

Practical Introduction to NodeJS

- **Himanshu Malik**

Overview: -

In this module we are going to learn NodeJS, with some awesome and real world uses of NodeJS: -

- First thing first, with the help of NodeJS we will open up the browser and open our “Pepcoding” website.
- Then we will try to grab Wi-Fi information with the help of NodeJS.
- After that we will print “#Pepcoders” using command line with the help of NodeJS.
- At the end we have to learn one newer module called “Path module”.

Introduction: -

What is NodeJS?

It allows us to take JavaScript out of the browser and it liberates it, allowing it to interact directly **with the hardware of a computer**.

Okay, let me explain you with an example, Consider NodeJS in an “Android Phone” and the phone have applications installed in it and by touching on the apps we can use them. Similarly, NodeJS have



modules like – “FS, OS, Path, Child Process”. So, by the help of JavaScript we can call these modules and interact with your computer.

Let’s talk about all the modules that NodeJS provide with some cool and fun experiment.

1. **Child Process:** - Node.js runs in a single thread, it manages concurrency using an event-driven architecture. It also makes it easier to create child processes in order to take advantage of parallel processing on multi-core CPU platforms. **This is an inbuilt module.**

Example –

By running this command, you can open up the calculator.

```
JS childprocess.js U ●
JS childprocess.js > ...
1 let cp = require("child_process");
2 console.log("Open up the calculator");
3 cp.execSync("calc");
4 console.log("Open calculator");
5 |
```

Or similarly you can open up the VScode using below command.

```

JS childprocess.js U ●
JS childprocess.js > ...
1 let cp = require("child_process");
2 console.log("Open up the VScode");
3 cp.execSync("code");
4 console.log("Open IDE");
5

```

Let's open up our lovely website,

```

JS childprocess.js U ●
JS childprocess.js > ...
1 let cp = require("child_process");
2 console.log("Trying to open our website");
3 cp.execSync("start chrome https:\\www.pepcoding.com");
4 console.log("It's opened");
5

```

By looking at above examples we can say that "child Process" *is a very modern module.*

2. **OS module:** - OS is a node module that provides information about the operating system of a computer. It provides the operating system's hostname and **the amount of free ram in bytes.**

Let me explain with some example –

This code line gives the information about your Machine what you are using.

```

JS os.js U ●
JS os.js > ...
1 let os = require("os");
2 // below line will check architcture wether its is 64 or 32 bit
3 console.log(os.arch());
4 // below line will check which machine we are using
5 console.log(os.platform());
6

```

Using these code lines, you can check your Wi-Fi information [*You become Hacker Now* 😊]

```

JS os.js U ●
JS os.js > ...
1 let os = require("os");
2 // below line will give you information about your Wi-Fi
3 console.log(os.networkInterfaces());
4

```

```
{
  'VirtualBox Host-Only Network': [
    {
      address: 'fe80::4462:5737:ccdc:50fb',
      netmask: 'ffff:ffff:ffff:ffff::',
      family: 'IPv6',
      mac: '0a:00:27:00:00:09',
      internal: false,
      cidr: 'fe80::4462:5737:ccdc:50fb/64',
      scopeid: 9
    },
    {
      address: '192.168.56.1',
      netmask: '255.255.255.0',
      family: 'IPv4',
      mac: '0a:00:27:00:00:09',
      internal: false,
      cidr: '192.168.56.1/24'
    }
  ]
}
```

Output: -

This code will give us a very important information about our server that do we are using 64- or 32-bit architecture server.

```
JS os.js  U ●
JS os.js > ...
1  let os = require("os");
2  // below line will give you information about server load
3  console.log(os.cpus());
4
```

3. **FS(Filesystem):** - The Node.js file system module allows you to interact with your computer's file system. **We can read, create, update, delete and rename the file** using "FS" module.

Let's explore it with some examples.

These command lines help you to read the file content.

```
JS fs.js  U ●  JS index.js U ●
JS fs.js > ...
1  let fs = require("fs");
2  // below line will helps in read the content written in that file
3  let buffer = fs.readFileSync("index.js");
4  console.log("bin data " + buffer);
5  // above line prints the data in concatenation(String)
6
```

```
JS fs.js  U ● JS index.js U ●
JS index.js > ...
1  var sum = 10 + 20;
2  console.log(sum);
```

The below code will create the file.

```
JS fs.js  U ● ≡ abc.txt U JS index.js U ●
JS fs.js > ...
1  let fs = require("fs");
2  // below line will helps in read the content written in that file
3  let buffer = fs.readFileSync("index.js");
4  fs.openSync("abc.txt", "w");
5  // above line create the file
6
```

The below code helps to write in the file, but we face one problem in that i.e. if you write new content the old one will replace automatically

```
JS fs.js  U ● ≡ abc.txt U JS index.js U ●
JS fs.js > ...
1  let fs = require("fs");
2  // below line will helps in read the content written in that file
3  let buffer = fs.readFileSync("index.js");
4  fs.writeFileSync("abc.txt", "Aaj mausam bda beiman hn");
5  // above line write the content in the file
6
```

```
≡ abc.txt
1  Aaj mausam bda beiman hn
```

To solve the problem of not replacing old content, use the following code. What it will do is it adds new content into the previous one.

```
JS fs.js U ● abc.txt U ● JS index.js U ●
JS fs.js > ...
1 let fs = require("fs");
2 // below line will helps in read the content written in that file
3 let buffer = fs.readFileSync("index.js");
4 // fs.writeFileSync("abc.txt", "Aaj mausam bda beiman hn");
5 // above line write the content in the file
6
7 fs.appendFileSync("abc.txt", "Bhai kiski yaad mn ho");
8 // Add the line in that file
9
```

```
JS fs.js U ● abc.txt U ● JS index.js U ●
abc.txt
1 Aaj mausam bda beiman hnBhai kiski yaad mn ho
```

Using these command lines help to create a folder, and in that folder creates a file and you can get your content over there.

```
JS fs.js U ● a.txt U ●
JS fs.js > ...
1 let fs = require("fs");
2
3 //below line creates a folder
4 fs.mkdirSync("ThisFolder");
5 fs.writeFileSync("ThisFolder/a.txt", "Yooo we are back");
6 //above line creates a file in that folder and write
```

```
JS fs.js U ● a.txt U ●
ThisFolder > a.txt
1 Yooo we are back
```

This command helps you to remove file present in a folder and that folder as well.

```

JS fs.js  U ●
JS fs.js > ...
1  let fs = require("fs");
2
3  //below for loops helps to remove all the file present in that folder
4  let content = fs.readdirSync("ThisFolder");
5  for(let i = 0; i < content.length; i++){
6      console.log("file", content[i], "is removed");
7      //remove all the files
8      fs.unlinkSync("ThisFolder/" + content[i]);
9  }
10
11  fs.rmdirSync("ThisFolder");
12  //above line removes the folder

```

This command gives you existence of file and path, return true/false if present or not.

```

JS fs.js  U ●
JS fs.js > ...
1  let fs = require("fs");
2
3  //The following code check files exist and returns True/False
4  let doesPathExist = fs.existsSync("a.txt");
5  console.log(doesPathExist);
6  doesPathExist = fs.existsSync("b.txt");
7  console.log(doesPathExist);
8  //The following code gives the path of working file and returns True/False
9  let detailsObj = fs.lstatSync(__dirname + "\\fs.js");
10 let ans = detailsObj.isFile();
11 console.log(ans);
12 ans = detailsObj.isDirectory();
13 console.log(ans);

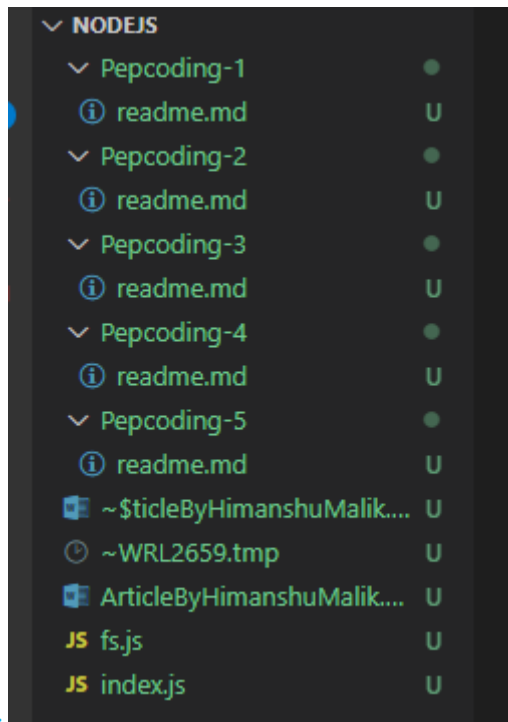
```

By using this command, you will be able to create multiple folders and you can add each file in each folder.

```

JS fs.js  U ●
JS fs.js > ...
1  let fs = require("fs");
2
3  for(let i = 1; i <= 5; i++){
4      let dirPath = `Pepcoding - ${i}`;
5      fs.mkdirSync(dirPath);
6      fs.writeFileSync(dirPath + "\\ " + "readme.md", `# readme for ${dirPath}`);
7  }

```



Output: -

4. **Path:** - The Path module allows you to work with file and directory paths, **depending on your Operating System.**

For example – If you are using Windows then there is different syntax to mention path in code, and if you are using MacOS it has different one.

Now one Question rise in your mind i.e. Why is that difference, it's because of the conflict between "Bill Gates" & "Steve Jobs". So, to resolve that we will use path, **let me show you how this will be gone work.**

```
JS path.js U ●
JS path.js > ...
1 let path = require("path");
2 for(let i = 1; i <= 5; i++){
3   let dirPath = `Pepcoding - ${i}`;
4   fs.mkdirSync(dirPath);
5   fs.writeFileSync(path.join(dirPath, "readme.md"), `# readme for ${dirPath}`);
6 }
7
```

Let's talk about extension, and by what code line you can get your file extension.

```
JS path.js U ●
JS path.js > ...
1 let path = require("path");
2 let ext = path.extname(path.join(__dirname, "abc.js"));
3 console.log("ext", ext);
4
```

One more thing to understand, what if we have very big path and we don't know file and folder name. **To get that we use Base.**

If we have big path then it will give the file name, and the folder name.

```
JS path.js U ●
JS path.js > ...
1 let path = require("path");
2 let name = path.basename(__dirname);
3 console.log(name);
4 name = path.basename(path.join(__dirname, "abc.js"));
5 console.log(name);
6
```

Till, now we have covered all the modules present in NodeJS.

Now I will teach you how to print “#Pepcoders” in CMD (Command Line).

For, that you have to install a package named “chalk” using your terminal by the following commands “npm i chalk”.

```
JS npm.js U X
JS npm.js > ...
1 let chalk = require("chalk");
2 console.log(chalk.red('Hello Pepcoding'));
```

```
JAIDEV SINGH MALIK@DESKTOP-A68GMR5 MINGW64 ~/C
ster)
$ node npm.js
Hello Pepcoding

JAIDEV SINGH MALIK@DESKTOP-A68GMR5 MINGW64 ~/C
ster)
$
```

Output: -

For more about chalk you can check out - <https://www.npmjs.com/package/chalk>

But we have to print “#Pepcoders”, for that we will use one more module name “figlet”. Similarly, we have to install package for “figlet” by the command “npm i figlet”. For more about figlet check out - <https://www.npmjs.com/package/figlet>

```
JS npm.js U X
JS npm.js > ...
1 let chalk = require("chalk");
2 let figlet = require("figlet");
3 console.log(figlet.textSync('#Pepcoding'));
```



```

ster)
$ node npm.js

```



Output: -

But, if you look at this this is not in red colour, so let us combine both chalk & figlet.

```

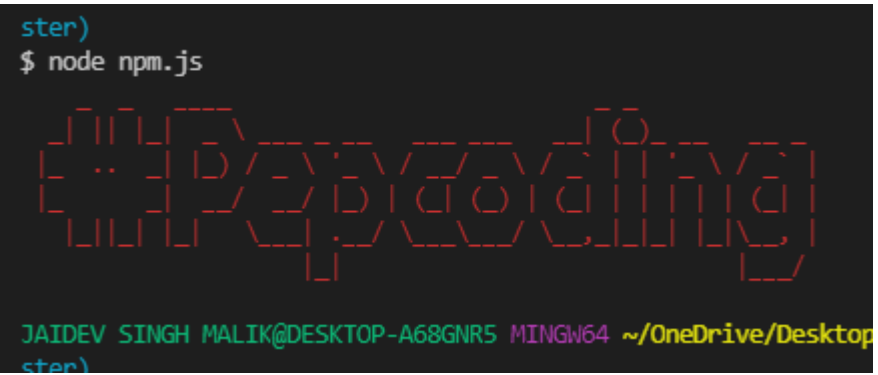
JS npm.js U X
JS npm.js > ...
1 let chalk = require("chalk");
2 let figlet = require("figlet");
3 console.log(chalk.red(figlet.textSync('#Pepcoding')));

```

```

ster)
$ node npm.js

```



Output: -

Conclusion: -

So, from this what we conclude is that using Node is having a lot of fun and you get a good more amount of knowledge regarding JavaScript and how to build real life software's like – Atom & lot more.

References to learn more about NodeJS: -

- <https://nodejs.dev/learn>
- <https://www.youtube.com/watch?v=cjJWtQpotE>