Node.js

Node.js is a JavaScript runtime environment, it allows you to run the JavaScript in the backend, you can perform various backend operations through node.js like accessing files, accessing os, accessing database, middleware, server and etc.

Earlier Javascript was used only for the front-end and it was understood by browser, but now with the help of node.js you can run the javascript without browser.

Node.js uses some libraries to run the JavaScript code, these libraries are called as node\_modules which provides the platform to execute JavaScript.

Note: You can run the JavaScript file directly using node.js, but if you want to run in browser, then you must include JavaScript in the HTML file and run the HTML.

Benefits

* Developers don’t have to switch from one language to another language when they want to communicate from the client to server applications i..e, front-end to back-end
* You don’t need ay interpreter because JavaScript can be executed directly from the node.js server
* Enables developers to write applications in one language for both front-end and backend
* JSON is the command format used to interchange the data between the front-end and backend, It is the native language for the JavaScript and you don’t need any kind of parsers to understand JSON
* JavaScript is the language used in various NoSQL databases like MongoDB, CoucheDB, interacting with them is much easier if you use JavaScript at the backend.

Event Loop & Asynchronous IO (non-blocking IO) mechanism

Node.js uses this mechanism at the backend which is very similar to browser to handle IO operations without blocking, because the IO operations are done by callback functions

Suppose your IO operation is synchronous

some statements; // 1st line

query(“select \* from emp”); // 2nd line

some statements; // 3rd line

The 3rd line is not executed until the query() execution completes, here the query() waits of the database to return the result, as it is interacting with the db, So here 3rd line is blocked, it is a blocked IO operation

In Non-blocking IO operation the tasks wouldn’t be blocked, because it is asynchronous

some statements; // 1st line

query(callback\_function); // 2nd line, callback would have the db interaction

some statements; // 3rd line

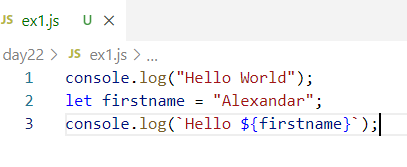
The 3rd line is not blocked, it will be executed after the query(), but callback is asynchronous and it might be executed later

Here node.js uses a single thread model and event loop will add the task to the event queue and pushes the task to the callstack for execution, if there’s a callback in any of the task, it will be called later.

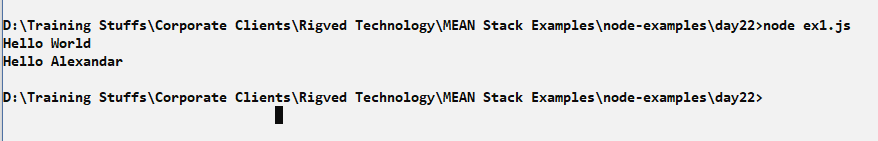
Note: Node.js supports all the new features of JavaScript like let, const, arrow, template string, padStart() padEnd(), rest, spread operators and so on.

Note: Since node.js runs the JavaScript at the backend, it doesn’t support some of the inbuilt objects we have in JavaScript that works in browser like document, sessionStorage, localStorage, alert()

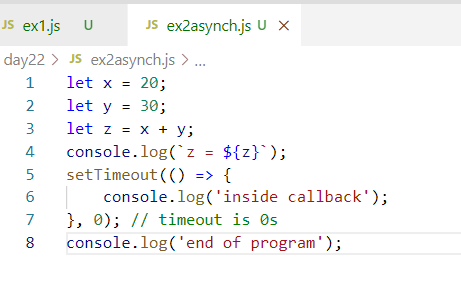
ex1.js



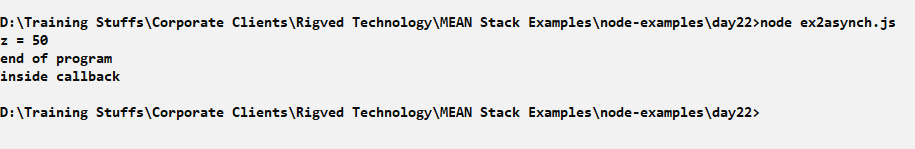
Output:



ex2async.js



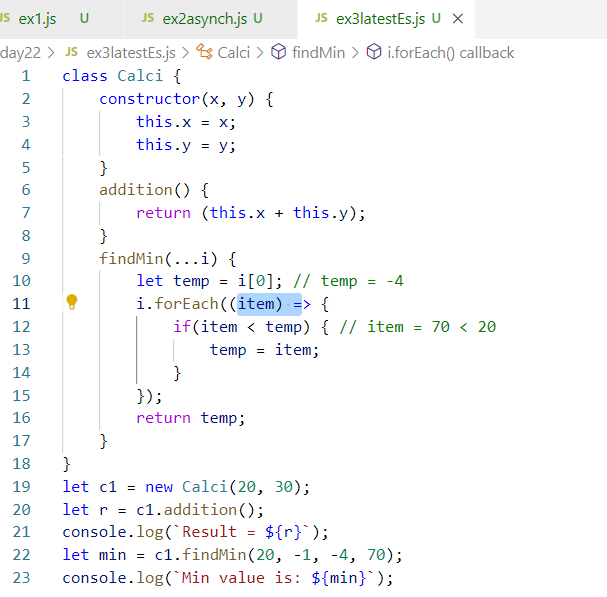
Output:



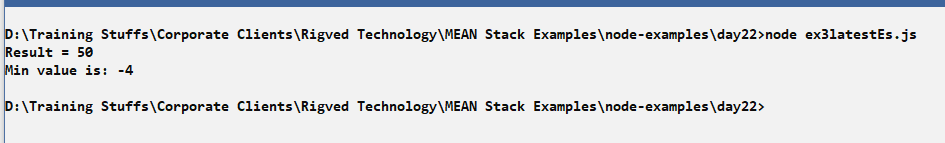
Here though setTimeout() is taking 0seconds the callback is added to the event queue after the console.log(‘end of the program’), hence its executed after end of the program

Using some new features of JavaScript

ex3latestEs.js



Output:



Node.js Modules

Modules are Javascript functions which are reusable and can be called from other Javascript functions.

There are mainly 3 types of modules

1. Core Modules: These are inbuilt modules that you can use in node.js by importing

Ex: http, fs, os, process, url

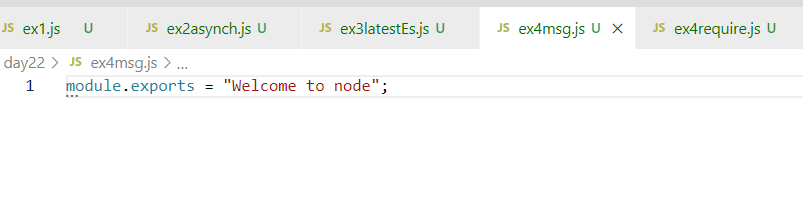
1. Local Modules: These are modules which you create and use in the program
2. Third party Modules: These are modules that are downloaded from the internet and can be used in the program

Ex: express, typescript, mongodb, bootstrap and etc

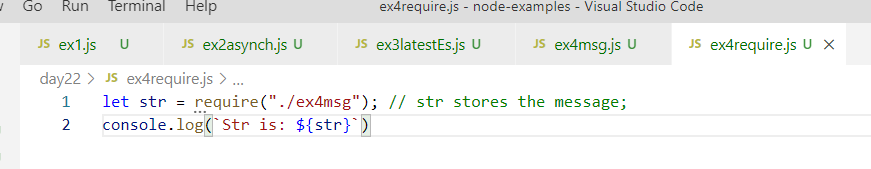
Local Modules:

These are the functionalities that can be created in a JavaScript file and import in another JavaScript file

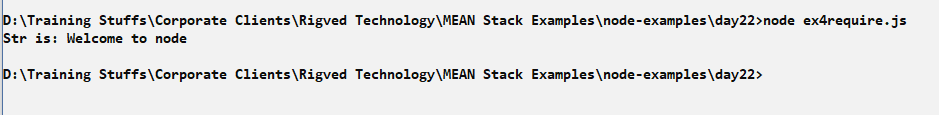
ex4msg.js



ex4require.js

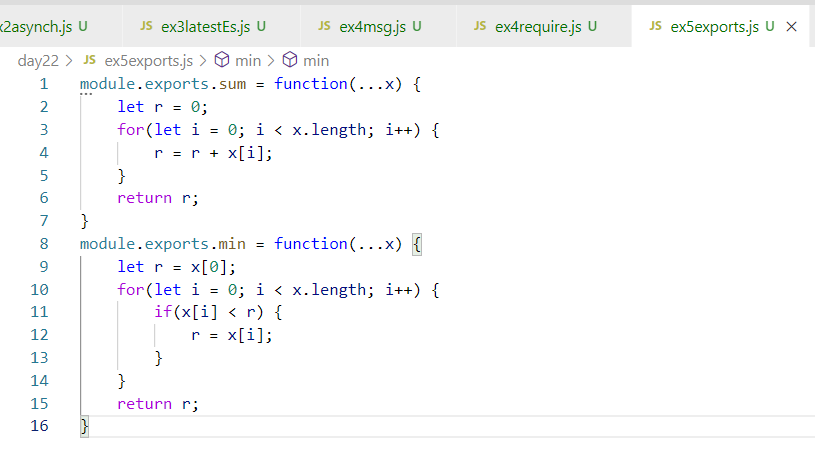


Output:

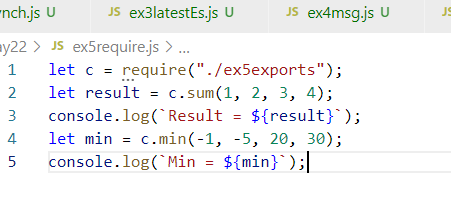


Creating multiple exports in the module with some name

ex5exports.js



ex5require.js



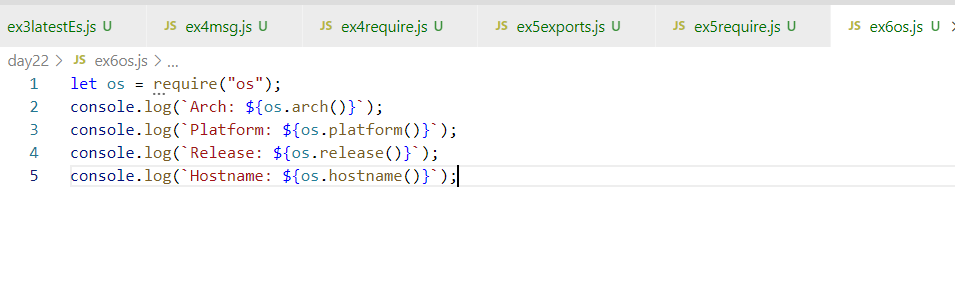
The above modules are local modules, which we created and imported, but we can also use lot of inbuilt modules & third party modules by downloading from the internet.

Using Inbuilt modules like os, fs, http, util, queryString and so on.

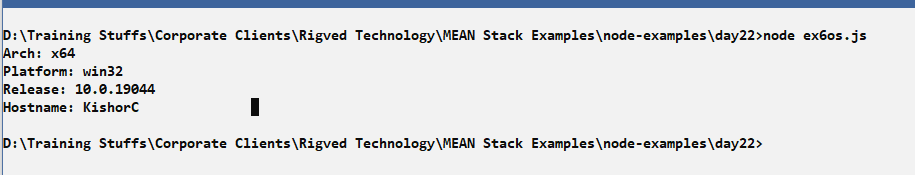
OS Module:

You get OS related informations like arch(), platform(), release(), type(), freemem(), totalmem() and so on.

ex6os.js



Output:



Third-party modules

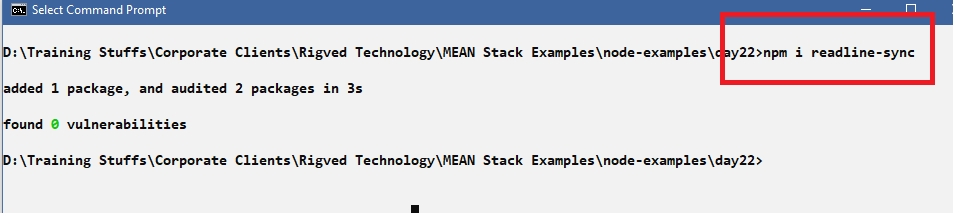
These are the modules available in the internet that need to be downloaded using npm

When you use npm a library will be downloaded and package.json file is the configuration file that will have the entry of the library downloaded.

Some of third party library are: readline-sync, express, typescript and so on.

readline-sync: It is used to take input from the keyboard, it is synchronous means all the functions are synchronous

Installing readline-sync

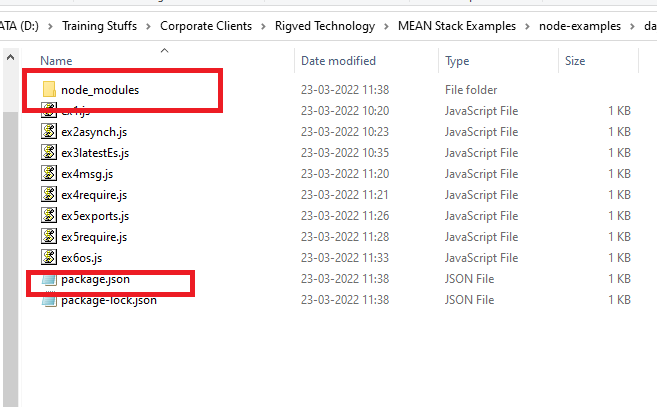


You will see node\_modules & package.json file in the folder that downloaded readline-sync.

node\_modules: Has the realine-sync library and their dependent libraries

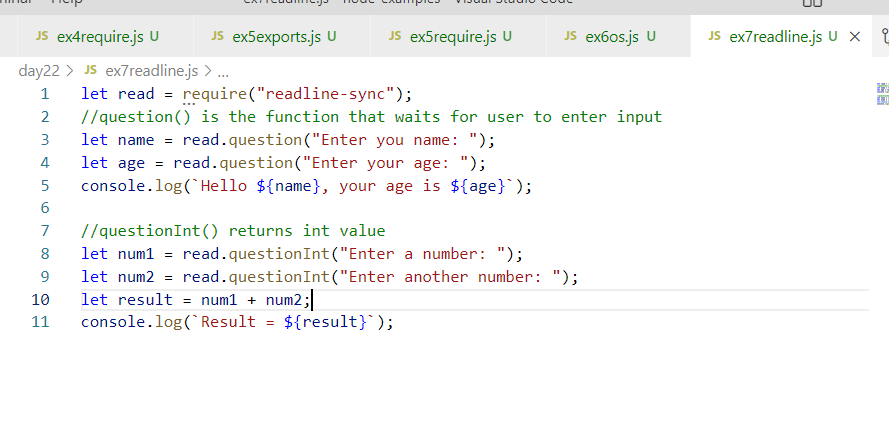
package.json: It is a configuration file in node.js this will have the entry of the library downloaded

package-lock.json: It is an auto-generated file you can ignore

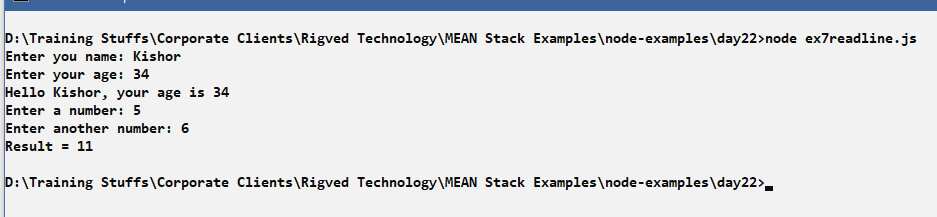


Now you can import the readline-sync library and take input from the keyboard

ex7readline.js



Output:



File handling

You can read/write data from/to file, it can be any kind of data like binary data or character data, in most of the case you also use json files which will have character data(text data). Node.js provides modules to perform read/write operations in both synchronous and asynchronous away.

fs module: It is an inbuilt module used to handle file data, it provides inbuilt functions to read / write data, it has functions as below

readFileSync: Reads the file in a synchronous mode

writeFileSync: Writes the file in a synchronous mode

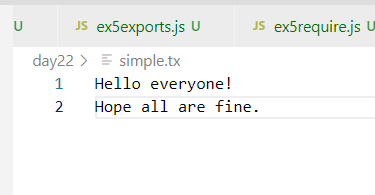
readFile: Reads the file in an asynchronous mode

writeFile: Writes the file in an asynchronous mode.

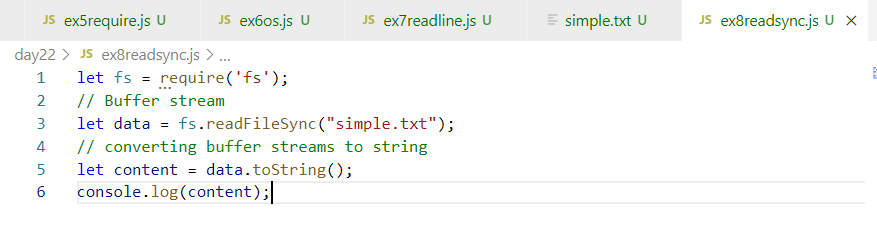
Read operation:

Create one text file that will have some data

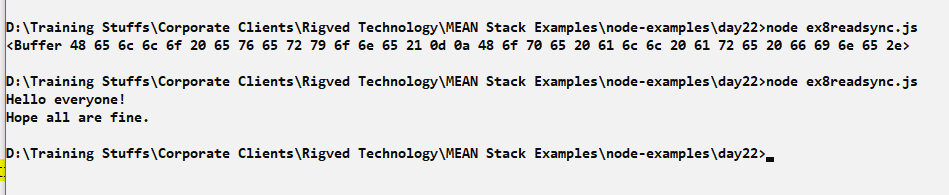
simple.txt



ex8readsync.js



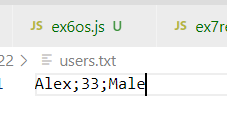
Output:



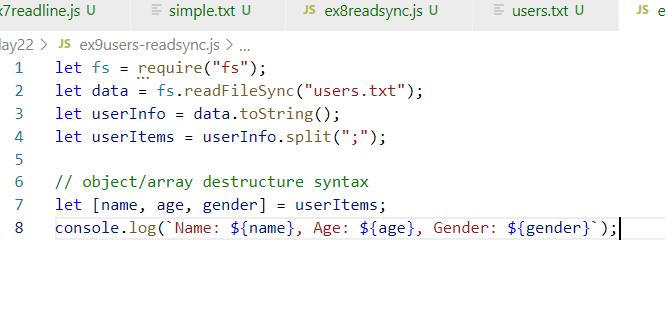
Reading the file that will have user information

Create a file that will have user data separate by a delimiter

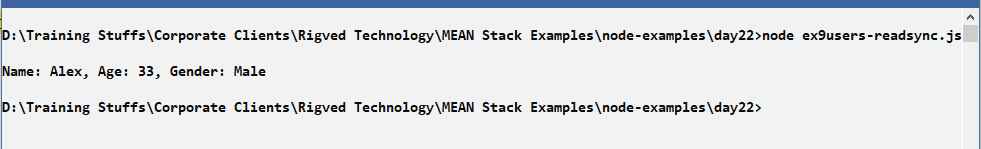
users.txt



ex9users-readsync.js



Output:



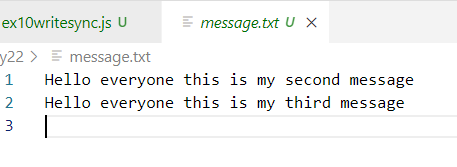
Writing file synchronously

You can use writeFileSync(), it takes 3 arguments, the 3rd argument is an object which will have a flag property that says append with ‘a+’.

ex10writesync.js



Output:



You can read and write json files

JSON files are a text files that can be used for configuration, database and so on, you can store some data and you can access the JSON data, JSON is native to JavaScript it can easily understand JSON and Javascript has some functions to convert JavaScript to JSON and vice versa

JSON.parse(jsonStringContent): It takes the JSON and converts to JavaScript object

JSON.stringify(javascriptObject): It takes the JavaScript object & converts to JSON

Note: We can’t store JavaScript objects’ in a file, we need to store them in a text format, JSON is the text format for JavaScript object

How the JSON data looks

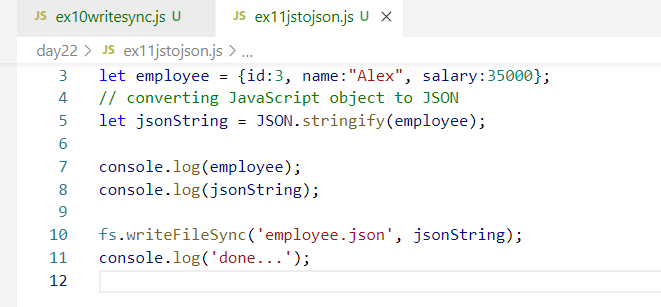
{“key”:value, “key:value”,…}, {“key”:value, “key”:value,…}

JSON file can also have JSON array which is a multiple JSON wrapped in []

[  
 {“key”:value, “key:value”,…},

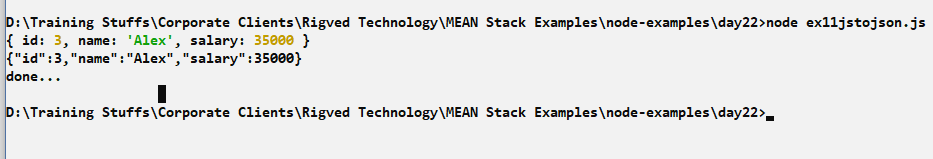
{“key”:value, “key”:value,…}  
]

Converting JavaScript object to JSON

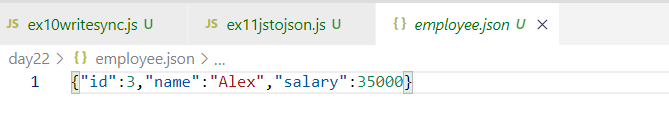


You can’t write JavaScript object into any file, you need to convert to JSON string and write to a JSON file

Output:

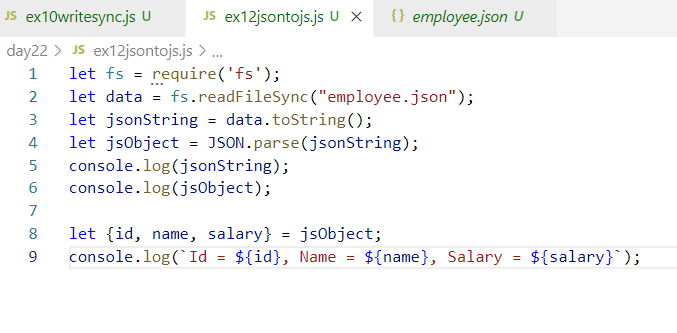


employee.json

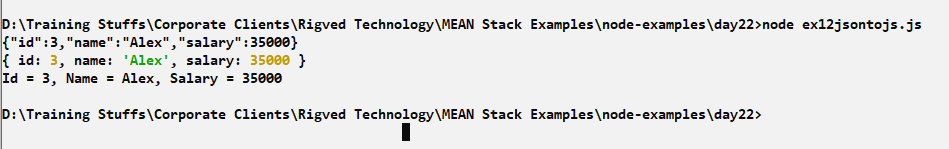


Note: Multiple JSON files need to be separated by , comma

Reading the JSON and converting to JavaScript object



Output:



Activity:

1. Try out all the above activities
2. Take input from the keyboard and read data like id, name & salary from keyboard and store this in a JSON file
3. Same above example do it for multiple records i.e., you must able to store multiple JSON data in the JSON file, but each data should be separated by comma
4. Continue the above example, and read the JSON file having multiple data and convert to Javascript object and print all the information in the console

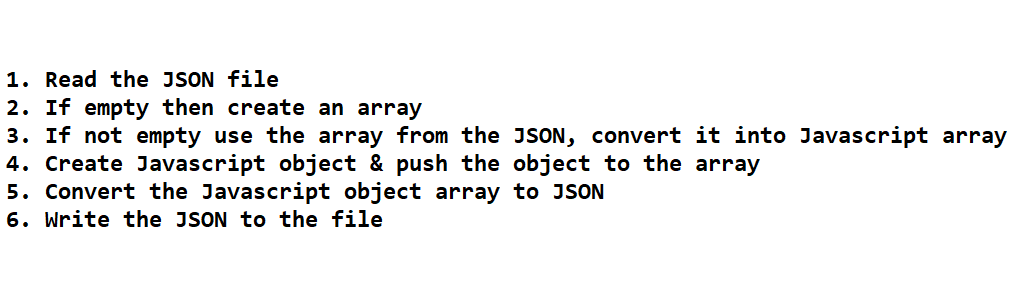
Hint: Multiple data/json/javascript object is represented by array

Solution:

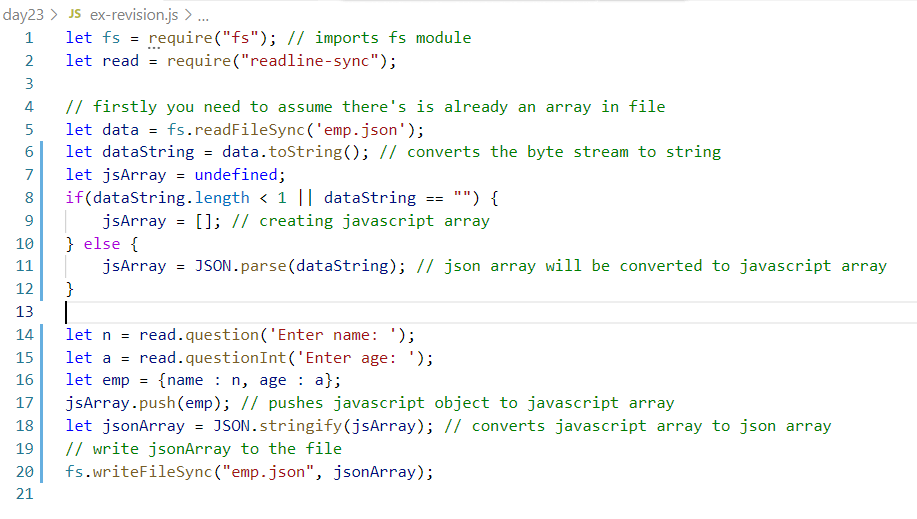
You need to store array of JSON in the file i.e.,

[{…}, {..}, {….}]

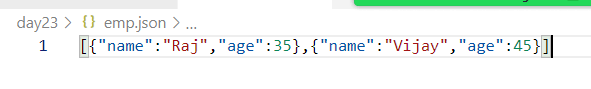
Means you need to write the json array not just json.



Solution:



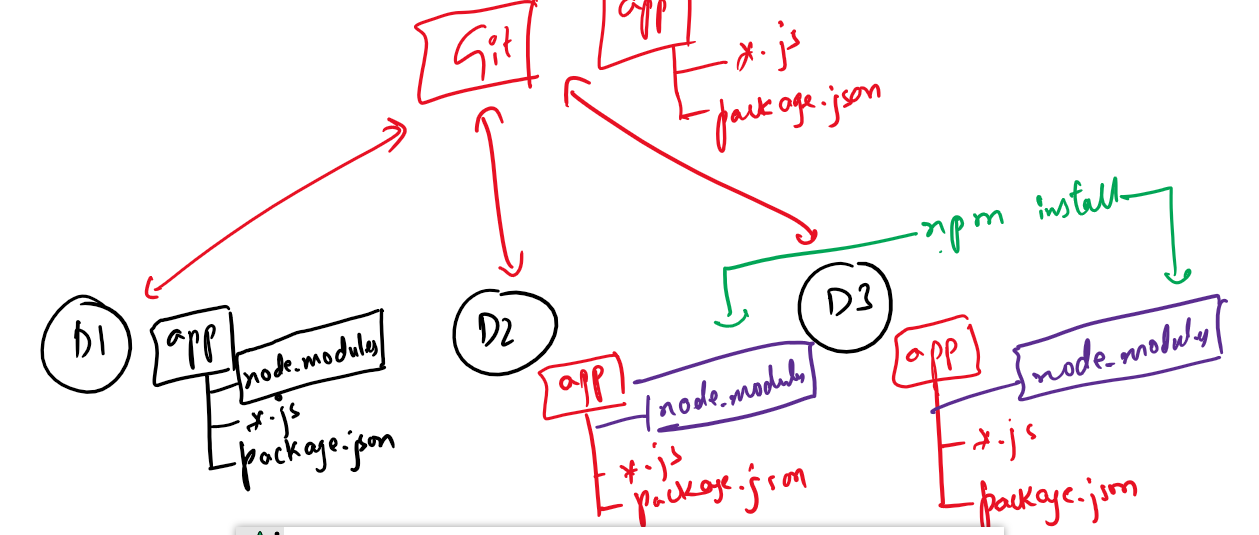
emp.json looks like below:



package.json:

It is a configuration file used in node.js,

* It can have dependencies entry that shows the libraries the node has downloaded in your working directory
* It can have commands that can be used with npm to start/run the application, test the application, build the application
* It can have even the entry point file, so that when you start the application the node can execute the entry point file
* It is used to re-download the dependencies in the working directory when developers share the application to multiple developers
* npm install would install all the node modules in the working directory.



Here D1, D2, D3 are developers who are working in the same project and they can download all the necessary libraries required for the application using npm install, which looks package.json and downloads the modules in node\_modules, so that you don’t have to push the node\_modules to the GIT

also, if any new library is downloaded in anyone’s machine its entry will be there in package.json, so when it is pushed to the GIT, other developers would pull the updated package.json, then enter npm install to get the new library in their working directory.

Node.js web application

It is used to write client & server programs where client sends the request & server generates the response, in order to develop web applications you need a server that can launch your application, node.js provides an inbuilt module called http module which can be used to develop web applications & create servers

Http module

It is one of the inbuilt module in the Node.js allows us to create server, handle request, generate response.

How to use HTTP module

let http = require(‘http’); // this references to http module, from this you can handle request, start server and so on.

How to handle request and start the server

*http  
 .createServer(callback)  
.listen(port, callback);* // all the 3lines can be written in a single line also as below

(OR)

*http.createServer(callback).listen(port, callback);*

Here

http is a module to create web application.

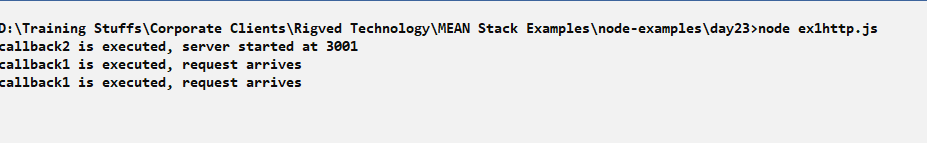
createServer: It is an inbuilt function inside http module to handle the request & generate the response, it takes a callback which is executed whenever the request comes to the server, it returns a server instance on which you can call listen

listen: It is also an inbuilt function that can specify the port number of the server in the 1st argument, the 2nd argument is a callback that is executed when the server starts

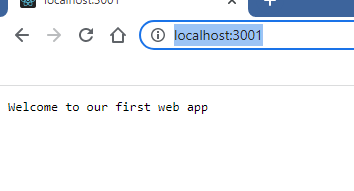
ex1http.js



Output in node console



Output in browser console

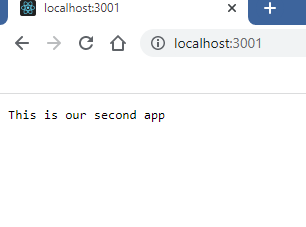


Note: You can pass the callbacks without creating by name in the function argument itself

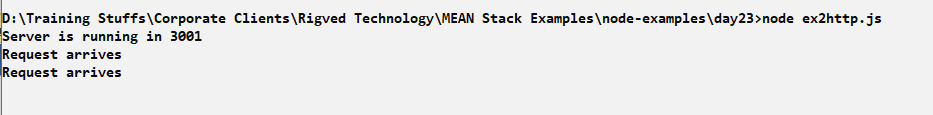
ex2http.js



Output in browser:



Output in console:



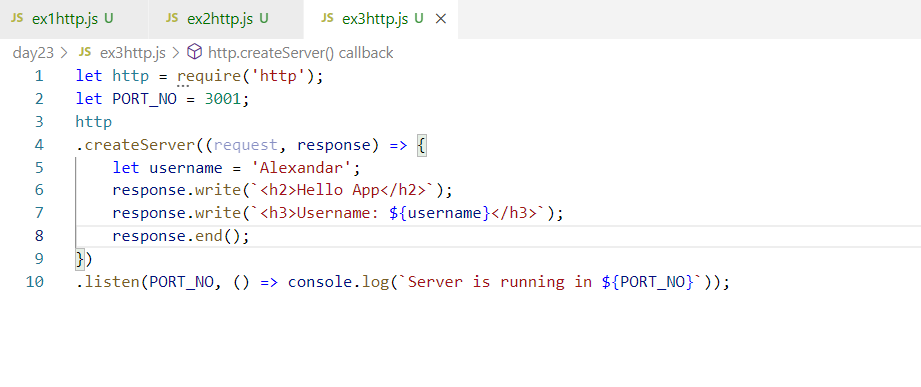
Writing HTML content in the response

By default the response content will be in text format, if you embed HTML code then it will be considered still as Text, to specify the text is an HTML content, you need to set response content using writeHead() function.

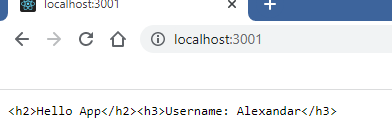
response.writeHead(status, {key : value});

Here status can be any HTTP status codes like 200 for success 404 for not found or errors, key: value are the response header properties like content-type, content-length and so o.

Without using writeHead() HTML code will not be recognized, as they are treated as text.



Since the content are treated as text, you will see html tags in the browser.



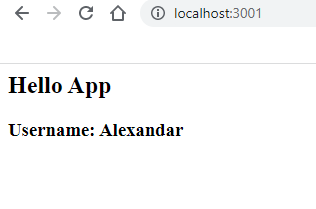
To avoid this we need to set the response header

ex3http.js



Because we have used response.writeHead(statusCode, headerProperties), we could able to see HTML content

Output:



Reading the data from the client

Client can send the data through query parameter or form body, we can extract those data at the server side

Query parameters: These are the data which are sent in the URL of the browser, query parameters will be in key value pairs after the url?, the data will be read at the server side

ie., <http://ip:port/path?key=value&key=value&> …

Form body: These are the data which user can enter and submit to the server so that server can read those data

How to read query parameter

You need to read the url using one url property of the request object i.e., request.url, this gives the resource url the user is trying to access, if the url has query parameters then you can read that too but for that you need to use url module and parse that query parmeter.

let url = require(‘url’);

If you read url through request as below

let reqURL = request.url; // if the reqURL is somePath?name=Alex&age=35

Then you can extract the query parameters as a javascript object through parse method of url module

i.e.,

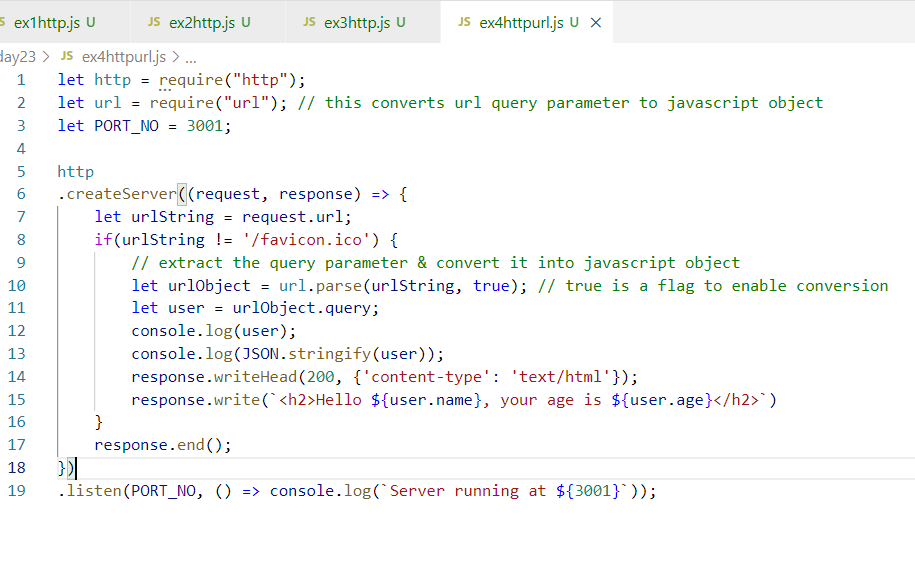
let urlString = url.parse(reqURL); // converts the url into javascript query object, then you can access that query property to get the query parameters

let obj = urlString.query;

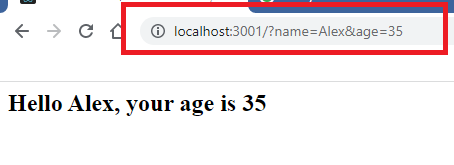
Note: Whenever you send a request to the server there will be two request sent one to the server and another to /favicon.ico, hence we must write an if condition to only get the request to the server.

Note: This problem you don’t face in other modules like express module

ex4httpurl.js



Output:



Activity:

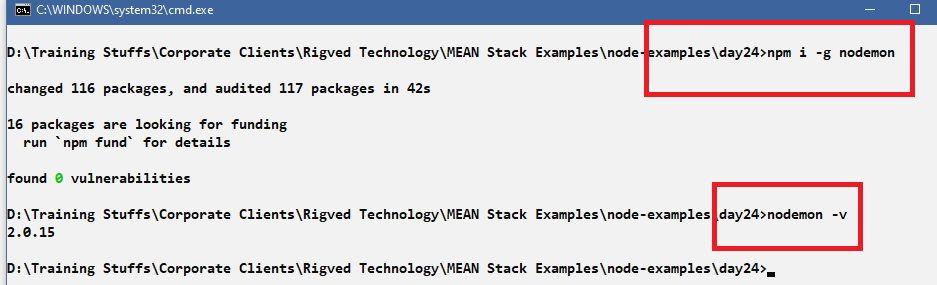
1. Try the above examples
2. Pass the query parameter in the URL and store that as a JSON data in the JSON file, it must retain the old data without erasing the old data, you can store name & age in json format.

Nodemon: It is a library used to auto-detect the changes in the code, instead of node you can use nodemon command to run the script, which can auto-detect the changes, so that you don’t have to stop & re launch your application

You need to install the nodemon library, you can install it globally so that it will be available in other directories also

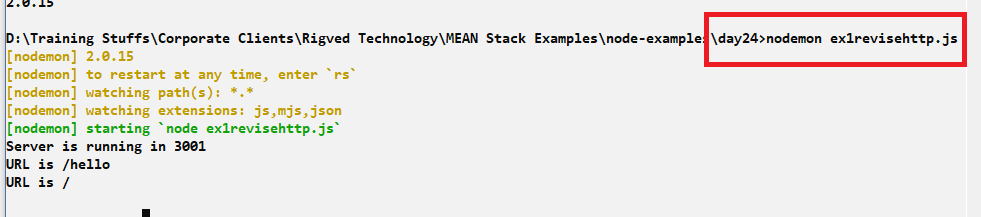
>> npm i -g nodemon

Installing nodemon

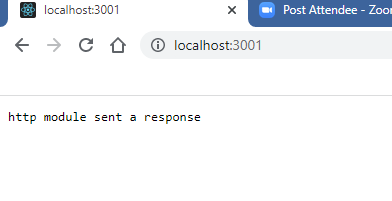


Now onwards we can use nodemon to run the script instead of node, but whenever you want run a different script you need to stop nodemon and run that script

Running the script



Output:



Do some changes in the code, you can see nodemon auto-detects it, example we will import the url module and read the query parameter



Output:

