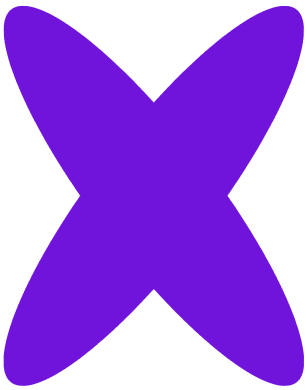
ВЗАИМОДЕЙСТВИЕ С СЕРВЕРОМ

Алексей Опалев




НА ПОРТАЛЕ
ОТМЕТИТЬСЯ ВАЖНО
imgflip.com

ПЛАН ЛЕКЦИИ

1. TLDR [что нужно сделать дома]
2. Компьютерные сети 101
3. Взаимодействие клиента с бэкендом
4. RTU
5. React
6. Домашнее задание

ЧТО ДЕЛАТЬ ДОМА (ДЗ)

ОБНОВЛЕНИЕ ДАННЫХ

Необходимо:

- Создать отдельный "общий" чат
- Передавать новые сообщения на бэк
- Получать новые сообщения с бэкенда в режиме реального времени (RTU)
- Можно использовать любую из технологий для RTU (polling, sse, ws)

КАК ЭТО СДЕЛАТЬ В КОДЕ

КЛАССОВЫЙ КОМПОНЕНТ

```
class SomeComponent extends React.Component {
  state = {
    // state declaration
    messages = []
  }

  componentDidMount () => {
    this.getMessages()
  }

  getMessages = () => {
    fetch(`${API_URL}`)
      .then(res => res.json())
      .then(data => {
        console.log(data);
        this.setState({
          messages: data.messages
        });
      });
  };
}
```

ФУНКЦИОНАЛЬНЫЙ КОМПОНЕНТ

```
import { useEffect, useState } from React;

const SomeFunctionalComponent = () => {
  const [messages, setMessages] = useState([])
  useEffect(() => {
    fetch(`${API_URL}`)
      .then(res => res.json())
      .then(data => {
        console.log(data);
        setMessages(data.messages)
      });
  }, []);
}
```

ОБНОВЛЕНИЕ В "РЕАЛЬНОМ ВРЕМЕНИ"

```
const pollItems = () => {
  fetch(`${API_URL}?key=value&another_key=another_value`)
    .then(resp => resp.json())
    .then(data => console.log(data));
}

const t = setInterval(() => pollItems(), 3000);

// clearInterval(t)
```

СЕРВЕР С ДАННЫМИ (NODEJS)

Репозиторий: **https://github.com/track-mail-ru/tt-front-server**

Роутинг: **https://github.com/track-mail-ru/tt-front-server/blob/master/server.js**

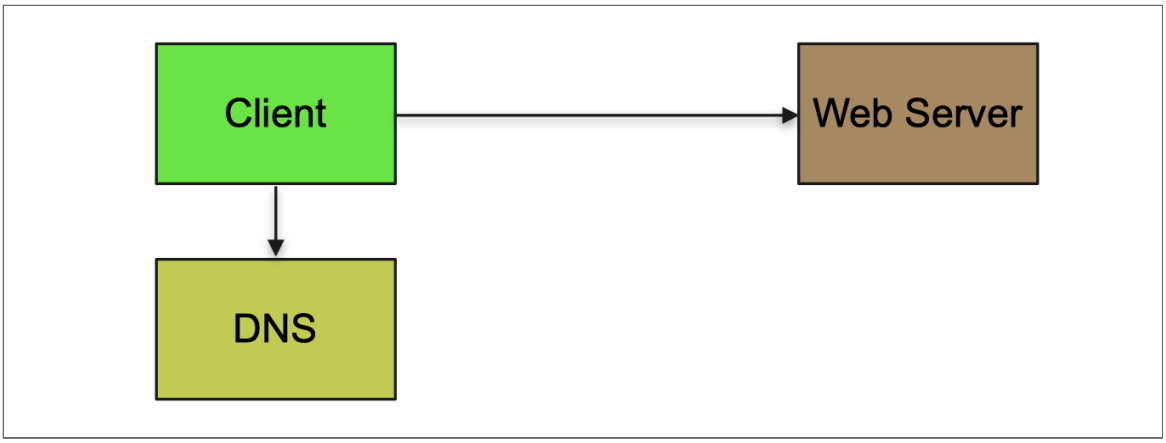
ПРИ ВЫПОЛНЕНИИ ДЗ

1. обращаться к локальному бэкенду до тех пор, пока не будет задеплоен бэкенд.
2. Сообщения необходимо отправлять с разных клиентов/вкладок.
3. При сдаче дз приложить гиф/видео с обновлением сообщений в общем чате.

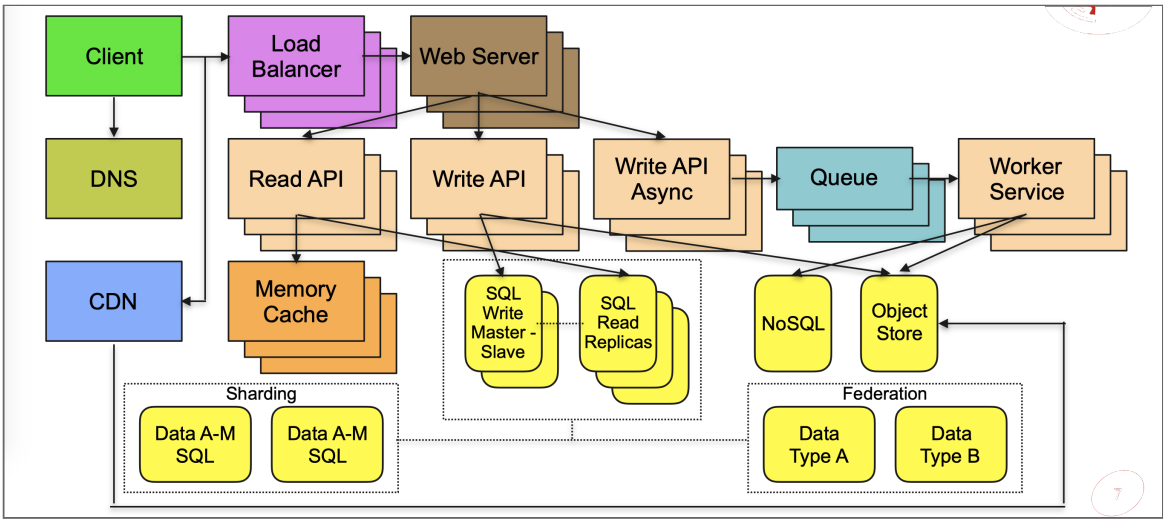
АРХИТЕКТУРА ВЕБ ПРИЛОЖЕНИЯ

ГЛАЗАМИ ОБЫЧНОГО ФРОНТЭНДЕРА

ГЛАЗАМИ ОПЫТНОГО ФРОНТЭНДЕРА



НА САМОМ ДЕЛЕ



Читать подробнее: **The System Design Primer**

СЕТЬ 101

Что такое модель сети TCP/IP, OSI

МОДЕЛЬ СЕТИ OSI

Уровень	Физический	Канальный	Сетевой	Транспортный	Сеансовый	Представления	Прикладной
Описание	Передача бинарных данных (кабель, радиочастота)	Физическая адресация между двумя объектами	Адресация, определение маршрута, контроль траффика	Надежная передача данных между несколькими участниками	Поддержание сеанса связи	Сжатие, шифрование, представление данных	Высоко-уровневые апи
Протоколы	802.15 809.11 GSM	802.3 PPP ARP	IP RIP OSPF	TCP UDP	L2TP SOCKS RPC	MIME TLS	HTTP WebSocket

TCP

Transmission Control Protocol

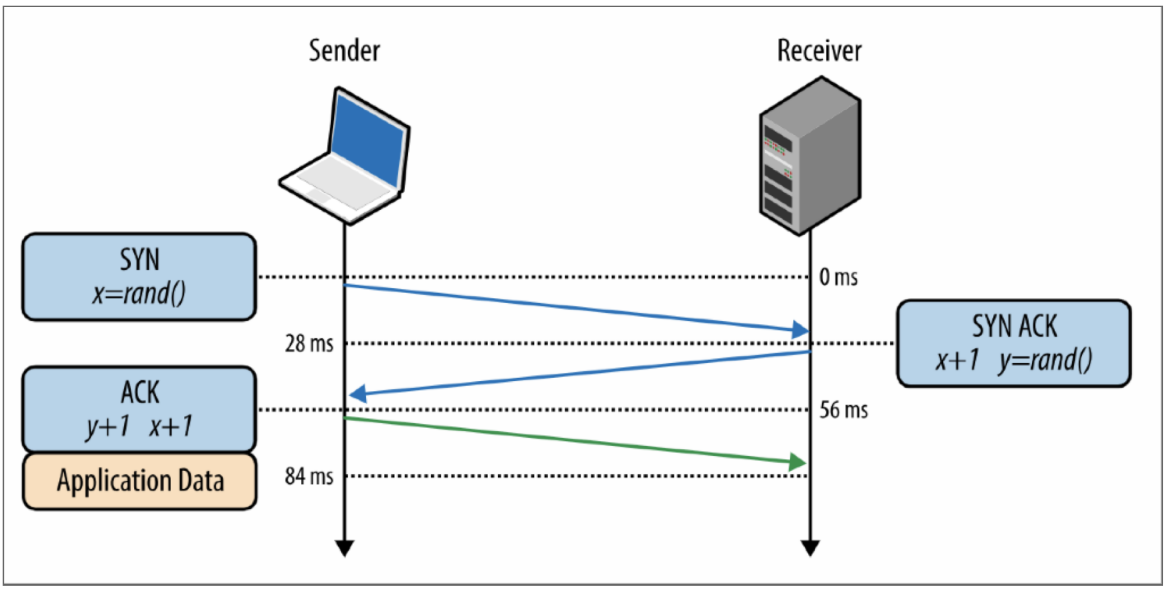
RFC 793: **https://tools.ietf.org/html/rfc793**

Протокол TCP – абстракция надежной передачи данных в сети (ненадежной среде)

ГЛОССАРИЙ

- Basic Data Transfer – передача потока данных между отправителем и получателем
- Reliability – восстановление данных после повреждения, сохранение порядка пакетов
- Flow Control – установка получателем ограничения на объем получаемых данных
- Multiplexing – установка портов для обслуживания процессов (создание сокетов)
- Connections – установленное соединение между двумя сокетами
- Precedence and Security – возможность установить уровень безопасности и приоритет для соединения

Установка TCP соединения



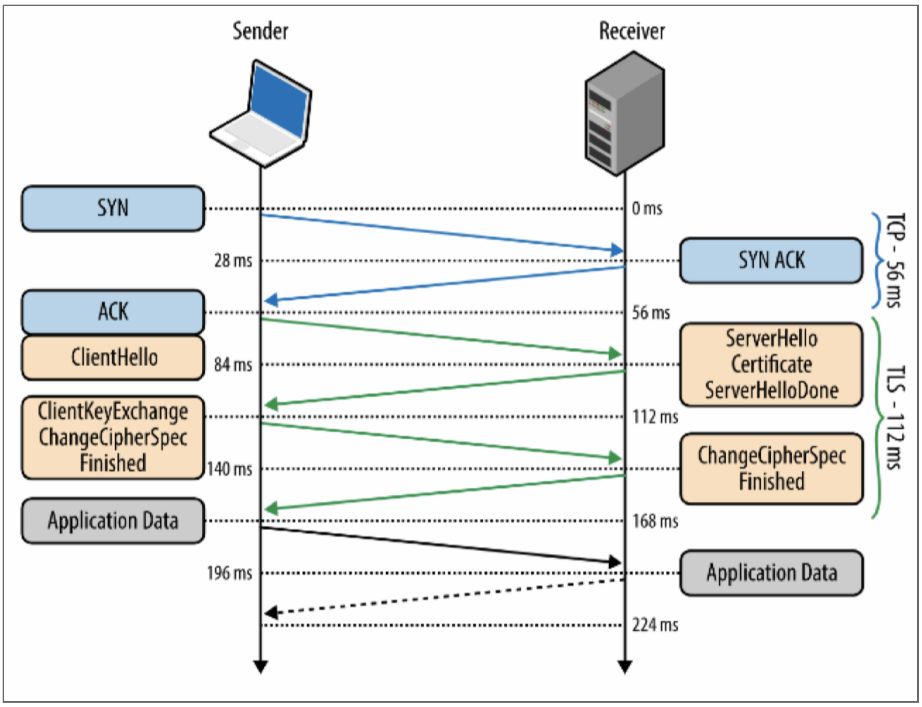
TLS

Transport Layer Security

RFC 5256:
https://tools.ietf.org/html/rfc5246

Протокол TLS – механизм безопасной передачи данных в сети Интернет

TLS handshake



HTTP

HTTP 1.1 RFC 7231:
https://tools.ietf.org/html/rfc7231

HTTP 2 RFC 7540:
https://tools.ietf.org/html/rfc7540

ЛУЧШИЕ ПРАКТИКИ HTTP

- Сократить количество запросов к DNS
- Сократить количество TCP соединений
- Сократить количество редиректов
- Снизить RTT
- Избавиться от ненужных ресурсов
- Кэшировать ресурсы на клиенте
- Сжимать данные при передаче
- Не запрашивать ненужные ресурсы
- Утилизировать обработку запросов и ответов
- Использовать оптимизации соответсвующие протоколу (http1.1 / http2)

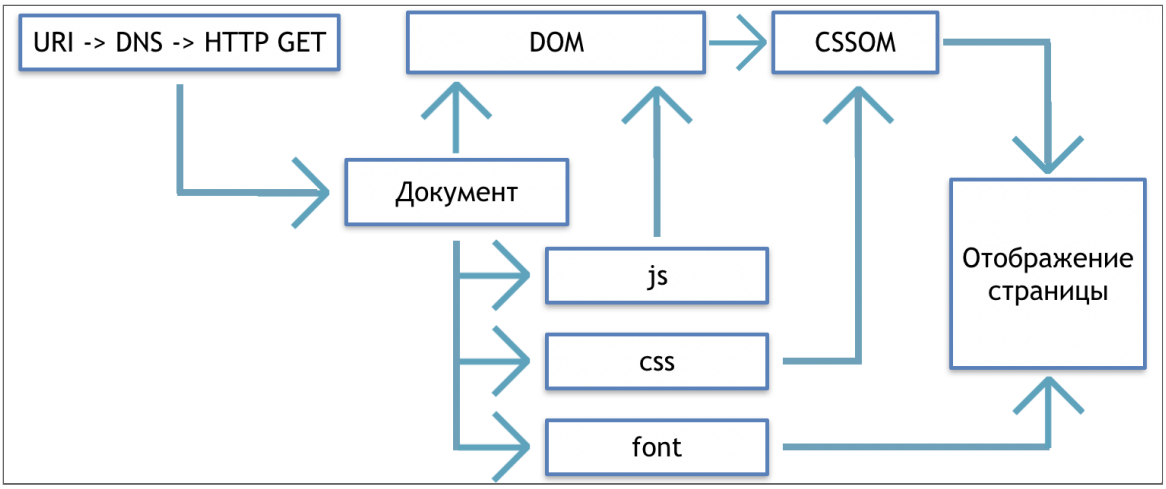
ПОМНИТЬ ПРО МОБИЛЬНЫЕ СЕТИ

При работе с мобильными устройствами необходимо беречь заряд устройства

ПРИМЕР

- не использовать polling
- агрегировать запросы на клиенте
- агрегировать ответы на сервере
- снижение активности скриптов аналитики

ПОРЯДОК ЗАГРУЗКИ СТРАНИЦЫ



Читать больше: **Ilya Grigorik: Analyzing Critical Rendering Path Performance**

ВКЛЮЧЕНИЕ РЕСУРСОВ

JS

```
<script async />  <!-- не блокирует DOM, выполнится как можно ско
<script defer />  <!-- выполнится после построения DOM -->
```

```
<script>
  var script = document.createElement('script');
  script.addEventListener('load', (event) => {});
  script.src = '/dist/main.js';
  document.head.appendChild(script);
</script>
```

Script Tag - async & defer:
https://stackoverflow.com/a/39711009/39841

CSS

```
<link href="/dist/touch.css" rel="stylesheet" media="(max-wi
<link href="landscape.css" as="style" rel="preload" media="(

@import "common.css" screen;
@import url('landscape.css') screen and (orientation:landsca
```

ПРИОРИТЕТ И ПРЕДЗАГРУЗКА

НИЗКИЙ ПРИОРИТЕТ

```
<link rel="prefetch">
```

Экраны, которые понадобятся позже

ВЫСОКИЙ ПРИОРИТЕТ

```
<link rel="preload" as="script">
```

Текущий экран

RESOURCE HINTS

```
<link rel="dns-prefetch" href="https://dobro.mail.ru>"
<link rel="preconnect" href="https://pets.mail.ru/external-s
<link rel="prefetch" href="https://pets.mail.ru/news" as "do
<link rel="prefetch" href="https://pets.mail.ru/news/?page=2
<link rel="prerender" href="https://pets.mail.ru/news/?page=
```

Читать больше:

- **Ilya Grigorik: Preconnect, prerender, prefetch [slides]**
- **w3.org: resource hints**
- **Ivan Akulov: Preload, prefetch and other tags Перевод статьи на хабре**

СОБЫТИЯ ЗАГРУЗКИ

DOMCONTENTLOADED

- Блокирующие ресурсы загружены
- Построение DOM завершено

```
document.addEventListener("DOMContentLoaded", (event) => {cc
```

https://developer.mozilla.org/en-US/docs/Web/API/Window/DOMContentLoa

LOAD

- Все внешние ресурсы загружены
- Построение CSSOM завершено

```
window.addEventListener("load", (event) => {console.log('Loa
```

https://developer.mozilla.org/en-US/docs/Web/API/Window/load_event

UNLOAD/BEFOREUNLOAD

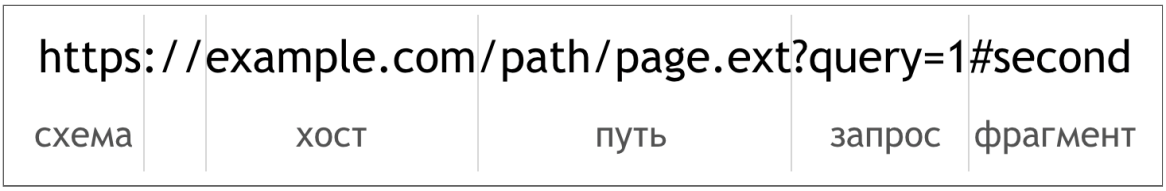
```
window.addEventListener("beforeunload", (event) => {
  event.preventDefault();
  event.returnValue = '';
});
```

https://developer.mozilla.org/en-US/docs/Web/API/Window/beforeunload_ev

https://developer.mozilla.org/en-US/docs/Web/API/Window/unload_event

URI

Uniform Resource Identifier



- window.location
- window.URL

```
Object.getOwnPropertyNames(window.location)
```

- protocol, hostname, pathname, search, hash
- port, host, searchParams, password, username
- href

```
const myUrl = new URL('https://example.com/path/page.ext?que
window.location.assign(myUrl);  // переход на новую страницу
```

SAME-ORIGIN POLICY

https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy

ORIGIN

protocol host port

- `https` – protocol
- `mail.ru` – host
- `443` – port

https://en.wikipedia.org/wiki/Same-origin_policy#Origin_determination_rules

`https://news.mail.ru`

Origin	Result
`https://news.mail.ru/politics/`	**OK**
http://news.mail.ru	**Fail**
https://auto.mail.ru	**Fail**

SUBDOMAIN

Повышение уровня домена

```
document.domain = 'example.com';
```

`https://news.mail.ru` -- `https://mail.ru`

РАЗНЫЕ ИСТОЧНИКИ

- Можно отправлять запросы и встраивать контент
- Нельзя программно читать ответы
- Нет доступа к DOM и свойству location

IFRAME

Заголовок: `X-Frame-Options`

```
X-Frame-Options: deny
X-Frame-Options: allow-from https://example.com/
```

CORS

Cross-Origin Resource Sharing

Техника, управления доступом к ресурсам из внешних источников

https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

- Access-Control-Allow-*
- Access-Control-Request-*

```
Access-Control-Allow-Origin: https://example.com
```

CREDENTIALS

Реквизиты доступа

Не будут отправлены по-умолчанию

Будут отправлены только с разрешения источника

CORS: ПРОСТЫЕ ЗАПРОСЫ

МЕТОДЫ

- GET
- POST
- HEAD

ЗАГОЛОВКИ

- Accept
- Accept-Language
- Content-Language
- Content-Type
  - application/x-www-form-urlencoded
  - multipart/form-data
  - text/plain

REQUEST

```
GET /public HTTP/1.1
Host: my-example.com
Origin: https://example.com
```

RESPONSE

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
```

CORS: PREFLIGHT

непростые запросы

- Отличается метод
- Присутствуют другие заголовки
- Другой Content-Type

Request [options]

```
OPTIONS /public HTTP/1.1
Host: my-example.com
Access-Control-Request-Method: POST
Access-Control-Request-Headers:
X-CSRF-Token, Content-Type
```

Response

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: https://example.com
Access-Control-Allow-Methods: POST, GET, OPTIONS
Access-Control-Allow-Headers:
X-CSRF-Token, Content-Type
Access-Control-Max-Age: 600
```

Request [post]

```
POST /public HTTP/1.1
Host: my-example.com
X-CSRF-Token: some-value
Origin: https://example.com
```

Response

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: https://example.com
```

CORS: CREDENTIALS

РЕКВИЗИТЫ ДОСТУПА

- Cookie
- Authorization
- TLS-сертификат

ОСОБЕННОСТИ

- Ответ на простой запрос нельзя прочесть
- Целевой запрос не будет отправлен, если в preflight нет разрешения

Request

```
POST /public HTTP/1.1
Host: my-example.com
Cookie: key=some-value
Origin: https://example.com
```

Response

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: https://example.com
Access-Control-Allow-Credentials: true
```

What exactly does the Access-Control-Allow-Credentials header do?

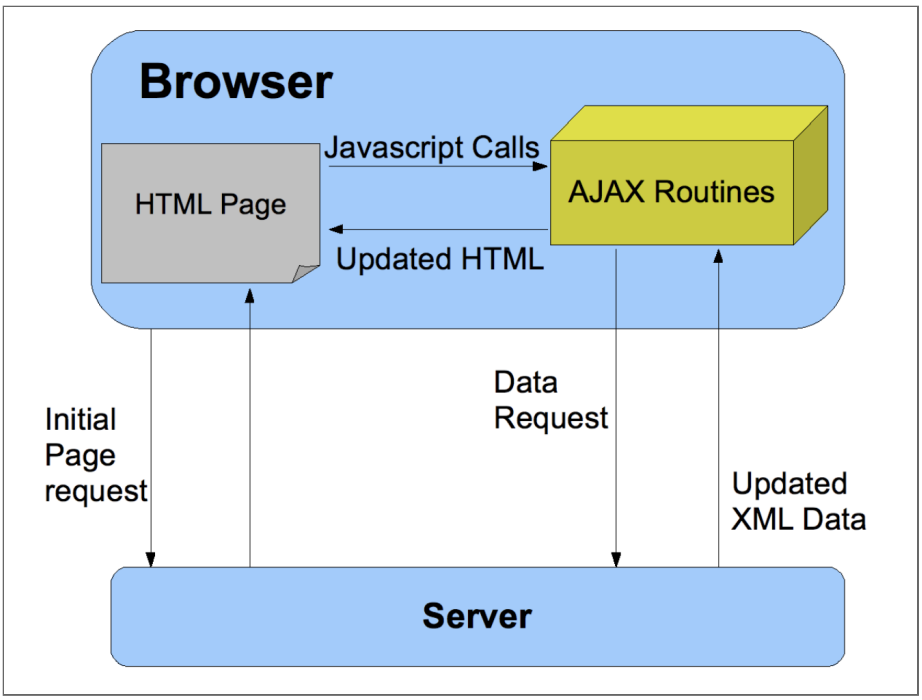
https://stackoverflow.com/a/24689738/3984

ВЫПОЛНЕНИЕ ЗАПРОСОВ

	Встроенное поведение	Обработка результата	Отмена запроса
Переход между страницами	Да	Нет *	Нет
Отправка формы	Да	Нет *	Нет
Добавление ресурса	Да *	Да *	Нет
XHR/Fetch	Нет	Да	Да *

XMLHTTPRREQUEST

XMLHttpRequest и AJAX – «исторические» названия



OpenCart – Create Ajax Function

```
const myRequest = new XMLHttpRequest;
myRequest.addEventListener('readystatechange', (event) => {
    if (myRequest.readyState !== XMLHttpRequest.DONE) {
      return;
    }
    if (myRequest.status === 200) {
        console.log(JSON.parse(myRequest.responseText));
    } else {
        console.log(myRequest.responseText);
    }
});
myRequest.open('GET', '/test.json', true);
myRequest.send();
```

```
const myRequest = new XMLHttpRequest;
XMLHttpRequest.prototype.readyState;
// 0  UNSENT
// 1  OPENED
// 2  HEADERS_RECEIVED
// 3  LOADING
// 4  DONE

XMLHttpRequest.prototype.onreadystatechange = () => {};

XMLHttpRequest.prototype.response/responseText

XMLHttpRequest.prototype.status/statusText
```

Читать больше про XHR:

- **javascript.info: XMLHttpRequest**
- **MDN: XMLHttpRequest**
- **MDN: Using XMLHttpRequest**

FETCH

- Считается удобной заменой XHR
- Возвращает Promise
- Отдельные объекты запроса, ответа и заголовков
- 404, 500, etc – вернут fulfilled Promise

```
const myRequest = fetch('/test.json');
myRequest.then(
    response => response.ok && response.json()
).then(console.log);
```

В качестве аргумента принимает строку или объект Request

```
fetch(input[, init]);
```

ОБЪЕКТ ИНИЦИАЛИЗАЦИИ

- method, body, headers
- mode, credentials
- cache, redirect

```
const myInit = {
    method: 'GET',
    mode: 'cors',
    cache: 'default',
    body: JSON.stringify({test: 'value')
};
const myRequest = new Request('/test.json');

fetch(myRequest, myInit).then((response) => {});
```

REAL TIME UPDATES

POLLING

Запрос с клиента на сервер каждые N секунд

Ответ с сервера возвращается сразу

```
const pollItems = () => {
  fetch('https://tt-front-server.track-mail-ru.now.sh/')
    .then(resp => resp.json())
    .then(data => console.log(data));
}

const t = setInterval(() => pollItems(), 1000);

// clearInterval(t)
```

LONG POLLING

Запрос с клиента на сервер, где

- сервер отвечает только когда будет, чем ответить
- после получения ответа от сервера выполняется новый запрос

SSE

Server Sent Events / EventSource

Клиент устанавлиеает соединение с сервером

Сервер отправляет данные по установленному соединению

```
html

head
    title sse

body

    div

    script.

        const sse = new EventSource('/messages');
        console.log(sse.withCredentials);
        console.log(sse.readyState);
        console.log(sse.url);
        let eventList = document.querySelector('ul');
        sse.onopen = function() {
            console.log('Connection to server opened.');
        };
        sse.onmessage = function(e) {
            const newElement = document.createElement("li");
            let html = ''
            let opType;
            try {
                const msg = JSON.parse(e.data)
```

WEB SOCKETS

Клиент устанавливает соединение с сервером

И клиент, и сервер могут отправлять данные друг другу

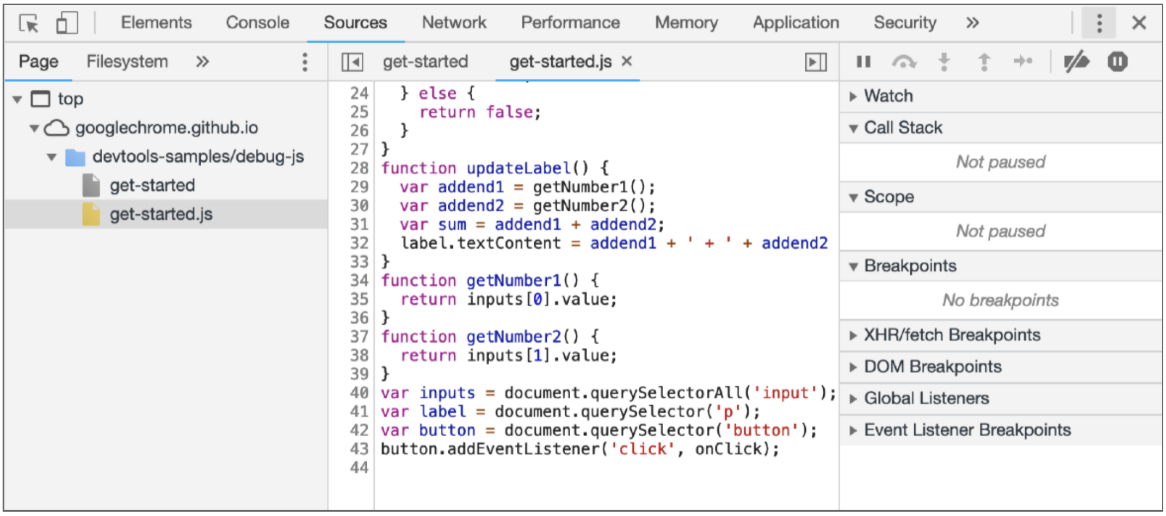
What are Long-Polling, Websockets, Server-Sent Events (SSE) and Comet?

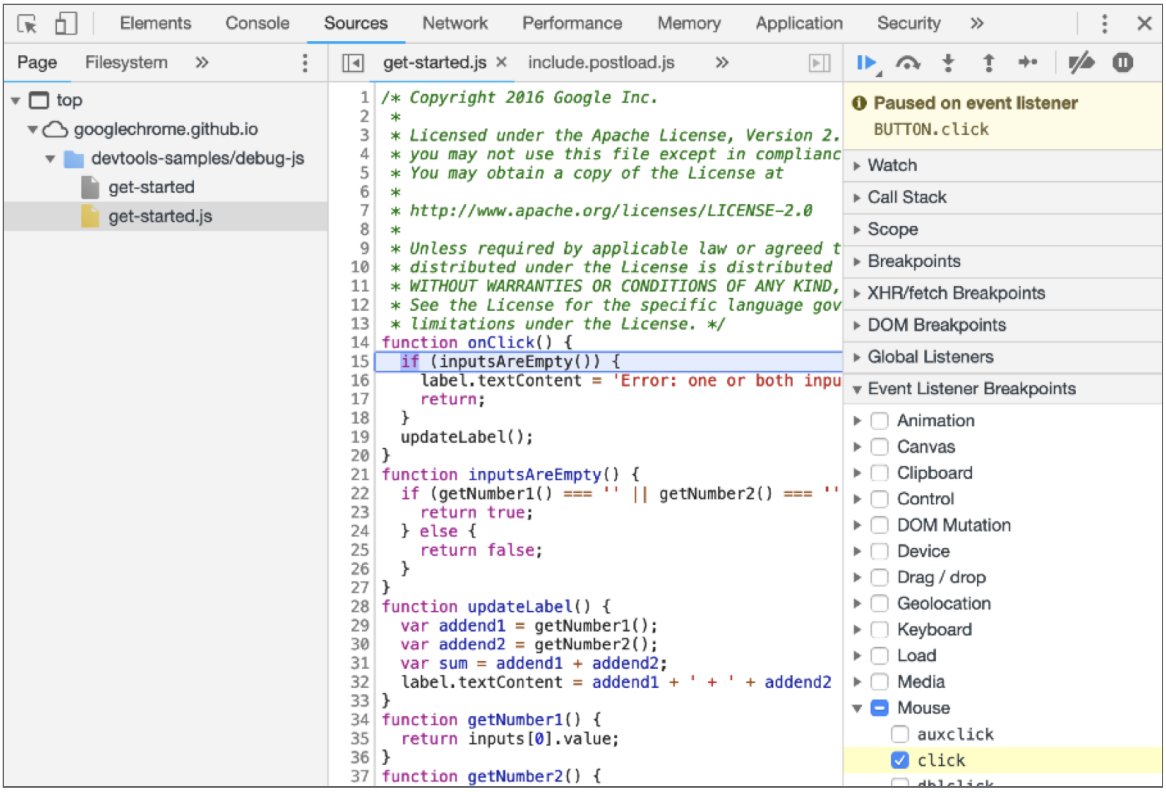
https://stackoverflow.com/a/12855533/39841

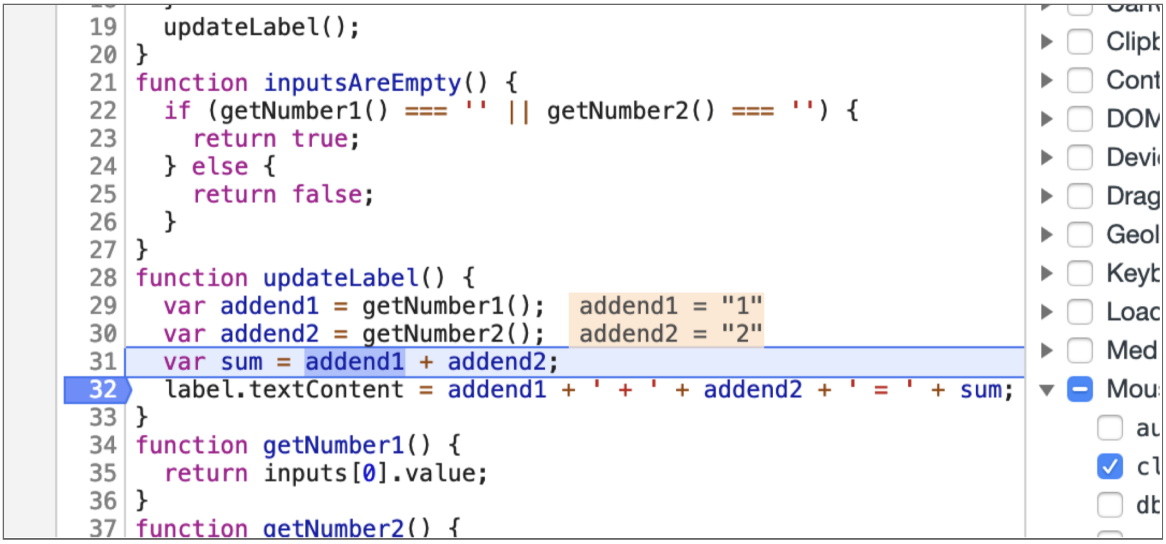
ИНСТРУМЕНТЫ РАЗРАБОТЧИКА

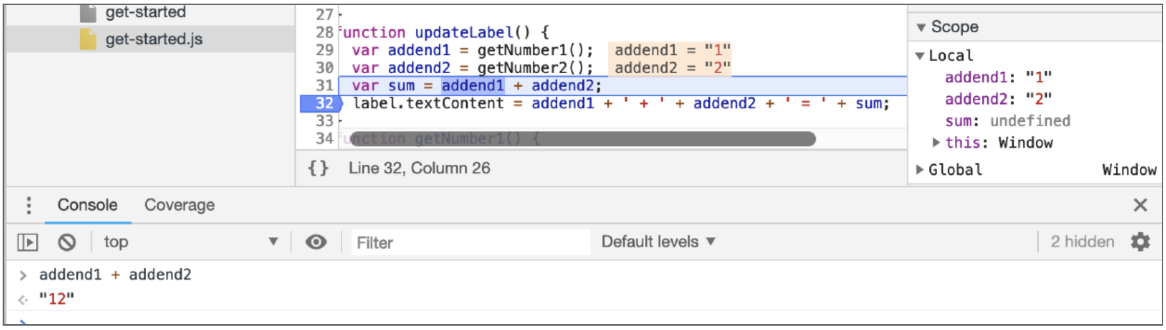
Ресурс для испытаний:

https://googlechrome.github.io/devtools-samples/debug-js/get-started


Elements
Console
Sources
Network
Performance
Memory
Application
Security
Page
Filesystem
get-started
get-started.js
top
googlechrome.github.io
devtools-samples/debug-js
get-started
get-started.js
} else {
return false;
}
}
function updateLabel() {
var addend1 = getNumber1();
var addend2 = getNumber2();
var sum = addend1 + addend2;
label.textContent = addend1 + ' + ' + addend2
}
function getNumber1() {
return inputs[0].value;
}
function getNumber2() {
return inputs[1].value;
}
var inputs = document.querySelectorAll('input');
var label = document.querySelector('p');
var button = document.querySelector('button');
button.addEventListener('click', onClick);
Watch
Call Stack
Not paused
Scope
Not paused
Breakpoints
No breakpoints
XHR/fetch Breakpoints
DOM Breakpoints
Global Listeners
Event Listener Breakpoints


Elements
Console
Sources
Network
Performance
Memory
Application
Security
Page
Filesystem
get-started.js
include.postload.js
top
googlechrome.github.io
devtools-samples/debug-js
get-started
get-started.js
/* Copyright 2016 Google Inc.
* Licensed under the Apache License, Version 2.
* you may not use this file except in compliance
* You may obtain a copy of the License at
* http://www.apache.org/licenses/LICENSE-2.0
* Unless required by applicable law or agreed t
* distributed under the License is distributed
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,
* See the License for the specific language gov
* limitations under the License. */
function onClick() {
if (inputsAreEmpty()) {
label.textContent = 'Error: one or both inpu
return;
}
updateLabel();
}
function inputsAreEmpty() {
if (getNumber1() === '' || getNumber2() === ''
return true;
} else {
return false;
}
}
function updateLabel() {
var addend1 = getNumber1();
var addend2 = getNumber2();
var sum = addend1 + addend2;
label.textContent = addend1 + ' + ' + addend2
}
function getNumber1() {
return inputs[0].value;
}
function getNumber2() {
Paused on event listener
BUTTON.click
Watch
Call Stack
Scope
Breakpoints
XHR/fetch Breakpoints
DOM Breakpoints
Global Listeners
Event Listener Breakpoints
Animation
Canvas
Clipboard
Control
DOM Mutation
Device
Drag / drop
Geolocation
Keyboard
Load
Media
Mouse
auxclick
click


updateLabel();
}
function inputsAreEmpty() {
if (getNumber1() === '' || getNumber2() === '') {
return true;
} else {
return false;
}
}
function updateLabel() {
var addend1 = getNumber1();
var addend2 = getNumber2();
var sum = addend1 + addend2;
label.textContent = addend1 + ' + ' + addend2 + ' = ' + sum;
}
function getNumber1() {
return inputs[0].value;
}
function getNumber2() {
addend1 = "1"
addend2 = "2"


get-started
get-started.js
27
28 function updateLabel() {
29 var addend1 = getNumber1(); addend1 = "1"
30 var addend2 = getNumber2(); addend2 = "2"
31 var sum = addend1 + addend2;
32 label.textContent = addend1 + ' + ' + addend2 + ' = ' + sum;
33
34 function getNumber1() {
{} Line 32, Column 26
Scope
Local
addend1: "1"
addend2: "2"
sum: undefined
this: Window
Global Window
Console
Coverage
top
Filter
Default levels
2 hidden
addend1 + addend2
"12"

cmd+s / ctrl+s для сохранений изменений

ОБРАБОТКА ДАННЫХ

Сгенерируем данные для экспериментов

Мок данных: **https://www.json-generator.com/**

На входе

```
[
  '{{repeat(2)}}',
  {
    id: '{{objectId()}}',
    title: '{{lorem(3, "words")}}',
    author: {
      _id: '{{objectId()}}',
      name: '{{firstName()}} {{surname()}}'
    },
    comments: [
      '{{repeat(2)}}',
      {
        id: '{{objectId()}}',
        content: '{{lorem(1, "sentences")}}',
        commenter: {
          _id:  '{{objectId()}}',
          name: '{{firstName()}} {{surname()}}'
        }
      }
    ]
  }
]
```

На выходе


```
[
  {
    "author": {
      "_id": "5dc7007eeec76ec231243922",
      "name": "Jerry Buckley"
    },
    "id": "5dc7007e7fbd1abbdb88c7f8",
    "comments": [
      {
        "content": "Laborum excepteur magna non tempor adipi
        "commenter": {
          "_id": "5dc7007e0567d5ffab316a68",
          "name": "Tate Bass"
        },
        "id": "5dc7007e9ee629c52471c208"
      },
      {
        "content": "Consectetur aliqua sint id dolor anim mi
        "commenter": {
          "_id": "5dc7007e4dfffbeff2faba79",
          "name": "Carson Kirk"
        },
        "id": "5dc7007e5cd4c6aca5fb6f20"
      }
    ],
```


Валидный json, который будем обрабатывать

FILTER, MAP, REDUCE

```
[8, 9, 3, 15].filter(n => n % 2 === 0)
// [8]
['one', 'two'].map(l => `special__${l}`)
// ['special__one', 'special__two']
[1, 2, 3].reduce((res, num) => {return res + num}, 0)
// 6
['one', 'two'].reduce((acc, item) => {acc[item] = item; retu
// {one: "one", two: "two"}
```

ОБРАБОТКА НАШЕГО МАССИВА ДАННЫХ

Данные определенного пользователя

```
const getDataByAuthorId (data, authorId) => (
  data.filter(block => block.author._id === authorId)
)
```

Комментарии определенного пользователя

```
const getCommentsByAuthorId(data, authorId) => (
  let comments = []
  for (const block of data) {
    for (const comment of block.comments) {
      if (comment.commenter._id === authorId) {
        comments.push(comment)
      }
    }
  }
)
```

Пользователь, оставивший больше всех комментариев

```
let commentsPerUser = {};

data.forEach(article => {
 article.comments.forEach(comment => {
   if (commentsPerUser[comment.commenter._id]) {
    commentsPerUser[comment.commenter._id]++;
   } else {
    commentsPerUser[comment.commenter._id] = 1;
   }
 })
});

let userWithMostComments = {};
Object.keys(commentsPerUser).forEach(userId => {
  if (commentsPerUser[userId] > (userWithMostComments.commen
    userWithMostComments = {
      user: userId,
      comments: commentsPerUser[userId]
    };
  }
})
```

Клиент отображает данные

Клиент не обрабатывает данные

НОРМАЛИЗАЦИЯ ДАННЫХ (REACT)

Normalizr:
https://github.com/paularmstrong/normalizr

```
import { normalize, schema } from 'normalizr';

const user = new schema.Entity('users', {}, {
  idAttribute: '_id'
});

const comment = new schema.Entity('comments', {
  commenter: user
});

const article = new schema.Entity('articles', {
  author: user,
  comments: [comment]
});

const normalizedData = normalize(myArticles, [article]);
```

Эксперименты:
https://stackblitz.com/edit/normalizr-playground-jzc2zj?file=index.ts

СПАСИБО ЗА ВНИМАНИЕ