# NODE JS

## Node.js :-

- Javascript Runtime Environment
- It is used for server side programming.

NOTE:- Node.js is not a language, library or framework.

## Node REPL :-

REPL → Read
Evaluate
Print
Loop

} → Write node in terminal

⇓

Now we can run JS commands in terminal

(But we can't access DOM elements like window object because they are created by browser).

→ .help gives us commands. } → print the help msg.

## Node Files :-

node fileName.js } → For running file through terminal.

NOTE:- Make sure to be present in the directory which contains fileName.js.

## Process in Node :-

- process : This object provides information about, and control over, the current Node.js process.

Syntax: process } → inside Node REPL

Eg: process.version
process.release
process.cwd()

- process.argv : returns an array containing the command line arguments passed when the Node.js process was launched.

→ Syntax/Example:-

Inside script.js

```
let args = process.argv;
for(let i=2; i<args.length; i++)
{
  console.log("Hello", args[i]);
}
```

Here i=2 because at i=0 & i=1, Executable path for node & path of file which we're running are stored.

Now in Node REPL:-

node script.js Aayush Tarun Krishna Nikhil Kartik

→ Arguments other than 2 inbuit arguments.

## Export in Files:-

→ module.exports:-
requiring files

- require() : a built-in function to include

external modules that exist in separate files.

- module.exports : a special object

Ex: →

→ Both files must be in same directory.

**script.js**

```
const math = require("./math");
console.log(math.sum(2,2));
console.log(math.PI);
```

**math.js**

```
module.exports.sum = (a,b) => a+b;
module.exports.g = 9.8;
module.exports.PI = 3.14;
```

→ module.exports :→
requiring directories

Eg:→ File structure

Backend
- Fruits
  - apple.js
  - banana.js
  - orange.js
  - index.js
- script.js

**apple.js**

```
module.exports = {
    name: "apple",
    color: "red",
};
```

**banana.js**

```
module.exports = {
    name: "banana",
    color: "yellow",
};
```

**orange.js**

```
module.exports = {
    name: "orange",
    color: "orange",
};
```

Name must be index.js while exporting b/w directories

index.js

```
const apple = require("./apple");
const banana = require("./banana");
const orange = require("./orange");

let fruits = [apple, banana, orange];
module.exports = fruits;
```

script.js

```
const into = require("./Fruits");
console.log(into);
```

# NPM (Node Package Manager):-

npm is the standard package manager for Node.js.

1. library of packages
2. command line tool

[Not a library exactly]

Command: npm

# Installing Packages:-

npm install <-package name->

Eg: npm install figlet

figlet → Dash-Designs package

- node_modules : The node_modules folder contains every installed dependency for your project.

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

- package.json : The package.json file contains descriptive and functional metadata about a project, such as a name, version and dependencies.

## Local v/s Global :-

npm install -g <-package name-> } → To install package globally
npm link <-package name->

## require v/s import :-

import {sum} from "./math.js"

We can't selectively load only the pieces we need with require but with import, we can selectively load only the pieces we need, which can save memory.

Loading is synchronous for 'require' but can be asynchronous for 'import'.

Eg:

math.js

```
export const sum = (a,b) => a+b;
export const g = 9.8;
```

script.js

```
import {sum} from "./math.js";
console.log(sum(1,2));
```

NOTE :- Make sure to create package.json in the directory & add "type":"module"