

Node REPL

→ यह cmd 'node' type करेंगे तो यह node का environment और enter कर जाएंगे अब terminal command run करी देंगे।

→ ~~प्र०~~ यहाँ पे JS का input, output का नोटिकल लिंग का उदाहरण है।

"alle" + "Anwind"

'alle Anwind'

→ इसी exit करने के लिए exit Command दूंगा।

Node file

→ यह VS Code में बना JS file को cmd में execute करना
होते हैं इसमें use होता है `node fileName.js`

Process in node

process

→ यह एक object है जो कि current Node.js को control करने से help करता है

Process Version

→ Current में जो Node.js use हो रहा और process का version show करता है

Process.cwd(); → Current directory location Path show करता है

Process.argv → यह एक array को return करता है



Terminal (cmd)

node

node app.js

node app.js Hello. allu here

'/usr/local/bin/node'
'current path'
'Hello'
'allu'

VS code (app.js)

let arg = process.argv;

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

{

console.log("Hii ", arg[i]);

}

node app.js

Hello, any thing

Export file

(`require()`)

→ JS file से दूसरी file में data को transfer and use

(`module.exports`)

→ data को export करा है in the form of object

दूसरे file में वही function, value, properties
को करके use करे जो help करते हैं।

Step ①

Math.js

~~Sum~~

Const Sum = (a, b) => a + b;

Const Mul = (a, b) => a * b;

// Paste here

module.exports = "obj";

let obj = {

Sum: sum,

Mul: mul,

};

module.exports.div = (a, b) => a / b;

temp

app.js

Math.js

Step ②

app.js

Const SomeV = require("./math")
~~Const SomeV =~~.

Console.log(SomeV);

Console.log(SomeV.sum(5, 7));
~~Module.exports = SomeV;~~

Step ③

Terminal (cmd)

Node

set path before

node app.js

Export in Directory / Folders

→ इसमें एक folder ने multiple file को require करता है।
उसी folder location से।

Step ①

apple.js

module.export = {

name : "Apple",

color : "red",

}

Step ②

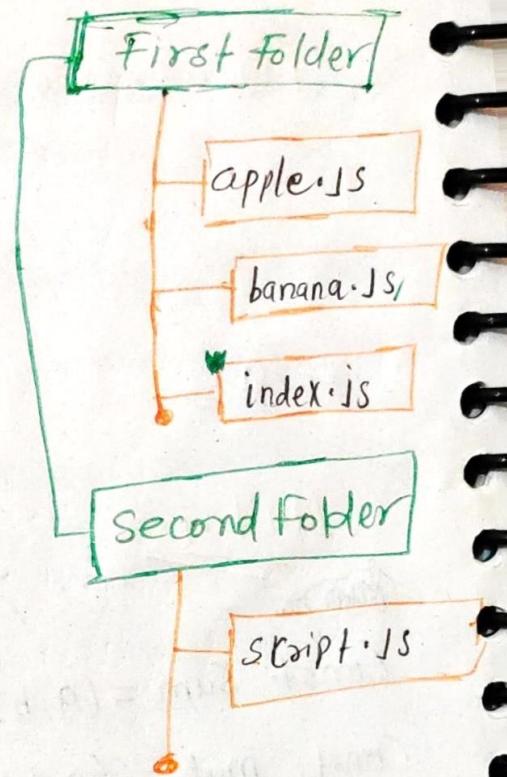
banana.js

module.export = {

name : "banana",

color : "yellow",

}



Step ③

index.js (special file)

const apple = require("./apple");

const banana = require("./banana");

let fruits = [apple, banana];

module.exports = fruits;

Step ④

script.js

const info = require("./first");

console.log(info);

गले index.js का नहीं special file का है।

script.js में यह folder access करना भी नहीं है।

उसके सिर्फ उस file को access करना है।

यों कि index.js नाम से save होता

जाता है।

NPM

<https://www.npmjs.com>

Node Package Manager

- NPM is the standard package manager for node.js
- अब आप ~~कोड~~ कोड बहिर्भास Algorithm use करके लहूत ही कुछ developer किसी और उसको package में लाने के लिए NPM use करते हैं जाकि वह लोग भी use कर सके।
- react.js जो कि एक code है और उसको लहूत वाले developer use कर सकते हैं उसी तरह आप उसको कर सकते हैं using NPM ऐसे code को git में फ़िक करने का
- library of package
- Command line tool
- npm पर library है जो कि लहूत से developer use कर सकते हैं to access the package name यो ~~कोड~~ Algorithm work Perform करता है।
- simple गाइड में कुसरे ~~बड़े~~ लिखा code के Access कर सकते हैं in the form of package

npm

→ To check in terminal

npm install packageName

→ for install package

Installing Package

node_module → इस folder के अंदर खिलता है project के लिए necessary file होते हैं जो यहाँ install होते हैं।

Ex - suppose कोई Project में किसी API use किया गया है तो उस API के लिए code work करने के लिए उसी API dependency हो सकती है for your project!

package-lock.json → It record the exact version of every installed dependency, including its sub dependencies and their version.

package.json → इसी project का package.json ऐसा होता है कि इसका aim delete किए गए node-module folder bhi backup कर सकता है या using **(npm install)** folder path in terminal.

प्रत्येक package होता है जो इस personal package का हिस्सा है।

जो आप GitHub पर package.json push कर देते हैं वह उसी जानकारी के साथ file का download करता है और using **(npm install)** to run in terminal.

Make Package

→ यह मापको अपने directory में package को बनाना हो तो करें
cmd terminal में ↴

(npm init)

(npm init -y)

then fill your option:

↓
then binaly package ready

↳ default
package
automatically

→ यह मापके धास already कोड पहले से package.json file बना रखी
है या download कर रखी है। और उसी same folder में
माप एक नया package install करते हो तब उस माप की
तीन की folder में कोड add हो जाए with new
dependency.

- 1) node_modules
- 2) package-lock.json
- 3) package.json

→ यह की माप कोई package को install करे तो पर
वह लिखा भाइ रखे क्षेत्र local की install करे मानु global
install करे।

PTO
→

Global package install

(`npm install -g packageName`)

(`npm link packageName`)

→ यदि कोई package को आप globally install करते हो तो
उसमें वो वो कारोंहूँ उसे -g लिए globally install कर
पिछे उसको link करें।

→ लोकल locally package install better होता है

Import file vs require

- import की जांच करना है जिसके लिए require की तरह जिसमें भी पुरी files विशेष folder से कोई particular function को आवानी से import कर लेता है जो किंवद्दन require पुरी file को access करता है।
- जिसी जटि function, Variable, Output को use करना है तो simply उसके उपराने export किया देता है।
- import अन्य use के compare to require को कम memory consume करता है।

import {sum} from "./filename"

→ Location file

→ यहाँ तके function को Name, VariableName, (उपराने)
export करना हो तो
Name

- individual चियों को Access कर सकते हैं। जिनका जा Access करना है उनके साथ export लगाओ। और import code लिख दो।

PRO

Note

यहाँ यही जाप्य import use करेंगो। तब जाप्को घरे folder में package.json बनाना हो जाए तो उसके पास से "type": "module" add करना होगा। error से

→ यहाँ कैसे import को use करते हैं ये process हैं।

Step ①

प्रिंटिंग Access करना है ताकि साथे export किया।

Step ②

प्रिंटिंग file में Access करना है वहाँ **or import** की code लियो।

Step ③

Package.json file भी होना है जो "Type": "module" add करो। यह JSON file नहीं है create करो।
using of **(npm init)**

Step ④

replace the **.js** into **.mjs**

like -

script.js → script.mjs

Math.mjs

export const sum = (a, b) =>
 a + b;

export const mul = (a, b) =>
 a * b;

script.mjs

import { sum, mul } from ".(Math.mjs);
console.log (sum(2,34));

package.json

{
 "name": "script",
 "version": "1.0.0",
 "description": "A simple script",
 "main": "script.mjs",
 "scripts": {
 "start": "node script.mjs"
 },
 "dependencies": {}
}

"type": "module"

3

Node script.mjs

Run Terminal