Forth Pillar of JS

Intro to callbacks :

```
function fun (x, fn) {

}
```

fn → function → HOF : it takes fn as argument

x → number

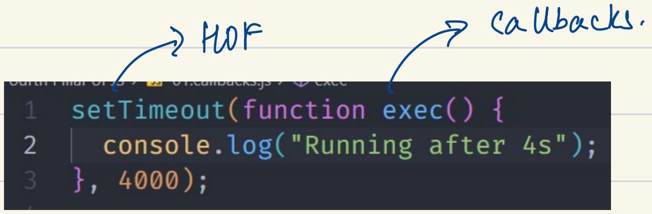
```
function fun (x, fn) {

    for (i=0; i<x ; i++) {
        console.log(i)
    }
    fn ()
}
fn (10, func exec () {
    console.log(" I am executed also")
})
```

func exec → call back function

o/p :

```
1
2
3
:
9
I am executed also
```



```
setTimeout(function exec() {
  console.log("Running after 4s");
}, 4000);
```

HOF are those function which take a function as argument the function you pass as a argument is a call back function.

Problem of callback:

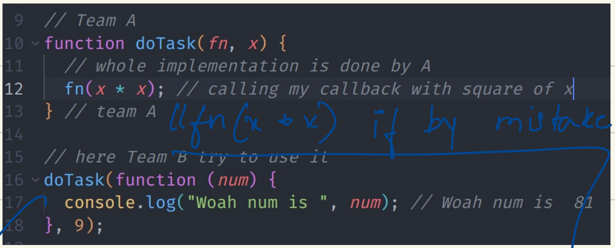
- Callback hell (but not exactly)
- Inversion of control (main issue)

Call back hell:

- Callback inside callback there will be a readable issue to understand what it do.

- It doesn't hamper the logic issue.

Inversion of control:



```
// Team A
function doTask(fn, x) {
  // whole implementation is done by A
  fn(x * x); // calling my callback with square of x
} // team A

// here Team B try to use it
doTask(function (num) {
  console.log("Woah num is ", num); // Woah num is  81
}, 9);
```

//fn(x * x) if by mistake you log it twice, for e.g. while booking a seat in flight and it get reserve twice.

→ due to callbacks, I am passing control of exec should be called to doTask, this is IOC.

You gave someone else the authorite to execute it it's known as inversion of control.

If there is somepart of code whose control we are passing it to someone else and we don't known how that part will be executed this problem is called inversion of control.

To handle this we use Promise.