Thapa technical Video 1

Node.js

Thapa Technical

Q) What is Node.js ?

→ ① Nodejs is a Javascript runtime built on chrome V8 Javascript engine

→ ② Nodejs is a Javascript running on the server.

↘ Javascript runtime environment on the spot if we change Anything in code it give direct output (during runtime).

Note:- Initially, we run our Javascript code inside the browser only But now with the help of Nodejs we can run our code on Both browser and server.

Javascript ⟹ client side scripting language.

Nodejs ⟹ server side scripting language.

built on chrome V8 Javascript engine

V8 engine (chrome)

In 2009 an idea came in the mind Ryan Dahl (goggle engineer) that why Not run javascript outside the browser, so he took V8 engine and embedded in a C++ program called Node.exe later Node.js.

V8 Engine simply converts our code into machine understandable code which is binary Number (0 or 1)

So, with the help of V8 engine we can run our javascript code outside the browser (server)

- V8 Engine → Chrome
- SpiderMonkey → Mozila fire fox
- Nitro → Safari

↳ for different browser different engine.

Node.js is not a framework.
Node.js is not a library.

Node.js is simply a javascript runtime built on chromies V8 Javascript engine.

When to use Node Js?

(1) I/O bound.

(2) Data streaming Applications.

(3) Real time chat Applications.

Note:- Node.js is NON-blocking (fast and we can request multiple times)

Ex:- Netflix means if we scroll infinity it never ends showing or giving content suggestion.
or
Instagram.

Q) framework ? Why we use framework?
framework is a Readymade tool

Q) why we use Nodejs as Backend language?

(1) JS fullstack ⟹ we can use single javascript for Both front-end as well as Backend development.

(2) Scale ⟹ Easily Scalable building Application.

(3) NoN-blocking ⟹ fast and NoN-blocking which means it follows Asynchronous Behaviour.

(4) Ecosystem ⟹ It has been huge ecosystem and communities.

Basic command of Node.js

```
mkdir   my-node-app
cd      my-node-app
npm     init -y
```

Explain the code?

(1) `mkdir my-node-app` } creates new directory in the current location.

- `mkdir` → make directory
- `my-node-app` → name of new directory.

(2) `cd my-node-app` } This line changes the current working directory to my-node-app that you just created.

- `cd` → change directory

(3) `npm init -y`

- `npm` → node package Manager
- `init -y` → automatically accepts all default settings
- This line initialises a new Node.js project inside the "my-node-app") directory using NPM (Node package Manager).

⇒ This command creates [Package.json] file in your project directory, which contains metadata information about project's

Node.js

Video-2 Thapa Technical.

Note:- `console.log()` is not javascript code. Our V8 Engine supports this's code that's why we are using it.

*. Running Our first Node.js script.

Step-1 ⟹ open folder in VS code which contains HTML, CSS, Javascript code. (move or open Js code)

Step-2 ⟹ Open Terminal ⟹ New Terminal.

Step-3 ⟹ Inside new terminal you can see your folder path.

Step-4 ⟹ Inside new terminal write, node Enter -V + (checks Node is downloaded or not)

Step-5 ⟹ dir + Enter (This shows you All file inside folder)

Note:- If your file name is Not showing then you have to check from previous steps because you are not able to proceed for further steps.

Step-6 ⟹ If showing your file name (ex!- script.js), then proceed.

Step-7 ⟹ ~~node~~ node script.js + Enter (this will show you the output)

Completed

Nodejs Node Js pre-requisites

video-3 Thapatechnical

Node js prerequisites

(1) Basic Javascript (Advanced JS will be Bonus)

(2) ES5 & ES6 | ECMAScript 6

(3) client Server Model (optional)

* We Must focus on ECMAScript 6 *.

⇒ ECMAScript 6, also known as ES6 or ECMAScript 2015.

⇒ ES6 introduced host of new features and syntactic improvements aimed at making javascript development more efficient, maintainable and powerful.

Key features of ES6

1. Arrow functions

```
const add = (a, b) => a+b;
```

Consise syntax
Automatically bind 'this's surrondig code.

2. let and Const Declarations.

```
let x = 10; // let => Allows block-scoped variable declarations.
Const y = 20; // const => Defines block-scoped constants.
```

③ Template Literals.

Allow embedding expressions within string literals using Backticks and `${}` syntax.

```
const name = 'John';
const greeting = `Hello, ${name}!`;
```

④ Destructuring Assignment.

Enables unpacking values from arrays or properties from objects into distinct variables.

```
const [a,b] = [1, 2]
const {name, age} = {name: 'Alice', age: 30};
```

⑤ Default parameters.

Allow function parameters to have default values if no argument is provided.

```
function greet(name = 'Guest'){
    return `Hello, ${name}`;
}
```

⑥ Rest and spread operator (Represented using three dots (...))

① Rest operator

Gathers multiple elements into a single array or object.

Ex:-

```
Const person = {
    name: 'Alice'
    age: 25
    city: 'New York'
    country: 'USA'
};

Const {name, ...details} = person

console.log(name) // output: Alice
console.log(details) // output {age: 25, city: 'New York', country: 'USA'}
```

② Spread operator

- Spread elements from an array or object into another array, object function arguments.

Example:- 1 (Merge)

```
const arr1 = [1, 2, 3]
const arr2 = [4, 5, 6]
const combinedArr = [...arr1, ...arr2]
console.log(combinedArr)
```

output

[1, 2, 3, 4, 5, 6]

Example 2 (copy an array)

```
const arr1 = [1, 2, 3];
const arr2 = [...arr1];
console.log(arr2)
```

output

[1, 2, 3]

Example 3 (spread elements of an Array into a function call)

```
Const arr = [1, 2, 3]
Console.log(Math.max(...arr));
```

output

3

Note: Spread Also works for Math.min.

7. Enhanced object literals.
8. classes
9. Modules
10. promises.
11. Iterators and Generators.

Learn it yourself

Node.js

video 4

REPL

Thapa Technical

⇒ REPL (Read, eval, print, loop) is a computer environment. Similar to Shell (Unix/Linux) and Command prompt.

Note: Just like command prompt present in windows. Same as REPL present in nodejs.

⇒ Nodejs comes with REPL environment when it is installed.

⇒ Nodejs REPL is an interactive Shell that processes Node.js expressions. The REPL provides a quick and easy way to test Javascript code, debug and perform computations in a live environment.

⇒ Repl used to experimenting Nodejs expressions.

REPL stands for

① Read ⇒ Reads the input provided by the user. Parse input into Javascript data structure and Stores in memory

② Eval ⇒ Evaluates the input string as Javascript code.

③ print ⇒ print the result of the evaluation.

④ loop ⇒ Repeats the process, allowing for

→ loops the above command until the user presses ctrl-C twice.

Steps to see REPL in Node JS.

Step-1 create a folder name Nodejs for youtube

Step-2 open new terminal

Step-3 write inside terminal ⇒ node } Start the Repl

Step-4 ⇒ .help

Step-5 ⇒ .exit } exit from the Repl.

Step-6 ⇒ cls } clear All above written codes.

Step-7 PS E:\nodejs for youtube > type nul > index.js
(nul → empty file; index.js → file name)
→ this creates empty index.js file, inside Nodejs for youtube folder

Step-8 > dir + Enter

index.js } file name getting means everything good.

// REPL

// 1: js Expression.
// 2: use variables.
// 3: Multiline code. → underscore
// 4: use (_) to get the last result.
// 5: we can use editor mode.

1. Js Expression (Evaluating Expressions)

Step-9 > node + Enter } Start the Repl

Step-10 > 3+3 enter
6

2. use variables (variable Declaration)
. Another example for Step-10

Step-9 > node + Enter } Start the Repl

Step-10 > var a = 'thapa'
> a
'thapa'
> var b = 'technical'
> a+b
thapatechnical

3. Multiline code (Multiline Expression) {}

Step-9 node + Enter } Start the Repl

Step-10
```
var x = 0;
> x
0
> do
2
x++;
console.log(`my x value ${x}`);
} while (x<5);
```

output:
my x value is 1
" " " 2
" " " 3
" " " 4
" " " 5

4. Use (_) to get the last result

Step-9 node } start the Repl.

Step-10 > 10 + 20
30
> _ + 50
80

this represents the last result

5. We can Use an editor Mode

Step-9 > node

Step-10 > .editor

```
const name = (myname) => {
    Console.log(`my name is ${myname}`);
name(`thapa Technical`);
```

Output-

my name i's thapa technical.

* Special commands in Repl prefixed with dot (.)

1. .help ⟹ Display list of special commands
2. .break / .clear ⟹ Exit the current multiline expression.
3. .exit ⟹ Exit the REPL session.
4. .load filename (.load myscript.js)
   ⟹ loads a Javascript file into the current REPL session.
5. .save filename (.save session.js)
   ⟹ saves the current REPL session's history to a file.

Node.js video 5

Mastering the Node.js core Modules

thapatechnical

* **Modules** ⟹ Modules can be single file or a collection of multiple files/ folder. They are similar to Javascript libraries

⇒ Module is a set of functions you want to include in your Application.

⇒ Node.js has a set of built-in modules which you can use without any further installation

you can explore more about Modules online

⇒ Nodejs official website

⇓

⇒ docs option

⇓

⇒ file system

⇓

⇒ see

Steps

Step-1 open nodejsforyoutube folder inside vs code.

Step-2 create a file index.js and write

```
const name = "thapa";
console.log(name);
```

Step-3 Inside terminal write,

> node index.js + Enter

index.js → filename

You get- "thapa" as output.

This is normal procedure to see output- Now we further proceed to include Module in our code.

Step-4 clear or Erase all code written inside index.js, Now write

require("fs") → module

It will include module in our code.

Step-5 write code inside index.js.

this include module in our code

module name

```
const fs = require("fs");
fs.writeFileSync("read.txt", "welcome to my channel");
```

create a new file

synchronous file.

filename

content written inside file.txt.

It will create new file named as read.txt inside nodejsforyoutube and written text- welcome to my channel.

Step-6 Again clear and write code inside index.js (override the existing written content step5+step6)

```
const fs = require("fs");
fs.writeFileSync("read.text", "thapa technical welcome to my channel")
```

synchronous file.

This will not create new file. The file read.text already created in step-5 this will just override the content or text written inside read.text.

Note:-
Synchronous means in RealLife
→ if two customer went to a restaurant then chef takes order for food from 1's customer and after that he cooked the food. After completing the order for 1st customer. Chef takes order from 2nd customer. This is Synchronous Behavior. (one after one)

Asynchronous means
chef takes order from 1st customer while the preparing the food for 1st customer. Chef takes the order for 2nd customer. (one is executing parallely with other)

Sync means Synchrous

Step-7 Again write (Add content or text to existing file)

```
const fs = require("fs");
fs.writeFileSync("read.txt", "thapa technical, welcome to thapa");
fs.appendFileSync("read.txt", " now are you I am fine ");
```

this will add content to the existing file read.txt

Step-8 Again write. (Read the file)

```
Step-7 All content
const buf_data = fs.readFileSync("read.txt")
console.log(buf_data);
```

this will read the text file

Note:
- Node.js includes an additional data type called Buffer. (not available in browser's Javascript)
- Buffer is mainly used to store Binary data while reading from a file or receiving packets.

Step-9 Again write,

```
Step- 7. All code
const buf_data = fs.readfileSync ("read.txt");
org_data = buf_data. toString();
console. log (org_data);
```

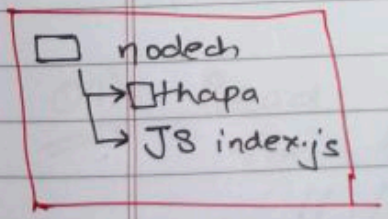
Node.js Thapa Technical

video6 CRUD operation using FS Module.

Challenge Time

1. Create a folder named it Thapa?
2. Create a file in it named bio.txt and data into it?
3. Add more data into the file at the end of the existing data?
4. Read the data without getting the buffer data at first? // file encoding.?
5. Rename the file name to mybio.txt?
6. now delete both the file and the folder?

Q1) create a folder named it thapa ?



step-1 open folder nodech

step-2 create index.js

write code

```
const fs = require("fs")
fs.mkdirSync("thapa");
```

step-3 node index.js

step-4 cd .. (move out of index.js)

step-5 dir

step-6 cd .\nodeCh1 } go inside nodech (select nodech)

step-7 node index.js → this will create file named thapa. above executed code

→ inside terminal

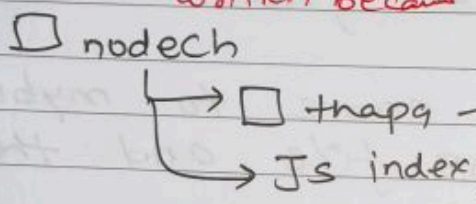
Q2) create a file in it named ~~bio.txt~~ bio.txt and data into it ?

Ans

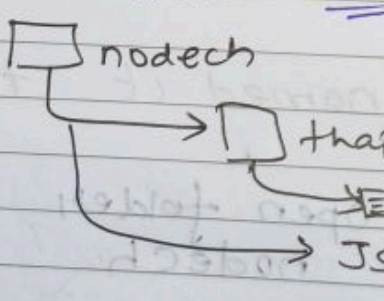
```
const fs = require ("fs");
// fs.mkdirsync ("thapa")

fs.writefilesync ("thapa/bio.txt",
                  "my name is vinod thapa");
```

written becaus



~~Outp~~ Results or output



Inside terminal write
node index.js + Enter

Q3) Add more data into the file at the end of the existing data?

Ans

index.js

```
const fs = require("fs");

// fs.mkdirSync("thapa");
// fs.writeFileSync("thapa/bio.txt",
        "my name is vinod bahadur thapa");

fs.appendFileSync("thapa/bio.txt",
        "plz subscribe to my channel");
```

Inside terminal, write:
node index.js + Enter.

Output
plz subscribe to my channel → added to existing content.

Q4) Read the data without getting the buffer data at first?
// file ~~enc~~ encoding?

Ans

getting Buffer data
6d 79 20 61 6d ....

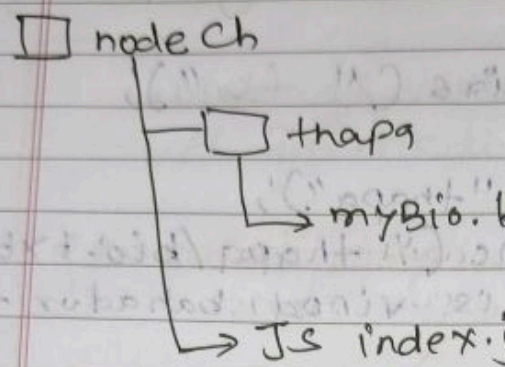
Same Answer of Q3

```
const data = fs.readFileSync("thapa/bio.txt");
console.log(data);
```

getting string value means without getting Buffer data.

```
const data = fs.readFileSync("thapa/bio.txt",
                             "utf8");
console.log(data);
```

Q5 Remove myBio.txt ?

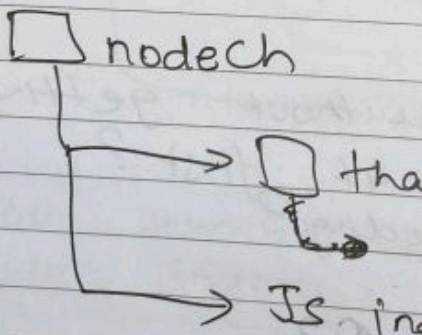


Ans

```
const fs = require("fs")
fs.unlinkSync("thapa/myBio.txt");
```

→ this will remove myBio.txt file.

Q6 Remove thapa folder ?

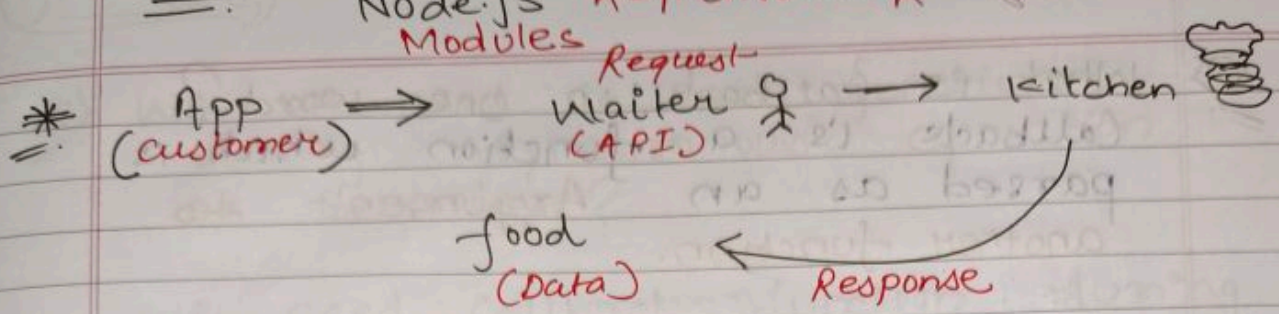


Ans

```
const fs = require("fs");
fs.rmdirSync("thapa");
```

→ This will remove thapa folder

Video 7 thapa Technical File System, Node.js Asynchronous, core Modules



Q) What you can observe while requiring filesystem between Synchronous and Asynchronous file?

⇒ In Synchronous we don't have callback option within function while. In Asynchronous callback option within bracket is present.

Synchronous
Ex:- fs.writefileSync ("read.txt", "today i's welcome").

Asynchronous
Ex:- fs.writefile ("read.txt", "today welcome)", (err) ⇒ {
console.log ("files is created");
});

Q) What is Callback in one word?

Callback is a function which is parsed as an Argument to another function.

Note Remember whenever you worked like Asynchronous behaviour then you have to specifying Callback functions.

* Node JS Asynchronous File System

```
const fs = require("fs");

fs.writeFile("read.txt", "today is awesome day :)",
    (err) => {
    console.log("files is created");
});
```

Inside terminal write,
node index.js + Enter

Output

read.txt file has been created inside this file today is awesome day is written.
Inside terminal written files is created.

Q) Why we need callback function during Asynchronous operation or Behaviour?

we need callback function during Asynchronous operation because let see example

```
fs.writefile ("read.txt", "today is
                awesome day :)";
    (err) => {
    console.log ("files is created");
    console.log (err);
});
```

Once
fs.writefile ("read.txt", "today is awesome day :)"
this completes then what.

⇒ Callback has an argument that tells you whether the operation completed successfully.

// checking for error (if no error return null value)

Q) How to Add data to existing created file in Asynchronous file System?

```
const fs = require("fs");
/* fs.writefile("read.txt", "Hello world",
// (err) => {
// console.log("file is created");
//     console.log(err);
//     });

fs.appendFile("read.txt", plz like and share subscribe my channel",
, (err) => {
  console.log("task Completed");
});
```

output

This will add content text.
plz like and share and subscribe my channel to the existing file (read.txt) and at the terminal gives Task completed.

02) How to Read data of existing created file in Asynchronous file System?

```
const fs = require("fs");
```

[Same as 1 comment → write file

[Same as 1 comment → append file.

```
fs.writeFile("read.txt", (err, data)
    => {
        console.log(data);
    });
```

output

```
node index.js
<Buffer 74 6f 64 61 79
. . . . . 10 more bytes>
```

But we Don't want Buffer data, for that we need to Add.

UTF-8

```
fs.writeFile("read.txt", "UTF8",
        (err, data)
    => {
        console.log(data);
    });
```

output-

today is awesome plz Like share and Subscribe channel.

Remember ⇒ We Don't want Buffer Data then we Add.

UTF-8 to our code.

Synchronous Vs Asynchronous programming in Node JS?

[Node JS → 8] video-8

Q1) why we should prefer Asynchronous programming over Synchronous programming?

In Synchronous programming,

```
const fs = require ("fs");
const data = fs.readFileSync
    ("read.txt", "utf-8");
console.log (data)
console.log ("after the data ");
```

output

```
node index.js + Enter
today is awesome :) plz subscribe
after the data
```

firstly above program runs the Next-program runs.

In Asynchronous pragramming

```
const fs = require ("fs");
fs.readFile ("read.txt", "utf-8",
    (err, data) => {
        console.log (data);
});
console.log ("after the data)
```

output

```
node index.js + Enter
after the data
today is awesome ; plz
Subscribe
```

code line which takes less time this will execute firstly.

video 9 Asynchronous CRUD Operation

Thapa Technical using File System Modules

CHALLENGE Time

1. create a folder named it thapa?
2. create a file in it named bio.txt and data into it.
3. Add more data into the file at the end of the existing data?
4. Read the data without getting the Buffer data at first.
   file encoding.
5. Rename the file name to mybio.txt
6. now delete both the file, and the folder?

* * * * *

① create a folder named it thapa?

```
const fs = require("fs");
fs.mkdir('thapa', (err) => {
    console.log("folder created");
});
```

Inside Terminal write. node index.js

output

thapa file created
and folder created text written in terminal.

② create a file in it named bio.txt and data into it.

```
const fs = require("fs");
```

Same as 1 commented.

```
fs.writeFile("./thapa/bio.txt", "my name is vinod thapa",
    (err) => {
        console.log("file created");
    });
```

Inside Terminal write " node index.js

output

bio.txt file created inside file thapa and
my name is vinod thapa written inside bio.txt

③ Add more data into the file at the end of the existing data?

```
const fs = require ("fs");
    same as 2 commented
fs.appendFile ("./thapa/bio.txt", 
    plz lik and share my video "3",
  (err) => {

    console.log ("file ~~created~~ data appended");
});
```

output Inside Terminal write node index.js
my name is vinod thapa plz lik and share my video text added to text file. bio.txt

④ Read the data without getting the Buffer data at first?

```
Const fs = require ("fs");
    Same as 3 commented
fs.readFile ("./thapa/bio.txt", 'Utf-8',
    (err, data) => {
      console.log (data)
});
```

output Inside terminal write node index.js
you get string value (No Buffer value) with the help of (UTf-8)

Note if we write bios.txt which is not exists instead of file bio.txt.

```
const fs = require ("fs");
        same as 3 commented
fs.readfile ("./thapa/bios.txt", 'utf-8',
        console.log (err)
    });
```

see it clearly

this code having bios.txt which is undefined but we had given console.log (err)

this will generate an error

Error: no such File or directory Exist.

(5) Rename the file name to mybio.txt?

```
Const fs = requir ("fs");
        Same as 4 commeted
fs.rename (". /thapa/bio.txt", "./thapa/myBio.txt
    , (err) => {
        console.log ("rename done")
    });
```

Output-

rename done.

Bio.txt file name changes to myBio.txt.

06) now delete both the file and the folder ?

first Delete the file.

```
const fs = require ("fs");
    same as 5 commented

fs.unlink ("./thapa/myBio.txt", (err) =>
    {
        console.log ("file deleted");
    });
```

Output

file deleted
myBio.txt get deleted.

Second Delete the folder.

```
const fs = require ("fs");
    saved as 5 commented

fs.rmdir ("./thapa", (err) => {
        console.log ("folder deleted");
    });
```

Output

folder deleted
thapa folder deleted.

Node JS Module to Get Operating System Info

video - 10

Thapa technical

for getting info of our Operating system we need to require ("os")

(1)

```
const os = require("os");

console.log(os.arch());
console.log(os.hostname());
console.log(os.platform());
console.log(os.tmpdir());
```

→ gives Os Architecture (x64)
→ gives os hostname (DESKTOP-4BH1206)
→ gives os platform (win32)
→ gives os template directory (c:\users\shri\App

(2)

shows free memory

```
const os = require("os")

const freeMemory = os.freemem();
Console.log(`${freeMemory}/1024/1024/1024}`);
```

output
→ 11.3939...

helps to convert Bytes into Gigabyte.

(3)

shows total memory.

```
const os = require("");

const totalMemory = os.totalmem();
Console.log(`${totalMemory/1024/1024/1024}`);
```

output
15.5...

converts Bytes to GB.

NodeJs Path Module video 11

Path Module in NodeJs

thapatechnical

* Path *

The path module provides utilities for working with file and directory paths. It can be accessed using

```
const path = require('path');
```

(1)

```
const path = require("path");

console.log(path.dirname("E:/nodejsyoutube/pathModule/path.js"));
```

directory name

paste Actual path + And change Backward slash (\) to forward slash.

extension name

```
console.log(path.extname("E:/nodejsyoutube/pathModule/path.js"));
console.log(path.basename("E:/nodejsyoutube/pathModule/path.js"));
```

base filename

output

E:/nodejsyoutube/pathModule

output

.js

output

path.js

(2)

```
const path = require("path");
console.log( path.parse("E:/nodejsYoutube/pathModule/path.js"));
```

output

Gives five

Gives

root : 'E:/'
dir : 'E:/nodejsYoutube/pathModule'
base : 'Path.js',
ext : '.js',
name : 'path'

(3) Note :- But If you want to print particular property from parse then write.

```
const path = require("path");
const myPath = path.parse("E:/nodejs youtube/pathModule/path.js");
Console.log(myPath.name);
```

output

name : path

Node JS — How to create and Export our own Module in Node Js. — video - 12

thapa Technical.

Step-1 create two Javascript file having different names like index.js and oper.js

Step-2 let see an example (when we use single function)

index.js

```
Const {addy = require ("./oper");
Console.log( add(6,6));
```

oper.js

```
Const add = (a,b) => {
    return a+b;
}
module.exports = {add};
```

This means go go option.js (index.js) and Assign those js file which you want to Access. (oper.)

output
12

(we have only one function called add

we can use oper.js inside index.js with the help of module.exports. This will export the oper.js file and ./oper written inside index.js which can Access oper.js inside index.js.

Step-3 When we use Multiple function to perform Operation

index.js

```
const { add, sub, name, mult } = require ("./oper");

console.log (add (6, 6));
console.log (sub (10, 5));
console.log (mult (10, 5));
console.log (name);
```

oper.js

```
const add = (a, b) => {
    return a+b;
}
const sub = (a,b) => {
    return a-b;
}
const mult = (a,b) => {
    return a*b;
};
const name = "Vinod";

module.exports = { add, sub, mult, name };
```

output

12
5
50
Vinod

Node JS Complete NPM → Library video-13

Thapa Technical

(*) NPM stand for Node package Manager. → Library

⟹ npm is the world's largest Software Registry (which means npm hosts the most collection of software packages that developer can freely use, share, collaborate) — open source + to free use

Note:- ⟹ you can download npm public software packages without any registration or logon.

⟹ NPM is the default package manager for Javascript runtime Node.js. (which means NPM is already installed when node.js get downloaded.)

* NPM consists of two Main Parts *

(1) CLI (command line interface) for publishing and downloading packages / install software package

Ex- npm install <package>

(2) online Repository that hosts Javascript Packages.

file which co... metada... of proje...

Note

Note:-

* Software Package Manager *

⇒ All NPM packages are defined in files called **package.json**

⇒ The content of package.json must be written in **JSON**

⇒ **package.json** file which contains metadata of project. job to describe the project (it means which software package is used inside project)

Note Package.json will be generated when npm init is run to initialise Javascript / Node.js project.

Basic metadata provided by developers.

name
version
description
License

* Managing Dependencies *

⇒ NPM can Manage dependencies

⇒ NPM can (in one command line) install all the dependencies of a project.

Note:- Dependencies are also defined in Package.json

Dependencies

(1) dependencies ⟹ Specifies packages required by application to run. These packages are installed using `npm install`

(2) devDependencies ⟹ Specifies the packages required only for development and testing. (not necessary in production environment).

---

from video

NPM is library where you get readymade code which is defined as Package. we can simply use them and our project more interactive.

npmjs.com → official website

Step-1 Create folder npmMod

Step-2 Create file index.js inside folder

Step-3 open new terminal and write `npm init` (initialize npm)

→ write this on you main file.

Package name "npmmod"
version
description "thapa technical node tuts"
entry point main "index.js"
test comnd
git res
keyword
Author
licence
yes

Note Whatever you installed from NPM (npmjs.com) that get stored in package.json as dependencies.
(Simply Details of all package used from NPM).

Now trying to install package from npm
"chalk"

i → install

Step-1 Open terminal → write
npm install chalk or npm i chalk
+Enter

Step-2 Open Package.json shows details that-
Package is install.
if you want full explanation of Package.json
then open packe-lock.json

Step & Some codes Examples

(1)
```
const chalk = require("chalk");
console.log(chalk.blue("Hello world!"));
```

inside terminal write
node index.js

Output
Hello world ← blue color

(2)
```
const chalk = require("chalk")
console.log(chalk.blue.underline("Hello
world"));
```

inside terminal write
node index.js

Output
Hello world ← Blue color and underline

(3)
```
const chalk = require("chalk");
Console.log(chalk.green.underline.
inverse("Success"));
```

inside terminal write
node index.js

Output → background in color green

Note

Difference b/w Package.json vs package-lock.json?

Package.json	Package-lock.json
① It focuses on high level metadata and Direct dependencies.	① contains detailed and exact information about the entire dependency tree.
② Manually edited by developer can add or update info and dependences.	② Automatically generated and updated by npm when dependences are installed or modified.
→ gives Overview	③ gives detailed Explanation

Validator

(1)

```
const chalk = require ("chalk");
const validator = require ("validator");

Console.log (chalk.red.Underline.inverse
        ("false"));
Const res = validator.isEmail ("thapa@thapa
Console.log (res);
```

Before writing above code
~~install~~ npm i validator

After that,
node index.js

~~false~~
true

(2)

```
const chalk = require ("chalk");
const validator = require ("validator");

// Console.log (chalk.red.underline.
                inver ("false"));

const res = validator.isEmail ("thapa@thapa.
                com");
Console.log (res? chalk.green.inverse (res):
        Chalk.red.inverse (res));
```

output

~~true~~

Video 14 Nodemon in node.js

Thapa technical

Important-1:- please watch this video from youtube to get clear understanding.

* Which ever package we import from npmjs.com to our Javascript (Node.js) code. we can use these packages locally. (Not globally)

Ex:- In previous example we can see Chalk and validator package from npmjs.com. (which we can use locally).

Q) If we want to use Packages Globally then what should I have to do?

→ Open npmjs.com and Search nodemon

⇒ Import nodemon javascript (Node.js) code.

Inside terminal write,

[npm install nodemon -g] → global

Again write

[nodemon -v] → version

Again write

nodemon index.js → this will execute just like live server. if you made any change in the code. then it get executed directly.

Nodemon package

Nodemon ⟹ Nodemon is a utility that helps developers in the Node.js ecosystem by monitoring changes in their code and automatically restarting the Application.

Example:

```
const chalk = require("chalk");
const validator = require("validator");

const res = validator.isEmail("thapa@thapa.com");
console.log(res ? chalk.green.inverse(res)
    : chalk.red.inverse(res));
```

Now

Inside terminal write.

```
npm install nodemon -g
```

Again write

```
nodemon -V
```

Again write

```
nodemon index.js
```

Output

false

if we change thapa@thapa.com to thapa@thapa.mon then

Output

True

Note:

video 15 Module, wrapper function in node.js

Thapa Technical

* Module wrapper function *

→ Every module is wrapped in a function before it is executed. This function is called the module wrapper function.

→ Module wrapper function provides a private scope for the modules code and variables.

Please Elaborate.

we simply write this

```
const a = require("fs");
const name = "vinod";
console.log(name);
```

execute it-

But Behind the scene it's

Module wrapper function

```
function (export, require, module, __filename, __dirname) {
    const a = require("fs");
    const name = "vinod";
    console.log(name)
    module.export = {...}
});
```

this code is private

Note:- with the help of Module wrapper function we can simply use require, export, module and etc. This comes from Module wrapper function.

16. video Creating Our Own web server in Node.js.

thapa Technical

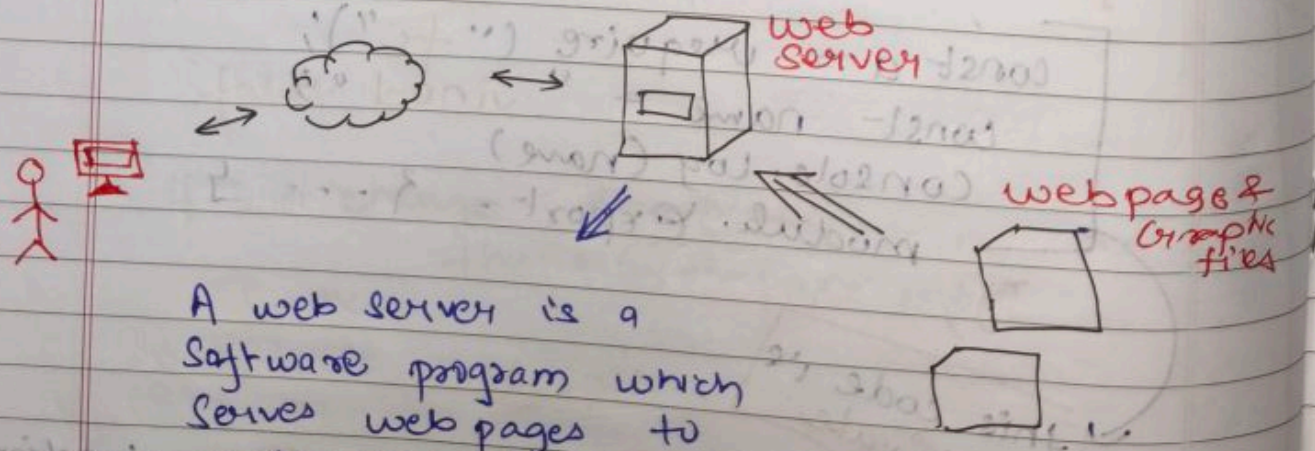
⇒ To access web pages of any web application, you need a web server.

⇒ The web server will handle all the http requests for the web application.

eg. IIS is a web server for ASP.NET web applications and Apache is a web server for PHP or Java web applications.

⇒ Node.js provides capabilities to create your own web server which will handle HTTP requests asynchronously.

⇒ You can use IIS or Apache to run Node.js web Application but it is recommended to use Node.js web server.



A web server is a software program which serves web pages to web users (browsers).

creating our own web server

```
const http = require("http")

const server = http.createServer((req, res) => {
    res.end("Hello from the other sides thapa");
});

server.listen(8000, "127.0.0.1", () => {
    console.log("listening to the port no. 8000");
});
```

inside terminal write
nodemon index.js

output

listening to the port no. 8000

17. video Node Js Routing

Thapa Technical Handled HTTP Requests in Node Js in Hindi

```
const http = require("http");

const server = http.createServer((req, res) => {

  if (req.url == "/") {

res.end("Hello from the home sides");
  } else if (req.url == "/about") {

res.end("Hello from the about us sides");
  }
  else if (req.url == "/contact") {

res.end("Hello from the contactus sides");
  }
});

server.listen(8000, "127.0.0.1", () => {

console.log("Listening to the port no. 8000");
});
```

output

Now search

localhost:8000/about <enter>

printed

localhost:8000/contact enter

See video 17 to get full information about

Client error (400-499)
Server error (500-599)
Succesful responses (200-299)
redirects (300-399)

Now if user enter some URL which doesn't exist then he got an error (code below)?

```
Same code as 1.
after res.end ("contacts"

else {
res.end ("404 error pages.page
          doesn't exist.");
});
```

Same as 1

it gives 404 error page but inside console getting status as 200 which means fetching succesful page.

So, we need to chang its status (200) to 404

```
Same code as 1.
else {
res.writeHead (404, {"Content-type":
                    "text/html"});

res.end ("<h1> 404 error pages.
          page doesn't exist </h1>");
});
```

Same as code 1.

Now getting output 404 and status also 404 and no document.

18. Video Complete JSON in Node JS

thapa Technical

JSON

⇒ JSON stands for Javascript object Notation.
⇒ JSON is a lightweight.
⇒ format for storing and Transporting data.
⇒ JSON is often used when data is sent from a server to a web page.

Code ①

```
const bioData = {
    name : "vinod"
    age : 26,
    channel : "thapa Technical",
};
Console.log( bioData.channel);
```

output

thapa technical

* Important :

There will be two important Methods for JSON

① parse( ); ⇒ JSON changes into an obj
② Stringify( ); ⇒ object changes into Json.

Convert object into JSON.

② Const bioData = {
name : "vinod",
age : 26,
channel : "thapa technical",
};

Const jsonData = JSON.stringify (bioData);
console.log (jsonData);

object changes into JSON.

output

{"name": "vinod", "age": 26, "channel": "thapa technical"}

③ Convert JSON into object

Same codes as 2 unit (bioData)
+
Const objData = JSON.parse (jsonData);
console.log (objData);

challenge to do

~~const fs = require()~~

1. convert to JSON ⟹ done
2. dusre file me add kardena
3. read file.
4. again convert back to js obj original
5. console.log

```
const fs = require("fs");
const bioData = {          (same as (2) write)
};

const jsonData = JSON.stringify(bioData);
fs.writeFile("json1.json", jsonData,
    (err) => {
  console.log("done");
});

fs.readFile("json1.json", "utf-8",
        (err, data) => {
  ~~console.log(data);~~
  ~~});~~
  const orgData = JSON.parse(data);
  console.log(data);
  console.log(orgData);
});
```

creating simple API

output

"name": vinod "age": 26 "channel": "thapa technical

name : 'vinod' age 26 channel 'thapa tech

difference • is only about quotes.

19. Create Simple API in Node JS

Thapa Technical

```
const server = http.createServer((req, res)
    => {
if (req.url = "/") {
  res.end ("Hello from the home side");
  }
  else if (req.url = "/about") {
   res.end ("Hello from the About-Us
                side");
  }
   else if (req.url = "/contact") {
   res.end ("Hello from the contactus side");
   }
   else if (req.url = "userapi") {
fs.readfile ('${__dirname}/userAPI/
        userapi.json', "utf-8",
          (err, data) => {

console.log (data);
res.end(data);
  });
} else {
res.writeHead (404, {"content-type":
                "text/html"});
res.end ("<h1> 404 error page.
          page doesn't exist </h1>");
});
server.listen (8000, "127.0.0.1", () => {
   console.log ("Listening to the port
                no. 8000");
});
```

creating simple API

inside userapi you get all information written in server.

output

(2) Same as 1 only chang req part -

```
else if (req.url = "/userapi"){
fs.readfile(`${__dirname}/userAPI/
    userapi.json`, "utf-8", (err,data)
    ⟹ {
    console.log(data);
    const objData = JSON.parse(data);
    res.end(objData[0].name);
    });
```

Output

you get particular name
Leanne Graham

(3) Same as 2 changie only index value to [2].

Important Remainder

classmate
Date
Page

After completing video

1 to 19 (thapa - Technical)
(thoroughly or practically)

+

then

Watch video ⇒ 20, 21, 22 from youtube.

⟹ final Completion.

20. Events Module in Node.js : Handling Events in Nodejs with EventEmitter.

thapa technical

classmate Date Page

challenge

① Example -1

Registering for the event to be fired only one time using once.

② Example 2

create an event emitter instance and register a couple of callbacks.

③ Example 3

Registering for the event with callback parameters.

Solution for 3

Code-1

Solution for 1 and 2:

```
const EventEmitter = require("events");
const event = new EventEmitter();

event.on("SayMyName", () => {
  console.log("Your name is vinod");
});
event.on("SayMyName", () => {
  console.log("Your name is Bahadur");
});
event.on("SayMyName", () => {
  console.log("Your name is thapa");
});

event.emit("SayMyName");
```

output

your name is Vinod
your name is Bahadur
your name is Thapa

Solution for 3

```
const EventEmitter = require("events");
const event = new EventEmitter();

event.on("checkpage", (sc, msg) => {
    console.log(`Status code is ${sc}
        and the page is ${msg}`);
});
event.emit("checkpage", 200, "ok");
```

output

Status code is 200 and the page is ok.