Libuv & async IO

episode - 6

As we Know,

JS is a Synchoronous Single Threaded,

which means,

Single Thread → can Run on a Single Thread or a Single Process

Synchronous → one after another.

JS engine or V8 will run the JS code as soon as it is sent to JS engine (very fast execution, quick

TO Run JS we dont need multiple Threads we just need one thread

If we are executing line 3 then line 4 will only be executed after that.

JS is only capable of Running on a Single piece of thread.

Restaurant

		(waiting time) Synchronous	Asynchronous
person (A)	→ COKE	0 min	0 min
person (B)	→ Noodle	5 min	5 min
person (C)	→ Pizza	15 min	10 min
person (D)	→ COKE	15 min	0 min
person (E)	→ Noodle	20 min	5 min

Here, with this example we are going to understand Synchronous & asynchronous execution

* In Synchronous Execution,

Next order can only be fulfilled once the previous order is Fullfilled.

Generally not a good way & is Blocking

* In async execution

All people with Coke will get it at 0 min

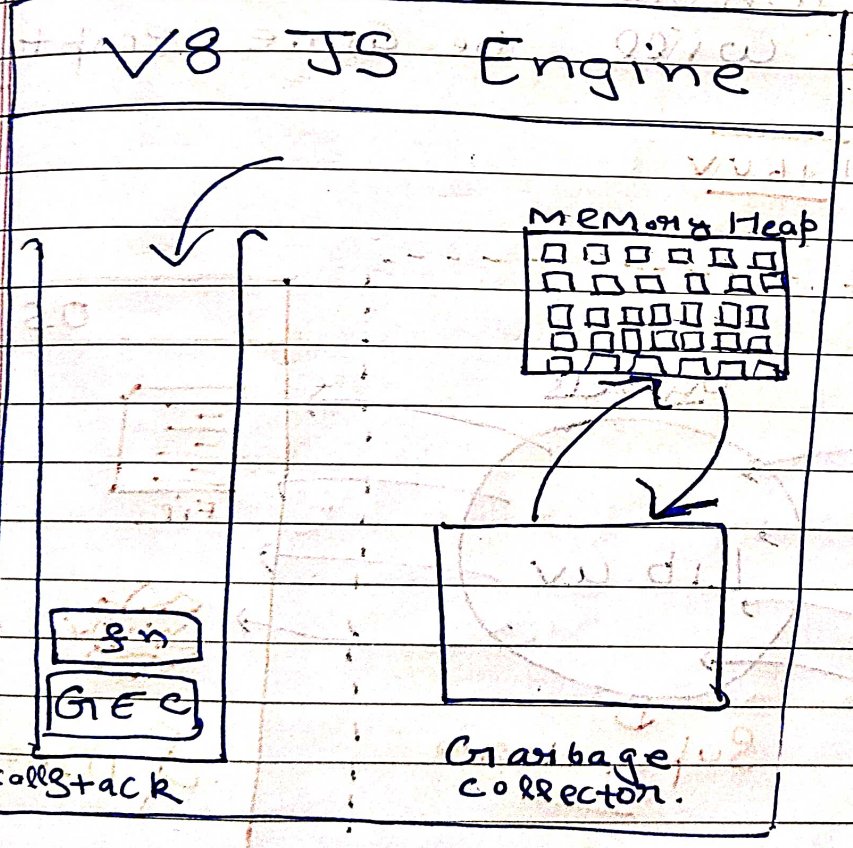
All people with pizza will get at 10 min

All people with noodles will get at 5 min

No body has to wait for any other order

Non blocking operation

How Synchronous Code is executed?



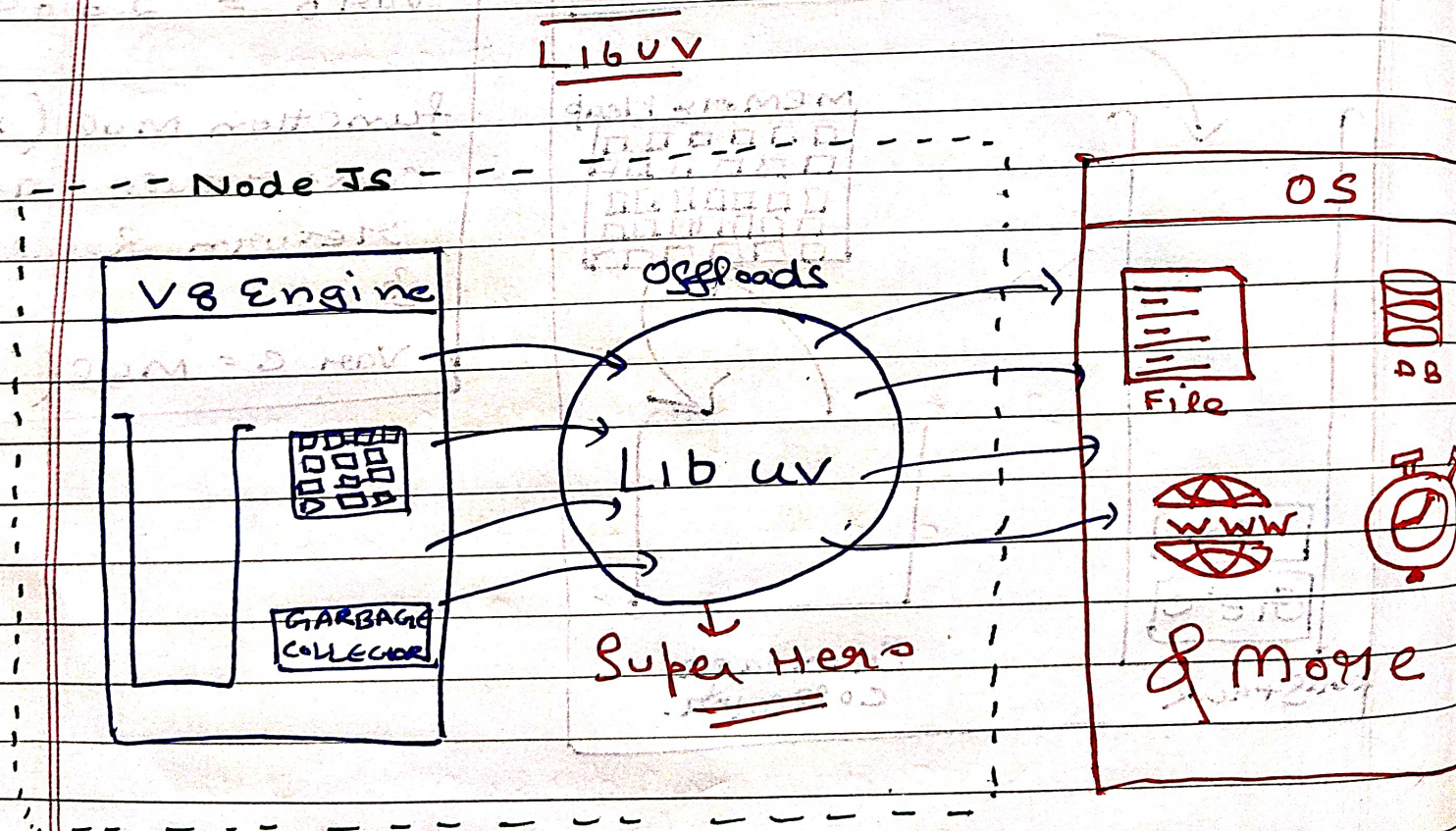

```
Var a = 10,10 ;
Var b = 20,910 ;

function Mul (x, y)
Const result = a*b;
Return Result ;
}
Var c = Mul(a, b)
```

Whenever you run the code a Global execution Context is created and it is pushed to the Call Stack.

Whenever a function comes it is put into call stack and all the Result of the Calculations will be stored into Heap once execution is over the result is Returned to GEC and Function Moves out of the Call Stack.

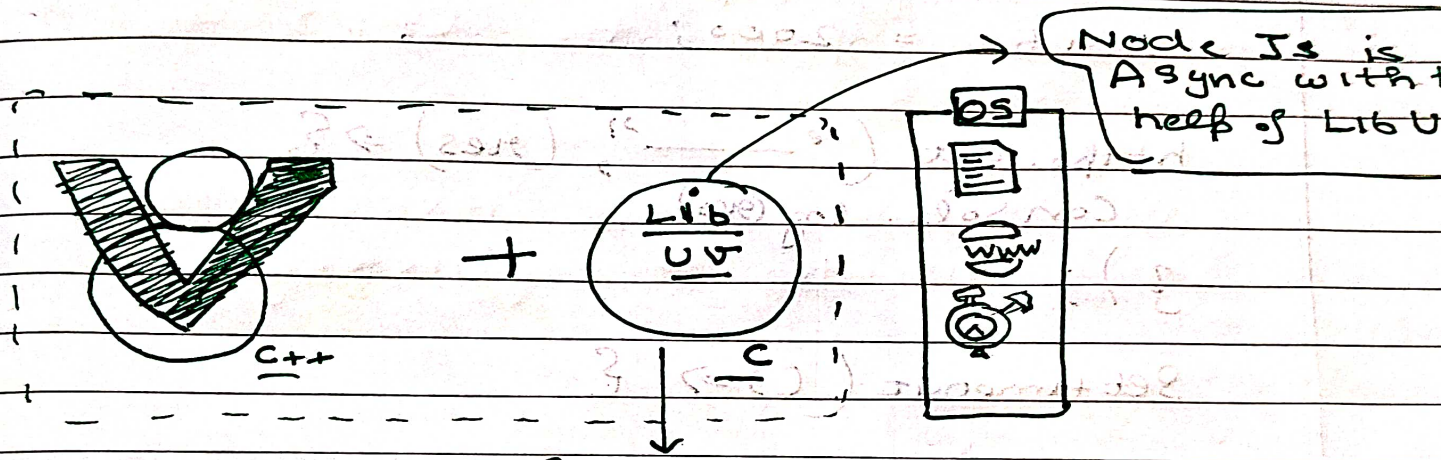
Once the whole code is executed Call Stack will be come empty



Js engine alone is not capable of reading data from File, Connecting to OS, Fetching Data, Using Timers, performing async operations

All this is done with the help of LibUV.

How Asynchronous Code Works?




```
fetch('/link').then(){
==========
}
fs.readfile()
setTimeout(() => {
},5000);
```

When NodeJs sees an Api call it asks libuv to get the data then gives to V8 to Execute, libuv performs all the operations of talking to OS,

How a Code Containing both Sync & ASync operation works

```
Var a = 1000;
Var b = 2000;
https.get("_____", (res) => {
  Console.log(res).
});
Settimeout((c) => {

}, 5000);
fs.readfile()
function multiply(a, b) {
  Const result = a * b
  return Result;
}
Var c = multiply(a, b);
Console.log(c)
```

Steps →

1. Firstly GEC is created inside Call Stack Code inside GEC will now execute in Sync Single threaded way.

2. Memory will be allocated to a, b vars & Garbage Collector will work in sync with Memory Heap

3. For Api calls, Libuv will manage the Api call. it will Register the Api call, and takes the Callback Mean while libuv is managing the Api call Js engine will move to the next line

4. For Settimeout again Js engine connects to libuv and libuv registers the Timer & stores the Callback Functio

5. Now we got readfile which is again async Code & it will be sent to libuv

6. All async tasks will be offloaded to libuv.

7. Now Js engine will quickly execute Multiply fn and new fn Context will be created & it will be executed

8. Once the Call Stack gets empty, all the memory will be cleaned by garbage Collector.

Once libuv is done with all the tasks and it sees that the call stack is empty

As soon as File data is returned to libuv it will give the callback function to JS engine & it will be putted into the call stack.

Similarly Api call is success it will put the CBC() to call stack

Call stack will execute all the stuff inside it quickly.