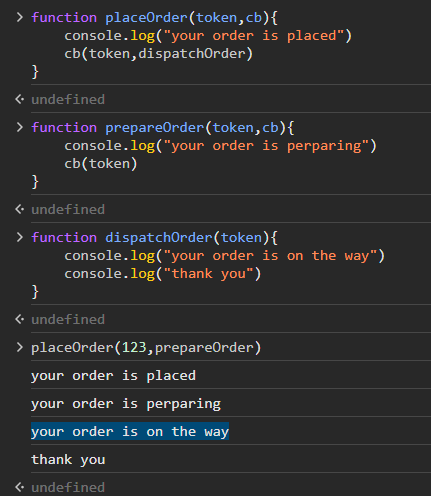
1. Why there is callback function in js
2. Example of callback function



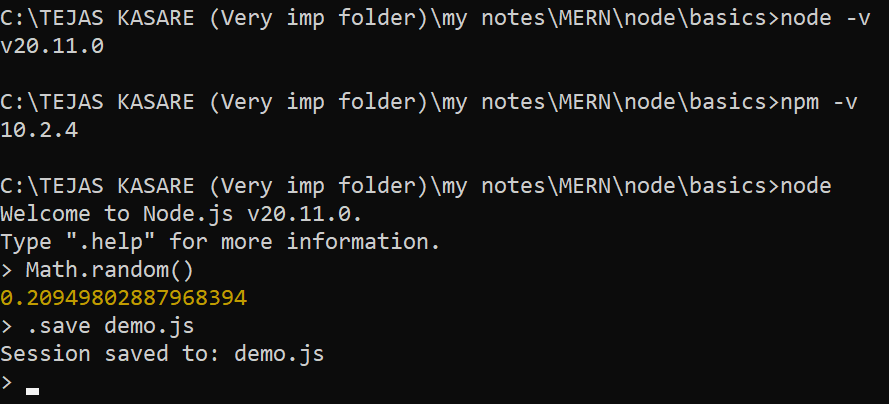
1. REPL functionality of JS

> Math.random() => Read and Evaluate

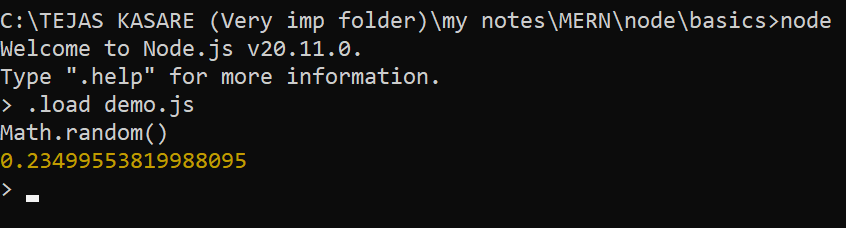
> 0.979437508321547 =>Print

> => Loop

1. Study Promises in JS
2. Install node LTS
3. Saving terminal code into files using node



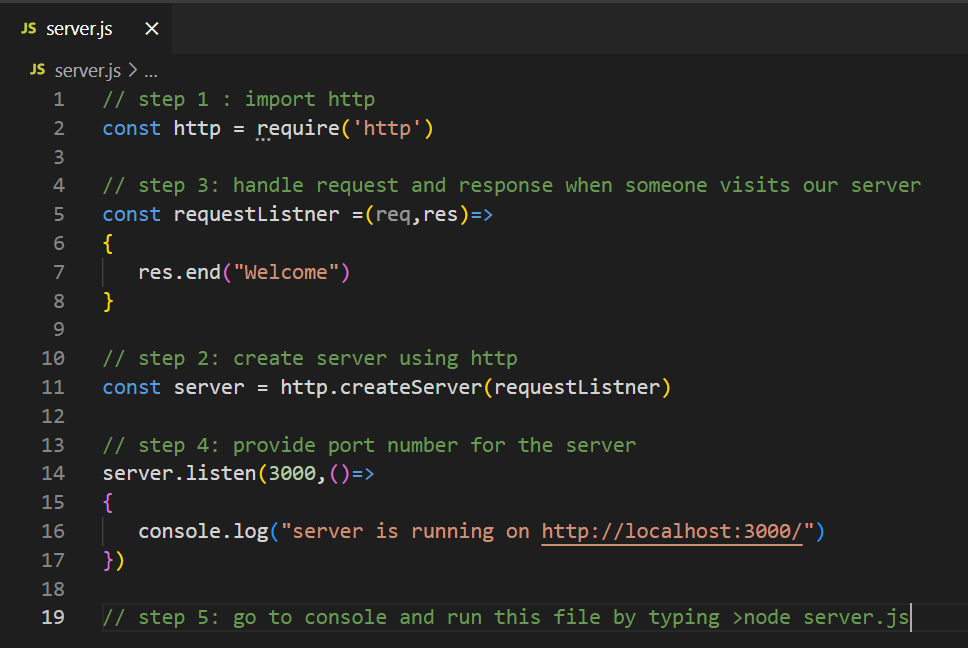
1. Lading same file



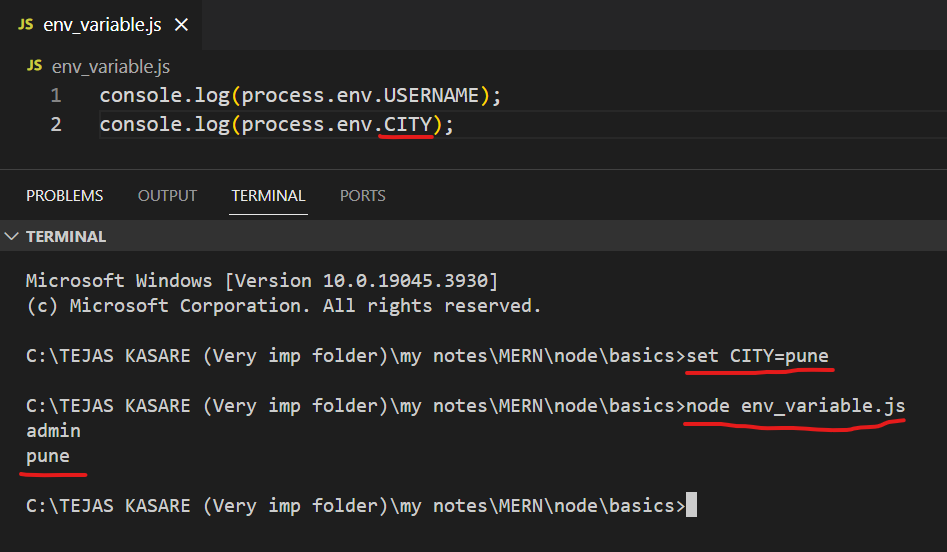
1. Creating simple js file and executing it via node



1. Creating server using node

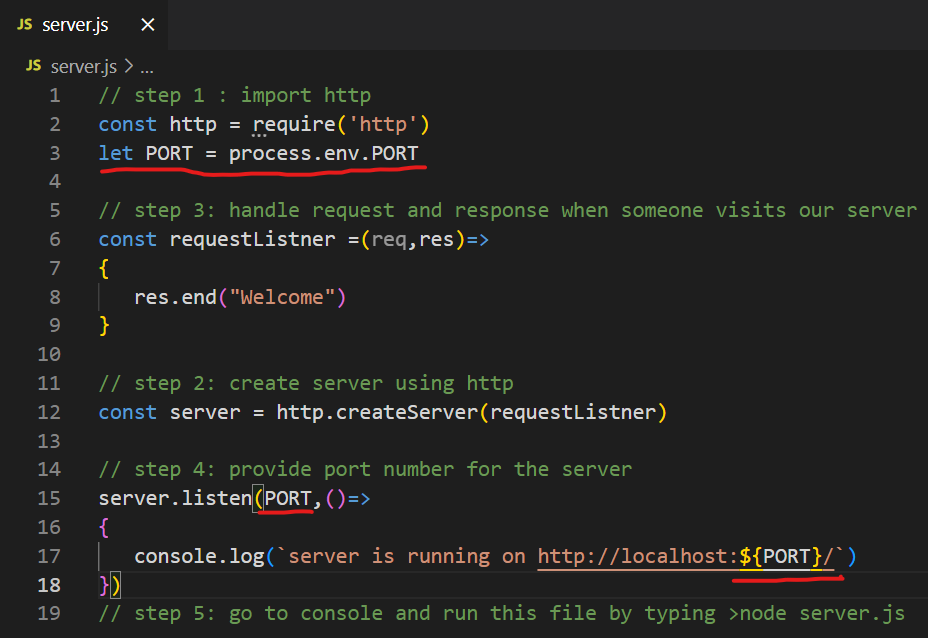


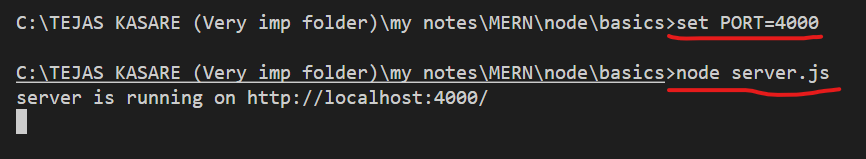
1. Storing and accesing environmental variable



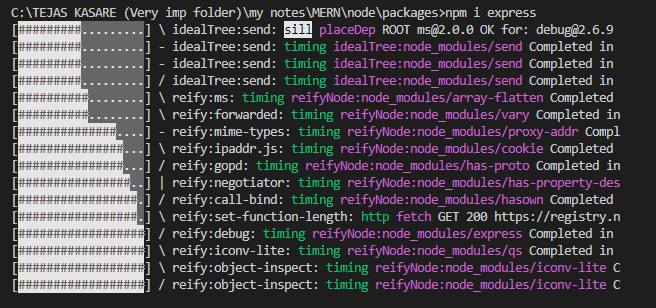
*VIMP : open cmd not powershell in VS code*

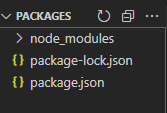
1. Setting port number dynamically from environmental variable





1. Checking npm version and installing packages
   1. Package : any js file or a folder that consist of js file(s) is called as package
   2. > npm –v
   3. > npm i package\_name =>install package locally (only for selected application)
   4. > npm i –g package\_name => install package globally
   5. > npm i –D package\_name => install for development only (not for production)
2. Package.json
   1. Let we have installed express package by > npm i express





* 1. When we install any package using npm command, it will automatically generates package.json file which has registration of that package. Ex express



* 1. Package-lock.json => consist of sub dependencies of package installed in above command

1. Importance package.json
   1. Delete node\_modules folder and go to terminal and execute
      1. > npm i
   2. By doing this, npm will go through our package.json file and install all required packages mentioned in package.json file
2. Creating package.json manually

Delete (node\_modules, package.json, package-lock.json) from current folder

Open terminal and type

> npm i

After this, provide answer to all qustions

Done

If you don’t want to go through all above question, then just execute

> npm i --yes

1. Making production ready build
   1. > npm i –production
   2. By doing this, all develeopement related packages will not get added in node\_modules folder
   3. develeopement related packages means the packages that we installed with

npm i –D package\_name

ex : npm i –D nodemon

1. argument keyword
   1. open browser console and execute :

First :

function sum(){

console.log(arguments)

}

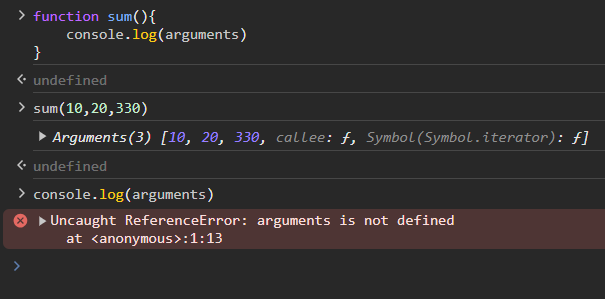
sum(10,20,30)

* you will get output for arguments

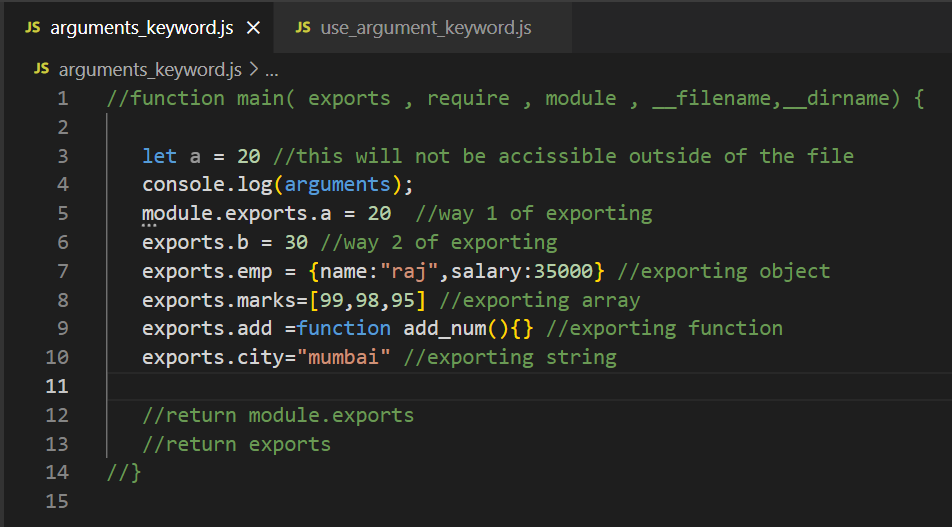
Later :

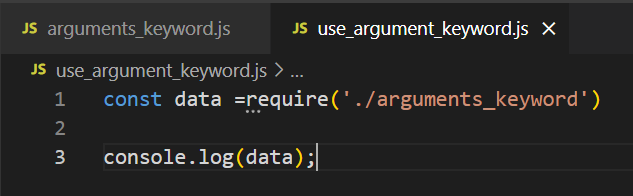
console.log(arguments)

* you will get error
* Conclusion : in browser, arguments keyword only work inside function.



* 1. Create folder -> argu



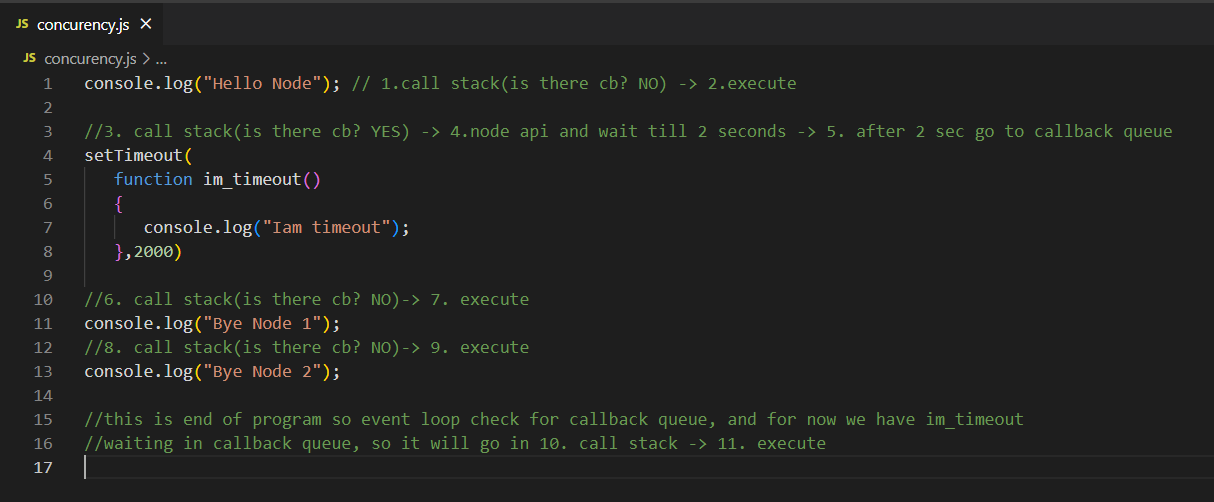


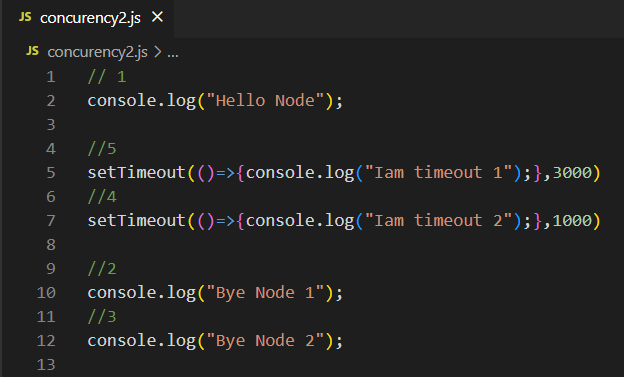
RUN :

>node arguments\_keyword.js

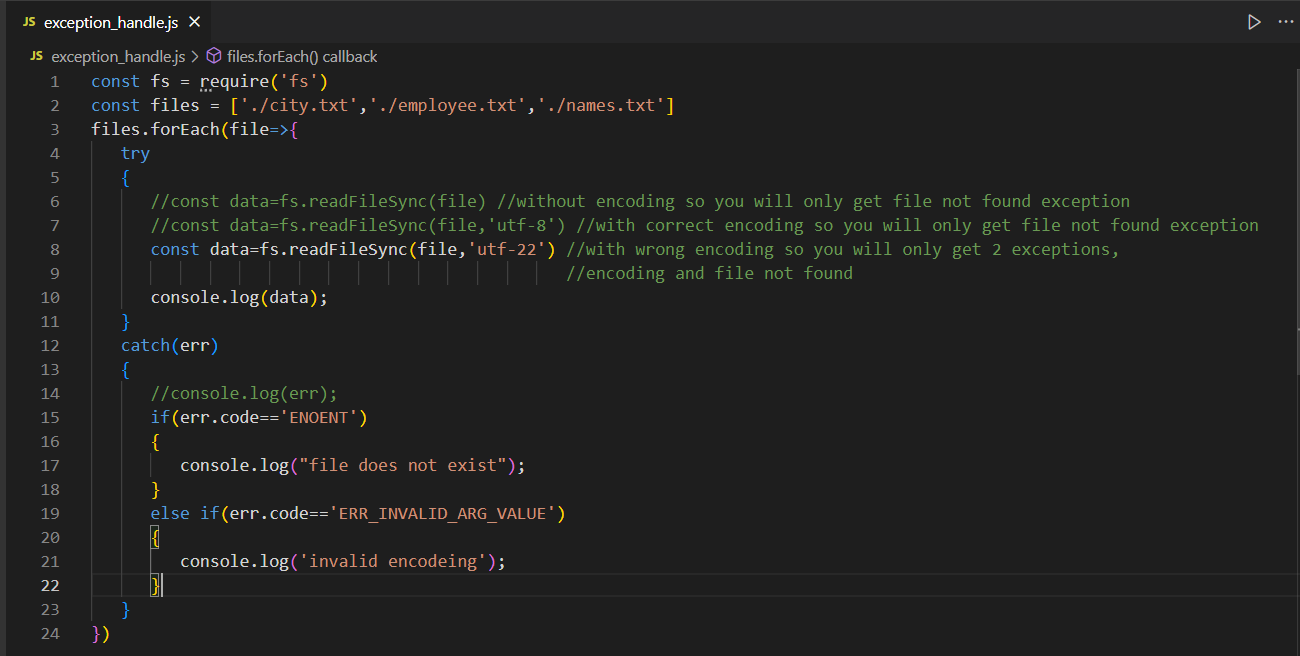
> node use\_argument\_keyword.js

1. Concurrency :

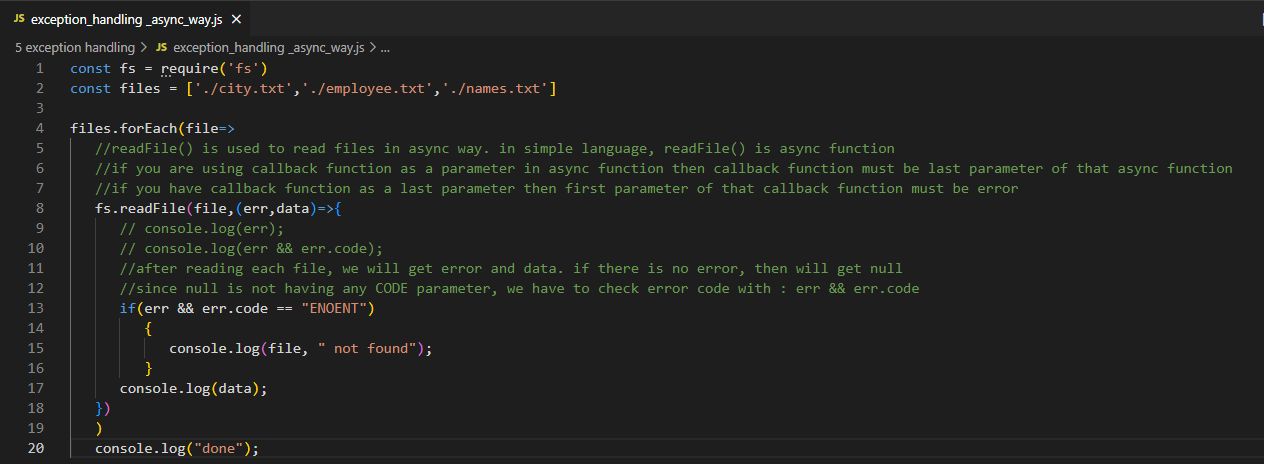




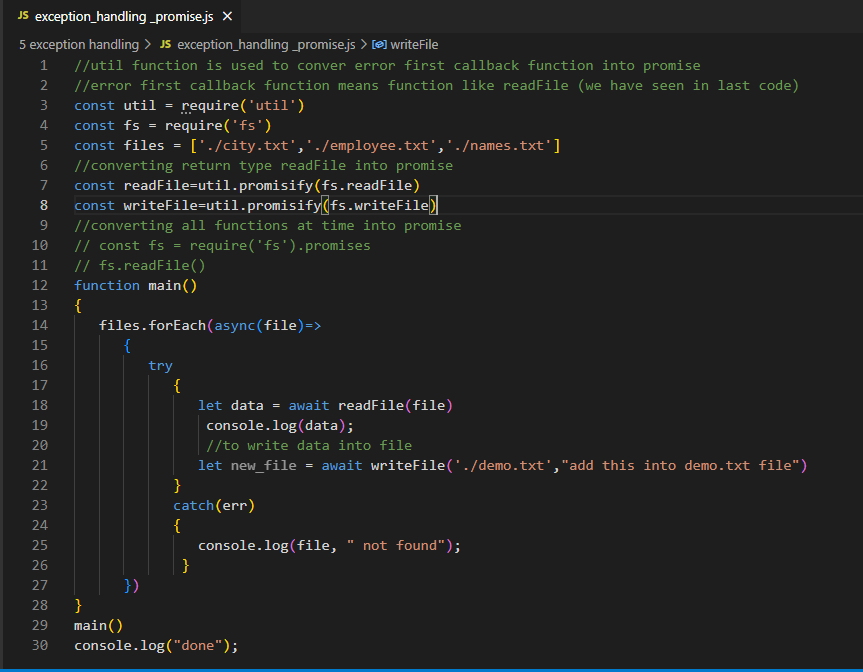
1. Exception handling
   1. Exception handling in synchronous way – using readFileSync() function
   2. Create 2 files in current folder – city.txt (add some city names) and names.txt (add some names)



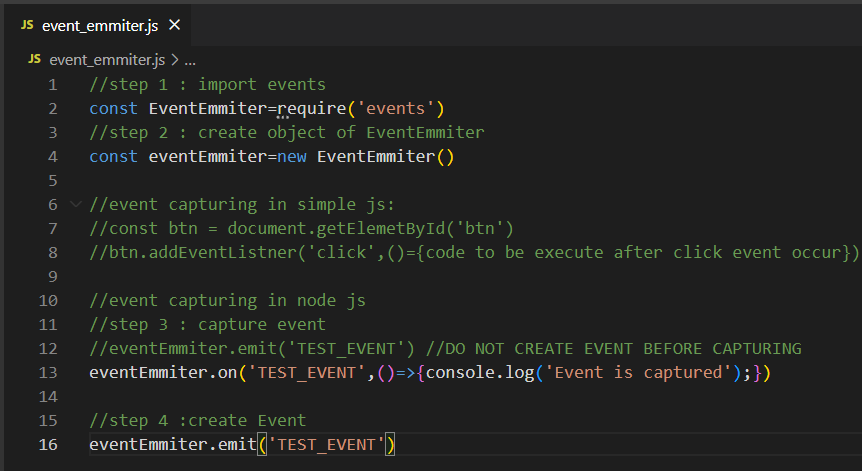
* 1. Exception handling in Async way
     1. There are 2 ways in which we can handle exception in async way – using callback function or using promise (async-await pattern)
     2. Exception handling in Async way using callback function



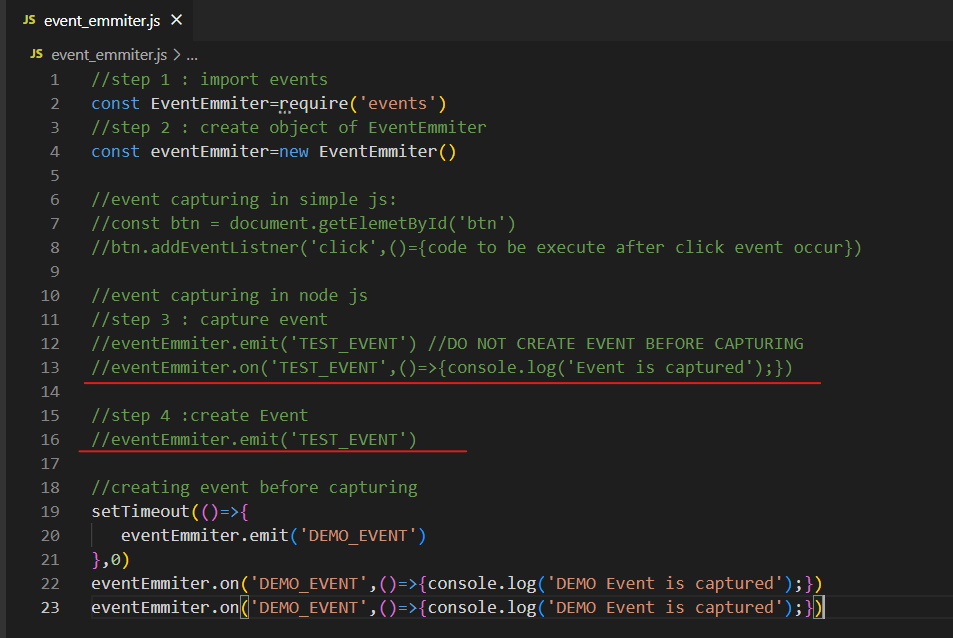
1. Exception handling in Async way using promise (async-await)



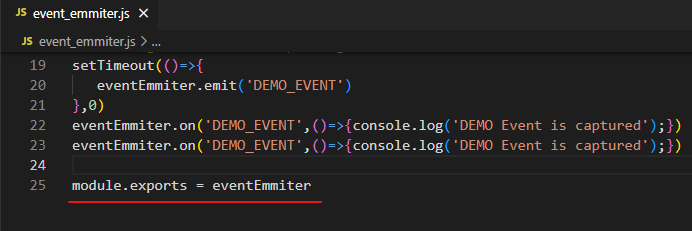
1. Event Emitter:
   1. Import events
   2. Create object of EventEmmiter class present inside events package
   3. Use this EventEmmiter class’s object and capture event (subscribe to event)
   4. Finally create event
   5. We can’t create event before capturing event (ex – throwing a ball and expecting fielders to catch the ball, if there are no fielders then who will catch (capture) the ball )
   6. Simple event capturing



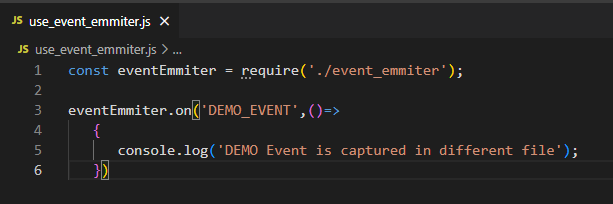
* 1. Capturing event before creating (use callback)



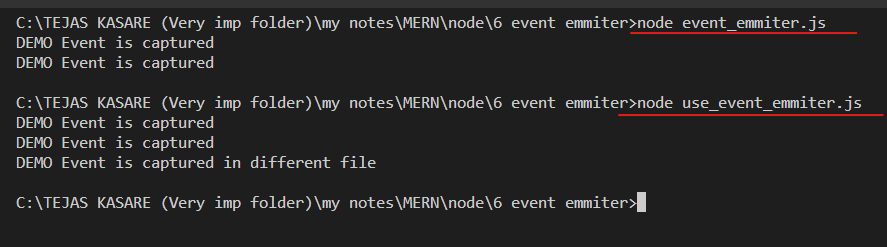
* 1. Capturing event in different file
     1. Export eventEmmiter from event\_emmiter.js



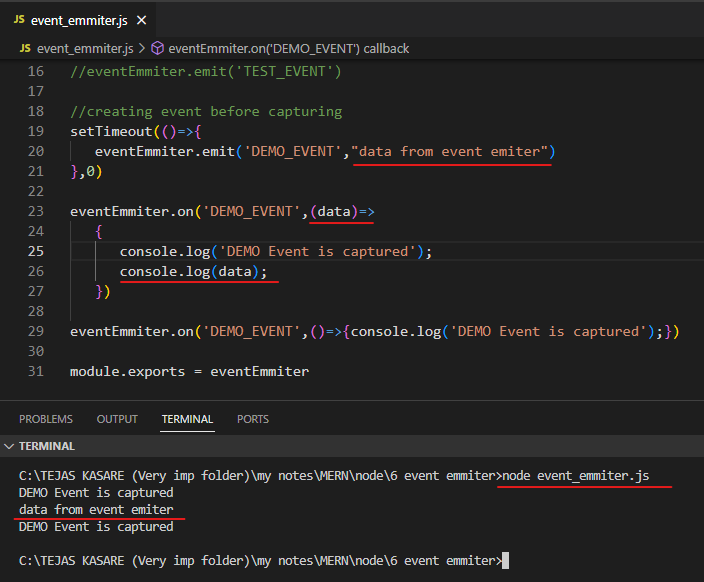
* + 1. Create new file (use\_ event\_emmiter.js) and import eventEmmiter exported from event\_emmiter.js and Capture event in use\_ event\_emmiter.js



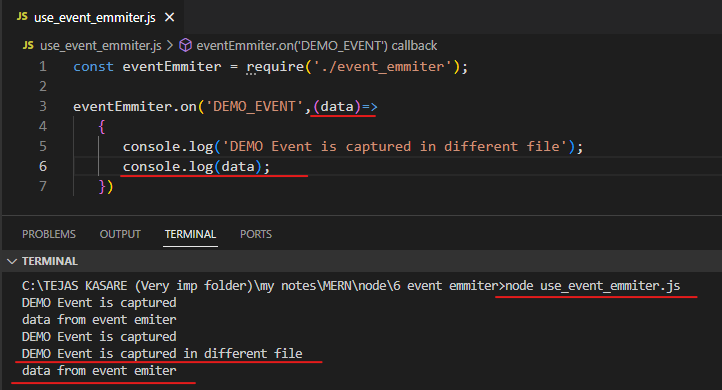
* + 1. Run use\_ event\_emmiter.js (>node use\_ event\_emmiter.js)



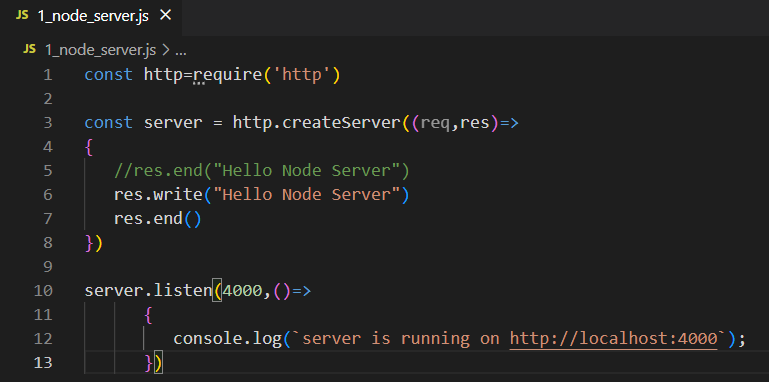
* 1. Passing data in events
     1. It is not compulsion on capturing data sent by eventEmmiter
     2. Capturing data sent by eventEmitter in same file

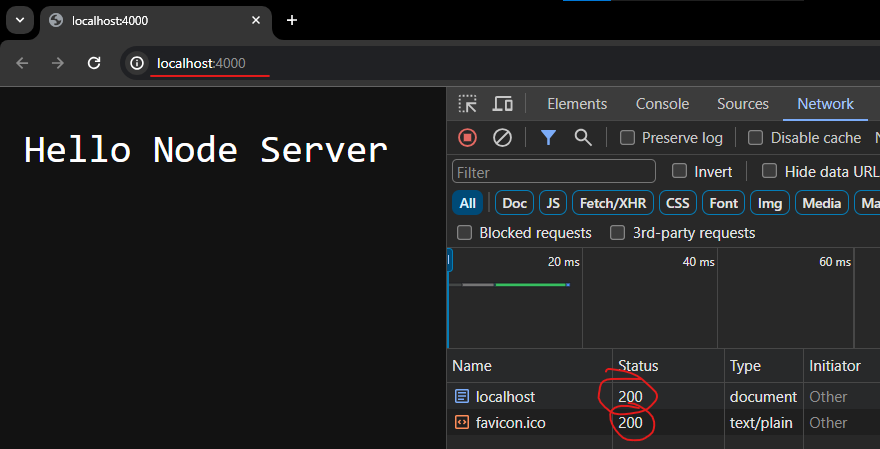
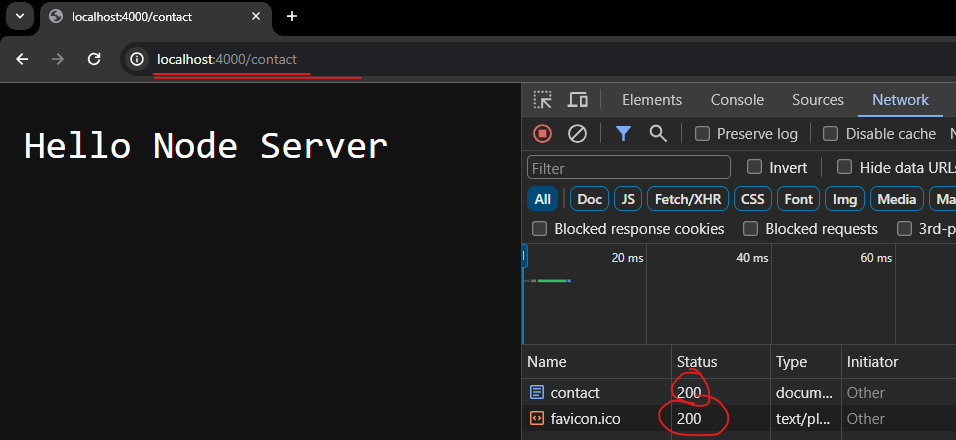


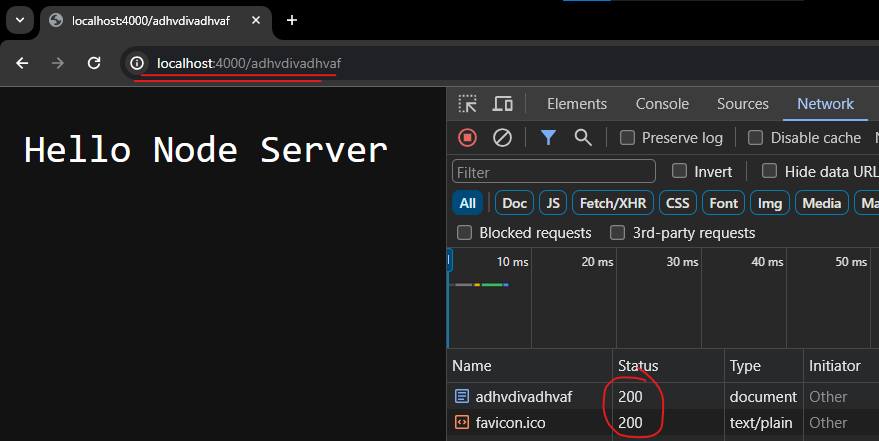
* + 1. Capturing data sent by eventeEmmiter in different file



EXPRESSSS

1. Drawbacks of creating server using node js (we have already seen how to create server using node)
   1. Drawbacks of creating server using node js (we have already seen how to create server using node)
      1. I have created folder “7 express” in that folder : 1\_node\_server.js
      2. 
      3. Run above code >node 1\_node\_server.js and check output on browser

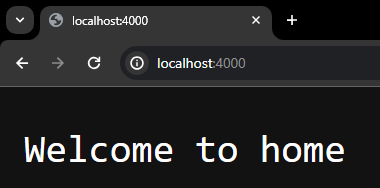
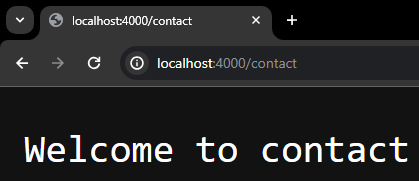
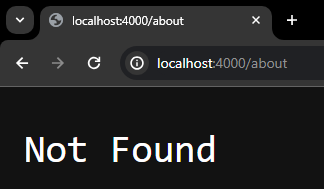


Explanation :

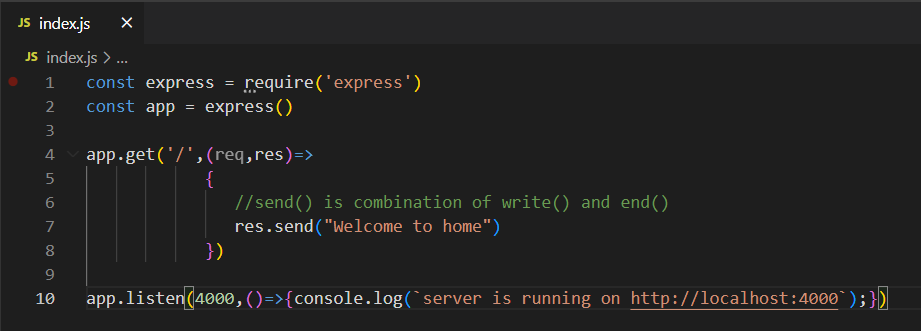
* + 1. Handling requests in node servers



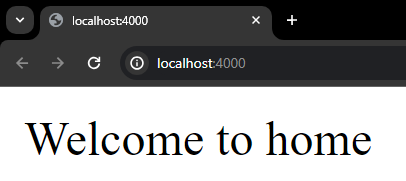
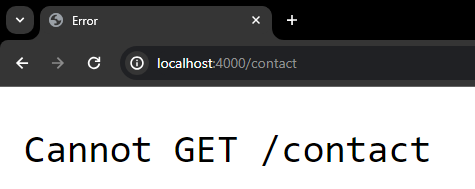
Result :

1. Creating server using express
   1. Create file index.js
   2. Creating package.json using >npm init
   3. Then install express by >npm i express
   4. Write code in index.js to create server using express

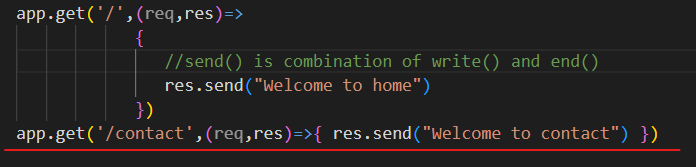


* 1. Run file and check output on browswer

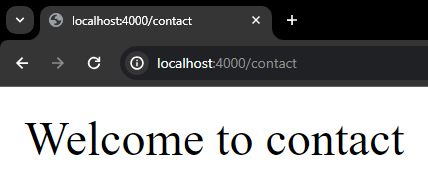
 

Since we are not habdling /contact url, so we will get error page

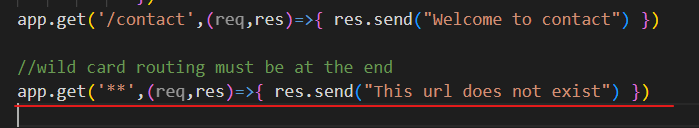
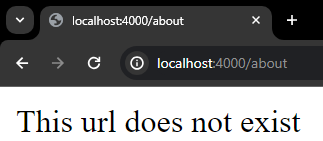
So handling /contact url

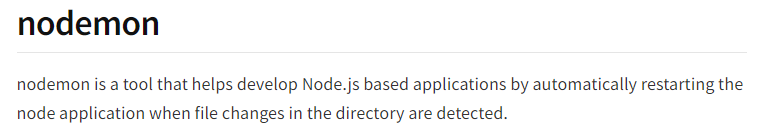
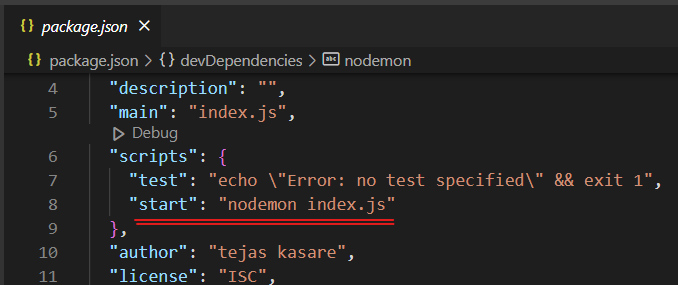


Output

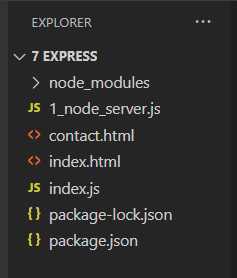


* 1. Handling wild card routing : handling url which not created by us

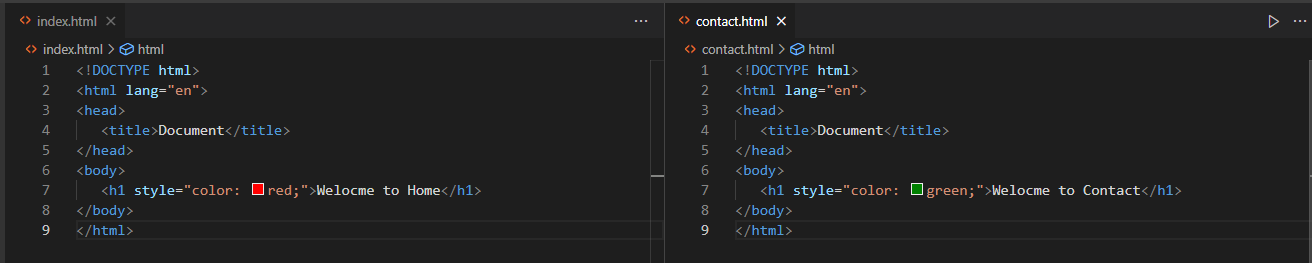
 

* 1. Installing nodemon package : to restart server as soon as we perform changes in file
     1. >npm i –D nodemon
     2. 
     3. Add script for nodemon in package.json
     4. 
     5. Here after whenever you start the server, use command >npm start instead of >node index.js

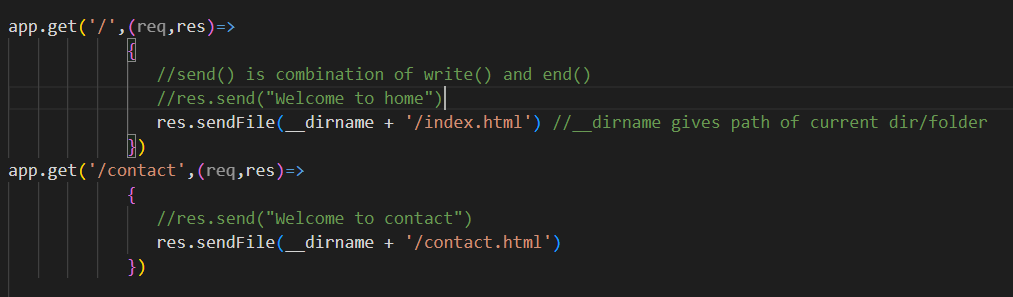
1. Displaying html file as a response
   1. Create index.html and contact.html



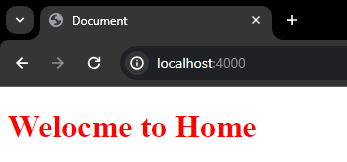
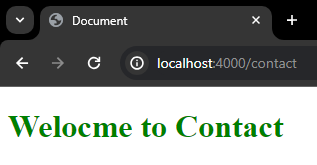
* 1. Write some html code in it



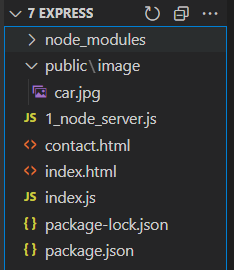
* 1. Update url mapping in index.js as –



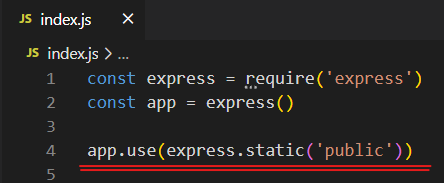
* 1. Start server >npm start
  2. Output

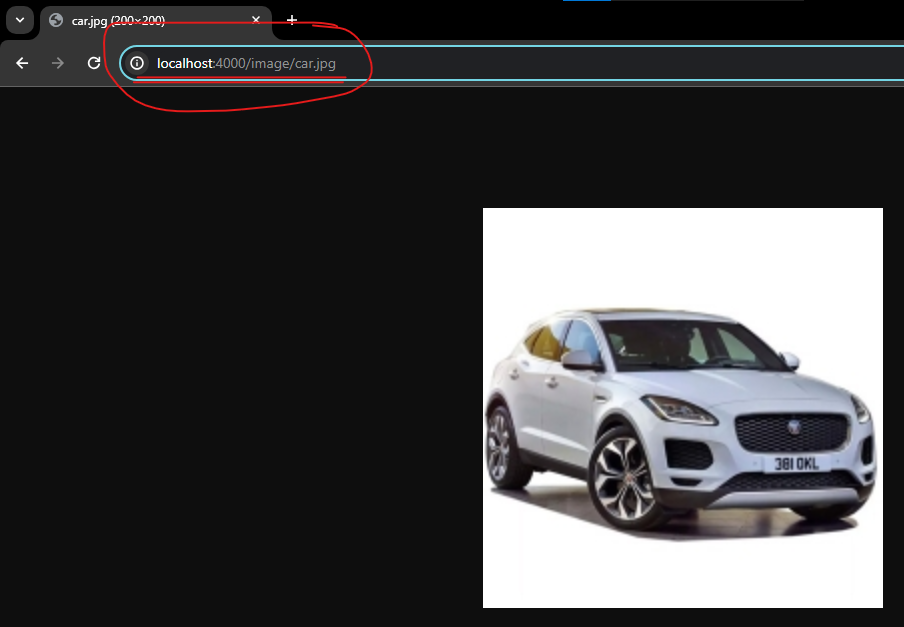
 

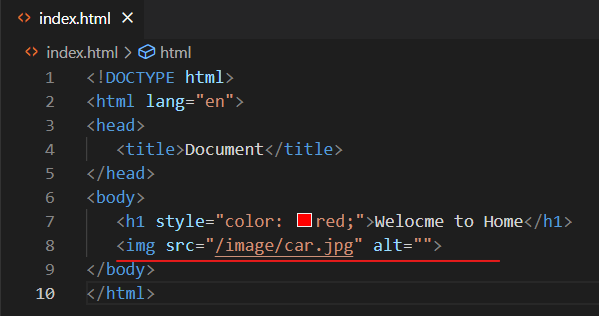
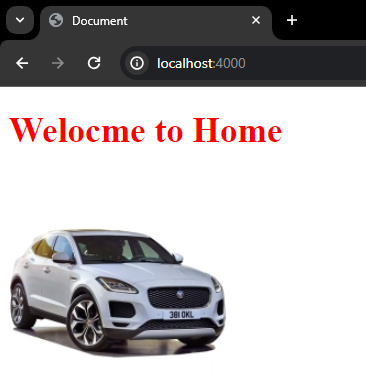
1. Displaying images in html
   1. Create public folder 🡺 Create image folder in public folder 🡺 add image in image folder



* 1. Make that folder static in index.js using express
     1. Static folder : can be accessed from server



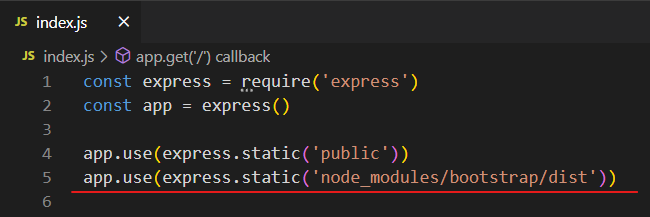
* 1. Get image on browser
     1. 
  2. Use image in index.html

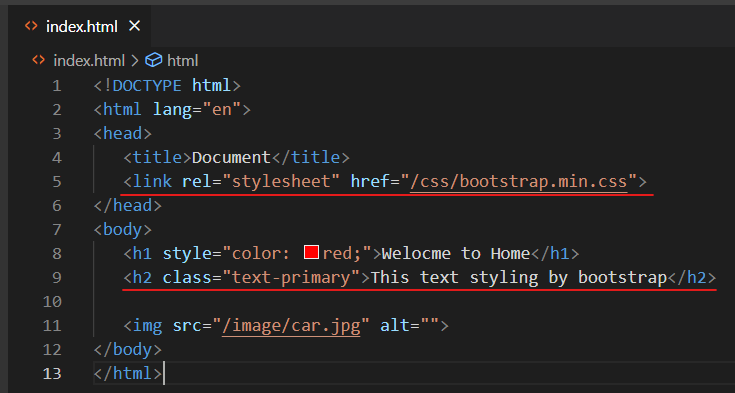
1. Adding Bootstrap
   1. Install bootstrap (get its npm command from its official website)

Currenty : >npm i bootstrap@5.3.2

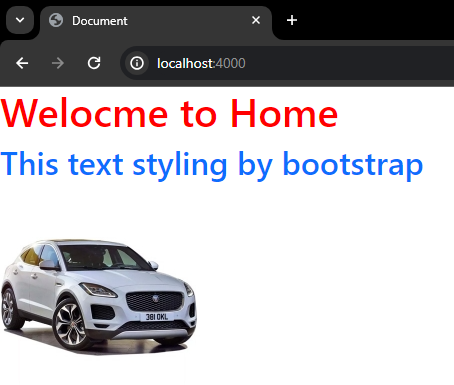
* 1. Make its dist folder static (like we did with public folder above)

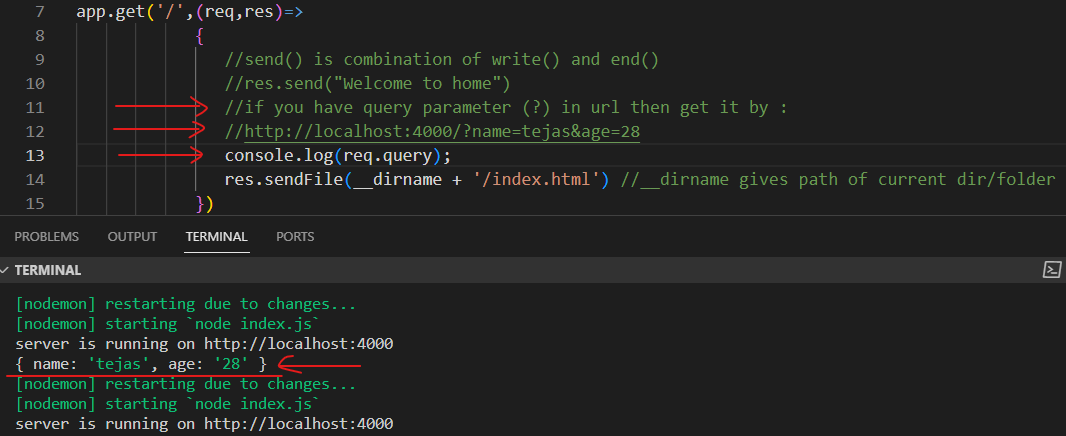


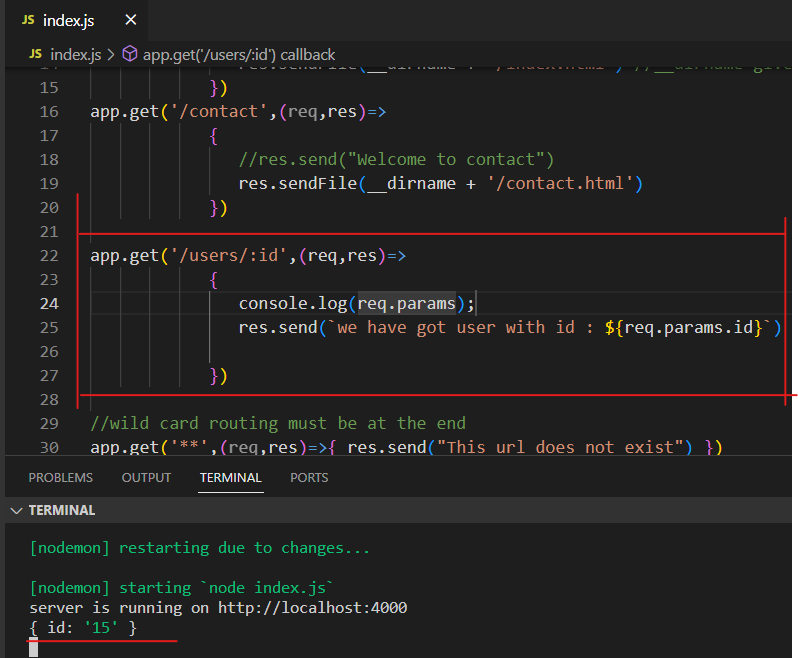
* 1. Use boostrap’s css file

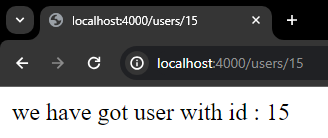
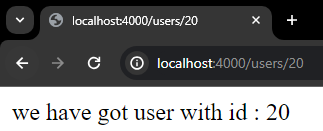


* 1. Re run server and check Output

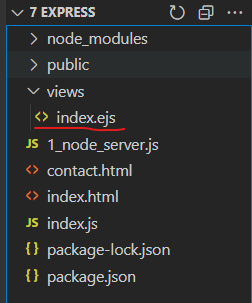


1. Collecting data from url
   1. There 2 types of url
      1. localhost:4000/?name=tejas&age=28 🡺 query parameter
      2. localhost:4000/users/1 🡺 dynamic parameter
   2. collecting data from query parameter
      1. 
   3. Collecting data from dynamic parameter

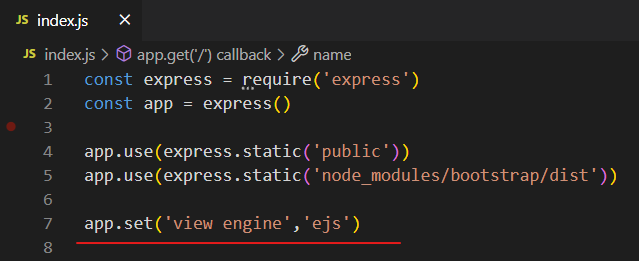


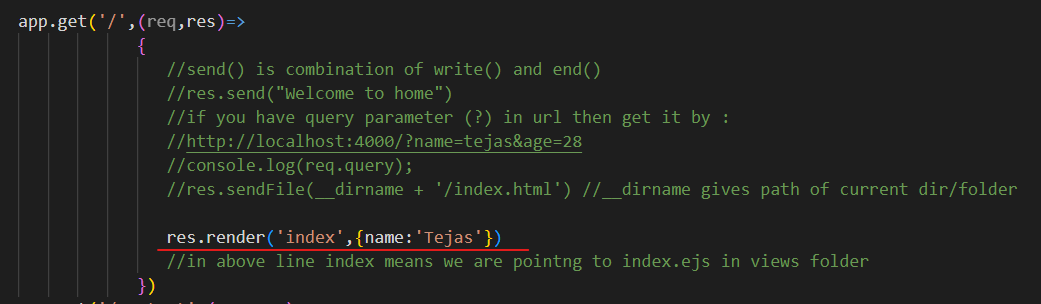
1. Sending data to html : working with ejs (template engine for js)
   1. Stop server and Install ejs >npm i ejs
   2. Create views folder (name is fixed : views)
   3. In that folder create index.ejs file (here index.ejs is same as index.html)



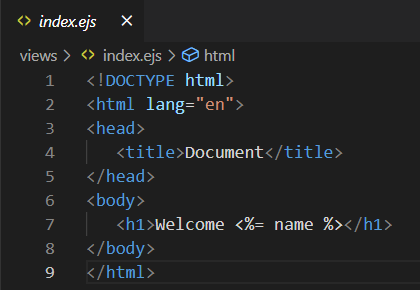
* 1. Set ejs as template engine in index.js



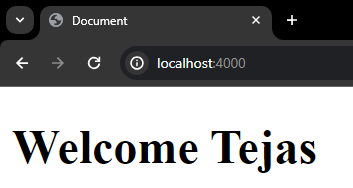
* 1. Send data to html using ejs as –

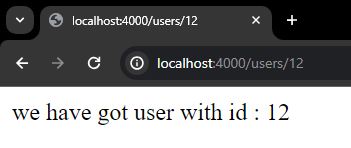
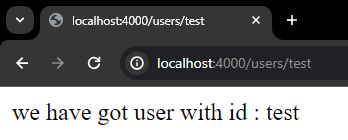
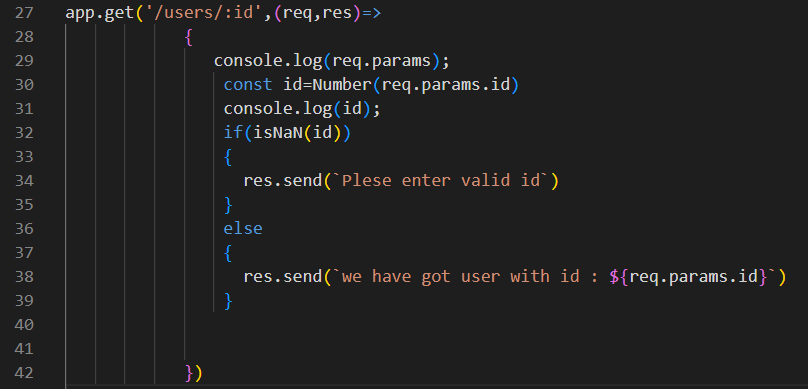
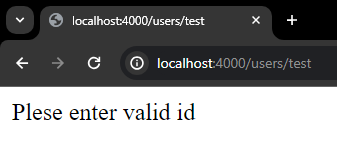
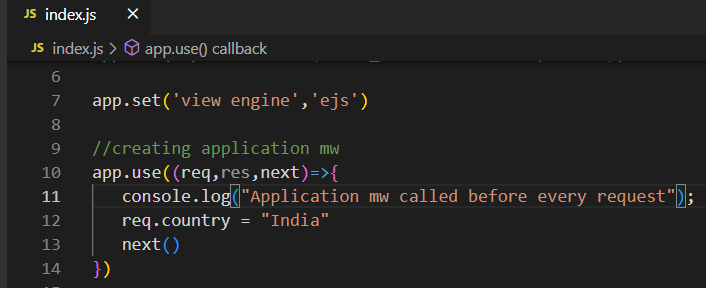
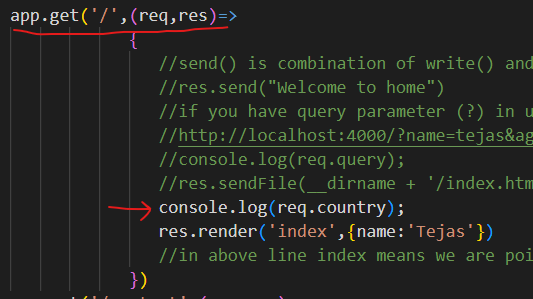
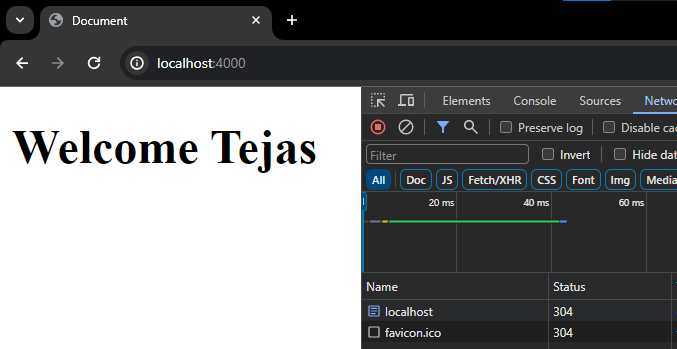
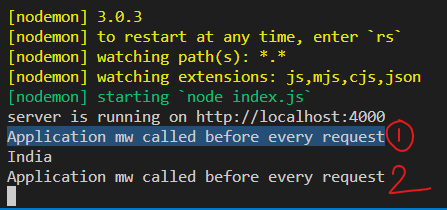
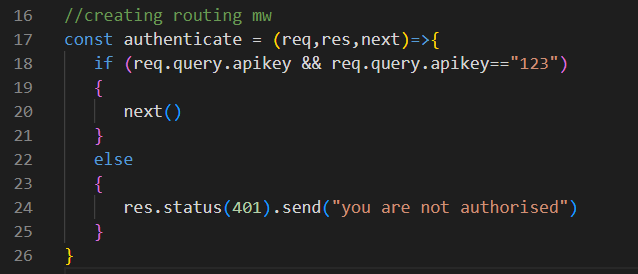
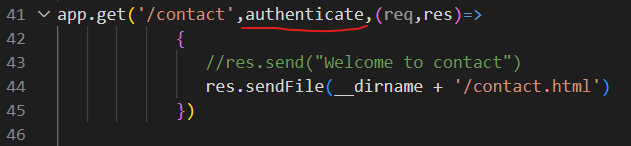
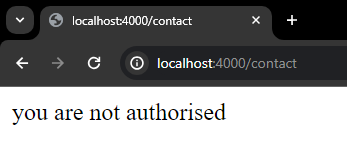
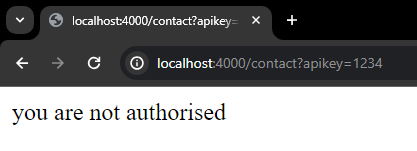
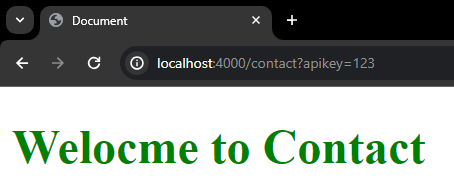


* 1. Accessing data in index.ejs



* 1. Check output



1. Handling data type of parameterize routing
   1. 
   2. 
   3. Our id must be numeric value, since “id” is dynamic parameter, so it will match everything that you pass after /users/. So to manage this, we have to perform type casting on dynamic parameter
   4. 
   5. Output :
   6. 
2. Middleware :
   1. There are 3 types of middleware : application level, route level and exception level
   2. Application mw get called before every of that application
   3. Routing mw : get called before any specific route (url)’
   4. Exception mw: get called before any exception
   5. Creating application mw
      1. 
      2. Use req.country variable has some data that you can access on any request
         1. 
      3. Now go to browser and call any request : <http://localhost:4000> or <http://localhost:4000/users/10> and check terminal you will get :
         1. 
         2. 
         3. 2 time because for 2 network call
   6. Creating routing mw
      1. Only work on specified routes (url)
      2. 
      3. Using on contact url (if you have entered correct api key then only show contact page)
      4. 
      5. Go and check /contact
      6. 
      7. 
      8. 
3. REST