Lec-19
Web Dev
Begins

To start with any language
→ Output
→ Variable
→ Input
→ Conditionals
→ Loop

Web Development

Lecture - 19.

21 Sep, 2020

We are going to start with Node.js/ JavaScript
→ To download the node.js
→ node.js download
↳ 64 bit for mac / windows.
download as per the requirement.

1. Output → In Java.

```
System.out.println("Hello world");
```

In node.js

```
console.log("Hello world");
```

2. Variable → Here we don't need to specify the datatype.

```
let i=10;
```

↳ 'let' is used to create a variable in JS

∴ JavaScript is dynamically typed language.
∴ Java is a static typed language.

Lec-20
Web Dev
Class-1

CLASS-1.

To begin with JavaScript

To do list :-
- → Is Prime
- → Pattern
- → first func in JS.
- → first array
- → first cool thing → Timer

Getting started with any language.
- → Output
- → variable.
- → Input
- → Conditionals
- → Loops.

1. Output

In Mac VS Code → Press "Ctrl + `" to open the terminal.

⊙ Syntax :-

→ console.log( );

eg → console.log ("Hello world");

→ // This statement will print the statement ie Hello world on the terminal/output window.

2. Variable

⊙ Syntax :-

'let' is used to create a variable in JS.

eg →

```
let i=10;
console.log(i0);
```

// this statement will print 10 on the screen.

Now,

cls → cls is used to clear the screen in window

clear is used to clear the screen in Linux & mac.

```
console.log("Hello world");
console.log("Hello world");
```
} Same line में print करने के लिए string बनाना पड़ेगा।

आउटपुट - 2 line में print होगा।

Hello world.
Hello world.

```
i = "Hello world";
console.log(i);
```
→ Java में अगर i को int बना दिया या उसमें 10 डाल दिया तो उसमें string नहीं डाल सकते because वो string type का variable नहीं है। वो int type का variable है

→ अब 10 भी print होगा, साथ में Hello world भी next line में।

10
hello world.

```
i = true
console.log(i);
```
→ boolean.

// Hello world gets added to the variable i. this statement will print 10 & Helloworld & tru

int String Boolean

10
hello world
true
} Each in separate line.

Interview Type Question

JAVA is a Static Typed Language.
Javascript is a Dynamically Typed Language.

3. Input

⊙ Syntax → Variable → CLI से input data program को provide करते हैं

```
let args = process.argv;
console.log(args);
```

→ यह array जैसी है. internally जिसमें input store होता है.

Now, Run this statement on the terminal (command line) by typing.

// Here, input is given after writing the file name.

```
node input.js 10
```

filename, input data

eg → file wise

```
let args = process.argv;
let i = args[2];
console.log(i);
```

array जैसी

Terminal wise

node input.js 10 10

node → 0th index

input.js → at 1st index

10 → input data → At 2nd index.

10 → At 3rd index

Input will start from 2 on the command line because 0 has node & 1 has the file name.

CLI → node input.js 10 abc def

```
let cmdline args = process.argv;
console.log("At 0" + cmdline args[0]); → node
console.log("At 1" + cmdline args[1]); → filename
console.log("At 2" + cmdline args[2]); → Input 1
console.log("At 3" + cmdline args[3]); → Input 2
console.log("At 4" + cmdline args[4]); → Input 3
```

CLI → node input.js 10 "abc def"
0 1 2 3 4

Use " " for adding multiple value to a single variable i.e. एक साथ पूरा करना हो तो " " का इस्तेमाल करें।

Now nothing is there at 4th psn. तो इसकी जगह पर nan लिखा होगा।

* Everything is coming in String by default.

console.log("At 4" + cmdlineargs[4]); लिखा आएगा as an output.

→ nan

Command Line	Output
1. let i = cmdlineargs[2];	node input.js 10 & (filename. Input)
2. console.log(i);	10
3. console.log(typeof i);	string
4. i = i+30.	→ String +30 = String30.
5 console.log(i);	1030.
6 let j = parseInt("200", 10);	// Base is 10 here.
	→ हमने input दी number लिखा
typecasting string into no. using parseInt	
7. console.log(j);	→ 200.
8. console.log(typeof j);	number
9. j = j+30;	
10. console.log(j);	→ 230.

By default input is always String.

4 Conditionals

* if-else : (condition).

Here ;

let clargs = process.argv → command line arguments

→ process.argv जो होता है वो command line se input हमारे program को लाकर देता है।

let n= parseInt (clargs[2]);

Odd - Even program:-

```
if (n%2=0){
    console.log (n + " is even");
}
else
{
    console.log (n + " is odd");
}
```

Command Line.

```
node Input.js  10  → 10 is even
node. Input.js  7  → 10 is odd.
```

⊙ Syntax.

```
⊙ if (condition) {
        statements
    }
  else { statements
    }.
```

5. Loops.

```
let cmdargs = process.argv;
let n = parseInt(cmdargs[2], 10)
```

// Here 10 not mandatory

Printing Counting Program.

```
for (let i=1; i<=n; i++)
{
  console.log(i);
}
```

→ statement.

this operator tells javascript when to print the numbers.

Command line:
node input.js 10 ⟶

Output
1
2
3
4
5
6
7
8
9
10

Syntax: (for loop).

```
for(declare a variable;
    condition;
    increment decrement)
{
  ....
  ....
  ....
}
```

statements;

23 Sept, 2021 Setup for Web Development

Lec-21 Setup Installation.

CLASS-2.

Setup Installation :- Installing VS Code & N

1. Installing VS Code

Step 1) Download VS Code software from its official site.
i.e from "code.visualstudio.com/download"
→ ⊙ From here, you can choose various options as per your machine's configuration.
(i.e for Windows, MacOS & Linux.
⊙ Size of the file is 75 Mbs around (In Windows).
M1 Chip → Silicon

Step 2) Open that VS code file (after the download), then accept the license and then click NEXT (few times) & the VS Code get installed.

2. Configuring the VS Code

Step 1. Create a folder on your desktop and name it whatever you want. Now open the VS Code : Go to files ; then open folders ; Select your folder which you want to open. You can work now in this particular folder.

Installing NODE.JS.

Step 1) Download NODE.JS from its official site ie from "nodejs.org/en/download/"

- from here, you can choose from various options as per your machine's configuration (ie for Windows, MacOS, Linux).
- Size of the file is around 29 Mbs (for Windows).
- for LINUX, please watch the video recording.

Step 2) Open the node.js file (after download), then accept the license and then click next (few time) and the NODE.JS get installed.

Step 3) To check that node.js is installed.
What we need to do is, basically :-

1. Open the cmd (command line).
2. Type : Node --version (*node hyphen hyphen version*)
3. If your machine display any version more than 10.0.0, you are good to go. (Latest version till now is v14.7.6.
4. Your Node.JS is installed.

→ Cmd + Space → Type Terminal
→ node --version
v14.17.6.
→ npm --version
6.14.15

Extensions to run JS code.
first option
1. Javascript es6.
2. Code Runner from Jun Han

Run → Ctrl + Option + n.
Stop Running → Ctrl + Option + m.

Cmd + Space → type activity Monitor

To open VS code
Cmd + Space → type VS code return

DATE

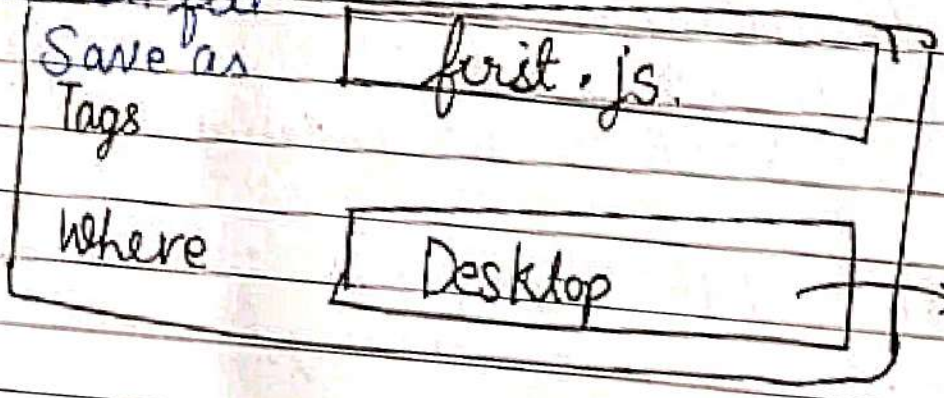
VS Code is now running in Apple Architecture

Familiar with VS Code

(Shortcuts for VS Code) (General)

1. → New file
2. → New folder.
3. ⊙ Go to files → Click on Auto save (to automatically save your word)

→ Command + N → To create a new file.
→ Click on file
→ Autosave ✓
→ Click on file



→ Select location where you want to save the file

4) Terminal
→ npm install prompt-sync.
→ Error in running
1) Code → Preference → Setting
2) Run in terminal
Go down → ☑ Run Code in Internal Terminal
Marka
Tick here.

5) → To open any file in the terminal, just go to that file, right click on that file & select "open in Integrated terminal". (How to open terminal for a particular file).

6) Now, when the terminal gets open, you can run the file by typing : node filename
node ./fisrt.js
Hello world
Hello world.
} Output.

7) Here also, we can check the version of node that is installed. (node --version)

8) for shortcuts (keyboard shortcut) go to setting button (bottom left corner), then click on Keyboard shortcuts. You can see various shortcut there.

1. ⊙ tab → terminal auto complete
2. ⊙ open/close terminal → [ctrl + `] *(` - Backtick)*
3. ⊙ for multiple views.
   ↳ slide bar → open close → [ctrl + b]
   ↳ We can make it as a canvas
4. // sidebar open close → cmd + b.

Cmd + space terminal

4. To split a file from multiple files ✓ Open the file you want to split & then click [ ] (Split Mode) present on the top right corner.

5. copy rapidly → [Shift + alt + down]
   [Esc] button to exit that.

6. Press [alt] (option) while clicking to have the multiple cursor support (press [esc] to move out).

7. To move launch of statement up & down all together, just select that statements & then press [alt + up/down arrows.] (option)

8. To search for any file present in the folder you are working, press ctrl + p.

9. To split horizontally, press [alt] (cmd) & then split button ([|]) to present on the top right corner.

Extension in VS Code

{ Open setting icon (bottom left corner) & then click extension

Note:- VS Code is actually similar to a web browser when it comes to code

→ eg → Bracket Pair Colorizer (to provide colors to bracket)
→ Live Server (to run a live server.)

Here in the extensions, just select what extension you want to install in the VS Code & after that just click on the [install] button. Your extension gets installed

Lec-22

CLASS-3

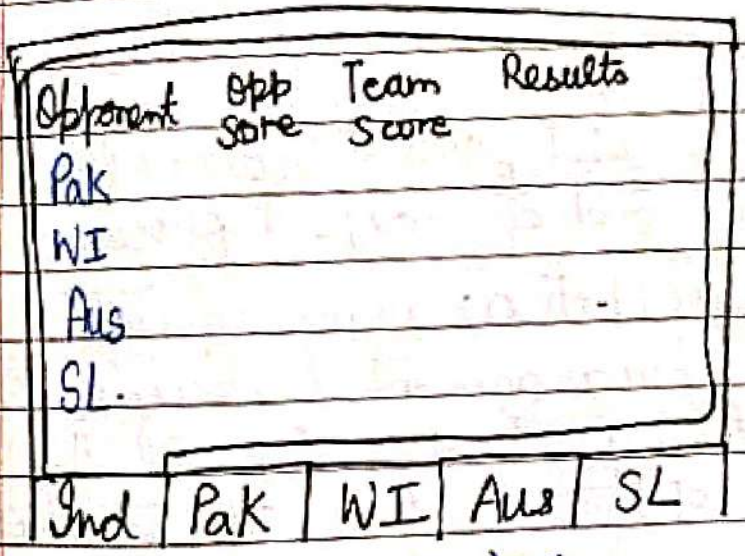
CLASS-3

Activity :- Web Scrapping

2.1 Read data from a source: Cricinfo World Cup 2019.
2.2 Process data: Get all teams
2.3 Write processed data in excel. Match results per team in their own sheet
2.4 Create folder: One for each team
2.5 Write files: PDF file for scorecard of each match in relevant folder.

WorldCup 2019 data निकालेंगे → इससे Web Scrapping वीडियो है।
Google → World Cup 2019 → Results & fixtures.

World Cup के सारे Match का data यहाँ होगा।



ये data होगा.

India

India vs Pak. pdf.
India vs WI. pdf.
India vs Aus. pdf.
India vs SL. pdf.

Pak.

Pak vs India. pdf.

1.1. First experience with functions
1.2. " " " arrays
1.3. " " " dependencies (external module).
1.4. " " " file manipulation.
1.5. " " " callbacks.
1.6 " " " timers
1.7 " " " downloading data from web.
1.8. " " " processing data from web
1.9 " " " writing data from web.
1.10. feirst " at writing pdfs.

Why Node.js ⇒ Mern Stack करना है।

1. First experience with function

Q.1. Print Name no. till n

```
let cLargs = process.argv; //similar to scanner.
```

जैसे Java में Scanner होता है, वैसे ही nodejs में process.argv

CLI से arguments उठाके program को provide करता है।

CLI पर सब कुछ string में आता है। हमें वो string form में नहीं चाहिए, int में चाहिए तो हम

```
let clargs = process.argv;
let n = parseInt(clargs[2]);

for(let i=2 ; i<=n;i++)
  let isPrime = [ IsPrime ](i);

  if (isPrime == true)
  {
  console.log(i);

  }
}

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

यहाँ let और var नहीं लिखते. बस argument का नाम लिखते।

vscode. control +`= terminal open.

cls → In window to clear screen
clear → " mac " " "
cmd+/ → at top window/ दोनों side coding area

Settings → 10 make your own shortcut & use some premade ones.

2. First experience with arrays.

जिसमे multiple values एक साथ होती है।

Use cmd/ctrl + b to open the sidebar.

→ let cargs = process.argv;
→ let n = parseInt(cargs[2]);
→ let arr = [];

JS में array का size बताने की जरूरत नहीं पड़ती।

यहाँ Java में array बनाते है तो size भी लिखना पड़ता है

[] लिखना पड़ता है

Java.

int[] arr = new int[6];

Size of array

→ for (let i=0; i<n; i++)
{

let val = parseInt(cargs[i+3]);

arr.push(val);

यहाँ पर array में push करेंगे, manually नहीं करते)

}

Q2) To print the values inside the array.
First take the Input & then return those values.

```
let clargs = process.argv;
let n= parseInt (clargs[2]);
let arr= [];
for (let i=0; i<n ; i++)
{
  let val= parseInt ( clargs [i+3]);
   arr.push (val);
}
```

Dry run

node fa.js 5 10 20 30 40 5

```
for (let i=0; i<arr.length; i++)
{
  console.log (arr[i]);
}
```

continue

```
console.log (arr);
console.log (arr.length);
```

```
let cliargs = process.argv;
let n = cliargs[2];
let arr = [];

for(let i=0; i<n; i++)
{
    arr.push(parseInt(cliargs[i+3]));
}
```

for loop in JS

or

```
{ let val = parseInt(cliargs[i+3]);
  arr.push(val);
```

```
for(let i=0; i<arr.length; i++)
{
    console.log(arr[i]);
}

console.log(arr);
console.log(arr.length);
```

→ node is a process to run the JS file, its a runtime. node.js में node वो program है जो इस program को interpret करके चला रहा है।

Output ↓

10
20
30
40
50
[10, 20, 30, 40, 50]
5.

npm के बहुत सारे modules लिखे पड़े हैं।

3. First experience with dependencies.

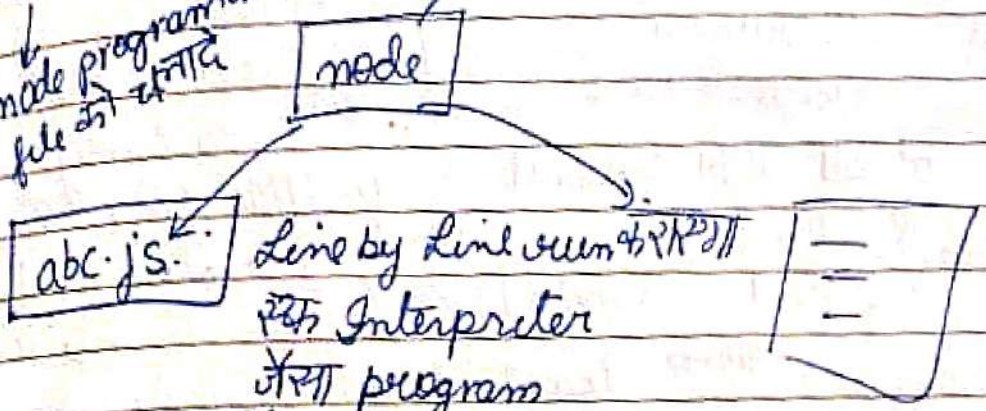
पहले से बहुत useful code लिखे पड़े हैं, हम उन्हे use कर सकते हैं।

NPM module

Libraries को रटना नहीं होता, समझना होता है।

node abc.js

node program abc.js file को चलाओ



Node एक Runtime Environment है।

Node
↳ RTE (Runtime Env)

1) Interpreter/Compiler
Program को Interpret या compile करके चलाने लायक बनाता है।

2) Runtime Environment
node abc.js
Libraries
3) Platform

node, abc.js को read करता है, Libraries होती हैं abc.js को file बनवानी है, कौनसी Library इस्तेमाल करनी है, वो code में लिखी हुई है।

NPM (npm is node package manager.
↳ ये Library downloader है।

1) There are libraries already there with node / already existent that can do the work we want to or can facilitate the same
2) Other than that most library are registered in npm repository

Node → 1) It is a Interpreter / Compiler.
2) It is a Runtime Environment
3) It is a Platform

Node वो करता है जो JS की file कहती है, on OS, JS file says को,

1) browser को automate करनी है।
2) Node JS की lines को चलाता है।
3) Excel की file में कुछ करना है तो वो library लिखी है।
4) HTML page से कुछ info निकालनी है।
5) google पर कुछ info निकालनी है।
6) सब running process में check करना है कि कौन कितनी memory use करता है।

npm एक site है। ← npm पर ये सब Libararies registered है।

तो हम अपने terminal (CLI) पर ये Libararies download करसकते है। खोलके Browser download करने की जरूरत नहीं है।
हमें बस लिखना है।

npm install library

Linux users must be familiar with apt get.
Python " " " " " pypi
Ruby " " " " " gem

Package managers.

node → RTE (Run Time Env).

abc.js → Node → Libararies

1) Node एक Run Time Env. है जिसको अगर हम abc.js file सौंप देते हैं
एक prog उस file में जो लिखत होगा, उसको ये पढ़ता रहेगा और
OS से कराता है।

2) Node के पास Libraries होती है बनी बनाई जो कि कुछ
Task perform कर सकती है। इसके अलावा और भी Libraries
है जो npm पर मिल सकती है।
npm एक repository है (database है).

↳ इसके अंदर लोगो ने बनाकर कई Libraries डाली है।
जिसको हम use कर सकते है।

node → एक Interpreter है जो program को
चलाता है।

npm → Downloader for Library.
↳ ये Library को download करके हमें
provide करवाता है।

Open Terminal → In the folder where we write programs

\Js > npm install minimist
↳ कोई भी Library

→ Google

→ minimist npm module

npmjs.com/package/minimist

↳ पहले Example फिर what does minimist actually do?

Example

```
$ node example/parse.js -x 3 -y 4
  -n5 -abc --beep=boop foo bar baz.
```

```
{ _: ['foo', 'bar', 'baz'],
  x: 3,
  y: 4,
  n: 5
  a: true
  b: true
  c: true
  beep: 'boop'
}.
```

var जो parse करके देता है
2 बार आती थी String में " "

→ let parser = require ('minimist');

library है minimist, इसको require से use करते हैं।

→ let args = parser (process.argv)

parser को दिया process.argv यानि input दिया।

Parser is minimist here

→ console.log(args) → [node, FirstModule] node और name of file को किसी _ में रखेगा

Input:
→ node FirstModule.js -x 4 -y 10.
2 3 4 5

अगर ये minimist नहीं होता parser variable में तो

minimist x को args की एक property बनायेगा।

]
x: 4
y: 10

args.x = 4
args.y = 10
ये automatically int लेगा instead of String

→ node FirstModule.js --name = "Sumeet Malik" --age = 34.

Double dash & single dash works the same here.

: [>node >FirstModule]
name: 'Sumeet Malik'
age: 34.
मतलब string

CLI से argements लेते हैं तो ये args.name हुआ और ये args.age.

```
let ~~parser~~ minimist = require ('minimist');
let args = ~~parser~~ minimist (process.argv);
let name = args.name;
let age = args.age;
if (age >30)
{
console.log ( "Hello" + name + "you must go to home");
}
else
{
console.log ("Heya" + name + "Where is the party tonight?");
}
```

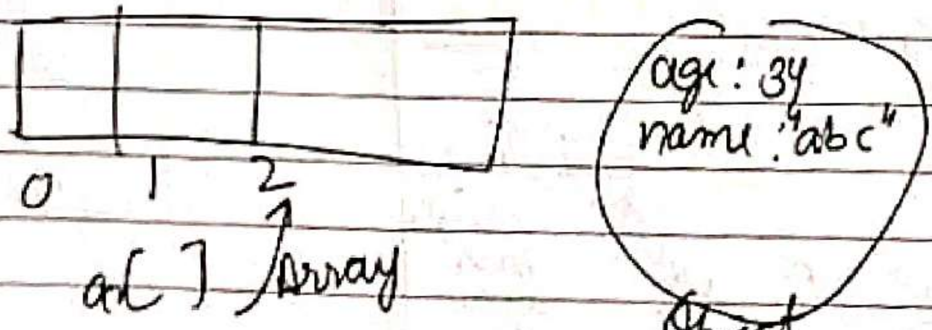
minimist में मैंने सभी argument pass कर दिया है।

args के अंदर args.name की property provide कर दी गई है।

args.age

args.name

इन दोनों name को एकसाथ change करना है तो alt press करके सब जगह click करो जो एकसाथ change करनी है।



Minimist ने हमें Object बना के दे दिया Strings को जो args में थी।

File Manipulation

4. First experience with file.

// read a file, Capitalize every word, Write the file

→ Install minimist if it's not installed in folder

→ let minimist = require("minimist");

→ let args = minimist(process.argv);

F1 (file 1) को पहले read करेंगे, उसके हर letter को Capitalize कर देंगे, उसको file 2 में transfer कर देंगे।

→ let fs = require("fs");
→ file system

fs में install करने की जरूरत नहीं पड़ी क्योंकि it's a runtime env. इसमें पहले से Library फाइल में पड़ी होती है।

file system का काम है file read करना, write करना।

Make f1.txt in same folder.

→ // node FirstFiles.js --source=f1.txt -dest=f2.txt
→ In f1.txt → cricket is a great game
it is also called gentleman's game
its origin came in English.
england recently lost to india.
ctrl+w → close.

source में f1.txt

इसको read करना है

→ let stext = fs.readFileSync(args.source, "utf-8");
source text. single

→ console.log(stext);

इससे string ही print होनी चाहिए।

String has split func ⇒ split ⇒ array

```
let words = text.split(" ");
console.log(words);
```

['cricket', 'is', 'a', 'great', 'game.\r\n', 'is',]

इससे हर एक word अलग-2 दिखाएगा

```
for (let i = 0; i < words.length; i++)
{
    words[i] = words[i].toUpperCase();
}
```

Array का join func space को join करेगा

```
let dtext = words.join(" ");
```

destination text — words Array के अलग-2 element को space separator से join करके एक String बनाएगा।

```
console.log(dtext);
```

```
fs.writeFileSync(args.dest, dtext, 'utf-8');
```

func to write in the destination file

bits format में text को convert करता।

Doc- 22
CLASS-4

DEV CLASS-4

25 Sept, 2021
Saturday

JAVASCRIPT

Todo
- 1.5 ⊙ first experience with callbacks.
- 1.6 ⊙ first experience with timer
- 1.7 ⊙ first experience with downloading data from web.
- 1.8 ⊙ first experience with writing data in excel file
- 1.9 ⊙ first experience at writing pdfs.
- 1.10 Take Input from source file, Make the letters capital and then copy the desired text at the desired destination

1.4 First Experience with file

Day 1

→ let minimist = require ('minimist');
// Using the library minimist.

→ let args = minimist (process.argv);
// parsing all the input to minimist library.

→ let fs = require ('fs'); // using the already installed library fs, node already have fs, we need not to install it.

→ let stext = fs.readFileSync (args.source, "utf-8");
↓
sourcetext.
// using fs.readFileSync function to read the data from the source file ie Source.txt

→ console.log (stext); // print the source text

→ let words = stext.split(' ');
// breaking the string into array
↳ space in b/w
↓
To Split the String

→ console.log(words); // now print the array ie the splited string

→ for (let i=0; i<words.length ; i++)
{
words[i] = words[i].toUpperCase();
}
To change the string letter to capital

// now converting the lowercase word array into uppercase word array

→ console.log(words);
// printing the upper case word array

→ let dtext = words.join(" ");

Space.

destination text

// now joining the words array & converting the array into string by joining it with spaces in b/w the words

→ console.log(dtext);
// printing the text ie to be printed in the destination file

→ fs.writeFileSync(args.destination, dtext, "utf-8");
// now writing the desired text at the desired destination

In CLI → node FirstExpwithFile.js --source source.txt --destination destination.txt

Way 2. Without Splitting the string to array & without joining the array to string

```
→ let minimist = require('minimist');
→ let args = minimist(process.argv);
→ let fs = require('fs');

→ let stext = fs.readFileSync(source, 'utf-8');
→ let dtext = stext.toUpperCase();
→ fs.writeFileSync(args.dest, dtext, "utf-8")
```

जैसे Java में import keyword पहले से है, वैसे ही JS में require keyword है।

1.5 First experience with Callbacks

Total time

Lack of Call Back

task 1 — 10 sec — 10 sec
task 2 — 10 sec — 20 sec
task 3 — 15 sec — 35 sec
task 4 — 15 sec — 50 sec.

पहले ज्यादा Time लगता था

कभी-2 computer में हर काम CPU से नहीं होता। Disk से भी होता है।
CPU उस time पर free होता mostly जब HD पर read/write वाला काम चल रहा हो।

With CallBack

CPU एक resource है। Hard Disk भी एक resource है।

T1 (10 sec) — Prime no. finding program.

T2 → ft.txtread — 5 sec.

T3 (15 sec) — Prime square number finding program.

T4 → f2.txtwrite — 7 sec

अब Time कम लगा।

15 sec.

अब 2 tasks simultaneously चल रहे हैं तो Time waste नहीं होगा। Cpu का wait time कम हो जाएगा।

Disk

t1 = f1.txt read
10 sec

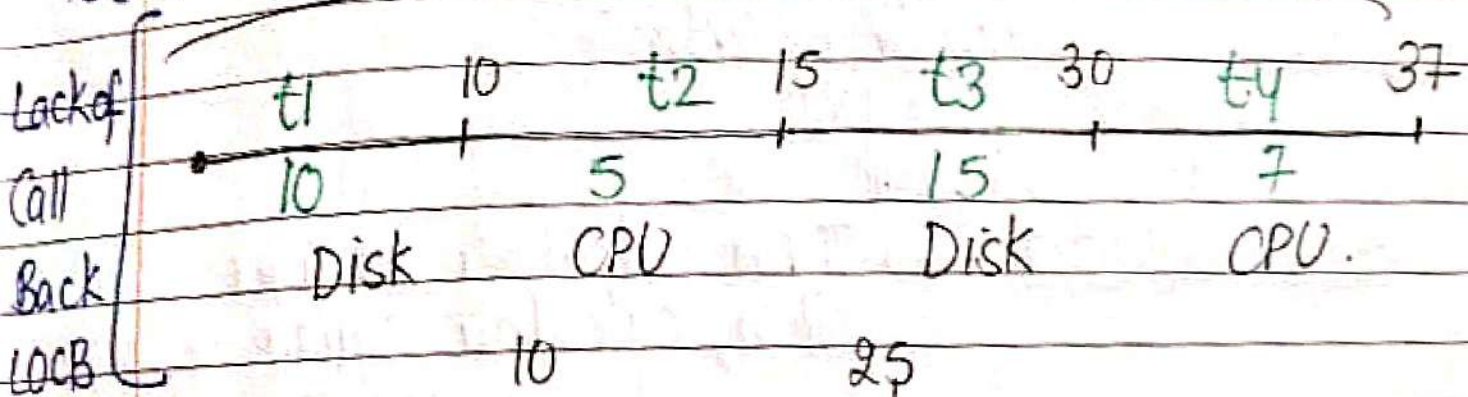
t3 = f2.txt write
15 sec

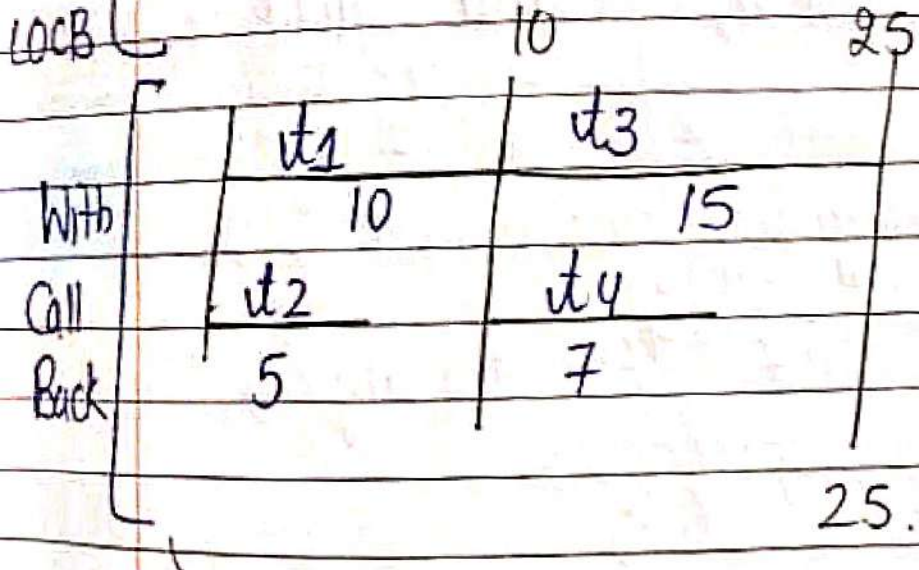
CPU

t2 = Prime
5 sec

t4 = PrimeSquare.
7 sec

In Series





In Parallel.

Q.1. Lack of Call Back (LCCB)

Only these 2 will be done for now

// t1 = Read a file (disk)
// t2 = Calculate Prime (CPU)
// t3 = Write a file (disk)
// t4 = Calculate Square of Prime (CPU)

T2 is done after T1

CLI

```
// node FirstLackOfCallBack.js
--source = f1.txt
--dest = f2.txt
--n = 50000 → Till n calculate all Prime.
              Till n, calculate square of all prime
```

0, 1, 2 index रखने की आवश्यकता नहीं पड़ी अतः because of minimist

```
→ 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
50,000
```

Now comment down all the console.log statements

इस तरह last के 5 digit print करा रहे हैं।

ये फिलहाल का time t1 मे डाल देगा in milli seconds

```
let t1 = Date.now();
console.log("starting task 1 at" +
  t2 % 100000);
let stext = fs.readFileSync(args.source,
  "utf-8");
```

↳ bits/0-1 मे data नही होगा अगर utf नहीं लिखेंगे

```
let t2 = Date.now();
console.log("finishing task 1 at" +
  t2 % 100000);
console.log(t2 - t1);     * //Duration *
```

TASK 1

Starting Task 1 at 22433.
Finishing Task 1 at 22442

] 9 milli second लगी।

f1.txt को थोड़ा बड़ा रखना ताकि reading time अच्छे से दिख सके।

TASK 2

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

TASK 2

```
function isPrime(x)
{
    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("Finishing task 2 at " + t4%100000);
console.log(t4-t3);
console.log("Total time till now is " + (t4-t1));
```

Terminal:

```
node firstLackofCallBack.js --source=f1.txt --dest=f2.txt -n=40000
Starting task1 at 92196
Finishing task1 at 93754
809
Starting task2 at 93007
Finishing task2 at 93754
747
Total time 1558
```

T1 Task: 92196 – 93005 = 809 ms

2 milli sec

T2 Task: 93007 – 93754 = 747 ms

1558

Callback म्हणजे function ला function पाठवणे, argument म्हणून function पाठवणे।

Q.1. Now, With CallBack

//t1 = Read a file (disk).
//t2 = Calculate Primes (CPU)

T2 is done in parallel with T1.

CallBack (Asynchronous Method).
(filename = firstcallback.js)

T2 is done in parallel with T1.

```
function isPrime(x) {
  ______
  ______
  ______
}
```

isPrime चा func इथे लिहायचा

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

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

task

```
fs.readFile(args.source, function(){});
```

fs.readFile → func; args.source → parameter1; function(){} → parameter2 is a function, function ला एका func ला देणे हे known as CallBack.

not readFileSync

fs.readFile(..args.source,) Parameter 1

जब तक ये data read हो रहा है

Parameter 2 { function () { } }

तब तक ये func चलेगा।।.

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

→ Callback.

→ fs.readFile(args.source, Function (data)

{

// Task 2

इसमें जब तक ये file read हो रहा है Task चल रहा है, चलने दो और नीचे वाले task करो, जैसे ही ऊपर वाला Task हो जाए, और file भी read कर ली जाए उसके बाद function execute करा देना, उसके बाद Task 2 खत्म हो जाए तो कर देना।

{ let t2 = Date.now();
console.log("finishing task 1 at " + t2 % 100000);
console.log(t2 - t1);
});

Task 1 still executing
Task 2 started
" " done.
" 1 done

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

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

function isPrime(x){
  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("finishing task 2 at" + t4%100000);
console.log(t4-t3);
// console.log("Total time of the duration" + t4-t1);
```

Terminal

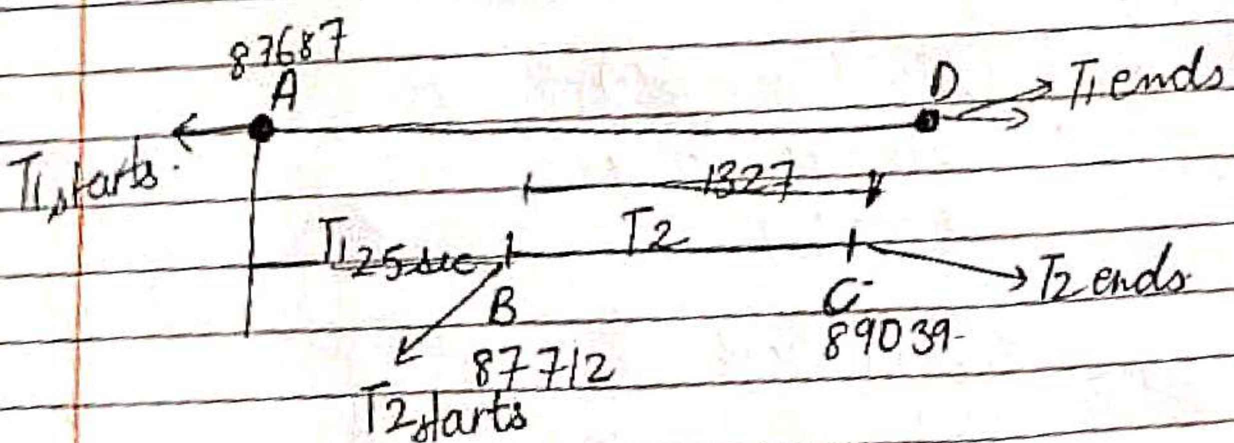
node FirstCallback.js --source = f1.txt
--dest = f2.txt
--n = 50000

Starting task 1 at 87687
Starting task 2 at 87712
finishing task 2 at 89039.
1327

finishing t1 at 90080
2393

★ Note: Node.js ने ~~Call~~ Multi Threads को Callback के पीछे छिपा दिया।

Callback is basically a technique.
जब एकसाथ बहुत से काम करवाना होता है। (similar to parallel computing)



Explanation:-

Here, as you can see;

Task T1 get started at A.
In the middle of the Task T1, Task T2 gets started at point B, after 25 milliseconds of task (T1)'s execution

Then;
Task (T2) ends at point C, taking 1327 milliseconds to get completed.
& after that;
Task (T1) gets completed at point D taking 2393 milliseconds in total.

This, is how parallel tasks, can work in Javascript.

Javascript ऐसी कोई features provide नहीं करता कि हम Threads बना सके लेकिन Behind the scene Threading provide करता है with the help of Multi CallBack

Lec-24
Class 5

Dev Class 5 27 Sep 2021

1.1 First experience with function
1.2 " " " arrays
1.3 " " " module/dependencies
1.4 " " " file manipulation.
1.5 Create a Big file.
1.6 First experience with lack of callback.
1.7 " " " Callback.
1.8 " " " downloading data from web
1.9 " " " processing data from web.
1.10 " " " writing JSON.
1.11 " " " reading JSON.
1.12 " " " array manipulation.
1.13 " " " writing data in Excel file.
1.14 " " " creating folders.
1.15 " " " writing pdfs.
1.16 " " " with Timer

TODO

ये नये folder में minimist install करना होगा।

1.5 Create a Big File

→ filename: CreateBigFile.js

CLI → // node createBigFile.js --dest=Big_data

Create this file in folder big data

CLI → npm install minimist
→ let minimist = require("minimist");
→ let fs = require("fs");
→ let args = minimist(process.argv);

```
let arr = [];
for(let i=0; i<500000000 ; i++)
{
 arr.push(i);
}
let str = arr.join('\n');
```

→ एक Line के gap से join हो जाओ और String बनाद array की।

```
fs.writeFileSync(args.dest, str, "utf-8");
fs.appendFileSync(args.dest, str, "utf-8");
fs.appendFileSync(args.dest, str, "utf-8");
fs.appendFileSync(args.dest, str, "utf-8");
```

Big Data] Reveal in file explorer → Now see its Size.

1.6. Lack of Call Back → Task 1 के बाद Task 2 भी चल सकते हैं।
1.7. Call Back → Task 1, Task 2 parallel
1.8. Downloading data from web → Web downloading
कैसे करते हैं। (Cricinfo से Match Data निकालना)
1.9. Processing data from web → Download html data
होगी, इस file से से उस information निकालनी है।
1.10. First Experience with writing JSON

JSON → Javascript Object Notation

info को निकाल JSON में convert करो।

1.5

Minimist को globally install किया जा रहा है। अब ये हर एक folder में अलग से install करने की ज़रूरत नहीं पड़ेगी।

file name :- CreateBigFile.js

→ npm install minimist -g

→ let minimist = require ("minimist");

→ let fs = require ("fs");

→ let args = minimist (process.argv);

→ console.log (args.dest);

Cli → > node 5.CreateBigFile.js --dest=big data → .data जैसा कोई ext नहीं है ऐसे ही लिखेंगे।

big data

↳ Output = args.dest.

```
→ let arr = [];
→ for (let i=0; i<50000000; i++)
  {
    arr.push(i);
  }
→ let str = arr.join ("\n");
```

(50000000 → we took example of 50; for now try for 50 2gb file)

array बना लिया। Array में 5×10^7 elements हैं।

एक string बना रहे हैं जिसमे array के हर element को जोड़ रहे हैं with \n in b/w.

eg a)

```
let str = arr.join (",");
console.log(str);
```

→ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 ——— $(5 \times 10^7 - 1)$

g b)

```
let str = arr.join("\n");
console.log(str);
```

0
1
2
3
4
5
6
1
1
1
$(5 \times 10^7 - 1)$

Eg:

```
let str = arr.join("|");
console.log(str);
```

→ Pipe

0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 — $(5 \times 10^7 - 1)$

→ console.log(str);
→ console.log(arr);

0
|
$(5 \times 10^7 - 1)$
[0, 1, 2, — $5 \times 10^7 - 1$]

writeFile सब कुछ file से हटा के शुरू से लिखता है।

Append, पहले वाले data को रहने देता है और उसके बाद ही शुरू करता है, add करता जाता है।

आगे

```
→ fs.writeFileSync(args.dest, str, "utf-8");
4 बार fs.appendFileSync(args.dest, str, "utf-8");
0-4 fs.appendFileSync(args.dest, str, "utf-8");
  fs.appendFileSync(args.dest, str, "utf-8");
```

यही file 2gb तक की बन जाएगी। 2gb से बड़ी file मत बनाना क्योंकि वो फिर upload नहीं हो पाती क्योंकि RAM इतनी ज्यादा नहीं होती।

CLI
- > node 5-CreateBigFile.js --dest=big.data → 420mb
- > node 5-CreateBigFile.js --dest=big.data → 420mb
- > node 5-CreateBigFile.js --dest=bigdata → 420mb
- > node 5-CreateBigFile.js --dest=big.data → 420mb

fs.writeFileSync → अगर file पहले से नहीं है तो write new file बना देगा।
- 1) file नहीं पड़ी तो बनादेता है।
- 2) पहले से पड़ी है तो data delete करके शुरुआत से लिखता है।

fs.appendFileSync → Append के पहले से file चाहिए नहीं है तो ये भी बनादेगा।
- 1) जो पहले से पड़ा है, उसके नीचे ही content लिखता है।
- 2) कुछ भी मिटाता नहीं है।

1.6 Just Experience with Lack of Call Back

→ npm install minimist -g

// t1 = Read a file (disk)
// t2 = Calculate primes (CPU)
// t2 is done after t1
// node 6_FirstLackOfCallBack.js --source=bigdata

Source =

→ Reading Big Data
Calculate Prime till n ←

t1	(fileread).
t2	(CPU (Prime))

ये दोनों Task 1 के बाद 1 करके होंगे।
ये दोनों साथ हो सकते थे with the help of CallBack लेकिन अभी के लिए Series में ही होंगे, parallel में नहीं (because it is Lack of Call Back).

→ let minimist = require ('minimist');
→ let args = minimist (process.argv);
→ let fs = require ('fs');
→ console.log (args.source);
→ let t1 = Date.now(); → console.log ("Task1 started at" + t1 + 100000);
→ let data = fs.readFileSync (args.source, "utf-8");
→ Time 2
→ let t2 = Date.now();
→ console.log ("Task1 ended at" + t2 + 100000);
→ console.log (t2 - t1);

बड़ी file को text form में read करो तो यही करना होगा।

Task 2

→
```
function IsPrime(x)
{
let isPrime = true;
for (let div = 2; div <= x-1; div++)
    if (x % div == 0)
    {
    isPrime = false;
    break;
    }
}
return isPrime;
}
```

→
```
let t3 = Date.now();
console.log("Task 2 started 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("Task 2 finished at" + t4 % 10000
console.log(t4-t3);
console.log("Total time in task" + (t4-t
```

Print Prime Till n

Write file & writeFile Sync?

konsa better है?

→ writeFile() better है क्योंकि वो non-blocking है।
main thread को consume नहीं करता, वो
parallel में चलता रहता है।

JS में Thread word use करने से बचना चाहिए।
JS behind the scene, do multithreading.

2 Task Concurrent चल पाते हैं।

Task 2 पहले शुरू करदिया तो Task 1 साथ में नहीं होगा
फिर तो Task 2 ही पहले होगा।

Lec-26
Class-6

Lec25 → How to improve DSA & Coding in Contest Prog.

28 Sep, 2021

PAGE NO.
DATE

DEV CLASS - 6

1.1. Frist experience with function → First function.js (1s Prime)
1.2 First experience with array → First Array.js
1.3 First experience with module → First module.js
1.4 First experience with file manipulation → First Files.js
1.5 Create a big file. → Create Big File.js
1.6 Frist exp with Lack of Call Back → FristLackOfCallBack.js
1.7 " " " CallBack.
1.8 " " " downloading data from web
1.9 " " " processing data from web
1.10 " " " writing JSON.
1.11 " " " reading JSON.
1.12 " " " array manipulation
1.13 " " " writing data in excel file.
1.14 " " " creating folders.
1.15 " " " at writing pdfs.
1.16. " " with timers.

Install Minimist in current folder

16 → First lack of Call Back.js

// t1 = Read file (Disk).
// t2 = Calculate prime (cpu)
// t2 will be done after t1 which is not good

→ cli → // node FirstLack Of CallBack.js --source=big.dat --n=70,000

→1. let minimist = require ("minimist");
→2. let fs = require ("fs");
→3. let args = minimist (process.argv);

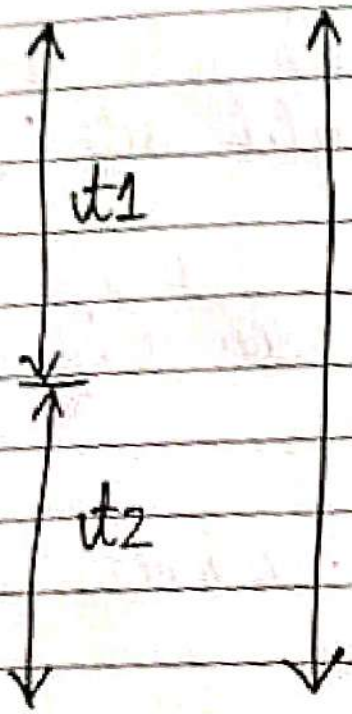
// task 1
let t1 = Date.now(); // no of millisiconds elapsed since 1st Jan 1970
console.log ("Task 1 started at" + t1 % 100000);
let data = fs.readFileSync (args.source);
let t2 = Date.now();
console.log ("Task 1 finished at" + t2 % 100000);
console.log (t2 - t1);
→ let t3 = Date.now(); no. of millisecond elapsed since 1 Jan 1970
// task 2. → console.log ("Task 2 started at" → t3 % 100000);
for (let i=0; i<args.n; i++)
{
let isPrime = IsPrime (i);
if (isPrime == true)
{
arr.push (i);
}
}
let t4 = Date.now();
console.log ("Task 2 finished at" + t4 % 100000);
console.log (t4 - t1);

```
function (IsBumi (x){
    let isBumi = true;

for (let div = 2; div < x-1 ; div ++)
{
    let isPrime = IsPrime (i);
    if (isBumi == true)
    {
        console.log (i);
```

Output

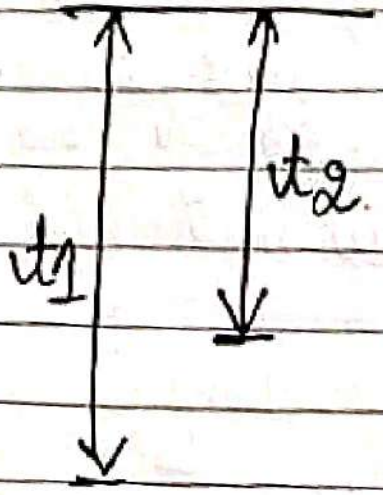
```
CLI > node FirstLackofCallBack.js --source=big:data --n=70000
Task 1 started at 62072
Task 2 finished at 65387
3315
Task 2 started at 65394
Task 2 finished at 67884
2444
5756
```

LACK OF CALL BACK.



$t_1 + t_2$ = Time taken by t_1 & t_2.

CALLBACK.



Max(t_1, t_2) = Time taken by t_1 & t_2

107

→ function के अंदर function डालते हैं CallBack.

→ FirstCallBack.js

//t1 = Read a file (disk)
//t2 = Calculate primes (cpu) parallel with t1, t2 will be done in parallel with t1, which is good.

→ Q) //node FirstCallBack.js --source-big.data --n=70000

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

let t1 = Date.now();
console.log ("Task1 started at " +
        t1 % 100000);
// let data = fs.readFileSync (args.source);
```

→ readfile sync doesn't block the flow

ये method Old है (Not as good at new method)

Read file blocks the flow

error आती है तो यहाँ डालते

data है तो यहा डाले

```
fs.readFile (args.source, function (error, data):
```

→ No name given func

अगर data मिल गया तो second parameter ie a function में भेज देगा otherwise error भेज देगा

if error = आती है

```
let t2 = Date.now();
console.log ("Task1 finished at " + t2 % 100000);
console.log (t2-t1);
} else { console.log ("Error happened"),
});
```

fs readfunc के args

function Newfunc (err, data).

Time To Print

Error डालना है तो first parameter में आएगा

Data डालते हैं तो second parameter

We need not declare data as parameter, data bcz data is passed to func की read file ही pass करेगा।

यहाँ पर program wait करता रहेगा जब तक Task पूरा नहीं होता।

```
func (isBuni(x){
}

{ let t3 = Date.now();
console.log("Task2 started 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("Task2 finished at" + t4 % 100000);
// console.log(t4-t1);
```

t4-t1 और t2-t1 में से जो बड़ा होगा वो होगा Total Time.

Output.

Task 1 started at 70380
Task 2 started at 70395
Task 2 finished at 72219.
1824.
Task 1 finished at 74890.
4510.

Lack of CallBack	CallBack
t1 | 3.5sec	t1 t3 | 4.5sec | 1.8 sec t4.
t2 | 2.4sec	t2 4.5 sec
5.9sec	

FirstWebDownload.js How to download from web using node module

1.8) First experience with downloading data from web

CLI → npm install axios → using this library
axios library

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

CLI → node FirstWebDownload.js --dest="download.html" --url="https://www.pepcoding.com"

--dest="download.html" : In which file we want to store the downloaded data.

--url="https://www.pepcoding.com" : from which url we want to download the data.

→ let args = minimist(process.argv);
→ console.log(args.url);
→ console.log(args.url);
→ console.log(args.dest);

There are many libararies to download the data from the web

→ downloadPromise वाला & axios library

→ let dldPromise = axios.get(args.url);

→ get data from url

dldPromise.then(function(response) {
fs.writeFileSync(args.dest, response, "utf-8");
}).catch(function(err) {
console.log(err);
});

→ यह func तब call होगा जब error आएगा।

→ let dwldPromise = axios.get(args.url);
→ dwldPromise.then(function(response) {
    let html = response.data;
    fs.writeFileSync(args.dest, html, "utf-8");
});

then function को response में data आता है

ESPN Cricinfo site पर match का data निकालना है

→ CLI 'node FirstWebDownload.js --url="https://www.espncricinfo.com/series/icc-cricket-world-cup-2019-1144415/match-results"

Google → Cricinfo 2019 World Cup
Fixtures & Result
Copy the url
& paste in the cli after url

→ fs.readFile() API का style है Async error भी संभालना है, और func भी

Lec-27
Class-7

29 September, 2021

PAGE NO
DATE

Dev Class-7

TAUGHT

1.1 First experience with function
1.2 First experience with array.
1.3 First experience with module
1.4 First experience with File Manipulation
1.5 Create a big file
1.6 First experience with lack of CallBack
1.7 First experience with CallBack
1.8 First experience with downloading data from web
1.9 First experience with processing data from web
1.10 First experience with writing JSON
1.11 First experience with reading JSON
1.12 First experience with array manipulation
1.13 First experience with writing data in excel file
1.14 First experience with creating folder.
1.15 First experience with writing pdfs.
1.16 First experience with timers

1.8 First experience with Downloading from web.

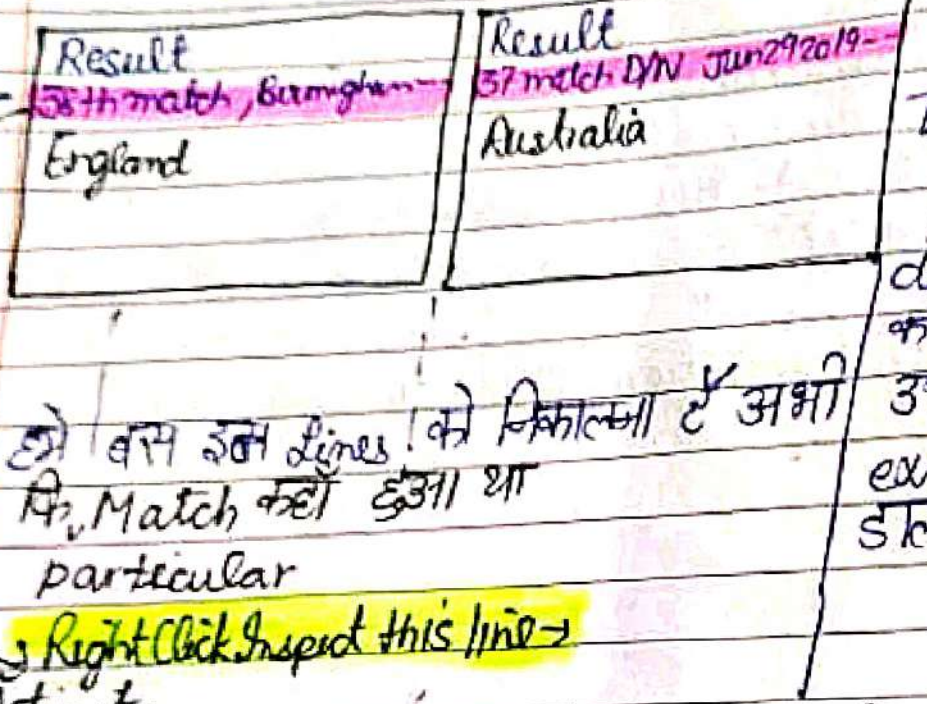
→ Name of file → FirstWebDownload.js
CLI → npm install axios
→ node FirstWebDownload.js --dest="download.html" --url="https://www.espncricinfo"

```
→ let minimist = require ("minimist");
→ let fs = require ("fs");
→ let axios = require ("axios");

→ let args = minimist (process.argv);

→ let downloadPromise = axios.get (args.url);
→ downloadPromise.then (function (response)
  { let html = response.data;
    fs.writeFileSync (args.dest, html, "utf-8");
  }).catch (function (err) { console.log (err); }
          );
```

1.9 First experience with processing data from web

ESPNCRICINFO

Google → Cricinfo 2019 World Cup
→ Fixtures & Result
→ हमें बहुत सारे Matches दिख रहे होंगे।



हमें अब यह data html form में नहीं चाहिए, HTML form में download तो करलिया। But अब उसको Process करके excel sheet में डालना है।

→ हमें बस उन Lines को निकालना है अभी कि Match कहाँ हुआ था particular
→ Right Click Inspect this line →
Activity

2.1 → Read data from a source : cricinfo world cup 2019.
2.2 → Process data: Get all teams.
2.3 → Write processed data in excel : Match result per team in their own sheet.
2.4 → Create folders : One for each team.
2.5 → Write files : pdf file for scorecard of each match in relevant folder.

तो अब हम Data को Process करने वाले हैं।
हमने Data तो download करलिया, get भी करके write भी करलिया था download.html में, अब उसको process करना है उसको excel की form में लिखना है।

तो जो Library अब हम use करने वाले हैं, वो हैं JSDOM.

Filename → First Processing HTML.js

Q1 → node FirstProcessingHTML.js --source=download.html

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

↓ download वाली file read करने के लिए

```
→ fs.readFile (args.source, "utf-8", function (err, data)
{ console.log(data);
})
```

अगर "utf-8" नहीं लगाया तो Buffer या bytes के form में data दे देगा

```
→ fs.readfile ( args.source, "utf-8", function (err, data)
{ let JSDOM = jsdom.JSDOM;
→ let dom = new JSDOM (data);
→ let document = dom.window.document;
console.log(document.title);
})
```

सबसे पहले JSDOM निकालो

JSDOM का एक नया object बनाया जिसमे html pass किया

ये line तो ऐसे ही लिखनी है अभी के लिए। अब इस dom के अंदर document नाम की चीज है और हमे वो document ही निकालनी है।

ये capital letters JSDOM होता है jsdom के अंदर, हमे इसे use करना है so इसे एक variable मे डालो JSDOM नाम के।

↳ title यहाँ print होजाएगा।

Google → espncricinfo.com
→ Right Click → Inspect
Browser हमे कुछ और show करता है। ये window /website पर जो show होता है, वो तो actually codes है।
or
→ Right click → View Page Source
पूरी हमने download किया है।
Code लिखे होंगे

→ Browser इस Code को read करता है और हमे उसे gui form मे present करता
→ Server तो img लिखा भेजता है, google chrome हमे real image show करता होगा है।
→ Browser reads the HTML
→ Browser की responsibility है कि Browser reads the HTML sent by server & presents a view

```
<button value = "run">
```

run

Server Browser

→ Browser 2 काम करता है।

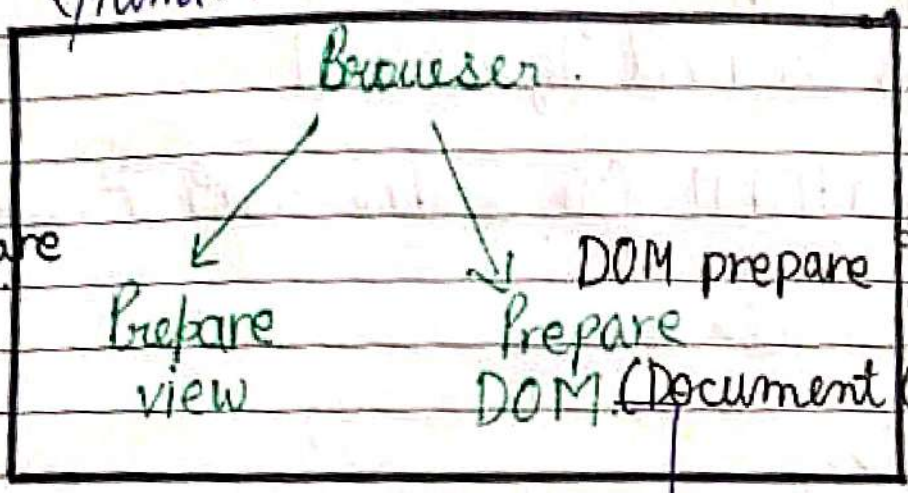
read करके view provide करता है

document
read करके object बना देता है। (dom) जिसको हम DOM कहते है।
Document Object Model

Server → HTML → <html> ... </html>

Server browser को HTML देता है।

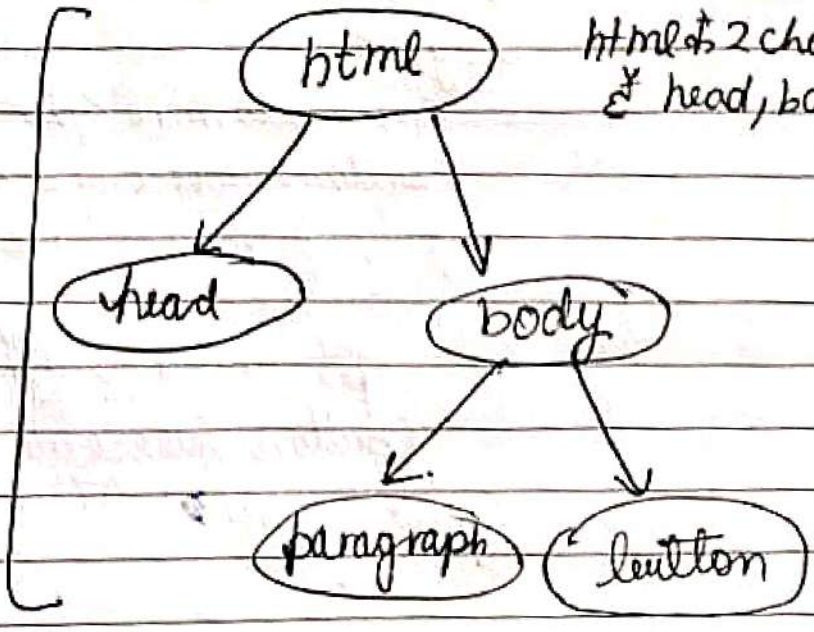
```
<html>
  <head>
  </head>
  <body>
    <p>  </p>
    <input type="button" />
  </body>
</html>
```



View prepare करता है user के लिए

DOM prepare करता है programming के लिए

DOM एक tree है

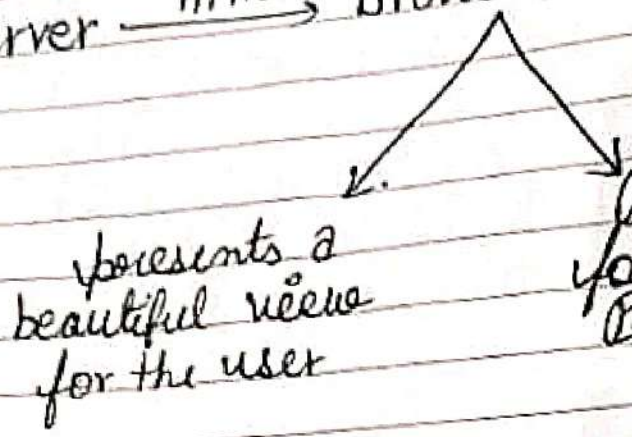


DOM

html के 2 child है head, body

Server → HTML file

Server —Html→ Browser reads the html.



Q → What is DOM?
DOM is Document Object Model.

Eg → Create a HTML file & then create its DOM.

File name:- sample.html.

```
<Html>
  <head>
  <title> My Sample Page </title>
  </head>
  <body>
    <p> Hello There </b>
    <button> Awesome 1 </button>
    <input type = "text" placeholder = "Sample1">
    <div>
      <p> abcdefghijklmnopqrstuvwxyz </p>
      <button> Awesome 2 </button>
      <input type = "text" placeholder = "Sample2">
    </div>
  </body>
</Html>
```

title.

↳ eg → जैसे ही zoom meeting खतम हुई, ऊपर लिखा आता है Tabs मे

Post Attendee-Zoom X

ये अलग है Tab

In current folder there will be a file created named Sample.html.

ये है Title.

<title>Post Attendee-Zoom </title>

Open with Google Chrome.

Output

My Sample Page x | New Tab x

Hello there

Awesome 1 | Sampl 1

abcdefghijklmnopqrstuvwxyz

Awesome 2 | Sample 2

↓

Right Click & do Inspect

head पर hover करे तो blue नहीं दिखेगा क्योंकि head website में दिखता नहीं है।

Elements

<html> → इसपे hover करेंगे तो पूरी screen blue दिखेगी।

▶ <head> ... </head> ← Can be opened up.

▼ <body>

<p> Hello there </p> → इसपे hover करे तो paragraph वाली line blue-2 दिखेगा।

<button> Awesome 1 </button>

<input type="text" placeholder="Sample1">

▶ <div> ... </div>

</body>

</html>

Browser ने 2 awesome चीजे की

1. हमे एक Simple से Code की जगह एक beautiful view दिखाया।
2. Browser ने programmer के लिए एक DOM Tree बनाया

Sample.html

```
<html>
  <head>
    <title>My Sample page</title>
  </head>

  <body>
    - <b> Hello there </b>
    - <button> Awesome 1 </button>
    - <input type = "text" placeholder = "Sample1">
    - <div>
      o <b> abcdefghijklmnopqrstuvwxyz </b>
      o <button> Awesome 2 </button>
      o <input type = "text" placeholder = "Sample2">
    </div>
  </body>
</html>
```

paragraph

button

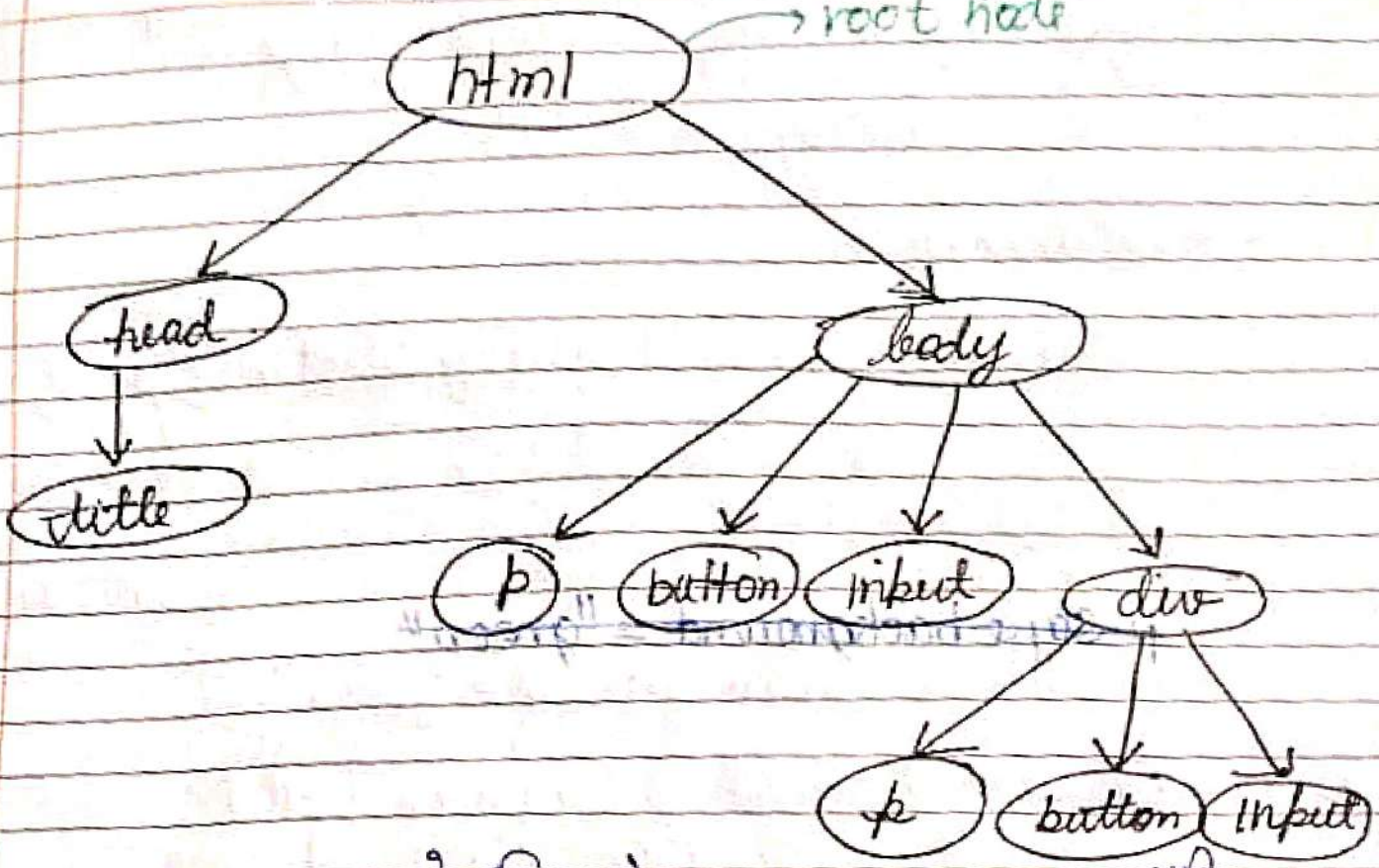
Text box

```
<button class = "b"> Awesome 2
</button>
```

Browser ने 2 काम किये

1. View दिया user को
2. Programmer के लिए DOM Tree बनाया।

DOM



Programmer के लिए ये DOM Tree बनाया ताकि Programmer इस DOM tree में से information निकाल सके।
इस Tree को Document Object Model बोलते हैं।

* DOM के अंदर काफी चीजे होती हैं.
eg→ Window
example→ अगर Program से हमें Window small करनी है तो

Programmer इसको Programming से change कर सकता है।

Sample html in Google → Right Click → Inspect → Console
> window.size (500, 500)
> window.open ("www.google.com")
> window.open ("www.facebook.com")
> window.close()
↳ ये Page sample.html चला जाएगा।

अगर हो Hello There का text change करना हो तो ?
Paragraph है ये तो

→ Console

let p = document.querySelector("p");
↳ 2 Para
है 1st Para आजाएगा

p.textContent = "aksdfjf".
p.textContent = 'strange'
p.style.color = "red".

p.style.backgroundColor = "green"

तो हमने ये सीखा कि Browser ये जो DOM देता है इससे हम information निकाल भी सकते हो।
2) चीजें change भी कर सकते हैं।
3) DOM से programmer browser को control कर सकता है।

DOM वाला काम तो Browser पर होता है।
हमने तो data download- html में रखा है।
JSDOM will load html & prepare the DOM for programmer just like the Browser do for the HTML that server provide it.

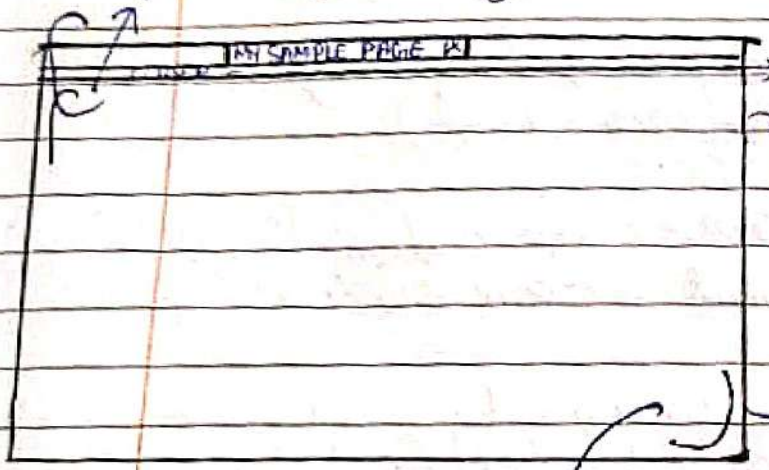
हमारे program के लिए HTML को load करके DOM बनाएगा JSDOM

DOM is an object tree which is created by Browser for the programmer. Programmer can interact with the view of page with DOM.

हमारे पास Browser नहीं है अभी हमारे program का DOM बनाने के लिए
JSDOM Browser की एक responsibility निभाएगा

Browser की 2 responsibilities थी
1) View दिया user को
2) DOM बनाना Programmer के लिए
यह responsibility JSDOM ने ले ली

window DOM पूरे browser को represent करता है.



url यह चीज dom.window change करेगा है.
specifically यह changes करने के लिए होता है dom.window
document.
Document की मदद से पूरे Page में से हम कुछ भी information निकाल सकते हैं।

Document → html को represent करता है, head

पूरे Browser को देखना → यह काम window देखती है
Content को देखना / page के अंदर देखना, यह काम document देखती है।

First Processing HTML.js

CLI → node First Processing HTML.js --source download.html

```
→ let minimist = require("minimist");
→ let fs = require("fs");
→ let jsdom = require("jsdom");
→ let args = minimist(process.argv);
→ fs.readFile(args.source, "utf-8",
        function(err, html){
        { let dom = new jsdom.JSDOM(html);
          let document = dom.window.document;
          let bz = document.querySelector(".b");
          console.log(bz.textContent);
```

class → class ".b" → पहला

```
→ let elements = document.querySelectorAll(".button");
→ console.log(elements.length);
```

दो तब देंगे जिनमें class b है।

```
→ console.log(elements[0].textContent);
→ console.log(elements[1].textContent);
```

जिस button के अंदर
class Button है। ← button of class

```
let elements = document.querySelectorAll("button");
console.log(elements[0].textContent);   Awesome1
console.log(elements[1].textContent);   Awesome2
```

For Processingdownload.js

```
fs.readFile(args.source, "utf-8",
            function(err, html)
{
    let dom = new jsdom.JSDOM(html);
    let document = dom.window.document;

let descs = document.querySelectorAll("div.description");
for (let i = 0; i < descs.length; i++)
{
    console.log(i, i.textContent);
}
```

description — जो matchinfo के अंदर लगी हो ← div.matchinfo; जितने desc लगी हैं। ; classname

Result
Match — 34th match, Chester-le-Street, 28 Jun 2019
Afghanistan → Right Click & Inspect

इसी तरह यहाँ div description चाहिए जो match info के अंदर है।

<div class="description"> ... </div>

class styling game के लिए होता है।

इस div पर class description है। मतलब class description है।

ये Project Data Analysis का Project था, web scrapping करके info निकाली।