



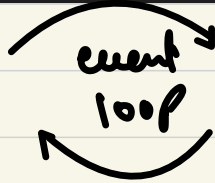
```

1 function createTimer(time, timerId) {
2   console.log("Creating a new timer with id", timerId);
3   setTimeout(() => {
4     console.log(`Timer with id ${timerId} is done`);
5   }, time);
6   console.log("Successfully created a new timer with id", timerId);
7 }
8 console.log("Starting the code");
9 createTimer(2000, 1);
10 createTimer(0, 2);
11 console.log("Starting a loop");
12 for(let i = 0; i < 1000000000000; i++) {
13   // something is going on
14 }
15 console.log("Loop is done");
16 console.log("Last line of code done");

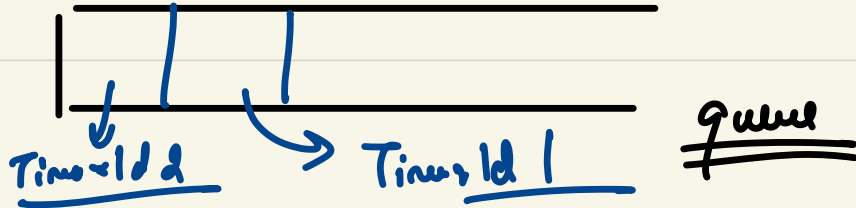
```

1 → 4-5 sec

Runtime Env
 → Timer: 1, 2 sec,
 → Timer: 2, 0 sec,



Call Stack



Starting the code

creating a new timer with id 1.

Successfully created a new timer with id 1

creating a new timer with id 2

Successfully created a new timer with id 2

Starting the loop

loop is done

last line of code is done

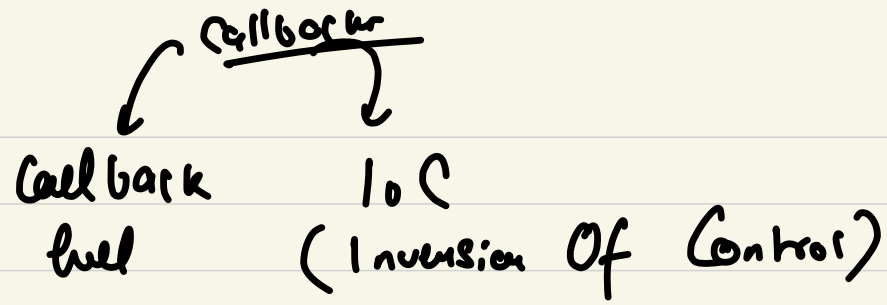
timer with id 2 is done.

timer with id 1 is done.

-# how is our callback ab'z is ours
local variable of a func" that doesn't
exist.

Closure is a mechanism using which a function remembers the variable present in the outer function scope, even when the outer function execution is completed.

↓
Scope class



✓✓ IOC

(call back hell)

Promises In JS

(officially part of JS)

JS object

(promise hell)

NodeJS

Runtime

Call back features


Promises

Promises are readability enhancers.

They are a special JS object, which can help us to control future related tasks.

Note → generally we prefer using promises when we have to deal with future tasks.

- download some data
- very a time



How to create
a promise?

how to
consume a promise

ρ ρ — —

Paine

→ —

→ —

→ —

→ download (" , - - - - -) ;

→

→

→

'w

→

2min

What to do after downloading the data 3?

The process of downloading &
what to do with the downloaded data is
independent of each other

what to do after downloading can be
decided by whosoever is calling our
down load func.

```

11 → function download(url, callback) {
12     console.log("Downloading from", url);
13     → setTimeout(() ⇒ {
14         console.log("Download is done");
15         let downloadedData = "Some data";
16         → callback?.(downloadedData);
17     }, 3000);
18 }

```

```

20 → function writeFile(data, fileName, callback) {
21     // fileName tells the name of the file to be created in which data will be written
22     console.log("Writing", data, " to file");
23     setTimeout(() ⇒ {
24         console.log("Writing to file ", fileName, " is done");
25         let status="Success";
26         callback?.(status);
27     }, 2000);
28 }

```

```

30 → function upload(fileName, url, callback) {
31     // fileName tells the name of the file to be uploaded
32     console.log("Uploading file ", fileName, " to ", url);
33     setTimeout(() ⇒ {
34         console.log("Upload is done");
35         let uploadStatus = "Success";
36         callback?.(uploadStatus);
37     }, 3000);
38 }

```

```

40
41 → function process() {
42     → download("https://www.example.com", function handleDownload(data) {
43         → writeFile(data, "file.txt", function handleWrite(status) {
44             → upload("file.txt", "https://www.example1.com", function handleUpload(uploadStatus) {
45                 console.log("All done");
46             });
47         });
48     });
49 }
50
51 → process();
52

```

R.E

and loop

9 min.



Downloady from www.example.com

download is done

Writing some data to file

Writing to file file.txt is done

Uploading file file.txt to www.example1.com

upload is done

All done