DATE : __/__/__
PAGE : ____

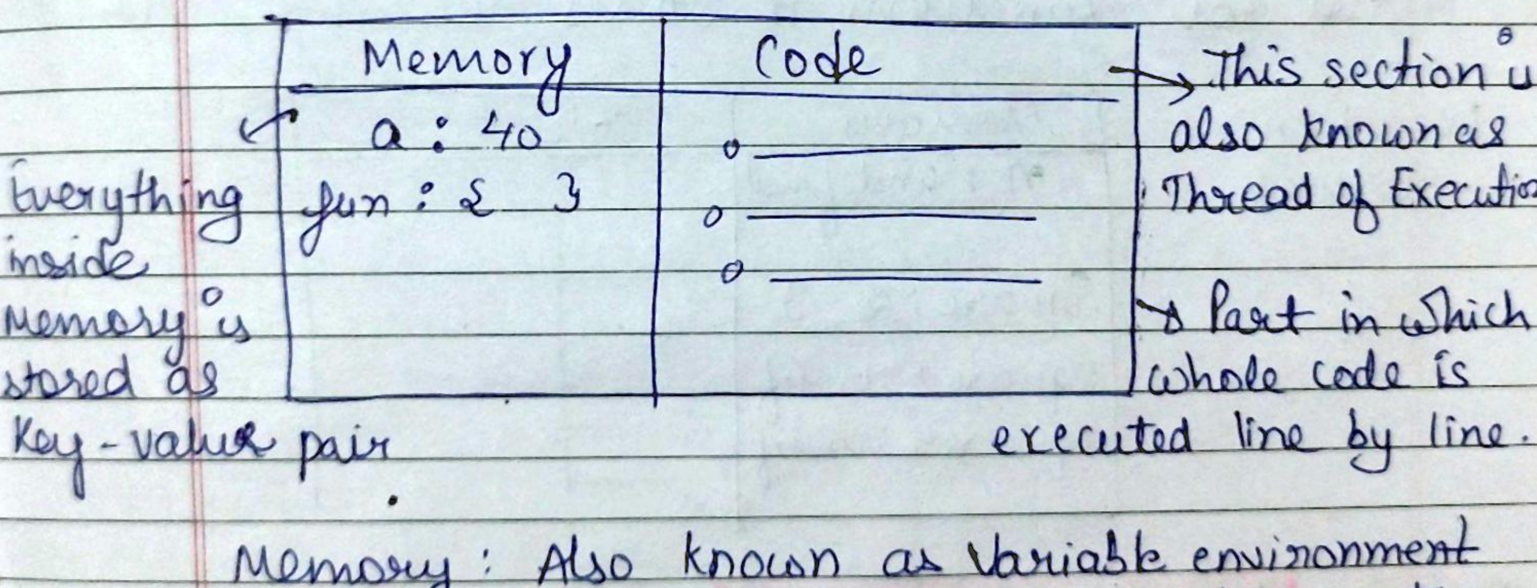
Javascript Working

JavaScript : It is a synchronous single threaded Language

Synchronous :- Specific Order
Single Threaded :- Can execute one Command at a time.

Everything in JavaScript happens inside the execution context.

Execution Context Structure :-


Memory
Code
a : 40
fun : { }
Everything inside Memory is stored as Key-value pair
This section is also known as Thread of Executio
Part in which whole code is executed line by line.

Memory : Also known as Variable environment
Code : Also known as Thread of execution.

- When we run a JavaScript program an Execution Context is created.

DATE: __/__/__
PAGE: ____

Ex code.

```
Var n=2;
function square(num)
{
  return num*num;
}
var Squar2 =square(2);
var Square4= Square(4)
```

Phase 1:- In Memory phase Creation, it allocates Memory for variable and assigned them undefined and for functions it stores the whole code.

Execution Context

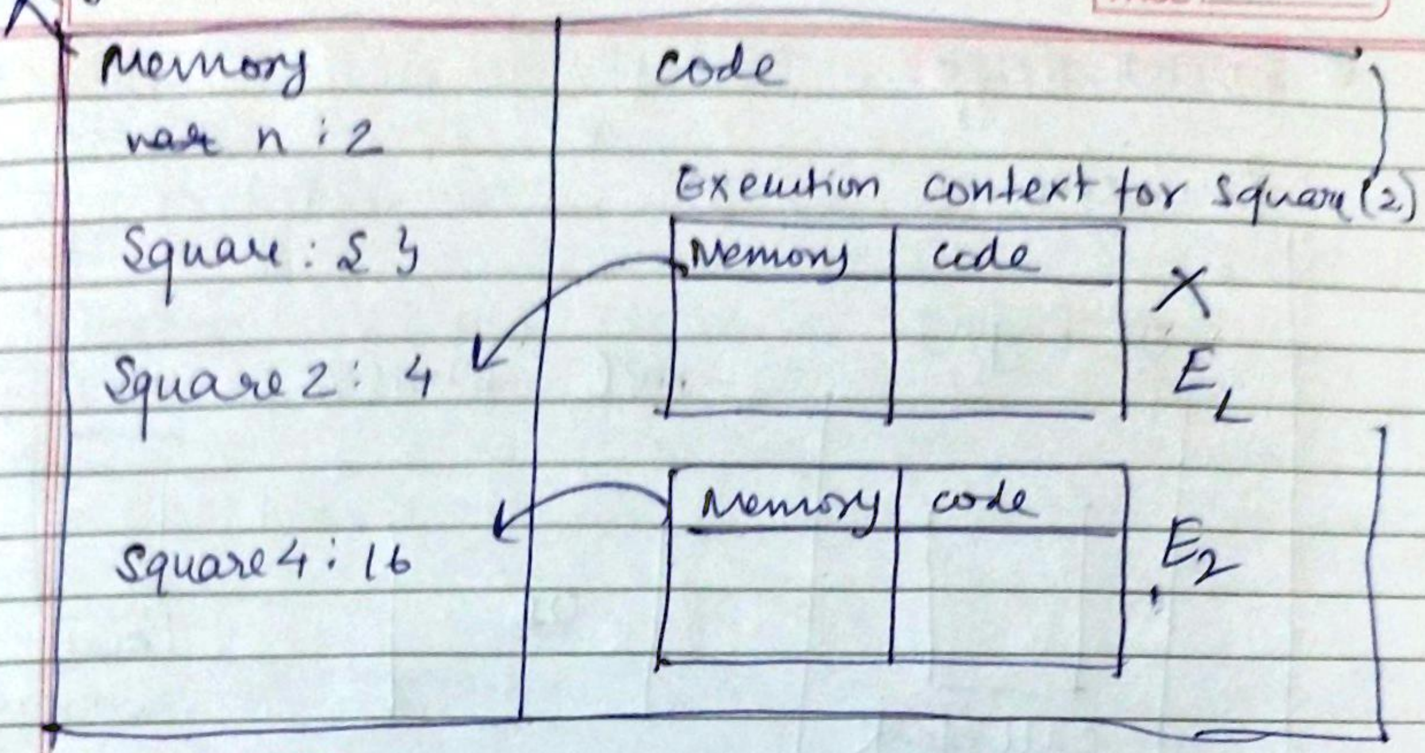
Memory	code
n : undefined Square : { } square2 : undef square4 : undef	

Phase 2:- Code Execution Phase.
In this phase JavaScript code is again run and variables are assigned there values.

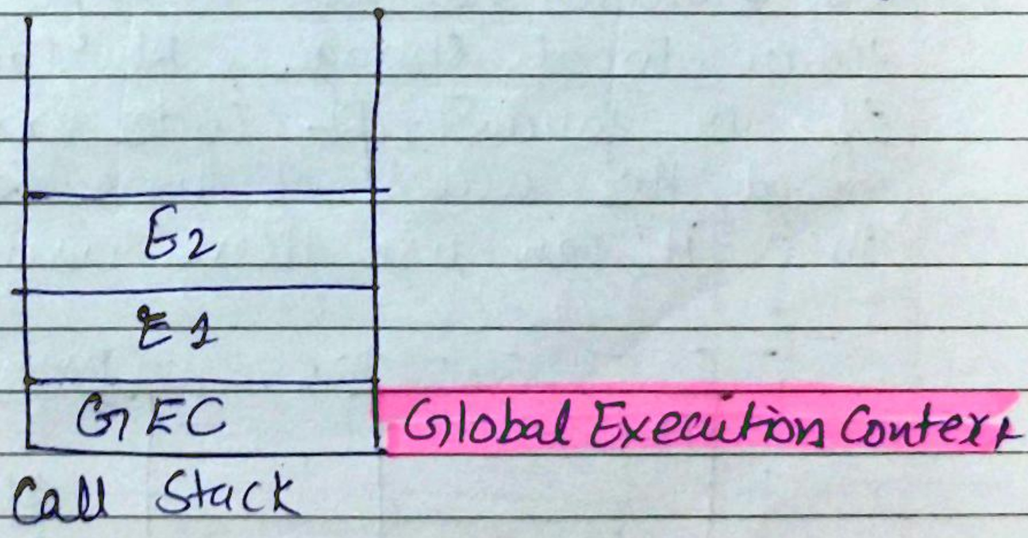
New execution context is created for every function call. And once the function is completed or finished its execution context is deleted.

Scanned by Scanner Go

Global Execution Context

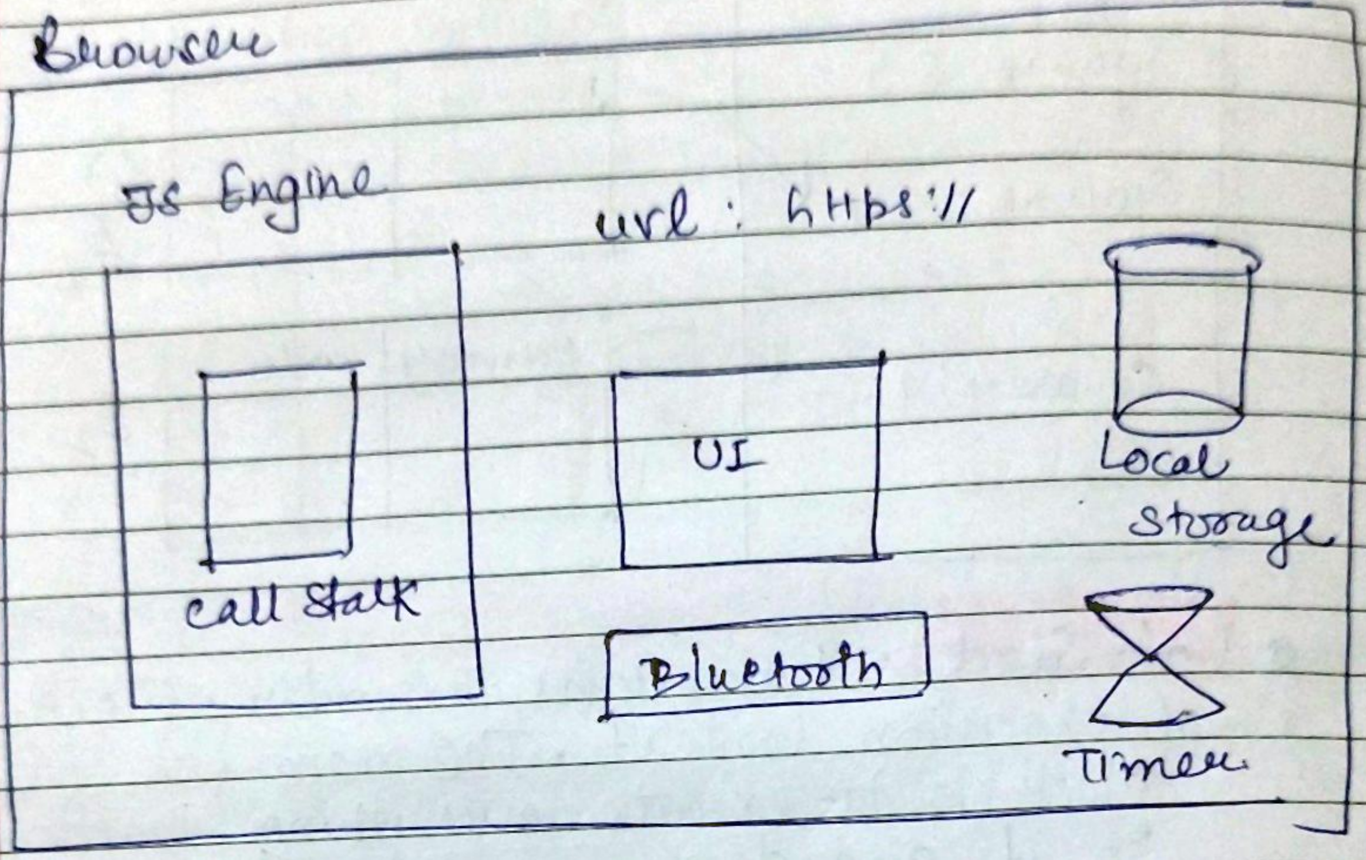


- Call Stack: It maintains the order of execution of execution context. The main job of call stack is to execute everything that comes in it and does not wait for anything

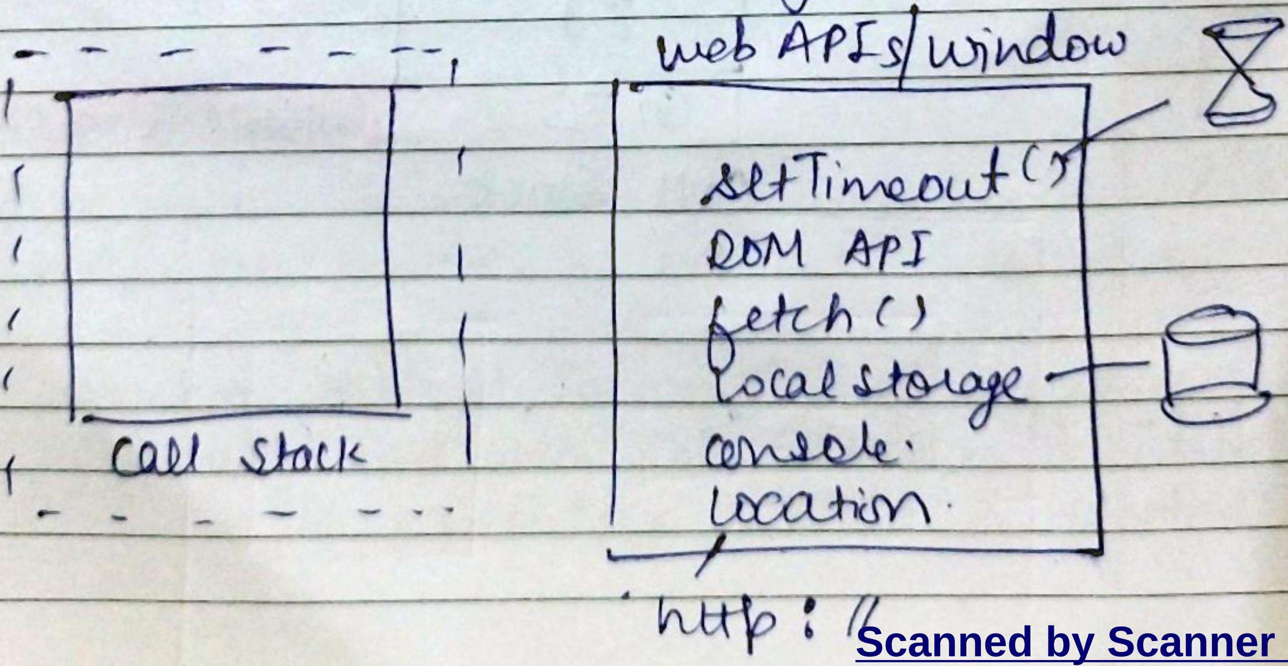


DATE : __/__/__
PAGE :

Event Loop



The Browser has all the superpowers like url, timer, local storage, Bluetooth etc.
If our javaScript code executing in callStack need ~~the~~ access of these ~~call~~ superpowers Then it can use these using web Api's



WebAPI : It is an Application programming interface for the web.

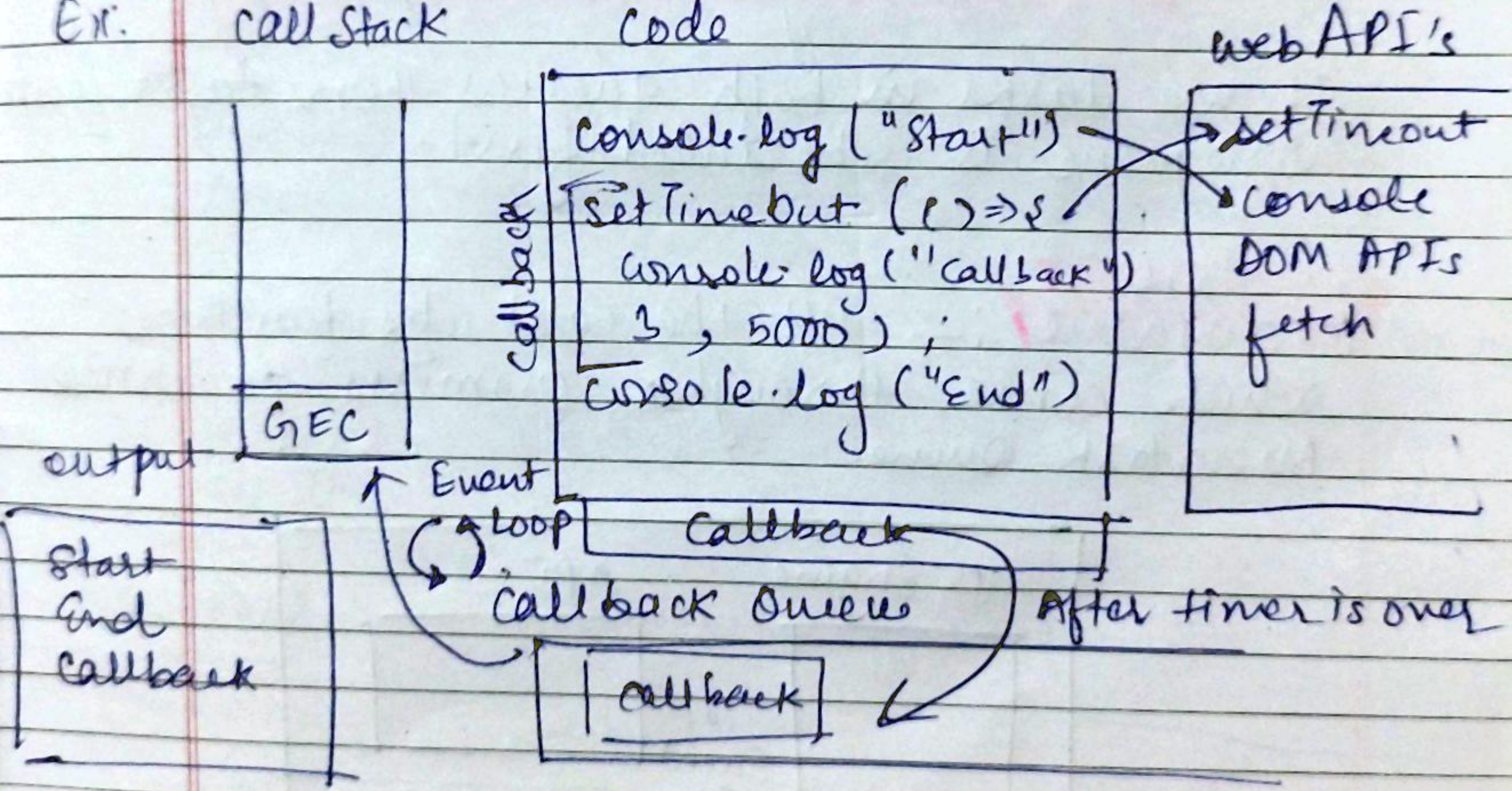
Application

API :- ~~A~~ Refers to the software with distinct function. Inteface refers to any contract of services between two application. This contract defines how two app communicate with each other using requests and response

These all API's are inside the window (global) object so that if we want to use setTimeout we can use it through

window.setTimeout() = setTimeout()

Ex.



```
console.log ("Start")
setTimeout (()=>{
  console.log ("callback")
  }, 5000);
console.log ("End")
```

```
Start
End
Callback
```

DATE : __/__/__
PAGE : ______

Event loop :- The function of the event loop is to put the callbacks from callback queues to callstack. It act as a gatekeeper, it checks whether we have something in callba queue, if we have something then its work is to put it inside the callstack.

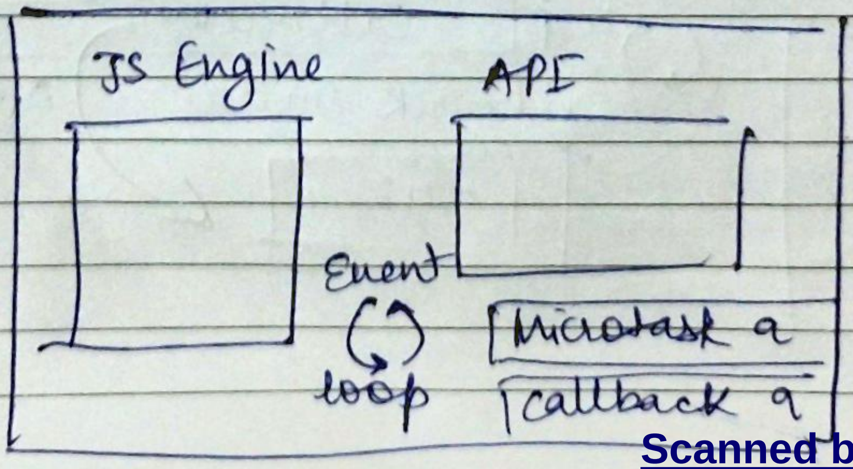
The role is to check whether callstack is empty or not if empty then it schedule the tasks present in callback queue and ~~Microqueue~~ Microtask queue.

Both queues differ in Priority

Microtask Queue > Callback Priority queue

If we tasks in both queues then tasks from Microqueue is scheduled first.

Task Microqueue :- All the callbacks functions which come throughs promises go inside Microtask Queue.

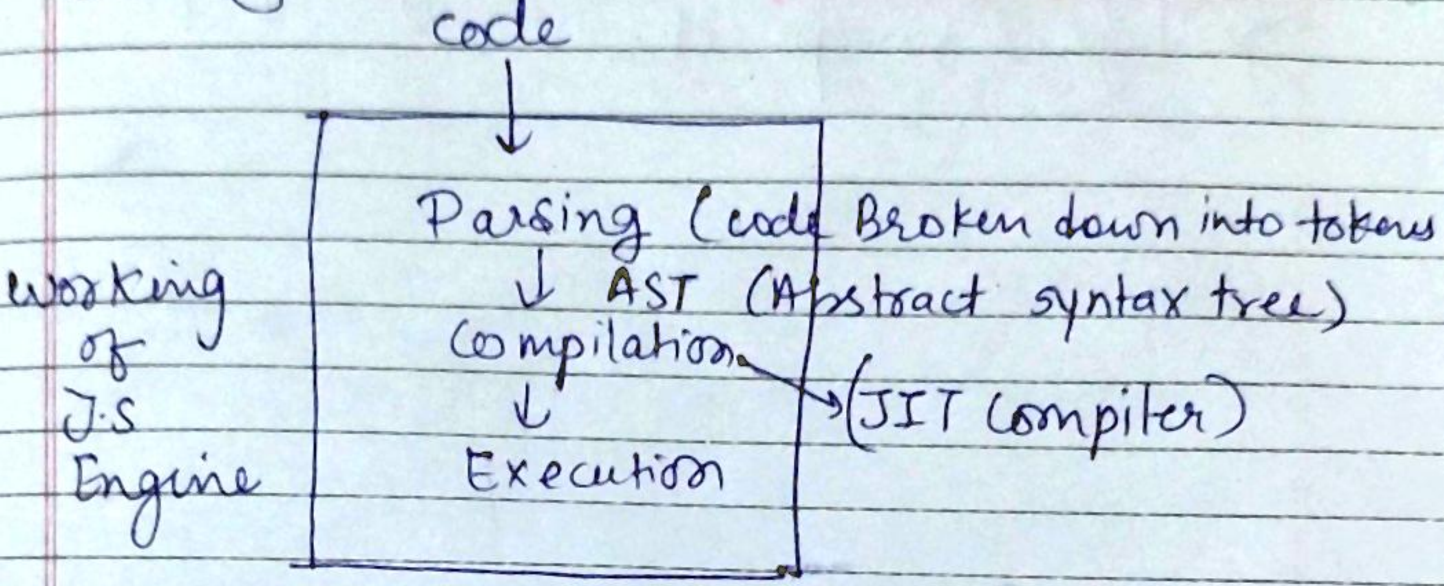


DATE : __/__/__
PAGE : ______

Browser Can Run javaScript Code because i
has javaScript RunTime Environment

Node.js → JavaScript Runtime Environment

JS Engine


code
Parsing (code Broken down into tokens
AST (Abstract syntax tree)
Compilation
(JIT Compiler)
Execution
working of J.S Engine

Interpreter	Compiler
Executes code line by line	Compiler first translates the code from high level language into a machine code and then run/Executes the code
Executes the code written in High level language to machine code line by line	
Fast	Efficient

DATE : __/__/__
PAGE : ______

JIT (Just in time)
JavaScript can ~~Run~~ act as interpreted and compiler language, its behaviour depends on JavaScript Engine.

Garbage Collector: Tries to free up space & collect garbage & sweeps it