LECTURE-23

(WEB DEVELOPMENT)

(CLASS-4)

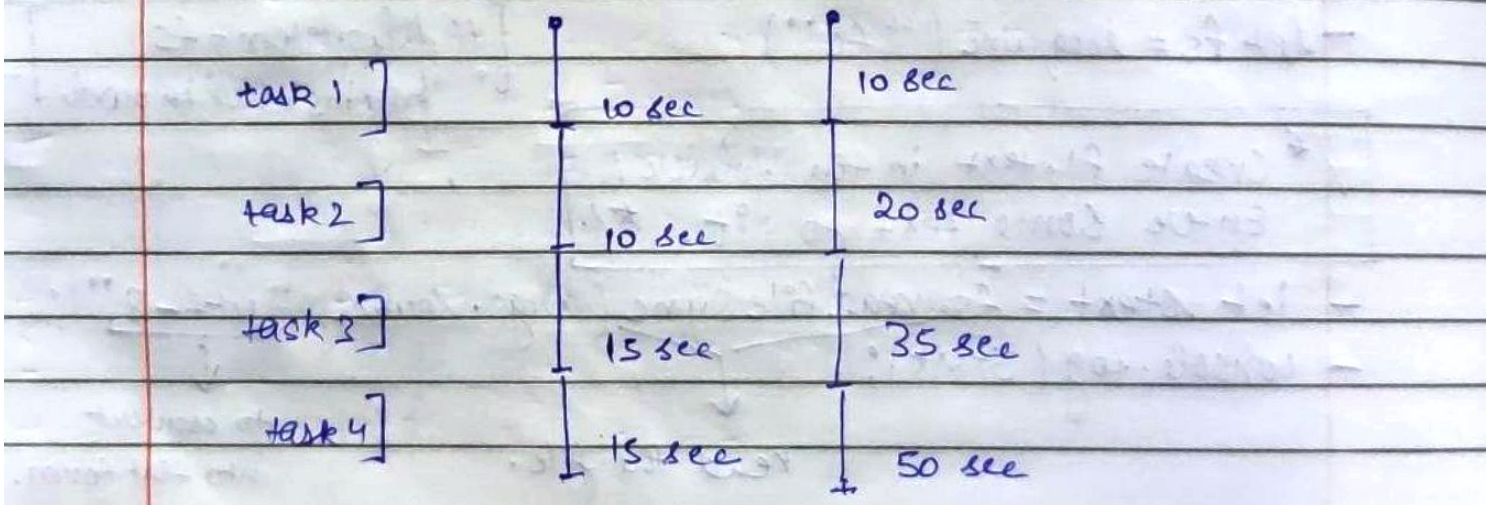
25th Sept 2021

(Saturday)

JAVASCRIPT

- first experience with callbacks
- first experience with timer
- first experience with downloading data from web
- first experience with writing data in excel file
- first experience at writing pdfs

first experience with callback.



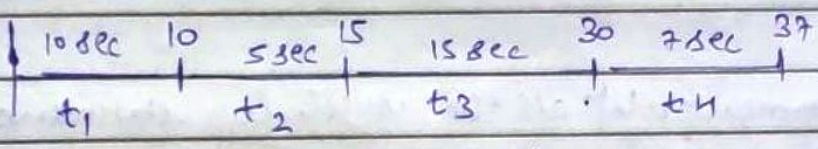
कोई कोई computer mai kaar kaam CPU se nahi hota, Disk se bhi hota hai.

Matlb,

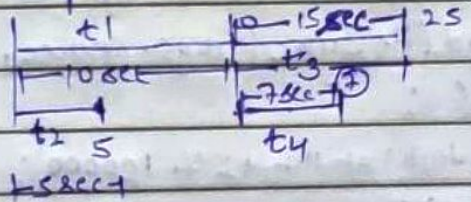
CPU का Resource hai	Disk का Resource है
T1 → Prime Number finding program (10 sec)	T2 → f1.txt read (5 sec)
T3 → Prime square number finding program (15 sec)	T4 → f2.txt write (7 sec)

If I take task manage like:

* 1 task at a time then;



* Multiple task at a time (using different resources)



Now; See;

filename: (firstLackofcallback.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

CLASSTIME Pg. No. 18
Date / /

```
let minimist = require("minimist");
let fs = require("fs");
let args = minimist(process.argv);
console.log(args.source);
console.log(args.dest);
console.log(args.n);
```

TERMINAL

```
node FirstLackofCallback.js --source=f1.txt --dest=f2.txt -n=50000
f1.txt
f2.txt
50000
```

Now,
comment out all the console.log statements;
and then; let t1 = Date.now(); console.log("Starting task1 at "+t1%100000);

task 1

```
let stext = fs.readFileSync(args.source, "utf-8");
let t2 = Date.now();
console.log("finishing task1 at " + t2 % 100000);
console.log(t2 - t1);          // Duration
```

```
let t3 = Date.now();
console.log("Starting task2 at "+ t2 % 100000);
let arr = [];
for(let i=2; i<args.n; i++){
  let isPrime = IsPrime(i);
  if(isPrime==true){
    arr.push(i);
  }
}
function IsPrime(x){
```

TASK-2

TASK-2

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

```
let t4 = Date.now();
console.log("      Awisking task 2 at " + t4 %.100000);
console..log(t4-t3);
console.log("Total time" + t4 - t1);
```

→ Terminal:

```
node firstLackofCallback.js --source=f1.txt --dest=f2.txt --n=40000
Starting task 1 at 92196
Awisking task 1 at 93005
809
Starting task 2 at 93007
Awisking task 2 at 93754
747
Total time 1558
```

T1 Task: 92196 – 93005 = 809
2 millsec
T2 Task: 93007 – 93754 = 747
2
1558

Now, Callback (Asynchronus method)
(FILENAME = firstcallback.js)

// T1 = Read a file (Disk)
// T2 = Calculate primes (cpu)

T2 is done in parallel with T1

```
Function IsPrime(x) {
    . . . . . .
    . . . . . .
    . . . . . .
    . . . . . .
    . . . . . .
}

let minimist = require("minimist");
let fs = require("fs");
let args = minimist(process.argv);

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

// let data = fs.readFileSync(args.source);    // old method,
// callback
fs.readFile(arg.source, function(data) {
    let t2 = Date.now();
    console.log("finishing task1 at" + t2 % 100000);
    console.log(t2-t1);
})

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

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

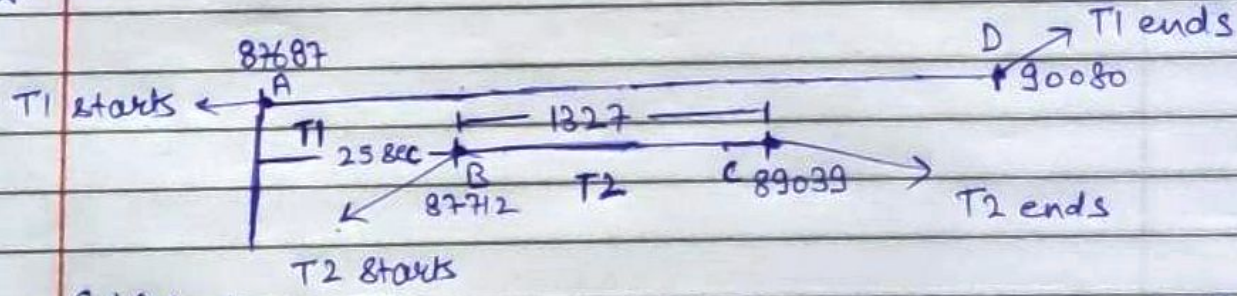
```
let t4 = Date.now();
console.log("finishing task 2 at" + t4 % 100000);
console.log(t4 - t3);
// console.log("Total time of the duration:" + t4 - t1);
```

TERMINAL :

```
node firstCallback.js --source=f1.txt --dest=f2.txt --n=50000
Starting task 1 at 87687
Starting task 2 at 87712
finishing task 2 at 89039
  1327
finishing task 1 at 90080
 2393
```

★ Note:

Callback is basically a technique ;
जब एक साथ बहुत सारे काम करे जा सकते हैं।
(similar to parallel computing)



Explaination:

Here, as you guys can see,
Task (T1) gets started At point A.
In the middle of the task (T1), task (T2) gets started at point B, after 25 milli seconds of task (T1)'s execution.

Thus;
task (T2) ends at point C, taking 1327 milliseconds to gets completed.
and after that;
task (T1) gets completed at point B taking 2393 milliseconds in total.

This; is how parallel tasks can work in JavaScript.