



Day 6

LIVE: Revision of Day (1-5) and assignments

1. Code for the fs error:

```
const fs = require("fs");

fs.readFile("anish.txt", "utf-8", function(err,data){
  // console.log(err);
  console.log(data);
});

// console.log("data");
```

2. Promises:

Syntactical Sugar

Easy way to understand Async Function.

It uses callback but can lead to **callback hell**.

Promises

Approach #1 (Wrapping another async fn)

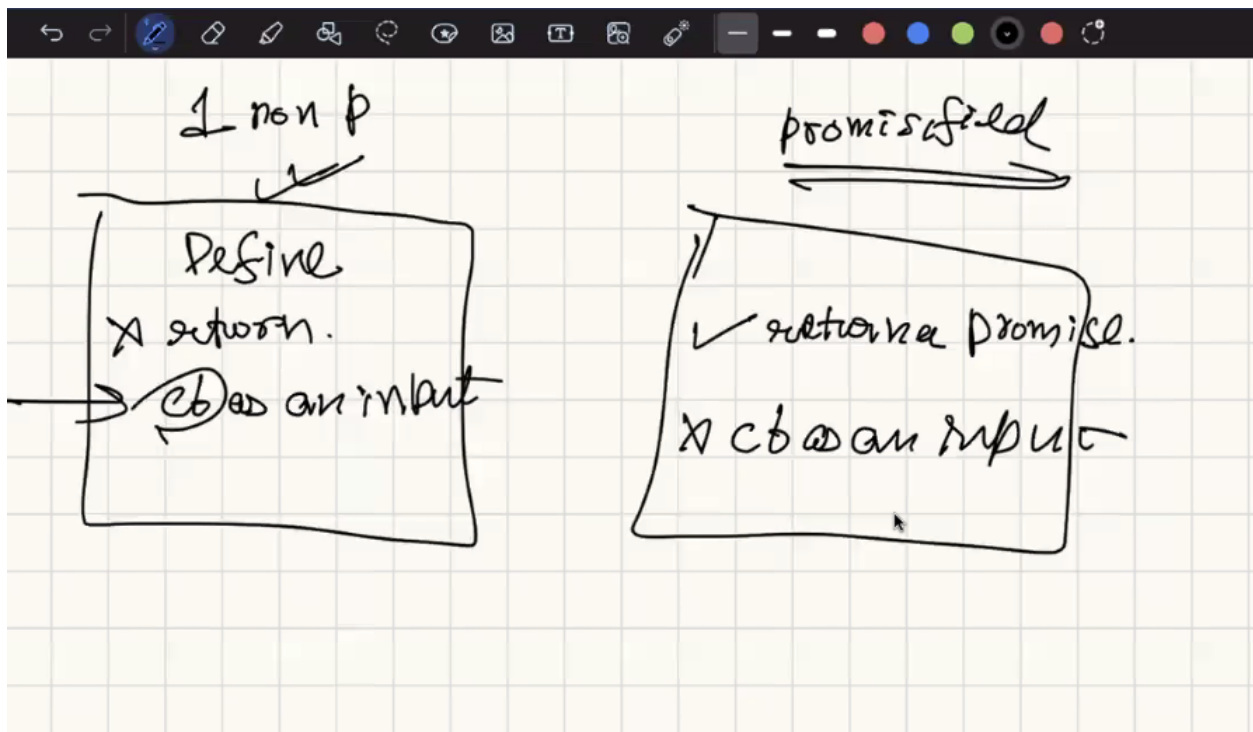
```
index.js > ...  
1 function myOwnSetTimeout(fn, duration) {  
2   setTimeout(fn, duration);  
3 }  
4  
5 myOwnSetTimeout(function() {  
6   console.log("hi there");  
7 }, 1000)
```

Approach #2 (Using promises)

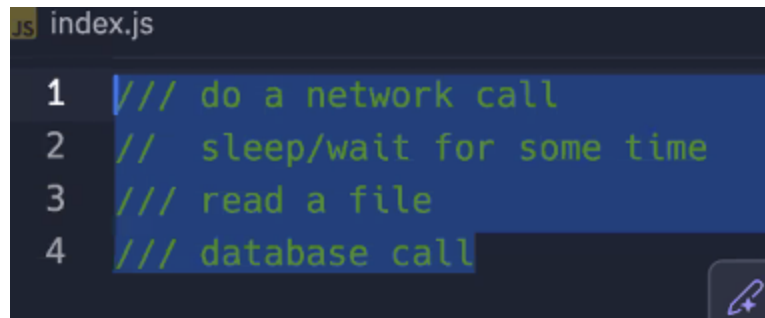
```
index.js > f myOwnSetTimeout > ...  
1 function myOwnSetTimeout(duration) {  
2   let p = new Promise(function (resolve) {  
3     // after 1 second, call resolve  
4     setTimeout(resolve, 1000);  
5   });  
6   return p;  
7 }  
8  
9 myOwnSetTimeout(1000)  
10 .then(function () {  
11   console.log("log the first thing");  
12 });  
13
```

<https://gist.github.com/hkirat/9a7a0ef9ad6788f645497a2cd2b92106>

3. Non - Promisified vs Promisified Function:



4. Cases when you need to write async code:



```
index.js
1  /// do a network call
2  //  sleep/wait for some time
3  /// read a file
4  /// database call
```

5. Promisified vs Non- Promisified Code:

```
//promise
const ans = promisifiedMyOwnSetTimeout(1000);
ans.then(function(){
  console.log("timeout is done");
});

// instead of--
promisifiedMyOwnSetTimeout(1000, function(){

});
console.log("Enter the string");

// Normal Callback -->
fs.readFile("a.txt", "utf-8", function(err, data){

});

//Promisified Callback -->
fs.readFile("a.txt", "utf-8").then(function(err, data){

});
```

6. Promises also go to the Call Stack.

7. The basic reason why you want to go to promise function is because you will go to the [async function](#) —> [await](#) with this.

Hence promises are a great way even if the syntax is cluttered at first.

8. Java, Golang, Rust are multithreaded languages while Javascript is a single threaded language, hence things like cluster [modelling](#) is not available in JS.

9. Know how to call a promise, the syntax for now is now that important.

```
const p = new Promise (function(resolve){
  resolve("Hi there");
})
p.then(function(arg){
})
```

10. [Promise call](#) will need a function which already has a parameter as in input.