

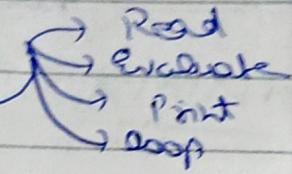
BACKEND - I

- Node.js & ^{JavaScript} Runtime environment is Node.js

→ It is used for server side programming.

→ Node.js is not a lang., library or framework.

- To check it in terminal `node -v`

- Node REPL & 

→ Terminal pe node lekha to ab ye node ki command ~~apni~~ terminal ki nhi (nhi) ^{or} work not work.

→ Ctrl + C (To exit from node ~~repl~~ ^{repl})

- Working in terminal.

→ js folder pe jana via `cd`

→ touch script.js, name of file.

→ `node script.js` [To run]

↳ (To work file)
/ (To access file)
↳ like of usual
command & or file
but we have to
execute it.

process

process object provide information about and control over the current node. is procen.

> procen . argv & returns array containing the command line arguments passed when the Node.js was launched.

for procen to go to node REPL then procen type t

procen . random, procen . uid(), & many more

• when no argument passed in procen . argv & node

[address → absolute path for node
address → current file path]

• when argument passed node ~~file name~~ ^{process.argv} is ~~hello bye~~ ^{is hello bye}

[
 "hello",
 "bye"
]

• for accessing in js code. (Returning .js)

let args = procen . argv;

for (let i = 2; i < args.length; i++) {

 console.log("hello to", args[i]);

}

Export in files

⇒ useful thing extracting from one file to another

⇒ module . exports → written in that file whose it wants to export
 ^{requiring files}

• require() is a built-in funⁿ to include ~~ext.~~ module that exist in separate file. ^{written in that in which we want the info}

• module . exports is a special object

• when we export & we use require then {} as by default empty object.

Ways

(i) module . exports . sum = (a, b) ⇒ a + b;

~~module~~ . exp

(ii) exports . sum = (a, b) ⇒ a + b;

Protest

Exports = 5;

11/2/2024

imports, exports = 6

~~Q. 1~~ : ~~Q. 1~~

Part 2: Part

3 : 3

3

2. 2. 2. 2. 2.

$$\frac{1}{2} \times \frac{1}{2} = \frac{1}{4}$$

Consolid. log(x.a(2,4))

$$1. (x, \max(2, 4))$$

↓ Export in Directories

- fronts & back

→ apple - 30 P, banana - 23

→ index is

max) extra require ("apple")

$$c+d = 11 \quad ("11/10000")$$

set $m = [r, d]$

1 module exports ~~map~~ = m

- outside fruits

→ ~~acem~~ fructus

Let $a_n = \log(n^{1/n})$

Low 2 - 2g (carbon)

Compt. by (Green 02)

- * NPM (Node Package manager)

npm is standard package manager for Node.js

(ii) Library of package →

make package manager

(ii) Common line load

- * installing packages

`npm install <- package name ->`

- node-modules & A folder contains every installed dependency for ur project

- package-lock.json & records the exact version of every installed dependency, including its sub-dependencies & their versions.

↓ package = 55 int

It contains descriptive & functional metadata about a project such as name, version & dependencies.

1. Module modules
2. code run error as the first module
3. To avoid this node modules has 1993.
4. In terminal in folder of file to npm install
5. Now code run properly

• Creating to package.json
 mkdir myproject
 npm init

// If we install another package in some folder
 only we change as dependencies like as before myproj
 // our package b/w add like as dependencies in
 in add more we

// Myr Kisi ko code dena hote to node modules ki
 oneste package.json bhagte hai.

* local v/s global
 ↓
 its folder in install this use.
 → we can install globally also.
 npm install -g <package name>
 npm link <package name>

} Tools
 npm, node

→ Require v/s import
 > we if require ya use import
 > math script.js
 export { sum, pi }
 import { sum, pi } from "math.js"

sum script.js
 import { sum, pi } from "math.js";

console.log(sum(1,2)); // 1+2=3

// To avoid error npm init then go create package
 via json ek new "type": "module" bnao.
 // Now work properly.
 node script.js

• Require

→ Old

→ No such in it
as all things are taken

→ Synchronous

Imports

→ New

→ we can selectively read only
the pieces we need.

→ asynchronous read via module Polge
use no sta so Polge
import.