**JavaScript Engine and run time**

**Definition:** JavaScript Engine: A computer program that executes JavaScript code

**Example:** Google V8 is the most well-known engine, it powers Google Chrome and Node.js

Chart

Description automatically generated with medium confidence

**Components of JS engine:**

**Call Stack** – Where code is executed (in the ***execution contexts)***. Also stores **primitives**

**Heap –** Unstructured memory pool which stores all the **objects** that application needs

**Diagram

Description automatically generated**

**Compilation vs Interpretation:**

Ultimately, computers only understand 0s and 1s. Therefore code has to be converted into machine code, using compilation or interpretation

**Process:**

**Compilation** – Source code is converted into machine code **at once.** The machine code is written into a portable file that can be executed on any computer.

A picture containing text

Description automatically generated

**Interpretation** – Interpreter runs through source code and executes it run by line.

Table

Description automatically generated with medium confidence

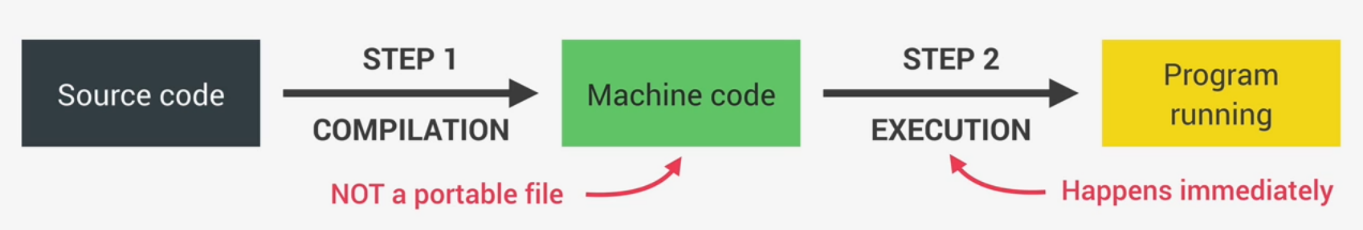
**Legacy JavaScript (interpretated)**

JavaScript used to be a purely interpreted language. Interpretation is much slower than compilation, so in the past this wasn’t a problem. Nowadays, low performance is no longer acceptable.

Nowadays, modern JS engine used a mix between interpretation and compilation -> **Just-in-time (JIT) compilation.**

**JIT**

* Code is compiled into machine code and then executes the machine code right away.
* No portable file
* Execution of machine code happens **immediately after** compilation



**Modern Just-In-Time Compilation of JavaScript**

* Details aren’t so important here

**Diagram

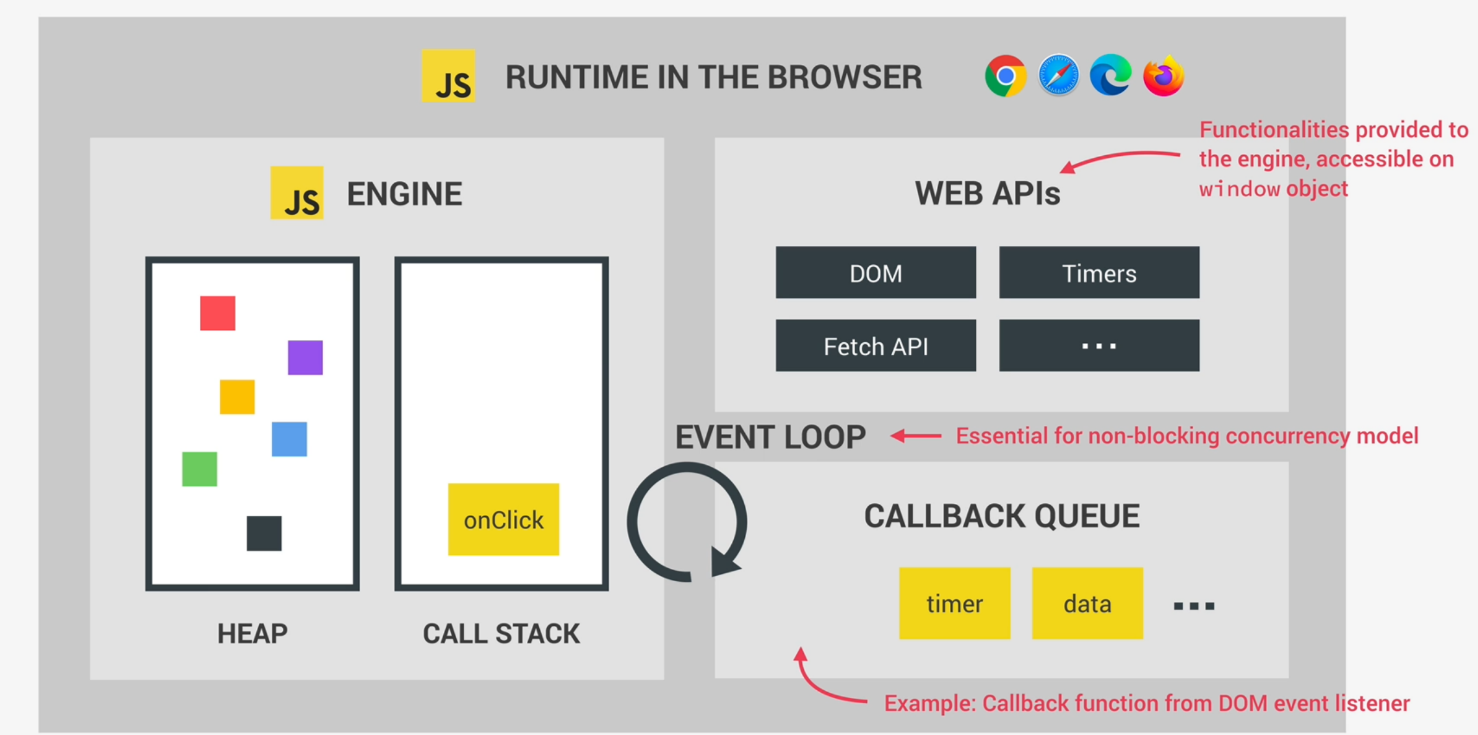
Description automatically generated**

**The Bigger Picture: JavaScript Runtime**

The JavaScript runtime can be imagined as a container which contains all the things necessary into order to use JavaScript in the browser

**Components of Runtime:**

* **JS Engine** – The core of the runtime
* **Web APIs:**
  + Access to Document Object, timers, fetch API, console.log – functionalities provided to the engine, but they are **not a part** of the engine or vanilla JavaScript
  + JavaScript gets access to these APIs through the **global window object**
* **Callback queue**:
  + **Data structure** that contains all the callback functions that are ready to be executed.
  + For example, **handler functions** (callback functions) on an **EventListener** are **waiting for a mouse click** to be executed.
  + On execution of EventListener, the callback function is **put into the callback queue**.
  + When **the Call Stack is empty**, **the callback function is passed to the stack** so that it can be executed, through the **event loop**
  + **Event loop** is essential for non-blocking concurrency model

****

**Components of Runtime in Node.js:**

* Main point here is that Node.js **doesn’t have access to Web APIs**, hence no access to **global window object**, document object, console.log, fetch API etc…

**Graphical user interface, diagram

Description automatically generated**