Lecture-2 of JS

Strings :-

let a = 10

1st method → let s1 = "string1" + a string1 10

2nd method → let s2 = 'string2' + a string2 10

3rd method → let s3 = `string3 ${a}` string3 10

***

→ methods:

let charAt4 = str.charAt(4); console.log("charAt 4", charAt4);

let slicedStr = str.slice(4, 10); console.log("sliced", slicedStr);

*String → array of string

let arraystr = str.split(" ");

toLowerCase, toUpperCase

* Remove whitespace from front & back

let trimmed = str.trim();

* Array → to string ***

let str = arrstring.join(" + "); → need to mention on the basis of what

+ to convert into number Number( )

→ Arrays :-

let arr = [1, 2, 3, 4, 5]

for (let i = 0; i < arr.length; i++)

console.log(i, " :", arr[i])

* push/pop → add/removeLast
* unshift shift → add/remove First
* slice → gives a copy of a subarray
* splice → deletes any no. of elements
* indexOf, contains

a = [1, 2, 3, 4, 5]

a.push(10) 1, 2, 3, 4, 5, 10

a.pop() 1, 2, 3, 4, 5

a.unshift(16) 16, 1, 2, 3, 4, 5

a.shift() 1, 2, 3, 4, 5

slice = a.slice(1, 4) → (2, 3, 4)

↳ doesn't change the original array

spliced = a.splice(1,3) → (2,3,4)

↳ delete elements

Changes the original array

indexOf :-

idx = arr.indexOf(11);

~~contains~~ → T/F includes

string to array - JSON.parse(input)

↳ string looking like array

→ [array, function, object]

```
function fn()
{
  console.log("I am a function");
  return "Hello"
}
let tempArr = [1, 2, 3, 15, 5]
let tempArr = tempArr
let arr=[
  1, true, 1.1, "string", null, tempArr, fn];
```

```
console.log("2dArray", arr[arr.length-2]);
console.log["access the last element", arr[arr.length-1]];
console.log("call the last element", arr[arr.length-1]());
```

24/07/21 class-4

NODE.js

List of all the commands:

node main.js tree "direct path"
node main.js organize "directoryPath"
node main.js help

(1) node main.js help → commnds list.

(2) node main.js tree "path" / (___)

- path
  - path/tree
  - current folder

(3) node main-js organize
- path → orgenize that path.
  - path
    - current paths

fs : has many functions, which has all documentation

// implementation → files / folder interact

let fs = require ("fs"); → to use fs module

let content = fs.readFileSync ("f1.txt"); → to read file

console.log ("content :" + content);

core modules :

FS, path, OS, util

- FS → file system
- path → paths → to make platform independence
- OS → operating → for ip address & all

documentation regarding files :-

let fs = require ("fs"); → for fs module

// for reading

fs.readFileSync ("f1.txt");

// for writing

fs.writeFileSync ("abc.txt", "eyyx is op");

// for appending

fs.appendFileSync ("abc.txt", "eyyx is too gud");

// for deleting

fs.unlinkSync ("abc.txt");

documentation regarding directory :-

```
// create
fs.mkdirSync("mydirectory");
// delete ⟶ fs.rmdirSync("mydirectory")
// path → check does it exist or not
// let doesExist = fs.existsSync("~~statSync("dir1")~~ fs1.js")
// console.log("this path exist?", doesExist);
// path → belongs to a directory or file
// let statsofpath = fs.lstatSync("dir1");
// console.log("is file?", statsofpath.isFile());
// console.log("is directory?", .isDirectory());
// directory → content.
// let address → abhi\\main.js
   let content = fs.readdirSync(address);
   console.log("directory content", content);
```

documentation regarding paths :-

Note ** to take input : overall :- → for removing node fs.j's

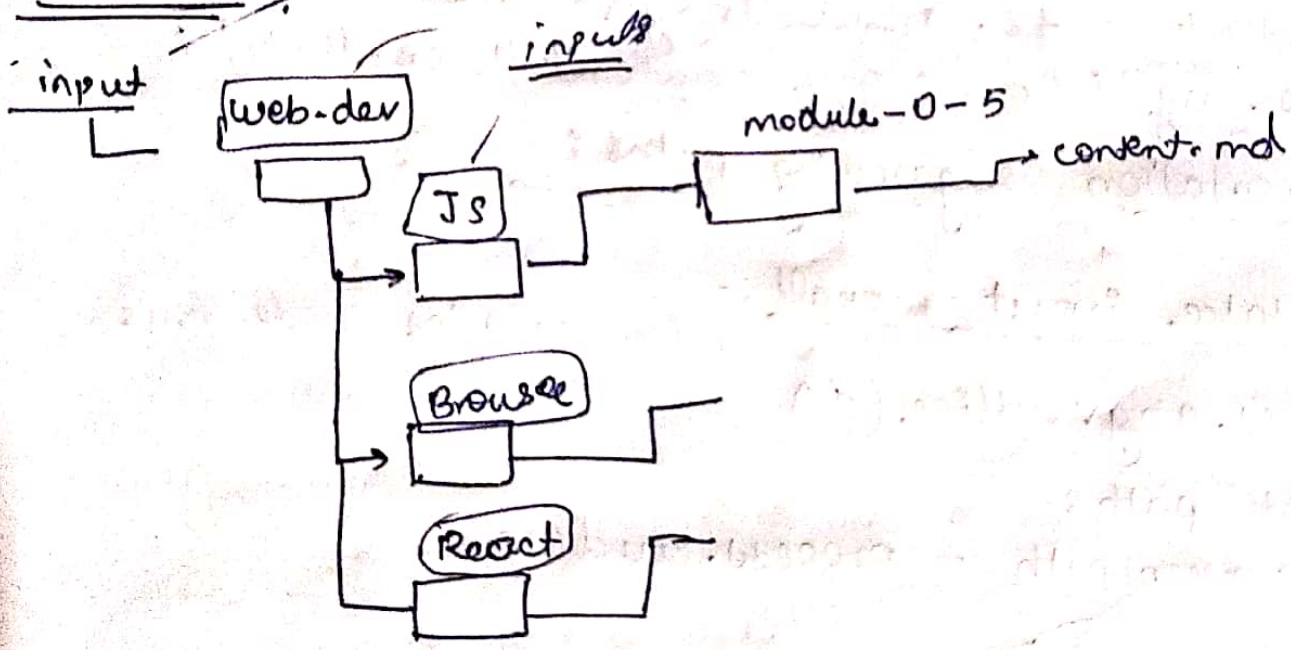
```
let arr = process.argv.slice(2)
```

① current path :

```
let currentpath = process.cwd();
console.log(arr);
let filename = arr[0];
let content = arr[1];
console.log("filename :", filename)
console.log("content", content)
// current path of directory
let currentpath = process.cwd();
```

```
// console.log ("currentPath", currentpath);
// path → paths → platform independent
// let joinedPath = path.join (currentpath, "abc", "def", "ghi");
console.log ("joinedPath", joinedpath);
let filePath = path.join (currentpath, "dir1", fileName);
console.log ("filePath", filePath);
fs.writeFileSync (filePath, content);
```

// Exercise 1 → process.cwd → if i need to move next just join.



>> node input.js webdev JS Browser React

```
let fs = require("fs");
let path = require("path");
let inputArr = process.argv.slice(2);
let mainDir = inputArr[0];
```

```
let cwd = process.cwd()
let mainDirPath = path.join(cwd, mainDir)
let isMainModulePresent = fs.existsSync
```

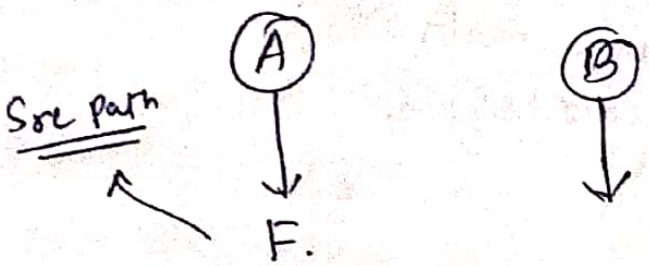
* test1

```
let fs = require("fs")
let inputdir = process.argv.slice(2)[0]
let allentities = fs.readdirSync(inputDir)
for(let i=0; i< allentities.length; i++)
{
    let entityName = allentities[i];
    let fullpath = path.join(inputDir, entityName)
    let statsofpath = fs.lstatSync(fullpath)
    if(statsofA path.isFile())
        { content += fs.readFileSync(fullPath);
        }
}
let filePath = path.join(inputDir, "summary.txt"
fs.writeFileSync(filepath, content);
console.log("summary path created.
```

→ extension name

```
let ext = path.extname
```

25/07/21.




```
let srcPath = "          "
let destDir = "          "
let tobeCopiedFileName = path.basename (srcFilePath, destDir)
console.log (tobeCopied ...)
let destPath = path.join (destDir, tobeCopi ..)
fs.copyFileSync (srcFilePath, destPath)
console.log ("file copied");
```

for code export" we needd to use in object form".

lib.js.

```
let a = 10;
function fn()
{ console.log ("Hello I am Fn"); }
function notToBeexported()
{ console.log ("I don't want to be exported"); }

// code exports
module.exports =
{
    varName : a,
    fxn : fn
}
```

client.js

```
let libexport = require("./lib");
console.log( ~~libex~~ "I am client file")
console.log ( libexport.variable)
console.log ( libexport.fxn());
```

Task:1.

Activity → main ke thru input.

main.js

Input → node main.js tree "path"
" " organize "path"
" " help "path"

```
let helpObj = require("./command/help");
```

code: check VS code. Lecture 4.

github :-

first time setup

git config --global user.email @ abhi.@.com
git config --global user.name " "
// to verify ① git configs --list

Always ① git init => create .gitignore => add node_modules to it
② git add . => ③ git commit -m "commit msg"

④ create a repo on github
⑤ git remote add origin Your repo name
⑥ git push -u origin master