Javascript → synchronous single threaded language

9/12/25

Libuv async IO

(Node.js has a event-driven architecture capable of asynchronous I/O.)

Synchronous.

```
var a = 1078698;
var b = 20986;
function multiplyFn(x, y) {
    const result = a * b;
    return result;
}
var c = multiplyFn(a, b);
```

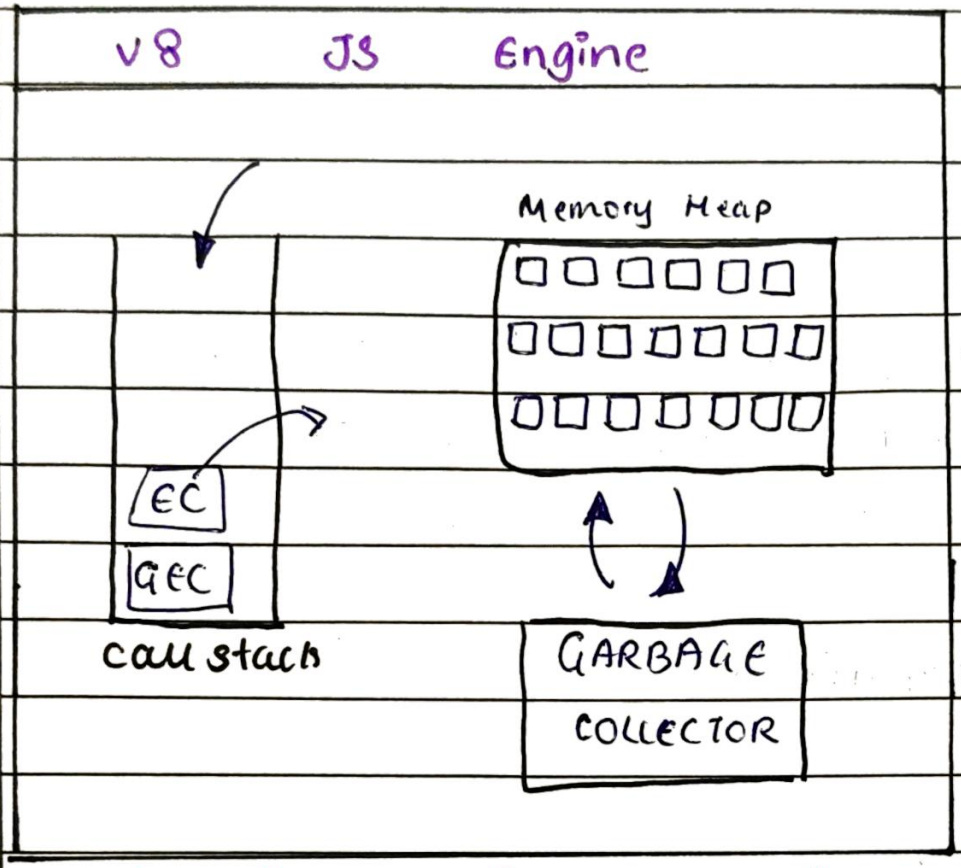
These task can execute immediately.

Asynchronous.

```
https.get("https://api.fbi.com", (res) => {
    console.log("secret data: res.secret");
});

fs.readFile("./gossip.txt", "utf8", (data) => {
    console.log("File Data", data);
});

setTimeout(() => {
    console.log("wait here 5 seconds");
}, 5000);
```

=> These tasks take time to execute.

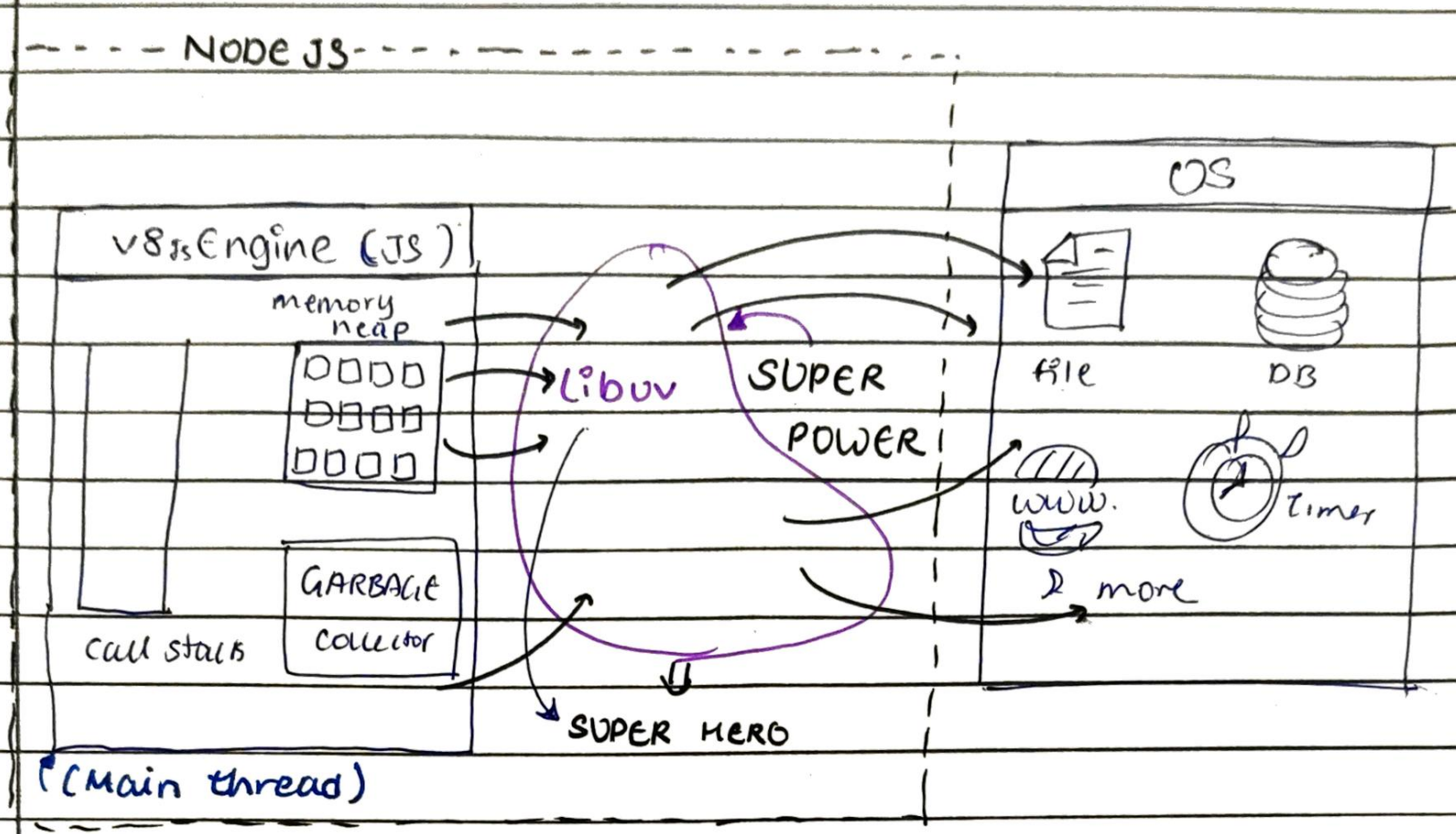
```
var a = 100837;
var b = 2098;

function multiplyFn(x, y) {
    const result = a * b;
    return result;
}
var c = multiplyFn(a, b);
```



(Running on single thread)

⇒ Time, Tide & Javascript waits for none".

- Js Engine don't have a concept of timer!
  it don't know how to wait.



libuv → genie for us.

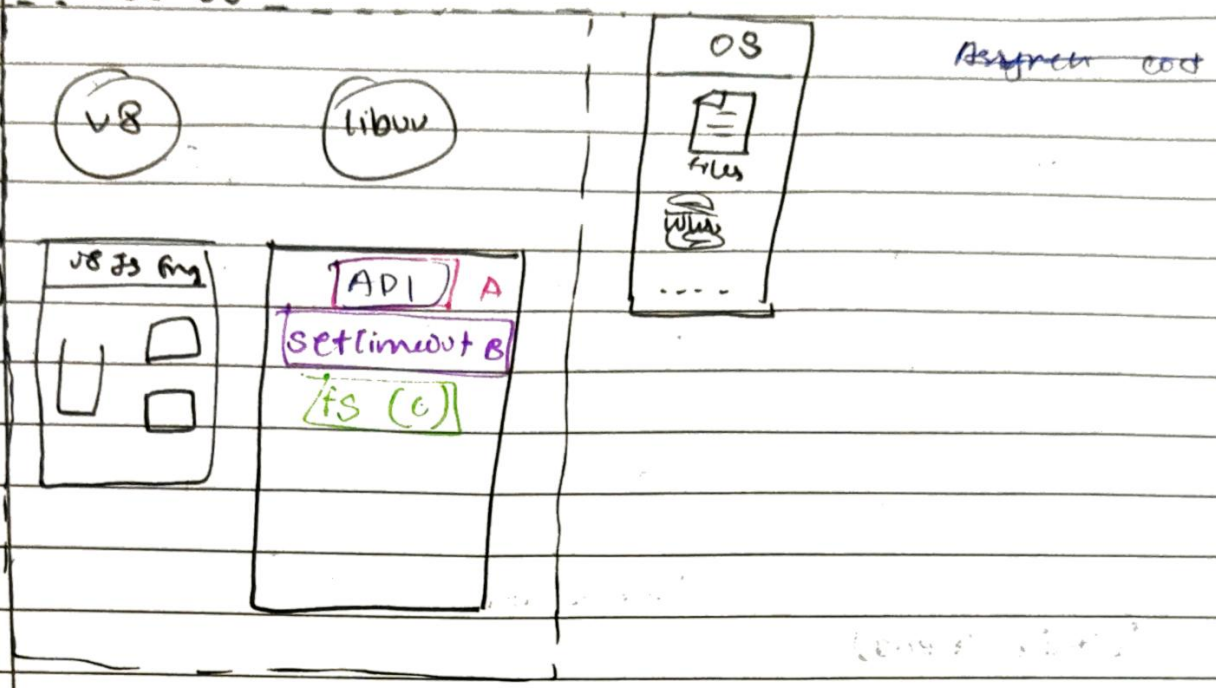
libuv : (Asynchronous I/O made simple)

↳ multi-platform C library that provides support for asynchronous I/O based on event loops.

(Non-blocking I/O) → using V8 engine

↳ Not blocking the main thread.

NODE JS




```
var a = 078876;
var b = 2098;
A  https.get("https://api.fbi.com", (res) => {
        console.log(res?.secret); });
B  setTimeOut (() => {
    consol.log("setTimeout"); }, 5000);

C  fs.readFile("./hello.txt", "UTF8", (data) => {
    console.log("File data", data); });

function multiplyfn(x, y) {
    const result = a*b;
    return result;
}
var c = multiplyfn(a,b);
    consol.log(c);
```