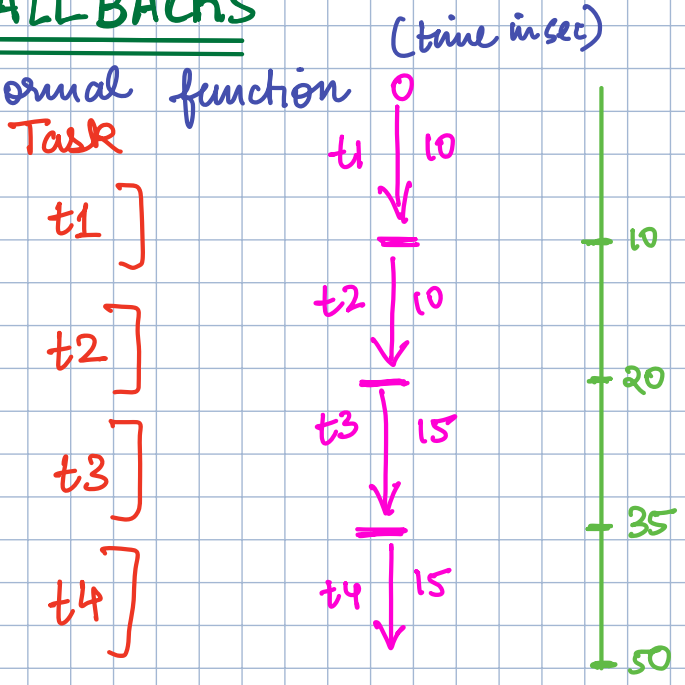
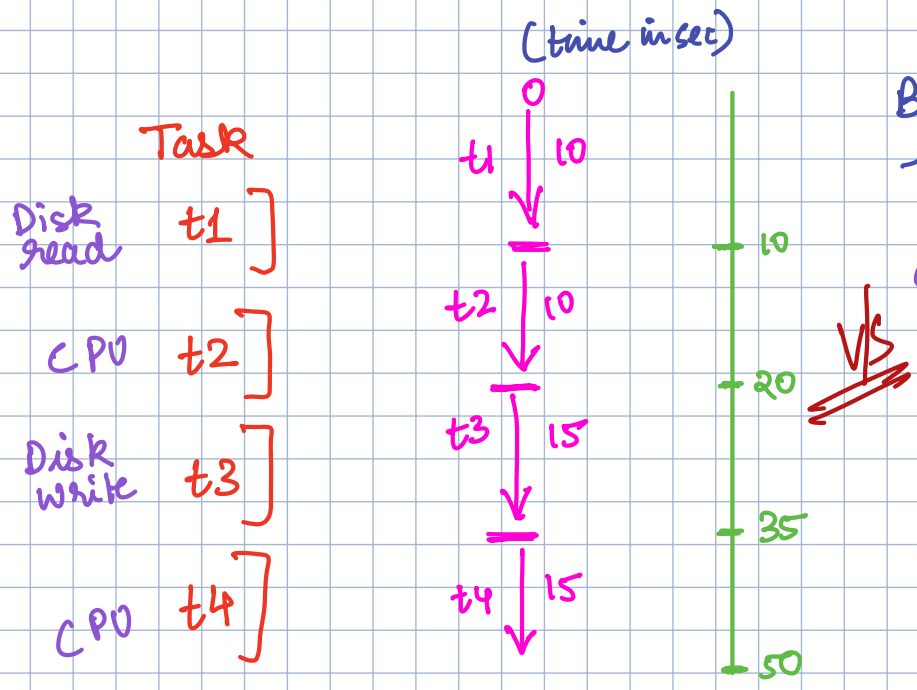
CALLBACKS

→ what is need of callback?
→ what callback does?

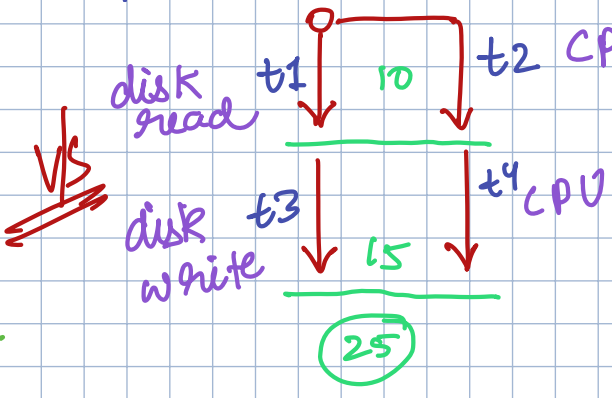
For normal function



→ Computer has different resources. Some tasks does not make CPU busy but they keep disk busy (eg- reading from hard disk)



But these tasks can be performed in parallel.

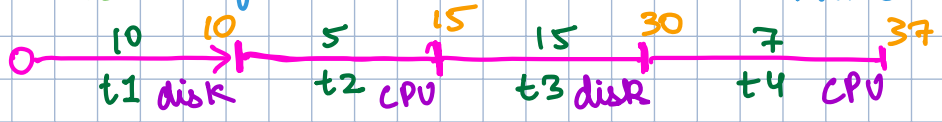


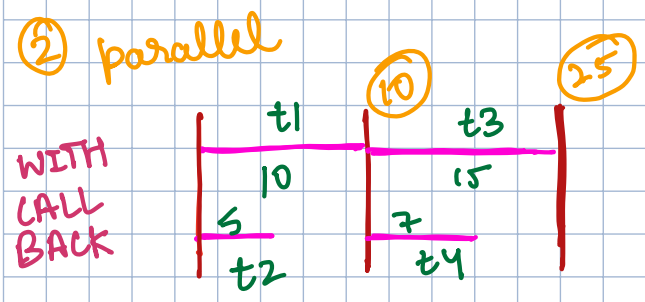
Suppose we are calculating prime number of 1 Lakh numbers This task will be done by CPU and it will keep CPU busy. Let say's we are reading from one file and this will keep disk busy. Both these task could be performed in parallel.

① Sequential

LACK OF CALLBACK

Disk	CPU
t1 → f1.txt read 10	t2 Prime no. 5
t3 → f2.txt write 5	t4 Prime Square 7



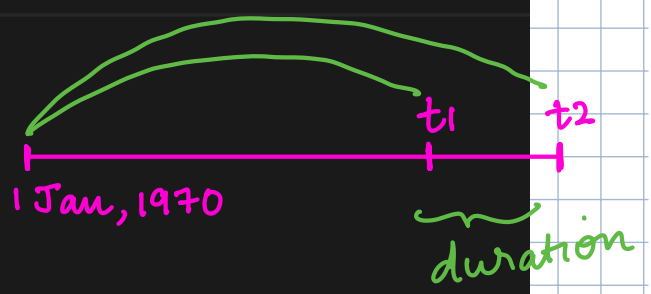


JS FirstLackOfCallback.js U ×

Javascript > Lecture 22 > JS FirstLackOfCallback.js > ...

```
1   // t1 = Read a file (disk)
2   // t2 = Calculate primes (cpu)
3   // t3 = Write a file (disk)
4   // t4 = Calculate square of primes (cpu)
5   // node FirstLackOfCallback.js --source=f1.txt --dest=f2.txt --n=50000
6
7   // CASE 1 : Task execution in sequencial manner
8
9   let minimist = require("minimist");
10  let fs = require("fs");
11
12  let args = minimist(process.argv);
13
14  let t1 = Date.now();
15  console.log("Starting task1 at " + t1);
16  let stext = fs.readFileSync(args.source, "utf-8");
17
18  let t2 = Date.now();
19  console.log("Finishing task1 at " + t2);
20
21
22
23
```

gives no. of miliseconds that has elasped since 1st January 1970



PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
→  Lecture 22 git:(main) ✗ node FirstLackOfCallback.js --source=f1.txt --dest=f2.txt --n=50000
Starting task1 at 1632664807450
Finishing task1 at 1632664807454
→  Lecture 22 git:(main) ✗
```

```
Javascript > Lecture 22 > JS FirstLackOfCallback.js > ...
```

```js
// t1 = Read a file (disk)
// t2 = Calculate primes (cpu)
// t3 = Write a file (disk)
// t4 = Calculate square of primes (cpu)
// node FirstLackOfCallback.js --source=f1.txt --dest=f2.txt --n=50000

// CASE 1 : Task execution in sequencial manner

let minimist = require("minimist");
let fs = require("fs");

let args = minimist(process.argv);

let t1 = Date.now();
console.log("Starting task1 at " + t1 % 100000);
let stext = fs.readFileSync(args.source, "utf-8");

let t2 = Date.now();
console.log("Finishing task1 at " + t2 % 100000);
```

truncate miliseconds

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

→ Lecture 22 git:(main) ✗ node FirstLackOfCallback.js --source=f1.txt --dest=f2.txt --n=50000
Starting task1 at 30484
Finishing task1 at 30488
→ Lecture 22 git:(main) ✗
```

```js
// task 1 area begins
// let t1 = Date.now();
// console.log("Starting task1 at " + t1 % 100000);

// let data = fs.readFileSync(args.source);

// let t2 = Date.now();
// console.log("Finishing task1 at " + t2 % 100000);
// console.log(t2 - t1);
// task 1 area ends


// task 2 area begins
let t3 = Date.now();
console.log("Starting task2 at " + t3 % 100000);

let arr = [];
for(let i = 2; i < args.n; i++){
    let isPrime = IsPrime(i);
    if(isPrime == true){
        arr.push(i);
    }
}

let t4 = Date.now();
console.log("Finishing task2 at " + t4 % 100000);
console.log(t4 - t3);
// task 2 area ends
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

→ Lecture 22 git:(main) ✗ node FirstLackOfCallback.js --source=f1.txt --dest=f2.txt --n=100000
Starting task2 at 46001
Finishing task2 at 46647
646
→ Lecture 22 git:(main) ✗
```

```
→  Lecture 22 git:(main) ✗ node FirstLackOfCallback.js --source=f1.txt --dest=f2.txt --n=100000
Starting task2 at 46001
Finishing task2 at 46647
646
→  Lecture 22 git:(main) ✗ node FirstLackOfCallback.js --source=f1.txt --dest=f2.txt --n=100000
Starting task1 at 6715
Finishing task1 at 7155
440
→  Lecture 22 git:(main) ✗
```

JS FirstLackOfCallback.js U ×

Lack of callback

Javascript > Lecture 22 > JS FirstLackOfCallback.js > ...

```
25    // task 1 area begins
26    let t1 = Date.now();
27    console.log("Starting task1 at " + t1 % 100000);
28
29    let data = fs.readFileSync(args.source);
30
31    let t2 = Date.now();
32    console.log("Finishing task1 at " + t2 % 100000);
33    console.log(t2 - t1);
34    // task 1 area ends
35
36
37    // task 2 area begins
38    let t3 = Date.now();
39    console.log("Starting task2 at " + t3 % 100000);
40
41    let arr = [];
42    for(let i = 2; i < args.n; i++){
43        let isPrime = IsPrime(i);
44        if(isPrime == true){
45            arr.push(i);
46        }
47    }
48
49    let t4 = Date.now();
50    console.log("Finishing task2 at " + t4 % 100000);
51    console.log(t4 - t3);
52    // task 2 area ends
53
54    console.log(t4 - t1);
55
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
→  Lecture 22 git:(main) ✗ node FirstLackOfCallback.js --source=f1.txt --dest=f2.txt --n=100000
Starting task1 at 90369
Finishing task1 at 90633
264
Starting task2 at 90633
Finishing task2 at 91313
680
944
→  Lecture 22 git:(main) ✗
```

FirstCallback.js U

Javascript > Lecture 22 > FirstCallback.js > ...

```
// t1 = Read a file (disk)
// t2 = Calculate primes (cpu)
// t2 is done in parallel with t1
// node FirstLackOfCallback.js --source=f1.txt --dest=f2.txt --n=50000

function IsPrime(x){
    let isPrime = true;
    for(let div = 2; div < x; div++){
        if( x%div == 0){
            isPrime=false;
            break;
        }
    }
    return isPrime;
}

let minimist = require("minimist");
let fs = require("fs");

let args = minimist(process.argv);

// task 1 area begins
let t1 = Date.now();
console.log("Starting task1 at " + t1 % 100000);

//let data = fs.readFileSync(args.source);

// asynchronous callback

fs.readFile(args.source, function(data) {
    let t2 = Date.now();
```

FirstCallback.js U

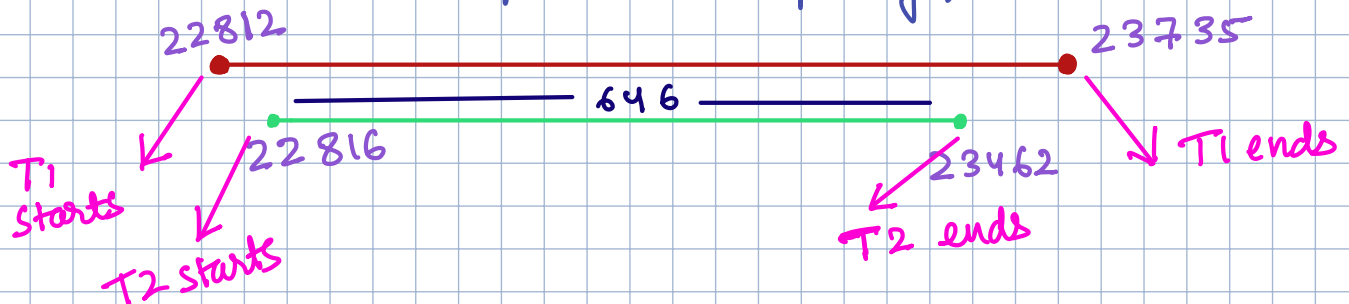
Javascript > Lecture 22 > FirstCallback.js > ...

```
// asynchronous callback

fs.readFile(args.source, function(data) {
    let t2 = Date.now();
    console.log("Finishing task1 at " + t2 % 100000);
    console.log(t2 - t1);
});


// task 1 area ends

// task 2 area begins
let t3 = Date.now();
console.log("Starting task2 at " + t3 % 100000);

let arr = [];
for(let i = 2; i < args.n; i++){
    let isPrime = IsPrime(i);
    if(isPrime == true){
        arr.push(i);
    }
}

let t4 = Date.now();
console.log("Finishing task2 at " + t4 % 100000);
console.log(t4 - t3);
// task 2 area ends
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
→ Lecture 22 git:(main) ✗ node FirstCallback.js --source=f1.txt --dest=f2.txt --n=100000
Starting task1 at 22812
Starting task2 at 22816
Finishing task2 at 23462
646
Finishing task1 at 23735
923
→ Lecture 22 git:(main) ✗
```

So, callback is basically a technique:
i.e many task can be done at same time
(similar to parallel computing.)



A function passed to another function as an argument is referred to as a callback function. The callback function runs after the completion of outer function. It is useful to develop an asynchronous Javascript code.

https://www.programiz.com/javascript/callback