DAY 29

* What is Node.js

Javascript Runtime Enviroment
It is used to Server side programming

* Node.js is not a language, library or Framework.

Node.js give power to javascript to Run outside the browser

e.g 1) Game 2) IoT 3) server-side

* Node REPL :

REPL : Read evaulate print loop

.help : give us Commands

* node give global object not window :. Window object apply only in browser.

* .help give :

.break : Sometimes you get stuck, this gets you out.
.Clear : Alias for .break
.editor : Enter editor mode
.exit : Exit the repel or ctr+C (Twice)
.~~hep~~ help : print this help message
.load : Load js form a File into the REPL session
.Save : Save all evaluated command in this REPL session to a file.

Page No.
Date

* How to Create Node Files & execute

1] Creation :

Filename.js

2] execution :

node filename.js

* Process :

This object provides information about & control over, the Current Node.js process.

cmd :

Process

Process.cwd()

Process.release

Property :

Process.argv : return an array containing the command-line argument passed when the Node.js process was launched.

give two value :

e.g : ['/usr/local/bin/node' , // executable path

'Users/sample/webclass/Backend/script.js'] // current path

* Export in Files :-

module.exports

requiring files

require() a built-in Function to include external module that exist in seperate files.

module.exports a special object

Math.js

```
const sum = (a,b) => a+b;
const mul = (a,b) => a*b;
```

* module.exports = {

```
    sum : sum;
    mul : mul;
};
```

Script.js

* ```
let res = require("./Math");
console.log(res.sum(2,3));
console.log(res.mul(2,3));
```

* For directory

1) require → index.js (entry-point)
2) export

Directory Hierarchy :

```
v Node JS
  v Fruits
      apple.js
      orange.js
      index.js
  Scripth.js
```
```

apple.js

```
Const apple = {
                name : "apple",
                color : "red"
                };

module.exports = apple;
```

Orangle.js

```
Const orange = {
                name : "orange",
                color : "orange",
              };

~~module.exports~~ module.exports = orange;
```

index.js

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

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

script.js

```
let res = require("./Fruits");
Console.log(res)
```

o/p : [{name : "apple", color : "red"},
{name : "orange", color : "orange"}]

Page No.
Date

* NPM (node Package manager)

npm is the standard package manager for node.js

(store) 1) library of package → code already written

(Access other) 2) Command line tool

* Difference bet'n NPM v/s NPX

	NPM	NPX
1)	NPM is a package manager used to install, delete, & update javascript package on your machine.	& NPX, is a package executer, & it is used to execute javascript package directly, without installing them
2)	NPM install package globally, which mean that your machine may be polluted by package that are not required anymore in the long run.	2) NPX does not install package, so package pollution on the machine is not a concern.
3)	To use create-react-app using NPM, we would first have to install it globally, & then run it, which make using NPM in both case redudant	3) The most common application of NPX is the create-react-app command. since we only need to use it once, such

Page No.
Date

* How to install Package:

npm install <- Package Name ->

e.g npm install Figlet

* After installing package : there are ~~1 dirctory & 2~~ 1 directory & 2 file are created

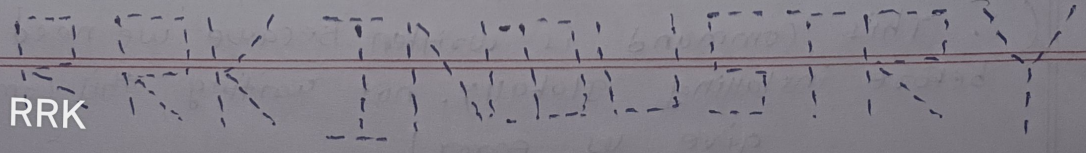
1) node_modules : The node_module folder contains every installed dependency for your project.

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

index.js

```
Var figlet = require("Figlet");

Var ("RRK INDUSTRY", Function (err, data){
    if (err)
    {
        Console.log ("Something went wrong...");
        Console.dir(err);
        return;
    }
    Console.log(data);
});
```



3) Package.json

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

~~npm init~~

***

suppose you delete node_module from your project file then error occur that error resolve using

- npm install

***

* 'npm init' is used in node.js project to create a 'Package.json' file, which is a metadata file for a node.js project.

* Local V/s Global

Use only within Project

use anywhere

② npm install -g <- Package Name ->

③ npm sink <- Package name ->

Befor install package ~~b~~ globally use command

① sudo chown -R $USER/usr/local/lib/node_modules

(∴ This command is written because we need admin access before installing globally. not writing this command will give us error)

Page No.
Date

* requrie V/s import :

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

we can't selectively load only the piece we need with requirе but with import, we can selectively load only the piece we need, which can save memory.

loading is synchronous for 'requirе' but can be asynchronous for 'import'.

Math.js

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

Script.js

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

Console.log(sum(1,2));

O/P : 3