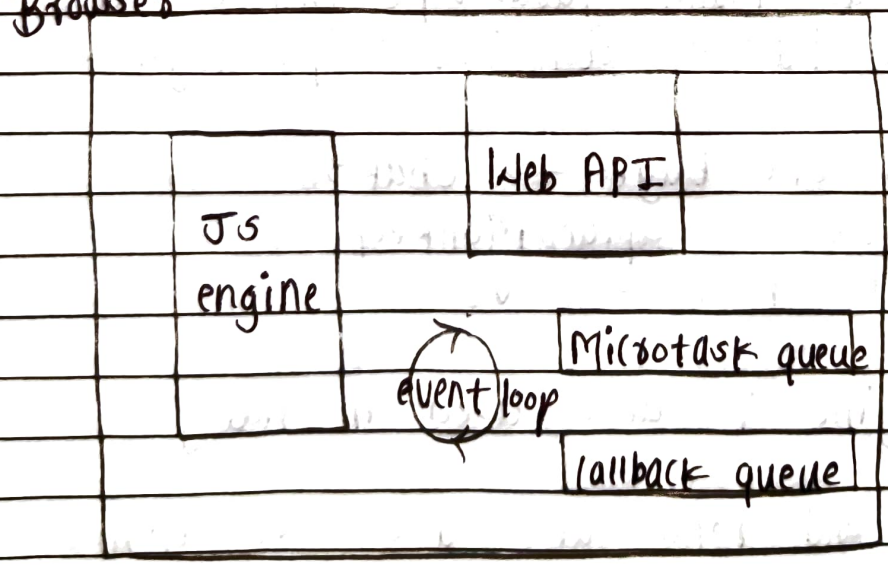
Episode 16

JS Engine Exposed
Google's V8 Architecture

Javascript can run inside a browser, it can run inside a server, it can run inside your smartwatch, it can run inside a light bulb, it can even run inside robots.
And this is all possible because of javascript runtime environment.

* Javascript Runtime Environment

Javascript runtime environment is like a big container which has all the things required to run javascript code.
Javascript runtime environment is not possible without javascript engine.



Page No.
Date

Browsers can only execute javascript code just because it has javascript runtime environment.

Every browsers has its own javascript runtime environment.
(ex. Nodejs can run javascript code outside the browsers)

There could be different API's for browsers and Nodejs. There could also be some same API like console, setTimeout, etc.
(there name is same, but internally they are implemented in a different manner)

Javascript engine is "heart" of javascript runtime environment.

* Some javascript engines
there are many javascript engines available, all browsers has its own javascript engine.

① Microsoft Edge → chakra
② Firefox → spiderMonkey
③ Google Chrome → V8

Nodejs and Deno also uses V8 engine.

We can also create our own javascript engine as well. The most important protocol for a javascript engine is to follow the ECMAscript standards.
ECMAscript is like a governing body of the javascript language.

* First JS Engine ever created in world

first JS engine was created by the creater of javascript, brendon eich.
He created javascript as well as javascript engine while he was working at netscape.
The engine which he created has evolved a lot and now it is called as spidermonkey, which is used inside mozilla firefox browser.

* Myths about javascript engine.

JS engine is not a machine. (it's not like hardware which run javascript code)
JS engine is like a normal program, like a normal code which is written in low level languages.
example :- V8 engine is written inside C++

* JS engine Architecture

JS engine basically takes the code as the input, the human readable code as ... input

This code now goes through three steps. first step is parsing. second step is compilation and third step is execution.

code
↓
parsing
↓
compilation
↓
execution

* syntax parsers and abstract syntax tree

During parsing phase, the code which you write is broken down into tokens.
Example :- let a = 10 , Here "let" is one token, "a" is one token, "=" is one token and "10" is one token.

There is also something called ↑ syntax parsers, the job of syntax parser is basically to take the code and convert that code ... into AST (abstract syntax tree).

Page No.
Date

Example :-

```
const name = "Chaitanya";          // code
```

⇓ converts into AST

Tree like structure

[Website astexplorer.net demo.

Now this AST is passed through the compilation phase.

* Compilation and execution of JS code.

* Just in time compilation (JIT)

javascript has something called just in time compilation.
Before understanding JIT, let's first understand what is interpreter, what is compiler, is javascript interpreted or compiled language. (imp)

interpreter → interpreter basically takes code and starts executing line by line in specific order, it does not know what will happen in next line.

compiler → in case of compiler, whole code is compiled first of all even before executing. and then new code is formed which is optimized version of code.
And then it is executed.

In interpreter, the code is fast, it is fastly executed and it does not have to wait for ___ first.
↓
compilation

In case of compiler, we have more efficiency.

* Is javascript interpreted or compiled language

So javascript can behave like interpreted language as well as compiled language, everything is dependent on ___ Javascript engine.

So initially when javascript was created by brendon eich, then it was supposed to be an interpreted language. The engine which he wrote uses interpreter to execute code, because it use to majorily run on browsers and browser's can't wait for that code to compile before executing.
So, it was interpreted language.

But now, in today's world, most of the modern browsers, most of the JS engines uses an interpreter plus a compiler, both together.
So now it all depends on JS engine, whether it is purely interpreted or it is just-in-time compiled.

JS engine can use an interpreter as well as compiler and this makes it as a just in time compile language.

so JIT is best, because it uses interpreter as well as compiler to execute code.

so now compilation and execution go hand-in-hand.
so after parsing we got AST
and now this AST goes to the interpreter.
so now interpreter converts our code into
high level
byte code and then that code moves to the execution step.
And while it is doing this, it takes help of the compiler to optimised the code
so compiler basically talks to compiler and while code is ~~interpreted~~ line by line, the compiler also basically tries to optimize the code as much it can.
So it is not just one phase process, but it can happen in multiple phases.
All JS engines have their own algorithms of doing this.

so the job of the compiler over here is to optimize the code as much it can on the runtime and that is why it is also known as just in time compilation.

In some javascript engines, there is something known as AOT (ahead of time compilation). In this case, compiler basically takes a piece of code which is going to be executed later and tries to optimize it as much it can and it also produces the byte code which then goes to the execution phase.

The execution is not possible without two major components of javascript engine.
And they are memory heap and call stack.

In memory heap, all the memory is stored.
It is constantly in sync with callstack and the garbage collector.
All variables and function are assigned memory in memory heap.

* Garbage collector - Mark and sweep Algorithm

Now we also have garbage collector into the picture. Garbage collector tries to free up memory space whenever possible.

Page No.
Date

Whenever some function is not been used or we clear setTimeout, so it collects all the garbage and sweeps it.
It uses an algorithm which is known as Mark and sweep algorithm to do this.

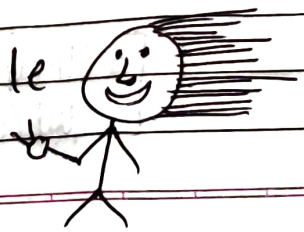
Mark and sweep algorithm is very important and widely used across the garbage collectors out there, not just for the javascript only.

There are other many forms of optimizations which a compiler does for the code and those optimizations are like inlining, copy elision, inline caching, etc. which compiler is doing while compiling the code.

This all process is very genesic. All the JS engine have their own way of implementing things, but most of the modern javascript browsers have something similar to this only.

Maybe the compilation logic inside google's V8 engine is very different from spidermonkey which is inside firefox.
And this is what each company is trying to achieve, they want to make their javascript engine as fast as possible.

V8 Engine of Google

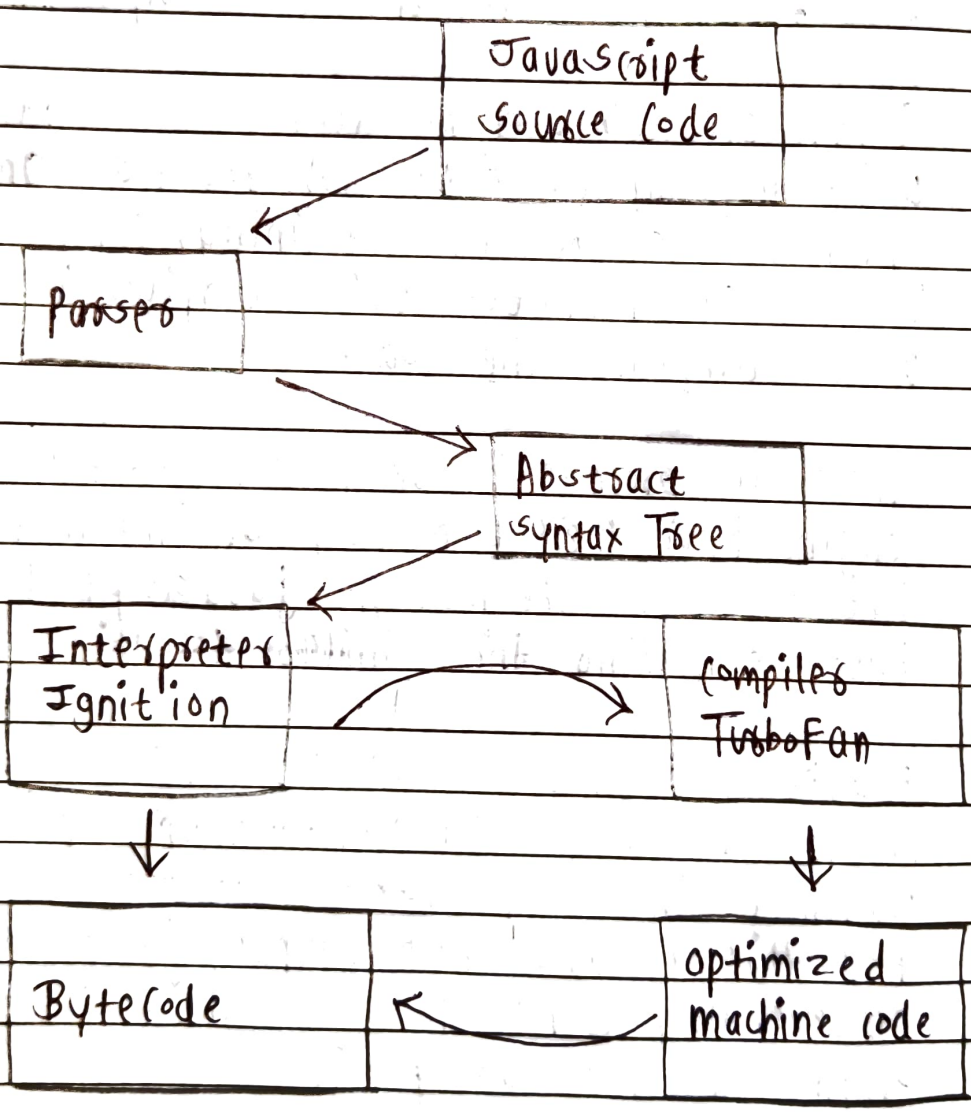


Page No.
Date

* Fastest JS Engine – V8 engine

V8 has a interpreter which is known as Ignition.
V8 also has turbofan which is optimizing compilers.

* Google's V8 JS Engine Architecture



They also has garbage collector called "orinoco" which uses mark and sweep algorithm.

* overview/summary

Javascript runtime environment