JS BASICS (DAY 1) Introduction

Some of examples and definitions are from very good JS docs - https://developer.mozilla.org/ .

Hello!

I am Mateusz Choma

I am a scientific mind, passionate of technology, an engineer "squared" - a graduate of two universities in Lublin :)
As well, I am a JS developer, entrepreneur and owner of small software house - Amazing Design.

1.
History of JavaScript

Language made in 10 days :)

History of JavaScript

Beginnings and standardized editions

- 10 days of may 1995
- Brendan Eich
- Netscape Navigator

https://en.wikipedia.org/wiki/ECMAScript

History of JavaScript
Standardized editions - ECMAScript

ECMAScript is a language specification standardized by Ecma International in ECMA-262 and ISO/IEC 16262.

- ECMAScript - 1997
- ECMAScript 5.0 - 2009
- ECMAScript 6 - 2015
- ECMAScript 7 - 2016
- ECMAScript 8 - 2017

History of JavaScript
Useage

- websites (interactivity animations)
- web servers (Node.JS)
- mobile applications (Cordova, React Native)
- IoT (internet of things)

History of JavaScript
JS engine and environment

JavaScript is an interpreted language so it needs an interpreter to run!

“In computer science, an interpreter is a computer program that directly executes, i.e. performs, instructions written in a programming or scripting language, without requiring them previously to have been compiled into a machine language program.” - *Wikipedia*

A JavaScript engine is an interpreter which executes JavaScript code.

History of JavaScript
JS engine and environment

SpiderMonkey is the **first** JavaScript created by Brendan Eich at Netscape, implemented in C++

V8 is **the most popular** and open source, developed by Google, written in C++

Others:

> **Rhino** is managed by the Mozilla Foundation, open source, developed entirely in Java
> **Carakan** engine developed by Opera Software included in the Opera web browser, until switching to V8 with Opera 15

History of JavaScript
JS engine and environment

Environment for JavaScript is typically the browser or Node.JS (server side environment)

“ Task 0 - Hello World

- *Open developers tools*
- *Go to console*
- *Write: alert(‘hello world’)*

2.
Keywords and embedding

Keywords and embedding
Reserved keywords

- break
- case
- catch
- class
- const
- continue
- debugger
- default
- delete
- do
- else
- export
- extends
- finally
- for
- function
- if
- import
- in
- instanceof
- new
- return
- super
- switch
- this
- throw
- try
- typeof
- var
- void
- while
- with
- yield

Keywords and embedding
Embedding JS

```
<script>
  ...
</script>


<script src="script.js"></script>
```

Keywords and embedding
Embedding JS

There are two areas in HTML document where JavaScript can be placed.

First is between `<head>......</head>` tag

Another is in `<body>......</body>` tag

Task 1 - Hello World 2

- *Place the hello world code - alert('hello world'), into the script tag and into separate file and connect it to HTML.*

3.
Variables and types

Grammar and keywords
Data types - primitives and objects

The **primitive values** are:

- undefined,
- null,
- booleans,
- numbers,
- strings

All other values are **objects**, including arrays and functions.

The primitives are passed by value and objects by reference.

Grammar and keywords
Literals

Null literal

```
null
```

Boolean literal

```
true
false
```

Numeric literals

```
1234567890
42
```

Grammar and keywords
Literals

String literals

```
'foo'
"bar"
```

Array literal

```
[1954, 1974, 1990, 2014]
```

Object literals

```
var o = { a: 'foo', b: 'bar', c: 42 }
```

Grammar and keywords
Grammar

We can use **typeof** operators to check type of given value.

“ *Task 2*

- *Open console in dev tools*
- *Make literal of any type*
- *Check type using typeof*

Grammar and keywords
Variables

“**Variable** - a storage location paired with an associated symbolic name (an identifier), which contains some known or unknown quantity of information referred to as a value.” - *Wikipedia*

Variables can store values or references!

Grammar and keywords
Variables

Variable declaration is the fact of putting the variable in certain scope (more about scope will be in functions workshop).

Variable initialization (definition) is the fact of giving the variable its first value. Further variable modifications are called **assignments**.

Grammar and keywords
Variables

```
var x; // declaration
var y;

x = 10; // definition
y = 10;

x = y + 2; // assignment
```

Grammar and keywords
Variables - re-declaration

```
var x=5;
var x;
alert(x); // result will be x
```

Grammar and keywords
Variables - valid names

Rules for constructing names for variables:

- names can contain **letters**, **digits**, **underscores**, and **dollar signs**
- names **must begin with a letter or $ or_**
- names are **case sensitive**
- **reserved words** (like JavaScript keywords) **cannot be used** as names

Grammar and keywords

Variables - naming good practices

- declare variables on the beginning of scope (on the top of document/script)
- don’t re-declare variables
- name variables in camel case
- name of variable should be self-describing

“ *Task 3*

- *Make a variables and assign to them values of any type*
- *Check types of these variables using typeof*

Grammar and keywords
Truthy and falsy variables

In JavaScript, a truthy value is a value that is considered true when evaluated in a boolean (true/false) context.

All values are truthy unless they are defined as falsy.

Grammar and keywords
Truthy and falsy variables

Falsy variables are:

false,
0,
"",
null,
undefined,
NaN.

4. Grammar

Read more - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Lexical_grammar .

Grammar and keywords
Grammar

White space doesn't matters!

Lines above do the same:

```
result = a + b
result=a+b
result = a          +               b
```

Grammar and keywords
Grammar

Lines should end by semicolon, but JavaScript has automatic semicolon insertion (ASI)!

Semicolons can be omitted!

Be careful about line breaks!

Grammar and keywords
Grammar

Variables are case sensitive!

```
result = 10
RESULT = 11
```

Lines above are different variables!

Grammar and keywords
Comments

One line:

```
// comment
```

Multi-line:

```
/*
comment
comment
comment
*/
```

Grammar and keywords
Comments

Comment should describe line of code above comment.

Code should be self-describing as much as we can write it.

Comment is last available option to make our code more readable.

5.
Basic math operations

Basic math operations
+ and -

```
var a = 1 + 1

var b = 2 - 3

var c = a + b
```

Basic math operations
+ and -

WARNING!

+ is also concatenation operator!

Try:

```
var result = ‘ala’ + ‘ ’ + ‘ma kota’
```

It can make problems!

Basic math operations

* and /

```
var a = 2 / 2

var b = a * 2

var c = a * b
```

Basic math operations

Modulo %

```
var a = 2 % 2 // result is 0

var b = 4 % 3 // result is 1

var c = 2 % 4 // result is 2
```

Basic math operations

+= and -= operators

```
var a = 2
var b = 4

a += a // result is 4
b -= b // result is 0
```

Basic math operations

Increment ++ and decrement -- operators

```
var a = 2
var b = 4
```

Checkout on console:

```
a--
a
--a
a
```

```
b++
b
++b
b
```

Basic math operations

Unary plus (+)

```
+3        // 3
+'3'      // 3
+true     // 1
+false    // 0
+null     // 0
+{}       // NaN
```