# JS Engine and Google's V8 Architecture

Saturday, 19 August 2023 2:37 PM

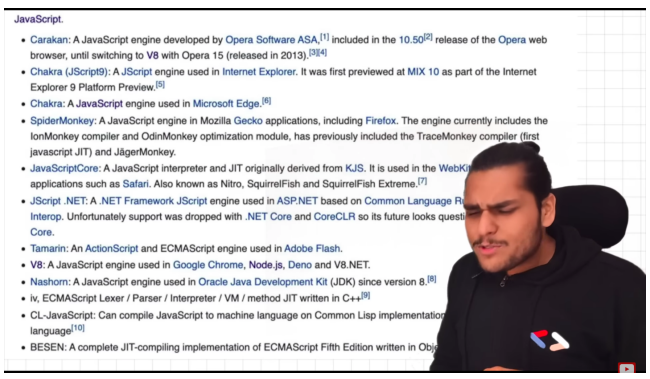
→ JavaScript Runtime Environment: It is like a big container which has all the things required to run JS code. It contains:

- JS engine
- Set of APIs to connect to the outer environment.
- Event loop . Callback queue . Microtask queue.

→ Javascript runtime environment is not possible without a JavaScript engine.

→ Browsers can only execute JS code because they have JS runtime environment.

eg: Node.js is a JS runtime.

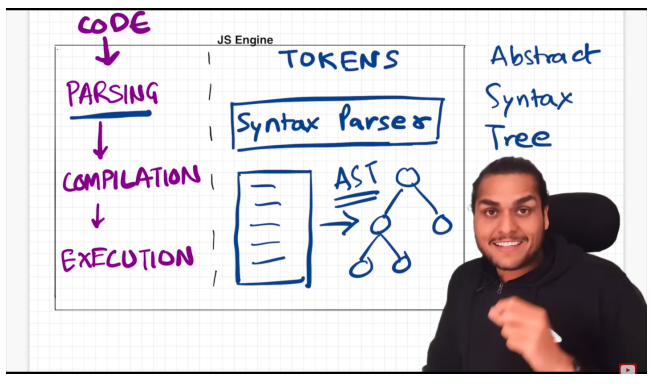



→ List of JavaScript engines

Note: ECMAScript is the governing body of the JS language.

Note: SpiderMonkey was the first JS engine created by the creator of JS Brendan Eich at Netscape communications which is now Mozilla Firefox.

Note: Google's V8 engine is written in C++.

→ 


JS engine makes the code go through 3 major steps.

1) Parsing 2) Compilation 3) Execution

→ Parsing: During this step the code is broken down into tokens. We have a syntax parser here which takes the code & converts it into AST or an abstract syntax tree.

→ Compilation & Execution phase : these 2 phases go hand in hand.

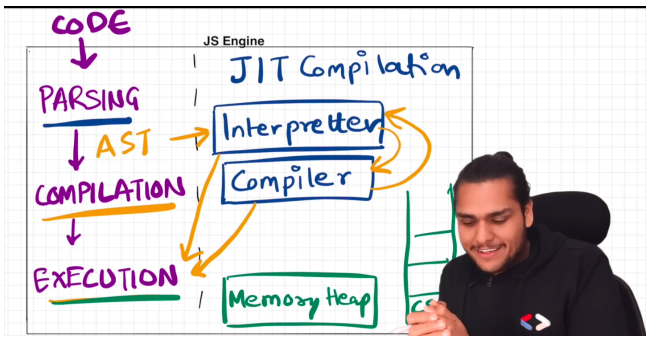
Interpreter : It executes the code line by line. It has faster execution.
Compiler : It takes the whole code & compiles it to give an optimized version of that code. It has better efficiency.

Note : JS can behave both as a compiled & an interpreted language.
Initially it was an interpreted language as browsers wouldn't wait for for compilation. But today most modern JS engines use an interpreter + compiler together. So, today it depends on the JS engine, whether JS is purely interpreted or just in time compiled.

The ability of JS engine to use an interpreter + compiler together makes JS a just in time compiled language. This is JIT compilation.

→ The AST received after parsing stage is then sent to the interpreter, now interpreter converts our high level code into byte code & that code moves to the execution step. While this is happening, the interpreter takes the help of the compiler to optimise this code. So, the compiler talks to the interpreter & as the code is being interpreted line by line, the compiler tries to optimize it as much as it can on the runtime, that's why it is called just in time compilation. This is a multiple phase process & all JS engines have their own algorithms of doing it.

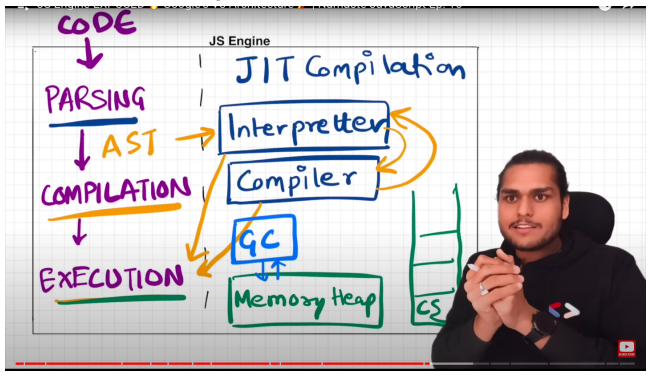
Note : In some JS engines we have AOT compilation i.e ahead of time compilation i.e it takes a piece of code that is to be executed later and tries to optimize it as much as it can and it also produces byte code which then goes to the execution phase.



→ Now execution stage is not possible without 2 main components : 1) Memory heap 2) Call stack

→ Memory heap is where the memory is stored, it is always in contact with the call stack, garbage collector, etc.

→ Garbage collector tries to free up memory space wherever possible.




It uses an algorithm known as mark and sweep algorithm to do so. This algorithm basically reduces the definition of "an object is no longer needed" to "an object is unreachable".

→ Mark and Sweep algorithm: This algorithm consists of 2 phases:

1) Mark Phase: All objects are marked as 0 initially (at creation) & during Mark phase the objects that will be accessible are marked as 1 (reachable) by a DFS graph traversal.

2) Sweep Phase: During this phase, the objects marked as 0 are removed from heap memory, and also reachable objects are again initialised with 0 (made unreachable) because the algorithm will run again.

Note: Compiler does many optimizations on the code, eg:
1) Inlining 2) Copy elision 3) Inline caching

Note: Currently the fastest JS engine is the V8 engine by Google. V8's interpreter is called Ignition and Turbofan is their optimizing compiler. Orinoco is their garbage collector's name. and oilpan

→ Google's V8 JS engine architecture: