**Upskills report**

**Nodejs session**

Monday 25-05-2020

Topic:

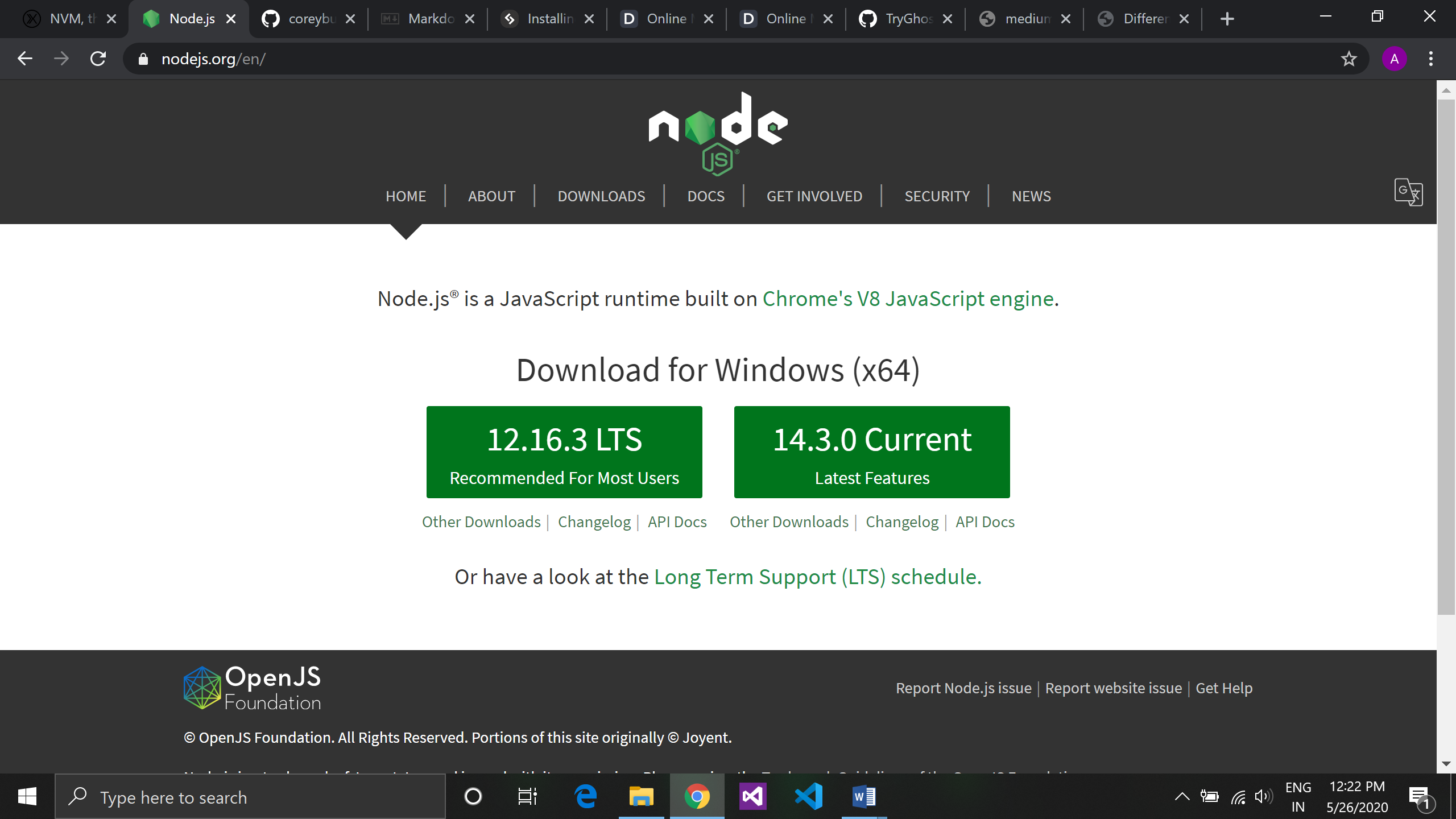
1. Installation of Nodejs
2. Installation of NPM and NVM with respect to linux.
3. Difference between NPM and NVM.
4. Semantic versioning in javascript
5. Creating a sample express application
6. Markdown

**Nodejs:**

* Node is lightweight open source.
* Node is a open source runtime environment includes everything you need to execute a program written in JavaScript.
* JavaScript and Node.js run on the V8 JavaScript runtime engine. This engine takes your JavaScript code and converts it into a faster machine code.
* Node.js is Free.
* Node.js runs on various platforms.
* Node.js uses JavaScript on the server.

**Installation of node in system**

* So basically the installation process is depends on user Operating system
* If user have windows then user need to download nodejs from the official site. link <https://nodejs.org/>
* After download double click on exe file.
* One prompt will appear read the instruction and click on next…go on until finish
* Node will be installed successfully in system.

ss

**Difference between NPM and NVM:**

* NPM stands for node pakage manager it is a tool that allows you to install javascript packages
* We can check a version using command npm –v, node -v after installation
* Npm is a package manager which holds one version we cannot switch our node versions
* To install NPM =>go to <https://nodejs.org/> download latest version of LTS node and install in system with this node npm automatically gets installed.
* NVM stands for node version manager it is a tool that allows you to install and download nodejs.
* We can check a version using command nvm –v, node –v after installation
* Nvm is used to keep multiple versions of nodejs and work with different versions at a time.
* To switch version Command => nvm use <version\_name>

**Semantic versioning:**

Semantic versioning is defined the versions of software and there updation bug fixing report.

* Version: 1.0.0

Semantic versioning is classified in 3 phases

* + Major : first digit
  + Minor : second digit
  + Patch : third digit
  + If major digit changed it means it is a major change in software the new features break backwards compatibility.
    - * Version: 1.x.x
  + If minor digit changed it don’t break existing feature it may not affected on major number.
    - * version :1.1.x
  + If there is some report on software that means it is bug, when developer fixed this bug the last digit of versioning get increment by one.
    - * Version:1.1.1

**Creating a sample express application:**

* Here we create one sample project in VS Code and look into that project and run on Browser for 1st Experience.
* Starting from Scratch create one Directory if needed.
* Write in Terminal **Express Sample** it means we create a project name sample in that directory.
* then **cd Sample** change th directory for installing its dependencies.
* **node install** – all the packages which are required get ito the node module.
* here the basic Sample project is ready to Run o Browser for that just write a command.
  + **npm start** .

**Markdown:**

* Markdown is a lightweight markup language based on the formatting conventions that people naturally use in email.
* It is a lightweight markup language with plain text formatting syntax.
* Markdown can be used for everything. People use it to create [websites](https://www.markdownguide.org/getting-started/#websites), [documents](https://www.markdownguide.org/getting-started/#documents), [notes](https://www.markdownguide.org/getting-started/#notes), [books](https://www.markdownguide.org/getting-started/#books), [presentations](https://www.markdownguide.org/getting-started/#presentations), [email messages](https://www.markdownguide.org/getting-started/#email), and [technical documentation](https://www.markdownguide.org/getting-started/#documentation).
* Markdown is portable.
* Markdown is platform independent. You can create Markdown-formatted text on any device running any operating system.

**Tueday 26-05-2020**

**Topic:**

1. Yarn
2. Package.json
3. How to create API
4. Understanding and deconstructing code:

**Yarn:**

* Yarn is new package manager for nodejs (javascript) work same as a NPM (node Package Manager).
* It is a fast, secure and reliable dependence management.
* Yarn caches all the installed packages and yarn packages installed simultaneously that’s why yarn is faster that NPM.
* With respect to npm, when we deploying the project on different machines, the versions of installed packages can be different that was the reason Yarn appeared in the first place.

**Package.json:**

* Package.json file used to sync software development for multiple application.
* It is a json file which contain data in text format and data in form of Key**: Value** pair.
* NPM (Node Package Manager) uses this package.json file information about Node JS Application information or Node JS Package details.
* Package.json file contains a number of different directives or elements.
* The directives helps NPM to manage and handle the packages.
* It manage all the dependencies.
* It can hold the entry point of the application.

Package.json file looks like:

{

"name": "thustask",

"version": "0.0.0",

"private": true,

"scripts": {

"start": "node ./bin/www"

},

"dependencies":

{

"cookie-parser": "~1.4.4",

"debug": "~2.6.9",

"express": "~4.16.1",

"http-errors": "~1.6.3",

"jade": "~1.11.0",

"morgan": "~1.9.1"

}

}

**How to create API:**

* + In the first session we was installed node, NPM, NVM and VS Code in our system.
  + To create API we need to run some few commands in our terminal that are follows below:
  + Express <project\_name> this will create a directory in the present directory.
    - e.g: express task;
  + cd <project\_name> this will change our directory to created directory.
    - e.g.: cd task;
  + npm install; this command is used to install node packages and there dependencies in directory.
  + Npm start; this command start your api server and run your project on web

**To create our new api follow below steps:**

Step 1:

* + Create a new file in routes directory with .js extenstion.
  + In that file write API code with some javascript libraries:

var express = require('express');

var router = express.Router();

/\* GET home page. \*/

router.get('/', function(req, res, next) {

res.render('index', { title: 'thusday' });

});

module.exports = router;

Step 2:

go to app.js register new api file and setup in view engine code are follows:

var task = require('./routes/task');

app.use('/', task);

Step3:

npm start (run this command in terminal) and open web browser enter localhost:3000/task to run api.

*Note: in route folder all available files are API.*

*By default in node every file is module.*

*And anything with module.export the file become a module.*

**Some understanding and deconstructing code:**

var express = require('express');

* require is a global function whih takes parameter as a string ans assigning to the variable express.

var router = express.Router();

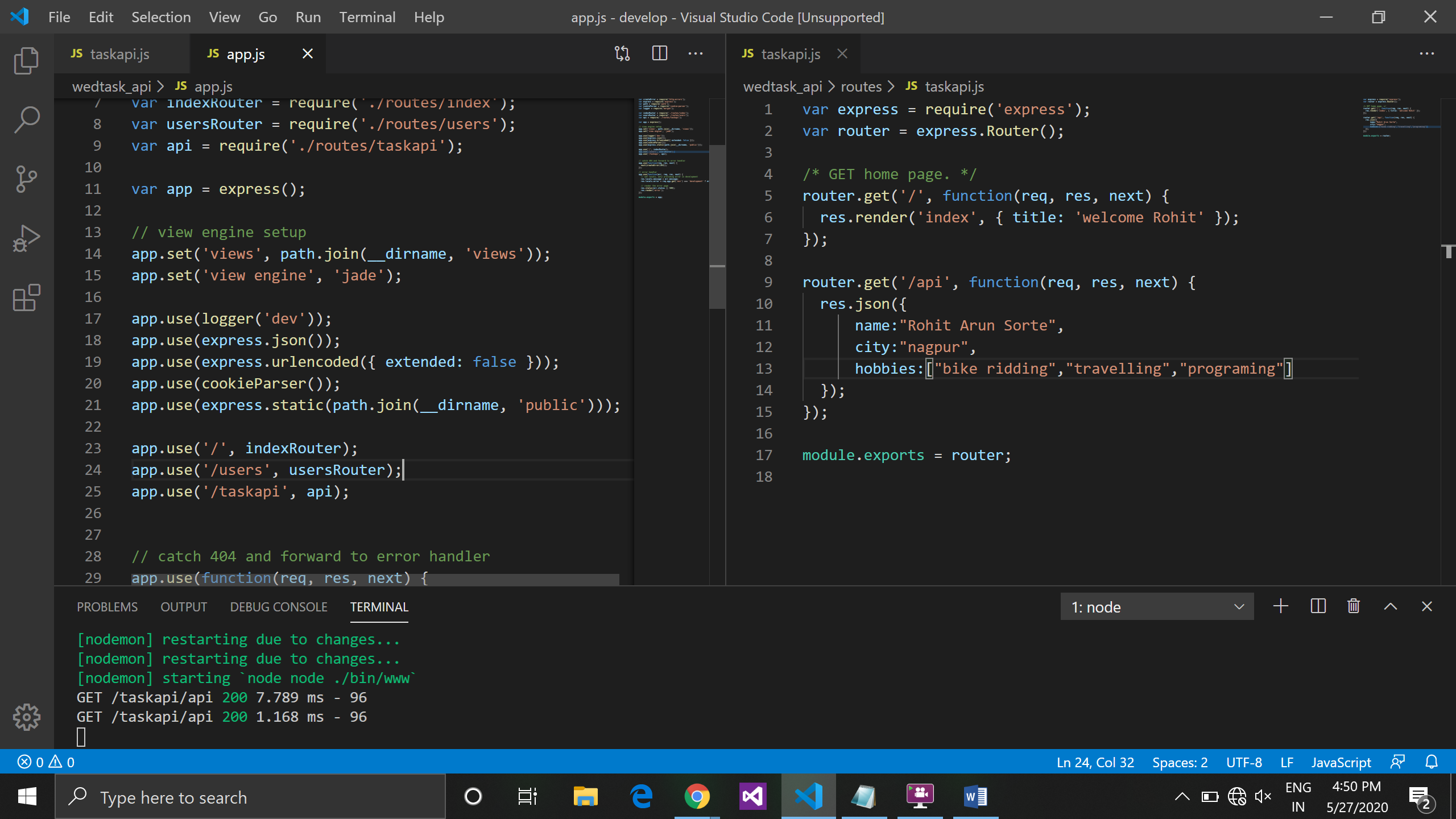
* express is a object which has a key router and assigning to variable router.

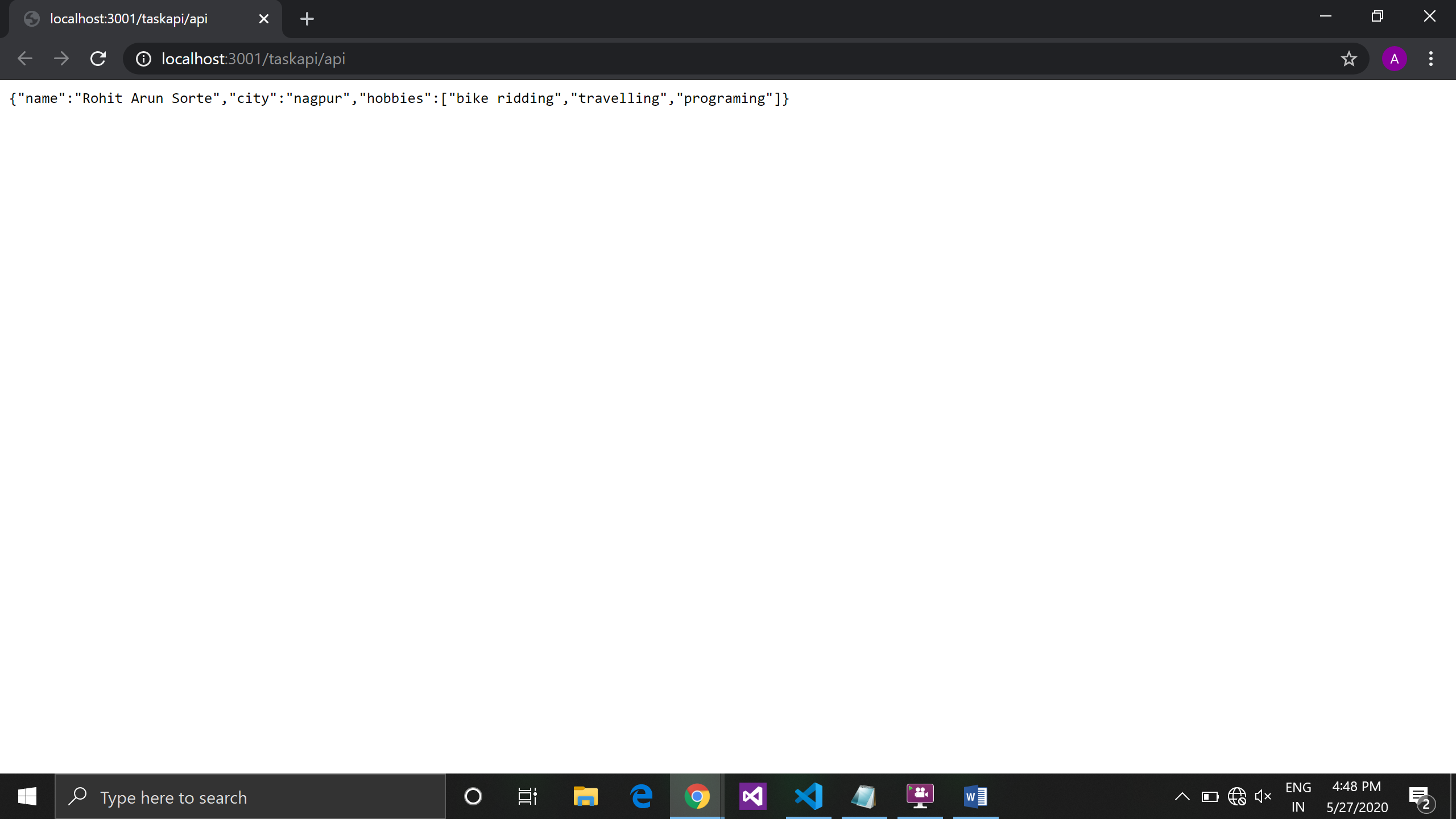
Module.export = router;

* Module is a object which has key export and which assign key as a router.

Example:

Create an express app and add one api in a new file module with JSON Data**.**





Thusday 28-05-2020

Topic:

1. Express framework
2. Built in mudules.
3. Blocking and non blocking code.

**Express framework:**

* Express is a framework its is designed for web application built in top of node using different modules.
* It combines a package to run code that is ew normally called boiler-plate code.
* Express is a command line tool and we can create application using command line.

**Built in module:**

* + Node has some built in modules and we don’t have to install those modules.
* We learned about File system module and path module.
* Fs is file system module used to handle file system in node. It is very heavily module in node
* To use fs module in code we need to import it in the file like, it is a library and fs is **object.**
* Because of its object it represent **key: value** either it could be value or function.

Const fs = require(‘fs’);

* All the operations like reading file, writing file, copy file, etc. all are do using fs module.
* Path is also a built in module which is heavily used in node used to create path structure.

Const path = require(‘path’);

* So in that we have 2 types of path i.e absolute path or relative path which is used to linking path from one to another

Absolute path:

* Absolute path is path which describe the complete location of the file form the root (full path).
* It represent the complete path of the file or folder like web address (<http://localhost:3000/api/read>)
* For e.g.
* In windows, all application stored in c drive, so if we are taking any application and search there absolute path so it gets like this c:\windows\programfiles\Nodejs\readme.md.

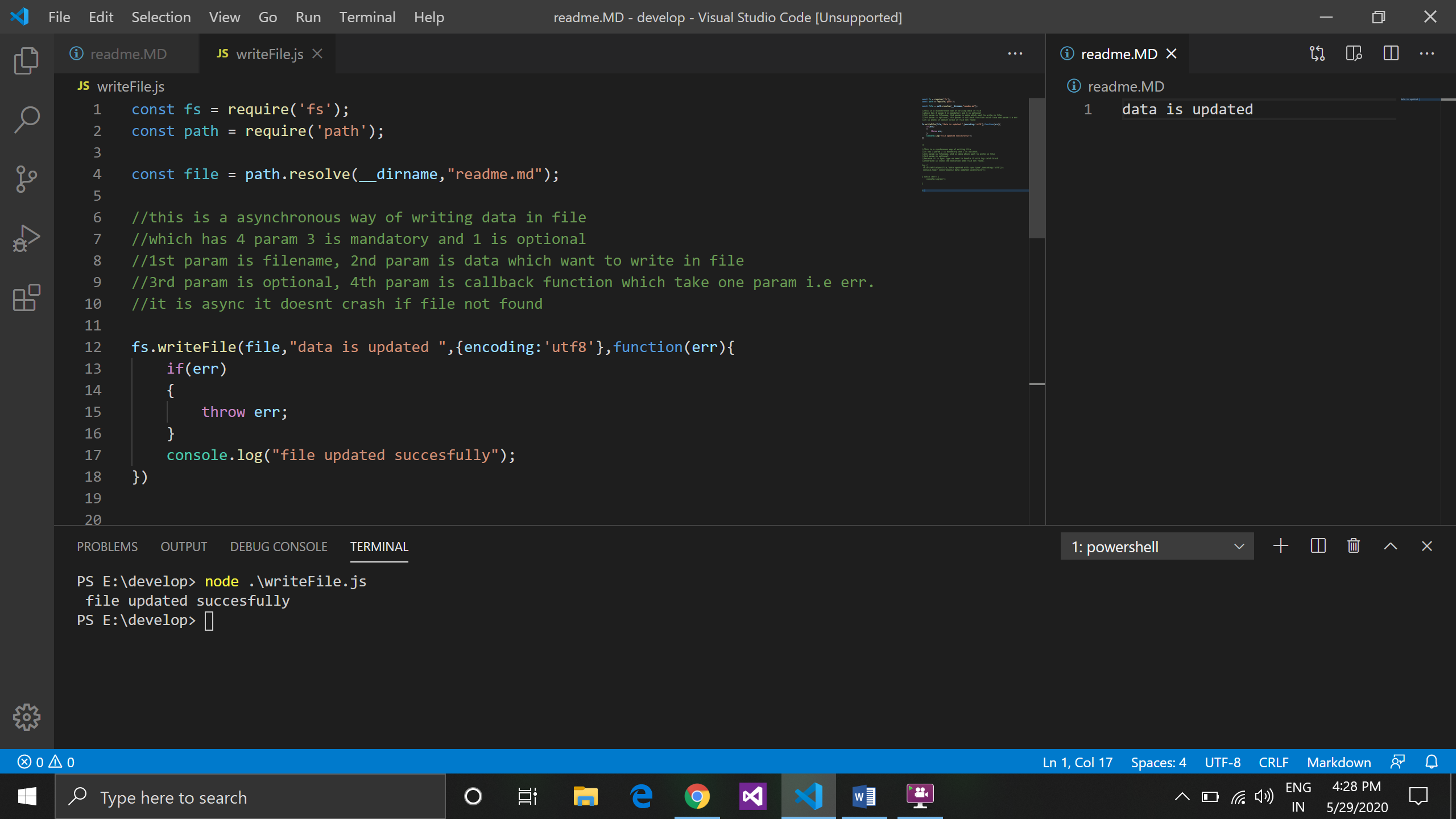
Relative path:

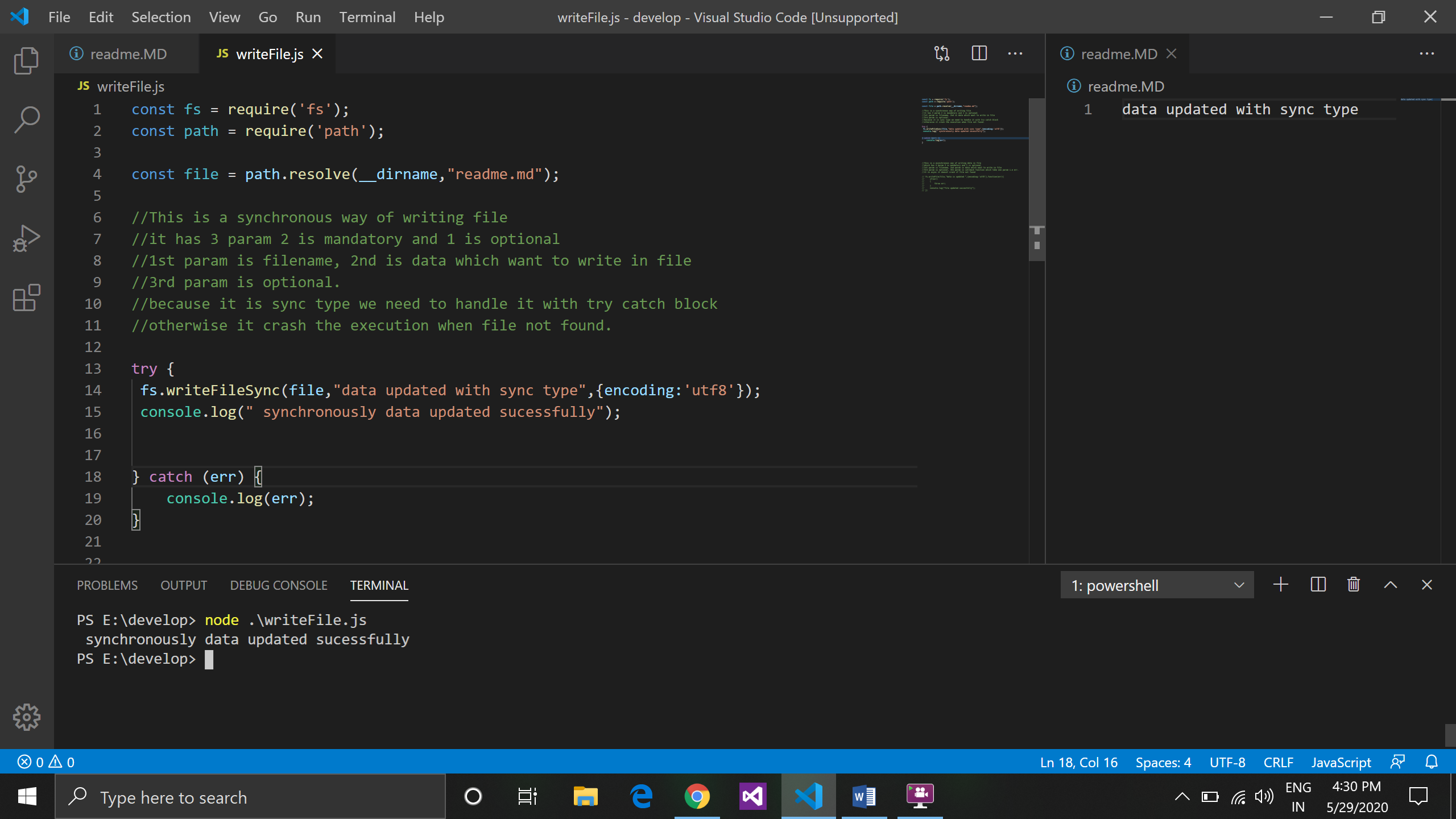
* Relative path is path that describe the location of the file related to the current location of the file or current working directory. It can be refer to the same location of the file.
* For e.g. : readme.md

If you are working in some Nodejs directory, in this directory readme.md file is available so the path should be like this > “readme.md”. so it is search in the current working directory if found it can perform operation if not found it gives err. It will not find in some other directory.

Example:

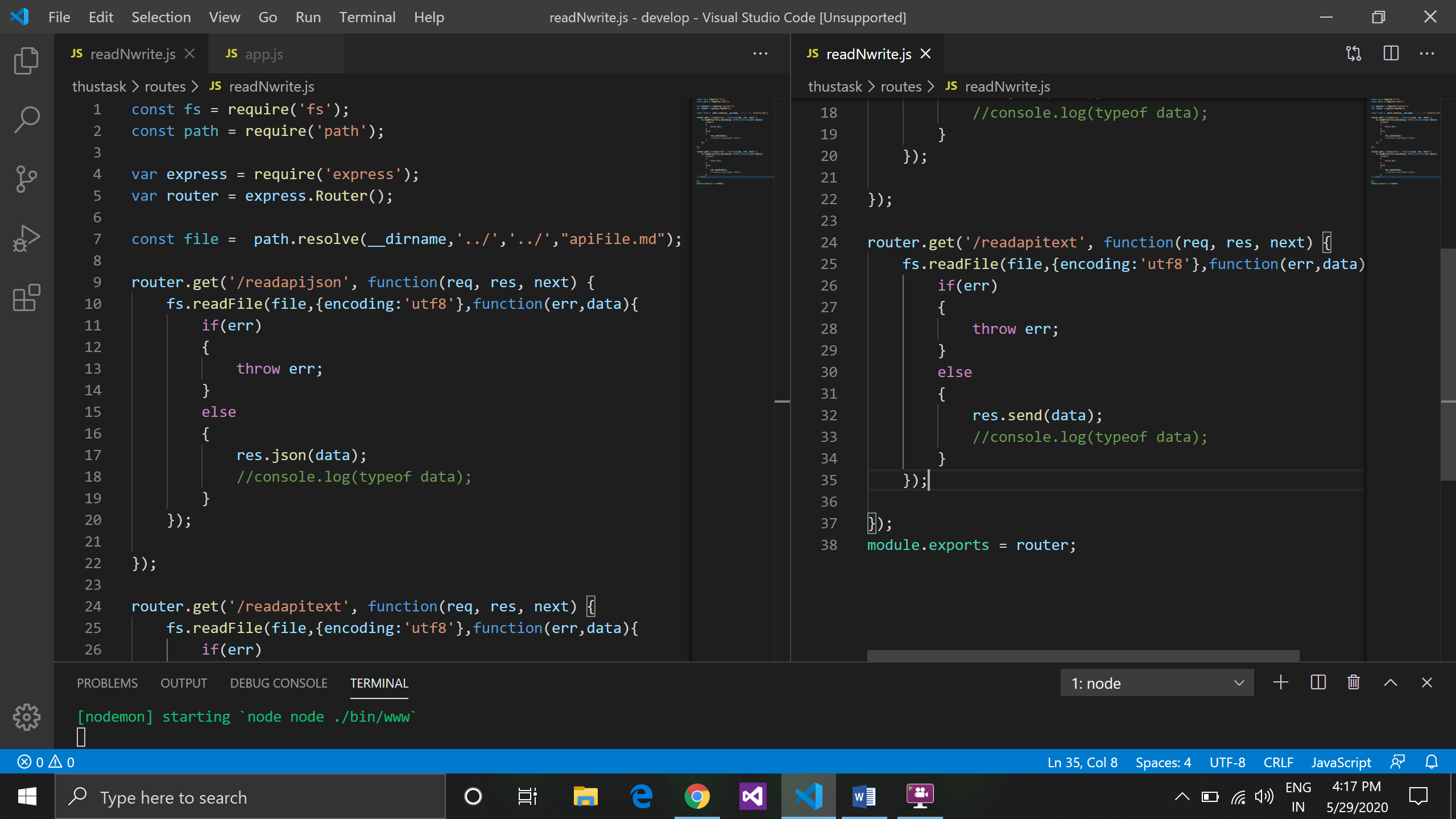
Read and write file using two different way with blocking and non-blocking difference:



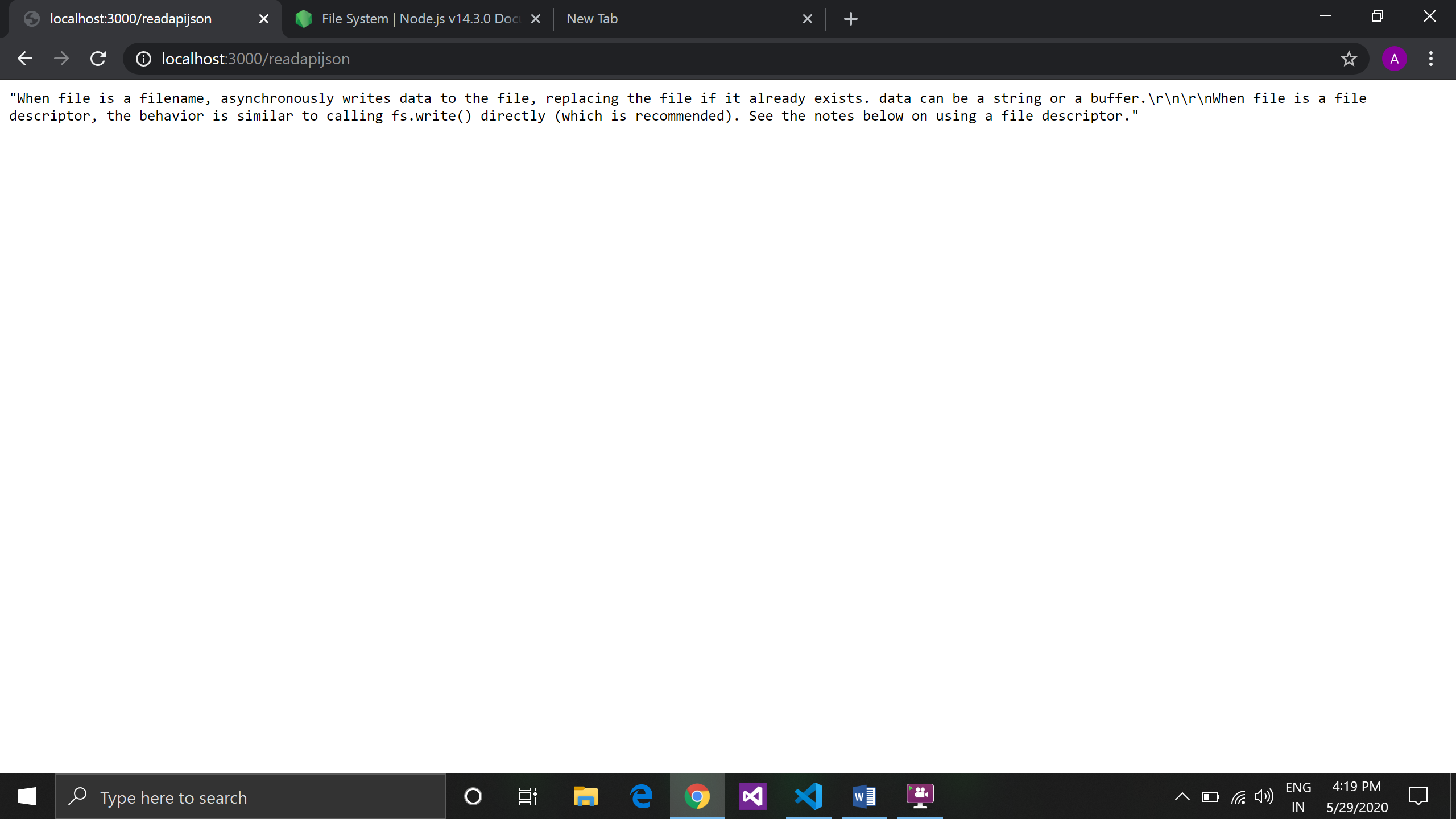


Example:

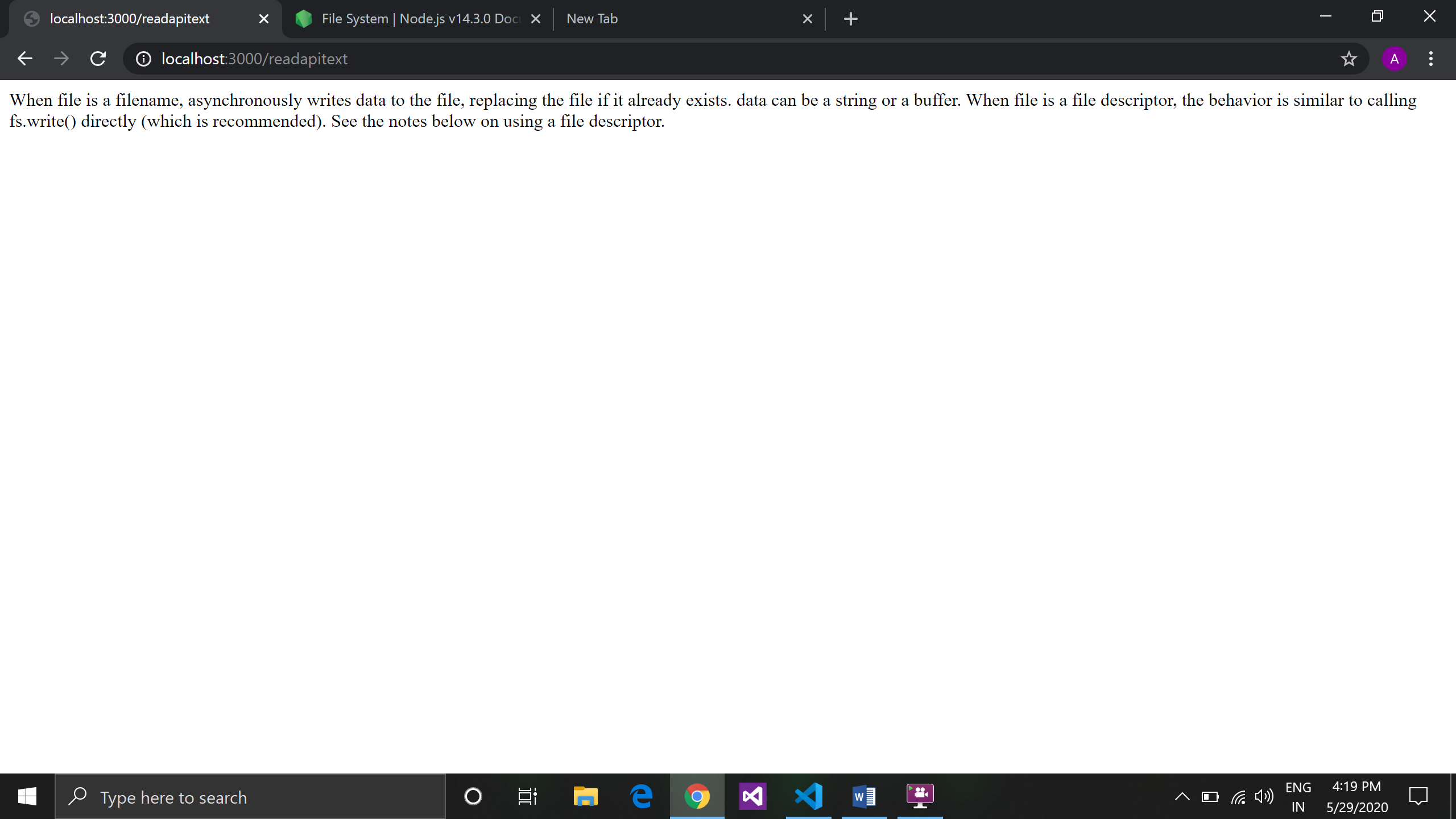
Creating an API which reads content from a file and return using API in JSON as well as text.



**Output in json format:**



**Output in Text format:**



**Friday 29-05-2020**

**Topic:**

1. Discussion on Nodejs
2. Synchronous and asynchronous blocking

* Node is runtime environment built on top of v8 engine and v8 engine created by Google.
* It is a single threaded programing Lang.
* It is a event driven
* It is asynchronous in mechanism.
* It is I/O based
* It is non-blocking
* It is works on event loop
* Node is very essential (suitable) for I/O applications.
* All the web applications are I/O applications
* Single threaded means it can utilize one thread normally cpu has two thread but it can use utilize one thread because of that it works very fast.
* Non-Blocking is First class Citizen in Node.js.

**Synchronous and blocking:**

* Java , python etc are sync they use threading that means if the first operation is not complete all the other operations are stand in queue.
* Running in main thread.

Example: bank withdrawal line queue

* **Asynchronous and non blocking:**
* The operations are not waiting in a queue .it just handover to background os thread using event loop.
* Node is async, whenever u make a request to node it just take a request and send it to the background thread and passes request using event loop. Event loop handle the request internally.
* Example: chai tapri

why non-blocking code and sync job is done in JavaScript.

* In Reverse pyramid non-blocking code in JavaScript doesn’t gets blocked and run the application the entire code will go to the event loop. This will run in asynchronous way.

**Node utilizes I/O some I/O devices are**

1. NIC
2. Scanner
3. Printer
4. Keyboard/ mouse
5. Hard drive
6. Speaker/microphones etc.

When we make a web request

First request goes to browser then it will send to NIC send request to router the it send to network provider and it goes on server.

Some i/o based applications are :

1. Facebook
2. What’s app
3. Telegram

**Saturday 30-05-2020**

**Topic:**

1. Learned some built in function and done practice on it.
2. Event Loop.
3. Event loop phases.

Some built in functions are:

SetTimeout ():

setTimeout ( function()

{

console.log (1)

}, 1000);

It will print the value in 1 second.

* SetTimeout run only once.
* Whenever we write in setTimeout (is used to delaying the application) it will say that the should not run in the give time period on run after the time interval for e.g in above code time interval is 1000 millisecond that means the thread can run after 1 sec only. It can control the execution of thread.

Set Interval () :

As we know setTimeout runs only once but setInterval runs multiple times.

setInterval ( function()

{

console.log (1)

}, 2000);

* Its keep on running its callback.

**Event loop:**

* It keeps on repeating itself whenever there is execution happens.
* It has 6 phases every iteration goes to all the 6 phases and then event loop restart again
* It depending upon the the callback type to go in which phase in event loop

The 6 phases are:s

1. Timer
2. Pending callbacks
3. Idle / prepare
4. Poll
5. Check
6. Closing callback

* It performed in queue manner and follows first in first out.
* Timer: whenever you write something in set timeout or set interval so this timeout callback function will added to the timer queue and event loop will wait for the timer expired. Once it is expired eventloop get picked and execute it.
* Pending callback: Which is used by system e.g. TCP connection callback will registered here and event loop run all the callback one by one.
* Idle/prepare: In this event loop do nothing it is used in internal use.
* Poll: Poll its very imp where most operation can performed it checks i/o completed or not if yes it run the callback.
* Check: Whenever there is a any function or method is added it goes In setImmediate it will added in check queue
* Closing callback: Once it is complete it will goes to closing phase.

*Note: this all normally call as microtask*

* In last we also learn about MicroTask which is advaanced

- nextTick : - Very less use

this is a part of v8 Engine.

HTTP Create Server:

const http = require('http');

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

res.end("OK");

});

server.listen(4000);