VŠB TECHNICKÁ UNIVERZITA OSTRAVA | FAKULTA ELEKTROTECHNIKY A INFORMATIKY | KATEDRA INFORMATIKY

Basics of Information Technology

Ing. Michal Radecký, Ph.D. MBA

JavaScript

What is JavaScript

A scripting programming language (interpreted, the source code is processed directly) designed to solve the dynamics of WWW pages on the client side.

Not only for web frontend (e.g. node.js)

- HTML Source Code Component (DOM)
- Multiplatform
- Dependent on the interpreter environment (browser)
- In principle, object-oriented, but classless (prototypes), today also native support for OOP
- Case-sensitive
- Syntax similar to languages such as C/C++/Java/Python
- Untyped

History of JavaScript

- First introduced (LiveScript) in 1995 as part of Netscape Navigator and as a response to the need to make web pages interoperable by means other than Java Applets.
- Fast expansion with respect to almost no input requirements (C/C++/Java syntax, no compiler, etc.)
- Microsoft's response was a form of VBScript that was only supported on the Windows platform. In 1996, the JScript porting was introduced as part of IE 3.0.
- In 1997, a version of **ECMAScript** was standardized, providing a kernel common and supported by all browsers.
- Browsers from HTML 4.0 onwards supported **DOM (Document Object Model, Level 1)**, but again in different implementations (document.layers vs. document.all)
- After 2000, W3C DOM to Level 3, which was already interpreted universally by browsers
- Today part of the HTML Living Standard (WHATWG, unversioned)

What JavaScript can do

Current JavaScript evolved from a simple scripting language in a **universal and comprehensive tool** not only for the web. He was inspired by Perl, C/C++, Java, Python and TCL.

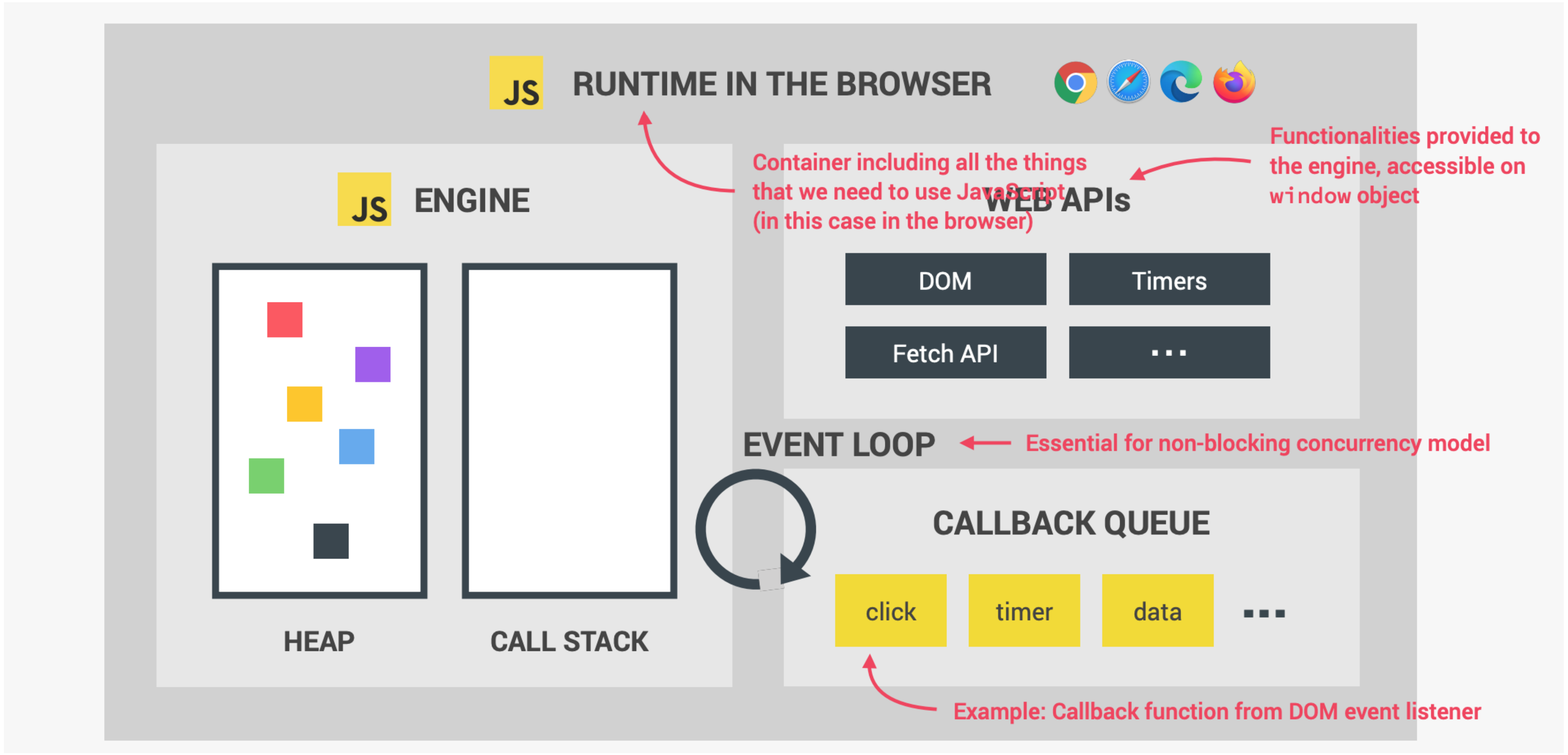
- Influence the appearance and content of an HTML document (DOM)
- Manipulate images and page elements
- Perform algorithms and mathematical calculations
- Control and process forms and their data
- Respond to user events (clicks, keys, mouse movement)
- Work with web APIs
- Work with browser-based storage (LocalStorage, SessionStorage) and data formats (JSON, XML, Data URL, etc.)
- Read and store cookies
- Work with external interfaces and plugins

In the web environment, the JavaScript **workspace is restricted by the browser** - it does not have full access to the system (for security reasons).

What JavaScript can't do

- Access the network directly – can only use the browser interface (e.g. HTTP fetch, WebSocket – but not TCP/UDP connections)
- Read or write files to local disk (except File APIs, user dialogs and interactions)
- Perform authentication and authorization on its own (uses server protocols and services – e.g. OAuth, JWT)
- Run applications or operating system commands
- Directly draw vector graphics (via SVG)
- Work if user don‘t want

https://code-kage.hashnode.dev/what-is-javascript-and-how-it-works-behind-the-scene


JS RUNTIME IN THE BROWSER
Container including all the things that we need to use JavaScript (in this case in the browser)
JS ENGINE
HEAP
CALL STACK
WEB APIs
Functionalities provided to the engine, accessible on window object
DOM
Timers
Fetch API
...
EVENT LOOP
Essential for non-blocking concurrency model
CALLBACK QUEUE
click
timer
data
...
Example: Callback function from DOM event listener

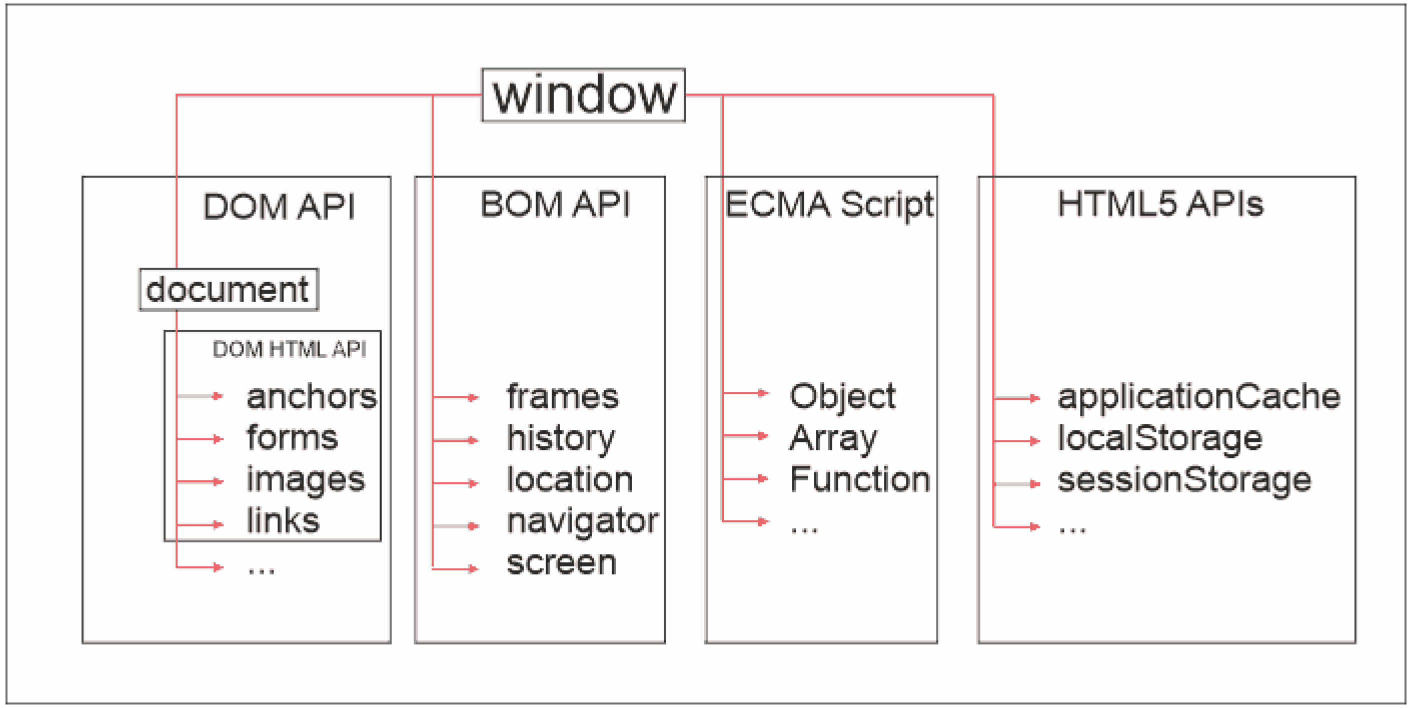
- Engine (V8 in Chrome): parser, JIT interpreter, heap, call stack
- In principle, it runs in a single thread, asynchronous behavior thanks to the browser (event loop, callback queue)
- https://www.youtube.com/watch?v=eiC58R16hb8

DOM (Document Object Model)

- The DOM is an object representation of an HTML document in the browser's memory as a "visualization background".
- Each element (tag, attribute, text) is a node in the tree structure.
- It allows programmatic access to individual page elements – their properties, attributes and methods, including event links.
- Changing the DOM means recalculating and redrawing the page by the browser.
- Virtual DOM
  - solutions in modern frameworks (React, Vue, ...)
  - layer between the DOM and the application logic
  - solves the optimization of changes in the DOM over time – it makes only those changes that are really necessary
- https://www.w3schools.com/js/js_htmldom.asp

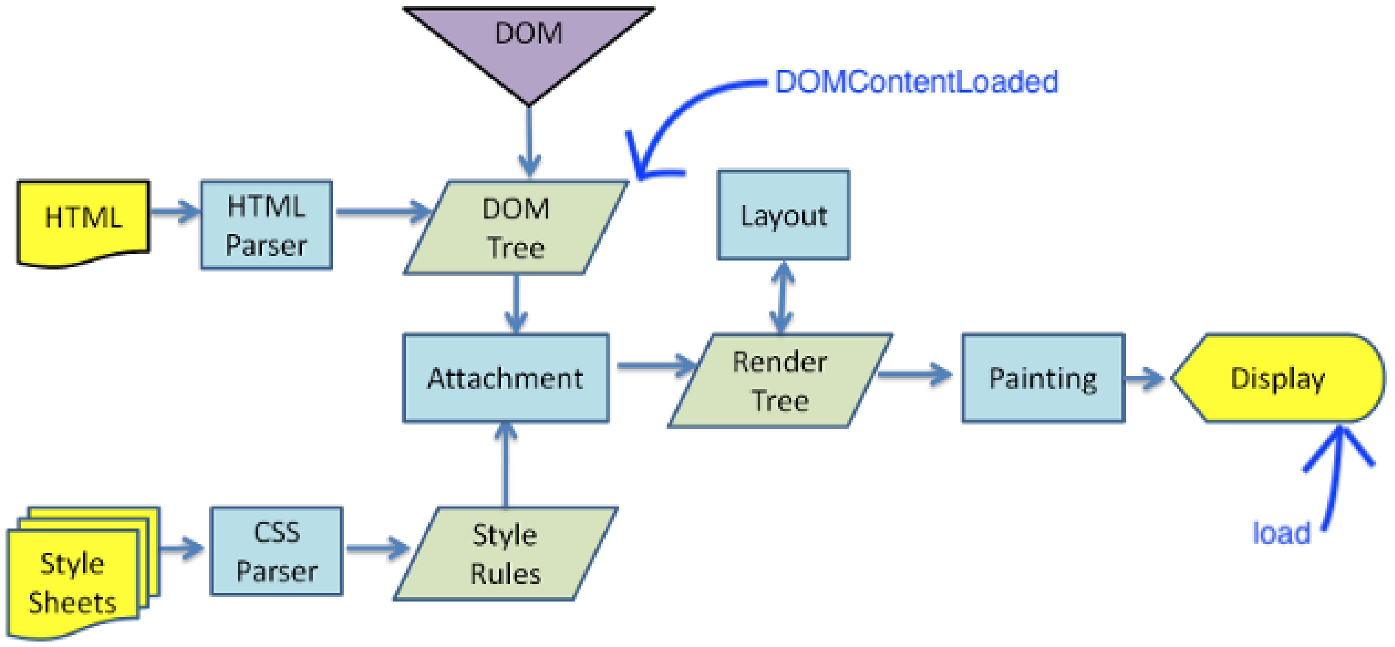
https://www.on-design.de/tutor/javascript

Basic objects



- Object `window` is the basic object of the browser environment – a "global container" for everything that happens on the page. In other environments, its role is fulfilled by other objects e.g. `global`.
  - Global variables and functions
  - Access to browser (HTML5) APIs
  - allows communication between windows or iframes
  - Provides information about the size, position, and status of the window
  - Allows you to work with window-level events

DOM Loading Process



```
window.addEventListener("load", (event) => {
  console.log("page is fully loaded");
});

document.addEventListener("readystatechange", (event) => {
  console.log("page DOM is ready in different states");
});

document.addEventListener("DOMContentLoaded", (event) => {
  console.log("page DOM is loaded with defered scripts etc.");
});
```

Object document

Events of individual elements

- each activity raises an event and it propagates through the tree (DOM) and can be captured by any element (addEventListener, onX, in HTML)
- Propagation method – capturing, bubbling
- interrupt - stopPropagation, preventDefault

Querying within the DOM (recursively)

- getElementById
- getElementsByTagName
- getElementsByClassName
- querySelector, querySelectorAll

Creation and modification of the DOM

- innerText, innerHTML
- createElement, createTextNode
- appendChild

Asynchronous programming

Switching operations and Event Loop

- No thread blocking for time-consuming operations
- fetch, setTimeout, addEventListener
- Event Loop handles the message queue – tasks are executed sequentially as soon as the main thread is free
- The order of processing cannot be precisely controlled, it depends on the completion time and priority of the tasks.

Callback Function

- passing a function/method as an argument to another function and then calling it
- Callback Function Chaining (Callback Hell)
- Immediate execution of passed methods, only one invocation

Promise objects, async/await

- Functional approach - future promise of return/value
- possibility of chaining and easier catching of errors (from the entire chain), multiple execution
- async/await – automation of Promise constructs, written synchronously, but works asynchronously

Web Worker

- allows you to run the algorithm on the next thread

ECMAScript

- ECMAScript (ES) is a standardized specification for JavaScript.
- It specifies how the language should work – its syntax, data types, operators, objects, functions, etc.
- JavaScript = implementation of this standard in the browser (e.g. V8, SpiderMonkey, JavaScriptCore).

- Turning Year 2009 – ECMAScript 5 – The Foundation for the HTML 5 API
- Modern JavaScript – ES6+ (ES2015) – (let, const, class), modules, arrow functions, Promises ... async/await

https://tc39.github.io/ecma262/

HTML 5 API (Web API)

LocalStorage, SessionStorage

- Storing data locally in the browser within the application
- Key/Value Pair

IndexedDB

- database storage based on the principle of indexed objects – a larger amount of structured data
- formerly also WebSQL

Service Worker

- Basic object for offline web application solution
- an algorithm running outside the main thread and providing caching of network communication and other functionalities (PWA)

Web Worker

- Ability to run algorithms on a separate thread
- It has a number of limitations and is isolated – passing data, running, passing the result back

Web Socket

- Advanced interface for two-way asynchronous communication (client-server) using an open communication channel (RFC 6455)
- necessary support on both sides of the communication (HTTP, TCP)

HTML 5 API (Web API)

Drag and drop

- it is possible to move any element in the DOM – draggable attribute
- Implementation of ondragstart, ondrop, ondragover events

File API

- Move an object from your local computer to Web page space
- is based on the Drag and Drop principle, where the "ondrop" event is captured on the corresponding element
- access to transferred files using DataTransfer.files
- File API provides File, FileList, Blob, FileReader, URL Base64

Fetch API

- modern network communication based on the principle of Promises
- compared to the XHR (AJAX) approach, which used callbacks
- better support for JSON, CORS, etc.

Geolokalizace

- the ability to obtain GPS (latitude, longitude, altitude, accurancy, speed, timestamp) coordinates of the user's location
- Subject to user permission and secure connection
- depending on the technical capabilities of the device, or localization using an Internet connection
- API accessible in navigator.geolocation

https://web.dev/articles/devices-introduction

JavaScript frameworks

These are JavaScript libraries that simplify and extend the capabilities of standard JavaScript. Thanks to the JS framework, the developer can focus more on the solution itself, not on optimization and debugging for different browsers, concepts, etc.

- These are basically scripts (written in JS) that extend existing objects, methods, etc.
- They usually have the ability to use and choose from a large number of plugins or principles that already deal with the usual activities and functions (animation, AJAX, DOM, etc.).

Library

- provides a set of functions that are called by the developer (e.g., jQuery, D3.js)

Framework

- defines the structure of the application and calls the developer's code (e.g. React, Angular, Vue)
- Comprehensive development of modern web applications
- Benefits: faster development due to reuse of components, less worry about browser compatibility, maintainable modular code, integration of advanced tools (routing, state management, build), active community and plugin ecosystem