SWIFTACADEMY

НИЕ ВЯРВАМЕ ВЪВ ВАШЕТО БЪДЕЩЕ

JavaScript Object Notation

SWIFTACADEMY

НИЕ ВЯРВАМЕ ВЪВ ВАШЕТО БЪДЕШЕ

Front-End Development

JSON

- JS данните бяха 2 вида:
  - прости: числа, текст, boolean
  - сложни: списъци, обекти
- Ние използваме тези данни постоянно, като ги съхраняваме в променливи, подаваме ги като аргументи на функции и т.н.
- От тези данни зависи как ще се "държи" страницата ни
- JSON е **унифициран** начин за представяне на javascript данните като **текст**
- Използваме го за предаване на данни при комуникация с **backend**-а или при съхраняването им във файлове

JSON формат

- Ето как изглежда един обикновен JS
  - … обект:

```
{
  property1: "value1",
  property2: "value2",
  …
}
```

  - … array (от strings):

```
["obj1", "obj2", "obj3", …]
```

JSON формат

- В JSON формат обектите изглеждат по следния начин:

```
{
   "property1": "value1",
   "property2": "value2",
   …
}
```

- а списъците, ето така: `["obj1", "obj2", "obj3", …]`

- Простите типове данни са същите като в javascript нотацията:
  `[1, true, "text"]`

JSON особености

- един JSON файл може да съдържа точно един обект, списък или примитив

- ако се опитаме да подадем на JSON.parse няколко обекта ще получим грешка

- **property** ключовете и стойностите на обектите и стринговете са **винаги** в двойни кавички!

- никога не трябва да оставяме запетайка след последният елемент в обекта или в списъка (при JS не е проблем, но при JSON е!)

JSON Syntax Rules

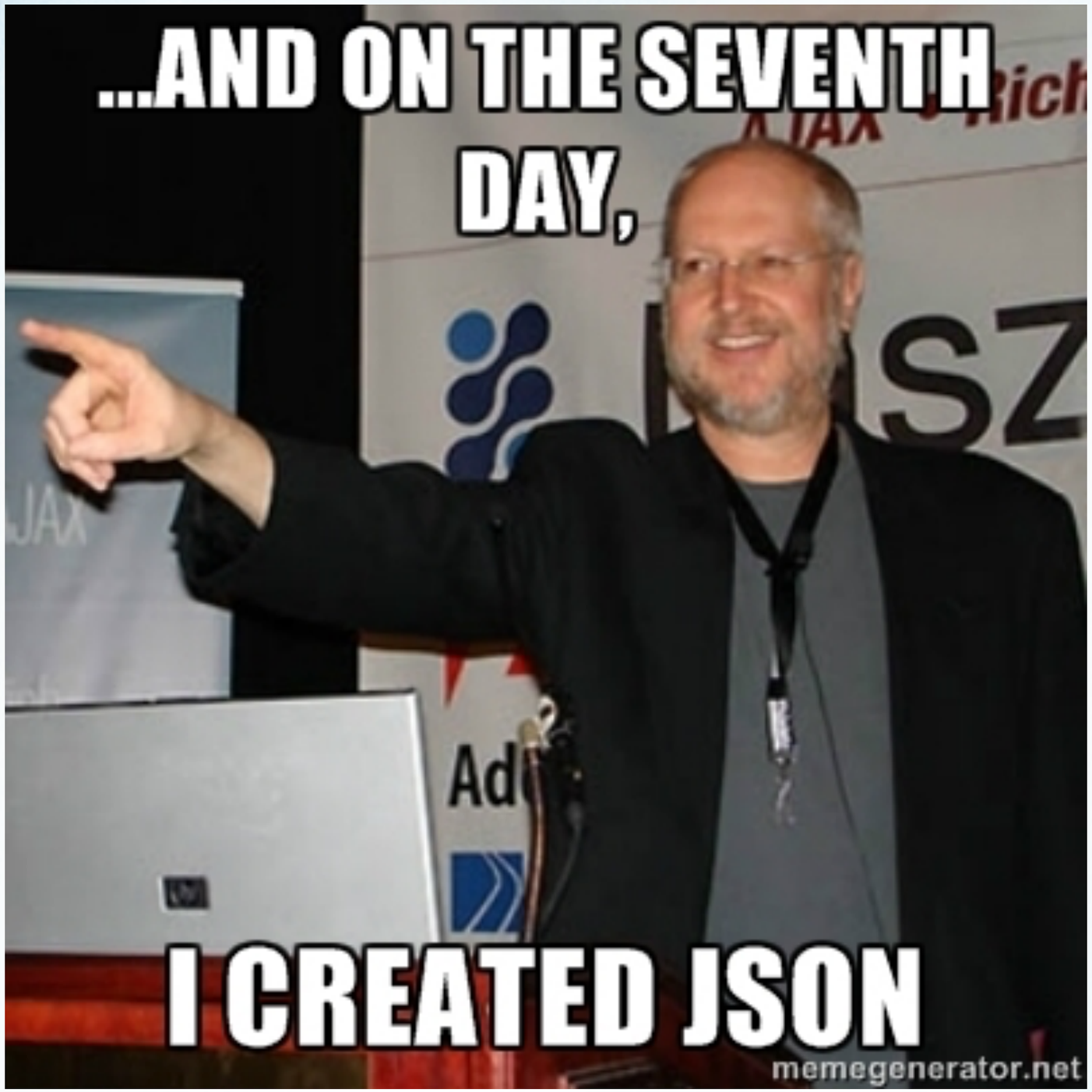
Data is in name/value pairs

Data is separated by commas

Curly braces hold objects

Square brackets hold arrays

http://www.w3schools.com/js/js_json.asp



Douglas Crockford

JSON.stringify() & JSON.parse()

- В JavaScript подобно на Date, Math и Array, има и един глобален обект (клас) JSON, който ни помага да работим с JSON данни
- Използваме го за форматиране на данни в json формат, както и за parse-ването им обратно в JS данни (деформатиране)
- Пример: http://codepen.io/jenie/pen/NrWKVy?editors=0012
- Документация: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON


JavaScript
JSON

jQuery.getJSON()

- getJSON() е функция в JQuery, която ни позволява да четем json файлове

- Пример:

```
var bikes;
$.getJSON('bikes.json', function (data) {
  bikes = data;
});
```

- Документация:
  http://www.w3schools.com/jquery/ajax_getjson.asp
  http://api.jquery.com/jquery.getjson/

Въпроси?


AJAX
HISTORY.COM
imgflip.com

AJAX

- Asynchronous JavaScript and XML

- Това е JavaScript функционалност, която позволява асинхронни заявки от уеб приложението към сървъра (т.е. заявки, които не изискват презареждане на цялата страница и следователно се случват незабелязано от потребителя)

- Пример за такива заявки е по време на регистрация да се проверява дали потребителското име вече е зето.
  Това се случва докато потребителя попълва регистрационната форма (т.е. асинхронно)

- AJAX използва или JSON или XML форматиране, за да обменя JavaScript данни със сървъра

Асинхронни процеси

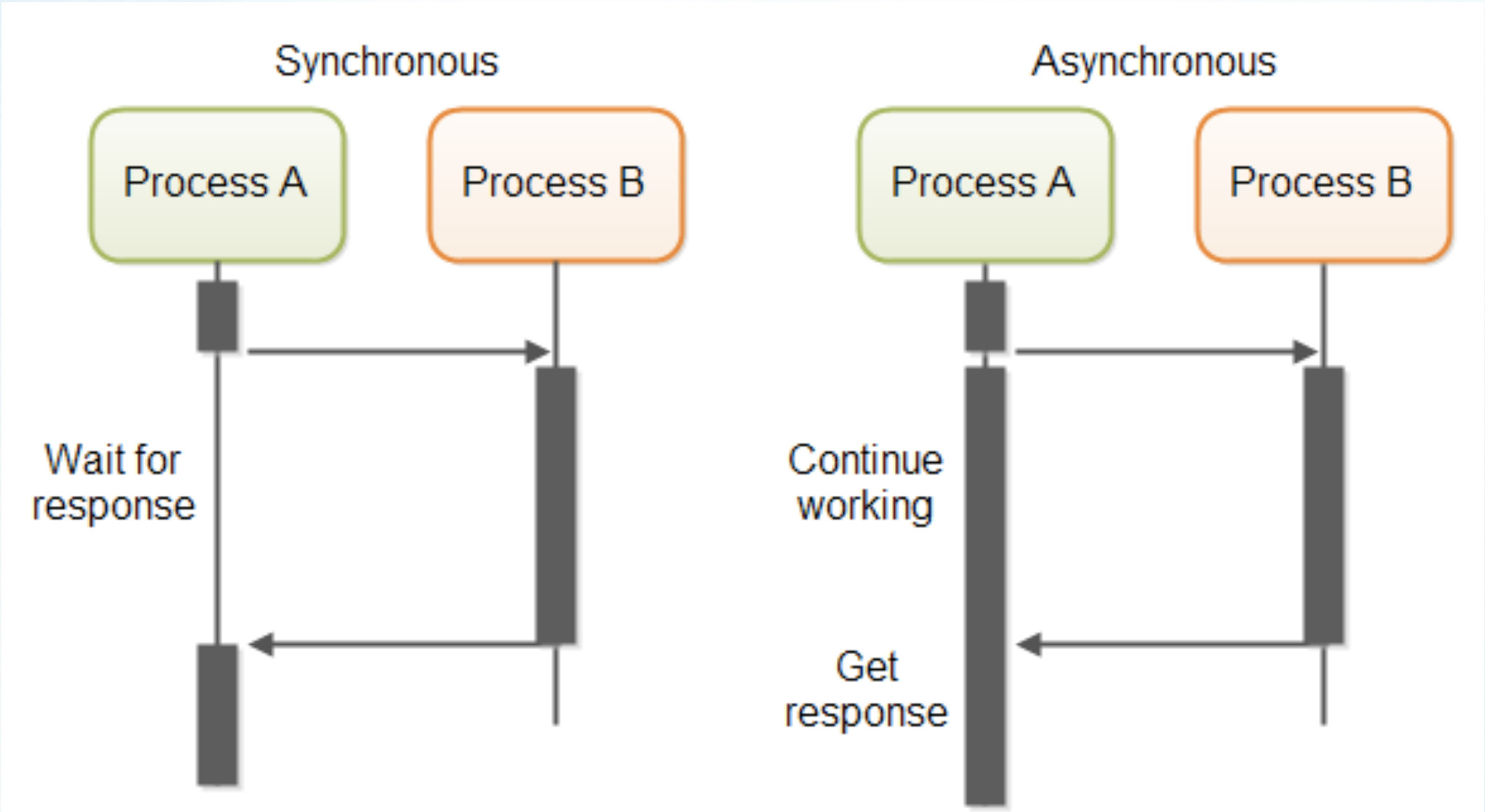
- Когато един процес се изпълнява асинхронно, това означава че останалите процеси се изпълняват успоредно с него, без да се интересуват кога и как ще приключи този процес

- За сравнение: Синхронните процеси се изпълняват един след друг и се изчакват

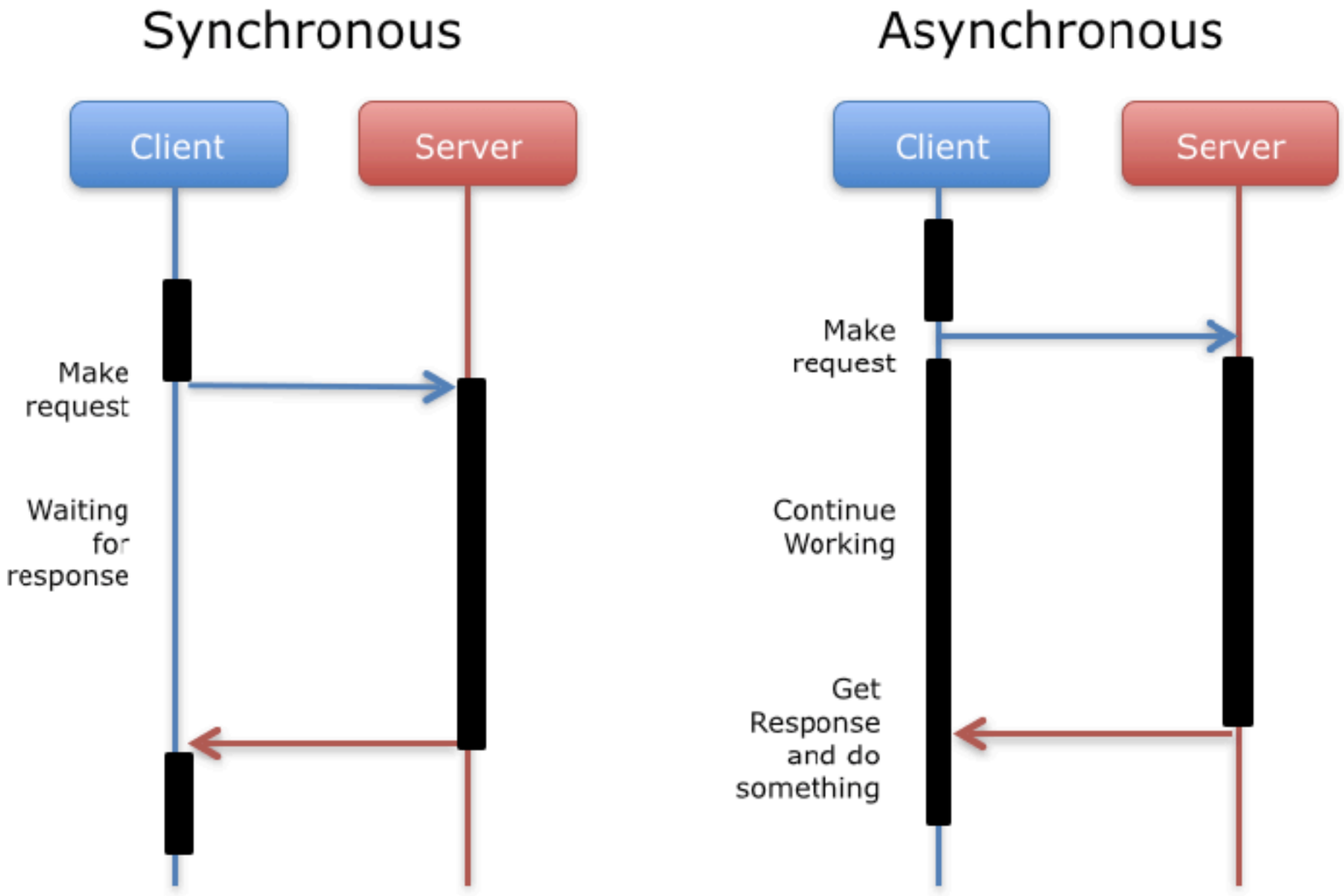
- Използването на асинхронни процеси увеличава бързината (performance-а) на уебсайта, както и потребителското изживяване (UX) многократно!

- AJAX заявките са асинхронни процеси

Асинхронни процеси



AJAX (асинхронни заявки)



Завършване и обработка на резултата

- Тъй като AJAX процеса не е синхронен, неговият резултат не може да се използва директно в JavaScript кода, който следва след заявката (защото този код се изпълнява успоредно със заявката а тогава все още резултата не е получен)

- За да можем да работим с резултатите от AJAX трябва да използваме специални callback методи, които се закачат към самата заявка

- Тези callback методи най-общо могат да се разглеждат като success, error и either_way (finally) callback

- Те дават възможност за обработка на успешен резултат, на грешка или и на двата (успех и грешка)

Класически AJAX request:

```
// This is the client-side script.

// Initialize the Http request.
var xhr = new XMLHttpRequest();
xhr.open('get', 'send-ajax-data.php');

// Track the state changes of the request.
xhr.onreadystatechange = function () {
    var DONE = 4; // readyState 4 means the request is done.
    var OK = 200; // status 200 is a successful return.
    if (xhr.readyState === DONE) {
        if (xhr.status === OK) {
            alert(xhr.responseText); // 'This is the returned text.'
        } else {
            alert('Error: ' + xhr.status); // An error occurred.
        }
    }
};

// Send the request to send-ajax-data.php
xhr.send(null);
```

AJAX Frameworks

- AJAX в чистата си форма е сложен и труден за използване
- По тази причина има много и най-различни frameworks, които дават сравнително приятен синтаксис и инструменти за работа с AJAX
- Аз лично предпочитам и препоръчвам ajax методите на jQuery

```
1 $.ajax({
2   method: "POST",
3   url: "some.php",
4   data: { name: "John", location: "Boston" }
5 })
6   .done(function( msg ) {
7     alert( "Data Saved: " + msg );
8   });
```

http://api.jquery.com/jquery.ajax/

This is AJAX!

Въпроси?

Примери

Coffee machine: link

Coffee machine with namespace: link

Coffee machine with more stuff: link

Домашно 19 (филми): link

Bootstrap template: link

Bike shop with JSON: link

All: http://swift-academy.zenlabs.pro/lessons/lesson21/examples/download.zip

Домашно

http://swift-academy.zenlabs.pro/lessons/lesson21/homework