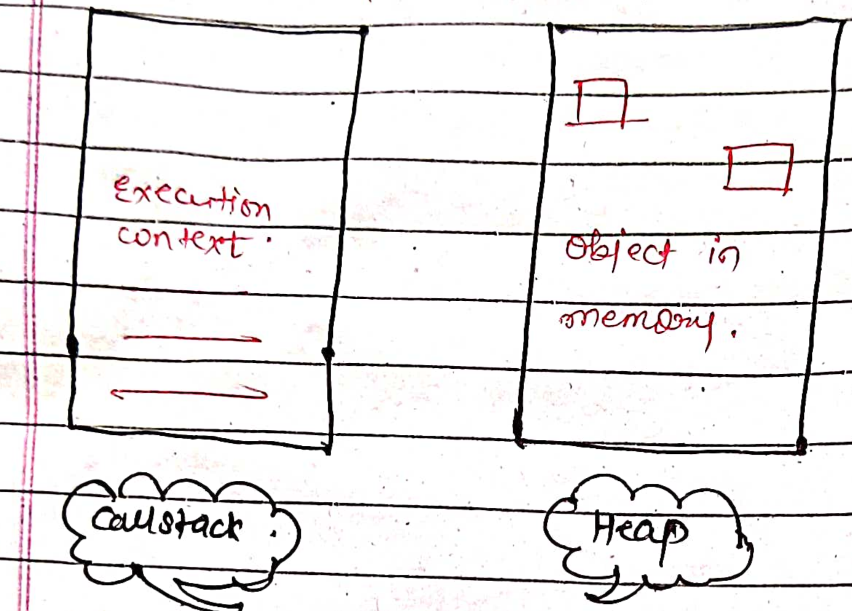
what is Javascript Engine ?

JS Engine :- Program that executes JS code.
E.g V8 & node.js
all other browsers have their own JS eng

It contains callstack &
Heap.



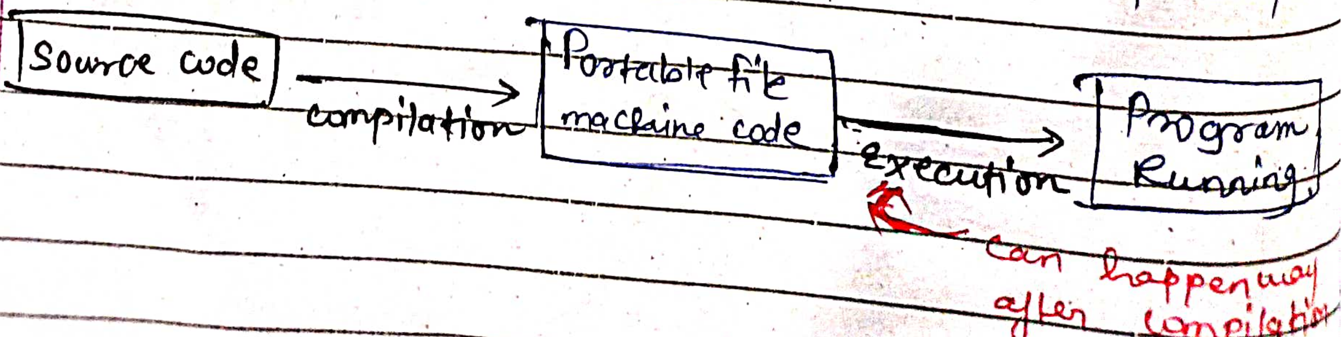
where your code is executed.
where objects are stored.

How the code is compiled to Machine Code ?

Computer only understands 0 & 1 i.e binary long it need't be converted automatically into machine code using compilation & Interpretation.

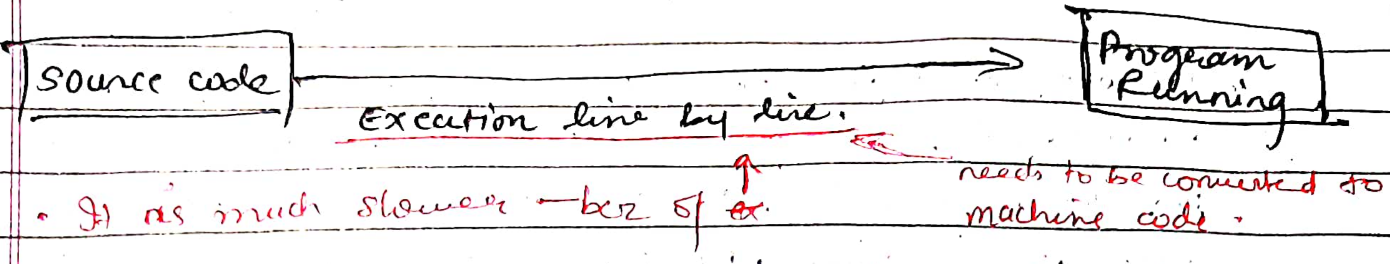
* Compilation :-

Entire code converted → machine code & written to a binary file that can be executed by a computer

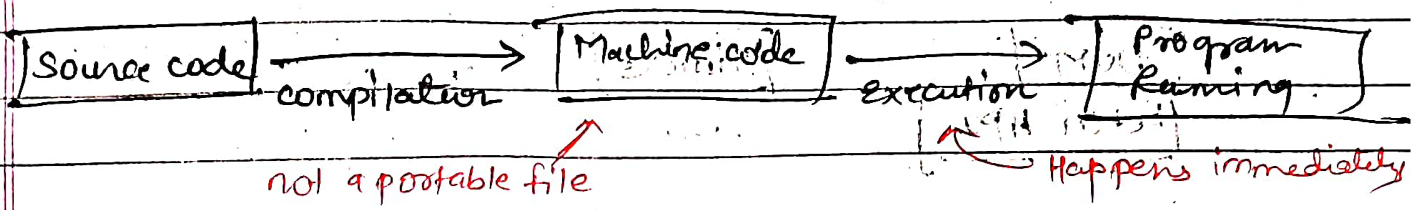


PAGE NO.

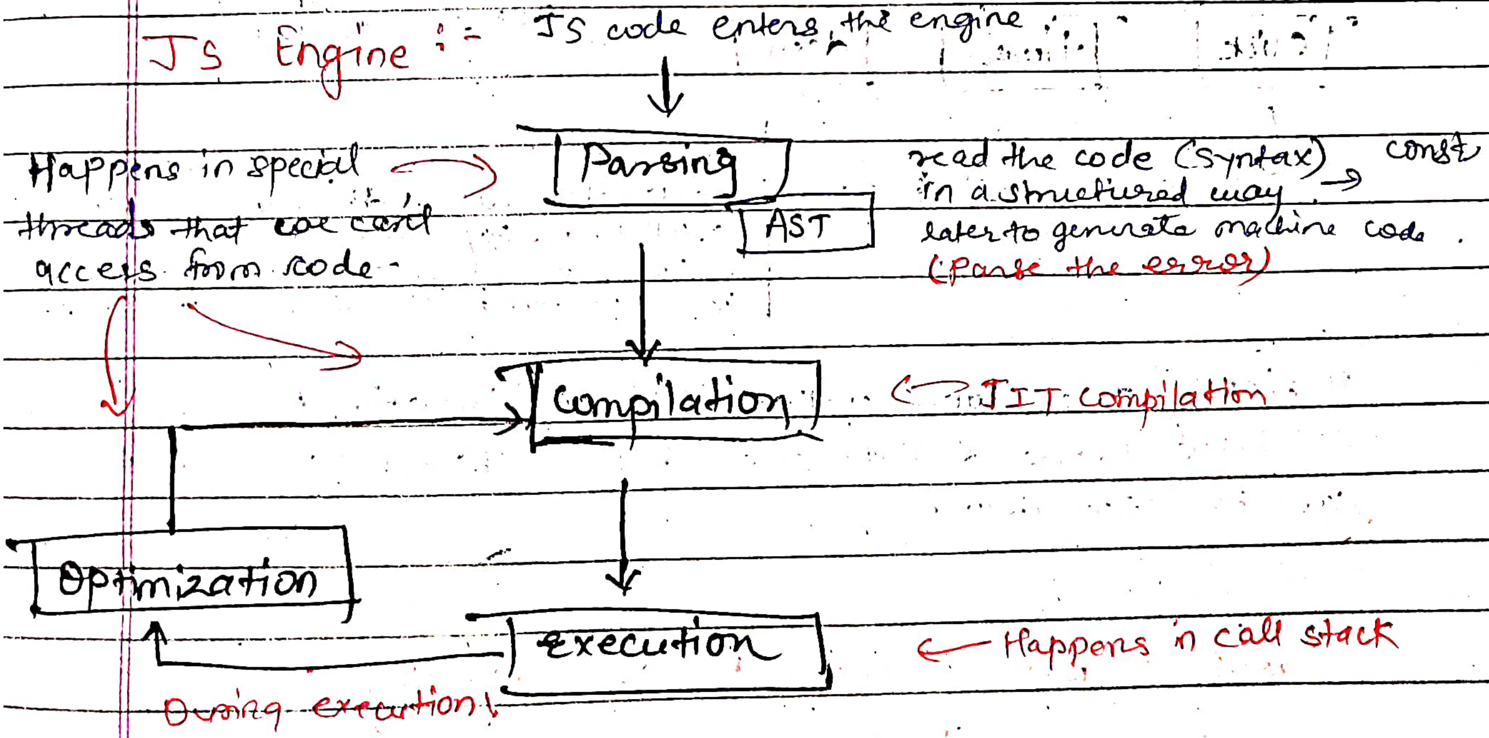
* Interpretation :- Runs through the source code & executes it line by line.


Source code
Program Running
Execution line by line.
needs to be converted to machine code.
• It is much slower bcz of ~~ex~~

* Just-in-Time (JIT) compilation : Entire code is converted into machine code at once, then executed immediately.



JS Engine :-


JS code enters the engine
Parsing
AST
read the code (syntax) in a structured way later to generate machine code
const
(Parse the error)
Happens in special threads that we can't access from code.
Compilation
JIT compilation
Optimization
Execution
Happens in call stack
During execution!

JavaScript Runtime.

JS. Runtime in the Browser.

container including all the things that we need to use JS.

Without JS Engine, their is no Runtime & no JS at all. However, Engine alone is not enough. In order to run properly we also need to access to the web API's. DOM, timer, fetch API, & even console.log. are the part of Runtime.

WEB API's ← Functionalities provided to ~~the engine~~, accessible on window object.

DOM → Timers

Fetch API

...

Callback Queue. ← ~~event handler~~ Callback function from DOM ~~DOM~~ event listner.

click | timer | data

↓ added to call stack.

callback function is put to the callback Queue then when call stack is empty callback function is passed to the stack so that it can be executed this called event loop.

Basically even loop takes callback function from the callback Queue & Puts them into callstack so that they can be executed.

* Event loop – Essential for non-blocking Concurrency model.

What is an Execution Context ?

{ In which piece of JS is executed
Stores all the necessary info for some code to be executed.
Such as local variables.

- Global Execution code
@ (top-level code) ← not inside the function

↓

*(Exactly one global execution context (EC))

Execution of top-level code ~~(inside global EC)~~

↓

Execution of functions & waiting for callbacks.

e.g:- click event callback.

* One execution context per function :- For Each function cal a new execution Context is created.

← All together make the call stack

What's inside Execution Context ?

1. Variable Environment.
   - let, const & var declaration
   - functions
   - Argument object.

2. Scope chain.

3. this keyword.

Not in arrow functions!

↑ Generated during 'creation Phase', right before execution.

* Call stack :- In order to keep track where we are in the program execution @ top & when its finished running it will be removed from the stack. Execution will go back to the previous execution context.

* Note :- JS is single threaded i.e code run one at one time. Call stack ensures that the order of execution never get lost.