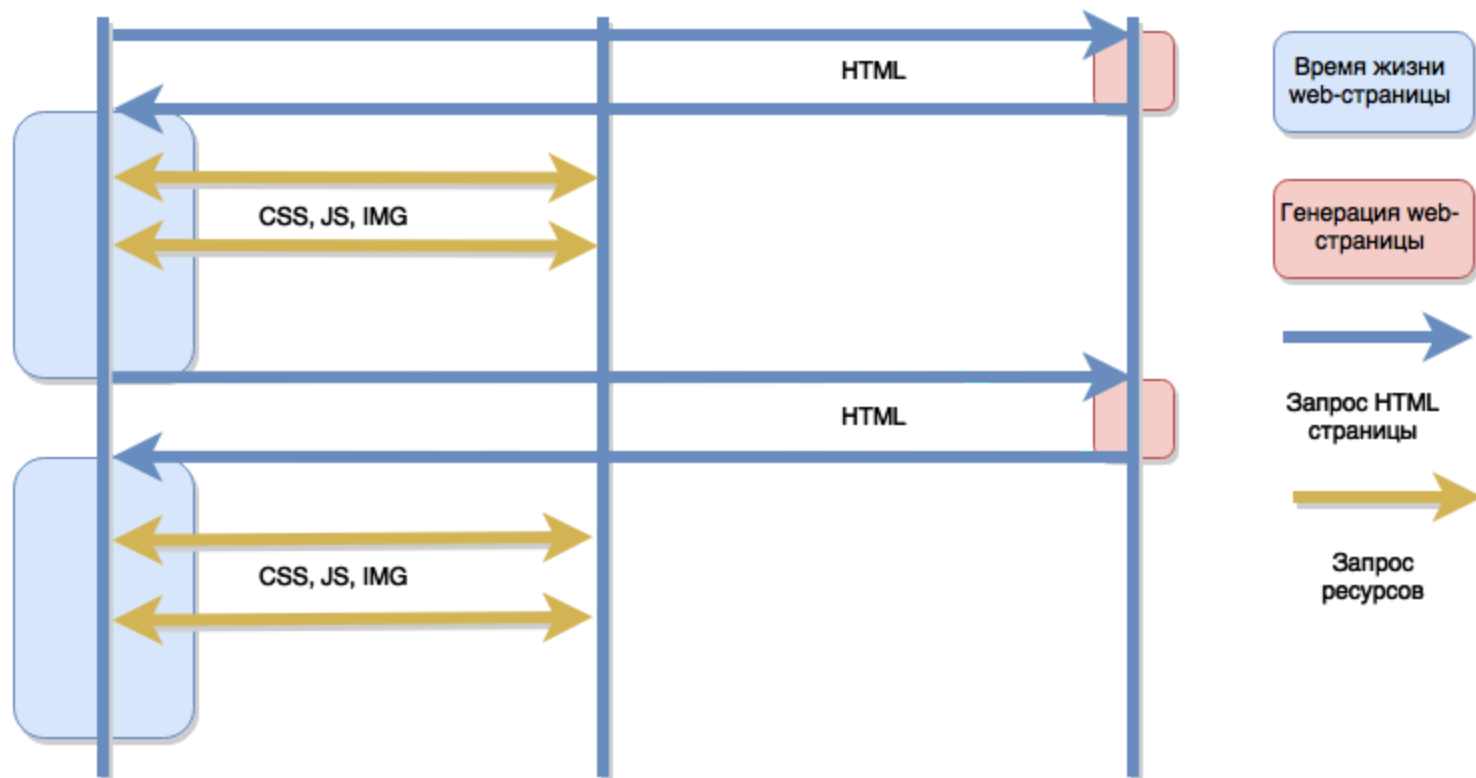
# Сценарий работы web приложения

- Пользователь вводит URL
- Браузер загружает Web страницу - HTML документ
- Браузер анализирует (parse) HTML и загружает доп. ресурсы
- Браузер отображает (rendering) HTML страницу
- Пользователь переходит по гиперссылке или отправляет форму
- Цикл повторяется

**Браузер**

**Web-Сервер**

**Application-сервер**



# Сценарий работы современного приложения

- Браузер загружает Web страницу, ресурсы и отображает ее
- JavaScript загружает данные с помощью AJAX запросов
- JavaScript обеспечивает полноценный UI на странице
- Пользователь взаимодействует с UI, что приводит к вызову JavaScript обработчиков
- JavaScript обновляет данные на сервере или загружает новые данные, используя AJAX

**Браузер**

**Web-Сервер**

**Application-сервер**



# Особенности современных Web-приложений

- UI находится на одной или нескольких страницах (Single Page Application - SPA)
- UI полностью статичен: HTML, CSS, JS - статические файлы
- Логика UI полностью работает на стороне клиента
- Используется шаблонизация в JavaScript
- Application сервер возвращает чистые данные (JSON или XML, а не HTML)