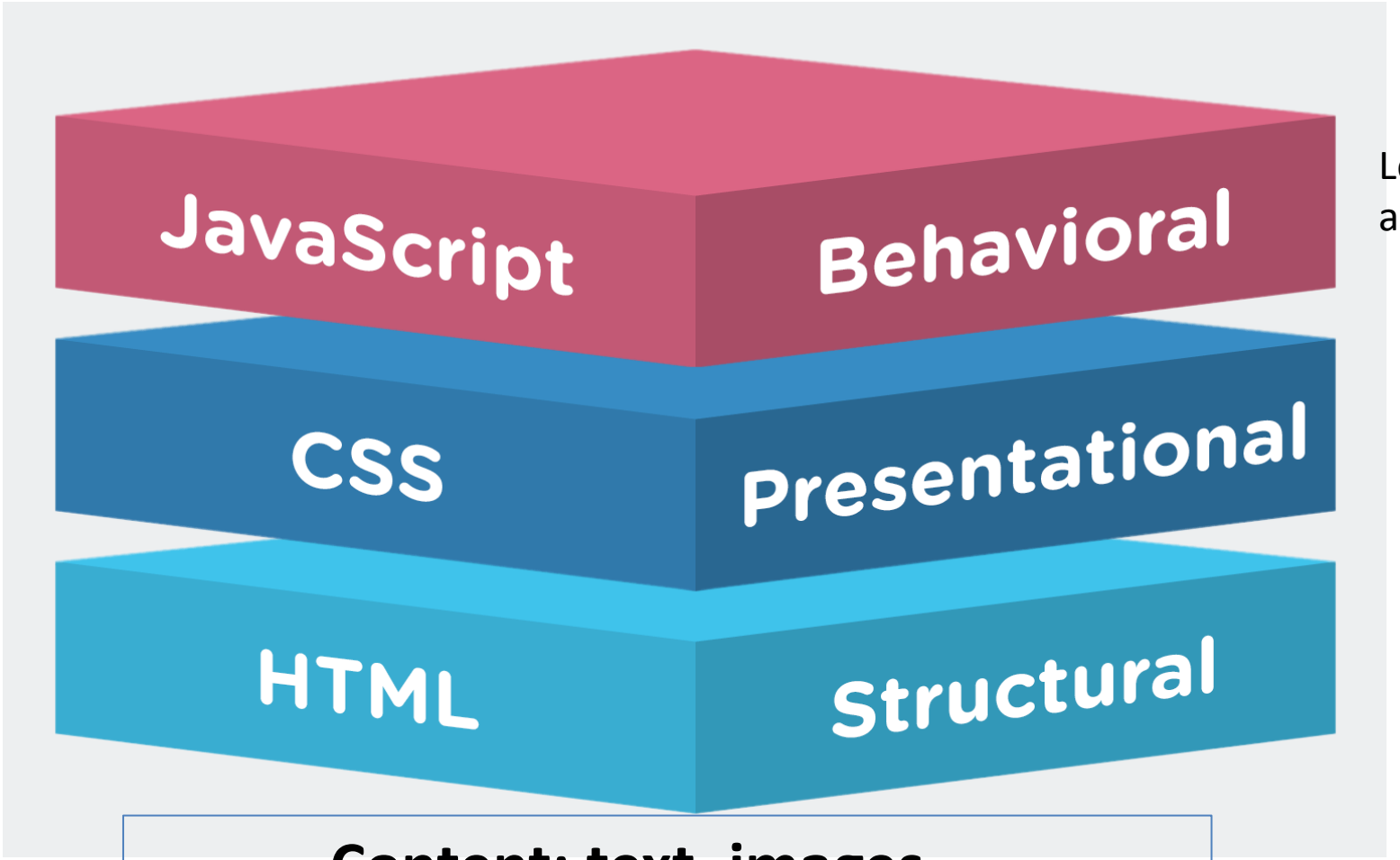
JavaScript
Part 1

Grzegorz Ostrek
g.ostrek@mini.pw.edu.pl
Wydział Matematyki i Nauk Informacyjnych
00-662 Warszawa
ul. Koszykowa 75



Logic, action

Content: text, images, …

Ver (ES#)	Official Name	Description
1	ECMAScript 1 (1997)	First Edition.
2	ECMAScript 2 (1998)	Editorial changes only.
3	ECMAScript 3 (1999)	Added Regular Expressions. Added try/catch.
4	ECMAScript 4	Never released.
5	ECMAScript 5 (2009)	Added "strict mode". Added JSON support. Added String.trim(). Added Array.isArray(). Added Array Iteration Methods.
5.1	ECMAScript 5.1 (2011)	Editorial changes.
6	ECMAScript 2015	Added let and const. Added default parameter values. Added Array.find(). Added Array.findIndex().
7	ECMAScript 2016	Added exponential operator (**). Added Array.prototype.includes.
8	ECMAScript 2017	Added string padding. Added new Object properties. Added Async functions. Added Shared Memory.
9	ECMAScript 2018	Added rest / spread properties. Added Asynchronous iteration. Added Promise.finally(). Additions to RegExp.
10	ECMAScript 2019	String.trimStart() and String.trimEnd() Object.fromEntries() const entries = [['foo', 'bar']]; const object = Object.fromEntries(entries);// { foo: 'bar' } Array.flat() and Array.flatMap() Symbol.description Optional catch
11	ECMAScript 2020	

JavaScript was invented by Brendan Eich in 1995, and became an ECMA standard in 1997.

LiveScript – name of the first JavaScript implementation by Netscape
JScript – Microsoft's dialect of JavaScript
ActionScript – Adobe Flash implementation

ECMA-262 is the official name of the standard.
ECMAScript is the official name of the language.
JavaScript (JS) is an implementation of ECMAScript

Ecma International (*European Association for Standardizing Information and Communication Systems*; former *European Computer Manufacturers Association*)

http://www.ecma-international.org/publications/standards/Ecma-262.htm

15 ECMA Script 2024

Element <script>

https://developer.mozilla.org/en-US/docs/Web/HTML/Element/script

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

- **async** script will be executed asynchronously as soon as it is available
- **defer** script is executed when the document has finished parsing
- **async defer** script falls back to the defer behavior if async is not supported

Data types

- **primitive** (primitive value, primitive data type) is data that is not an object and has no methods, primitives are immutable

6 primitive data types:
string,
number,
bigint,
boolean,
undefined,
symbol

valueOf() method returns the primitive value.

5 primitive values have object equivalents

String	for the string primitive.	
Number	for the number primitive.	
BigInt	for the bigint primitive.	ES11
Boolean	for the boolean primitive.	
Symbol	for the symbol primitive.	ES6

any structured type is derived from null ie. null is an object not primitive

The variable may be reassigned a new value, but the existing value can not be changed in the ways that objects, arrays, and functions can be altered.

https://developer.mozilla.org/en-US/docs/Glossary/Primitive

Data types Examples

- var x1 = {}; // new object
- var x2 = ""; // new primitive string
- var x3 = 0; // new primitive number
- var x4 = false; // new primitive boolean
- var x5 = []; // new array object
- var x6 = /()/ // new regexp object
- var x7 = function(){}; // new function object

Data types details

JavaScript **Number** type is a [double-precision 64-bit binary format IEEE 754]

most JavaScript engines store a small enough number without a decimal fraction as an integer (with, for example, 31 bits),

Array indexes, all of the bitwise operators (convert their operands to 32-bit integers)

Any character ~ USC-2/UTF-16 – DOM compilant
- function with string may return a UTF-16 **code unit**

variable declared without a value will have the value undefined.
In JavaScript null is "nothing". It is supposed to be something that doesn't exist.

```
null === undefined // false
null == undefined // true
```

Data types operators

- https://www.w3schools.com/js/js_operators.asp
- + Addition Numbers are added. Strings are concatenated.
- - Subtraction
- * Multiplication
- ** Exponentiation (ES2016)
- / Division
- % Modulus (Division Remainder)
- ++ Increment
- -- Decrement

Validation

- JavaScript/ECMAScript linters
  - JSHint
    - npm install -g jshint
    - jshint nazwapliku.js //in console
  - ESLint

DOM – Document Object Model

DOM is not JavaScript-specific, and indeed has been implemented in numerous other languages. For Web browsers, however, the DOM has been implemented using ECMAScript and now makes up a large part of the JavaScript language.

JavaScript implementation is made up of three distinct parts

- The Core (based on ECMAScript spec)
  browser is considered a host environment for ECMAScript

- The Document Object Model (DOM)
  Document Object Model (DOM) is a application programming interface (API) for HTML as well as XML.

- The Browser Object Model (BOM)
  allows access and manipulation of the browser window. Using the BOM, developers can move the window, change text in the status bar, and perform other actions that do not directly relate to the page content.

The DOM is the Document Object Model, which deals with the document, the HTML elements themselves, e.g. document and all traversal you would do in it, events, etc.

The BOM is the Browser Object Model, which deals with browser components aside from the document, like history, location, navigator and screen (as well as some others that vary by browser).

DOM cont.

- **DOM level 1** consisted of two modules: the DOM Core, which provided a way to map the structure of an XML-based document to allow for easy access to and manipulation of any part of a document, and the DOM HTML, which extended the DOM Core by adding HTML-specific objects and methods.

- **DOM Level 2** introduced several new modules of the DOM to deal with new types of interfaces:

  DOM Views — describes interfaces to keep track of the various views of a document (that is, the document before CSS styling and the document after CSS styling)
  DOM Events — describes interfaces for events
  DOM Style — describes interfaces to deal with CSS-based styles
  DOM Traversal and Range — describes interfaces to traverse and manipulate a document tree

- **DOM Level 3** further extends the DOM with the introduction of methods to load and save documents in a uniform way (contained in a new module called DOM Load and Save) as well as methods to validate a document (DOM Validation). In Level 3, the DOM Core is extended to support all of XML 1.0, including XML Infoset, XPath, and XML Base.

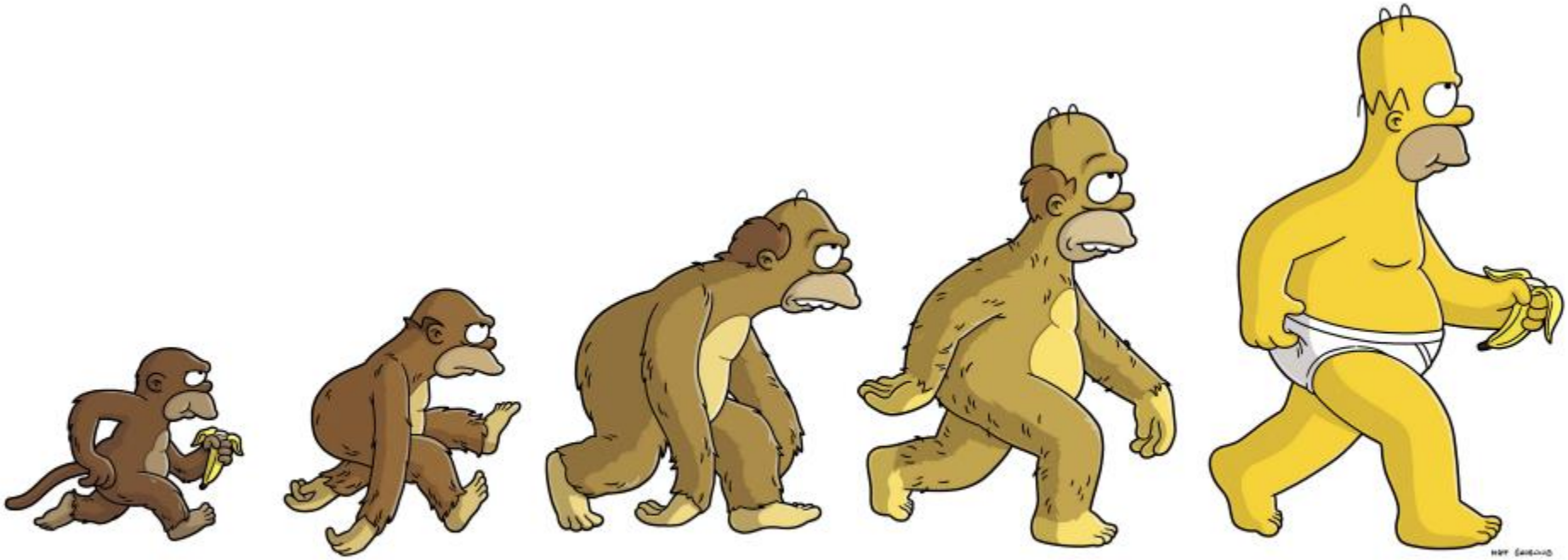
- Web APIs

• API Console	Console object provides access to the browser's debugging console.
• API Geolocation	Geolocation API is published through the navigator.geolocation object.
• API History	history object is part of the window object and is accessed through the window.history property.
• API Storage	provides access to the session storage or local storage for a particular domain.
window.localStorage	Allows to save key/value pairs in a web browser. Stores the data with no expiration date
window.sessionStorage	Allows to save key/value pairs in a web browser. Stores the data for one session

The DOM is the hierarchical representation data used to manage state in modern web browsers.

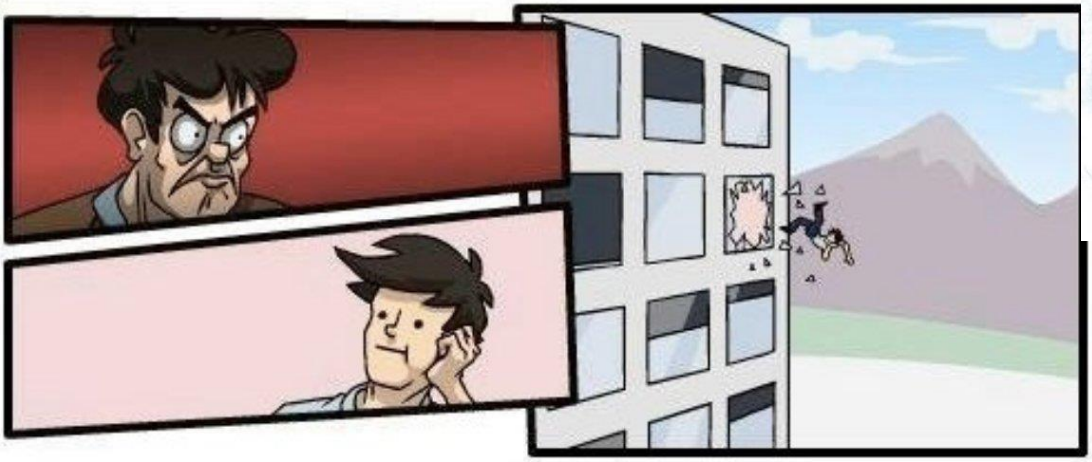
https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model

- https://javascript.info/
- https://262.ecma-international.org/5.1/
- https://262.ecma-international.org/6.0/
- http://es6-features.org/#Constants
- https://compat-table.github.io/compat-table/es6/
- https://pages.mini.pw.edu.pl/~okulewiczm/www/?Teaching:HTML2PostGIS


MACHINE
ASSEMBLY
PROCEDURAL
OBJECT ORIENTED
FUNCTIONAL


Alright. How do we bring ML models to edge devices?


Make models smaller
Make devices more powerful
Write models in JavaScript so they can run in browsers




DYNAMIC TYPING
STATIC TYPING
DONE!