How JavaScript Works Behind the Scenes

Amr A Khllaf

February 12, 2025

Contents

1 An High-Level "Overview" of JavaScript (Lecture 1)

1.1 Basic Definition

1.1.1 javascript is a high level, object-oriented programming , multi-paradigm language

---

2 JavaScript: is a high-level, prototype-based object-oriented, multi-paradigm, dynamic language, interpreted or "just-in-time" compiled language with first-class functions, single-threaded, non-blocking event loop, and a garbage collector for automatic memory management.

2.1 High-level:

It is a high-level language, which means it is closer to human language than machine language. It is easy to read and write. It is not necessary to remember the system architecture or manage memory.

2.2 Garbage collector:

It is a garbage-collected language, which means it has a garbage collector that automatically allocates and deallocates memory. This means that you don't have to worry about memory management.

2.3 Interpreted or "just-in-time" compiled language with first-class functions:

It is an interpreted language, which means that it is executed line by line. However, it is also a "just-in-time" compiled language with first-class functions, which means that it can be compiled into machine code at runtime. This makes it faster than other interpreted languages.

2.4 Multi-paradigm:

It is a multi-paradigm language, which means that it supports different programming paradigms, such as object-oriented, functional, and imperative programming.

2.4.1 Paradigm

An approach and mindset of structuring code, which will direct your coding style and technique.

1. Object-oriented programming (OOP)
2. Functional programming (FP)
3. Procedural programming (PP)

2.5 Prototype-based object-oriented:

It is a prototype-based object-oriented language, which means that it uses prototypes to create objects. This is different from class-based object-oriented languages, such as Java or C++, which use classes to create objects.

2.6 Dynamic language:

It is a dynamic language, which means that it is flexible and can change at runtime. This means that you can add or remove properties from objects, change the type of variables, and even change the behavior of functions.

2.7 First-class functions:

It is a language with first-class functions, which means that functions are treated as first-class citizens. This means that functions can be assigned to variables, passed as arguments to other functions, and returned from other functions.

2.8 Single-threaded:

It is a single-threaded language, which means that it can only execute one task at a time. This is different from multi-threaded languages, such as Java or C++, which can execute multiple tasks at the same time.

2.8.1 Concurrency Model :

- how the JavaScript engine handles multiple tasks happening at the same time, we need that because JavaScript is single-threaded (This is mean that it can only execute one task at a time).
- So What about the long -running tasks? like fetching data from the server, or reading a file from the disk, or waiting for a user to click on a button, or waiting for a timer to finish.
- This wil be handled by using the `Event Loop` and `Callback Queue`.

2.9 Non-blocking event loop:

It is a non-blocking event loop, which means that it can handle multiple tasks at the same time without blocking the main thread. This is done by using asynchronous functions, such as setTimeout or fetch, which allow the main thread to continue executing while waiting for the asynchronous function to finish.

---

3 The JavaScript Engine and Runtime (Lecture 2)

3.1 What is the JavaScript Engine?

- The JavaScript engine is a program that executes JavaScript code.
- It is responsible for parsing and executing JavaScript code.
- The most popular JavaScript engine is V8, which is used in “Google Chrome and Node.js”.

3.2 Js Engine Components:

1. Memory Heap: where the memory allocation happens.
2. Call Stack: where the code is executed.

3.2.1 Memory Heap (Unstructured):

- This is where the memory allocation happens.
- This is where objects, variables, and function calls are stored.
- This is where the garbage collector works.
- The memory heap is a large region of memory that is used to store objects, variables, and function calls.

3.2.2 Call Stack (Structured):

- This is where the code is executed Using the `Execution Context`.
- This is where the function calls are stored.
- This is where the function calls are executed in a last-in, first-out (LIFO) order.
- The call stack is a data structure that stores information about the active function calls in a program.

3.3 How the code is Compiled to machine code?

3.3.1 Compilation Vs Interpretation:

- Compilation: The entire source code is converted into machine code at once, and written to a binary (Portable) file that can be executed by a computer.
    - step 1 : Parsing the code : Converting the code into a data structure (AST) that the computer can understand.
        - Abstract Syntax Tree (AST) : A tree representation of the code that the computer can understand.
        - AST doesn't have a relation with the DOM tree.
    - step 2 : Compilation to a machine code (Portable file)
    - step 3 : Execution : Running the machine code directly (Program running)
- Interpretation: The source code is executed line by line.
    - step 1 : Parsing the code : Converting the code into a data structure that the computer can understand.
    - step 2 : Execution line by line

3.3.2 Just-in-time compilation (JIT):

- The code is compiled to machine code before it is executed.
- The code is executed line by line.
- JavaScript is an interpreted language, but it is also a "just-in-time (JIT)" compiled language, which means that it can be compiled into machine code at runtime.

- just-in-time compilation: The code is compiled into machine code at runtime, which makes it faster than other interpreted languages , (This is mixed between compilation and interpretation)
- 1. Parsing the code
- 2. Compilation to machine code (Just-in-time)
- Machine code isn't a portable file, it's a code that can be executed directly by the computer.

- After the Execution of the code, the machine code will be stored in the `Call Stack` and the `Memory Heap`and it will be optimized by the `Optimization Engine` to make it faster.
- This happens in the `V8 Engine` which is used in Google Chrome and Node.js.
- So this is mean the Js isn't an interpreted language, it's a compiled language, but it's compiled at runtime which is called "Just-in-time compilation".

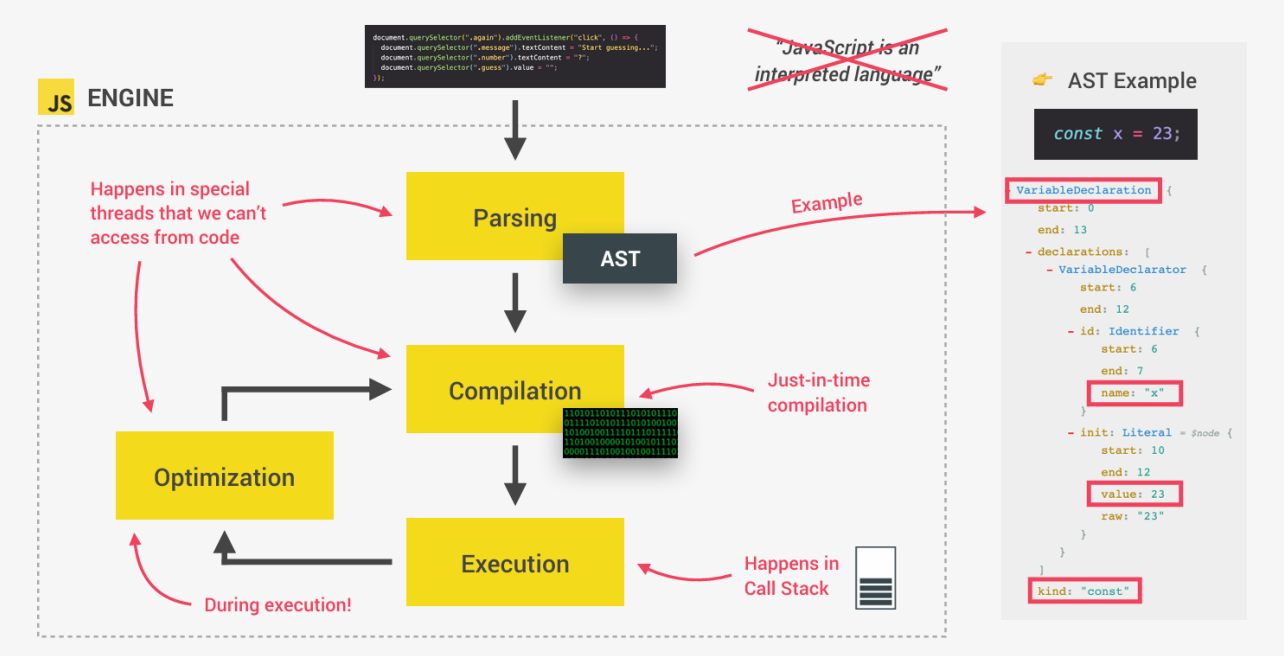

Figure 1: Modern JIT Compilation Of JS

3.4 JS Runtime in Browsers:

- The JavaScript engine is not the only component of the JavaScript runtime in the browser.
- The JavaScript runtime in the browser consists of the following components:
  1. The JavaScript engine (V8 in Chrome).
  2. The Web APIs (DOM, AJAX, Timeout , Timers , Fetch API, ...).
  3. The Callback Queue.
     - The Callback Queue is a queue that stores the callbacks of asynchronous functions.
     - like : setTimeout, fetch, DOM events, click events, ...
     - The Callback Queue is also known as the Task Queue.

4. The Event Loop.
    - The Event Loop is a loop that checks the Call Stack and the Callback Queue.
    - If the Call Stack is empty, it takes the first callback from the Callback Queue and pushes it to the Call Stack.

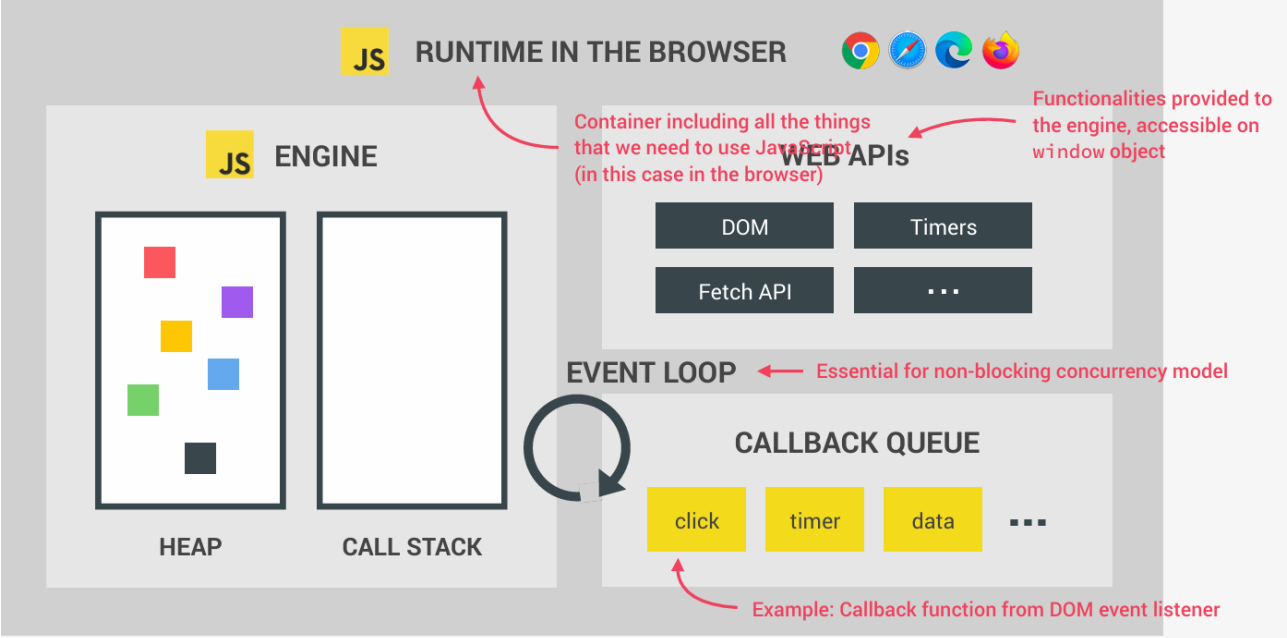

Figure 2: JS Runtime in Browsers

3.5 JS Runtime in Node.js:

- The JavaScript runtime in Node.js is similar to the JavaScript runtime in the browser.
- The JavaScript runtime in Node.js consists of the following components:

1. The JavaScript engine (V8 in Node.js).
2. The Node APIs (File System, HTTP, OS, ...).
    - C++ bindings that allow JavaScript to interact with the operating system & Thread pool.
3. The Callback Queue.
4. The Event Loop.

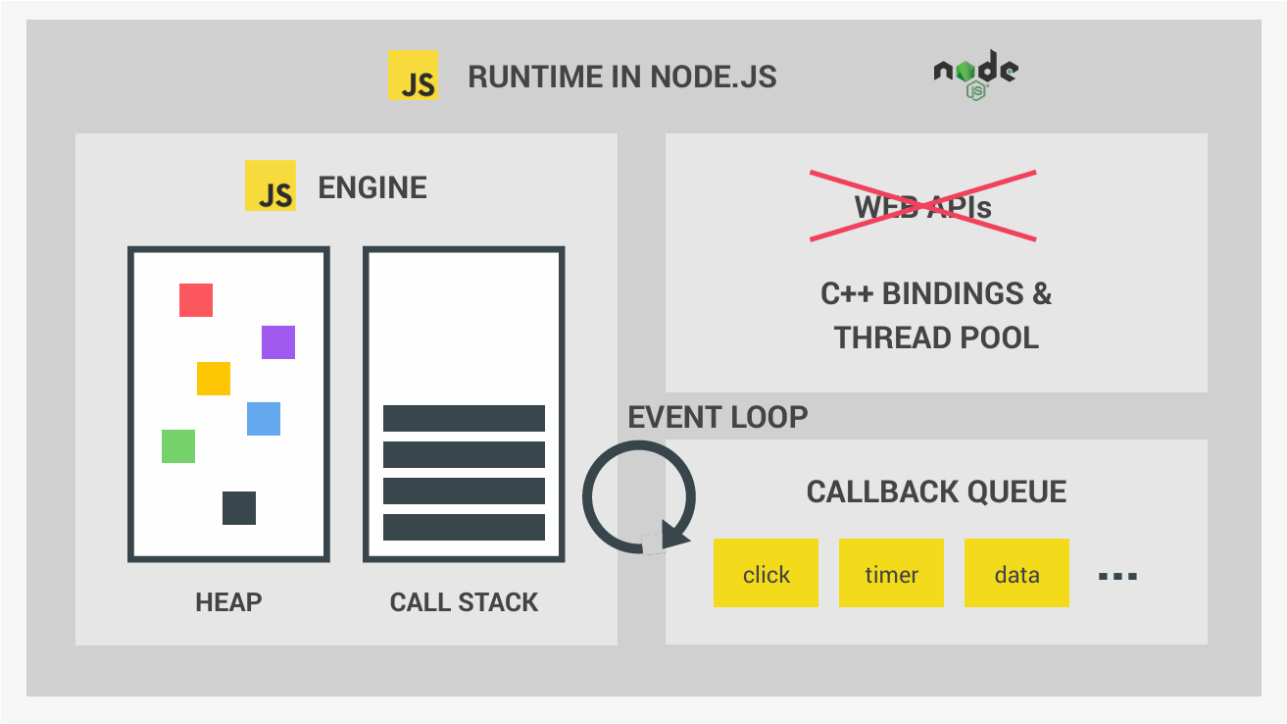

Figure 3: JS Runtime in Node.js