Agenda

(*) Intution behind Async programming
(*) Async js with call backs.

(*) All about the event loop with call back queue

(*) Serial and parallel execution of Async code.

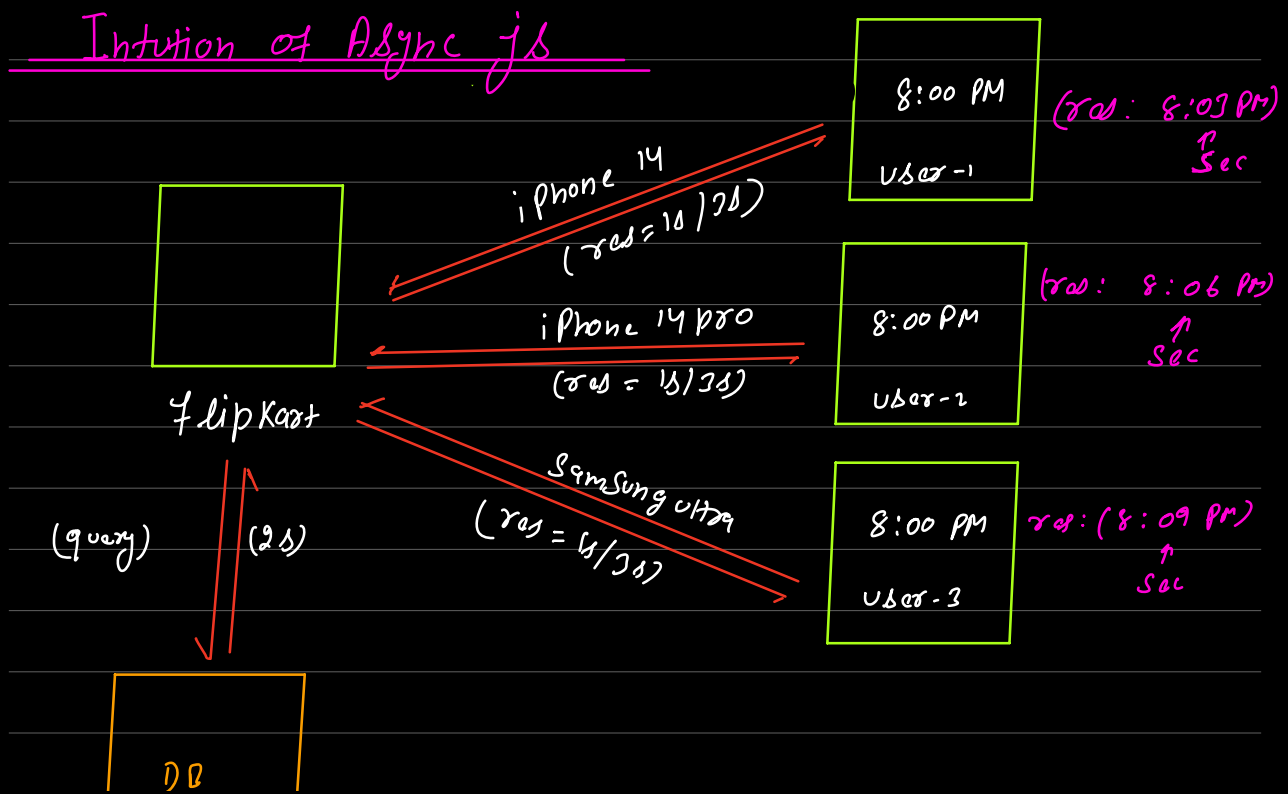
next class

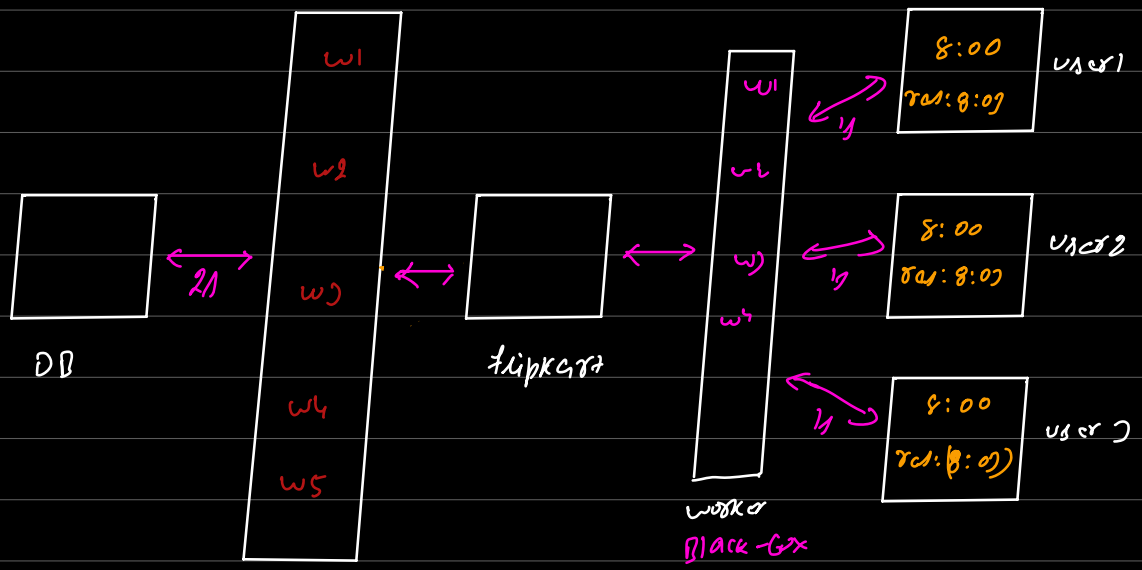
(*) Timer : setTimeout , setInterval , clearInterval

(*) setInterval and clearInterval poly fill.

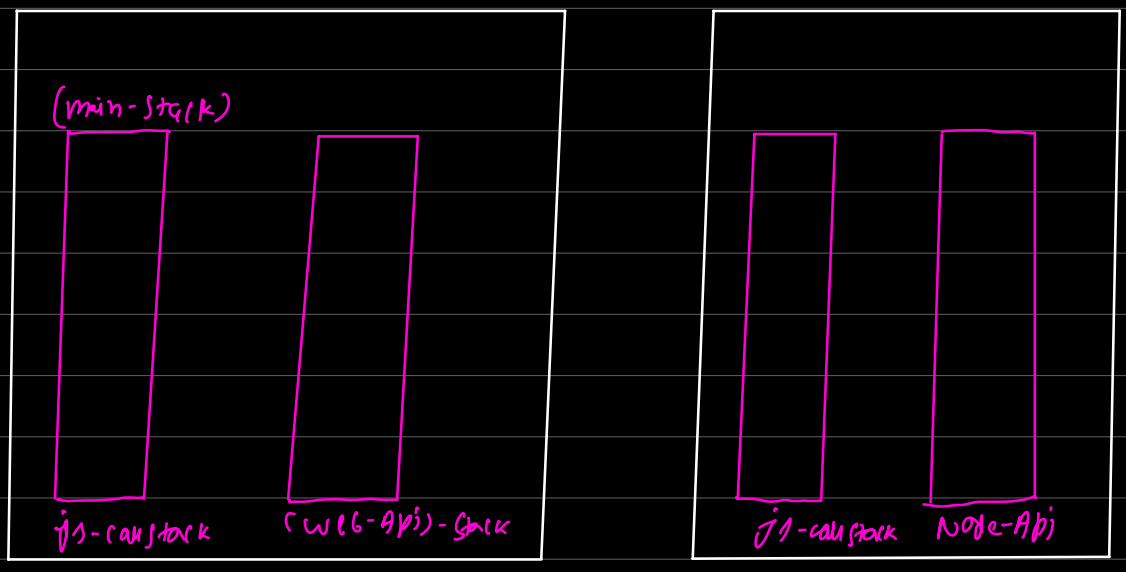
(*) dis cuss Home work

Intution of Async js





(worker / black box)

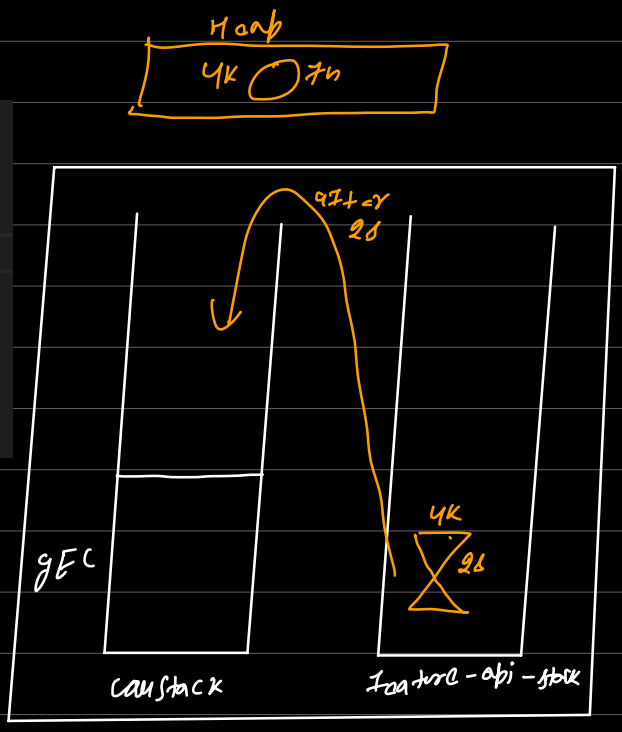


(Ryan Dahl)

(*) React Native → mobile app

(*) Electron → Desktop app

```
console.log("Before");

function fn(){
    console.log("I'm in function.");
}
setTimeout(fn, 2000);

console.log("After");
```



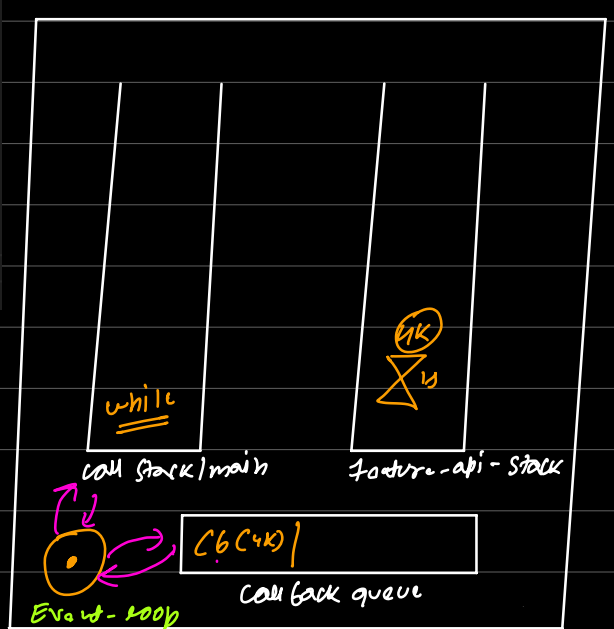
console

Before

After

I'm in function.

```
let a = true;
console.log("Before");

setTimeout(() => {
    a = false;
    console.log("I will broke the while loop.");
}, 1000);

console.log("After");

while (a) { }
```



console

Before

After.

∞

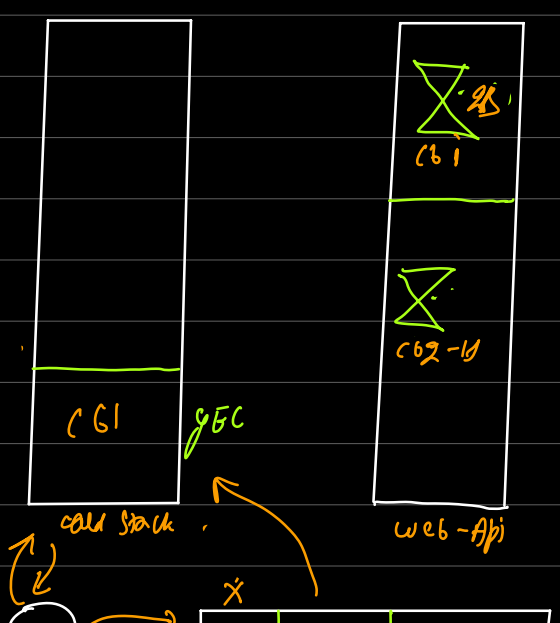
event loop: It will check whether call-stack is empty or not and if call-stack is empty it will pop a call-back process from call-back queue and push inside the call stack for execution

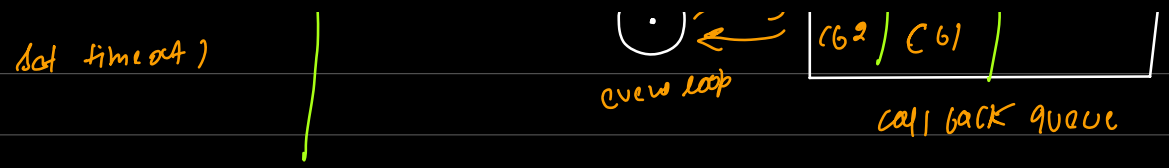
```
let a = true;
console.log("Before");

setTimeout(() => {
    a = false;
    console.log("I will broke the while loop.");
}, 1000);

console.log("After");

// while (a) { }

let timeFuture = Date.now() + 5000;
while(Date.now() < timeFuture){}
```

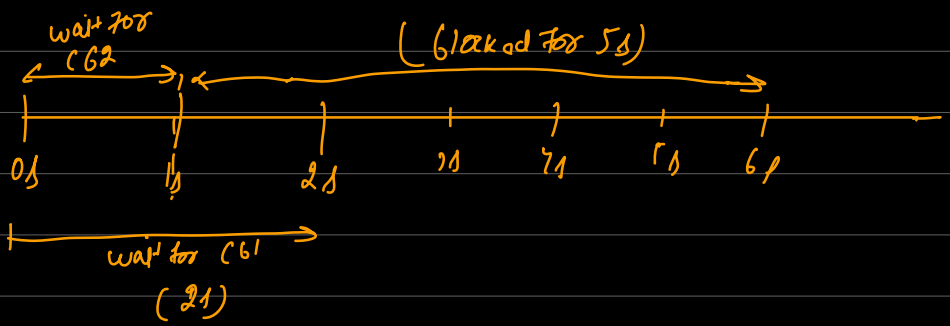
Total execution time: 5s

```
console.log("Before");
const cb2 = () => {
    console.log("set timeout 1");
    let timeInfuture = Date.now() + 5000;
    while (Date.now() < timeInfuture) {
    }
}
const cb1 = () => console.log("hello");
setTimeout(cb2, 1000)

setTimeout(cb1, 2000)

console.log("After");
```

Console
Before
After







Serial

(delay of 2ms) | Video → 3 GB → cloud. | Zoom

(download → compress upload on our server)

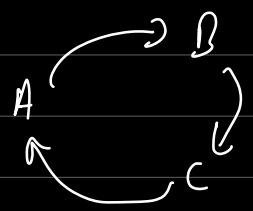
↘ divide vide in multiple chunks

(*) n files [of different name and different route]

read files in serial.

(array / list) : | 77 | 76 | 75 | 74 | 73 | 72 | 71 | no shifting will done.

| 71 | 72 | 73 | 74 | 75 | 76 | shifting take O(n) time



deadlock

(Serial)

A

compiler (blocked) looping

B

(parallel)

A

B

(compiler wait till it's done with reading)

```
console.log("Before");

fs.readFile("./f1.txt", f1cb);

function f1cb(err, content_1) {
    fs.readFile("./f2.txt", f2cb);

    function f2cb(err, content_2) {
        console.log("concated result: " + content_1 + " & " + content_2)
    }
}

console.log("After");
```