⑤ Caching

Libuv & async I/O

As we know,
Java script is a synchronous single Threaded

which means,

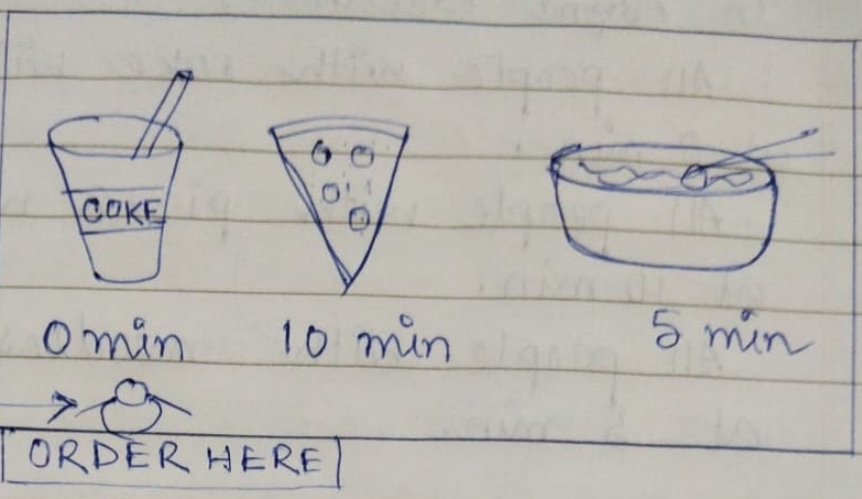
- Single Thread → can Run on a single thread and a single process.
- Synchronous → one after another.

JS Engine on V8 will run the JS code as soon as it is sent to JS Engine (very fast execution, quick)

To run JS we don't 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.



	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 fulfilled

Generally, blocking is not a good way

In Async Execution,

All people with coke will get it at 0 min.

All people with pizza will get at 10 min.

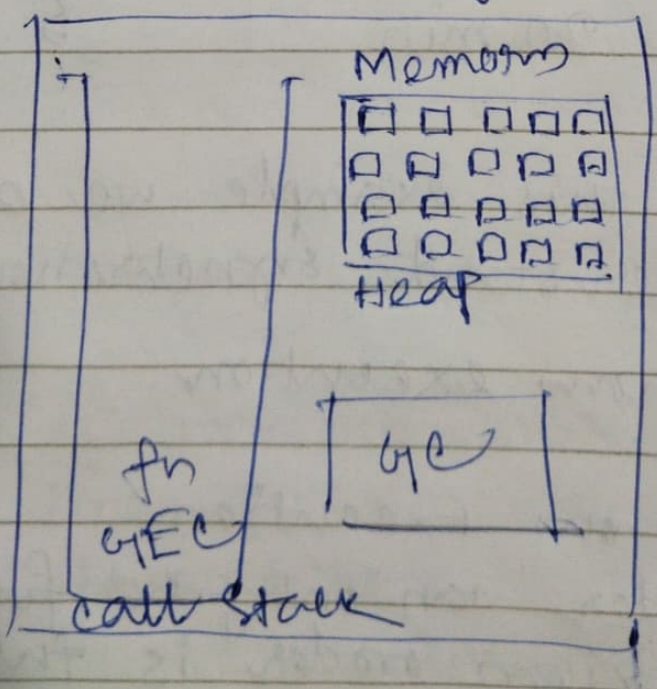
All people with noodles will get at 5 min.

No body has to wait for any other order

Non-blocking operation

How Synchronous code is executed?

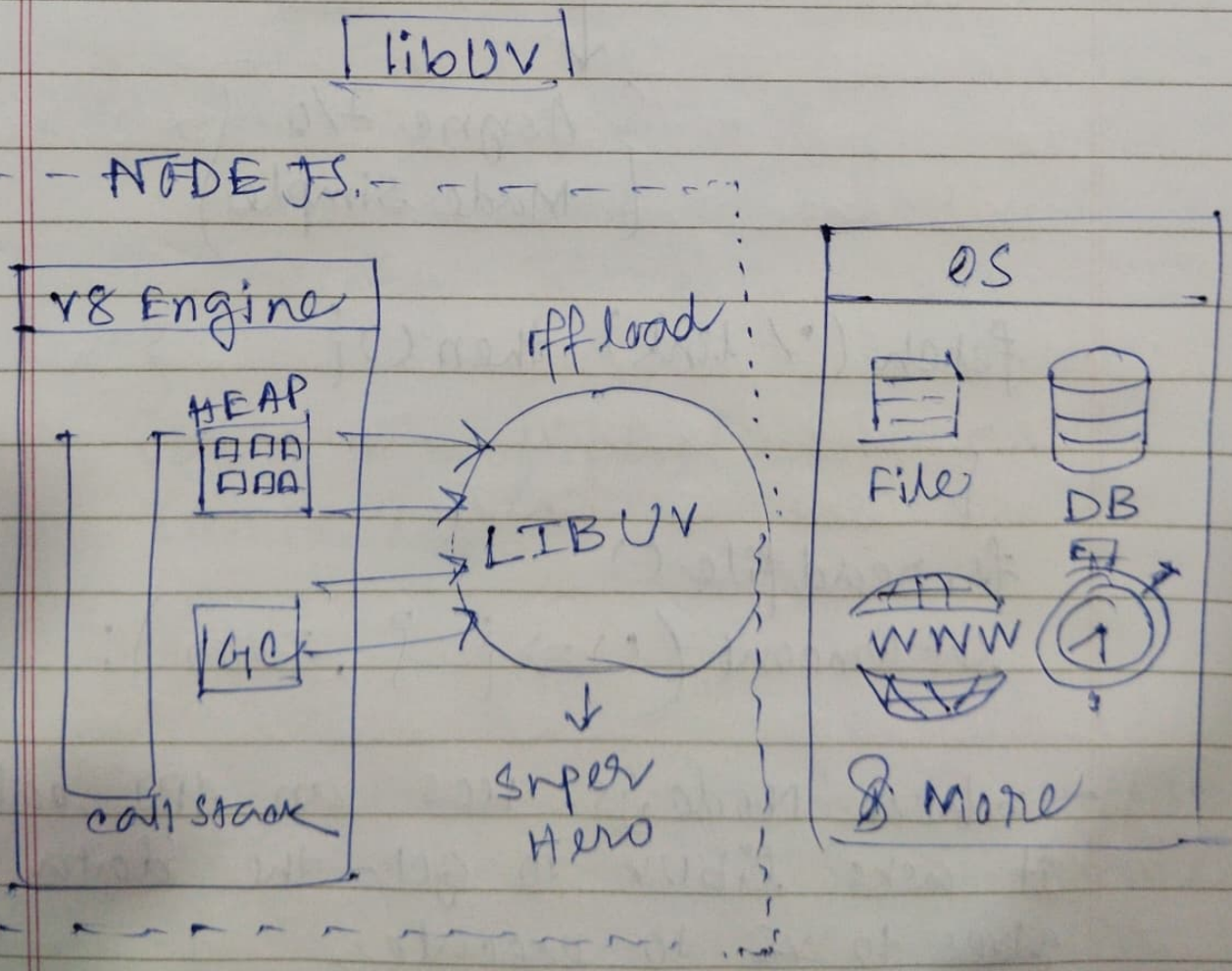
V8 JS Engine



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 calculation will be stored into heap. Once execution is over the result is returned to GEC and function moves out of the call stack.

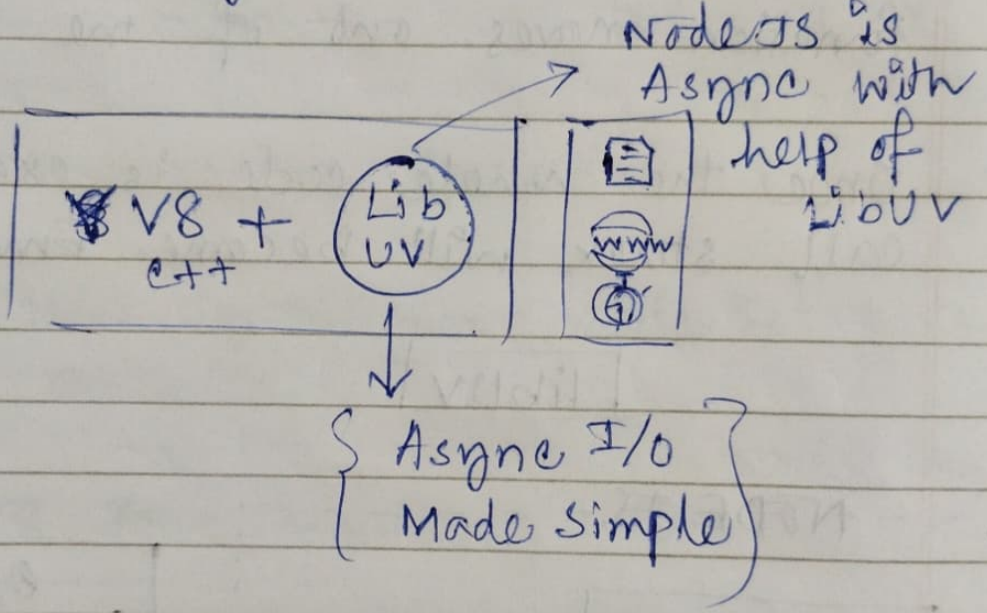
Once the whole code is executed call stack will become Empty.



JS Engine alone is not capable of reading data from file, connecting to OS, Fetching data, using Timers performing async operations.

All there is done with the help of LibUV.

How Asynchronous code works.




```
fetch('/link').then(() {

});
fs.readFile()
setTimeout(() => { }, 5000);
```

When Node.js 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(() => {
},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 callstack code inside GEC will now execute in sync single threaded way.

2. Firstly, GEC is created inside Memory will be allocated to a, b variable & garbage collector will work in sync with memory heap.

3. For API calls, LibUV will manage the API call and takes the call back. 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 function

5. Now, we got readfile which is again async code & it will be sent to libUV.

6. All async tasks will be off loaded to libUV.

7. Now JS Engine will quicking execute multiple function 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.

~~When~~ 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 put into the call stack.

call ~~back~~ stack will execute all the stuff inside it quickly.