**NodeJS Self Prepared Notes**

**(16/4/2024)**

**What is Node.JS:**

* It is an open source server side runtime environment built in chromes V8 JS engine.
* It provides an event driven, non—blocking(asynchronous) I/O and cross platform runtime environment for building highly scalable server-side application using JavaScript: Event ke through hone wale operations, Asynchronous operations & Provides Cross platform environment for build server side application using JS through the help of Node.JS.
* Node.JS can be used to build different types of application such as command line application, web Application, REST API server etc.
* It is mainly used for build Network programs like web servers, Similar to PHP, JAVA or ASP.Net.
* Node.JS was written & introduced by **Ryan Dahl in 2009.**

**Advantages:**

1. Nodes.js is an open-source framework.
2. Uses JS to build entire server side applications.
3. Lightweight framework that includes bare minimum modules:
4. Asynchronous by Default. So it performs faster than other frameworks.
5. Cross-platform framework that runs on windows, MAC or Linux.

**Setup:**

1. Node.js
2. Node package Manager (NPM)
3. IDE (Integrated Development Environment) or Text Editor

**Node.JS Console REPL:**

* Node.js comes with virtual Environmemt called REPL (aka node shell)
* REPL stands for Read-Eval-Print-Loop.
* It is a quick and easy way to test simple Node.js/JS Code.
* To launch the REPL, open cmd and type node (press enter) // ctrl+d: bahar aane ke liye

**REPL e.g:** Arithmetic operation, string concatination operations.

**For Save Code File:**

.save filename. extension

**Check Node.js is Available or not in System:**

Node -v

Npm -v //It will display NPM version

**How to enter REPL Environment:**

To enter REPL Environment for that write node (press enter).

* Save code from REPL Environment to file.

.save filename.js

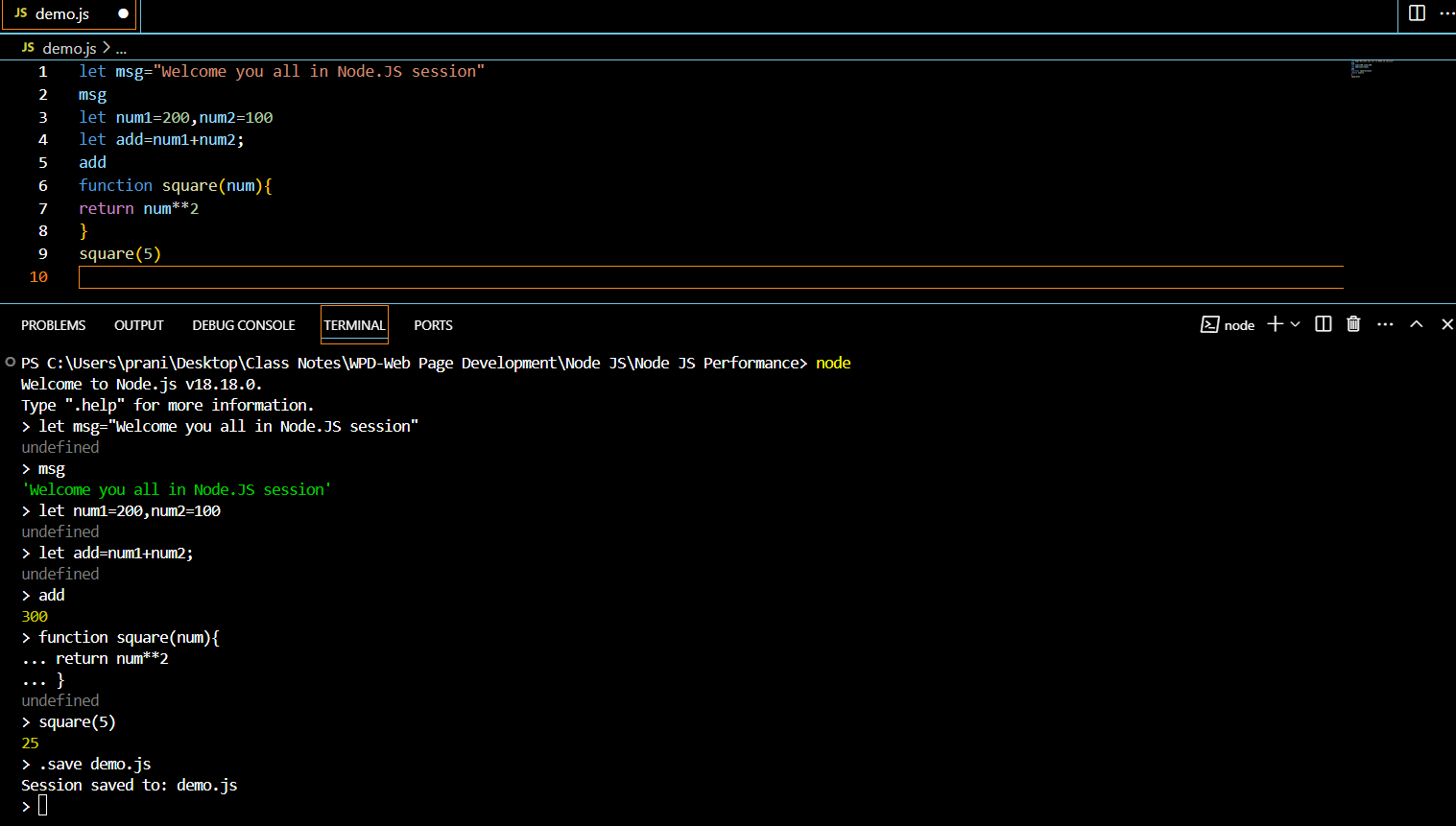
* To load code from file to REPL Environment.

.load filename.js

* To come out from REPL Environment // ctrl+d

**Demo Code:**

Run on Terminal and Reflects on Code file.





**Node.JS Module:**

* Module in Node.JS is simple or complex functionality organized in a single or multiple JS Files which can be reused throughout the Node.JS Application.
* Each module in Node.JS has its own context, so it cannot interfere with other modules.
* Each module can be placed in a separate.js file under a separate file.

**Node.JS Module Types:**

1. Core Modules: //Inbuilt Modules
2. Local Modules //We created
3. Third party Modules //use already Built & Installed modules.

**Core Modules:**

* The core module includes bare minimum functionalities of Node.js.
* These core modules are compiled into its binary distribution and load automatically when Node.JS process starts.
* However, we need to import the core module first in order to use it is in your application.

**Example of Core Module:**

1. http: To create Http Server in Node.JS
2. os: Provides information regards Operating System.
3. path: To handle the file path.
4. url: To parse the url string.
5. fs: To handle the file system in computer.
6. querystring: To handle the URL query string.
7. util: To access Utility function.

**Node.JS Modules:**

1. Core Module: Module Provide by Node.JS to Us.
2. Local Module: Module created by Developer for Own Use.
3. Third Party Module: Module developed by Third Party Vender.

* First, we need to install it & then Use It & then Import it.

**Core Modules:**

1. Path module:

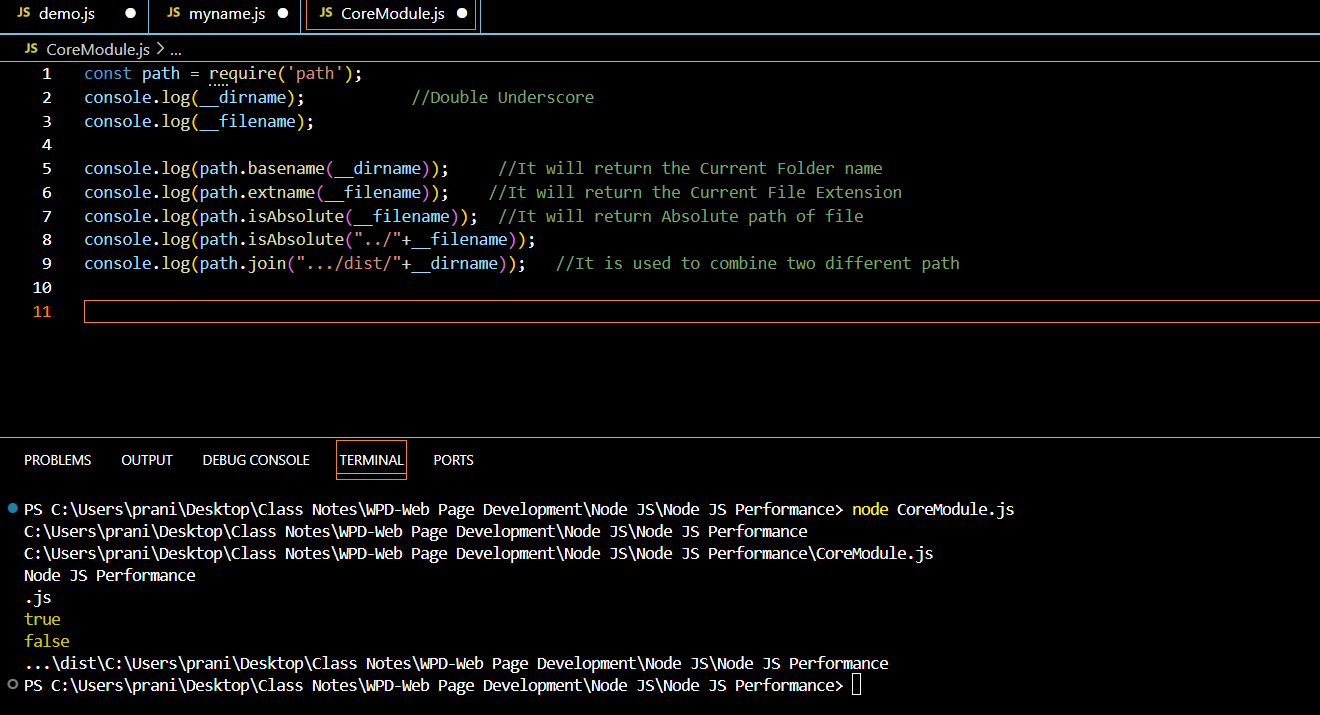
It provides a lot of very useful functionality to access & interact with the file system.

It is used for file related path operations.

In node.js by default we have Two Object:

\_dirname: It will return Current directory.

\_filename: It will return current filename with directory path.



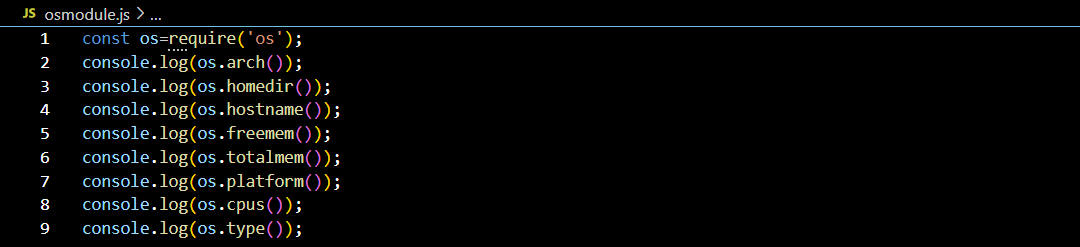
1. **OS Module:**

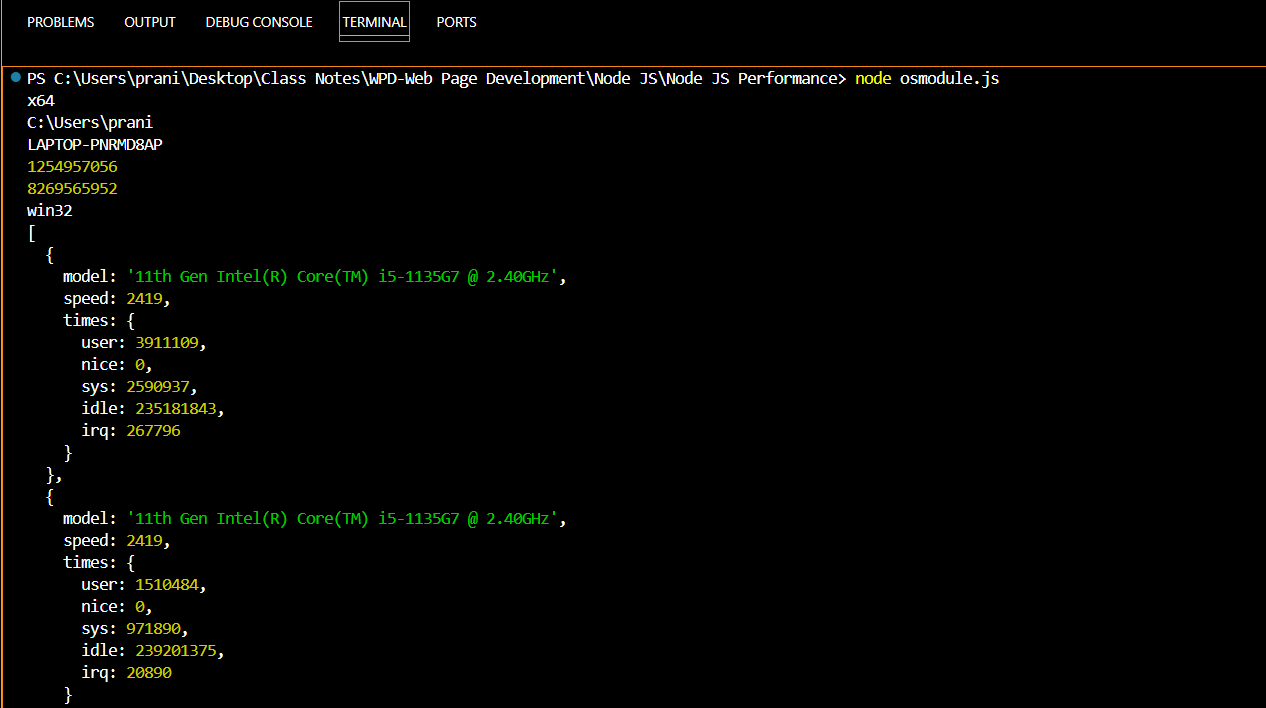
This module provides information of OS and the computer the program runs on, And interact with it.

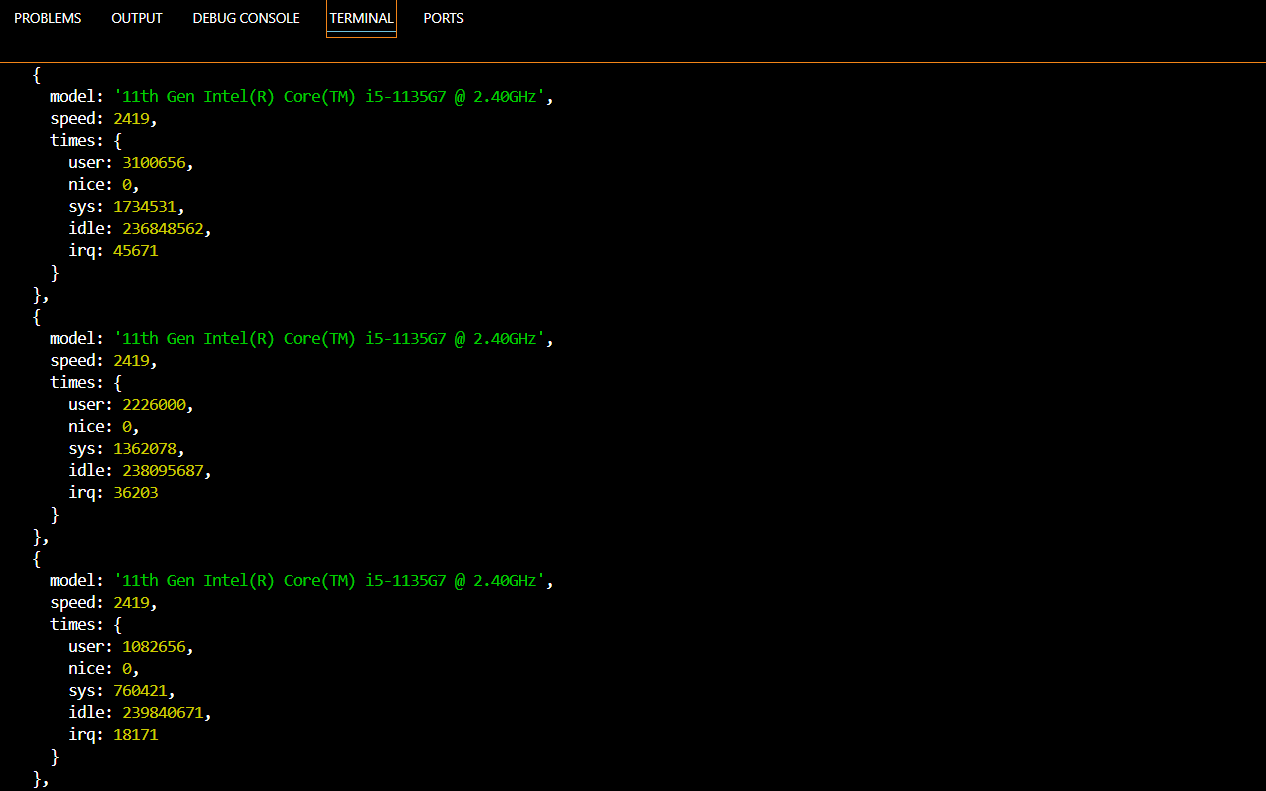
Const os= require(‘os’);

**Method:**

* **Os.arch():** Return the string that identifies the underlying architecture like arm,x64,arm64.
* **Os.freemem():** Return the number of bytes that represent the free memory in the system.
* **Os.homedir():** Return the path to the home directory of the current user.
* **Os.totalmem():**Returns the number of bytes that represent the total memory available in system.



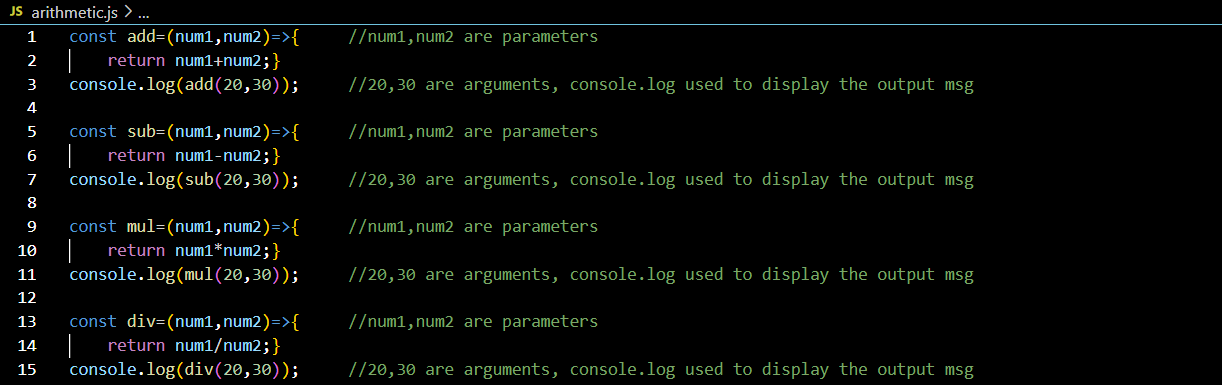


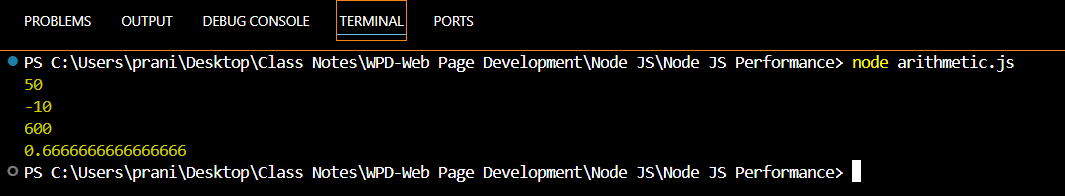




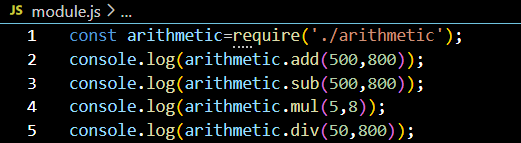
1. **Local Module:**

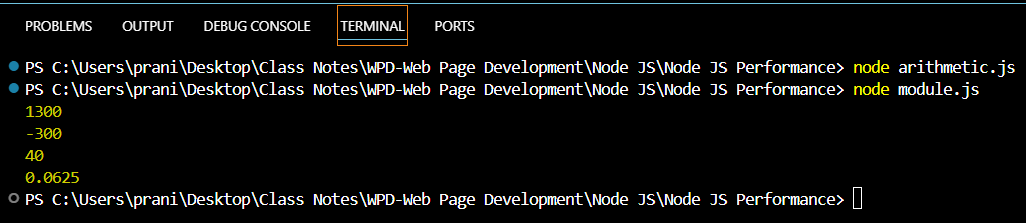
* The modules which are created locally in your Node.JS module.
* These modules include different functionalities of you application in separate files and folders.
* You can also package it and distribute it via NPM, So that Node.js community can use it.
* Export Literals
* Export function
* Export function as a class.
* The module.exports is a special object which is included in every JS File in the Node.JS Application by default.
* Use module.exports or exports to expose a function,object or variable as a module in Node.js.
* The require() function will return an object, function, property or any other JS type, depending on what the specified module returns.





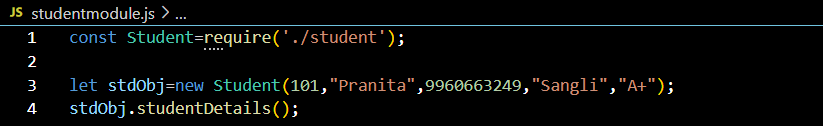
**Arithmetic operations code**

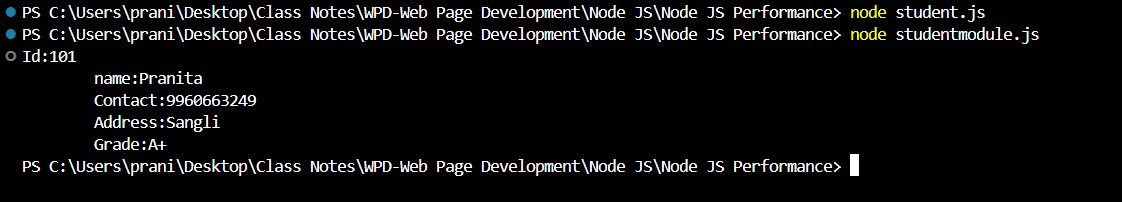




**Task: Create one class with name Student containing stdId, stdName, stdContact, stdAddress, stdGrade and one function with name studentDetails which will display student all information.**

****

****

****

1. **NPM: Node Package Manager:**

* Node Package Manager (NPM) is a command line tool that installs, updates or uninstalls Node.js packages in your application.
* It is also an online repository for open-source Node.js packages
* The node community around the world creates useful modules and publishes them as packages in this repository.
* NPM is included with Node.js installation and You can verify using npm-v command.
* If you have an older version of NPM then you can update it to the latest version using the following command.
* Npm install npm -g.

**Command Use With NPM:**

Npm install module-Name –sav //Install third party module locally and add dependency in package.json file

Npm update module-Name //Update third party module

Npm update module-Name -g //Install third party module globally

Npm uninstall module-Name //uninstall third party module

**How to Use Third Party Module:**

Step1: Install module using npm like=> npm install module-name-save

Step2: Import it in file where you want=> var chalk=require(‘chalk’)

**Name of some third party module:**

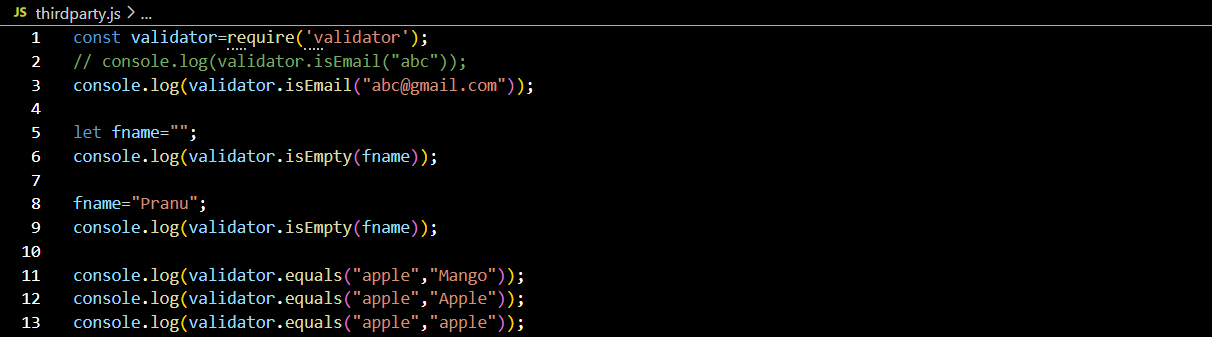
1. Chalk,
2. Validator
3. express

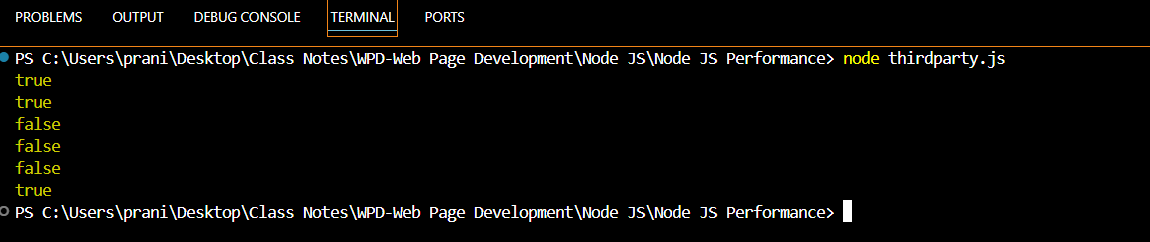
**Third party module:**

* First we have to install it.
* Then import it where want.
* After installation we have import it where we want.
* You will get all third party module and library in npmjs.com.

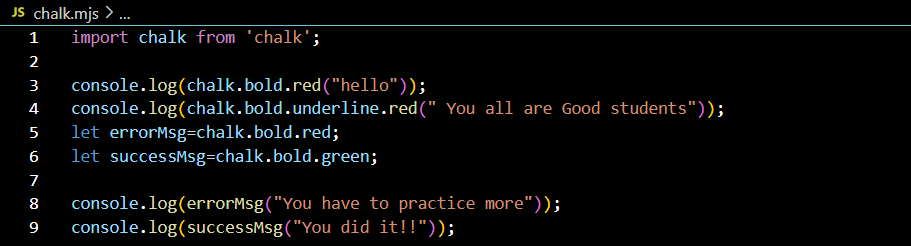
**Use of Validator module:**

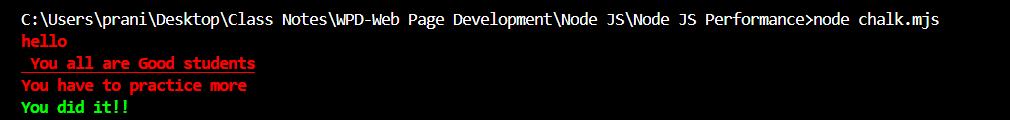
Npm i validator //In command prompt Run it





**Use of Chalk Module:**

****

****

**Blocking & non-blocking:**

* Blocking is when the execution of additional JS in the Node.JS process must wait until a non-Javascript operation completes.
* This Happens because the event loop is unable to continue running JS while a blocking operation is occurring.
* All of the I/O methods in Node.JS standard library provides asynchronous versions, which are non-blocking and accept callback functions. Some methods also have blocking counterparts, which have names that end with sync.
* Blocking methods execute synchromously and non-blocking methods execute asynchronously.

1. **Blocking: Synchronous**
2. **Non-Blocking: Asynchronous**

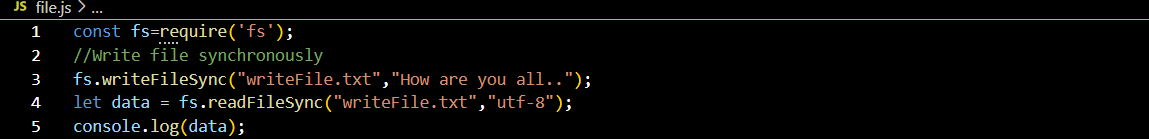
**Callback:**

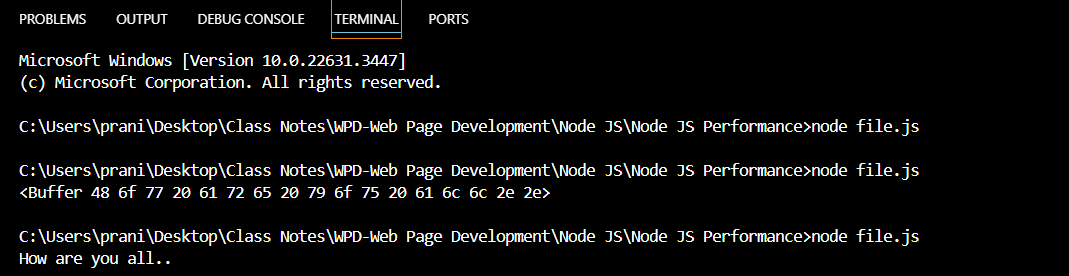
* A Callback is a function called at the completion of a given task, this prevents any blocking and allows other code to be run in the meantime.
* The general idea is that the callback is the last parameter. The callback gets called after the function is done with all its operations. Traditionally, the first parameter of the callback is the error value. If the function hits an error, then they typically call the callback with the first parameter being an error object. If it cleanly exists, then they will call the callback with the first parameter being null and the rest being the return values.

**Events:**

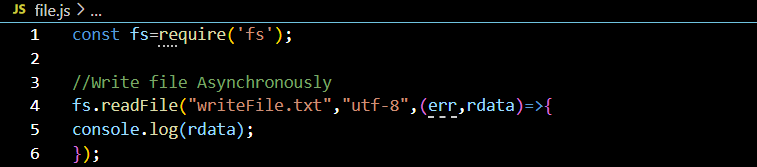
* Every action on a computer is an event,Like when a connection is made or a file is opened.
* Node.JS has a built-in module,called “Events”, where you can create,fire and listen for your own events.
* All event properties and methods are in instance of an EventEnitter object.To be able to access these properties and methods,Create an EventEmitter object.
* // Get the reference of EventEmitter class of events module: Var events=require(‘events’);
* //Create an object of EventEmitter class by using above reference: var em=new events.EventEmitter();
* Emit is used to trigger an event.
* On is used to add acallback function that’s going to be executed when event is triggered.

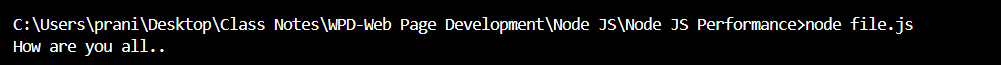
**For Synchronous file: Sync word before filename.**

****

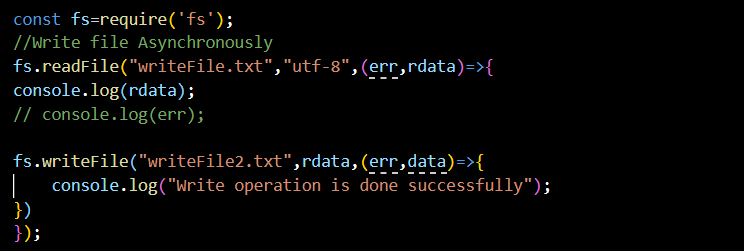
****

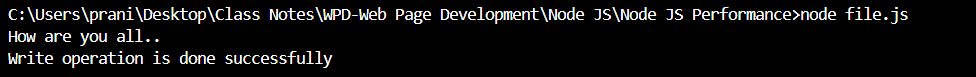
**For Asynchronous file: no any word before filename.**



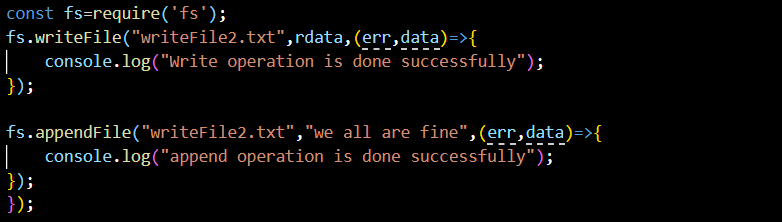


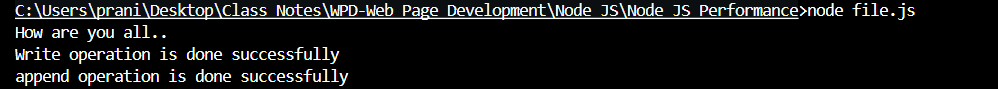
**Write asynchronously with error:**



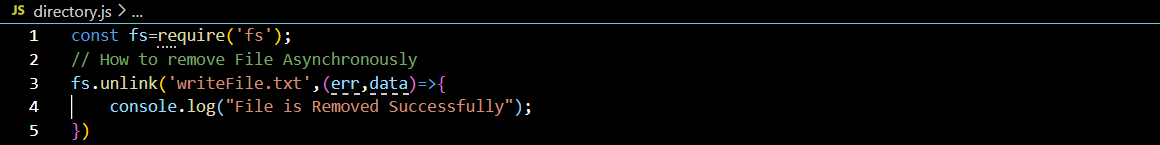


**Append the Data into file:**



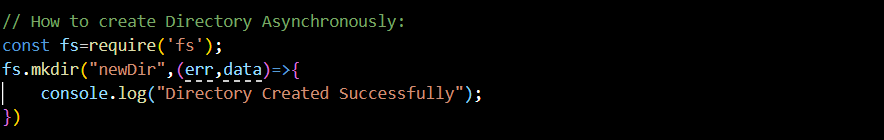


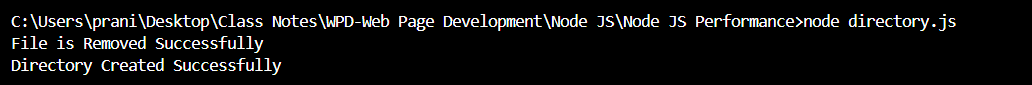
**How to Remove File Asynchronously:**



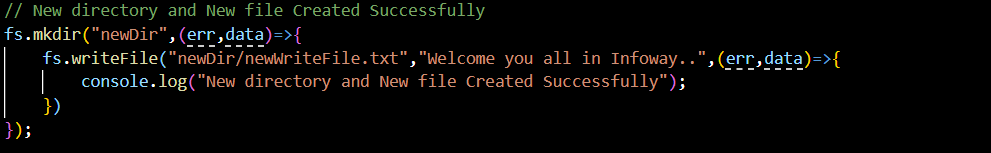


**How to create Directory Asynchronously:**

****

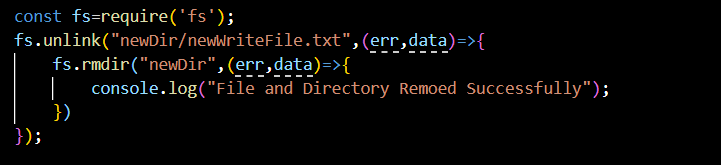
****

**New Directory & New File Created Successfully:**



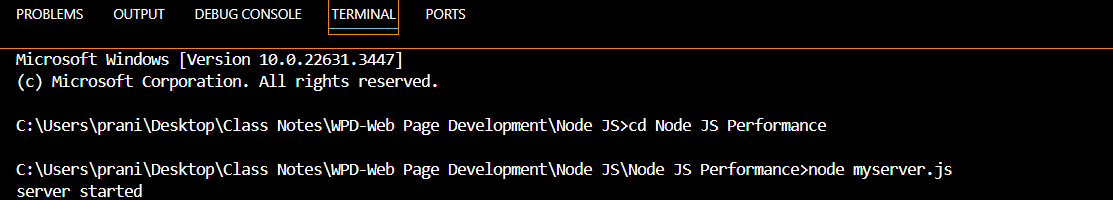


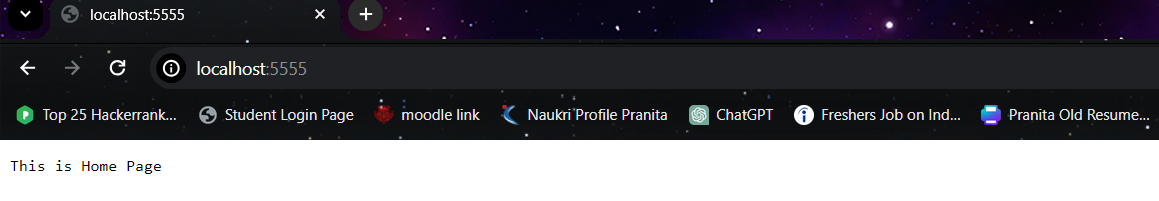
**New Directory & New File Removed Successfully:**

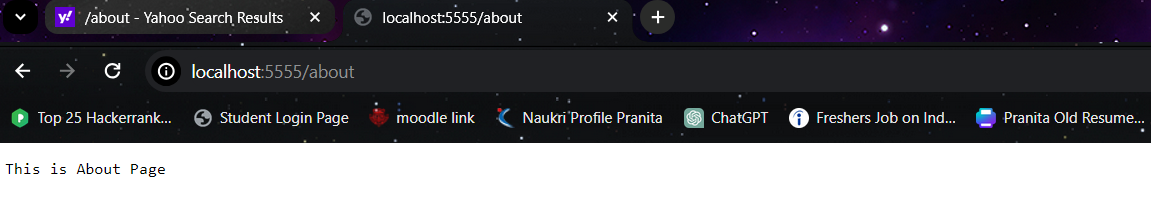
****

****

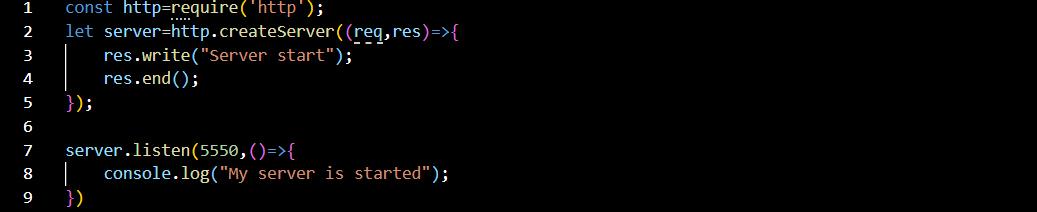
**On Server By default Index file is opened in server:**

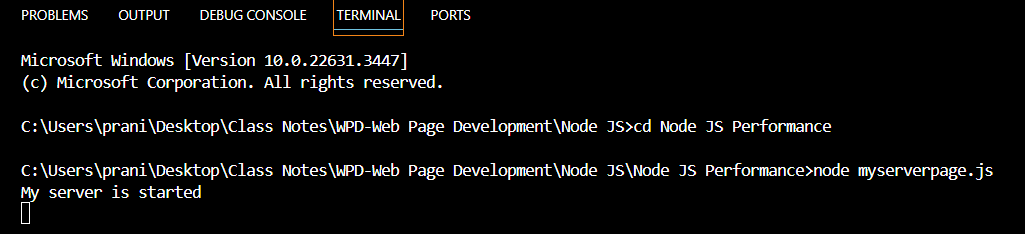




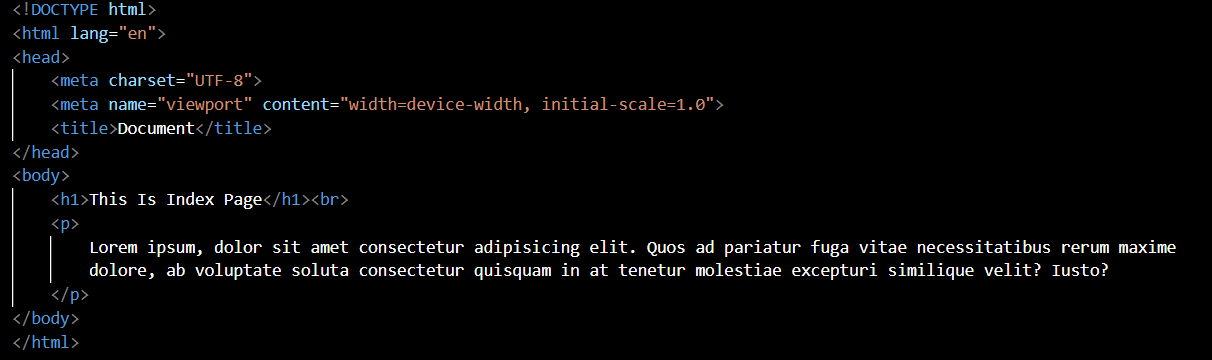


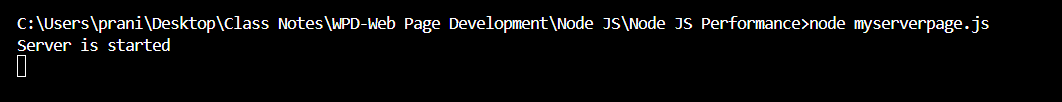
**Basic Code to Create a New Server:**





**How to create HTML Page:**





Run The myserverpage.js code on Chrome by typing: localhost:7770