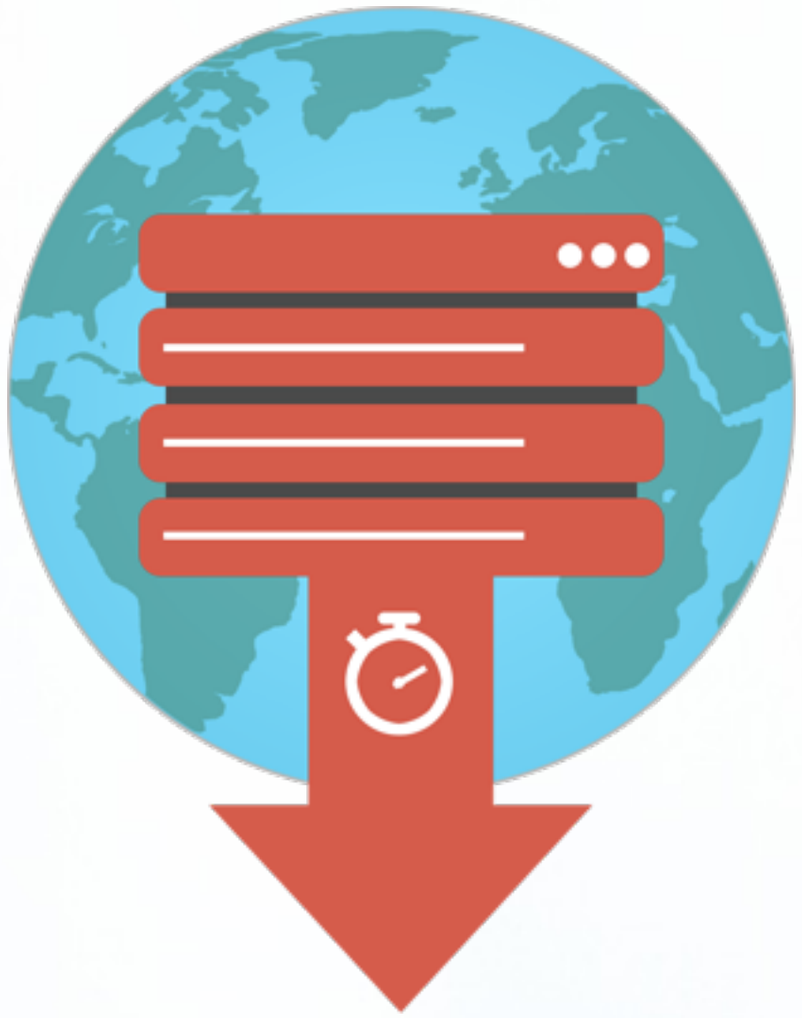
SWIFTACADEMY

НИЕ ВЯРВАМЕ ВЪВ ВАШЕТО БЪДЕЩЕ

Page Load Time



Зареждане и обработка

- За да покаже една страница в крайният й вид, браузъра минава през няколко основни етапа:
  - теглене на HTML документа (download)
  - прочитене на **head** частта и изтегляне на всички линкове в head-а (пак download)
    - парсване (т.е. обработка - идва от думата **parse**) на всичко в head-а: CSS файлове, фонтове, генериране на CSSOM-а, изпълняване на скриптовете, генериране на BOM (**window** обекта)
  - прочитане на **body**
    - парсване на html-а и генериране на DOM-а (**document** обекта), теглене на допълнителни скриптове и картинки, генериране на CSSOM-а
  - визуализация (чак след като всичко по-горе е изпълнено)

Good to know..

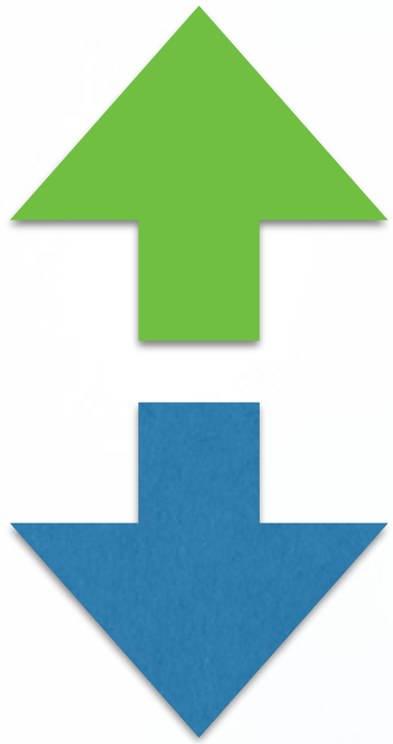
- Скоростта на зареждане е критична за потребителя
  - ако страницата се забави над 10секунди, много от потребителите се отказват да я чакат)
- Броят и видът на ресурсите, които страницата зарежда допълнително (фонтове, css, скриптове, картинки) са критични за времето за зареждане на страницата
- Начинът и последователността на включване на ресурсите може много да подобри performance-а на страницата
  - no <script> in the head!
  - or: **async/defer** the scripts (not fully supported so avoid)
  - generate dynamically all extra content after DOM ready

First things - first!

- Когато искаме страницата ни да се зареди бързо, трябва много внимателно да си организираме линковете и скриптовете
- Прието е всичко, което е необходимо за генерирането на визията на страницата, да се зарежда най-напред
- Това са CSS, fonts, background картинки и евентуално някои скриптове, от които зависи показването на съдържание
- Зареждането на всичко останало се отлага, за да може потребителя да не усети забавяне в зареждането на страницата

Златното правило

CSS



JS

стиловете: в <head>

скрипта: в края на <body>

Заключение

- Запомнете, че браузърът винаги зарежда линковете в head-а преди всичко останало

- Запомнете също, че скриптовете, които са там, ги тегли и ги изпълнява още преди да е парснат html кода !!
(т.е. преди да е заредил DOM-а в document обекта)

- Запомнете, че визуализацията на страницата се случва едва след като е зареден DOM-а

- И не забравяйте, че скоростта на зареждане и показване (визуализация) на страницата е критична!

За четене

- Performance by Google: https://developers.google.com/web/fundamentals/performance/?hl=en
- Optimizing the Critical Rendering Path: http://www.sitepoint.com/optimizing-critical-rendering-path/
- Малко по-адвансед: http://devnet.kentico.com/articles/optimizing-page-load-time (статията е от 2011!)
- Fast enough: http://www.vinta.com.br/blog/2015/web-page-optimization.html
- async и defer: http://peter.sh/experiments/asynchronous-and-deferred-javascript-execution-explained/

JavaScript Object Notation

SWIFTACADEMY

НИЕ ВЯРВАМЕ ВЪВ ВАШЕТО БЪДЕЩЕ

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
- 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"]

- Т.е. единствената разлика в представянето, е че **property** ключовете на обектите и стринговете са **винаги** в двойни кавички.
  Освен това, никога не трябва да оставяме запетайка след последният елемент в обекта или списъка (при JS не е проблем, но при JSON е!)

JSON Syntax Rules

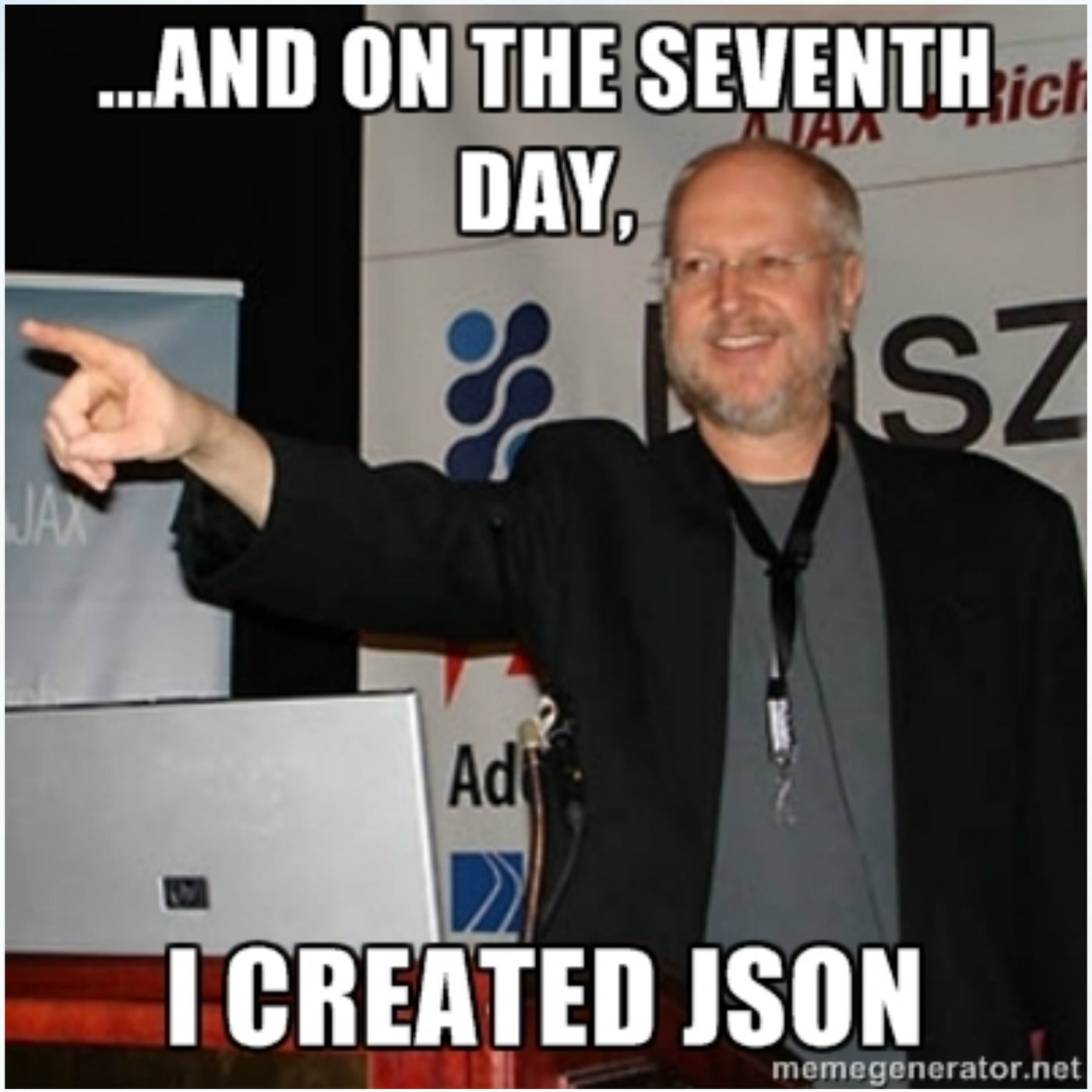
Data is in name/value pairs

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 се случват незабелязано от потребителя)

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

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

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/

Въпроси?

Полезни връзки

- Ефектни идеи за off-canvas навигация, странициране и afix header:

http://tympanus.net/codrops/2013/08/28/transitions-for-off-canvas-navigations/

Примери

http://swift-academy.zenlabs.pro/lessons/lesson18/examples/download.zip

Домашно

http://swift-academy.zenlabs.pro/lessons/lesson18/homework