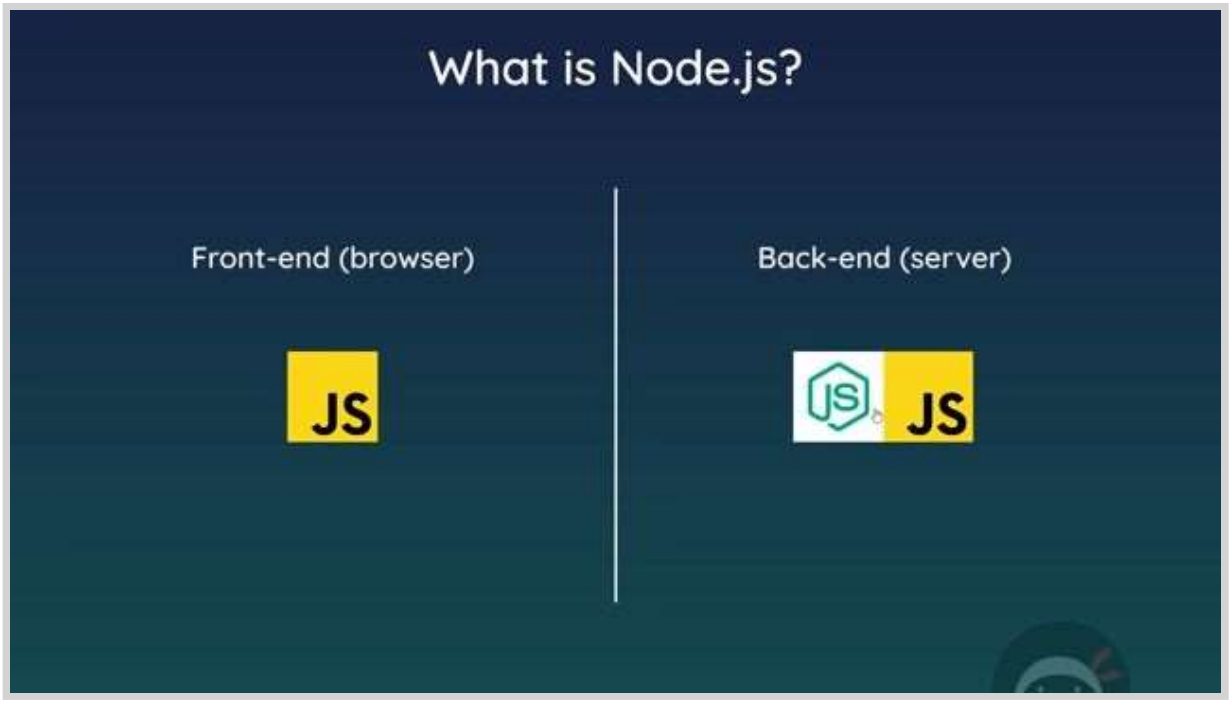
←

# Video1:

01:14




**Node allows JS to run it on server..**

**V8 engine compiles JS into machine code..**

**as js samjt nahi computer laa..**

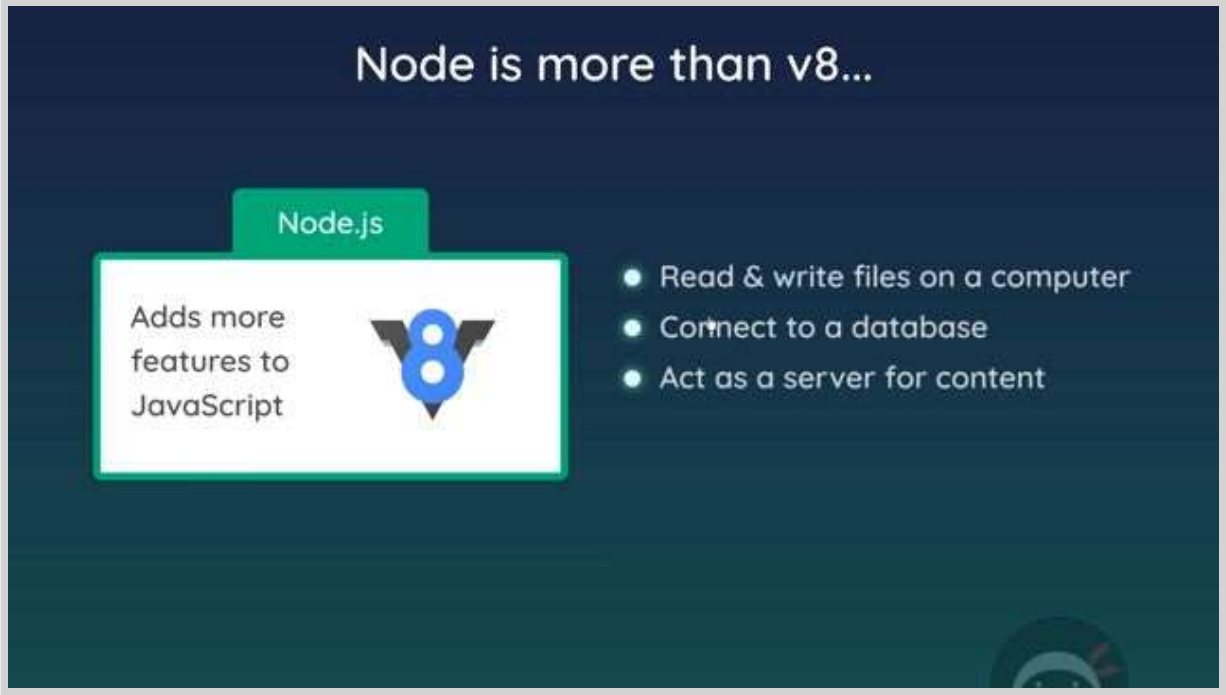
**we can't run js outside browser ..**

**Node.js is written in c++, we can run directly it on computer**

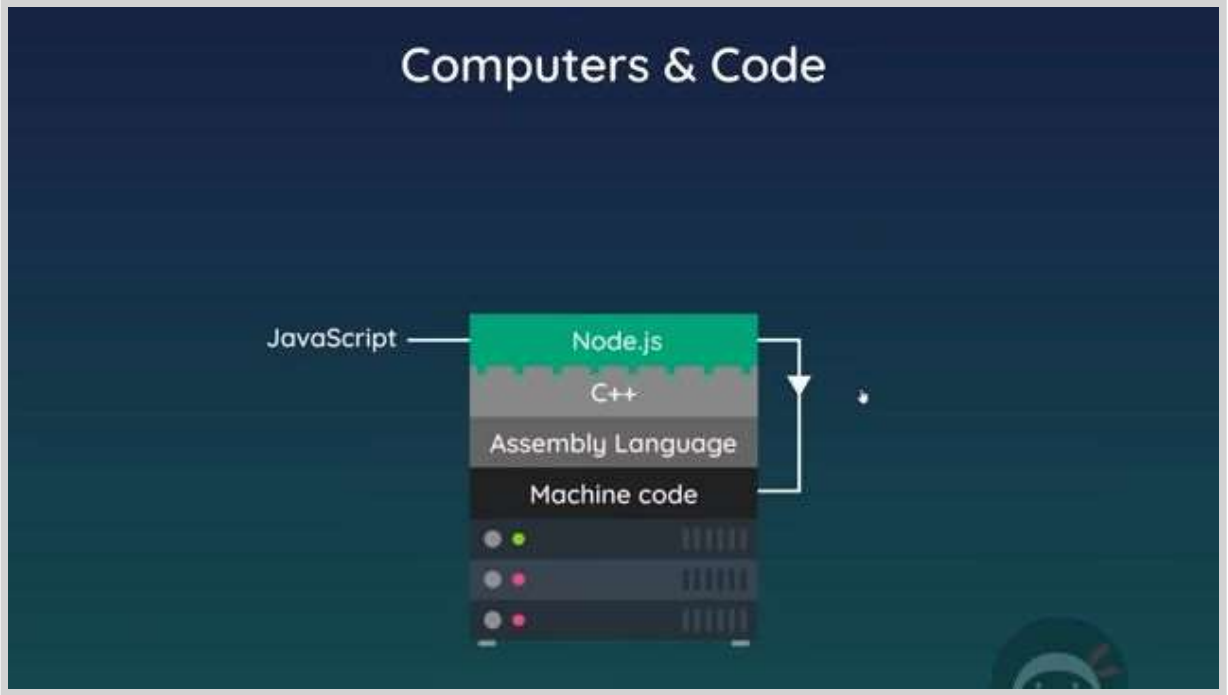
03:40


Enter Node.js
Node.js
JS
Machine Code
Node.js is written in C++

03:53


Node is more than v8...
Node.js
Adds more features to JavaScript
Read & write files on a computer
Connect to a database
Act as a server for content

05:02

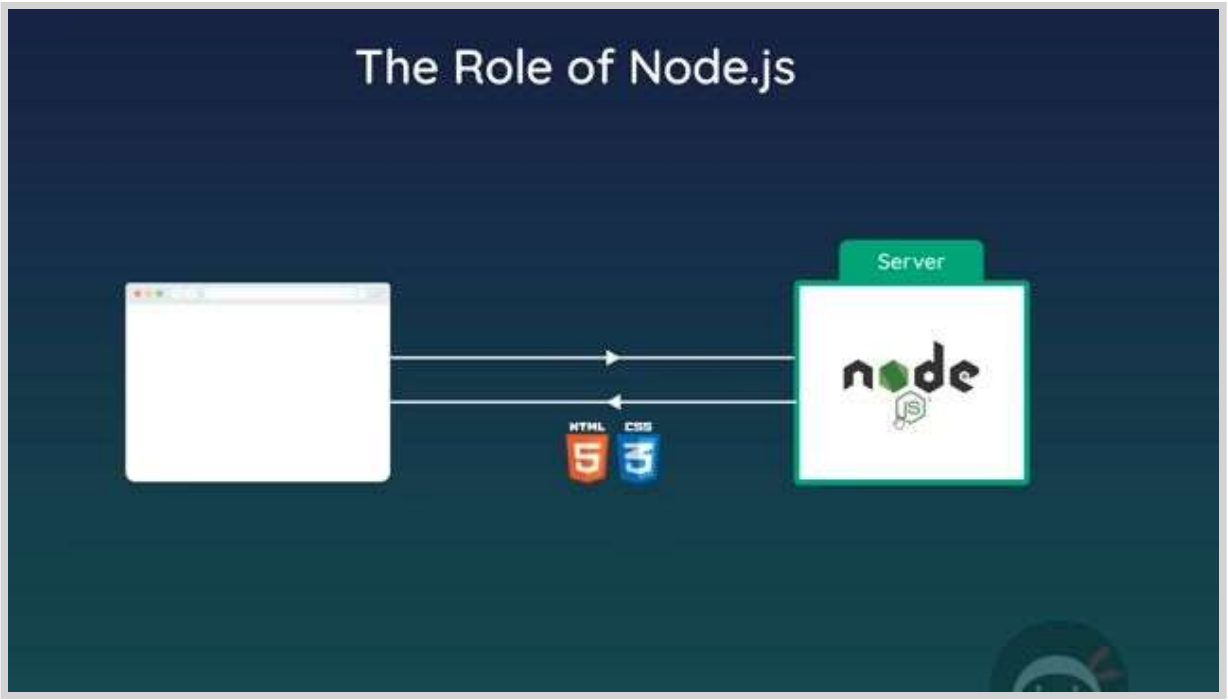



**we can write js on computer using V8 engine.**

**=>role of node js is to run js on backend or server side(we re handling request on server side)..**

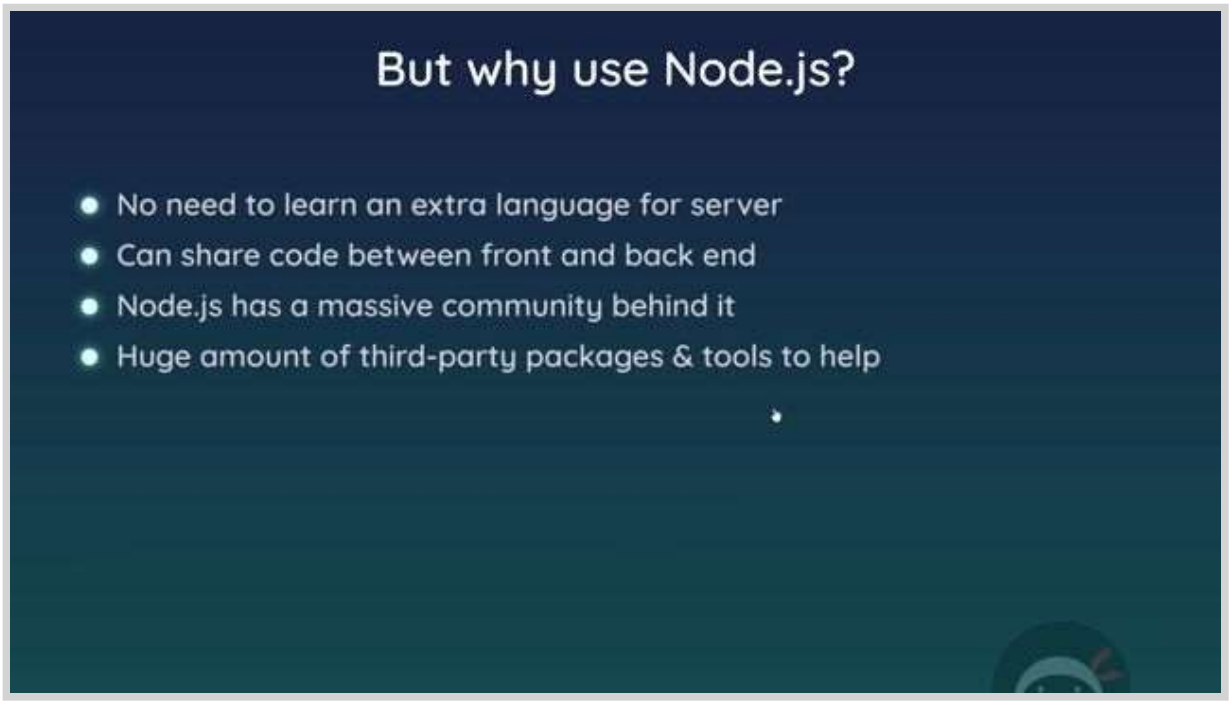
**browser make request to server and may communicate with db and formulate some kind off response and send back to browser..**

06:15




## benifits :

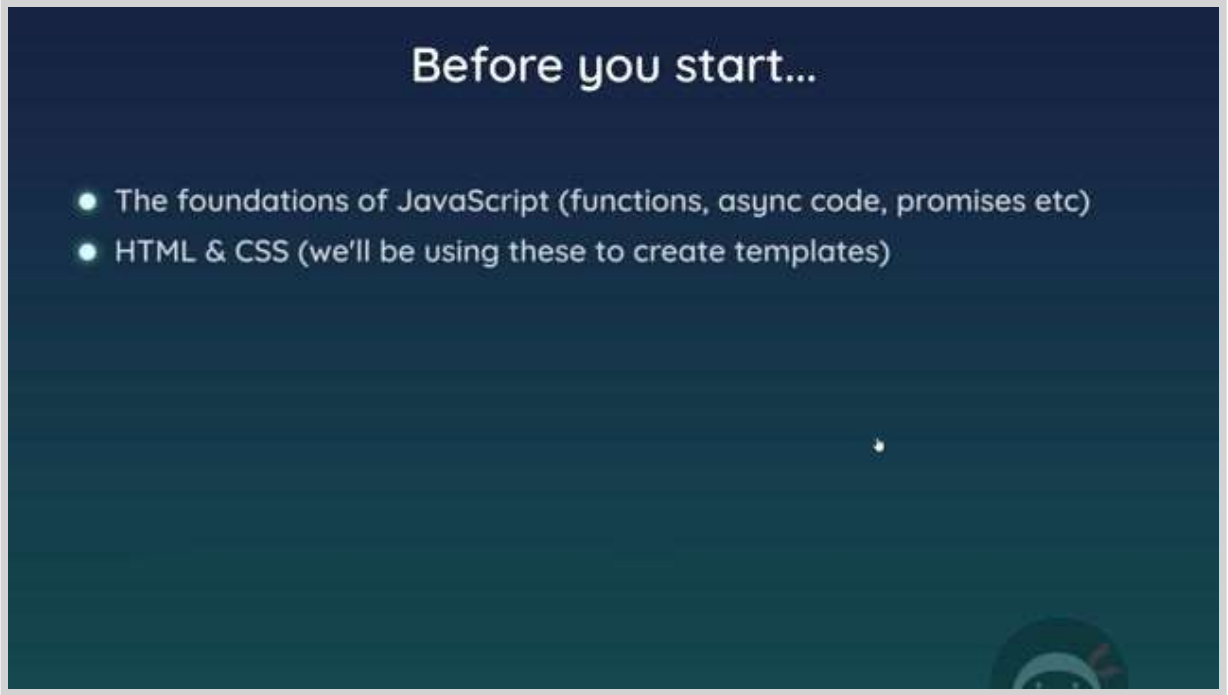
06:32




07:13

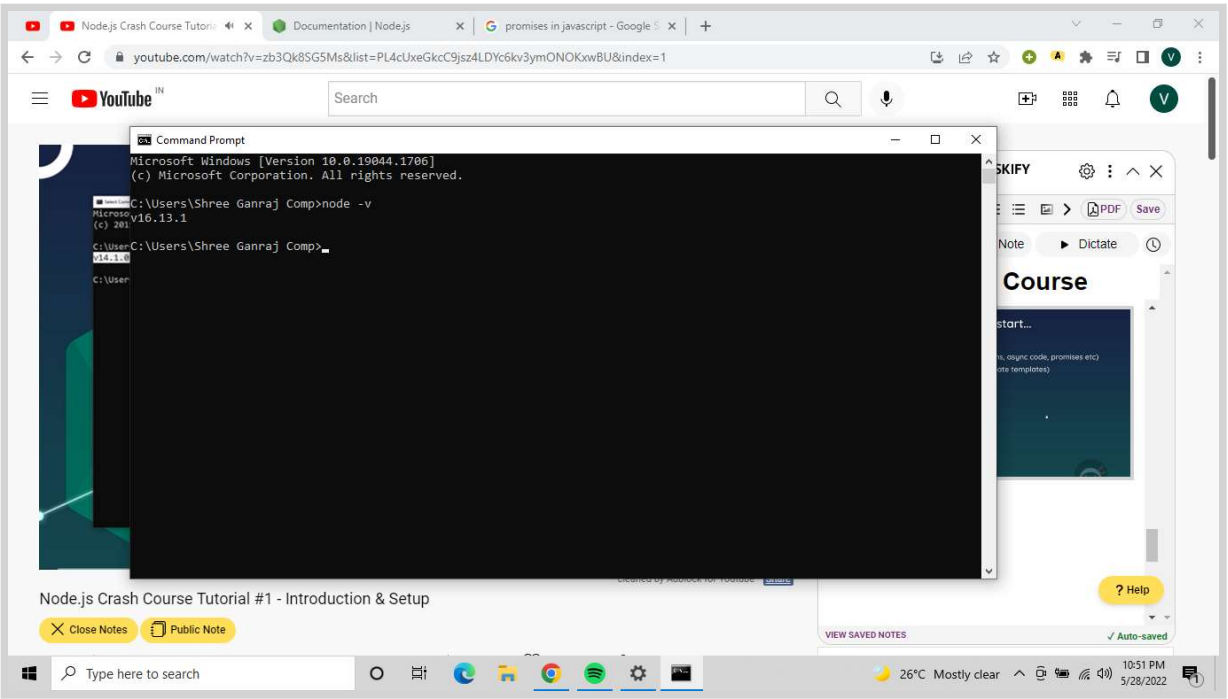

## EJS => for dynamic content

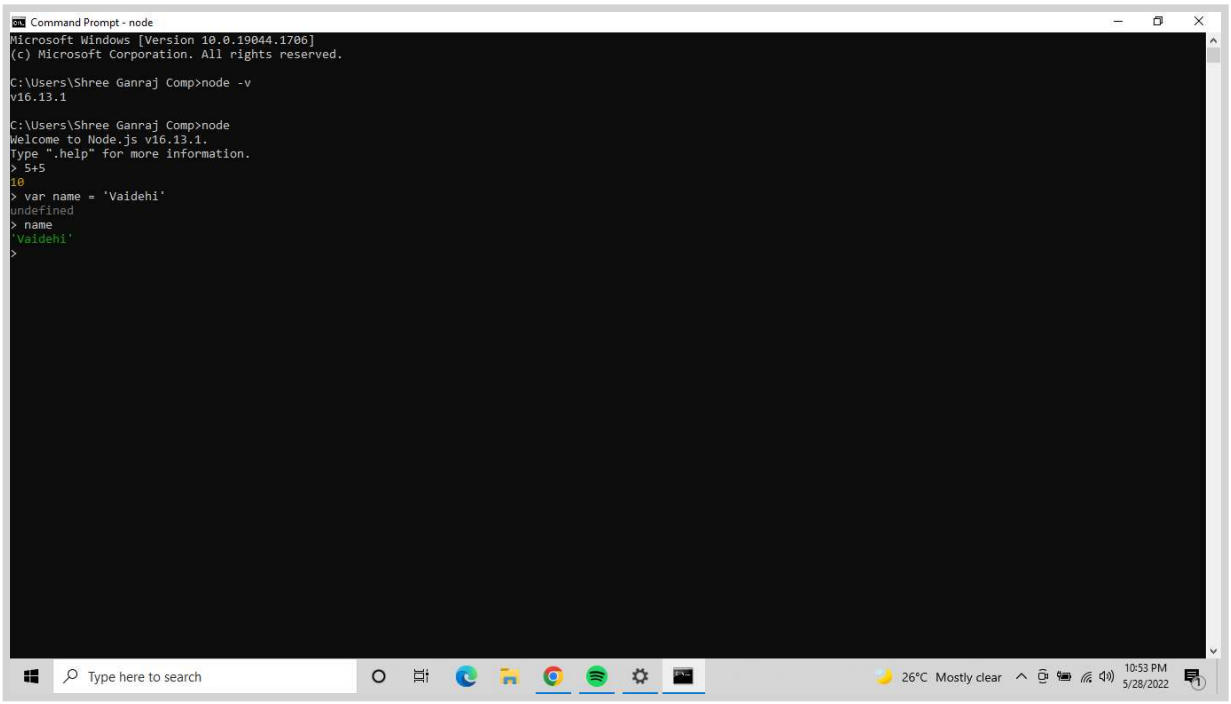
09:07

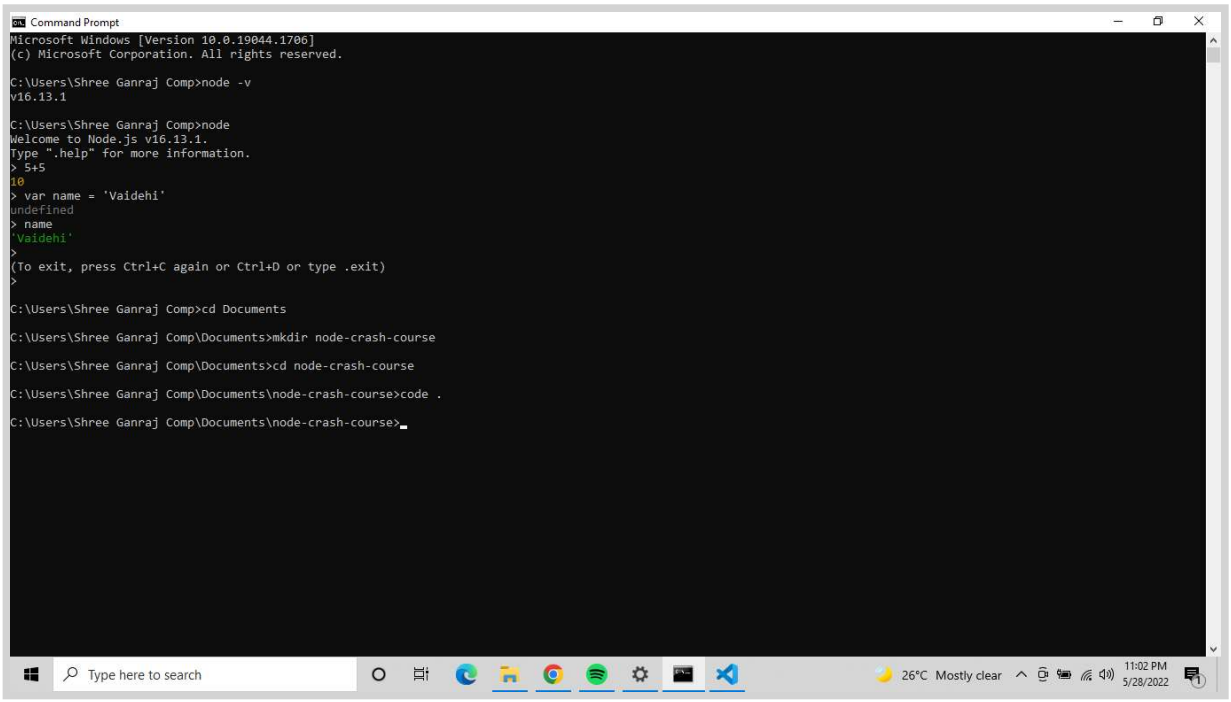



## Installing nodejs:

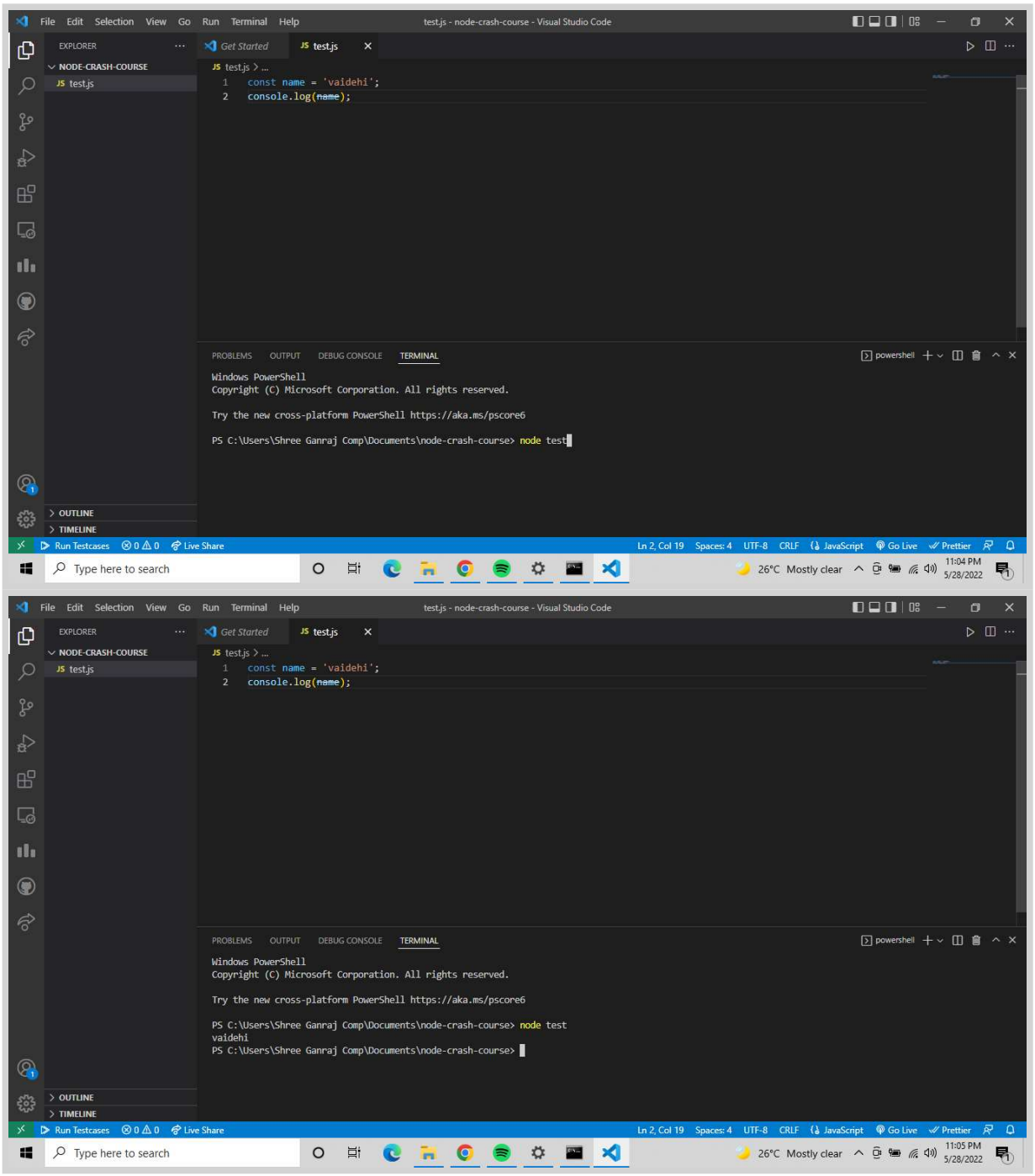
## https://nodejs.org/en/





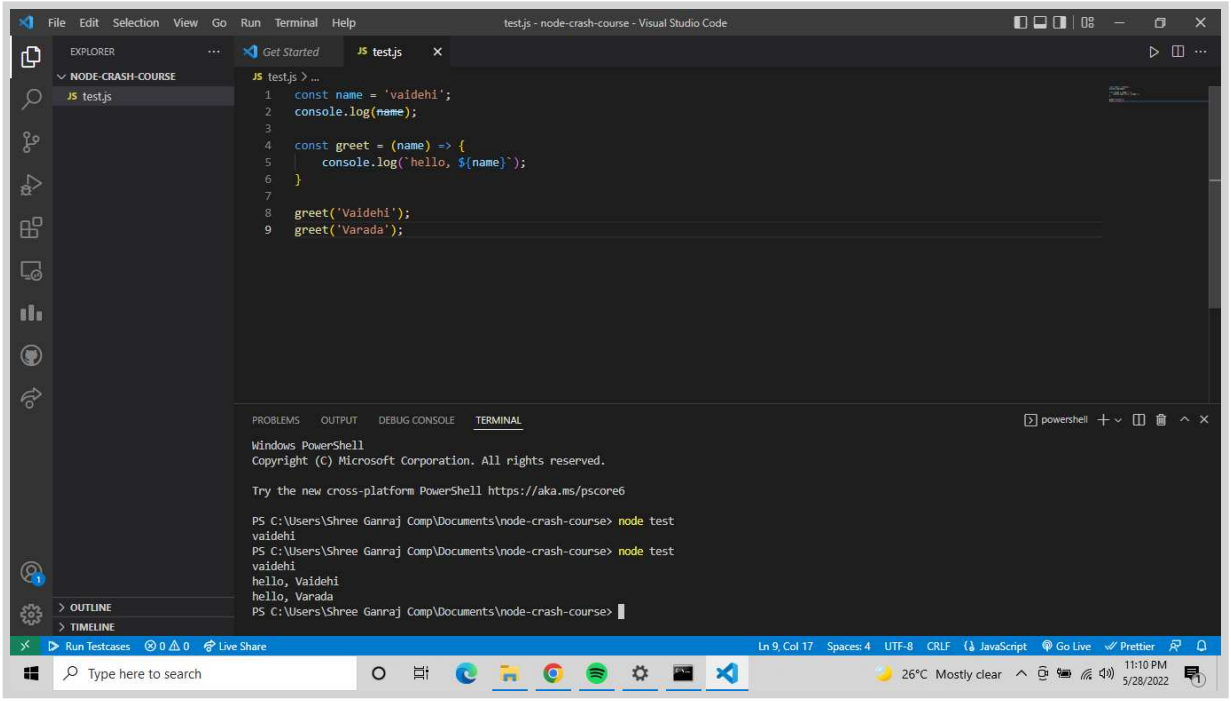

# Running File through Nodejs:




**so in this way we can run Javascript in our computer using Node**

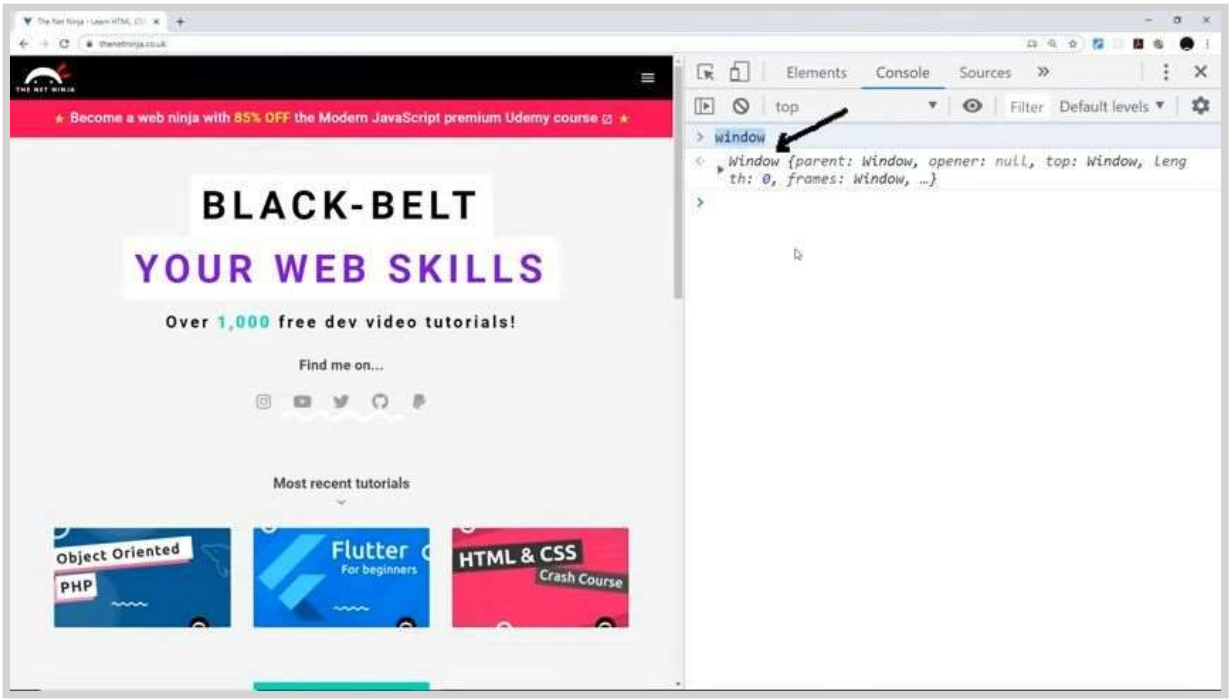
## Video2:

## Nodejs basics

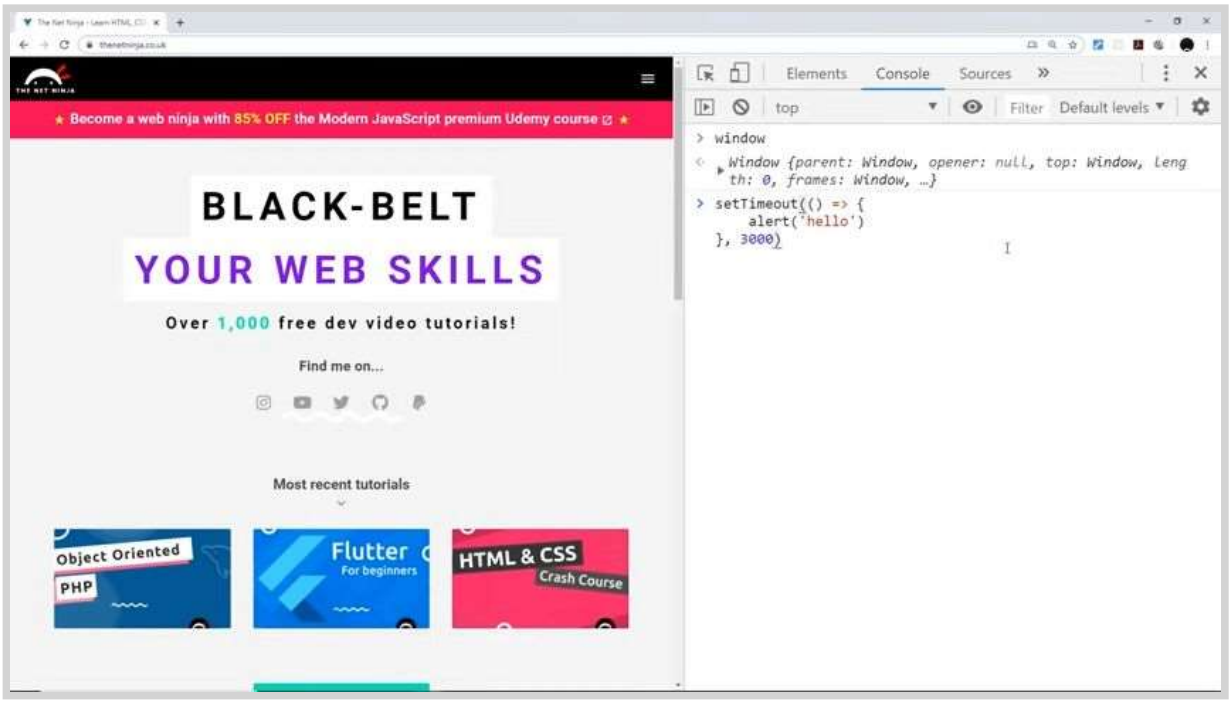



# The global object:

[02:47]



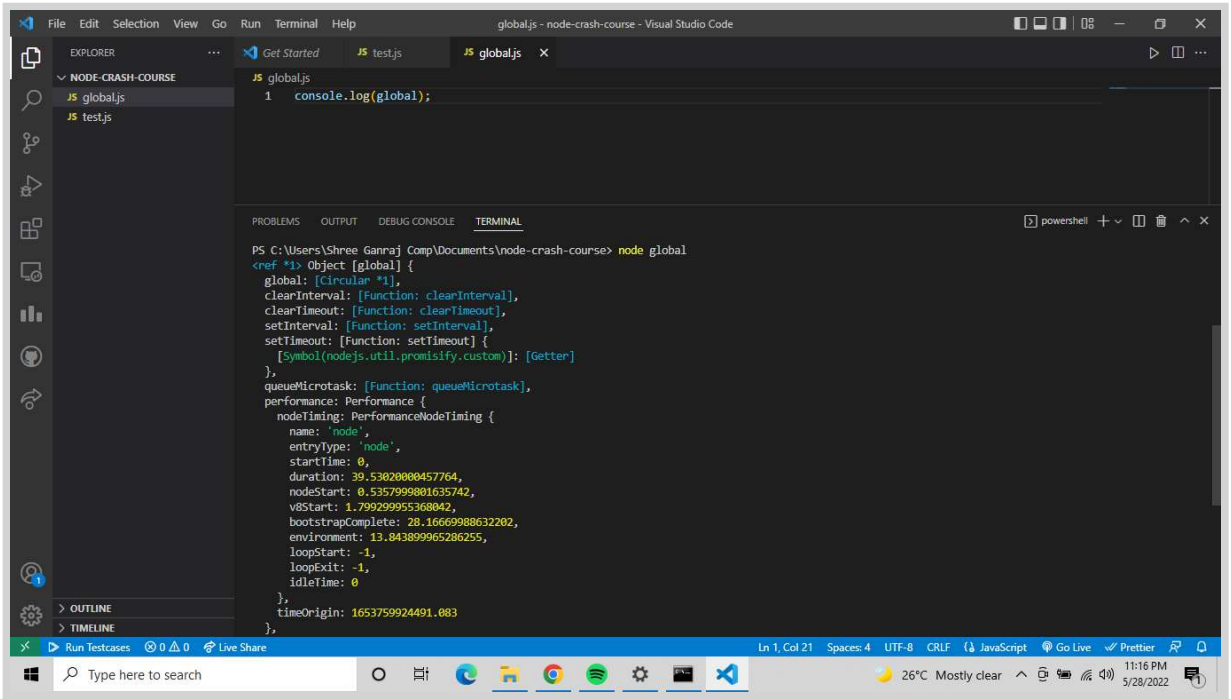

[03:06]

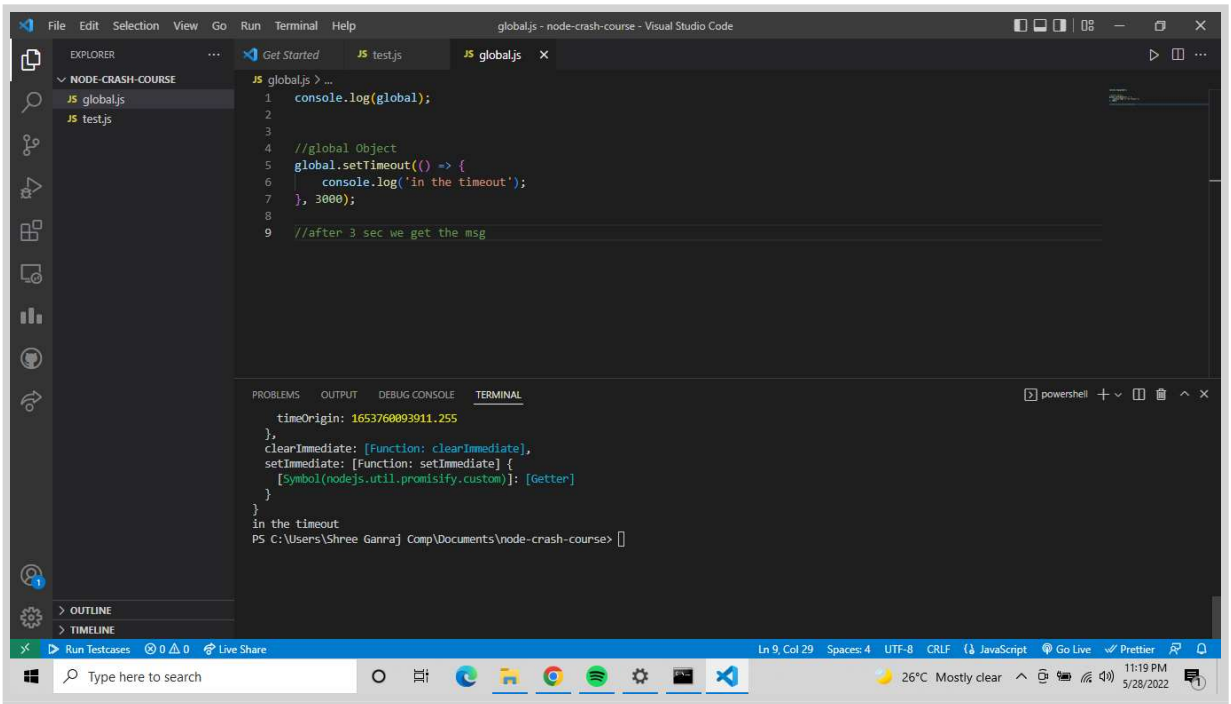


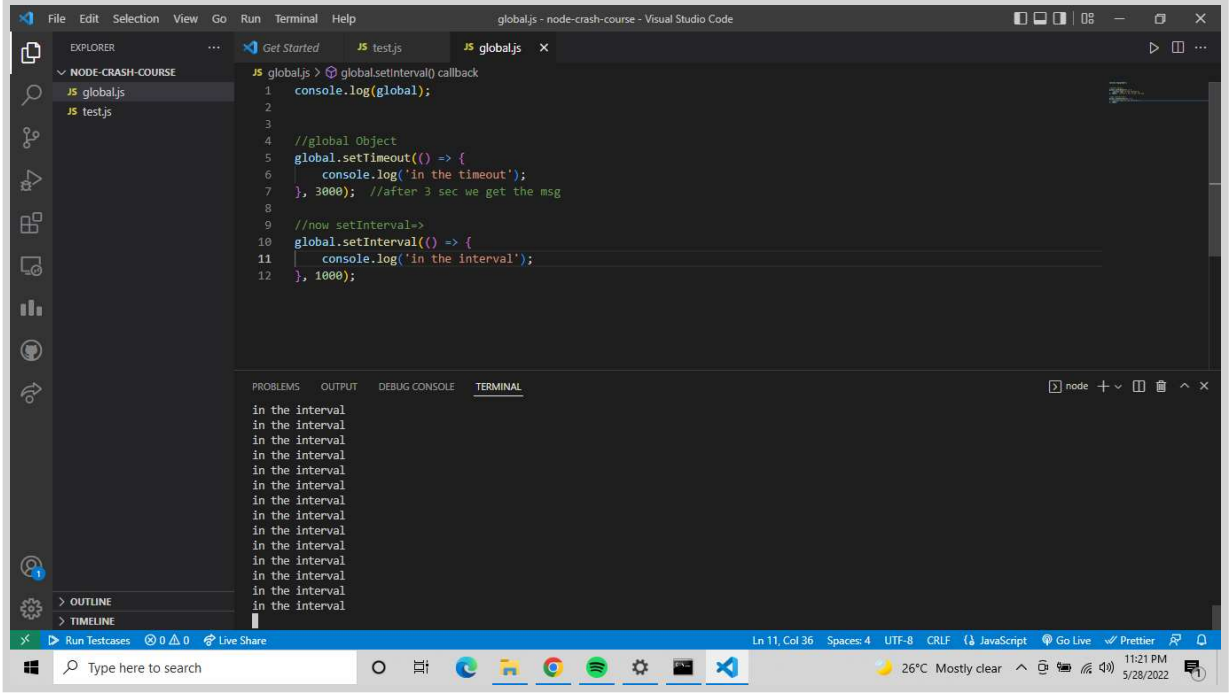

**explicitely window.setTimeOut() no need to write..inside the browser**

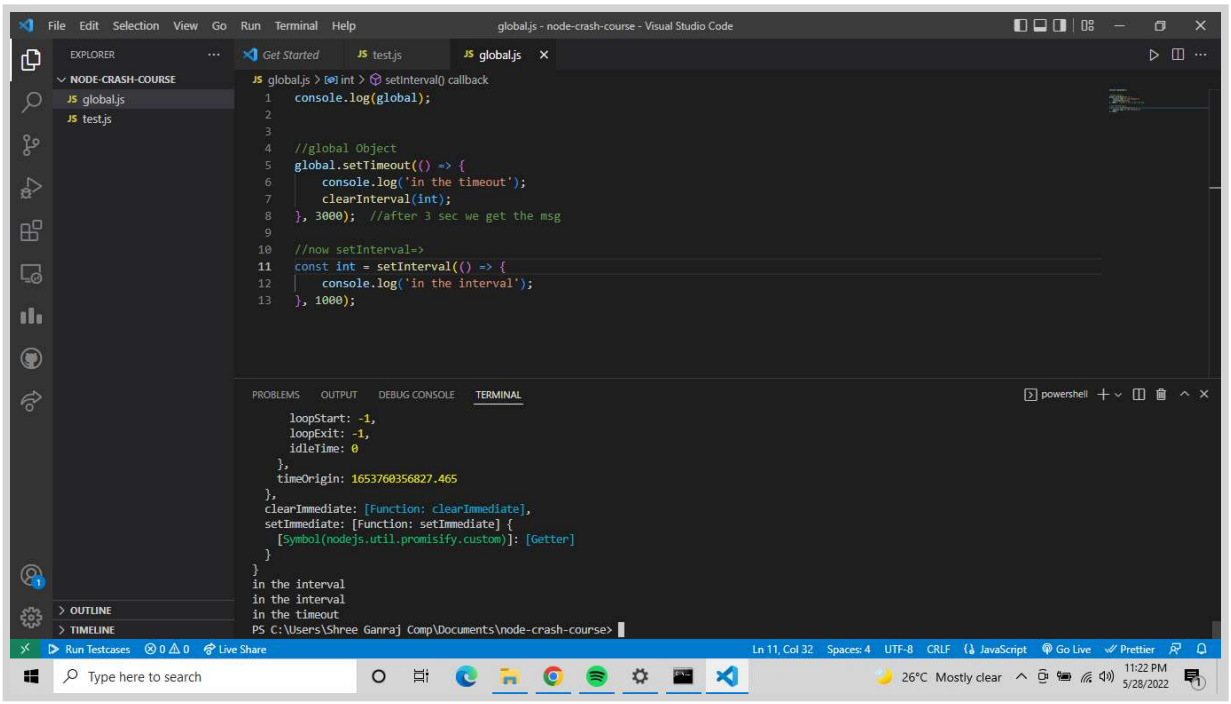
**setTimeOut() is the global object**

**but in node , window is not global object**



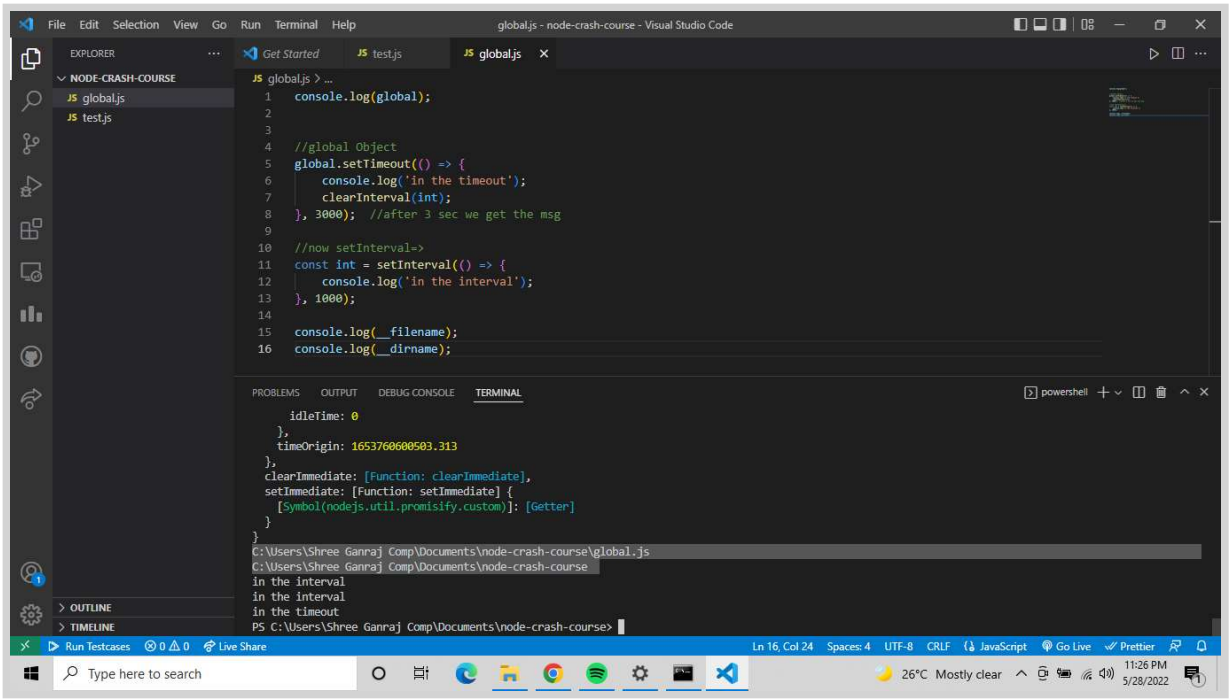

File Edit Selection View Go Run Terminal Help
global.js - node-crash-course - Visual Studio Code
EXPLORER
NODE-CRASH-COURSE
global.js
test.js
Get Started
test.js
global.js
global.js > ...
1 console.log(global);
2
3
4 //global Object
5 global.setTimeout(() => {
6 console.log('in the timeout');
7 }, 3000);
8
9 //after 3 sec we get the msg
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
powershell
timeOrigin: 1653760093911.255
},
clearImmediate: [Function: clearImmediate],
setImmediate: [Function: setImmediate] {
[Symbol(nodejs.util.promisify.custom)]: [Getter]
}
}
in the timeout
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course>
OUTLINE
TIMELINE
Run Testcases 0 0 0 Live Share
Ln 9, Col 29 Spaces: 4 UTF-8 CRLF JavaScript Go Live Prettier
Type here to search
26°C Mostly clear
11:19 PM
5/28/2022


File Edit Selection View Go Run Terminal Help
global.js - node-crash-course - Visual Studio Code
EXPLORER
NODE-CRASH-COURSE
global.js
test.js
Get Started
test.js
global.js
global.js > global.setInterval() callback
1 console.log(global);
2
3
4 //global Object
5 global.setTimeout(() => {
6 console.log('in the timeout');
7 }, 3000); //after 3 sec we get the msg
8
9 //now setInterval=>
10 global.setInterval(() => {
11 console.log('in the interval');
12 }, 1000);
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
node
in the interval
in the interval
in the interval
in the interval
in the interval
in the interval
in the interval
in the interval
in the interval
in the interval
in the interval
in the interval
in the interval
in the interval
in the interval
OUTLINE
TIMELINE
Run Testcases 0 0 0 Live Share
Ln 11, Col 36 Spaces: 4 UTF-8 CRLF JavaScript Go Live Prettier
Type here to search
26°C Mostly clear
11:21 PM
5/28/2022



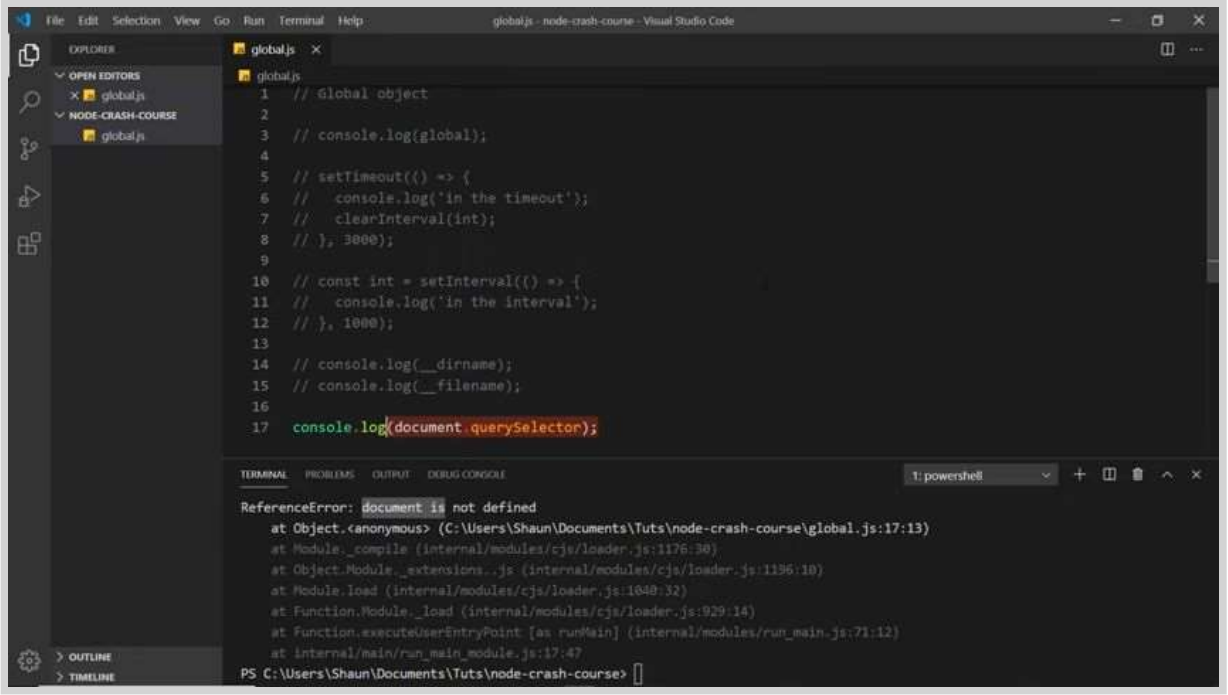

## __dirname = absolute path of current folder

## __filename = absolute path of folder with file name




## document. nahi use kru shkt coz to global object nahiyee...node madhe

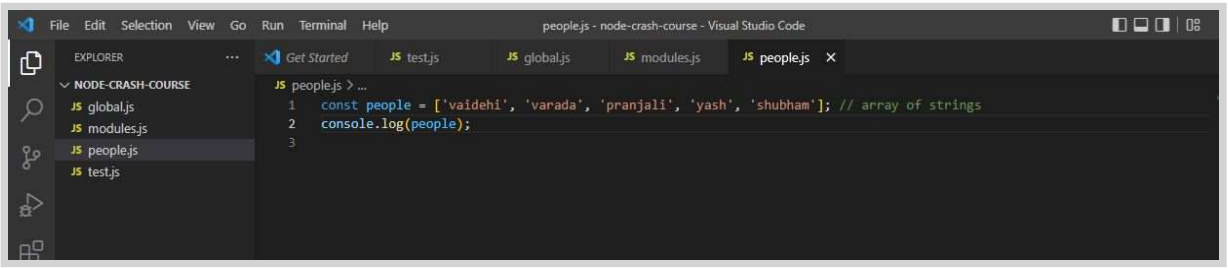
09:13

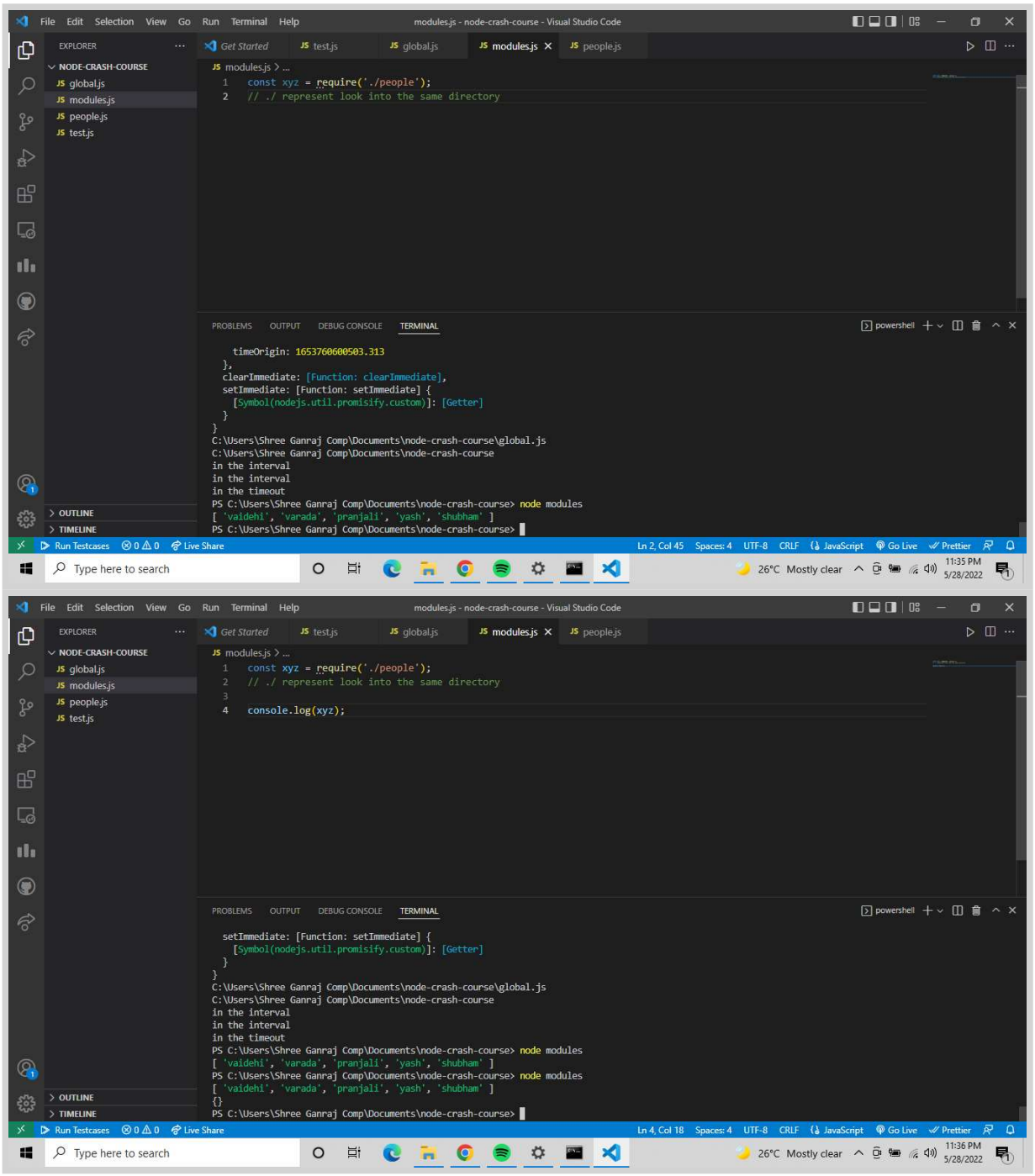



## Modules and Require:

## reusable banvu shkto

## hi people wali file import kraychiy module.js madhe... tr require statement lagel



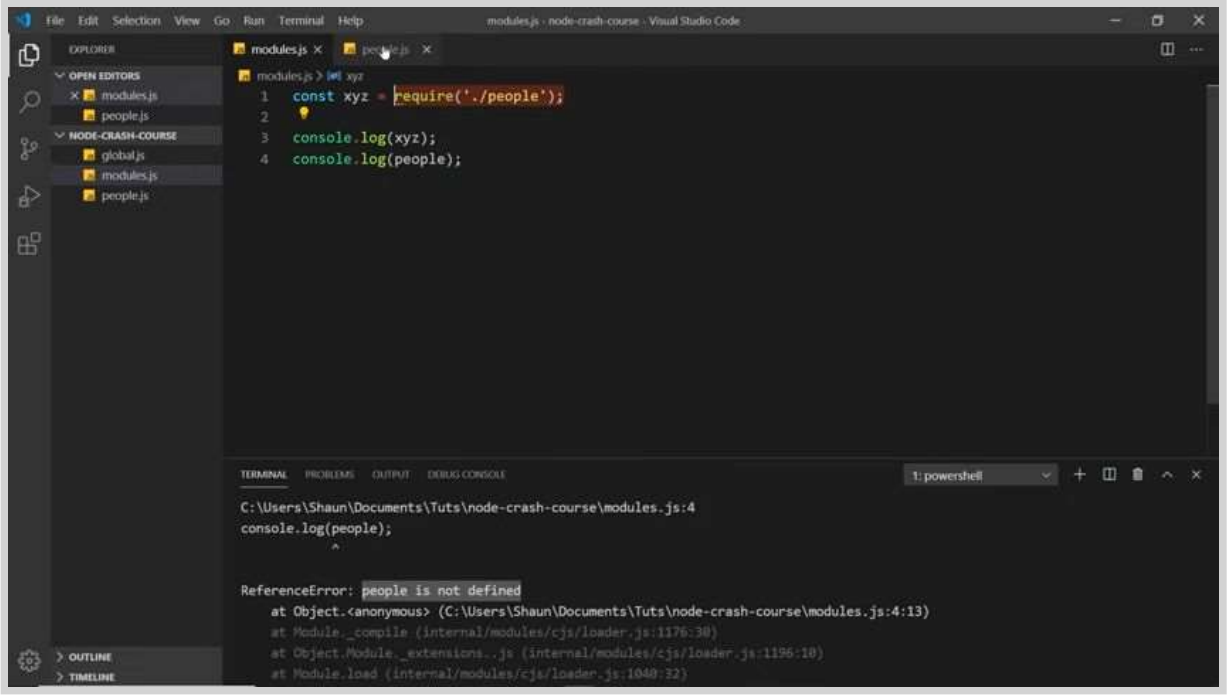



**xyz i.e, module file is returning empty object**

**but why?=>**

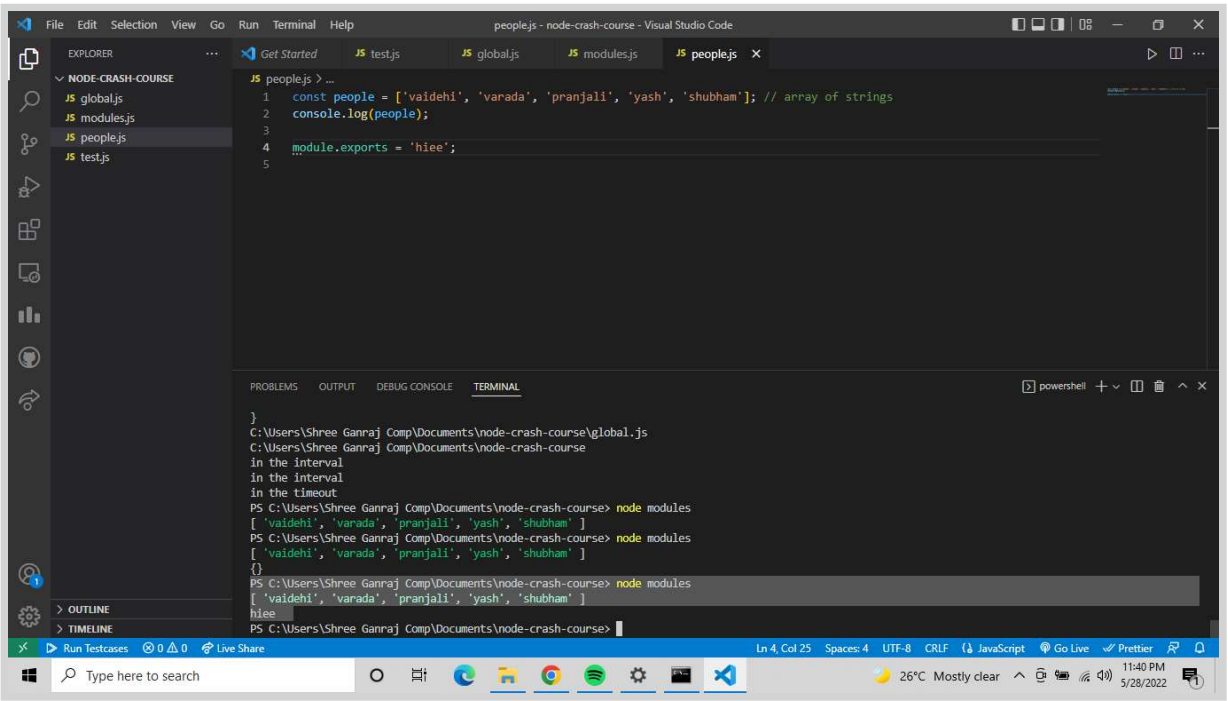
**and also nuste import krun aapn variables wagaire access nahi kru shkt hot nahi**

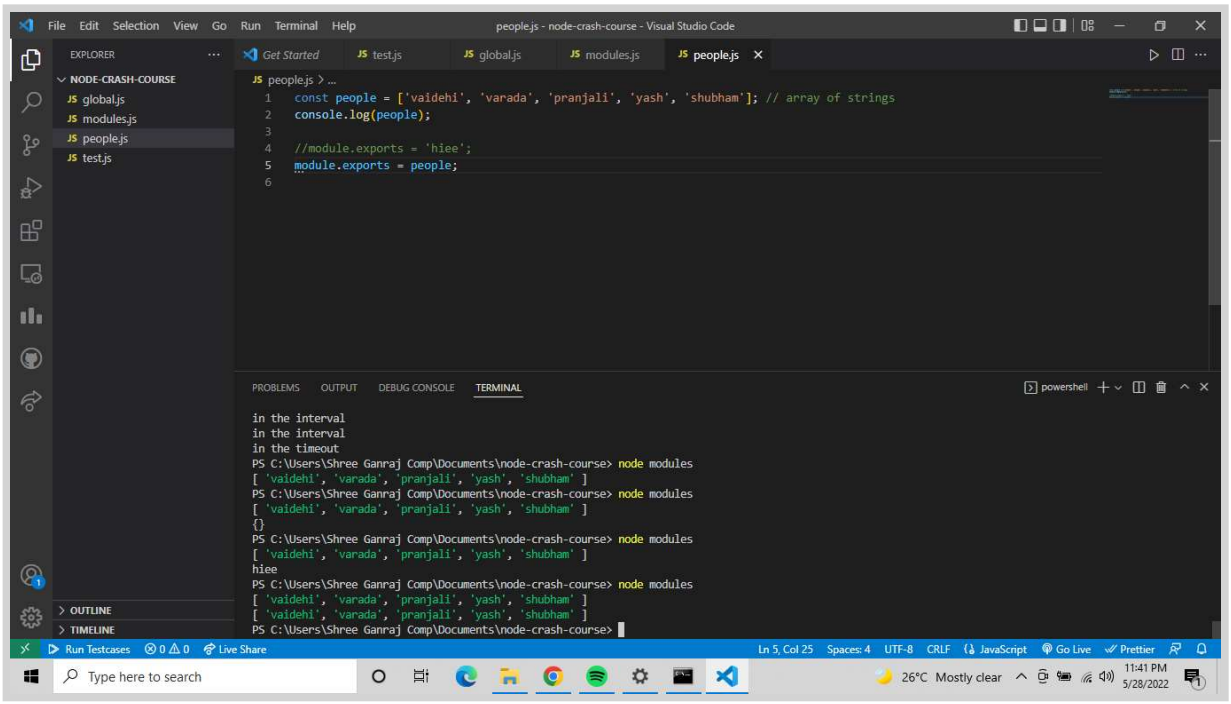
[12:01]




**to access people inside modules=>**

**we have to export**



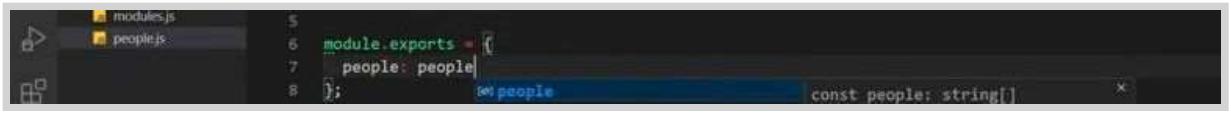


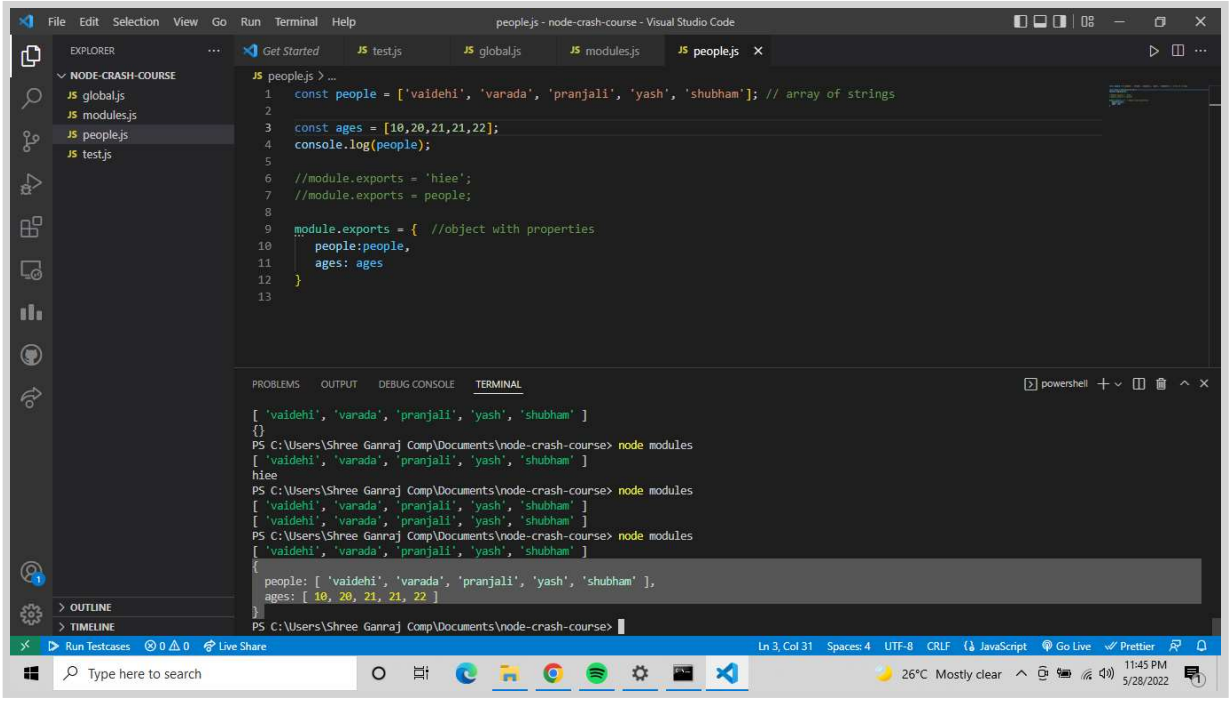

# that's how we re exporting single thing

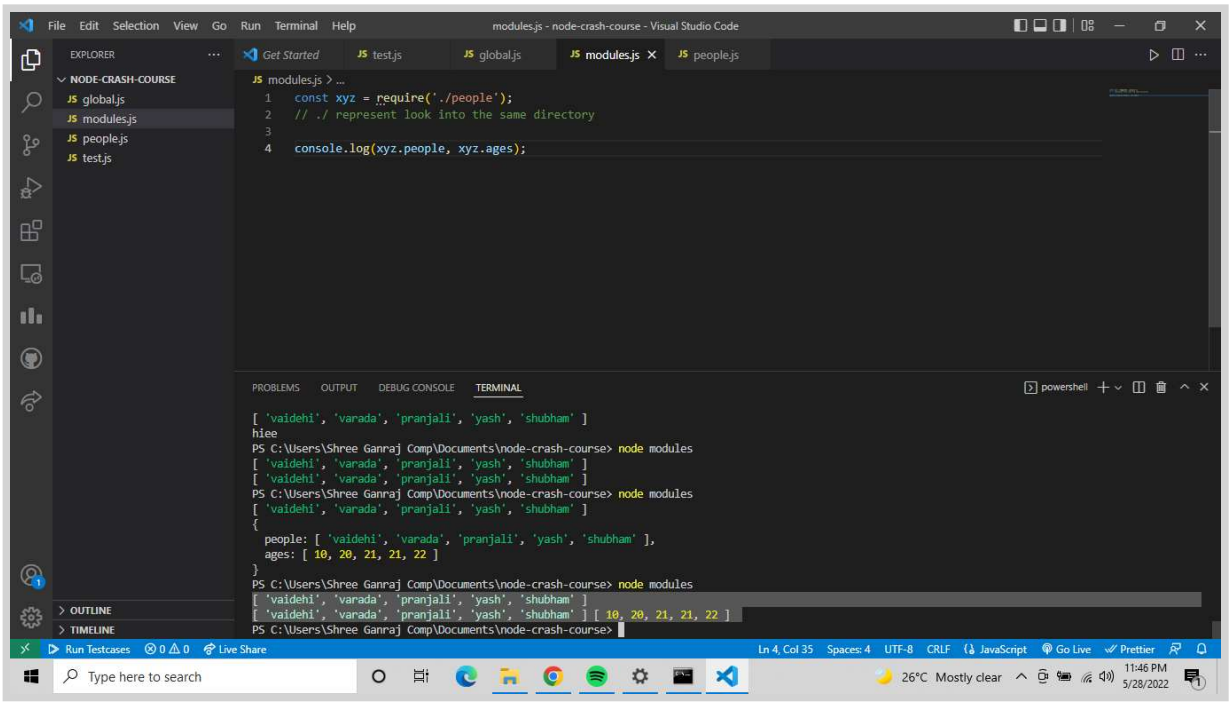
# now how we can export multiple things??

# =>

## 14:50

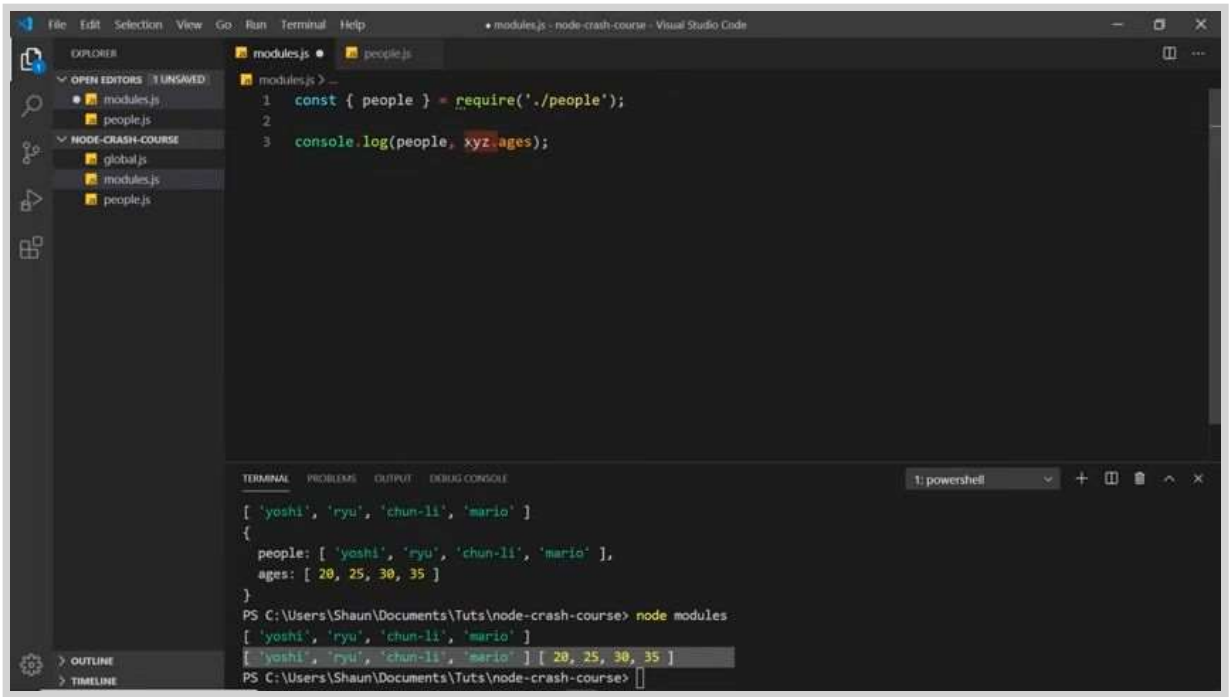


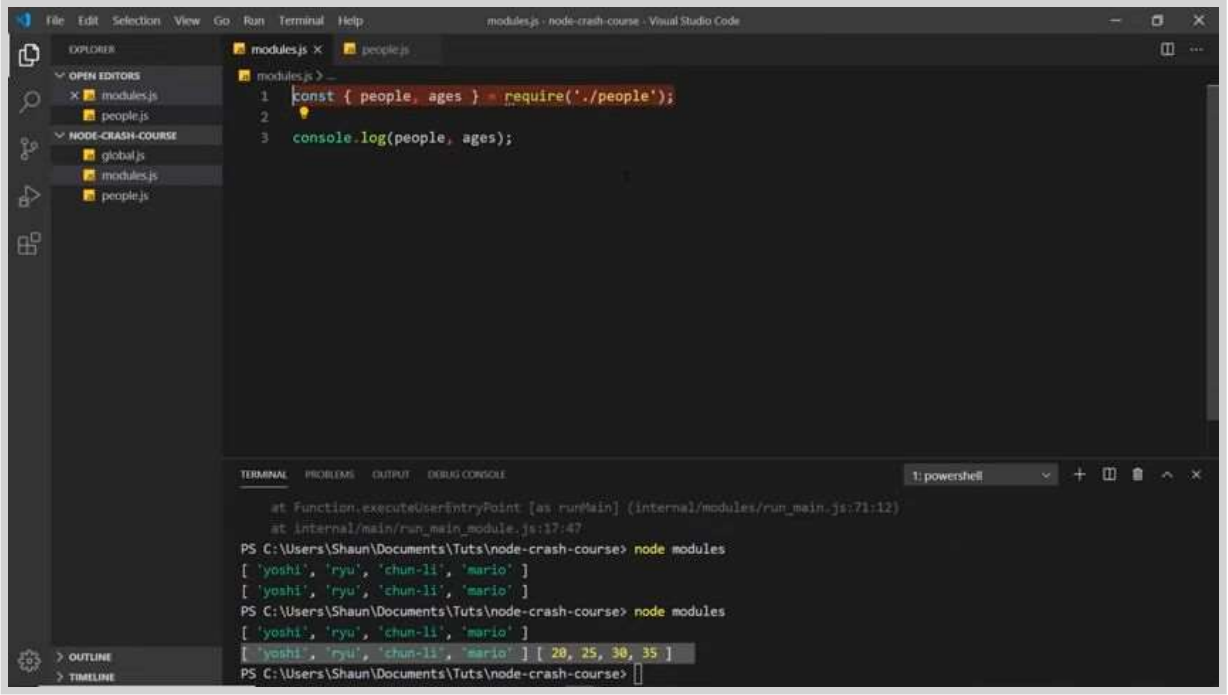


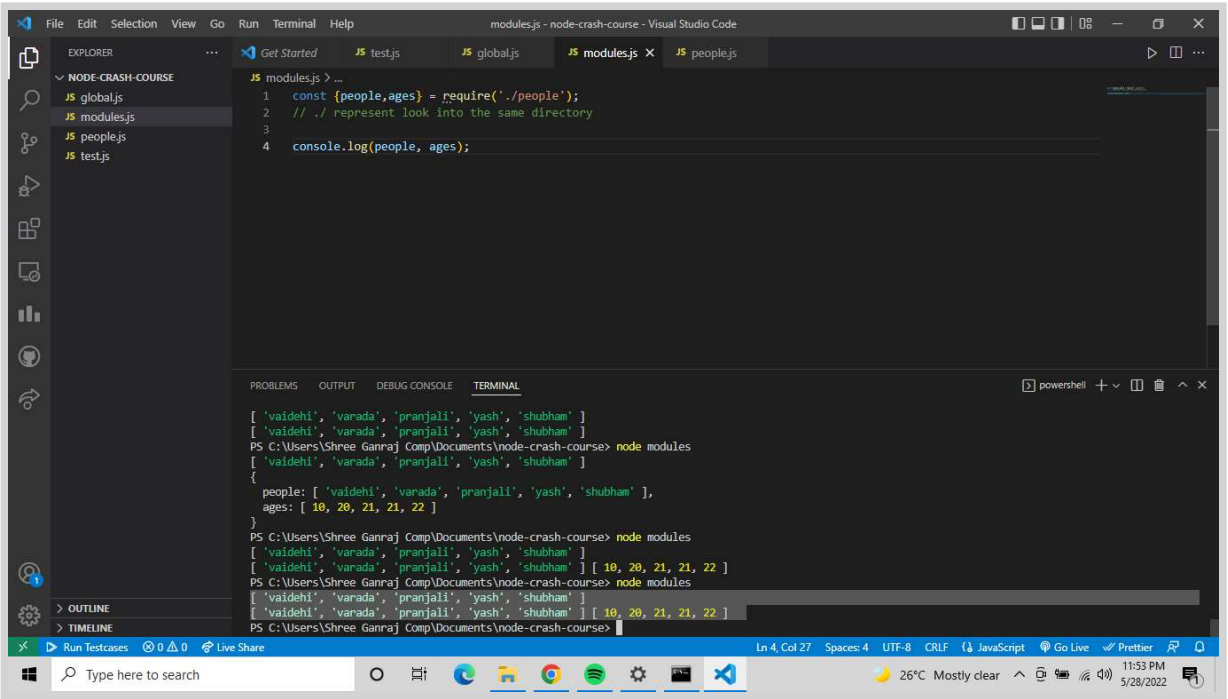

# importing from multiple files sathi we will use destructuring...

[16:21](#)



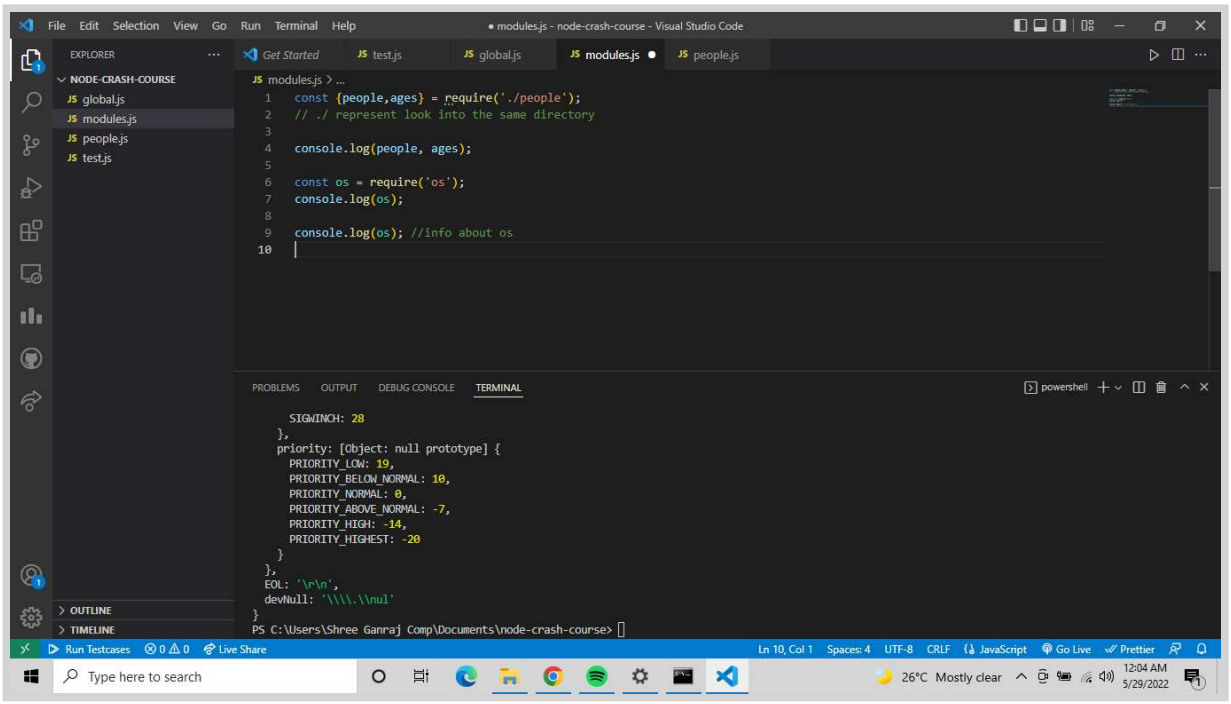

[17:00](#)



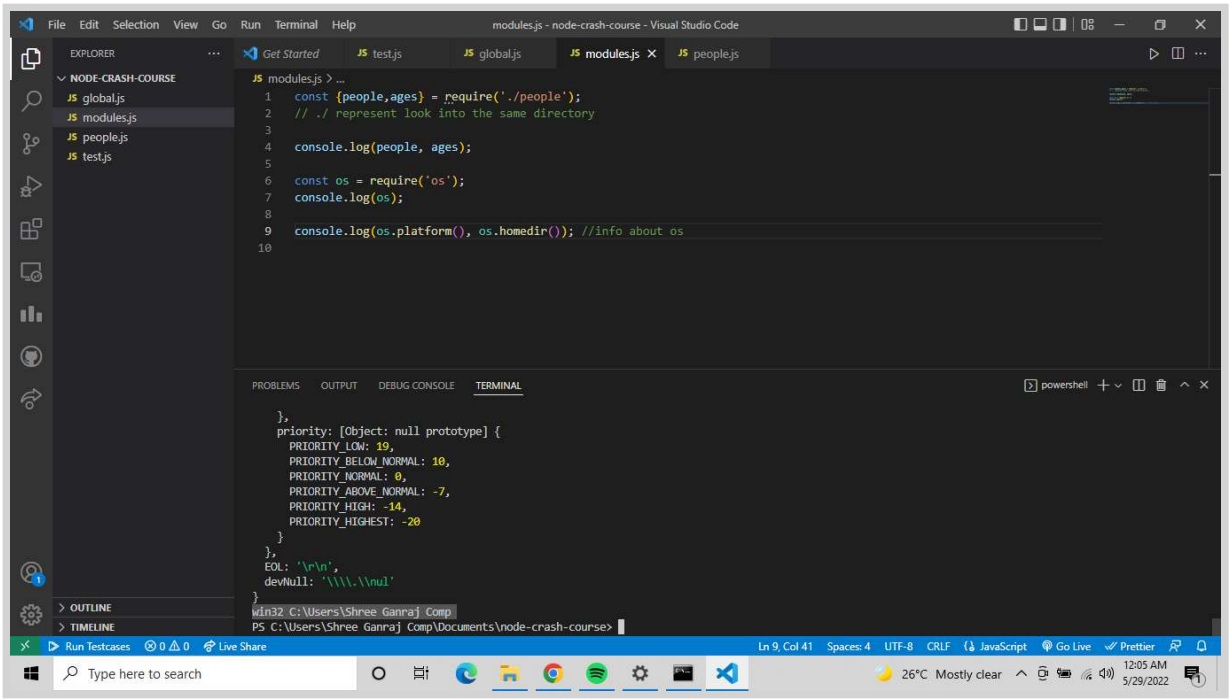




# build in modules astat kahi

# js ki "os";



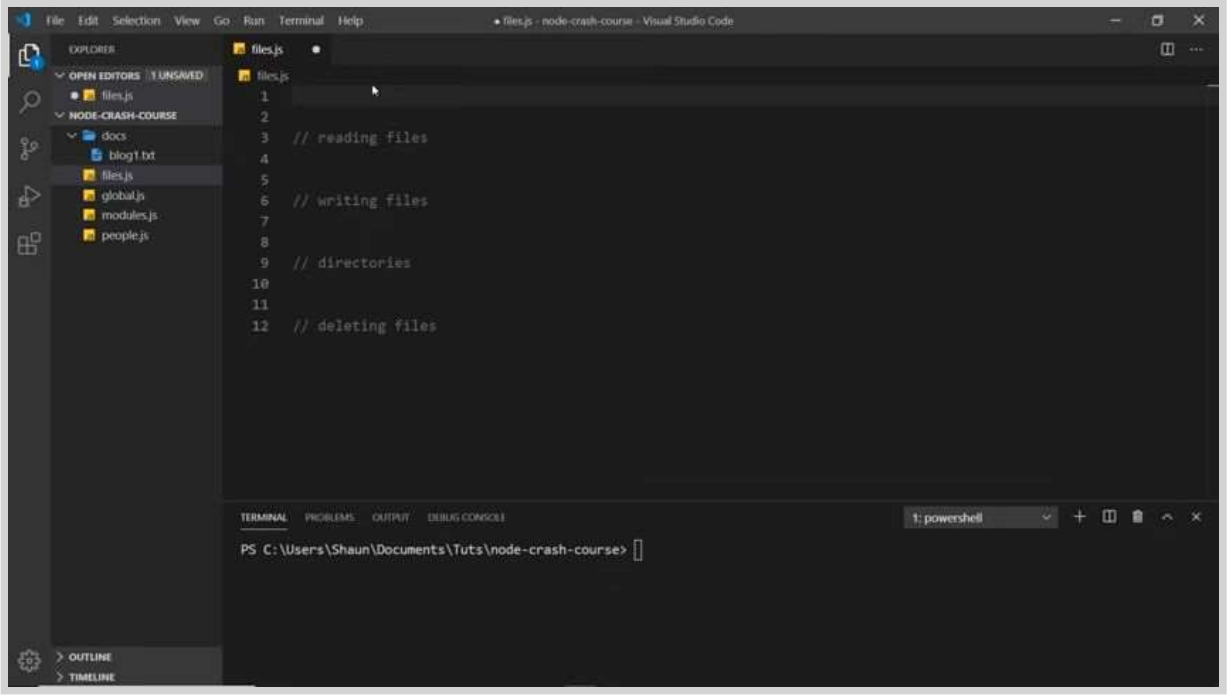

[18:23]

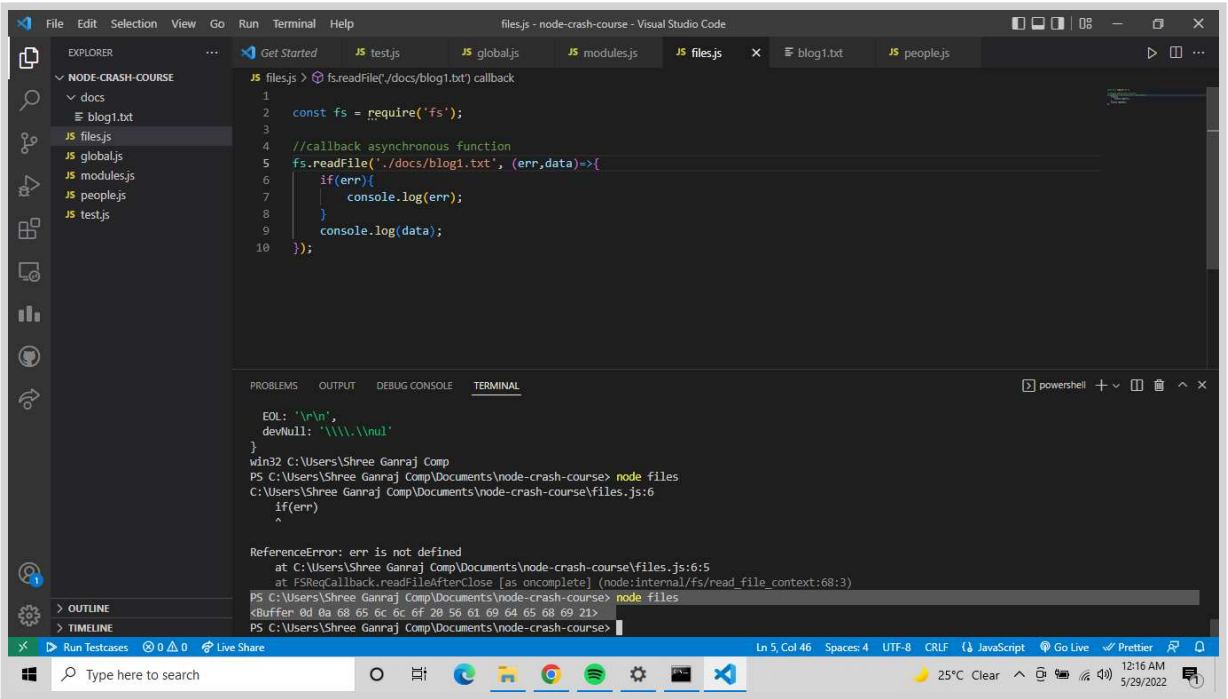


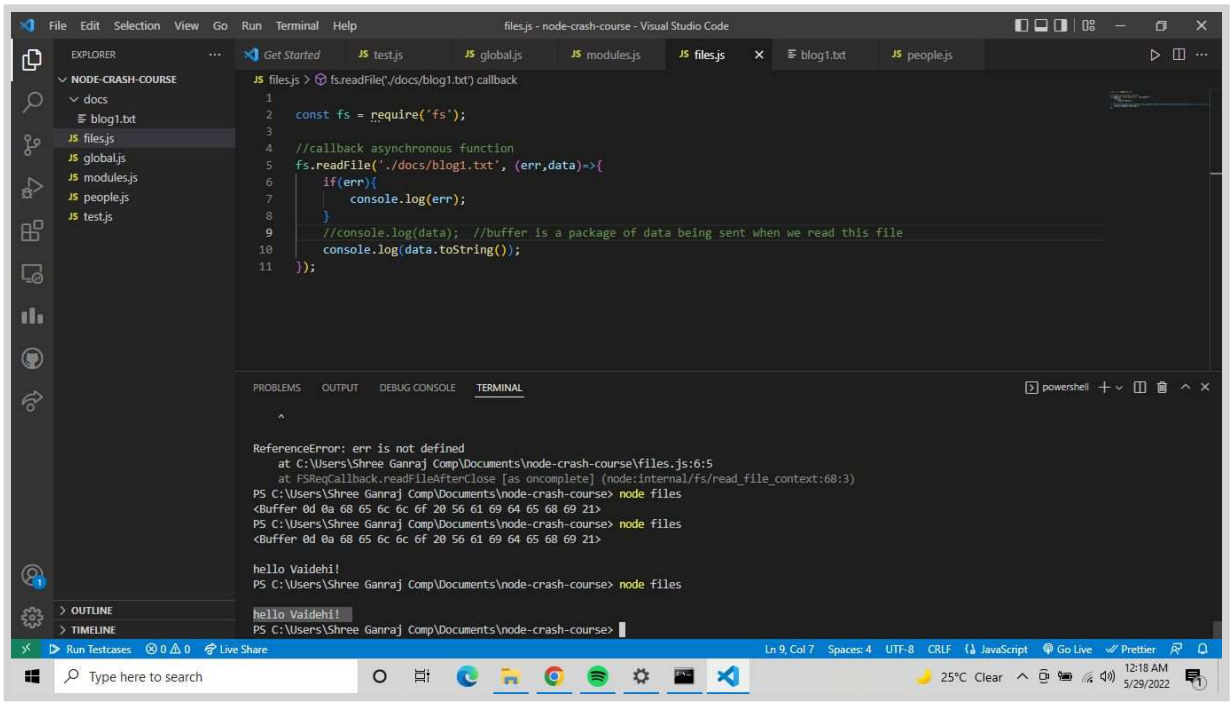

# The file system...modules use kru shkto using node..

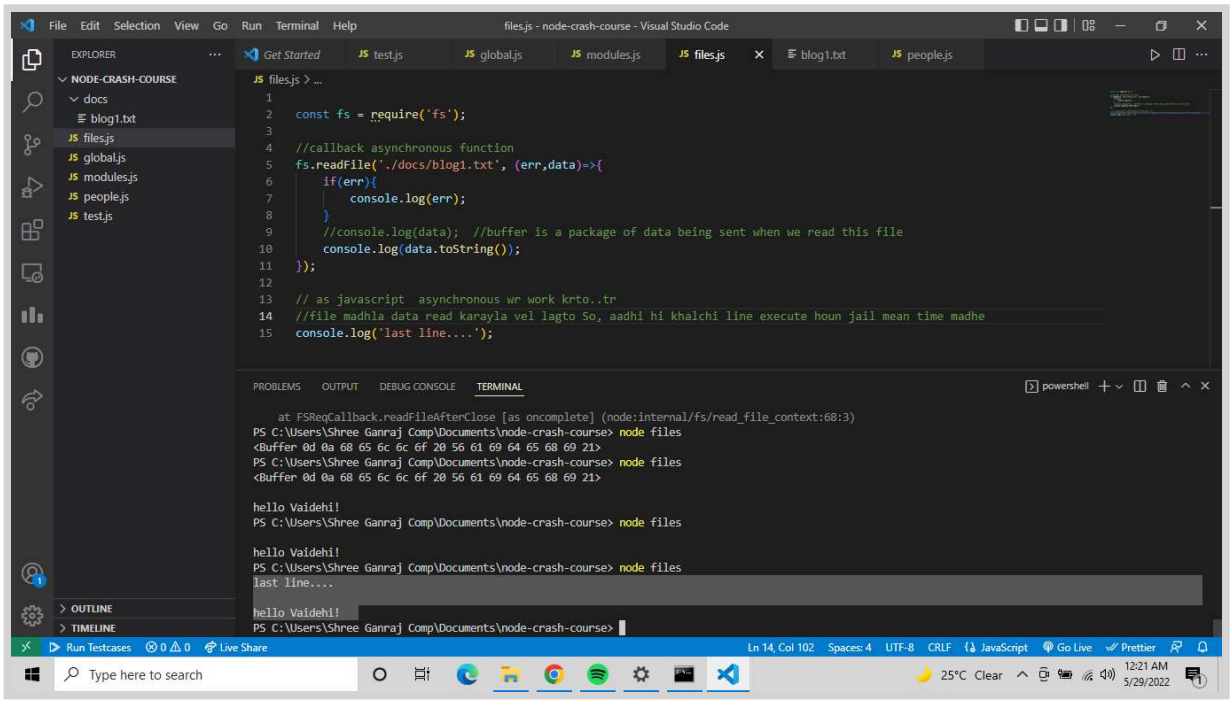
# fs module which is used for file system

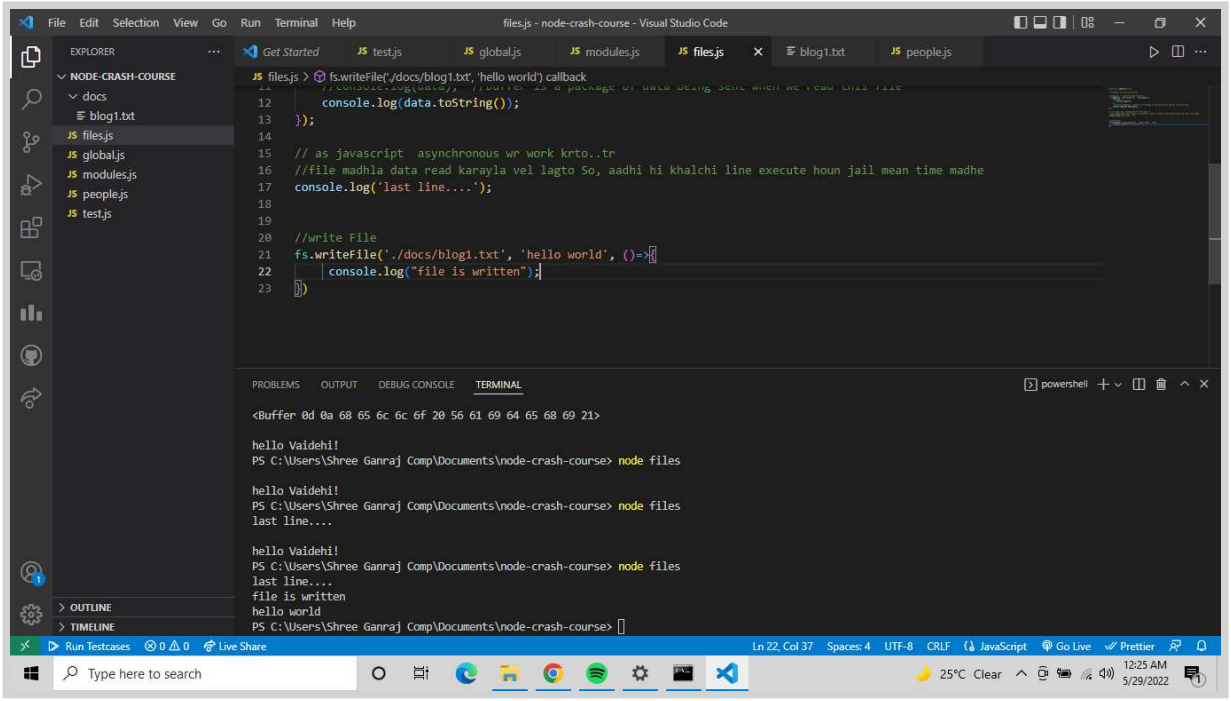
[19:33]

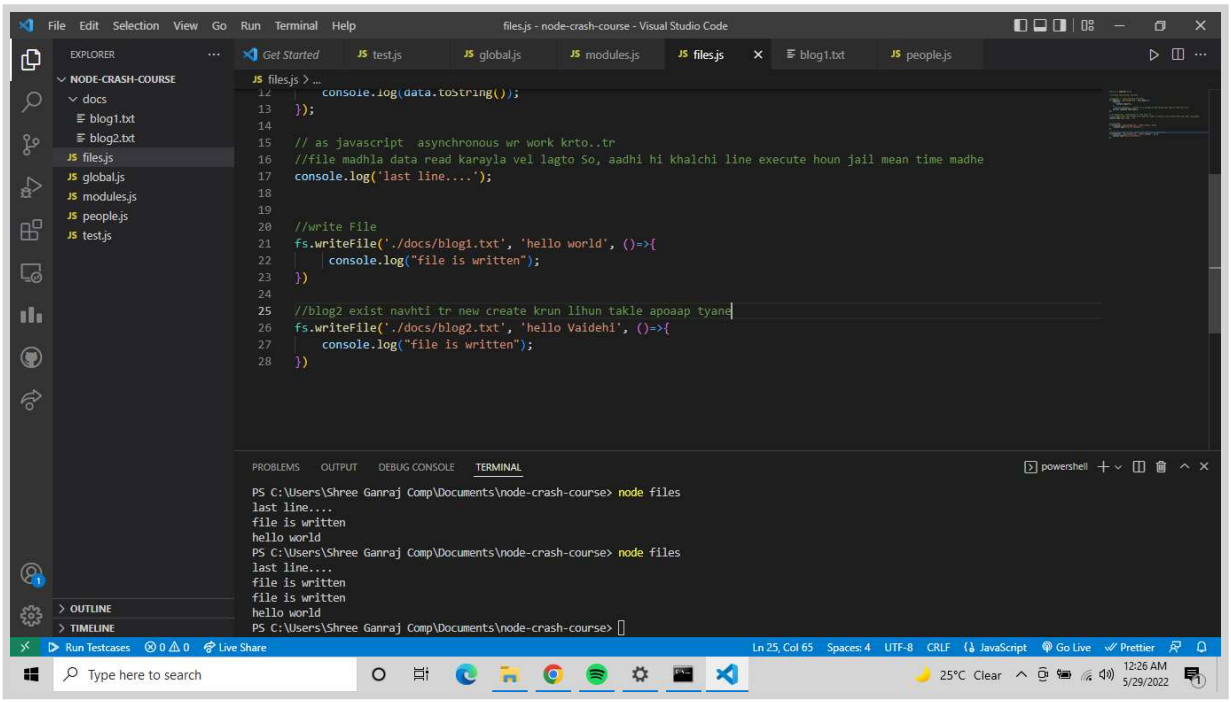

files.js - node-crash-course - Visual Studio Code
NODE-CRASH-COURSE
docs
blog1.txt
files.js
global.js
modules.js
people.js
// reading files
// writing files
// directories
// deleting files
TERMINAL
PROBLEMS
OUTPUT
DEBUG CONSOLE
1: powershell
PS C:\Users\Shaun\Documents\Tuts\node-crash-course>


files.js - node-crash-course - Visual Studio Code
const fs = require('fs');
//callback asynchronous function
fs.readFile('./docs/blog1.txt', (err,data)=>{
if(err){
console.log(err);
}
console.log(data);
});
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
powershell
win32 C:\Users\Shree Ganraj Comp
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node files
C:\Users\Shree Ganraj Comp\Documents\node-crash-course\files.js:6
if(err)
^
ReferenceError: err is not defined
at C:\Users\Shree Ganraj Comp\Documents\node-crash-course\files.js:6:5
at FSReqCallback.readFileAfterClose [as oncomplete] (node:internal/fs/read_file_context:68:3)
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node files
<Buffer 0d 0a 68 65 6c 6c 6f 20 56 61 69 64 65 68 69 21>
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course>
Type here to search
25°C Clear
12:16 AM
5/29/2022


files.js - node-crash-course - Visual Studio Code
NODE-CRASH-COURSE
docs
blog1.txt
files.js
global.js
modules.js
people.js
test.js
const fs = require('fs');
//callback asynchronous function
fs.readFile('./docs/blog1.txt', (err,data)=>{
if(err){
console.log(err);
}
//console.log(data); //buffer is a package of data being sent when we read this file
console.log(data.toString());
});
ReferenceError: err is not defined
at C:\Users\Shree Ganraj Comp\Documents\node-crash-course\files.js:6:5
at FSReqCallback.readFileAfterClose [as oncomplete] (node:internal/fs/read_file_context:68:3)
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node files
<Buffer 0d 0a 68 65 6c 6c 6f 20 56 61 69 64 65 68 69 21>
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node files
<Buffer 0d 0a 68 65 6c 6c 6f 20 56 61 69 64 65 68 69 21>
hello Vaidehi!
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node files
hello Vaidehi!
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course>


files.js - node-crash-course - Visual Studio Code
NODE-CRASH-COURSE
docs
blog1.txt
files.js
global.js
modules.js
people.js
test.js
const fs = require('fs');
//callback asynchronous function
fs.readFile('./docs/blog1.txt', (err,data)=>{
if(err){
console.log(err);
}
//console.log(data); //buffer is a package of data being sent when we read this file
console.log(data.toString());
});
// as javascript asynchronous wr work krto..tr
//file madhla data read karayla vel lagto So, aadhi hi khalchi line execute houn jail mean time madhe
console.log('last line....');
at FSReqCallback.readFileAfterClose [as oncomplete] (node:internal/fs/read_file_context:68:3)
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node files
<Buffer 0d 0a 68 65 6c 6c 6f 20 56 61 69 64 65 68 69 21>
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node files
<Buffer 0d 0a 68 65 6c 6c 6f 20 56 61 69 64 65 68 69 21>
hello Vaidehi!
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node files
hello Vaidehi!
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node files
last line....
hello Vaidehi!
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course>

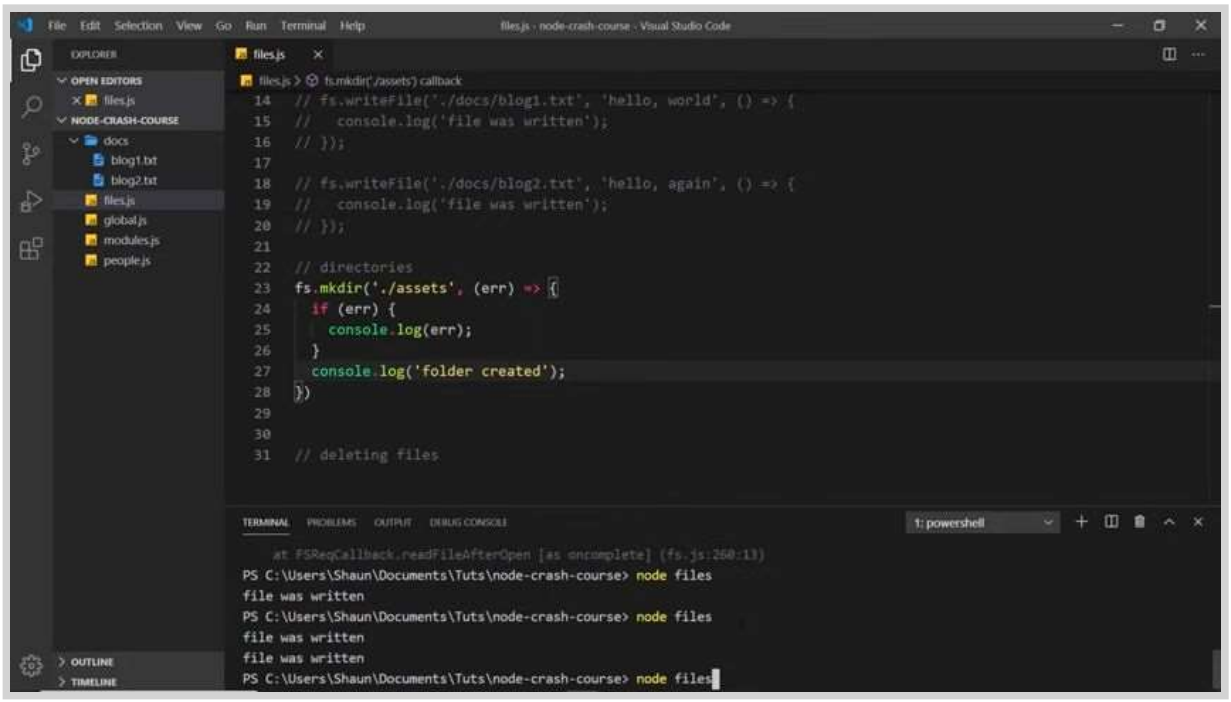


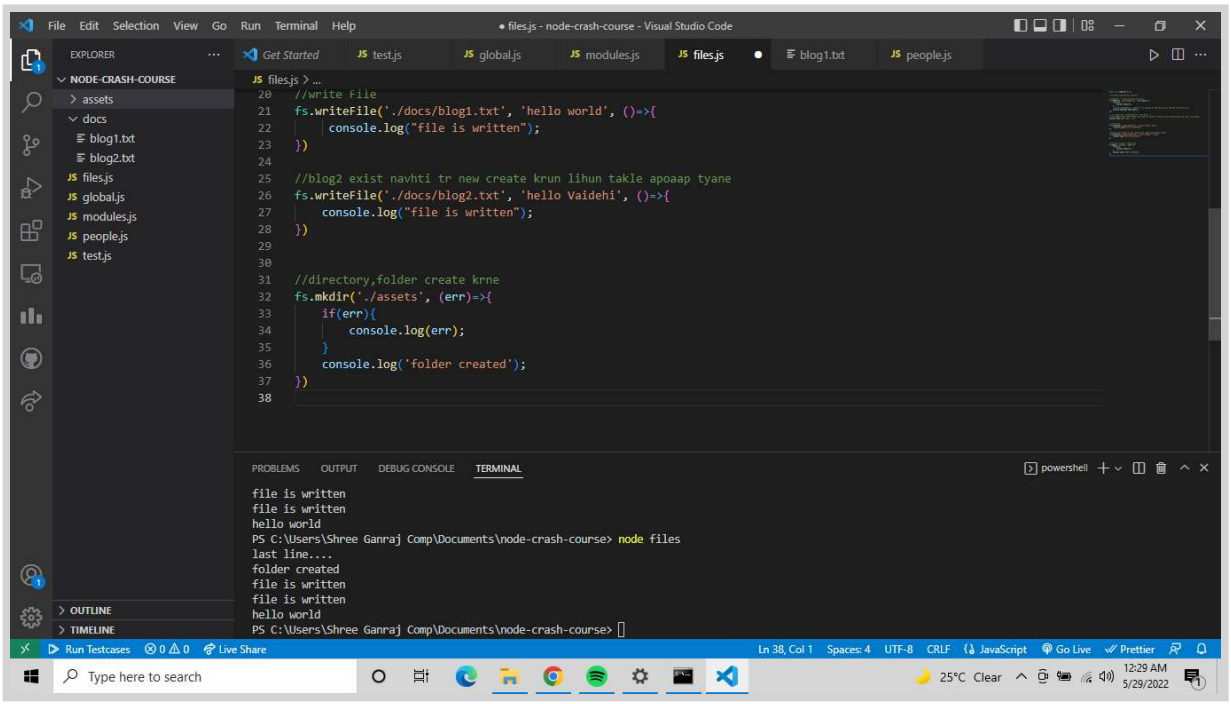



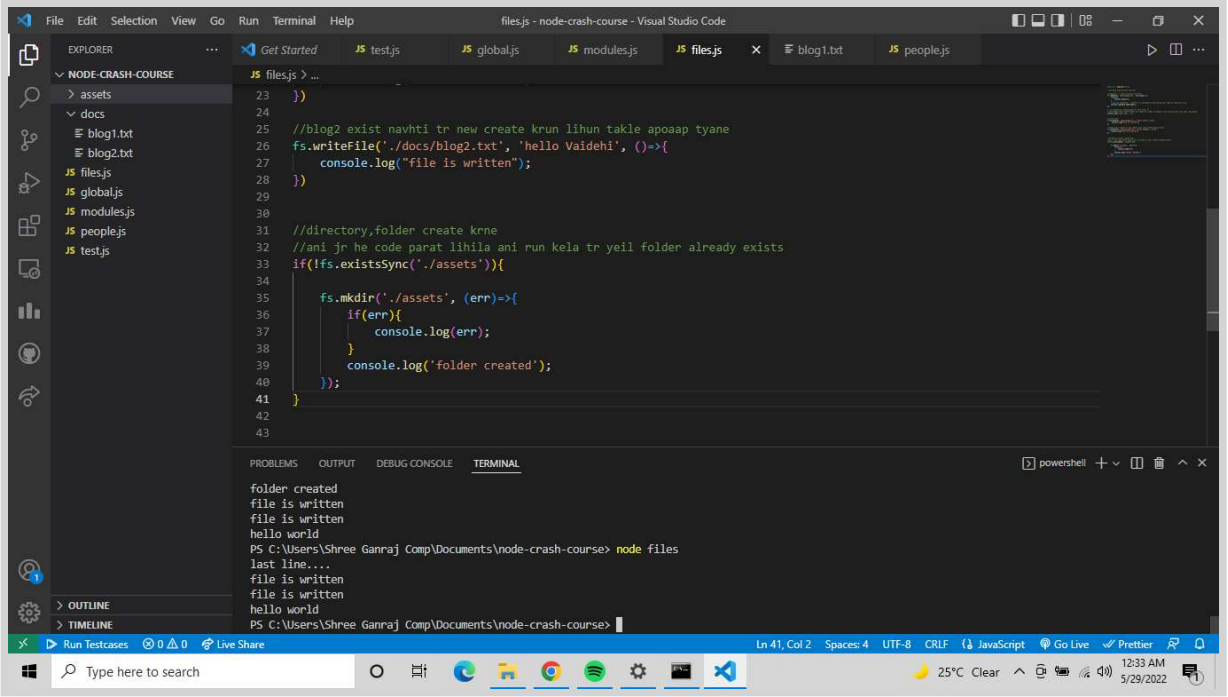

# //directories:

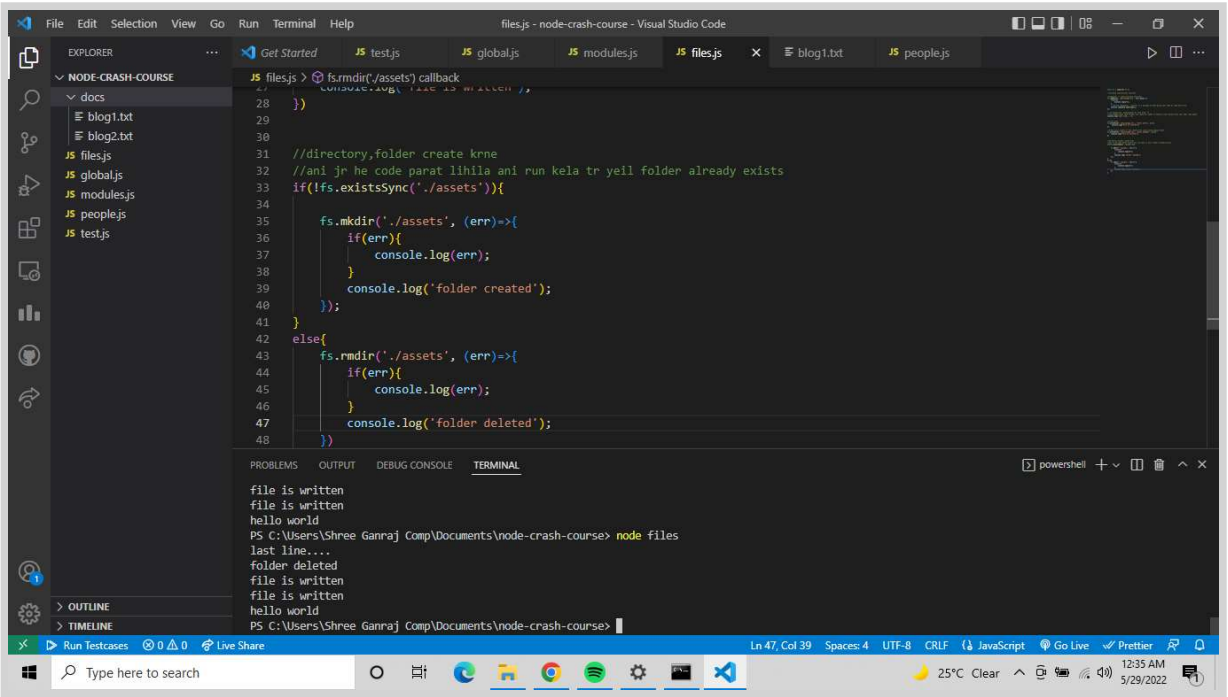
# asynchronous tasks aahet he sagle and it takes some time..

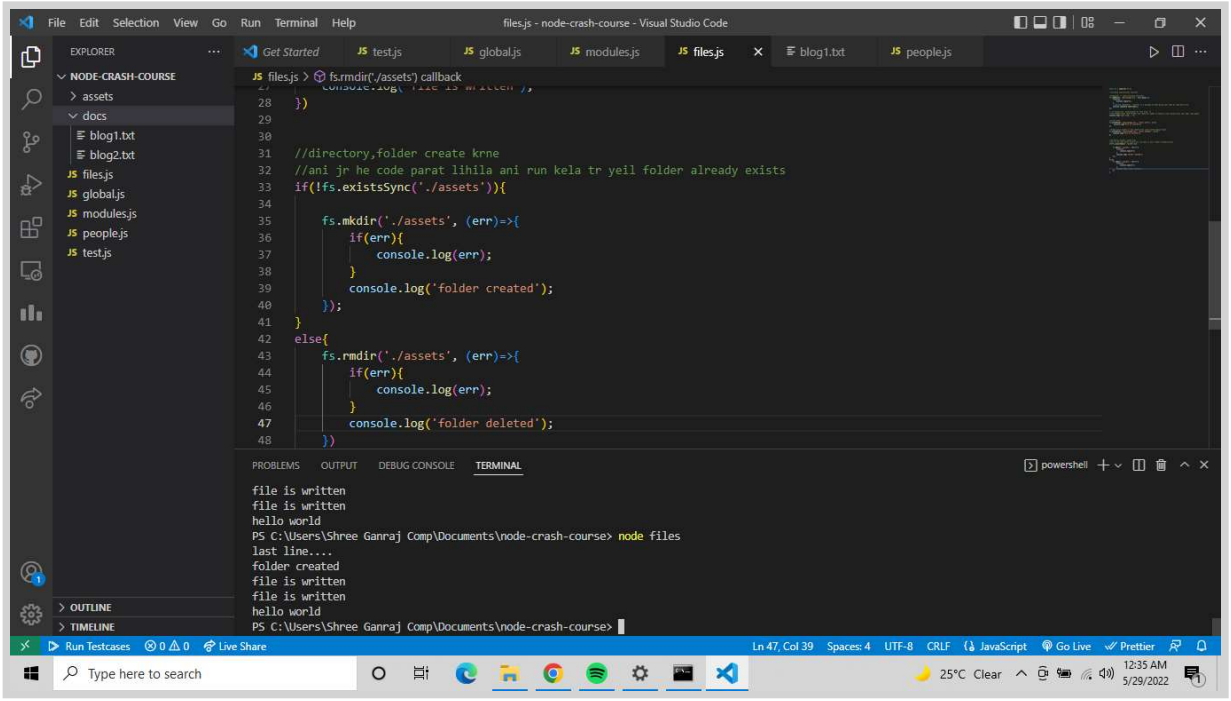
[26:16]

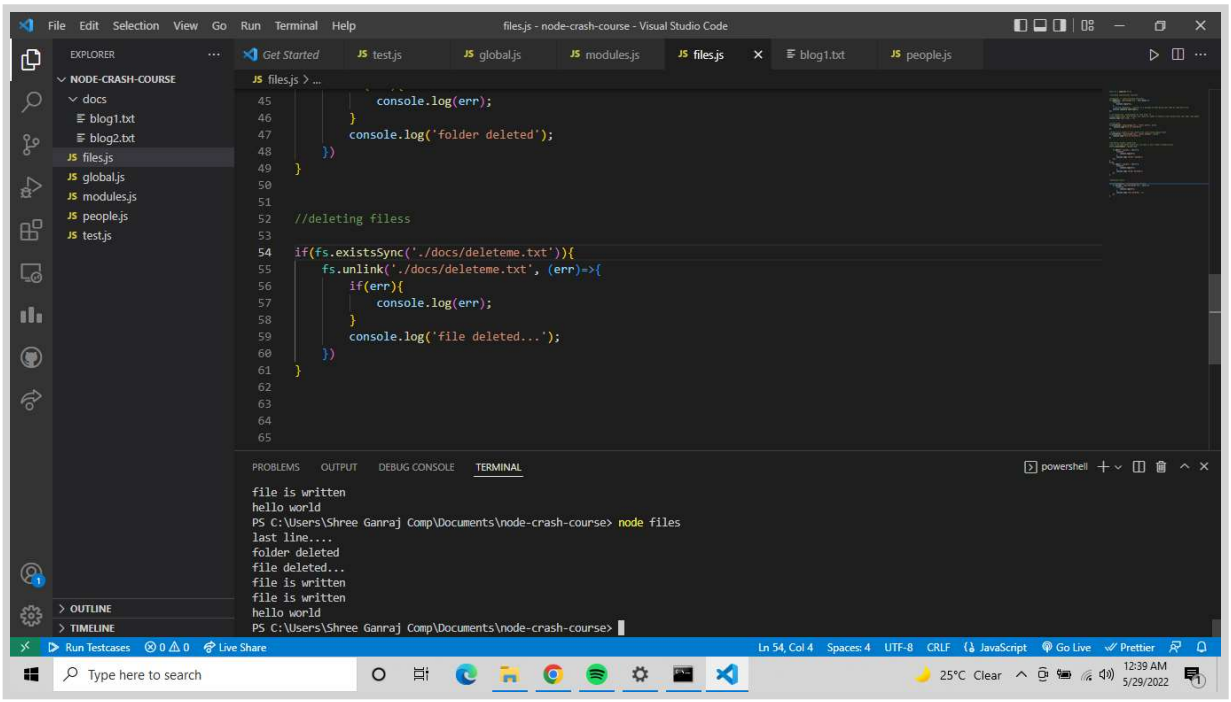

files.js - node-crash-course - Visual Studio Code
EXPLORER
OPEN EDITORS
files.js
NODE-CRASH-COURSE
docs
blog1.txt
blog2.txt
files.js
global.js
modules.js
people.js
// fs.writeFile('./docs/blog1.txt', 'hello, world', () => {
// console.log('file was written');
// });
// fs.writeFile('./docs/blog2.txt', 'hello, again', () => {
// console.log('file was written');
// });
// directories
fs.mkdir('./assets', (err) => {
if (err) {
console.log(err);
}
console.log('folder created');
})
// deleting files
TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE
1: powershell
PS C:\Users\Shaun\Documents\Tuts\node-crash-course> node files
file was written
PS C:\Users\Shaun\Documents\Tuts\node-crash-course> node files
file was written
file was written
PS C:\Users\Shaun\Documents\Tuts\node-crash-course> node files
OUTLINE
TIMELINE


• files.js - node-crash-course - Visual Studio Code
EXPLORER
NODE-CRASH-COURSE
assets
docs
blog1.txt
blog2.txt
files.js
global.js
modules.js
people.js
test.js
Get Started
test.js
global.js
modules.js
files.js
blog1.txt
people.js
//write File
fs.writeFile('./docs/blog1.txt', 'hello world', ()=>{
console.log("file is written");
})
//blog2 exist navhti tr new create krun lihun takle apoaap tyane
fs.writeFile('./docs/blog2.txt', 'hello Vaidehi', ()=>{
console.log("file is written");
})
//directory,folder create krne
fs.mkdir('./assets', (err)=>{
if(err){
console.log(err);
}
console.log('folder created');
})
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
powershell
file is written
file is written
hello world
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node files
last line....
folder created
file is written
file is written
hello world
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course>
OUTLINE
TIMELINE
Run Testcases
Live Share
Ln 38, Col 1
Spaces: 4
UTF-8
CRLF
JavaScript
Go Live
Prettier
Type here to search
25°C Clear
12:29 AM
5/29/2022


files.js - node-crash-course - Visual Studio Code
NODE-CRASH-COURSE
assets
docs
blog1.txt
blog2.txt
files.js
global.js
modules.js
people.js
test.js
//blog2 exist navhti tr new create krun lihun takle apoaap tyane
fs.writeFile('./docs/blog2.txt', 'hello Vaidehi', ()=>{
console.log("file is written");
})
//directory,folder create krne
//ani jr he code parat lihila ani run kela tr yeil folder already exists
if(!fs.existsSync('./assets')){
fs.mkdir('./assets', (err)=>{
if(err){
console.log(err);
}
console.log('folder created');
});
}
folder created
file is written
file is written
hello world
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node files
last line....
file is written
file is written
hello world
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course>


files.js - node-crash-course - Visual Studio Code
NODE-CRASH-COURSE
docs
blog1.txt
blog2.txt
files.js
global.js
modules.js
people.js
test.js
//directory,folder create krne
//ani jr he code parat lihila ani run kela tr yeil folder already exists
if(!fs.existsSync('./assets')){
fs.mkdir('./assets', (err)=>{
if(err){
console.log(err);
}
console.log('folder created');
});
}
else{
fs.rmdir('./assets', (err)=>{
if(err){
console.log(err);
}
console.log('folder deleted');
})
file is written
file is written
hello world
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node files
last line....
folder deleted
file is written
file is written
hello world
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course>

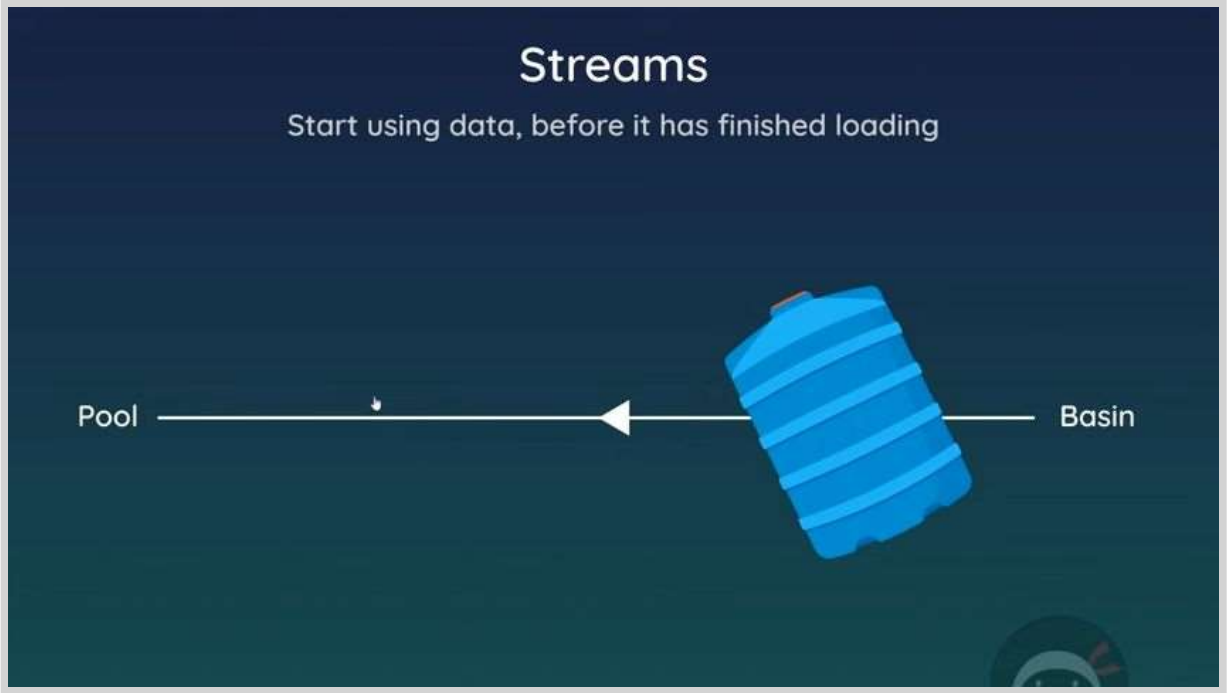



# streams & Buffers:

# streams => live streams

[32:50]



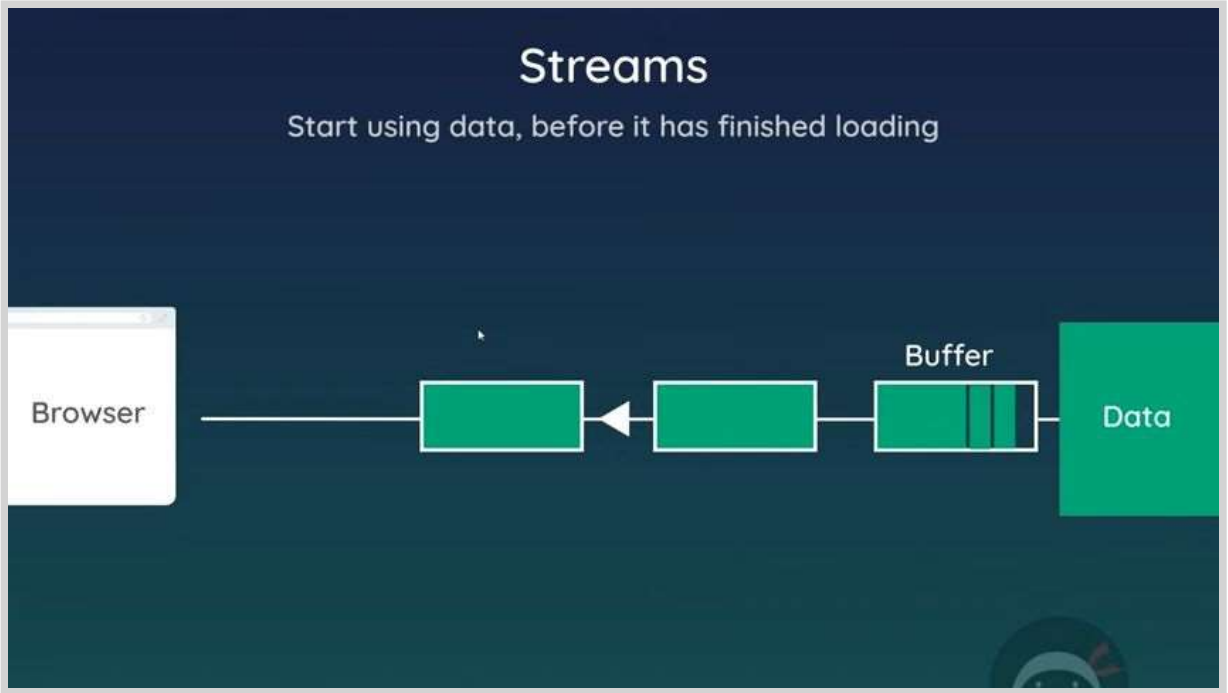

**to have a stream => so that aapn pani bhartanach pool use kru shkto**

**33:01**

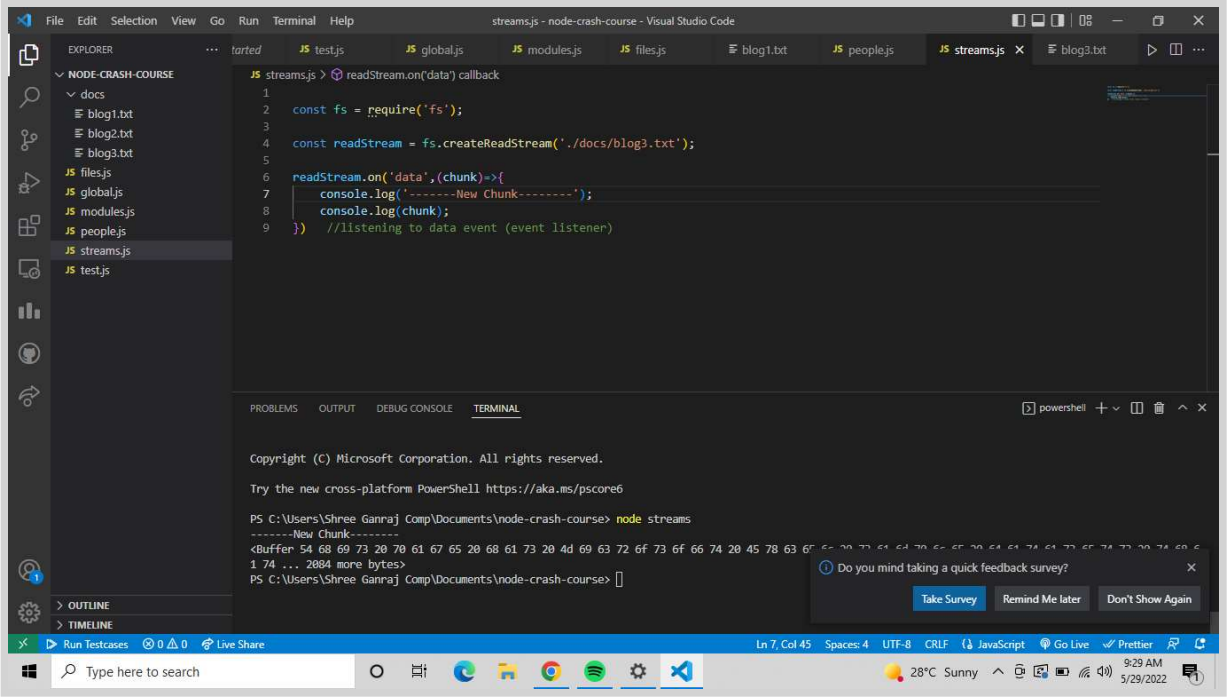

**if huge amount of file is there=>**

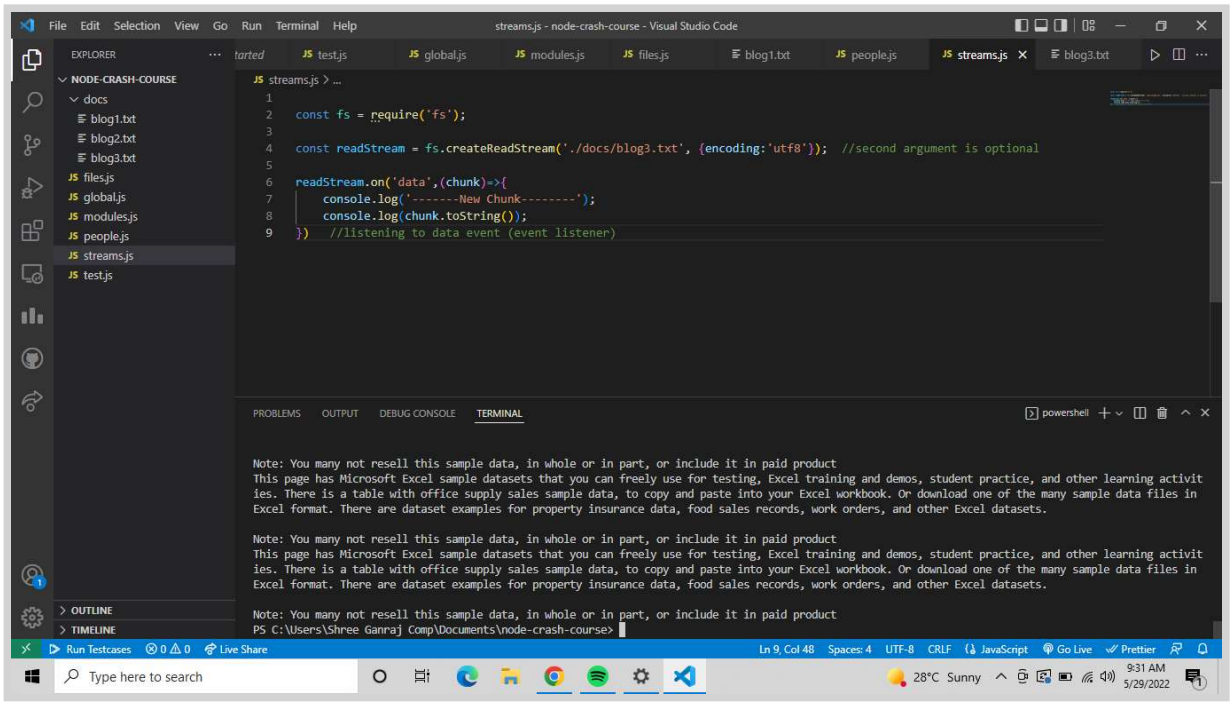
**small chunks madhe browser la detoyy so that vel nko lagayla**

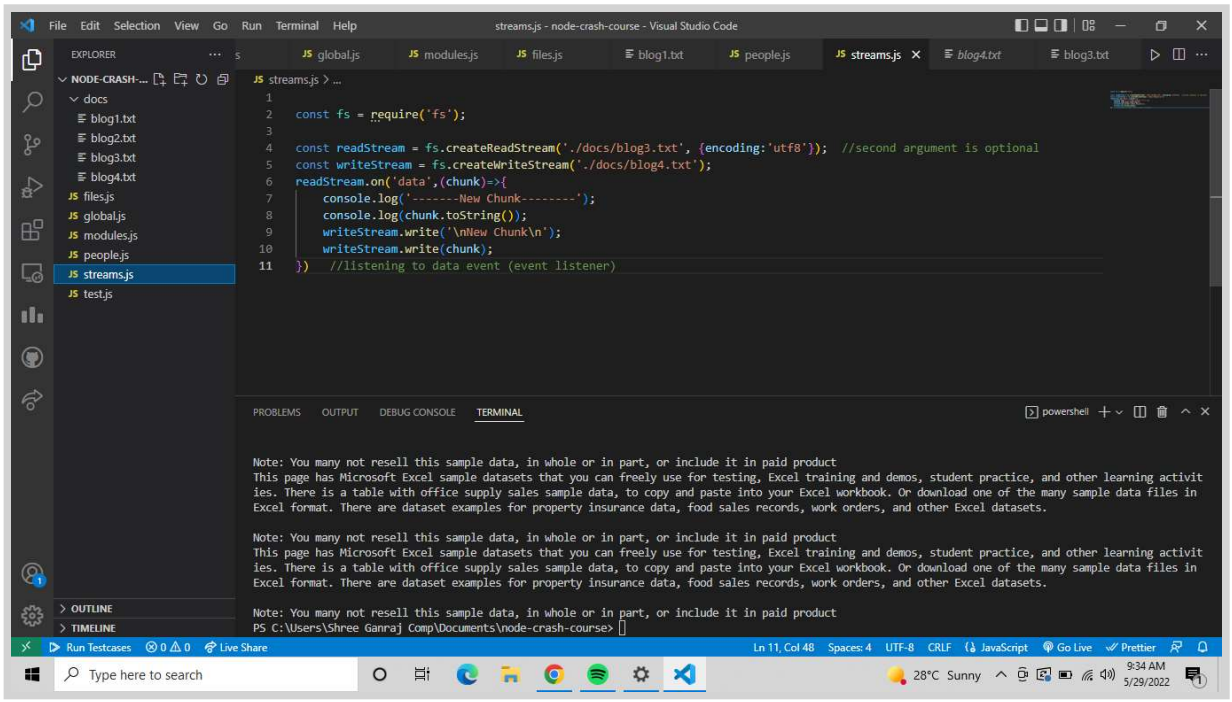
- [33:38](#)




# //read streams and write streams:

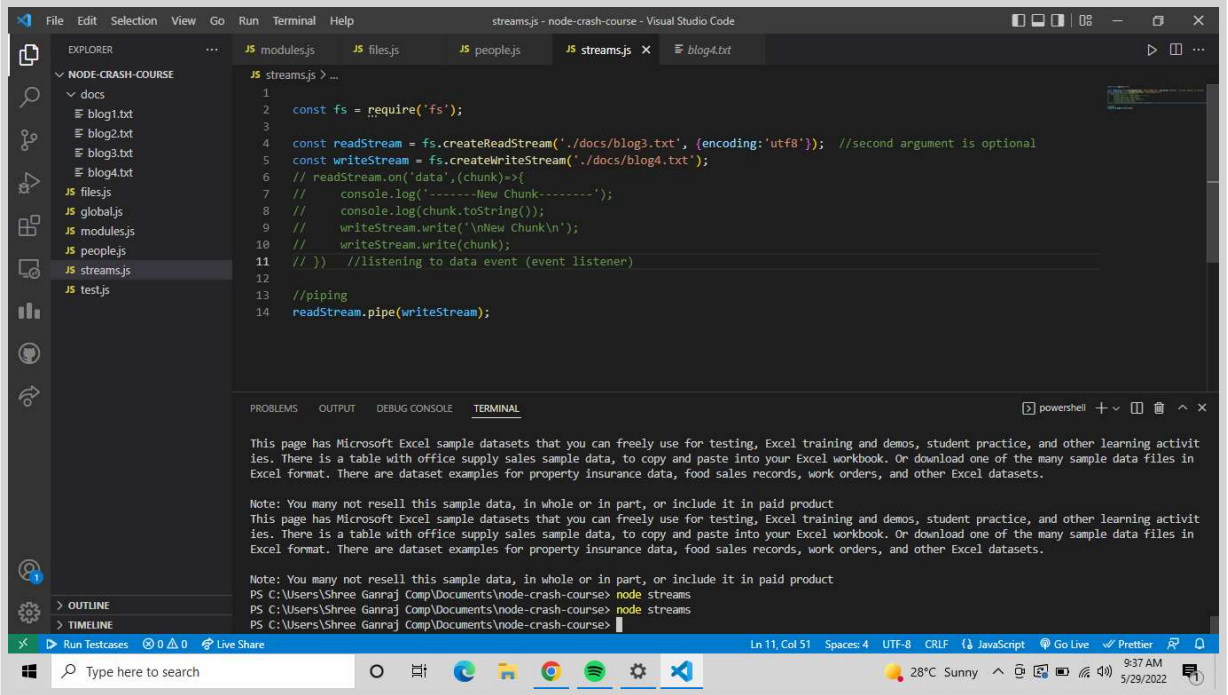





**instead of this use pipe => streams**

**everytime we getting chunk it will pipe ...that**




## duplex streams => r-w both

## Video3:

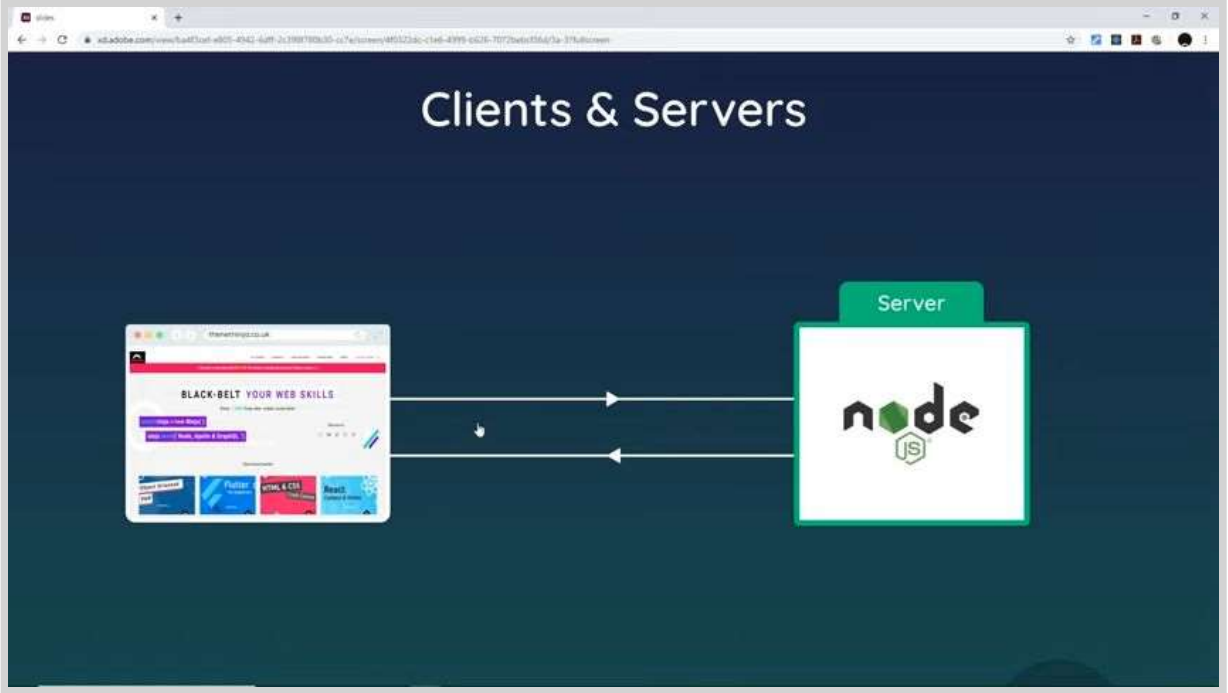
## Clients & Servers=>

## server => listenes incoming request from browser

## how this happens??

## => when we write some URL on browser and press enter that sends request to the server

[00:51]




**how server knows which server we want to connect?=>**

**IP addresses and Domains**

**address of Computer which is connect to the internet...**

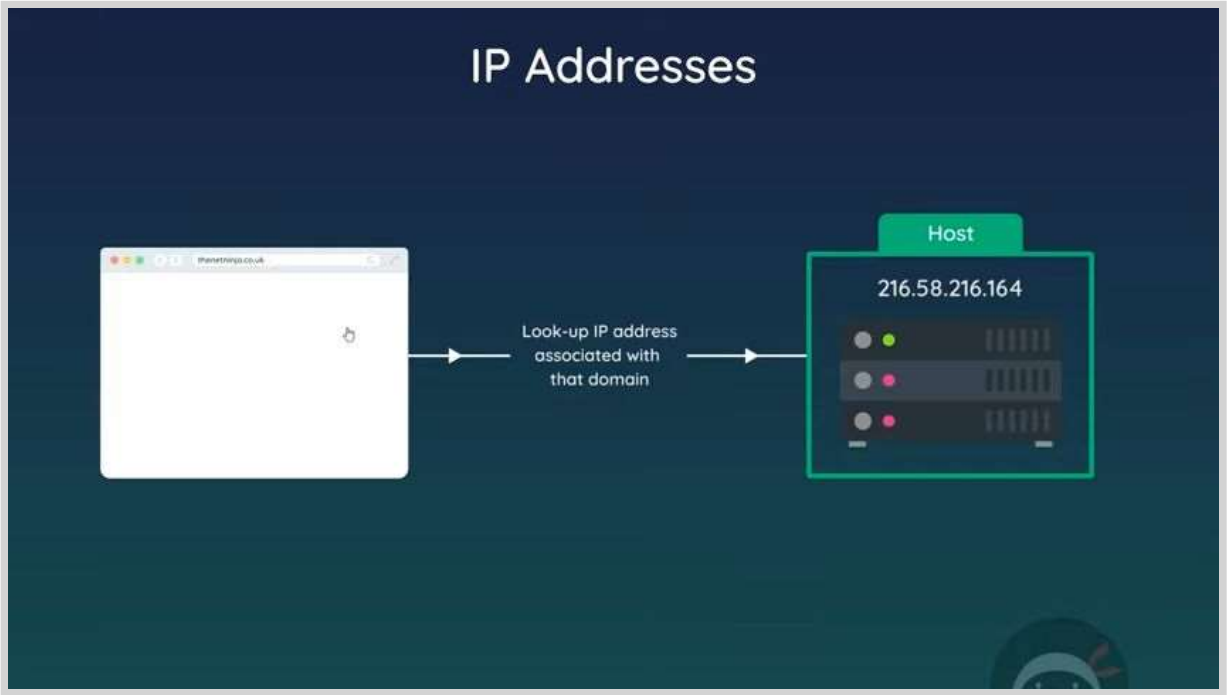
**01:51**


**if we want to connect to some computer then we should know the IP address of that**

**but as we know IP addresses are hard to remember so , we use domain name instead of IP addresses to mask this numbers/IP addresses..**

**Browser will look up the ip address associated with that domain**

02:48

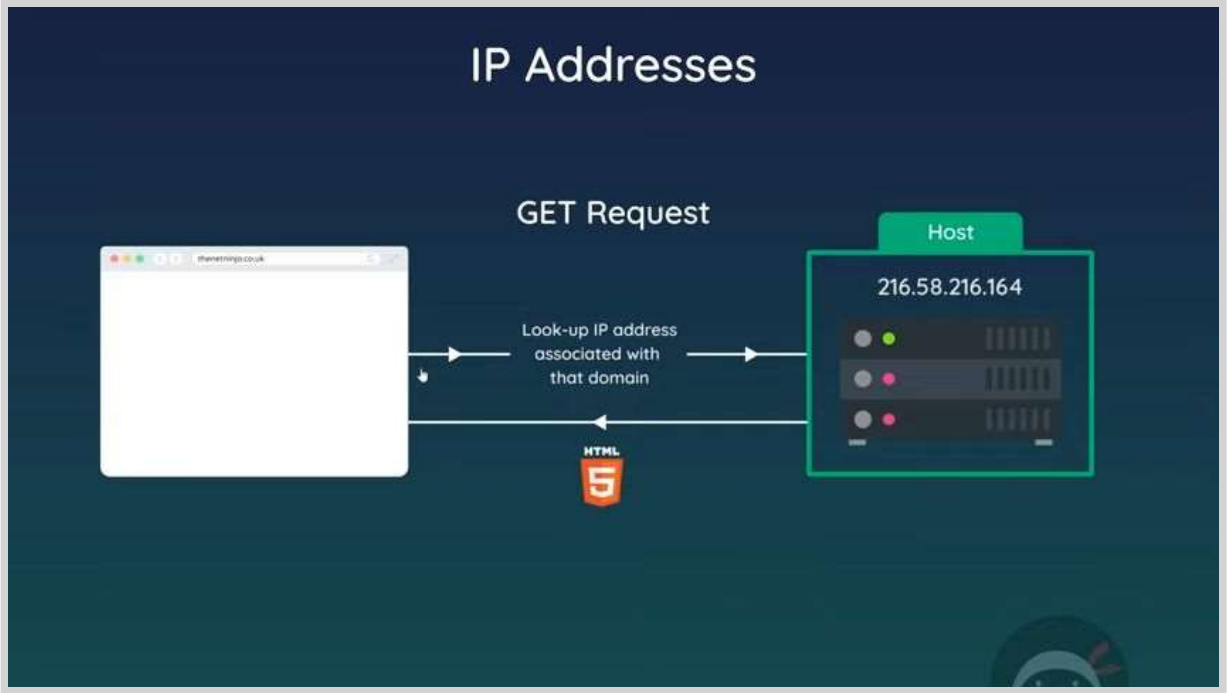



**whenever we type domain name on browser(URL simply) and hit enter it is simply get request to the server...**

**and we received ...html etc..**

**so this communication from browser to server is via HTTP...**

03:52



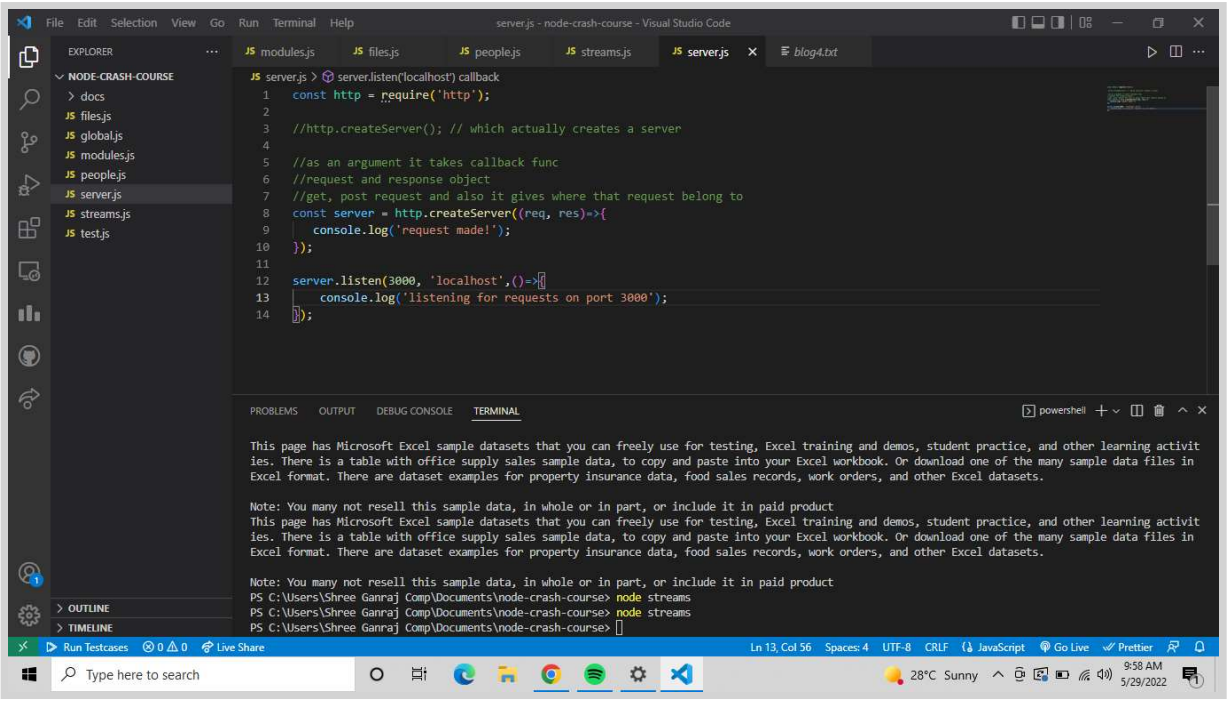

**client and servers will communicate with each other via ...HTTP**

**#Creating Server:**

**in php we don't have to create a server**

**but in node we create server manually**

**//creating a local server on a computer..**



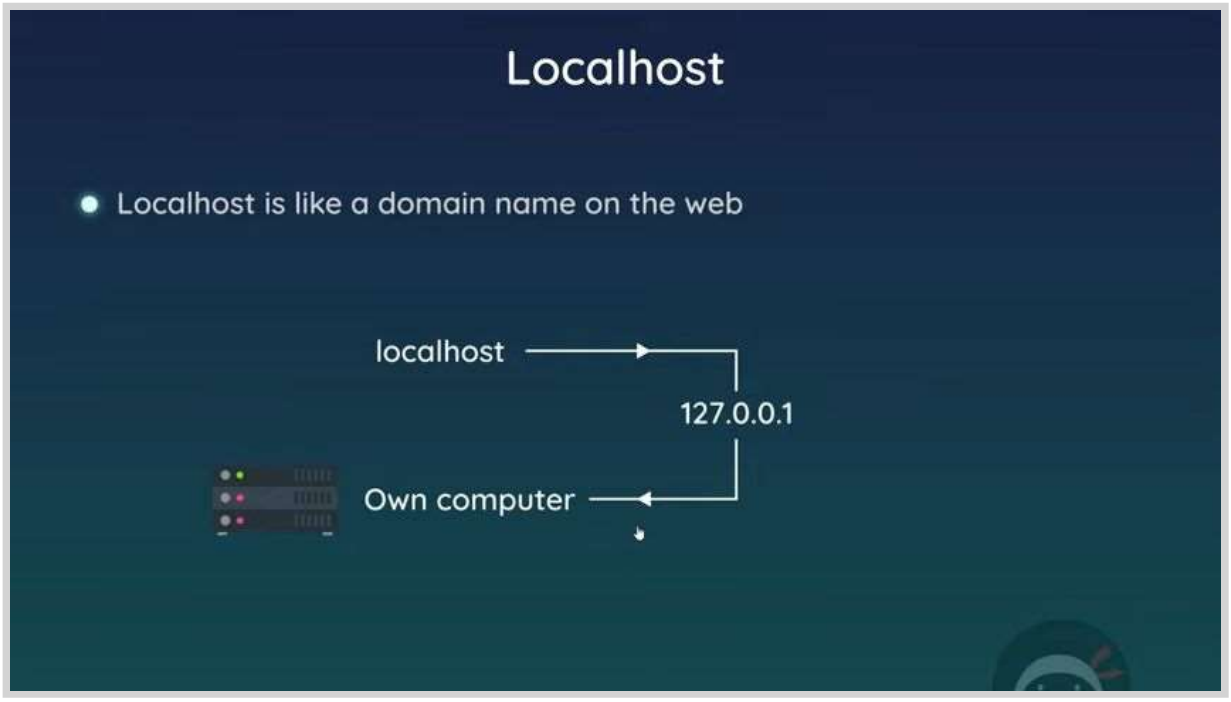
**//Localhost:**

**local lost is a like domain name on the web**

**eg: google.com**

**but localhost cha special IP address asto 127.0.0.1**

**09:21**



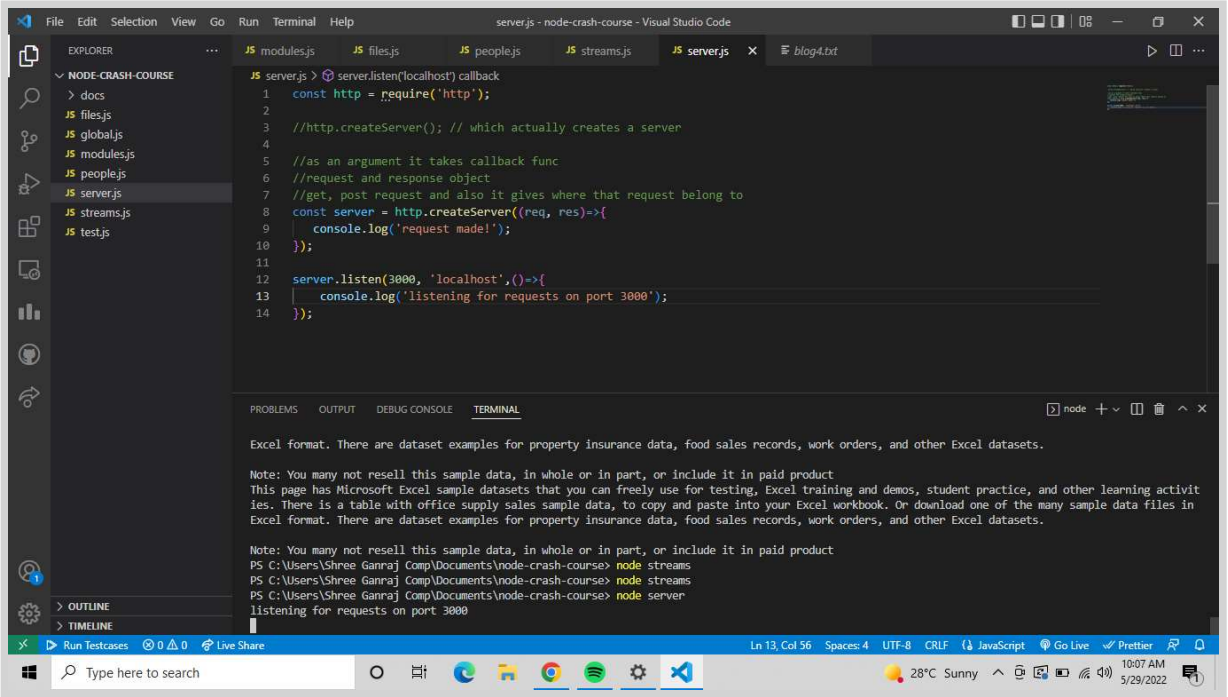

**Port Number:**

**specific channel on ur computer (jyala server sobt communicate karaychy)**

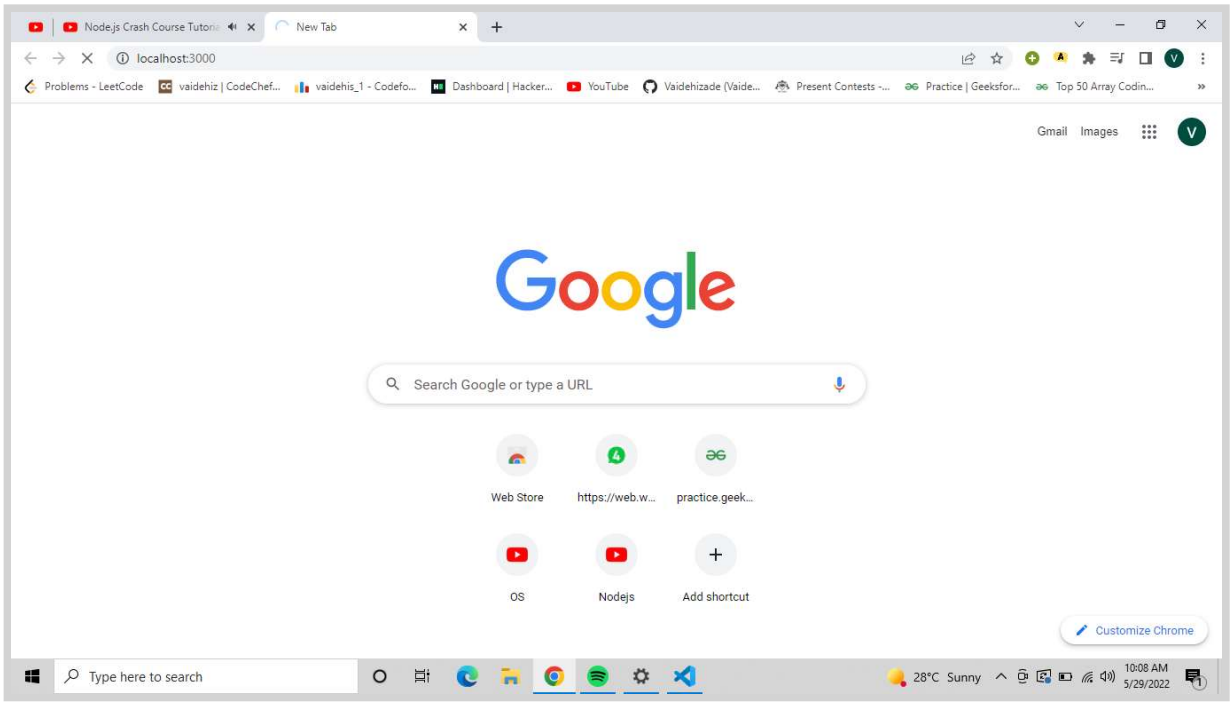
**through which internet communication is made**

**....localhost nantr we type port number..**

**10:42**


ata aplyala server la request karaychiy

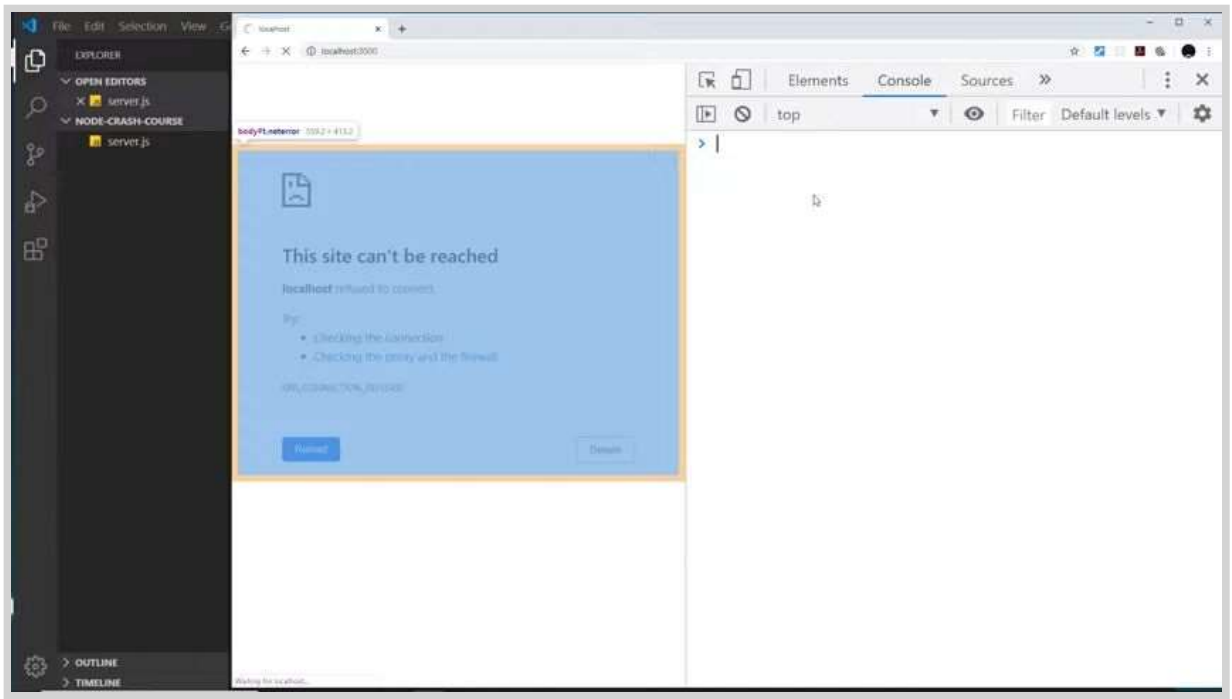



**waiting for server to respond ...**

**so he sagla server wr chally...not in browser**

**so aapn browser wr kahi disnar nahi inspect kela tr ...coz sagla backend wr chally**

**13:03**



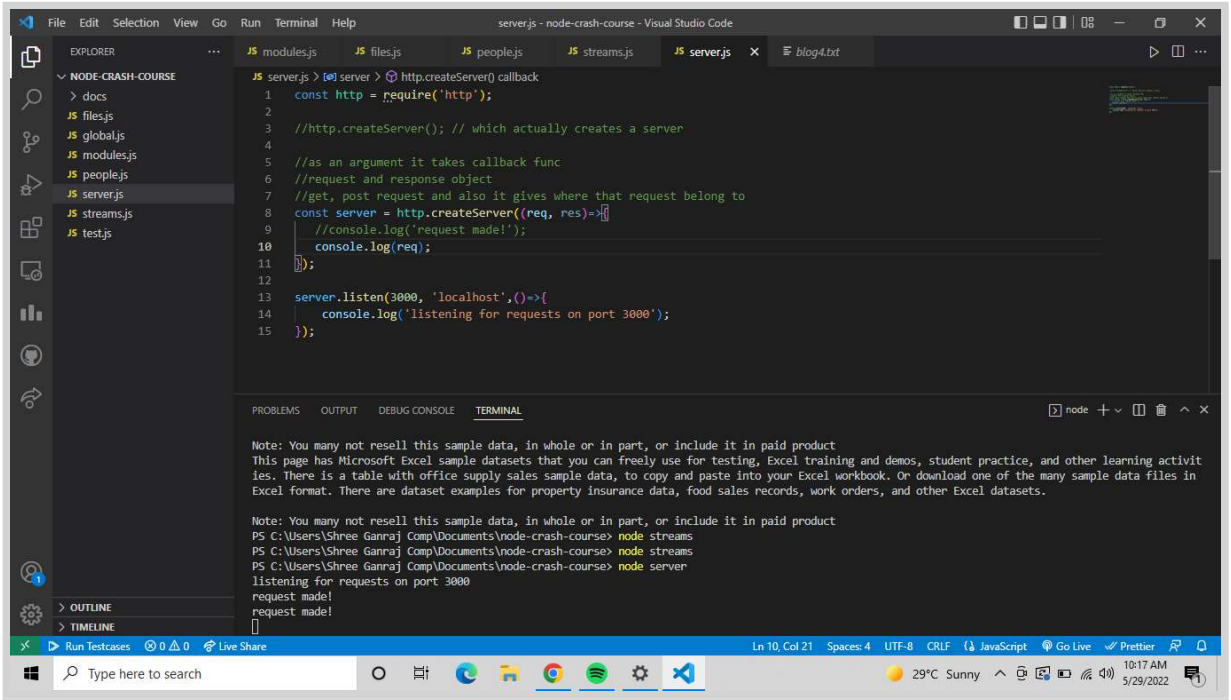

**how we can get response back...?**

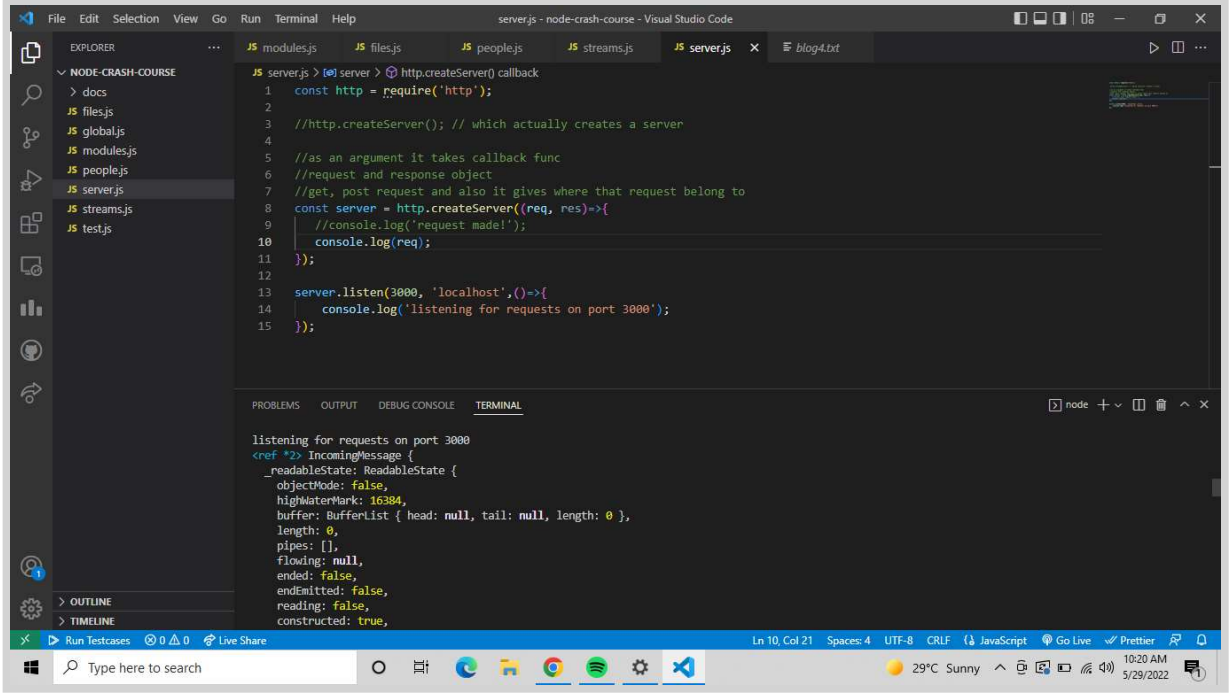
## Video4:

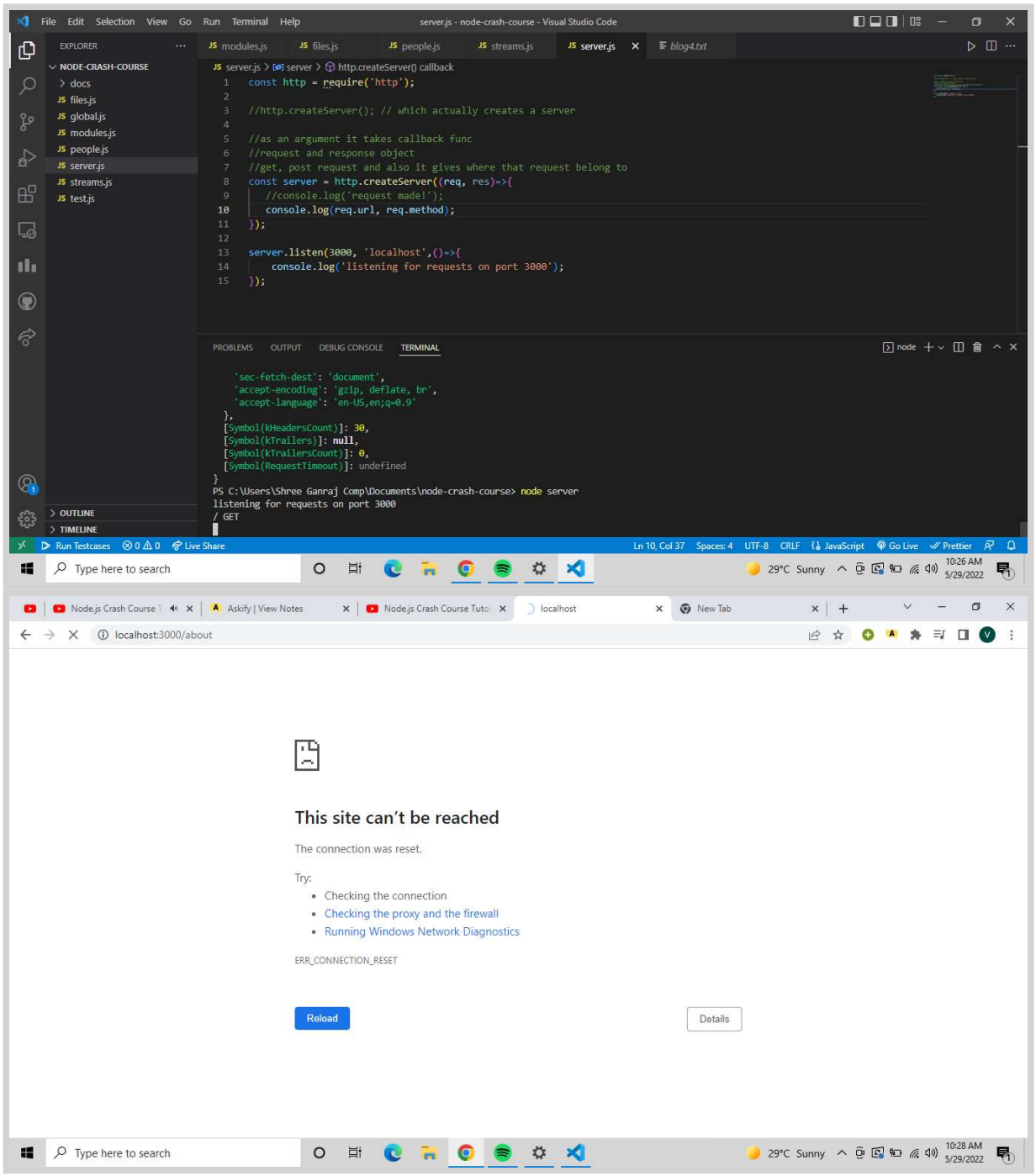
# Requests and Responses:

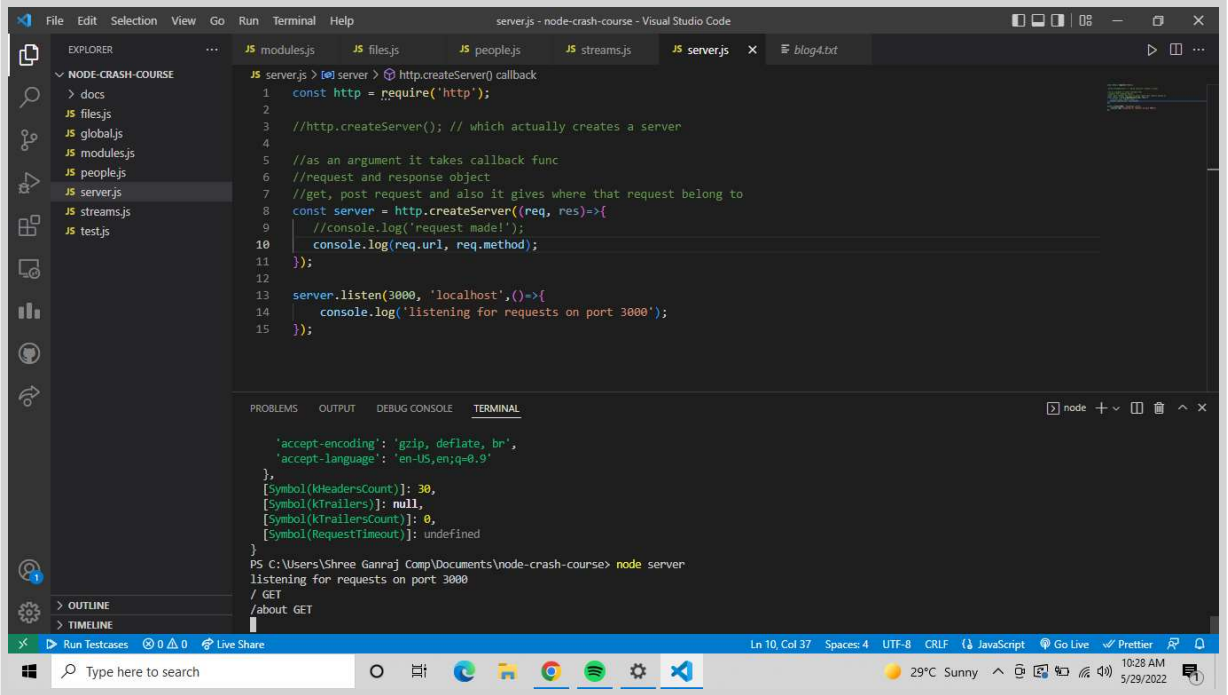
## => so ata we have created our server

## request object contains info about request send by user




server.js - node-crash-course - Visual Studio Code
NODE-CRASH-COURSE
server.js
const http = require('http');
//http.createServer(); // which actually creates a server
//as an argument it takes callback func
//request and response object
//get, post request and also it gives where that request belong to
const server = http.createServer((req, res)=>{
console.log(req.url, req.method);
});
server.listen(3000, 'localhost',()=>{
console.log('listening for requests on port 3000');
});
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node server
listening for requests on port 3000
/ GET
localhost:3000/about
This site can't be reached
The connection was reset.
Try:
Checking the connection
Checking the proxy and the firewall
Running Windows Network Diagnostics
ERR_CONNECTION_RESET
Reload
Details

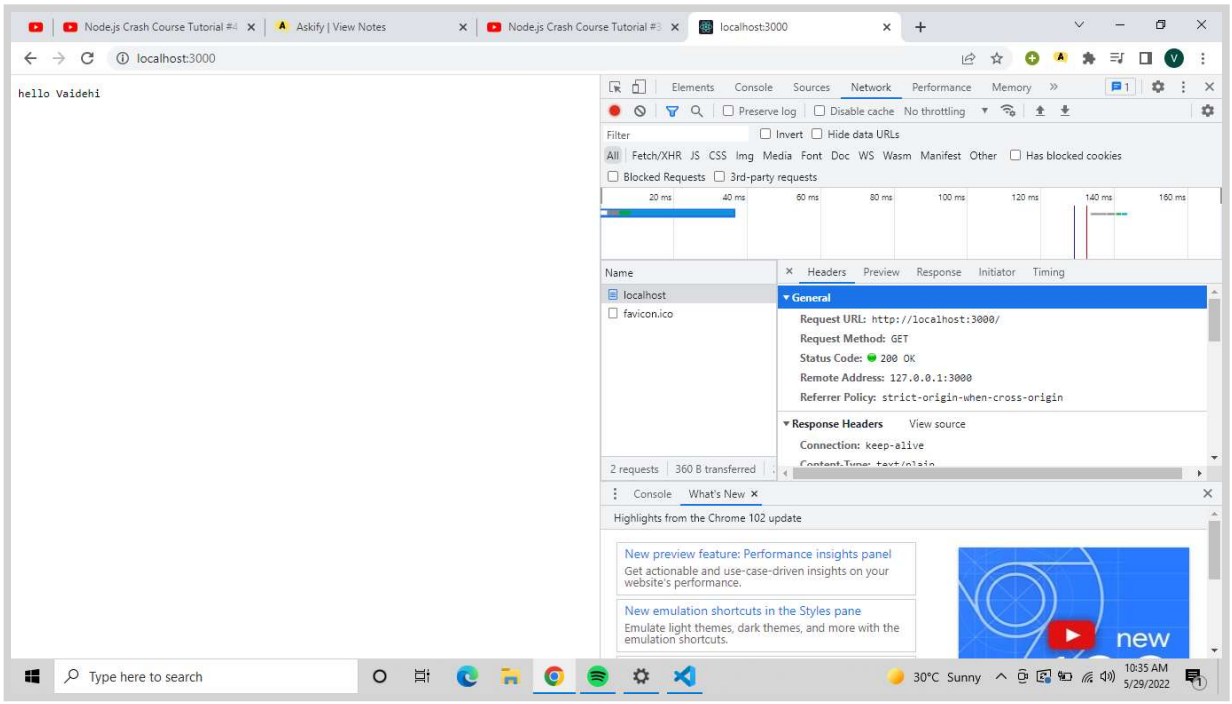


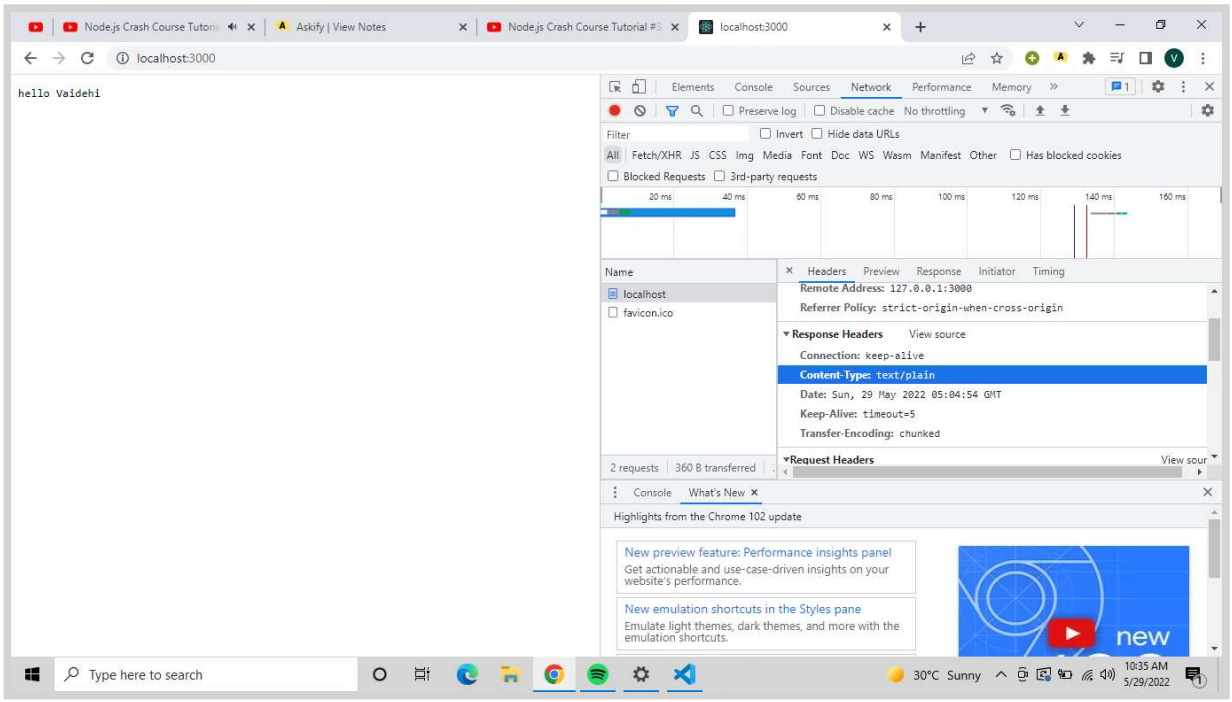

# #The response Object:

# response header => what type of data is sending back is in the form of??

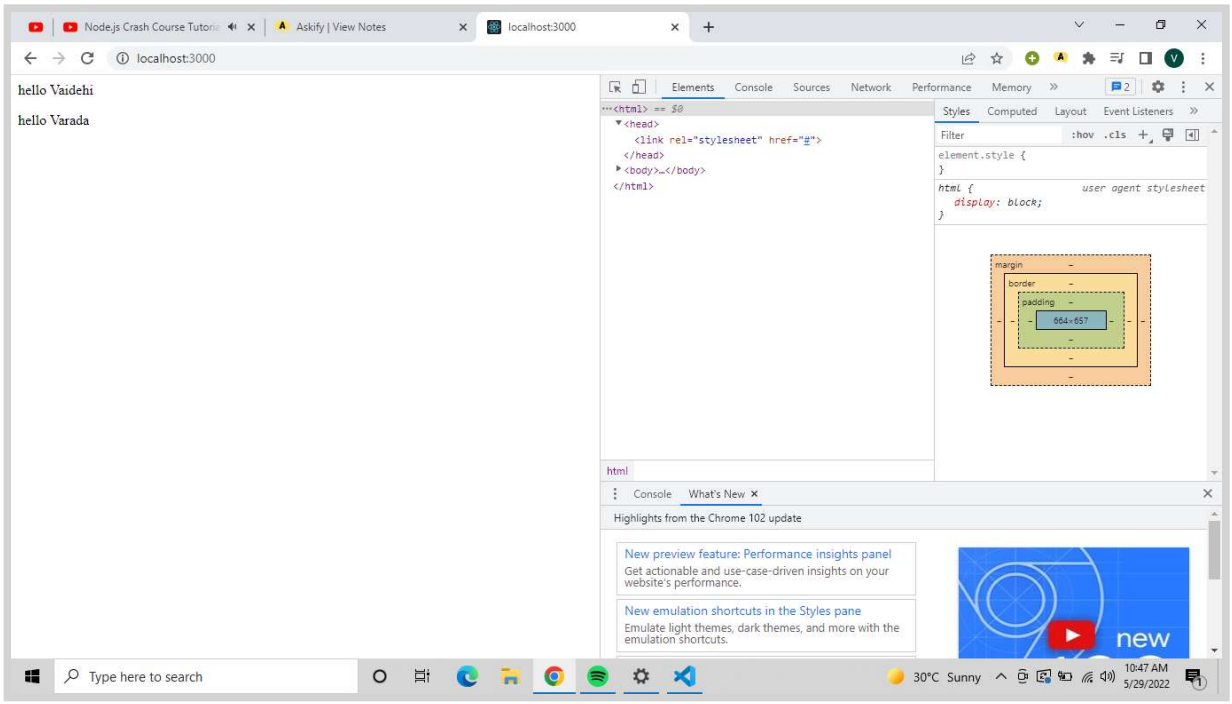
- **kontya type cha content**
- **jo main content send karaycha**
- **response.end();**

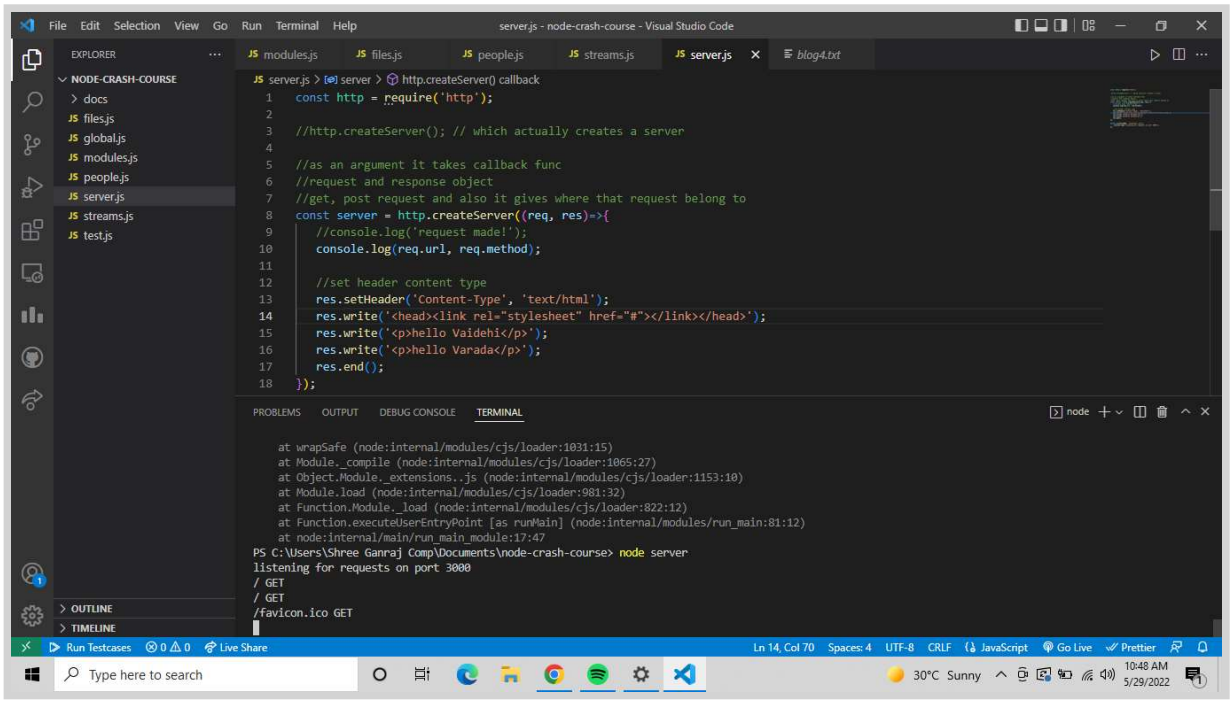
[05:32]




# what if we want to send back the html..



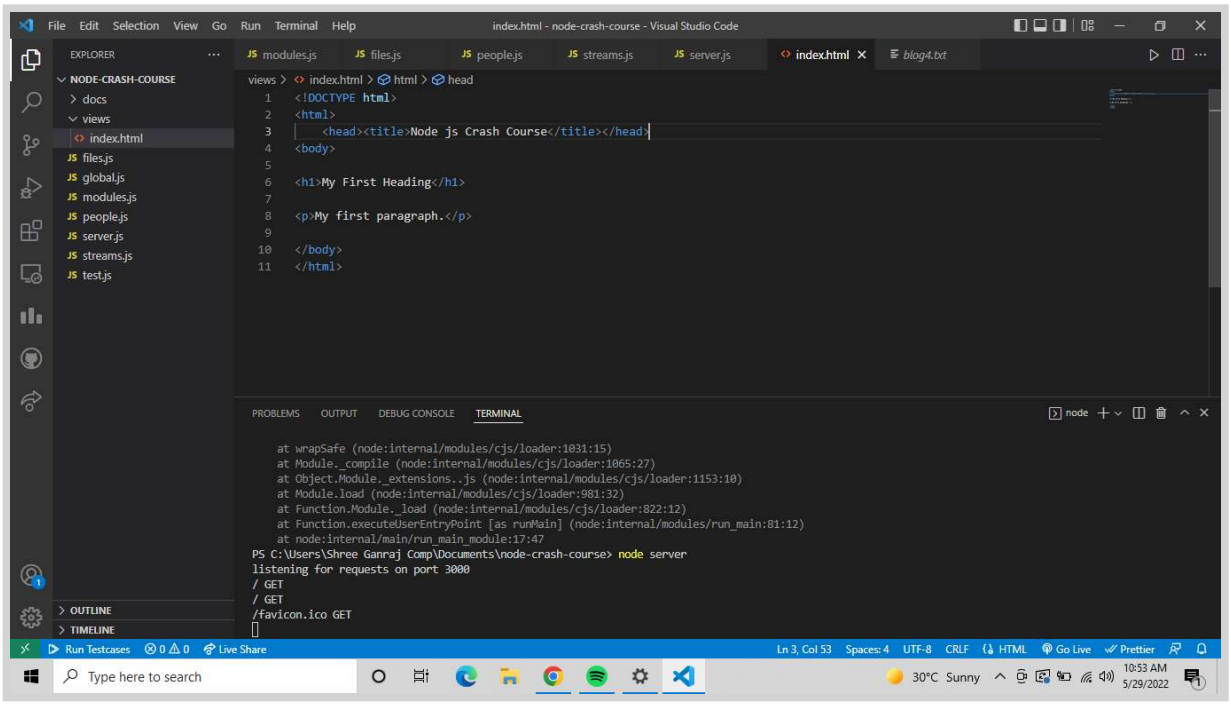

**But this way is quite bad to send html...**

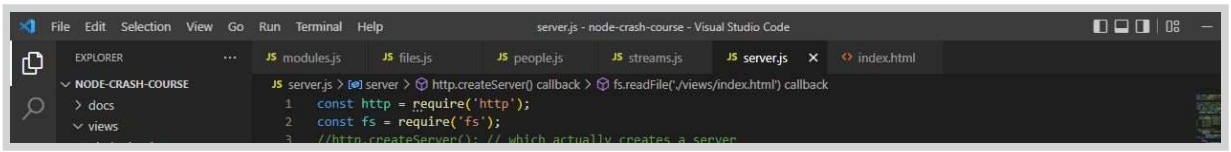
**so we have to use file system again!...**

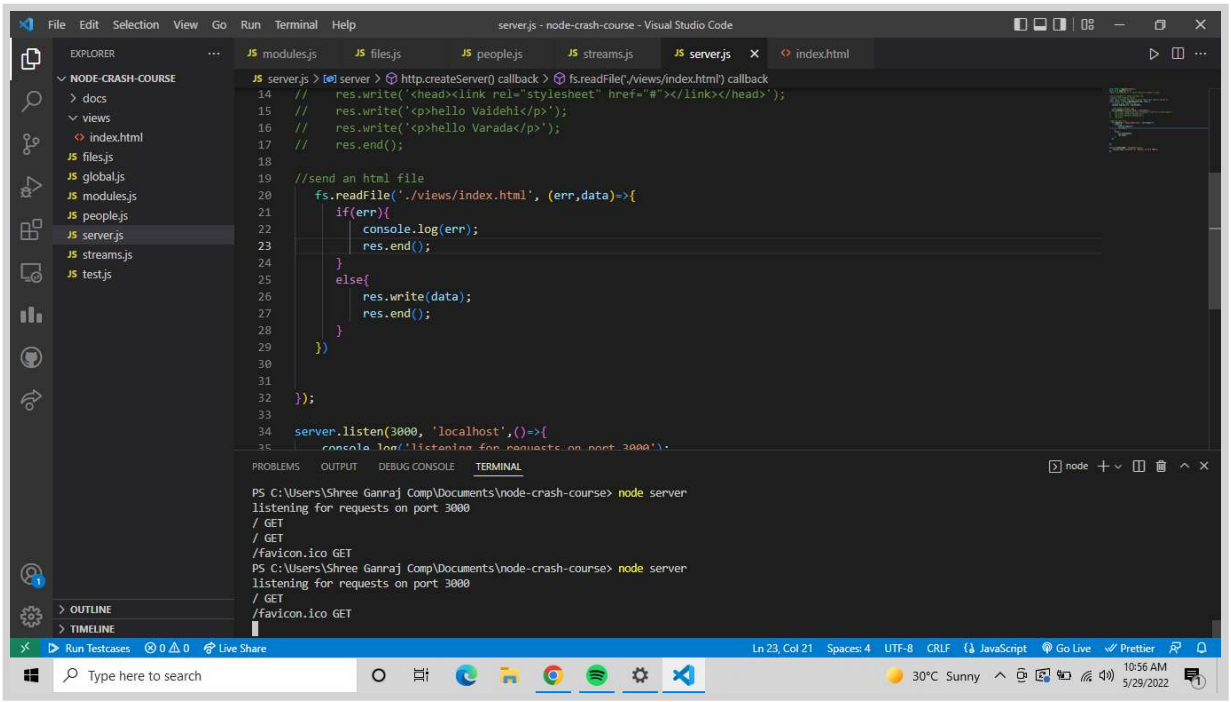
**#Returning HTML Pages:**

**how we can send full html document..**

**when user send a request , I want to send this document back ...**



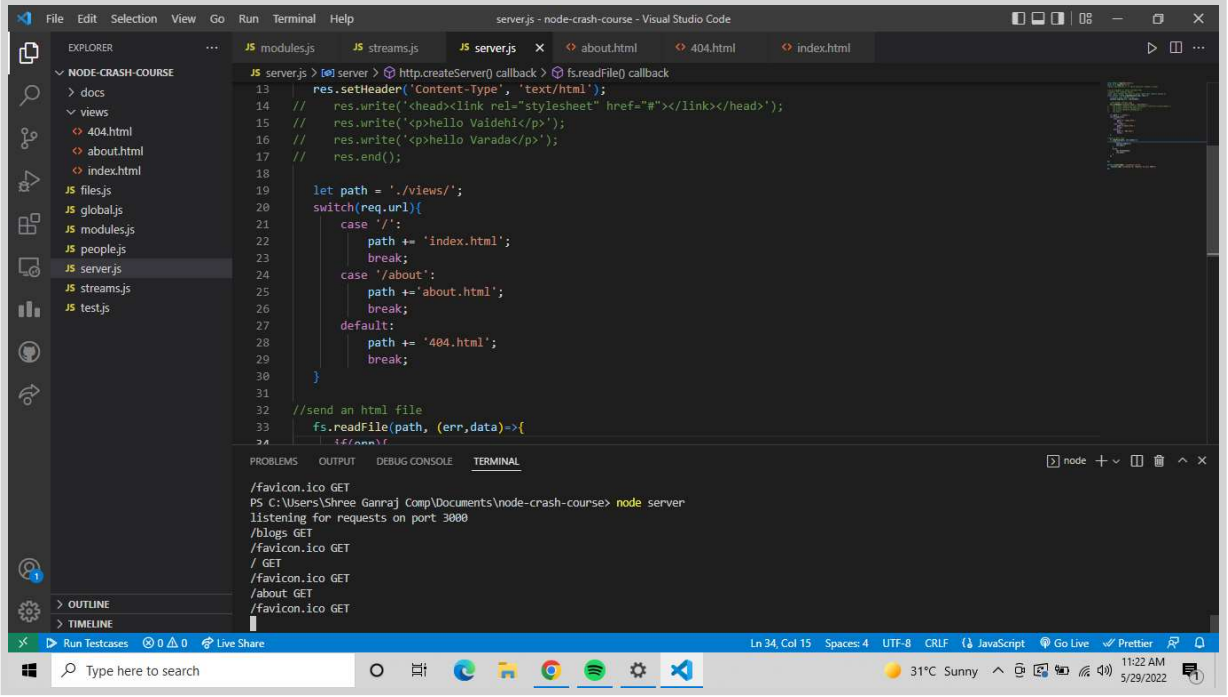




# more easy way=>

[12:34]


# #Basic Routing:

# with different url , send different html pages

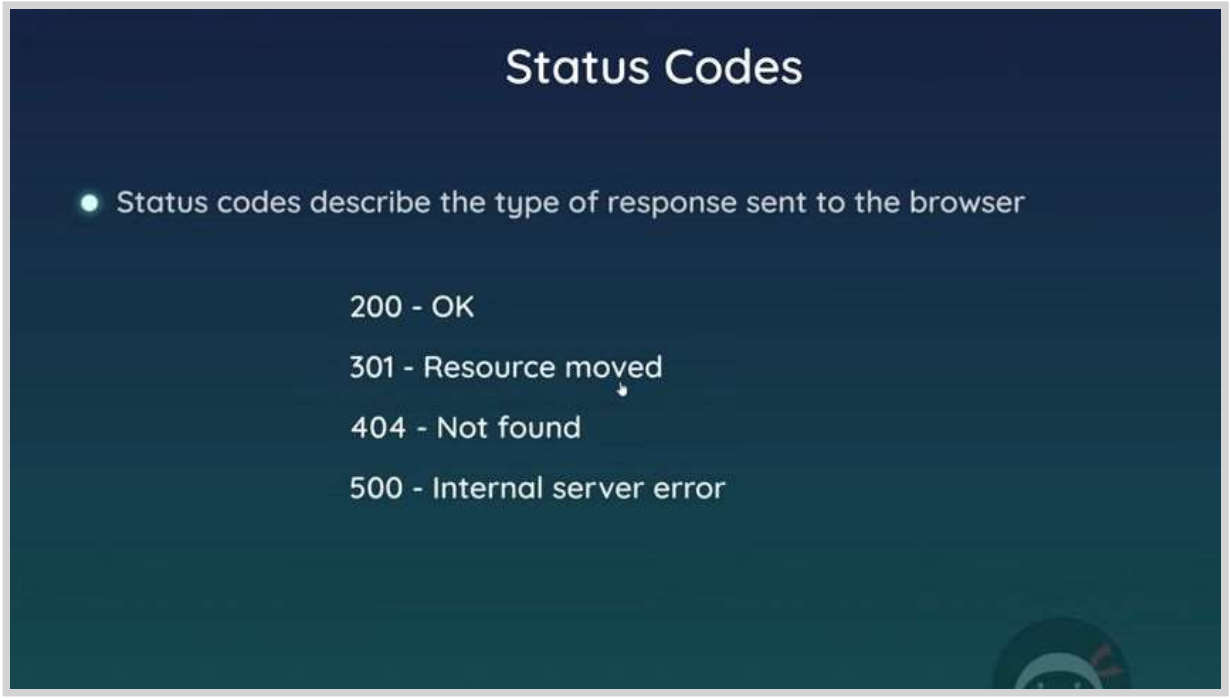



# #Status Codes:

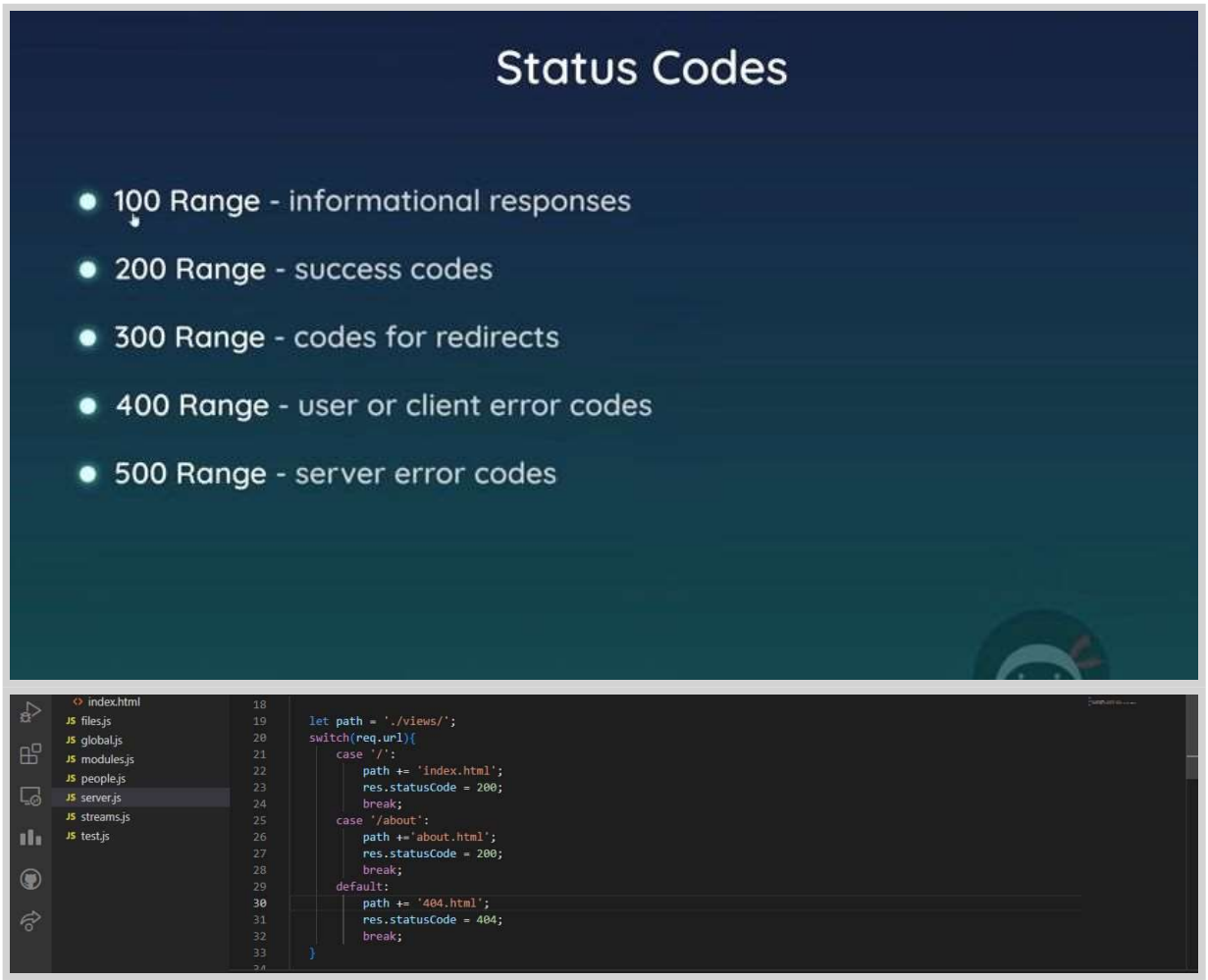
[19:01]

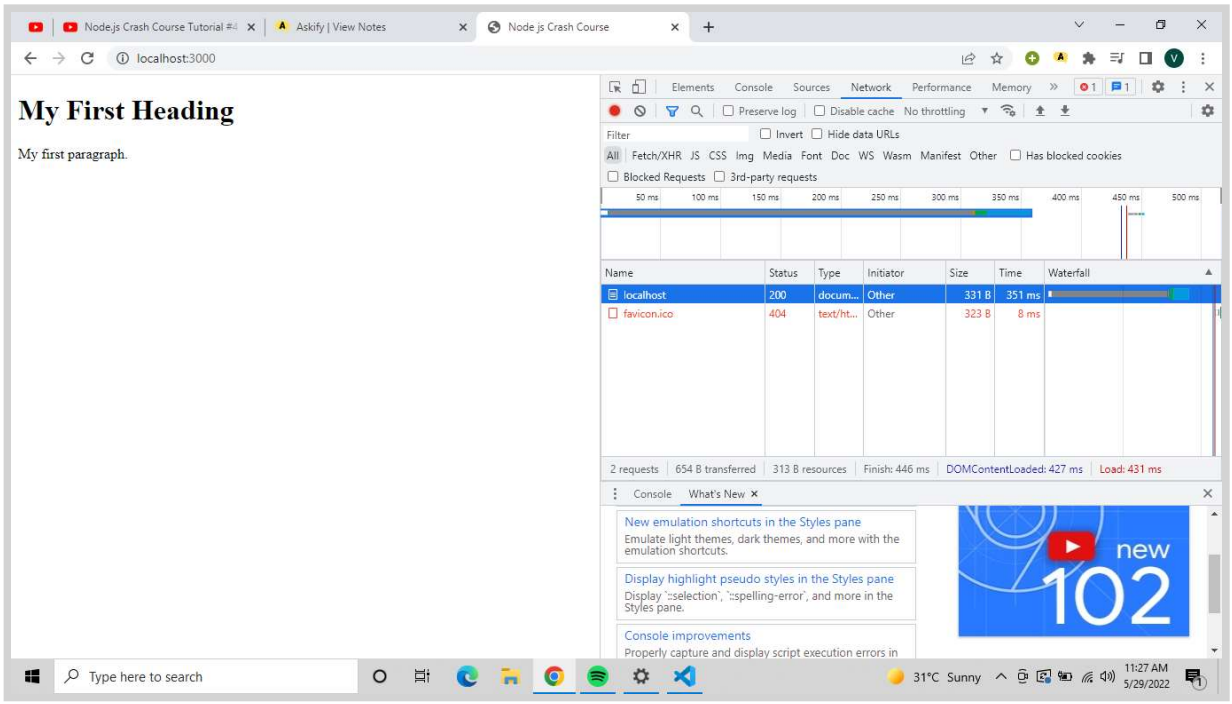

# and how successful the request was..

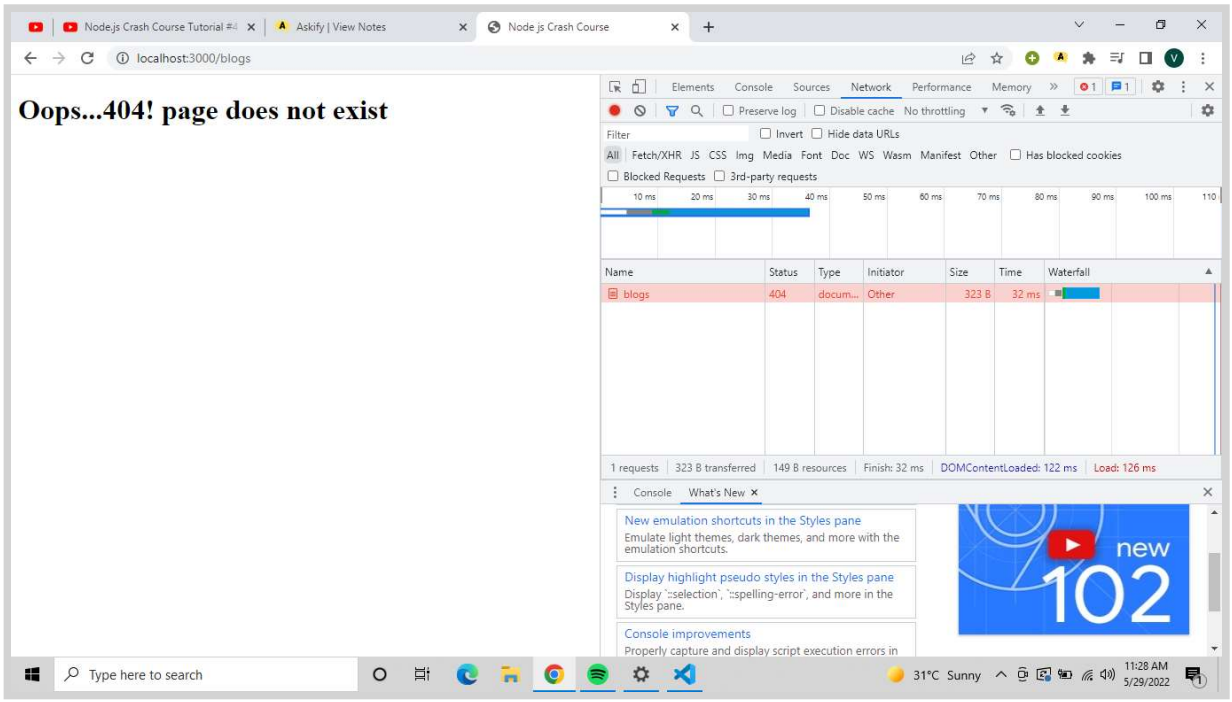
[19:31]


Status Codes
Status codes describe the type of response sent to the browser
200 - OK
301 - Resource moved
404 - Not found
500 - Internal server error

[19:34](#)

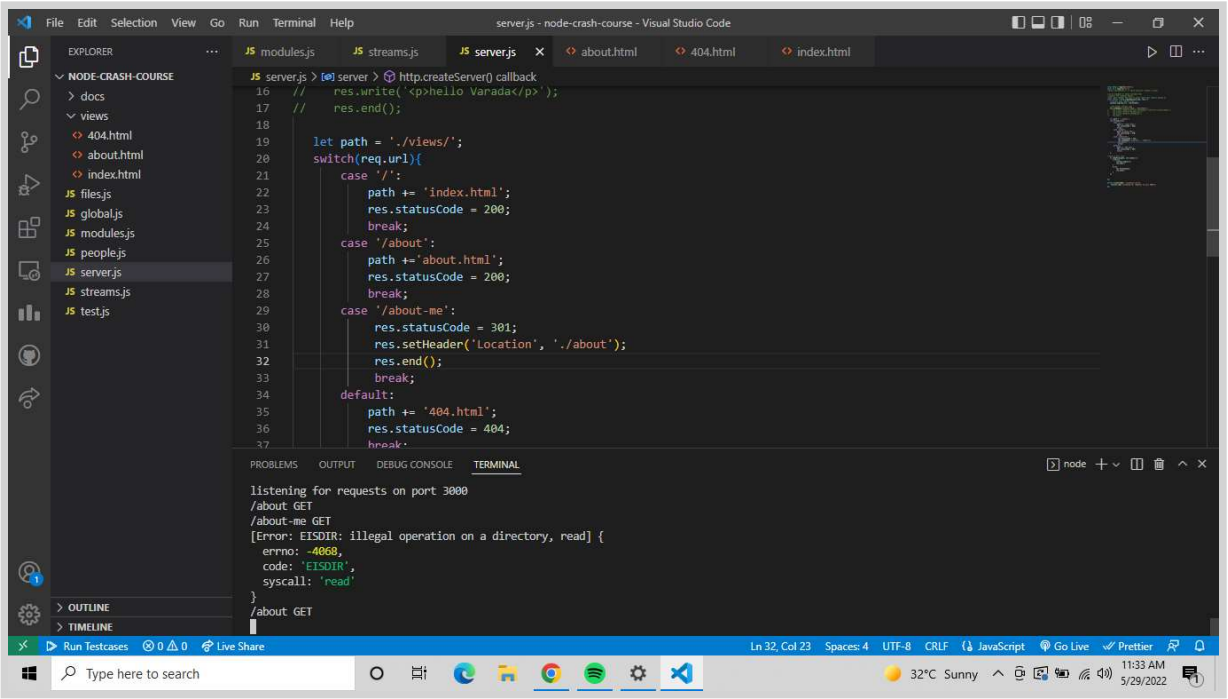

Status Codes
100 Range - informational responses
200 Range - success codes
300 Range - codes for redirects
400 Range - user or client error codes
500 Range - server error codes
index.html
files.js
global.js
modules.js
people.js
server.js
streams.js
test.js
let path = './views/';
switch(req.url){
case '/':
path += 'index.html';
res.statusCode = 200;
break;
case '/about':
path +='about.html';
res.statusCode = 200;
break;
default:
path += '404.html';
res.statusCode = 404;
break;
}




# #How to do redirects:

# i want to redirect from about to about -me




**so if khup mothe pages asle tr express(thirty party ) aplyala routing redirecting sathi help krte...**

**to handle all our routes and request=> we will use express package**

## Video5:

## #NPM:

**=> node package manager**

**using npm we can third party packages**

https://www.npmjs.com/

**//Installing packages globally:**

**punha punha server stop krun run nahi krav lagt...**

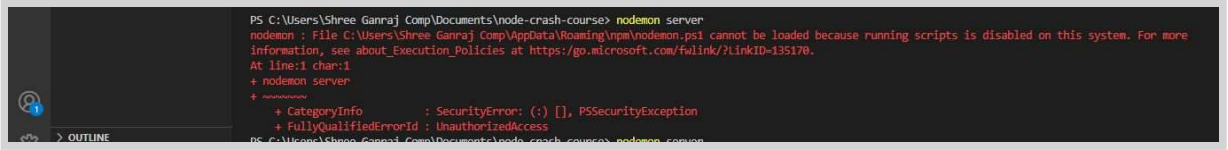
**nodemon mule automatically server madhe changes save hot jatat ani te ...chalat rahta**

**It automatically restarts the server**

**//we re installing package globally:**

```
npm install -g nodemon
```

**I had got an error:**




**to fix this:**
**https://stackoverflow.com/questions/63423584/how-to-fix-error-nodemon-ps1-cannot-be-loaded-because-running-scripts-is-disabl**

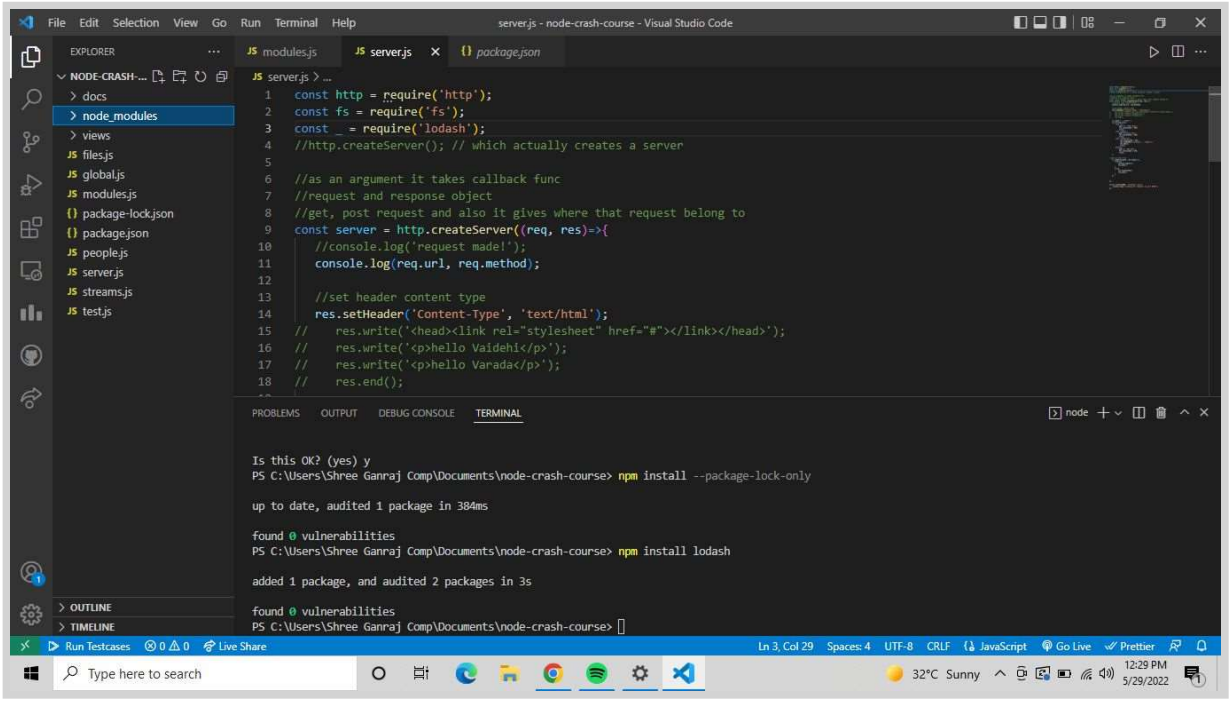
**#Packages that are only specific to our project:**
**//Package.json file:**
**=>Package file is json file....**
**jr aplyala third-party packages use karaychet tr package.json definately u should create..**
**we will create package.json using "npm init" command..**

**package-lock.json madhe diffreent dependencies che version asta...**
**(locally je install kelele)**


**#installing packages locally:**
**utility, date-time libraries...**
**lodash: a modern js utility library delivering modularity, performance & extras...**

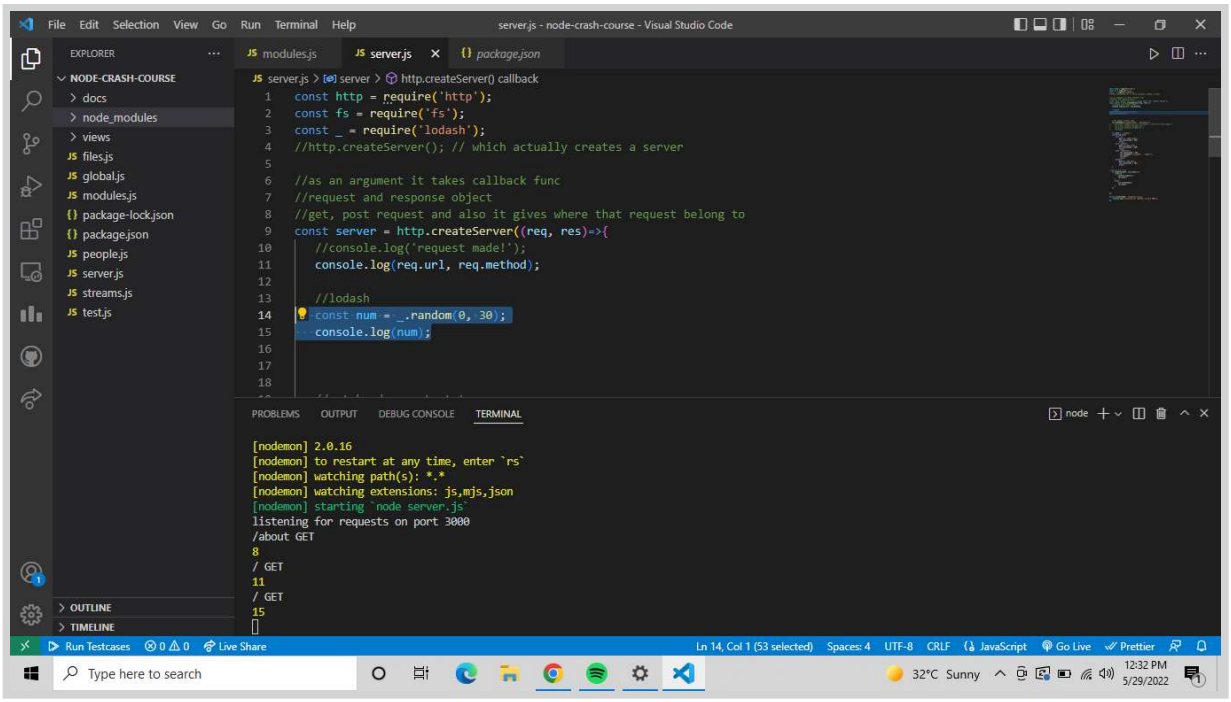
https://lodash.com/

## //npm install lodash (installed locally)

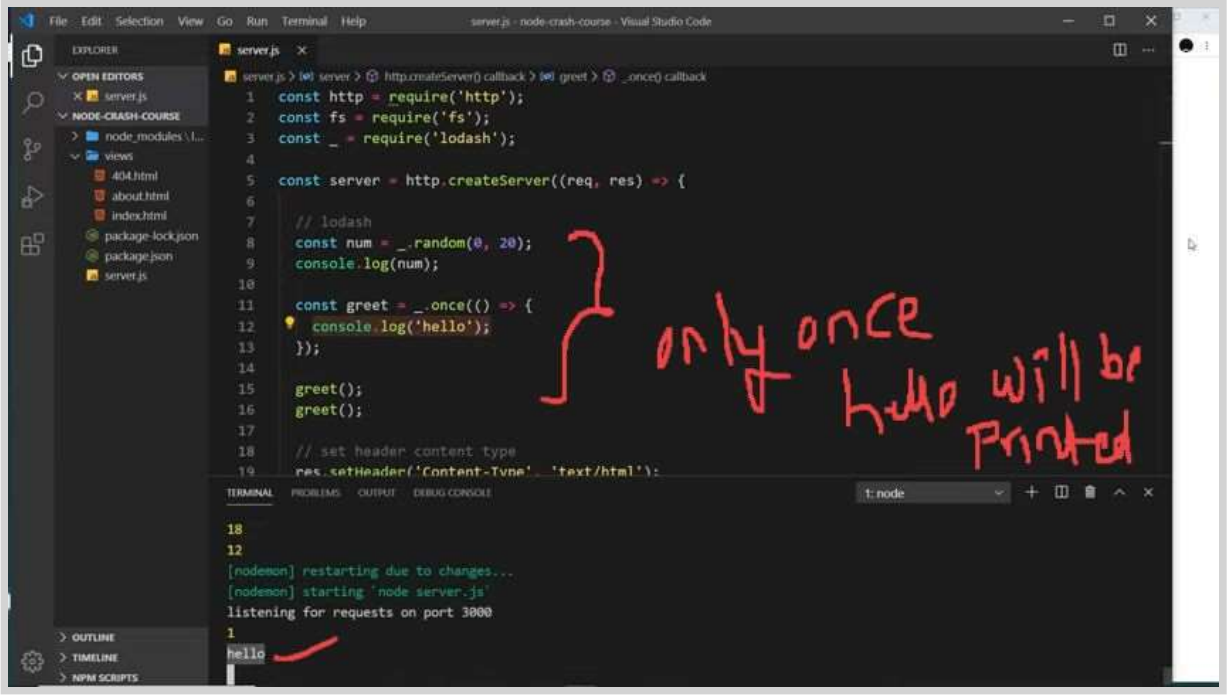



## lodash provides utility methods:

## //let's say we want to generate random numbers




12:59




**npm allows to easily share the project codess**

**=>on github aapn node modules nahi takat**

**so delete krto na**

**mg jevha run kru tevha adhi "npm install" (coz we have package.json already with uss)**

**evdhich command run karaychi ani mg project run karaycha ...**

## Video6:

## #Express Apps:

**Express is a framework**

**Express helps us to easily manage our routing, requests, server side logic, responses...**

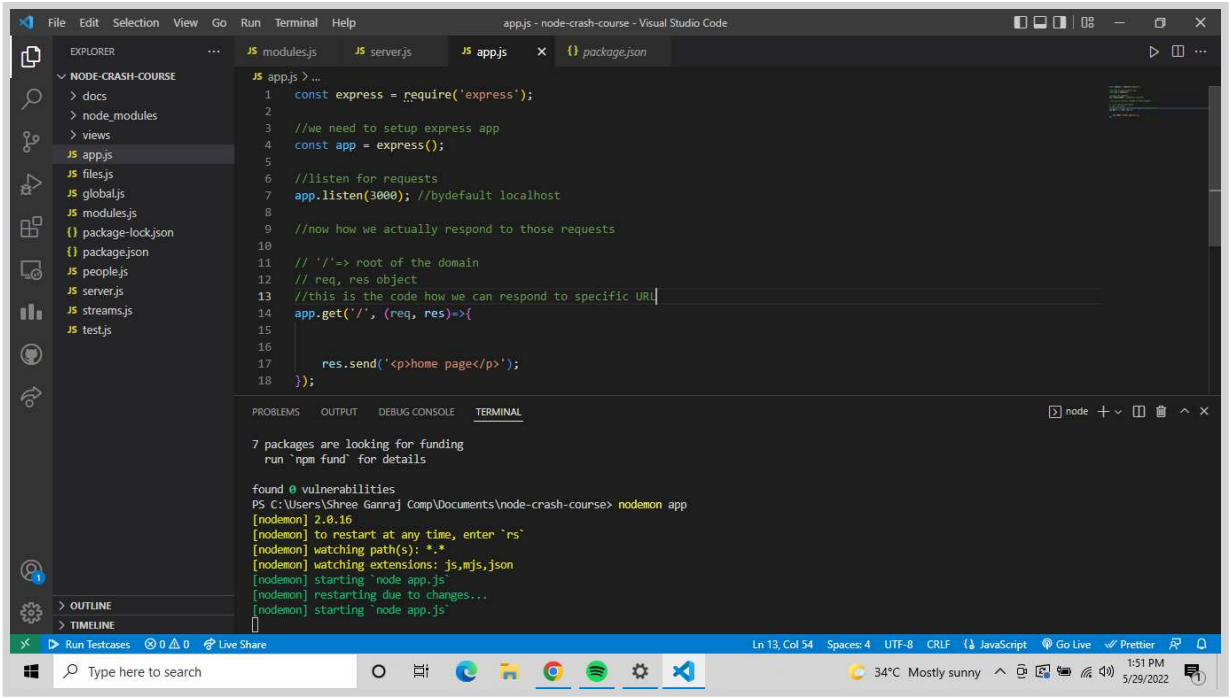
```
const express = require('express')
const app = express()

app.get('/', function (req, res) {
  res.send('Hello World')
})

app.listen(3000)
```

**//npm install express**

**//creating an express app:**

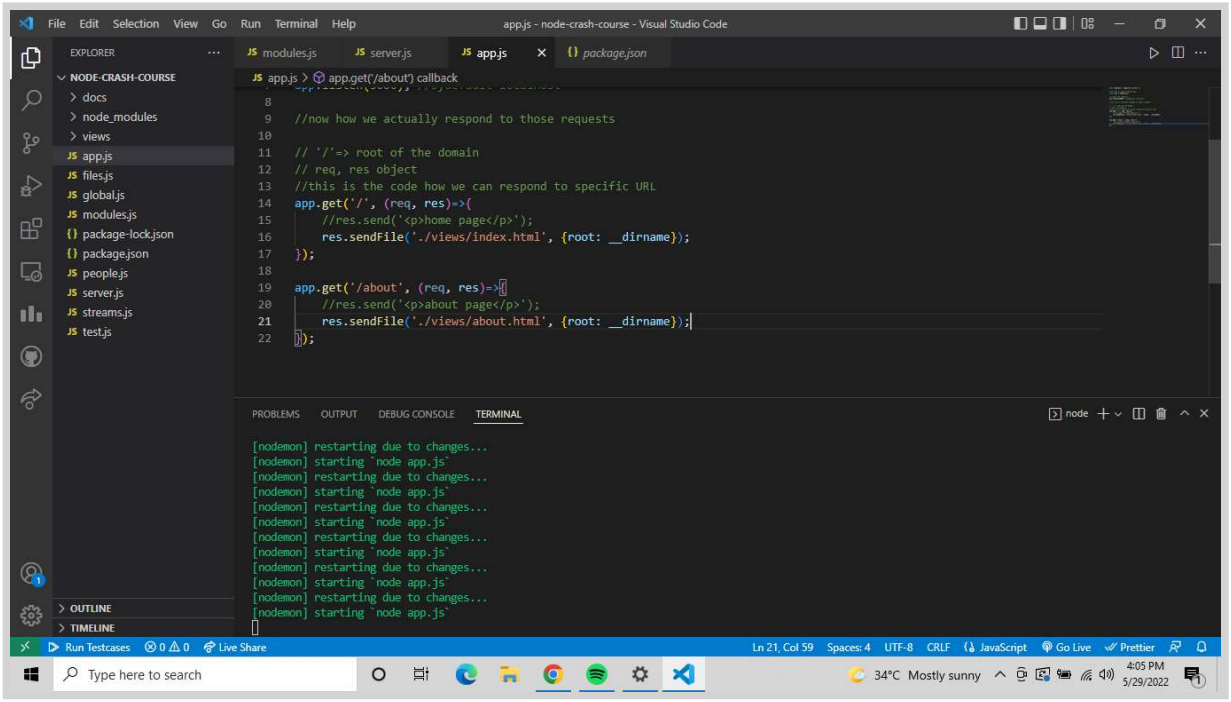
- **kontya type cha content**
- **jo main content send karaycha**
- **response.end();**

**ya goshtinchi take care express by default krto...ani statusCode also..**

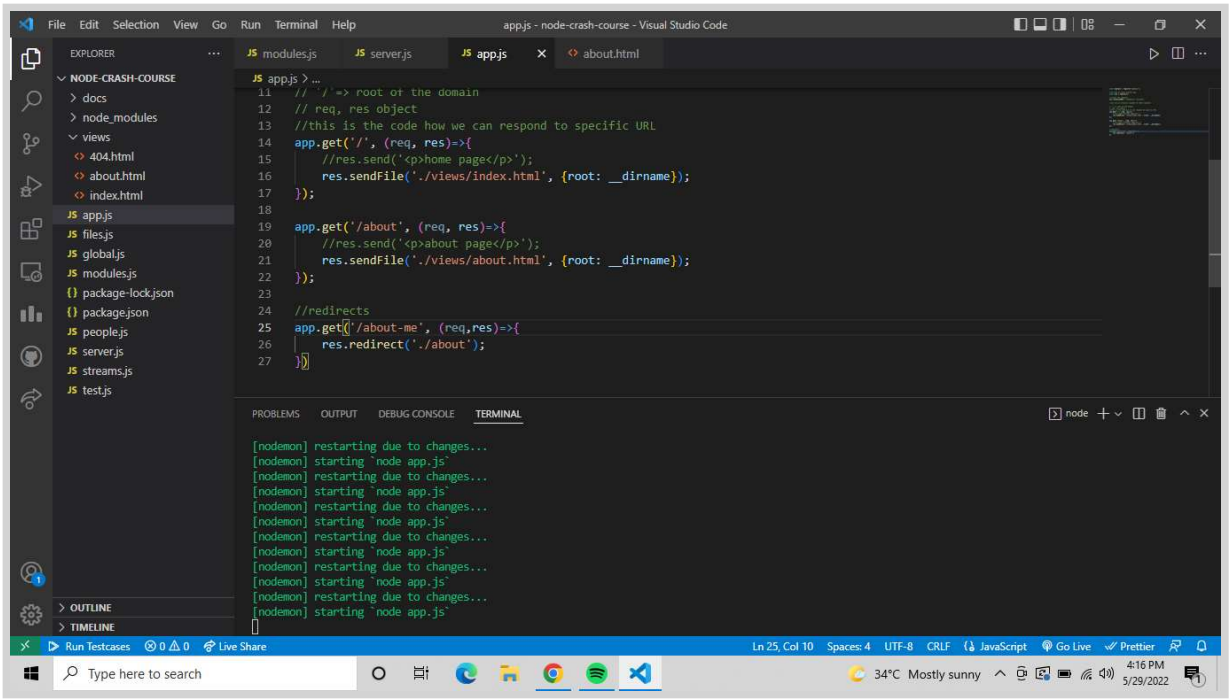



## //Routing HTML:

# => how to send html file using express

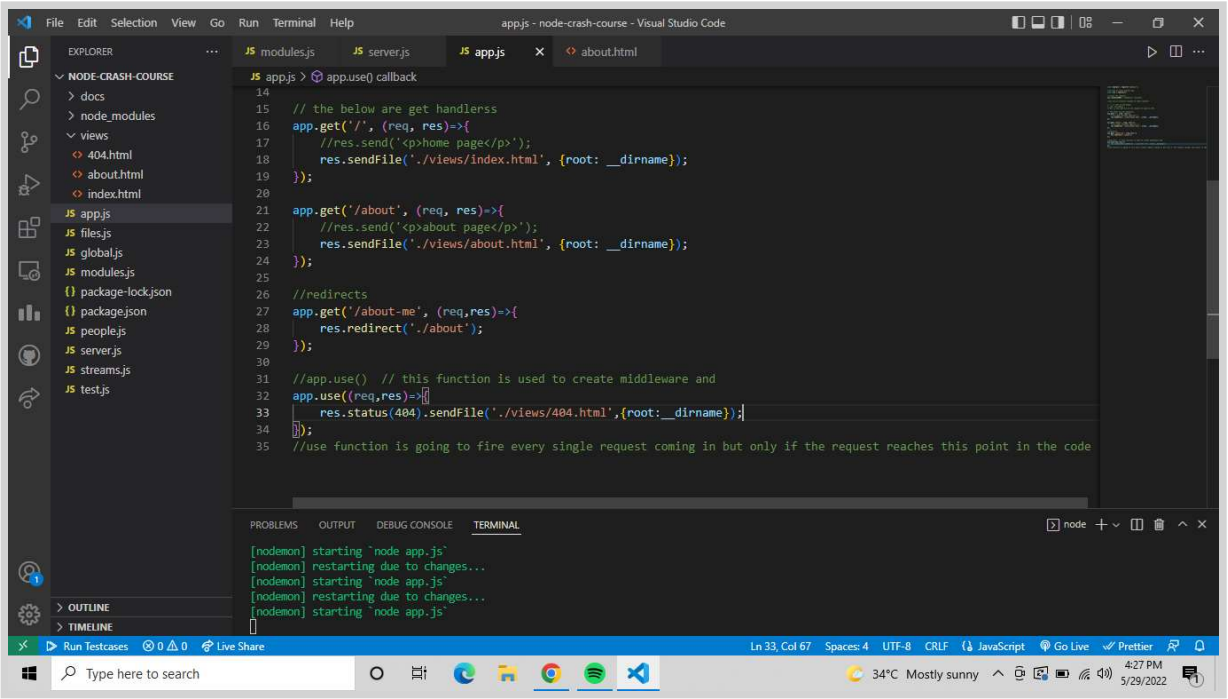



# //Redirects and 404Pages=>in express




# //express automatically sets the status code

# //404 Pages=>




**//server.js wala node ne kela hota to bgha**

**ani app.js wala express ne kelela compare kara...and see the difference**

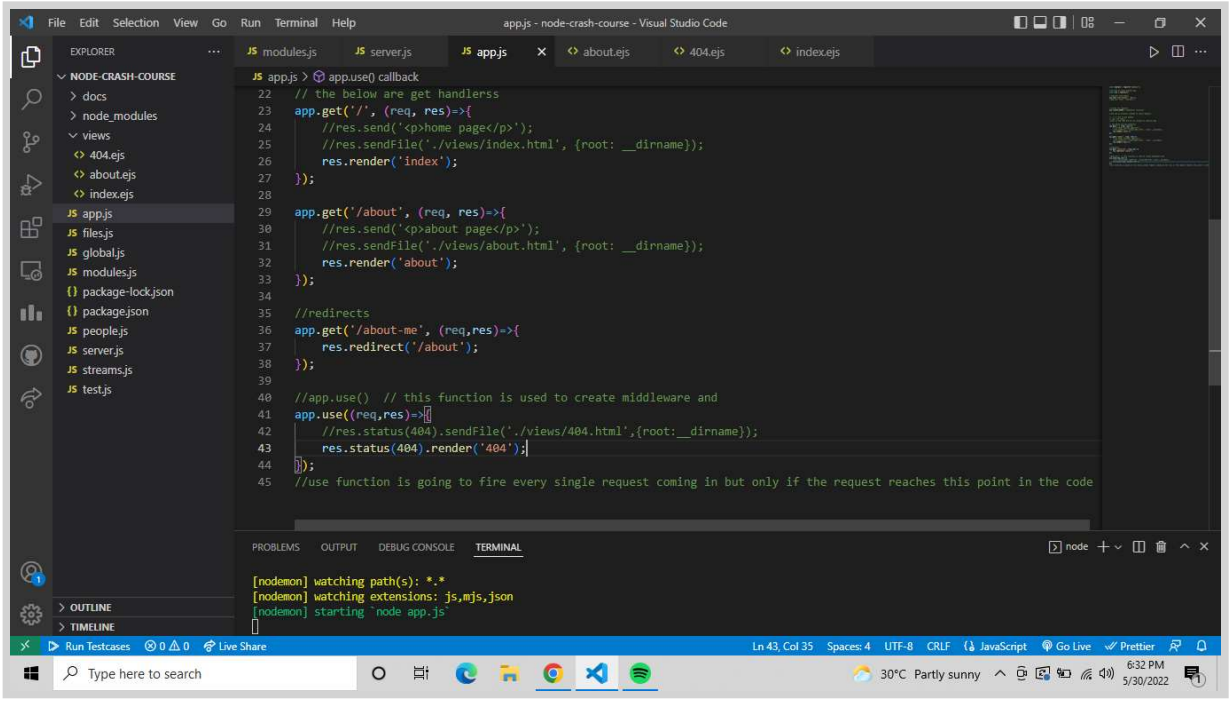
## Video7:

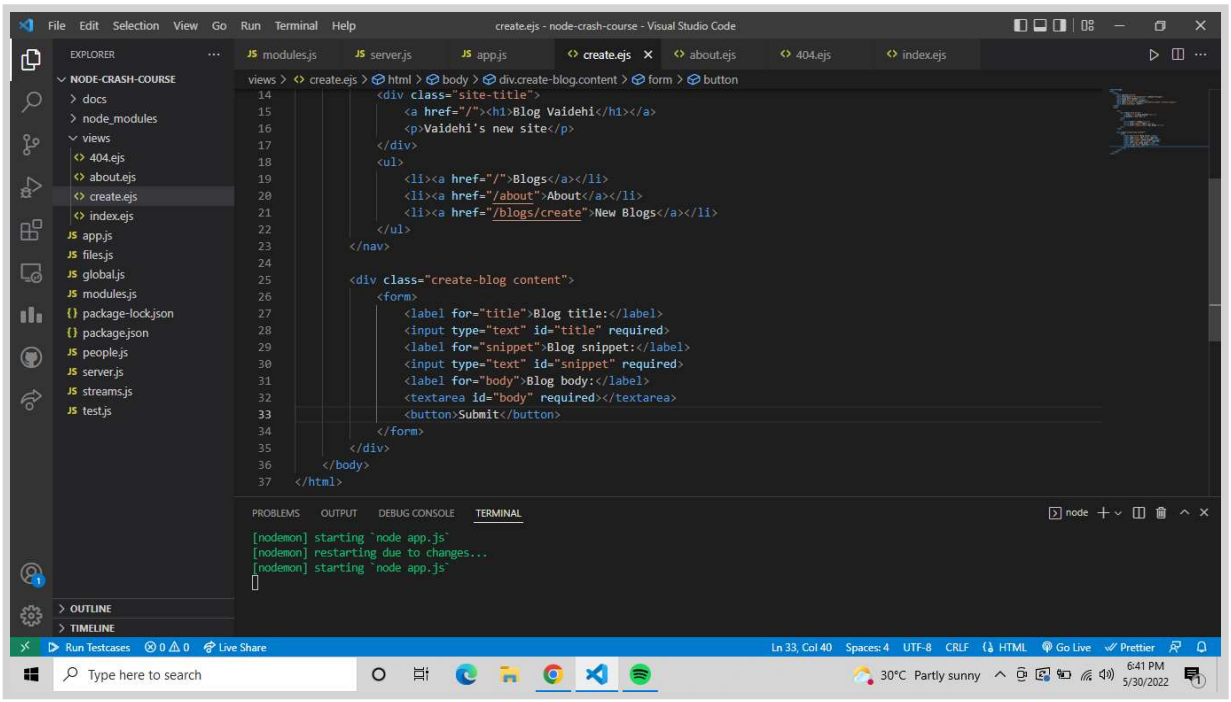
## #View Engines:

**dynamic data insert krnyasathi/display krnyasathi ...view engine/ template engine use krto....and express app use krta he engines...**

**1)Express handlebars**

**2)EJS(embedded javascript templating) view engine**



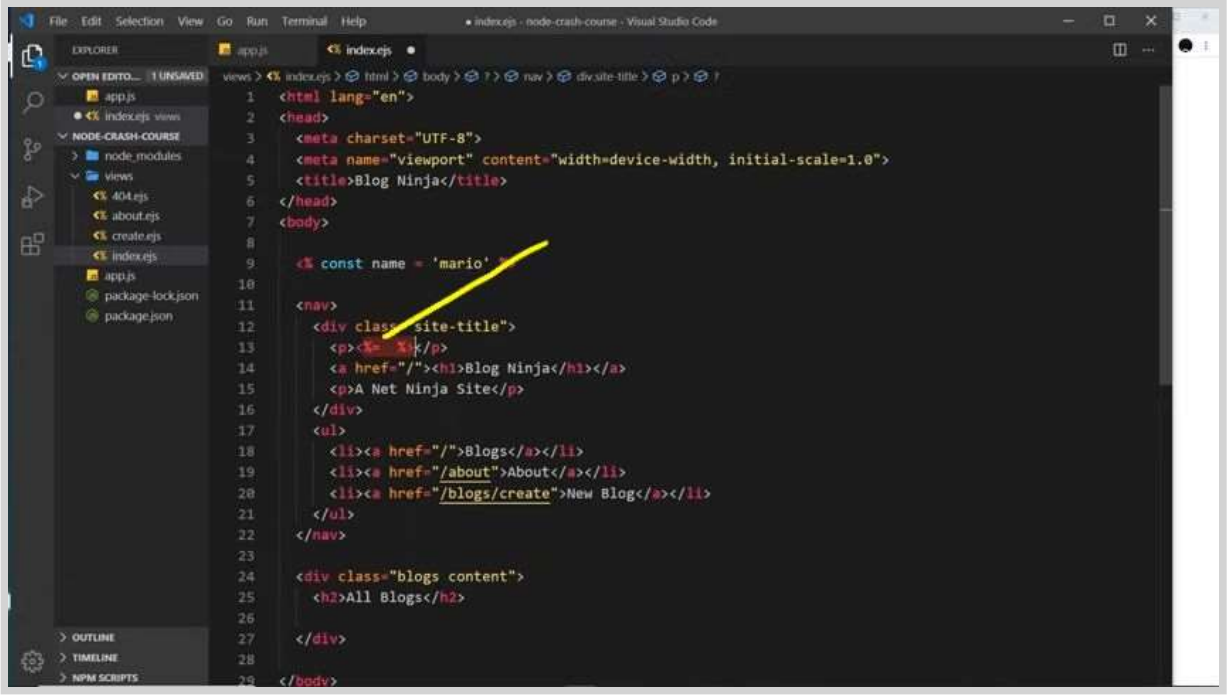




# Passing data into views:

# inside % we can do any kind of JS(dynamic)

# server wr js yeil...

[12:28]

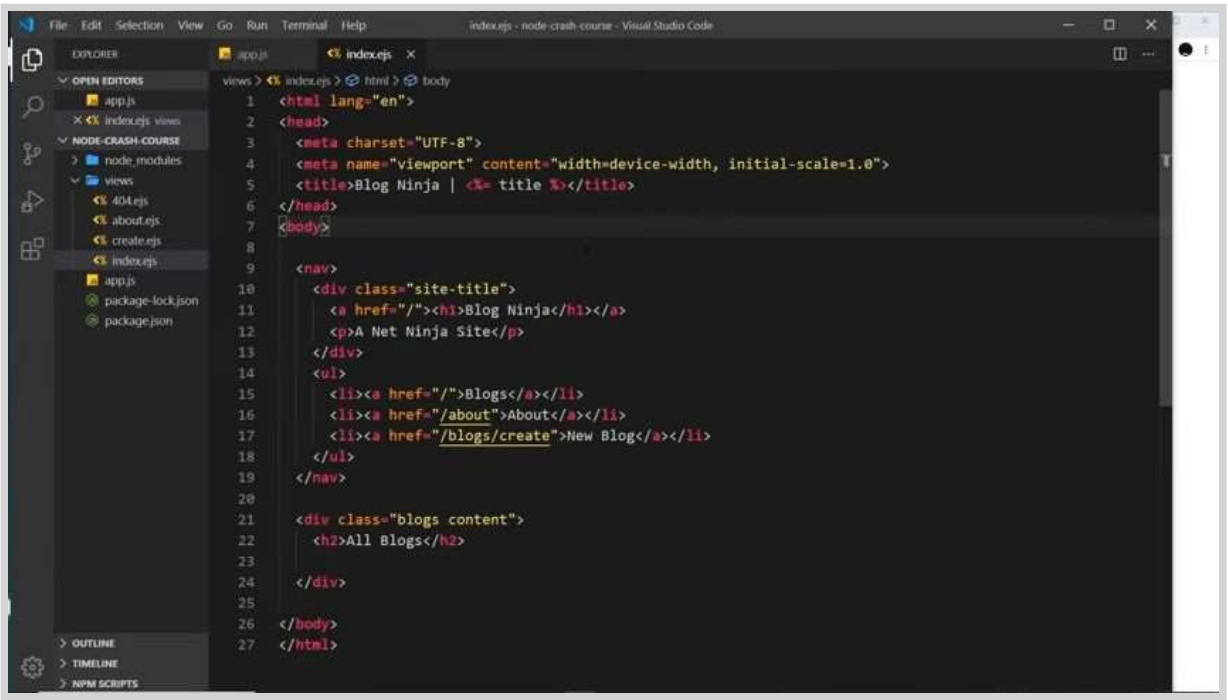


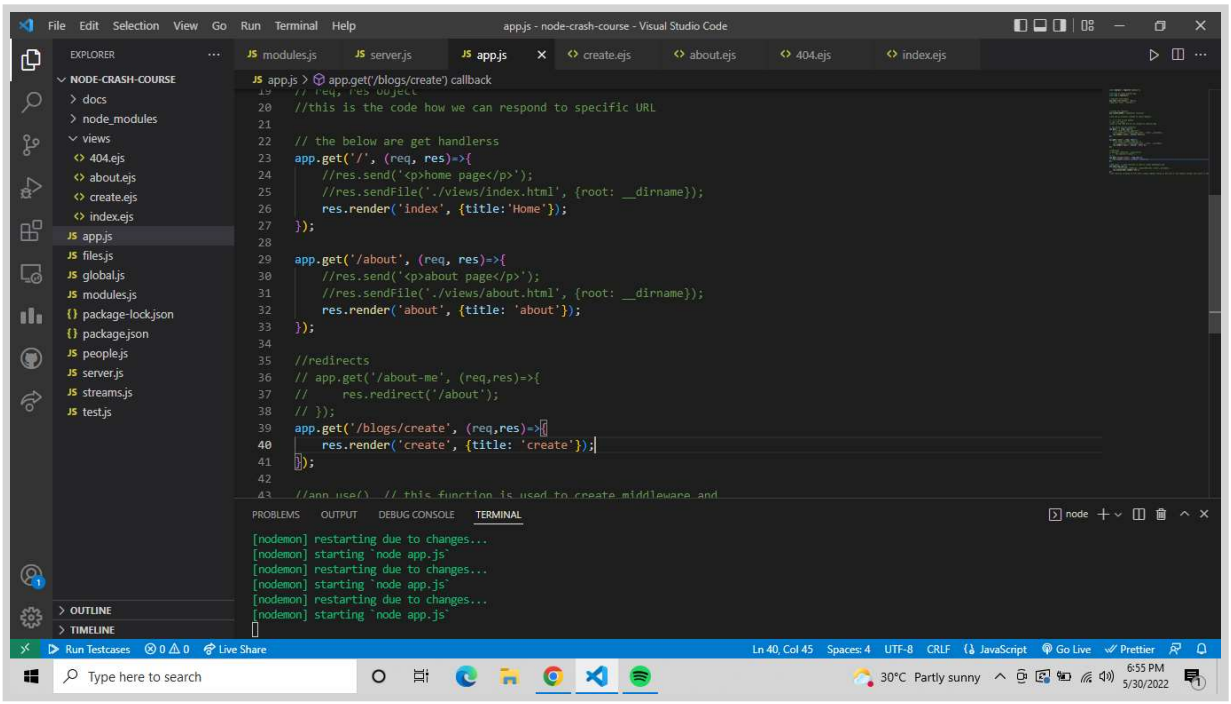

**but we want to pass data ...**

**so we can pass it using render method**

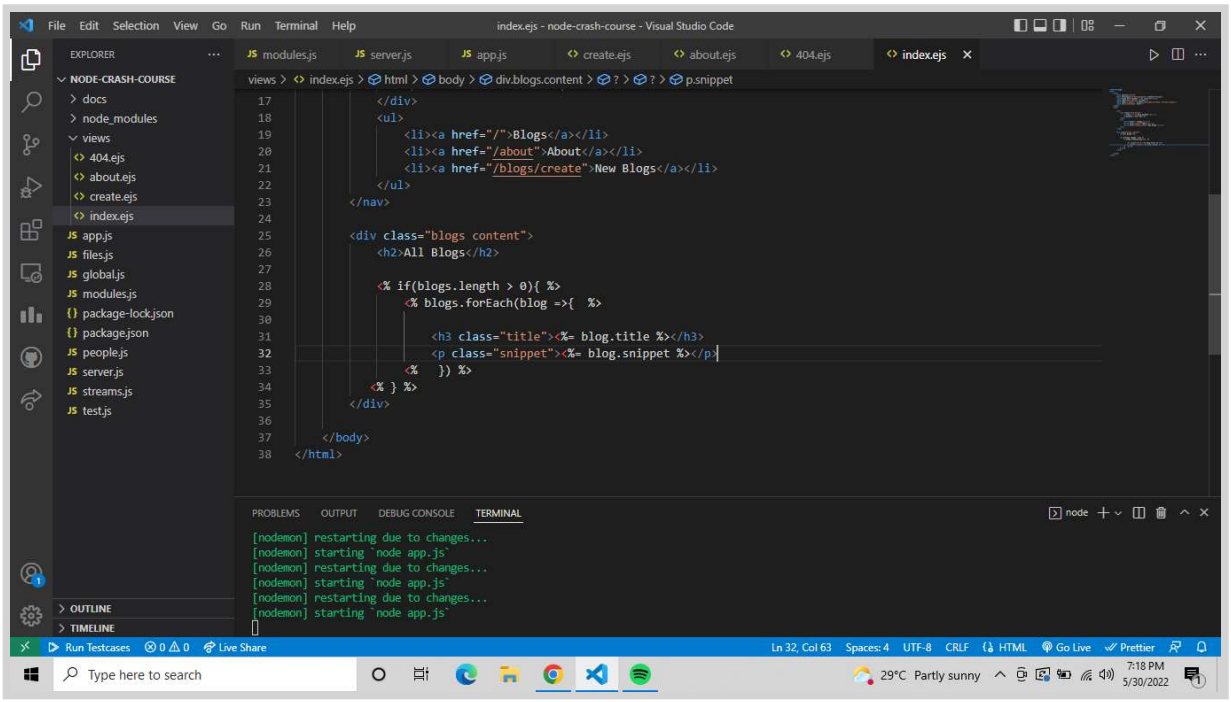
**by making the object...**

[14:32]






# if we want to output the array of blogs

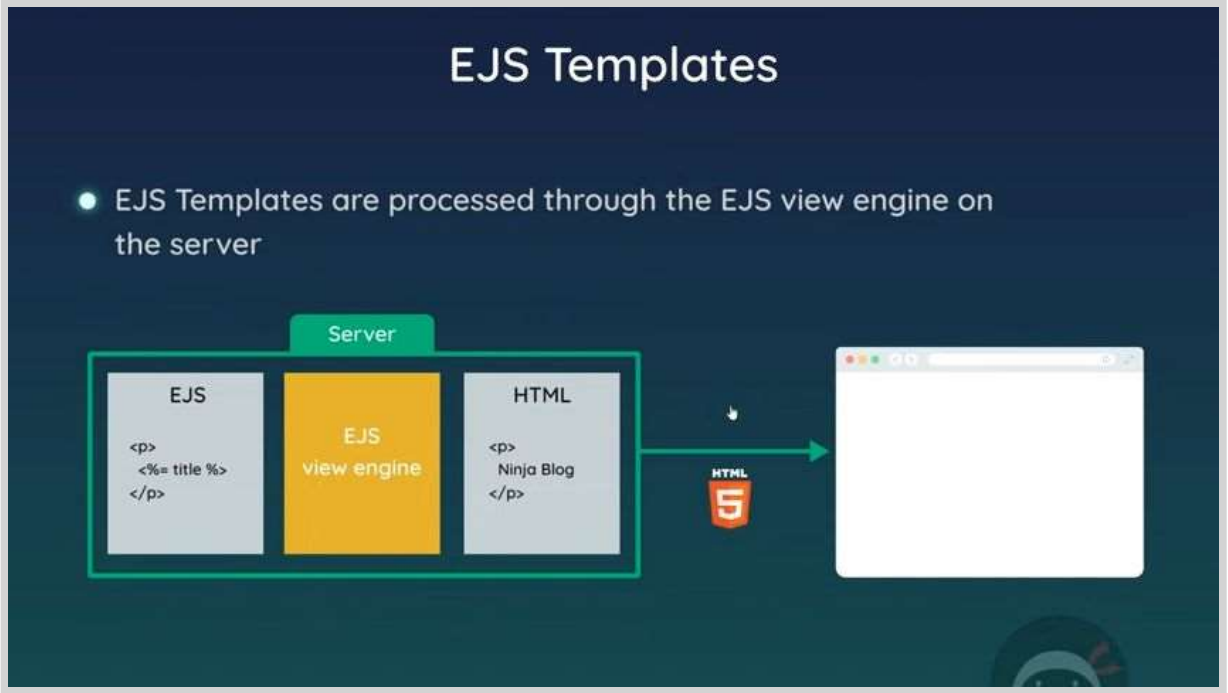



# ejs tags madhe Javascript ashya padhatine add kela

## views madhe data ksa insert karaycha te ..kela

## our view files live on the server....

23:55

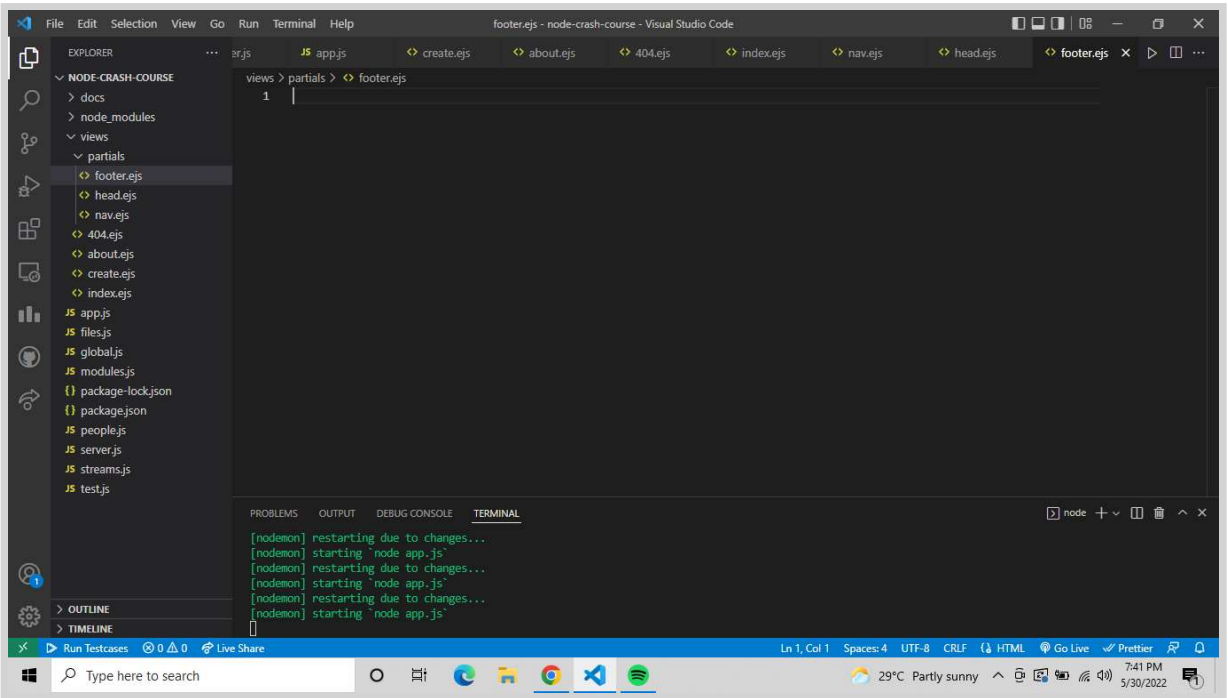


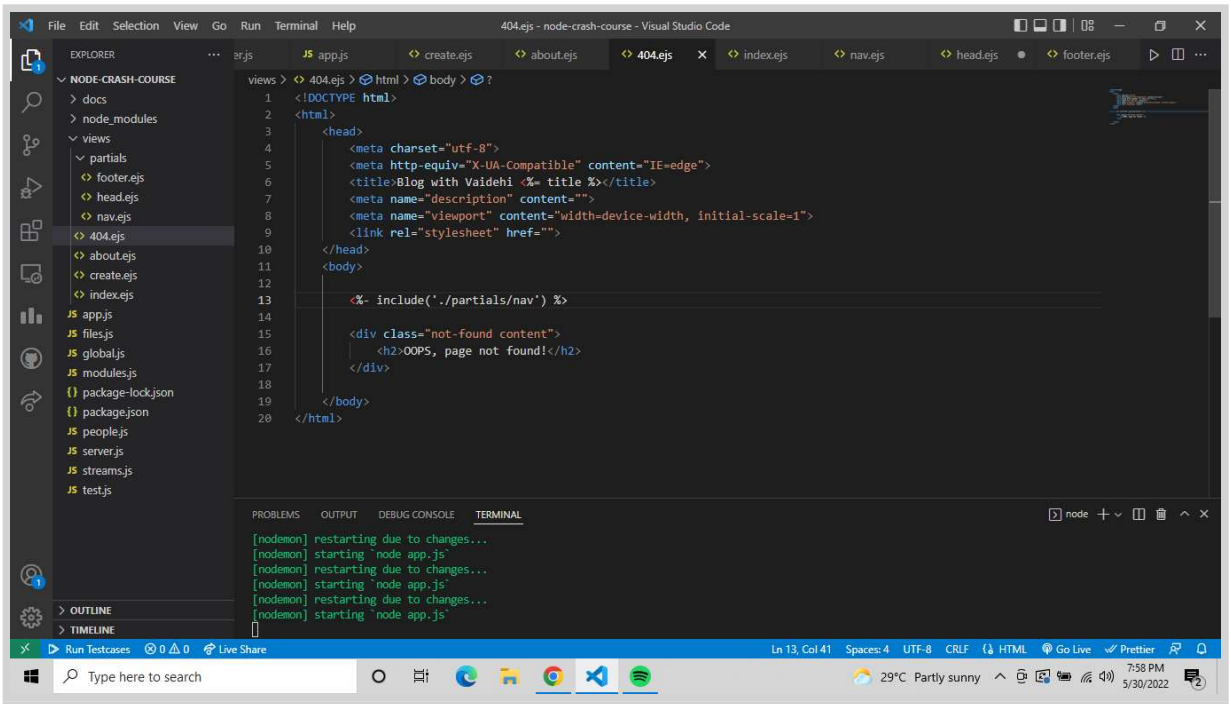

**This whole process is called server side rendering..**

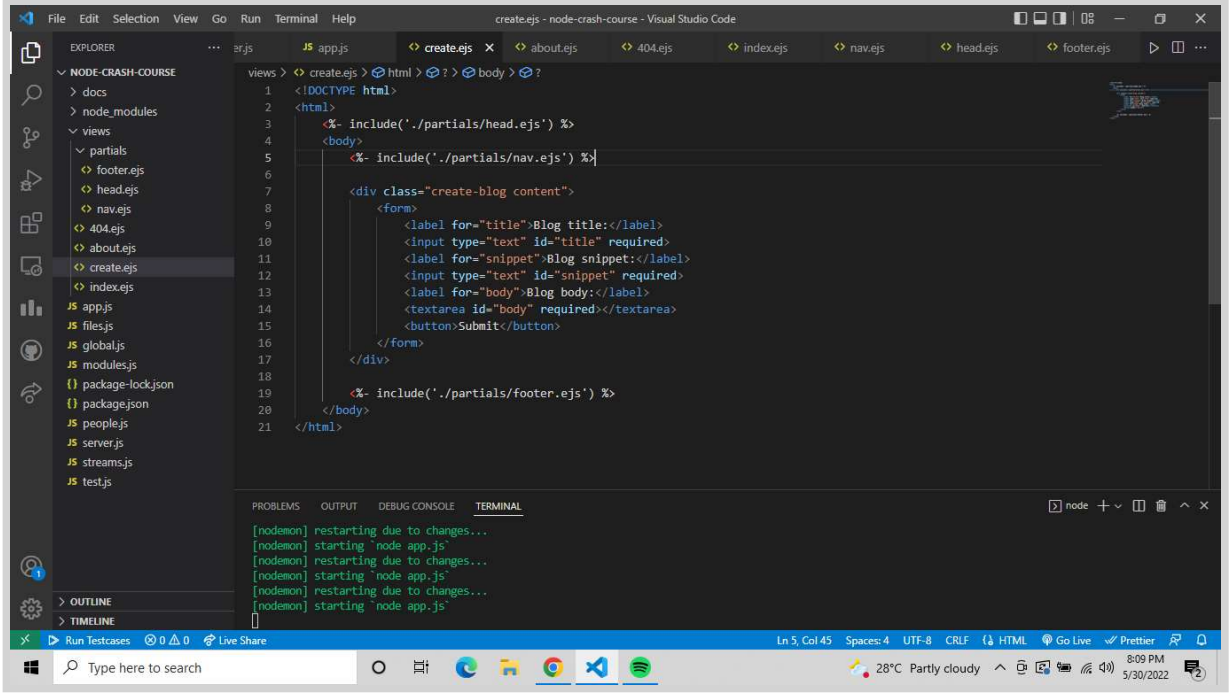
## Partials:

**ata bgha, aapn pratyek ejs page wr je nav lihily te suppose change karaychy tr pratyek page wr jaun karava lagel te... but he farch lengthy work ae soo..**

**(one file to update instead of individuals..must be there...so this type of files/templates are called partials)**






## //adding CSS:

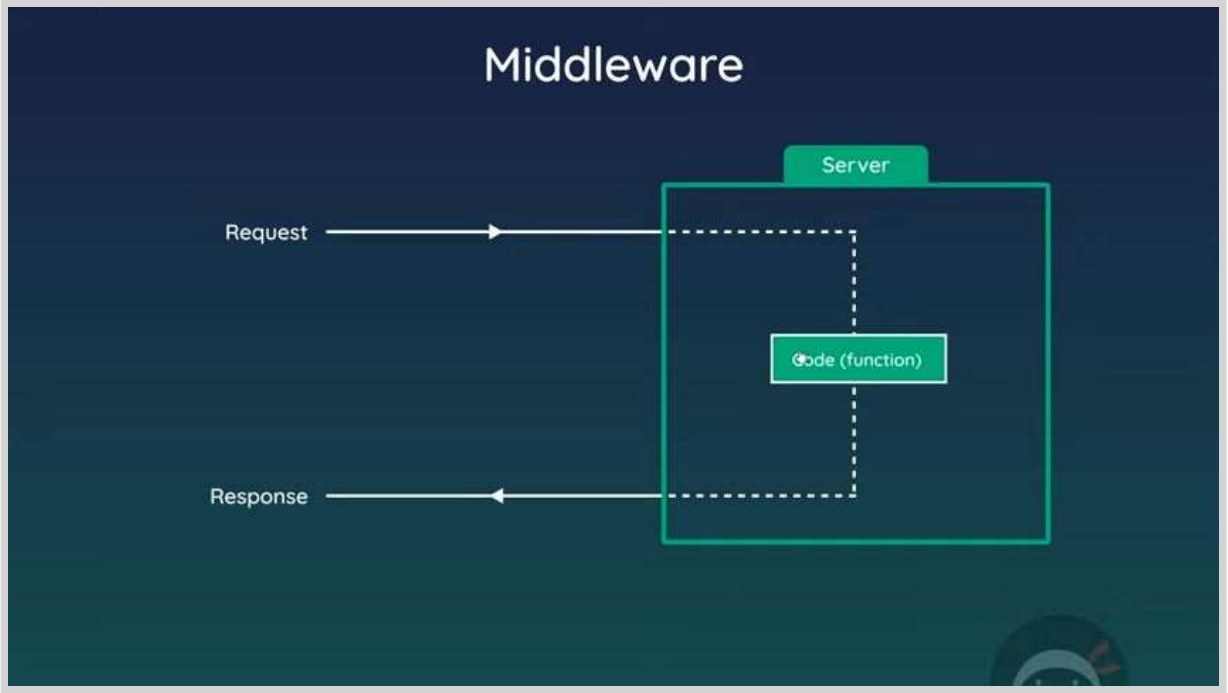
## serving static files using middle wares:

# Video8:

# Middlewares:

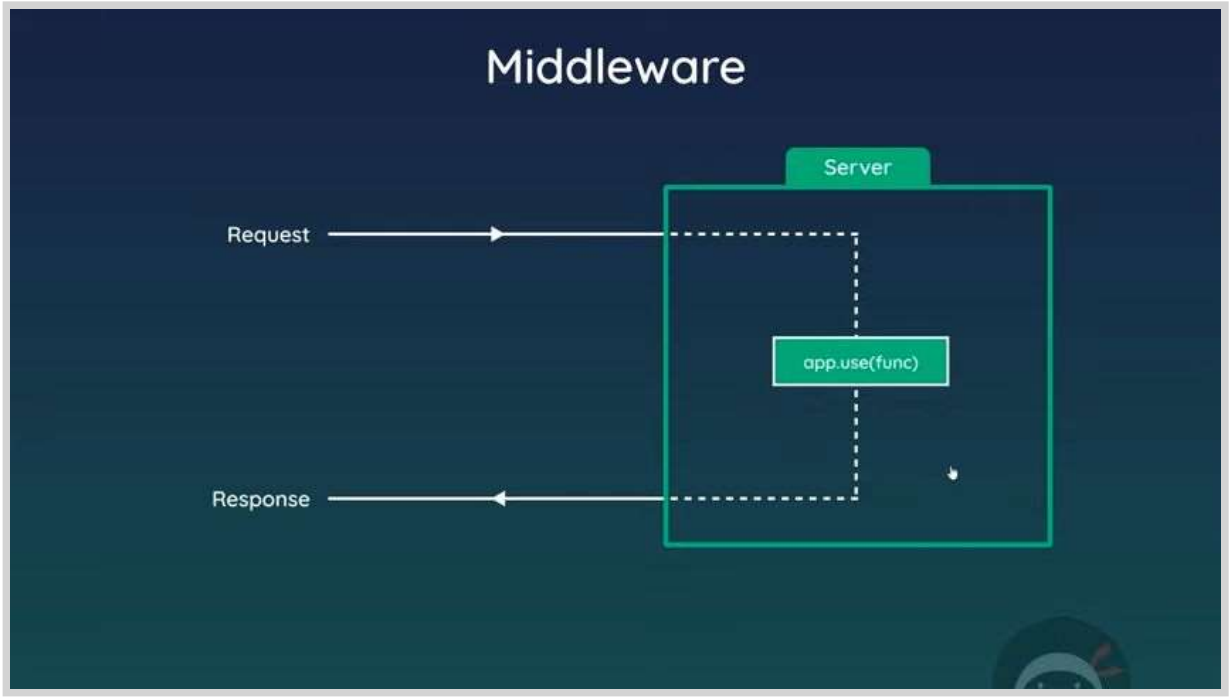
[00:03]


00:21

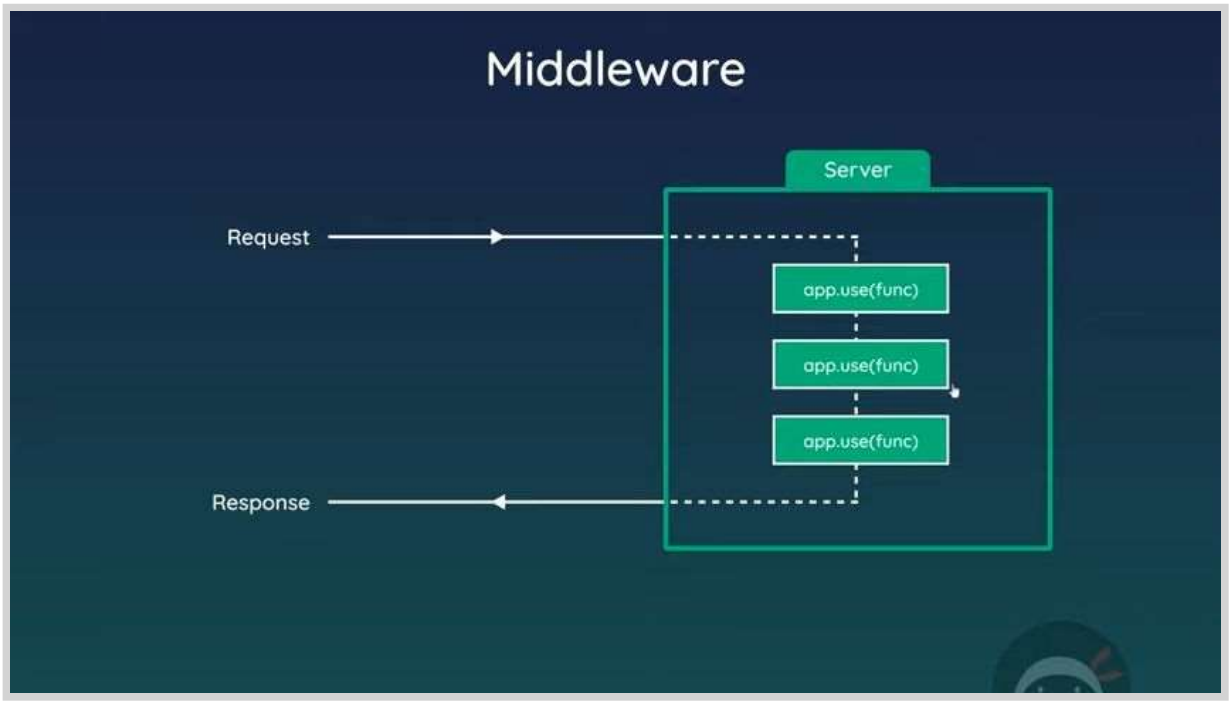



## use method is generally used as a middleware..

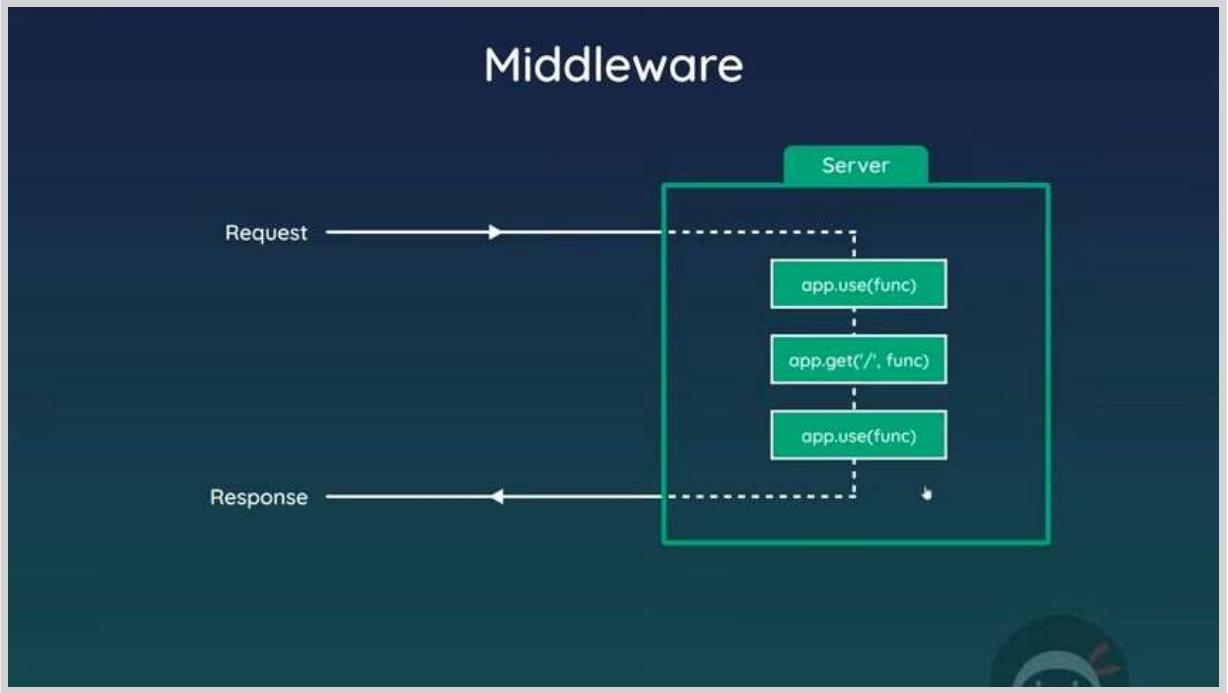
00:34


Middleware
Server
Request
app.use(func)
Response

00:45


Middleware
Server
Request
app.use(func)
app.use(func)
app.use(func)
Response

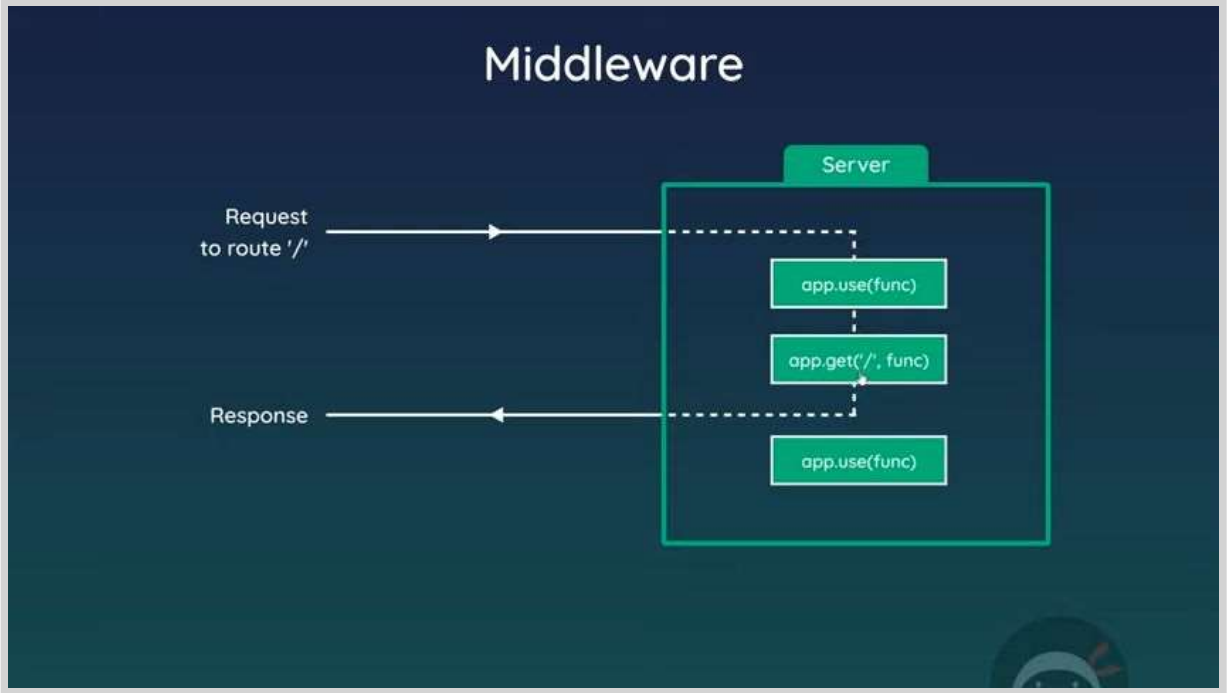
01:13




## use method => they run for all types of requests (get as well as post).....

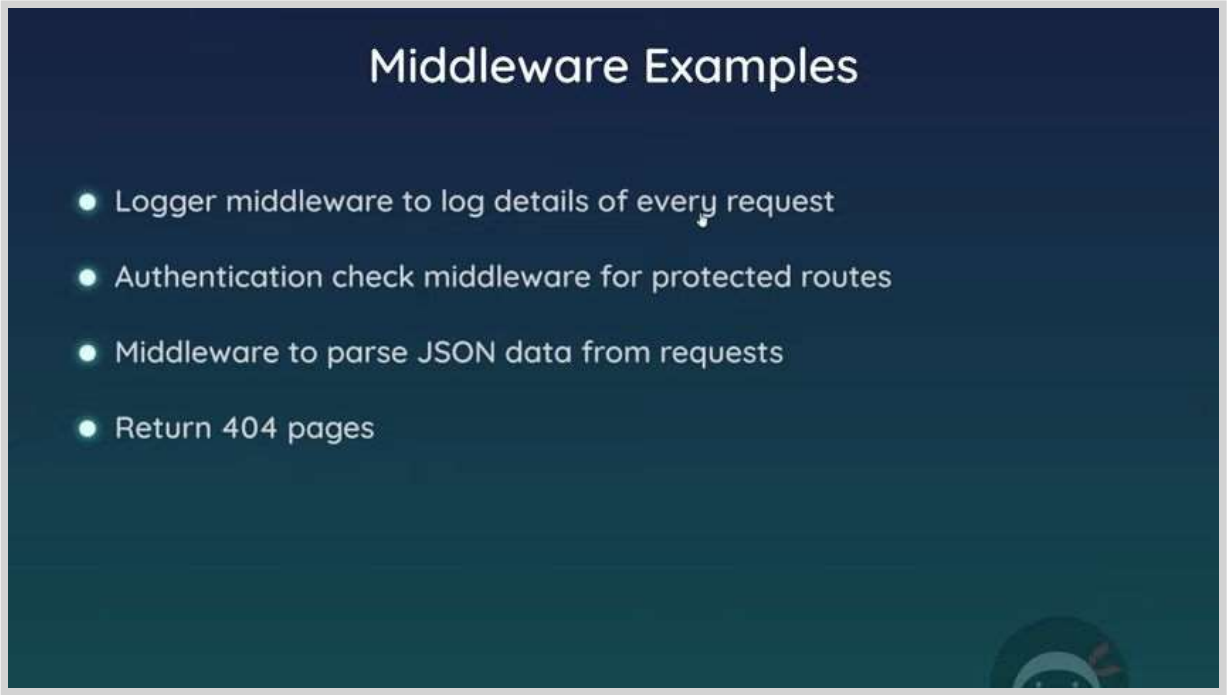
## app.get() is a get handler still middle ware

01:42




## inside (func) => we send response..

02:25

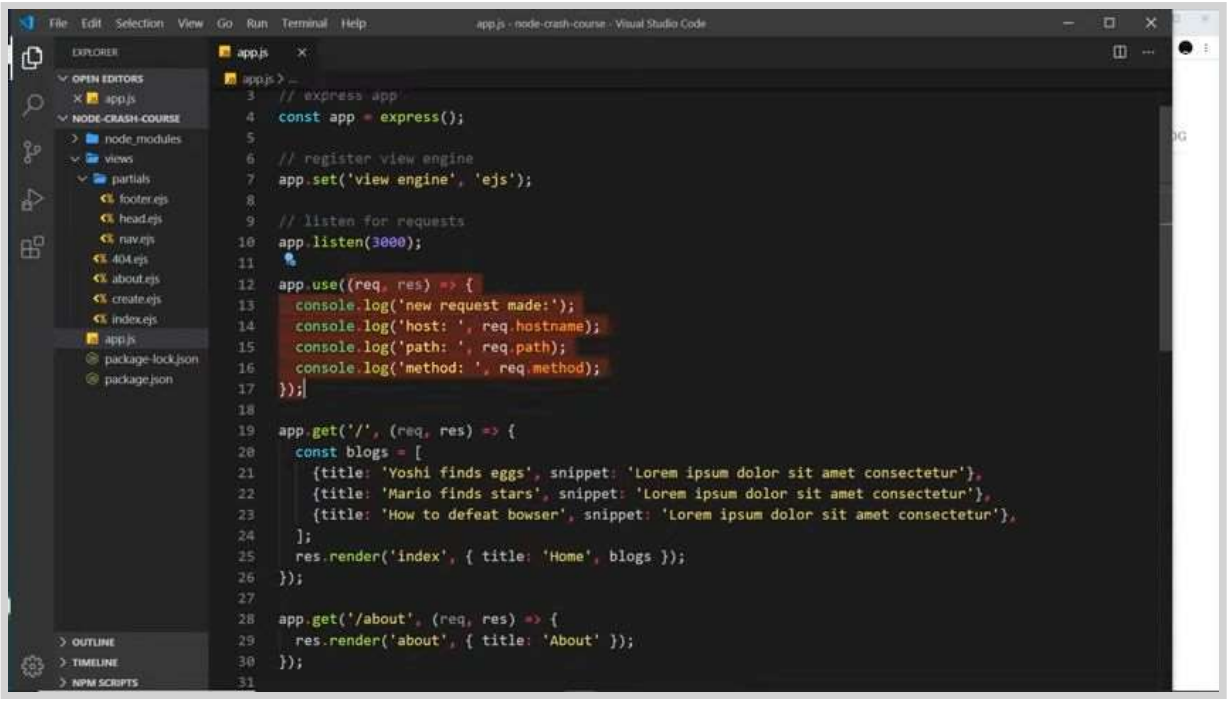


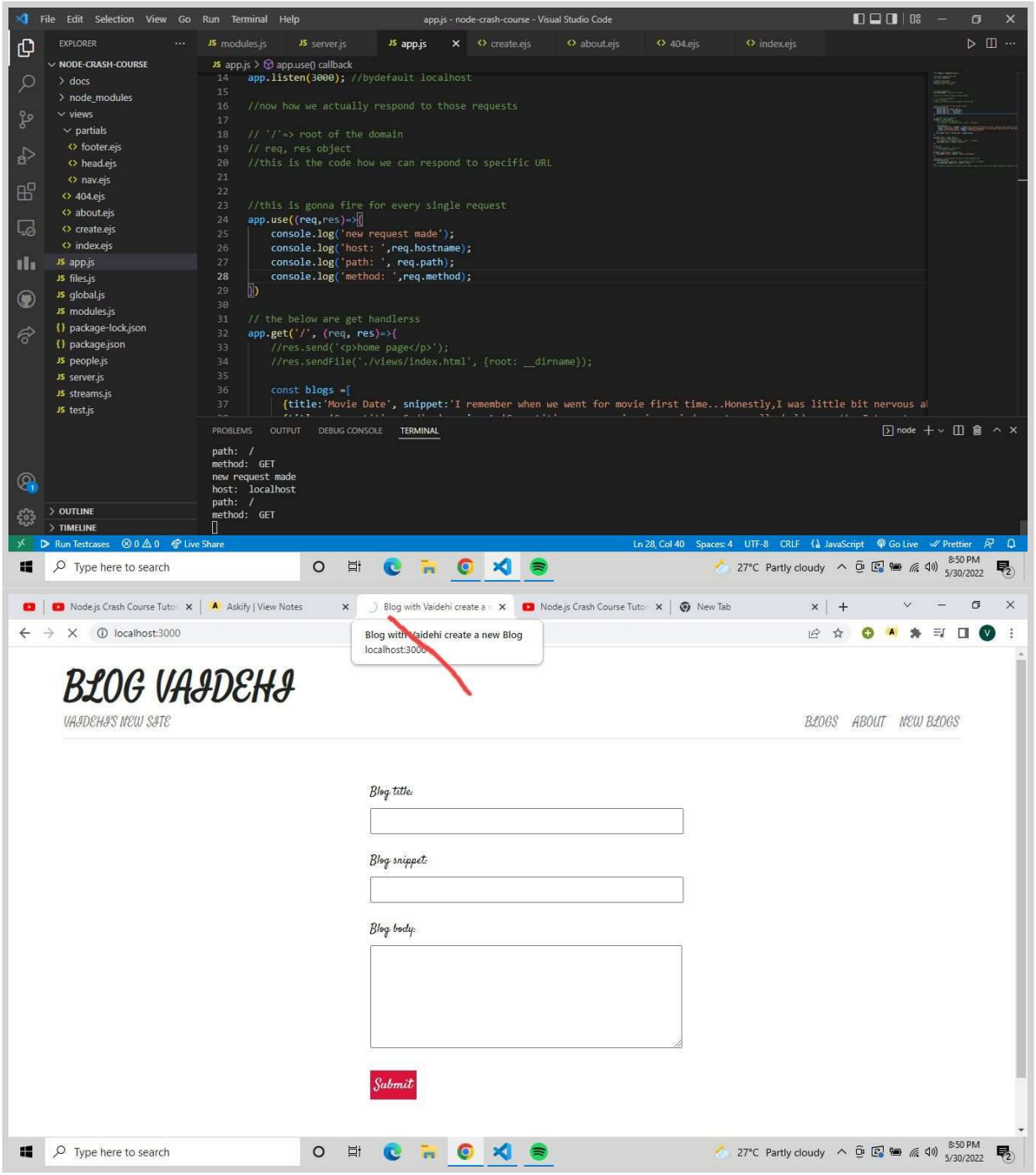

**Try to create own middleware in our code:**

**//now create our own middle ware..**

**for every request console wr mention kara..**

04:42



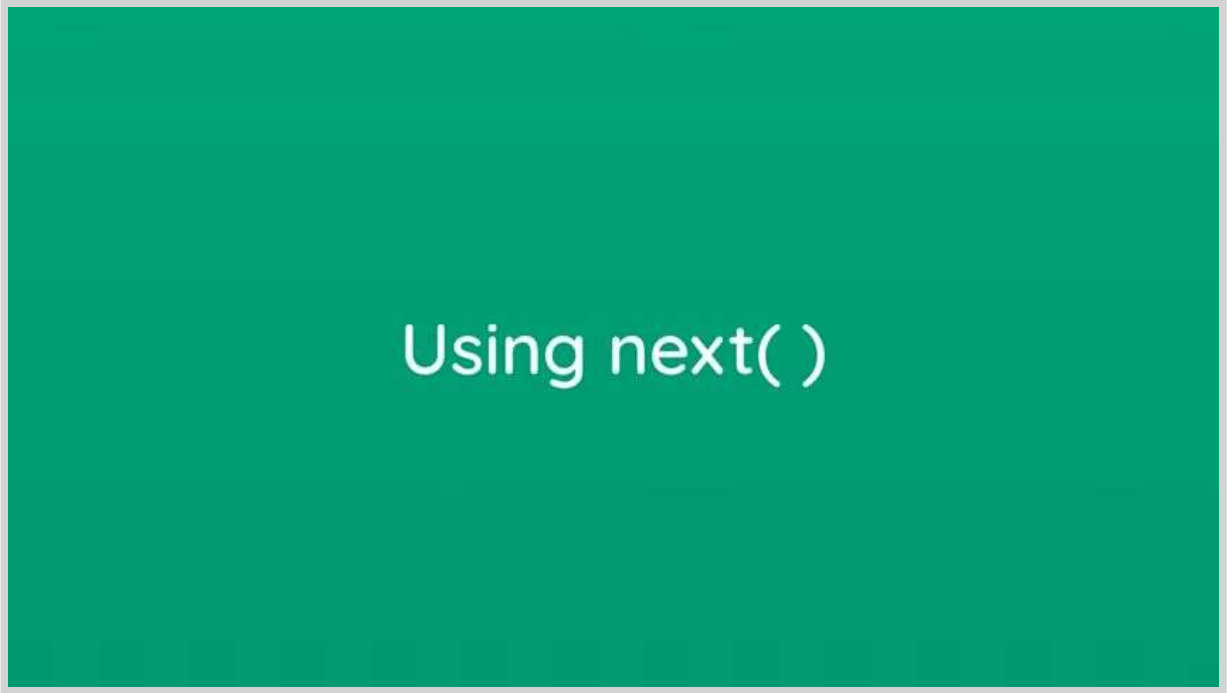



**But browser hang zaly, loadch hoty, dusrya page wr nahi jaty? asa ka??=> coz , jevha express app.use parynt jato ani te execute krto, nantr tyala kalat nahi ki ata kay krva??**

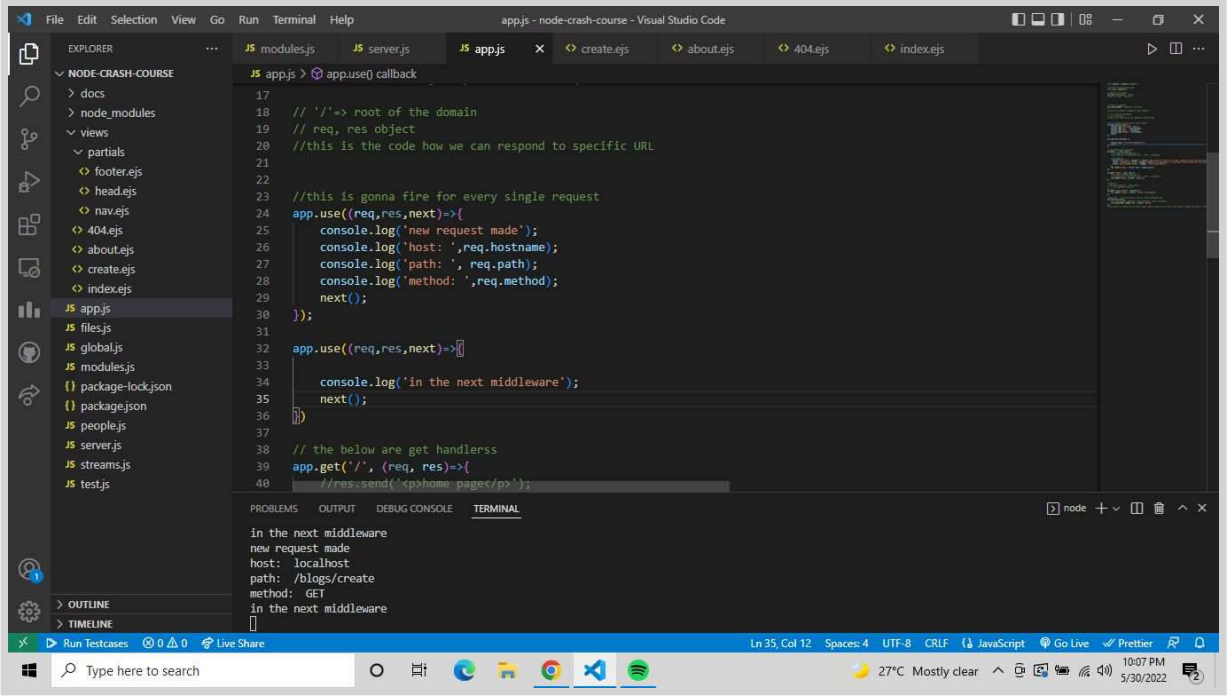
**express doesn't know automatically how to move on to the next middle ware?**

**=> tr mg using next() te karaych aapn**

[05:39]

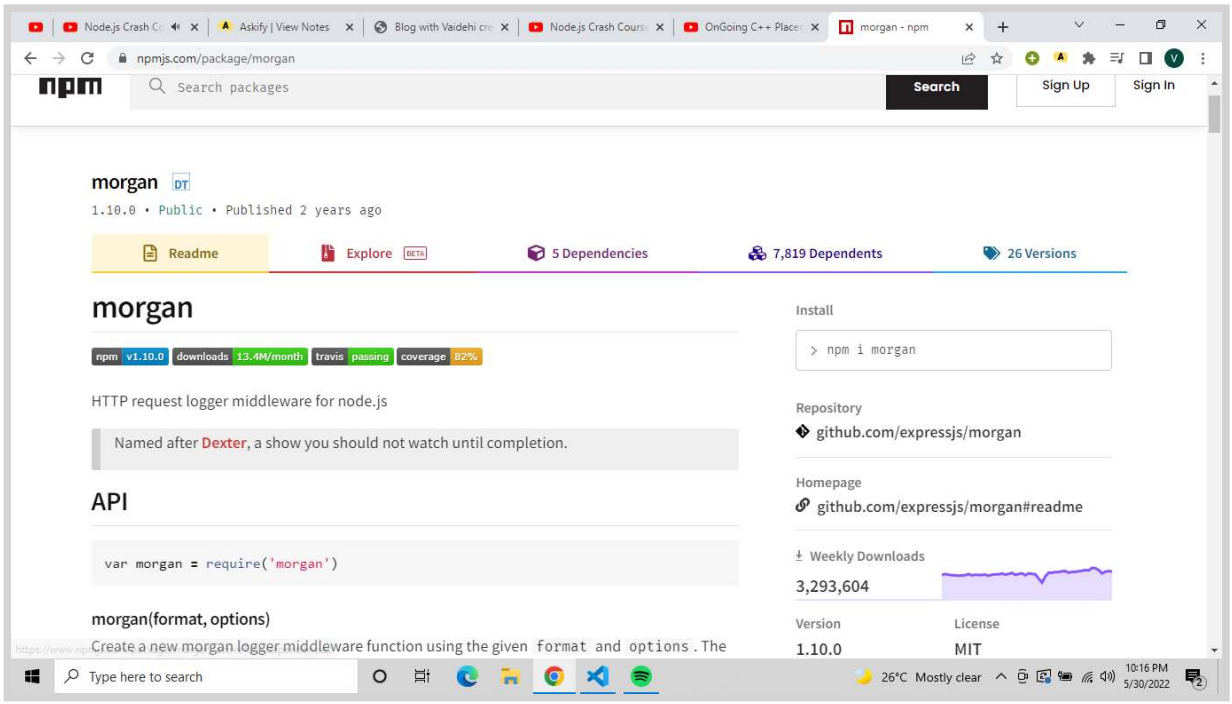


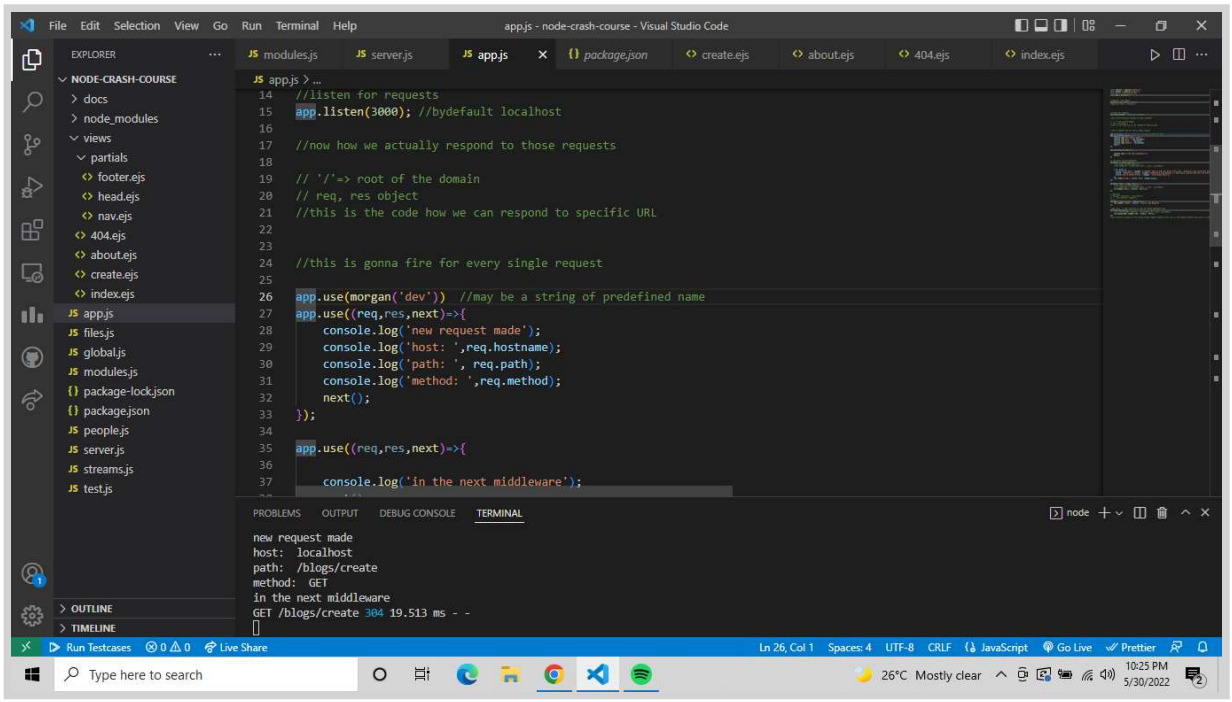

## now ata browser nahi hang hoty




## #3rd Party Middleware

## tons of middleware created already in express:

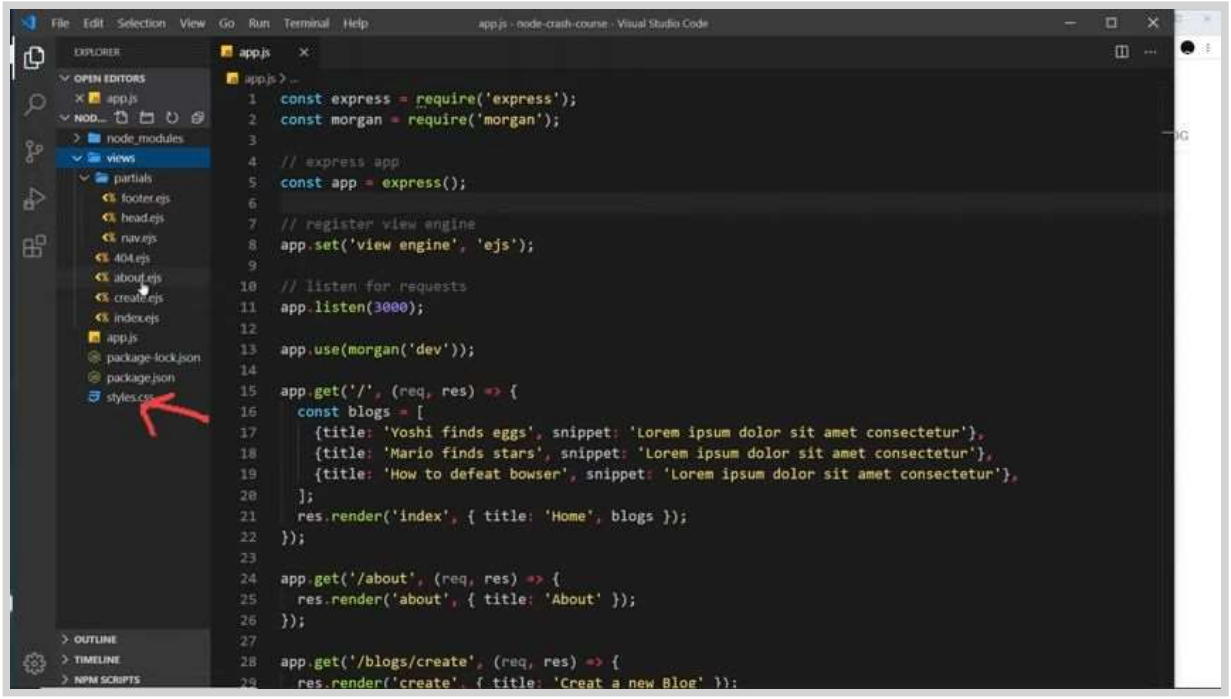



# //static files:

# => .css file nusta create krun access nahi kru shkt aapn ..

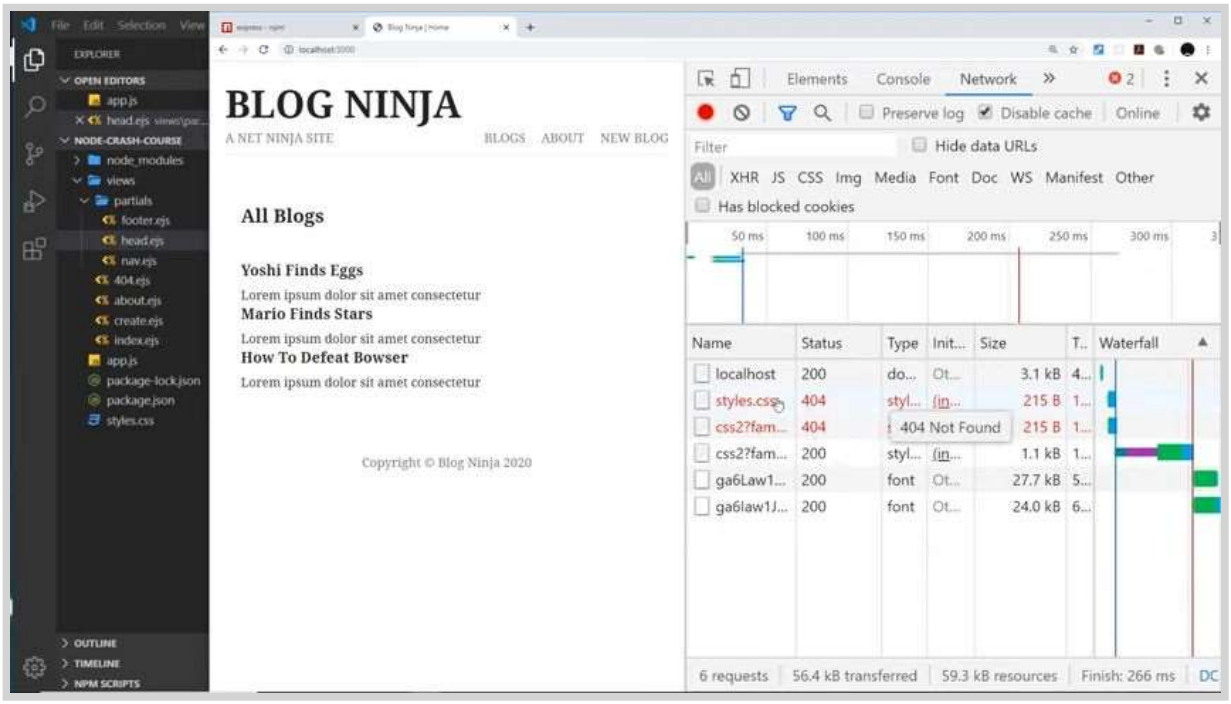
12:27




**this wouldn't work for the browser...**

**see , style.css gives 404 error**

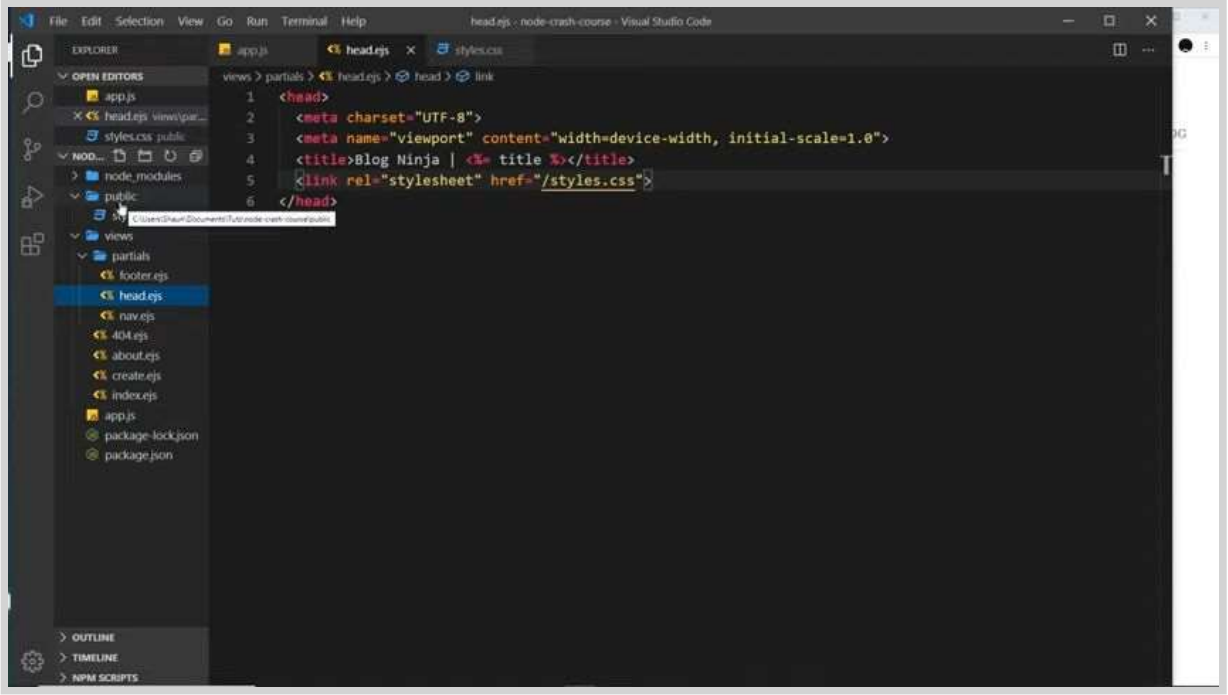
[13:02]

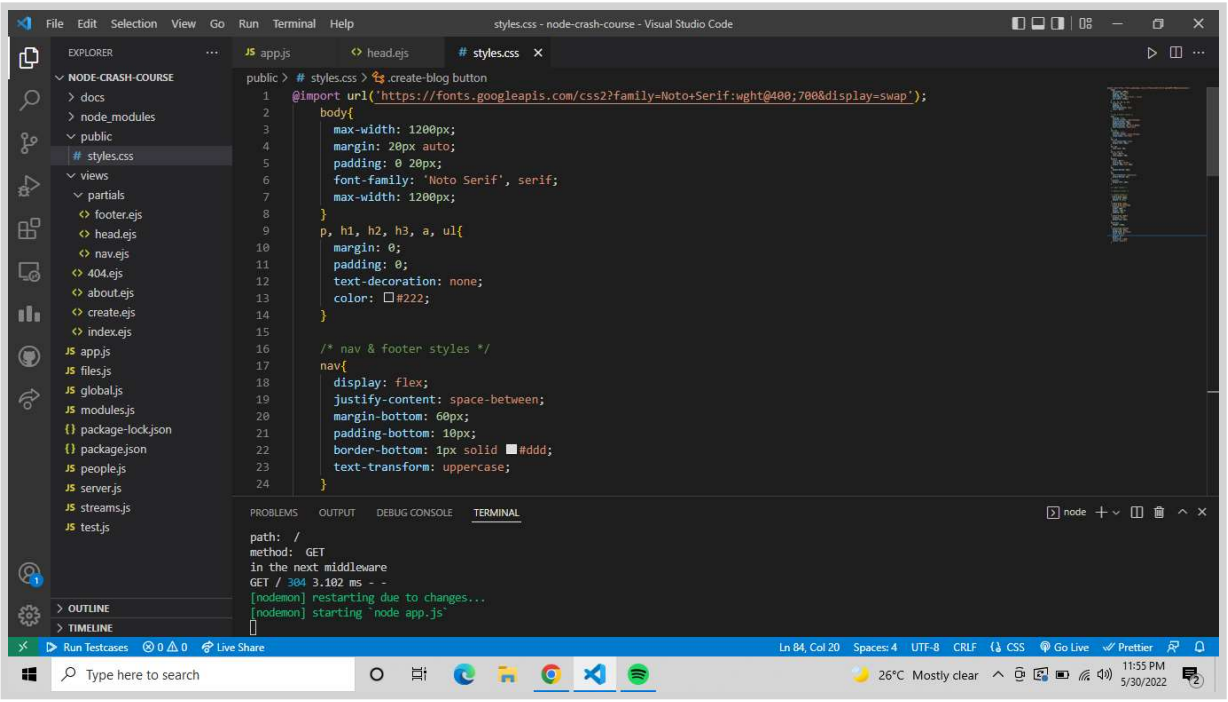



**//so middleware static banvu:**

**app.use(express.static(foldername));**

[15:44]




# Video9:

# //MongoDB

# =>

[00:32]

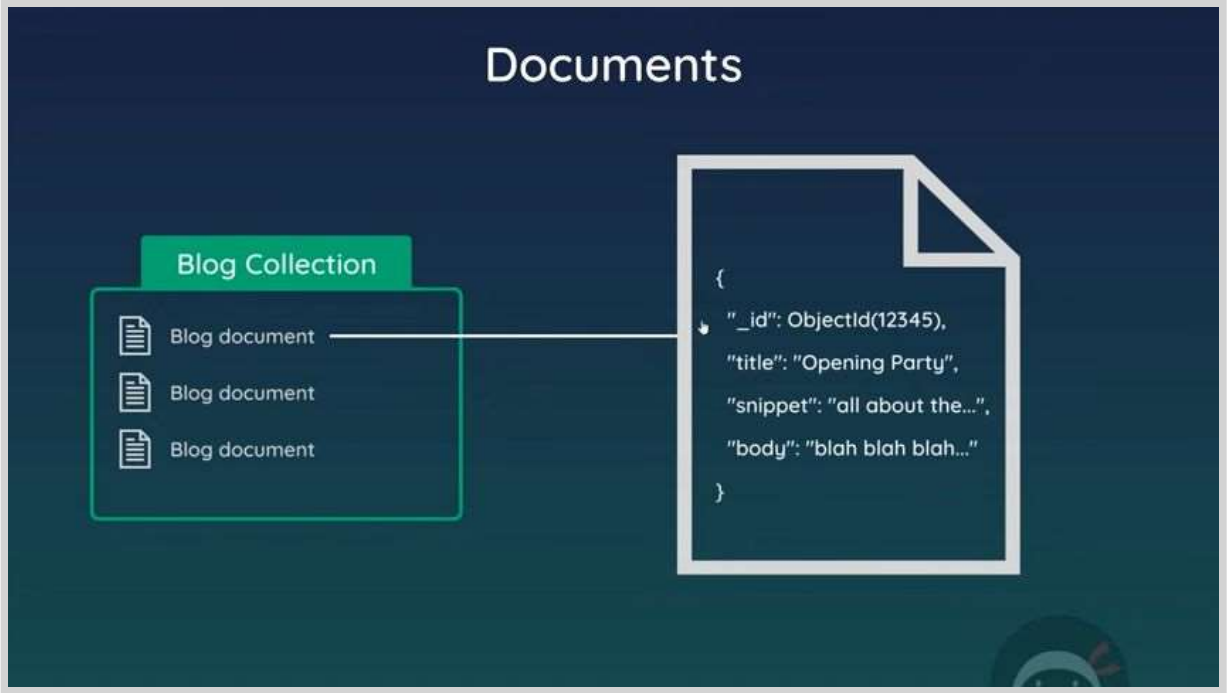

## //how it works:

01:04


## //each collection contain only one type of document:

## JSON Object chya form madhe

01:47

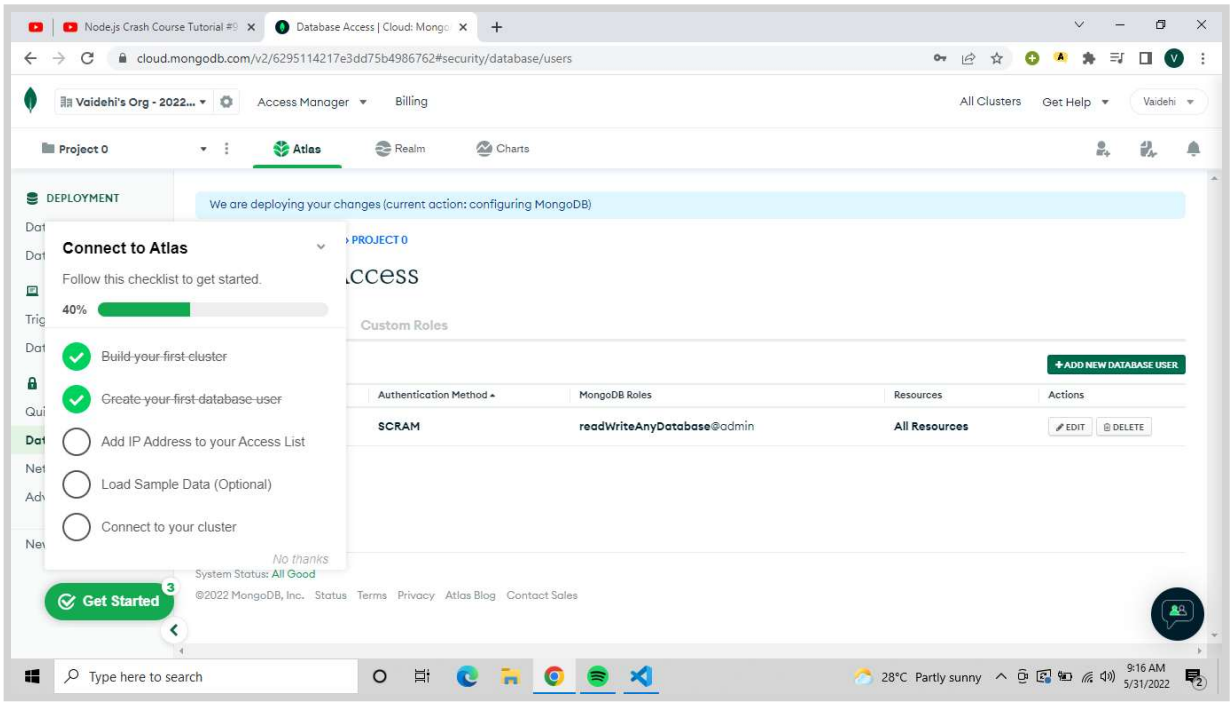


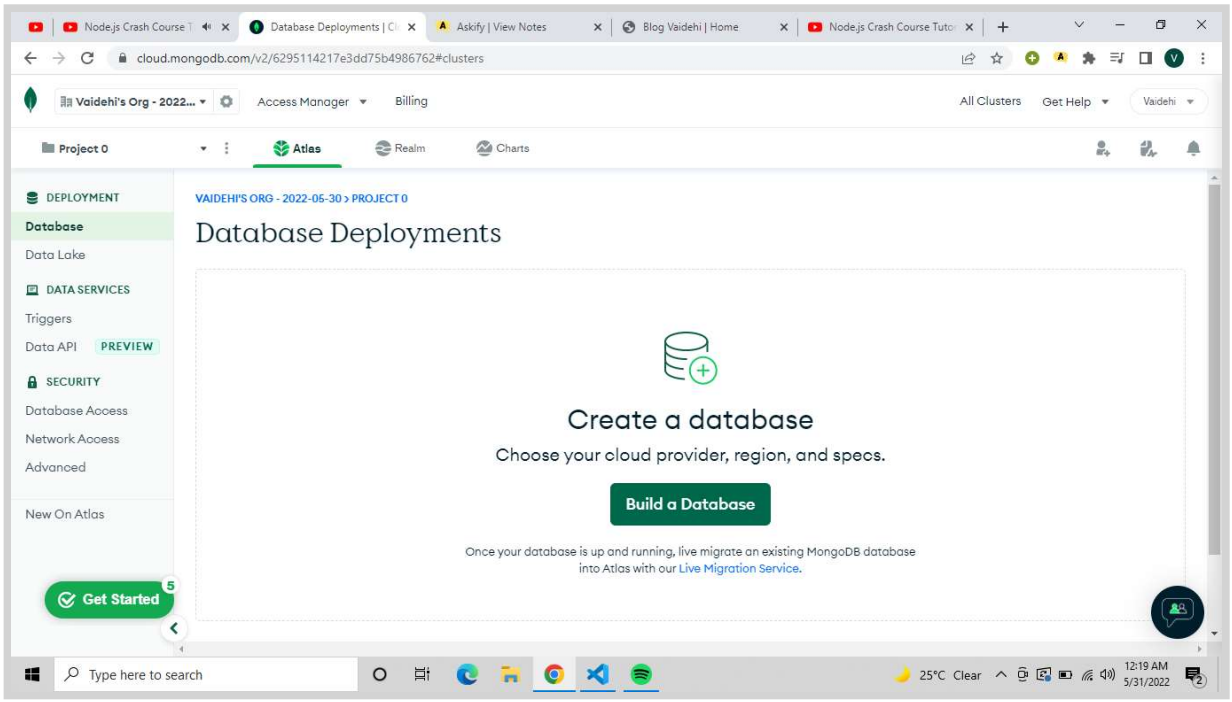

02:25

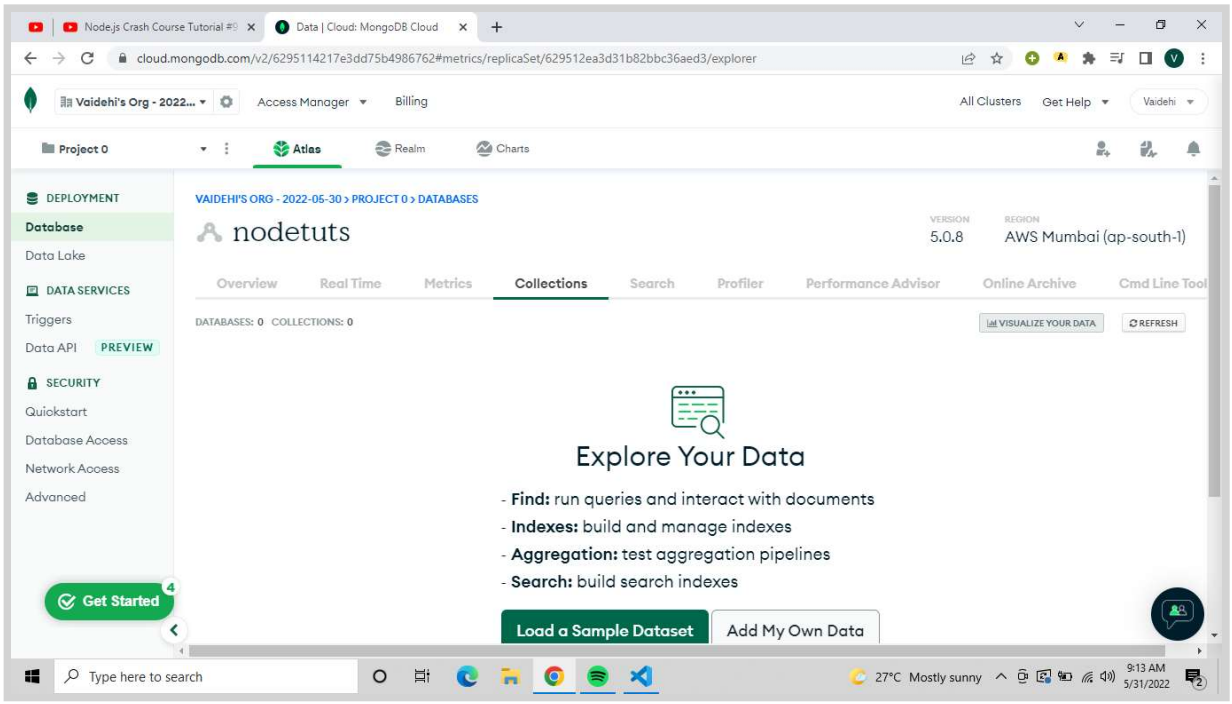
**southern when we're working with MongoDB there's a couple of different setup options that we can use we can either install MongoDB on our computer locally and use and then deploy that or we could use a cloud database which is already hosted for us and it can be quite a lot**

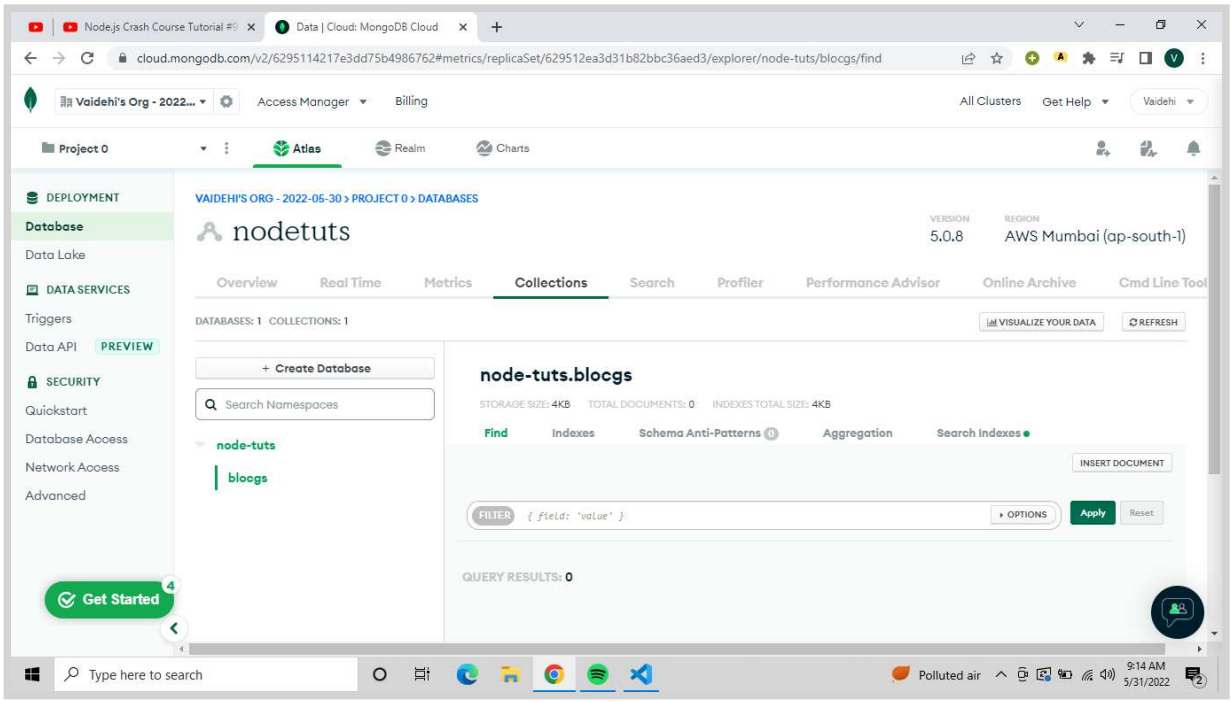
**//Mongodb setup & Atlas**

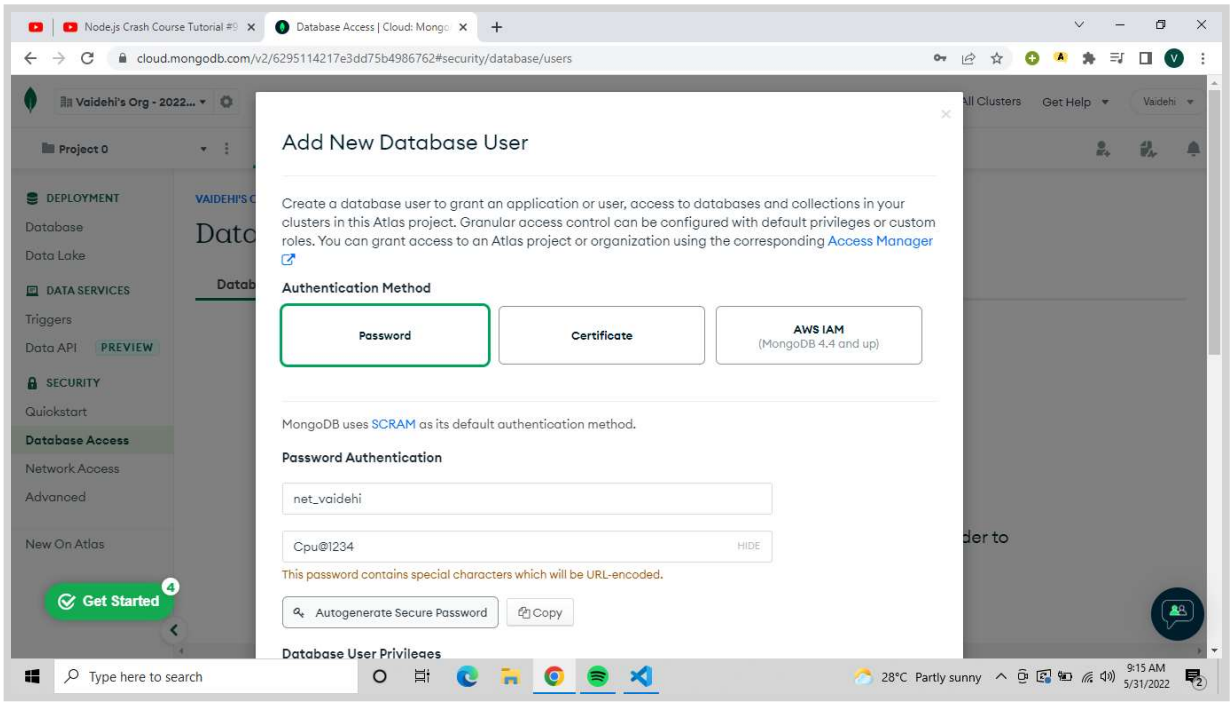
https://cloud.mongodb.com/v2/6295114217e3dd75b4986762#clusters

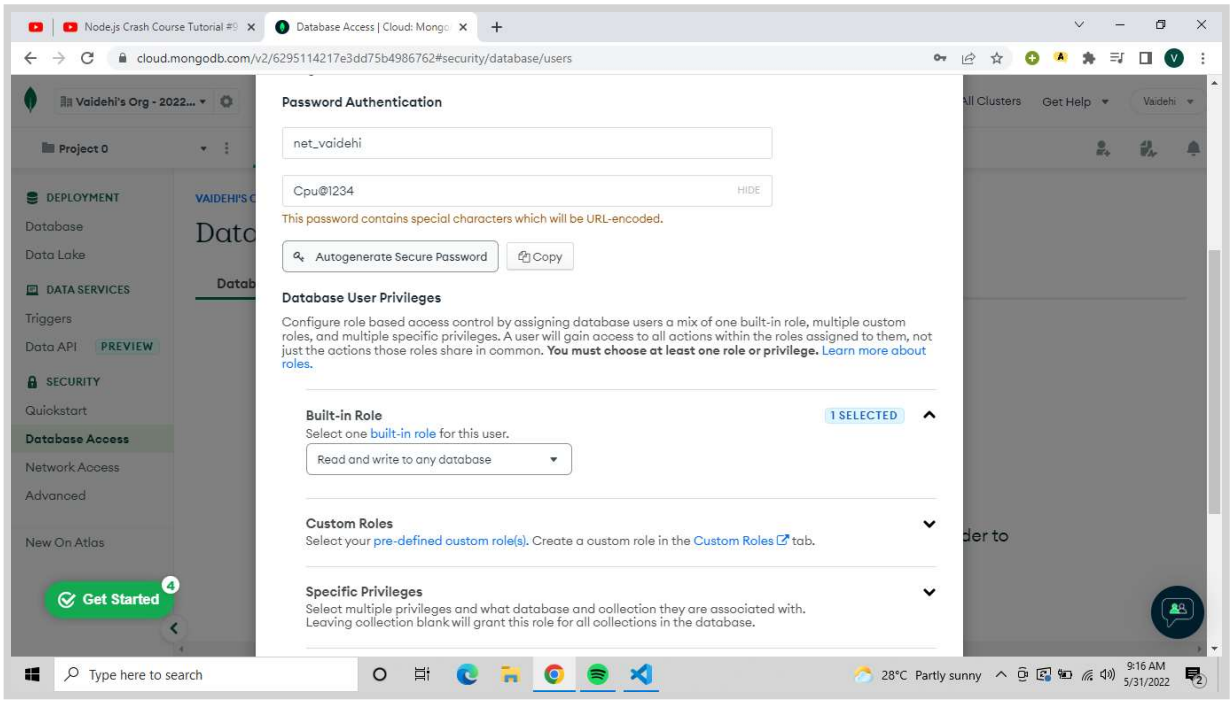

Node.js Crash Course Tutorial #5
Database Access | Cloud: Mongo
cloud.mongodb.com/v2/6295114217e3dd75b4986762#security/database/users
Vaidehi's Org - 2022...
Access Manager
Billing
All Clusters
Get Help
Vaidehi
Project 0
Atlas
Realm
Charts
DEPLOYMENT
We are deploying your changes (current action: configuring MongoDB)
Connect to Atlas
Follow this checklist to get started.
40%
Build your first cluster
Create your first database user
Add IP Address to your Access List
Load Sample Data (Optional)
Connect to your cluster
No thanks
PROJECT 0
Custom Roles
+ ADD NEW DATABASE USER
Authentication Method
MongoDB Roles
Resources
Actions
SCRAM
readWriteAnyDatabase@admin
All Resources
EDIT
DELETE
Get Started
System Status: All Good
©2022 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales
Type here to search
28°C Partly sunny
9:16 AM
5/31/2022

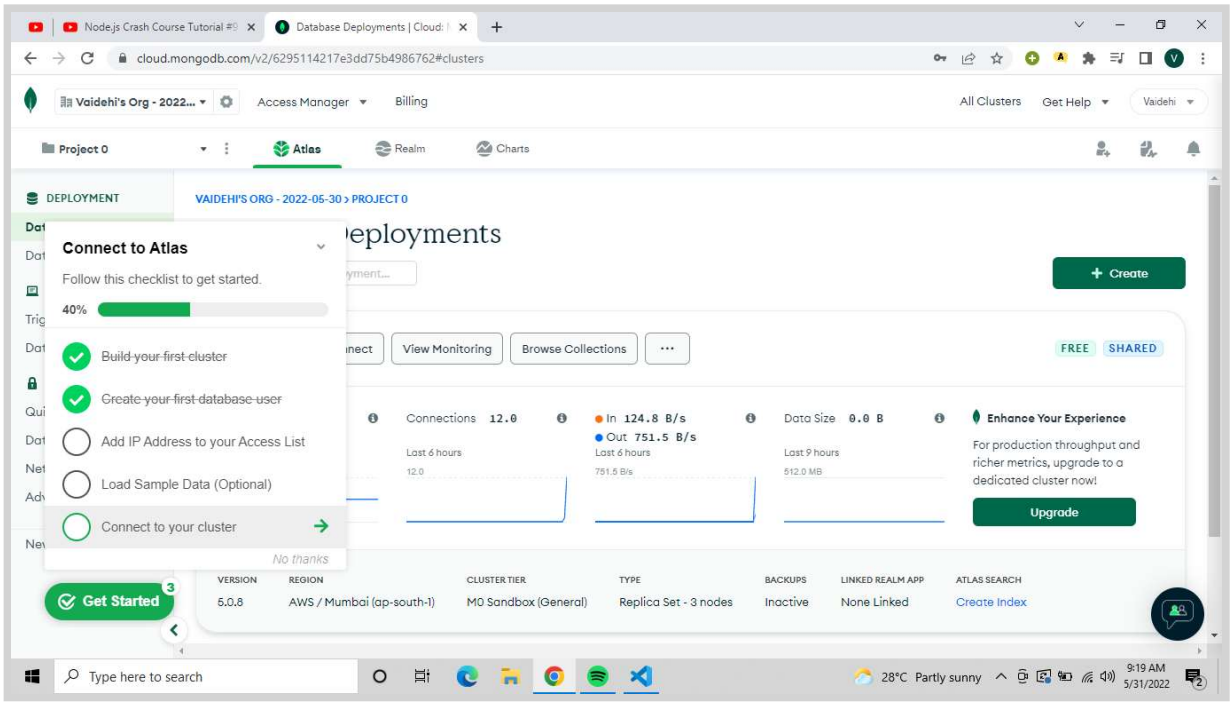

Node.js Crash Course T
Database Deployments | Cl
Askify | View Notes
Blog Vaidehi | Home
Node.js Crash Course Tutor
cloud.mongodb.com/v2/6295114217e3dd75b4986762#clusters
Vaidehi's Org - 2022...
Access Manager
Billing
All Clusters
Get Help
Vaidehi
Project 0
Atlas
Realm
Charts
DEPLOYMENT
Database
Data Lake
DATA SERVICES
Triggers
Data API
PREVIEW
SECURITY
Database Access
Network Access
Advanced
New On Atlas
VAIDEHI'S ORG - 2022-05-30 > PROJECT 0
Database Deployments
Create a database
Choose your cloud provider, region, and specs.
Build a Database
Once your database is up and running, live migrate an existing MongoDB database into Atlas with our Live Migration Service.
Get Started
Type here to search
25°C Clear
12:19 AM
5/31/2022

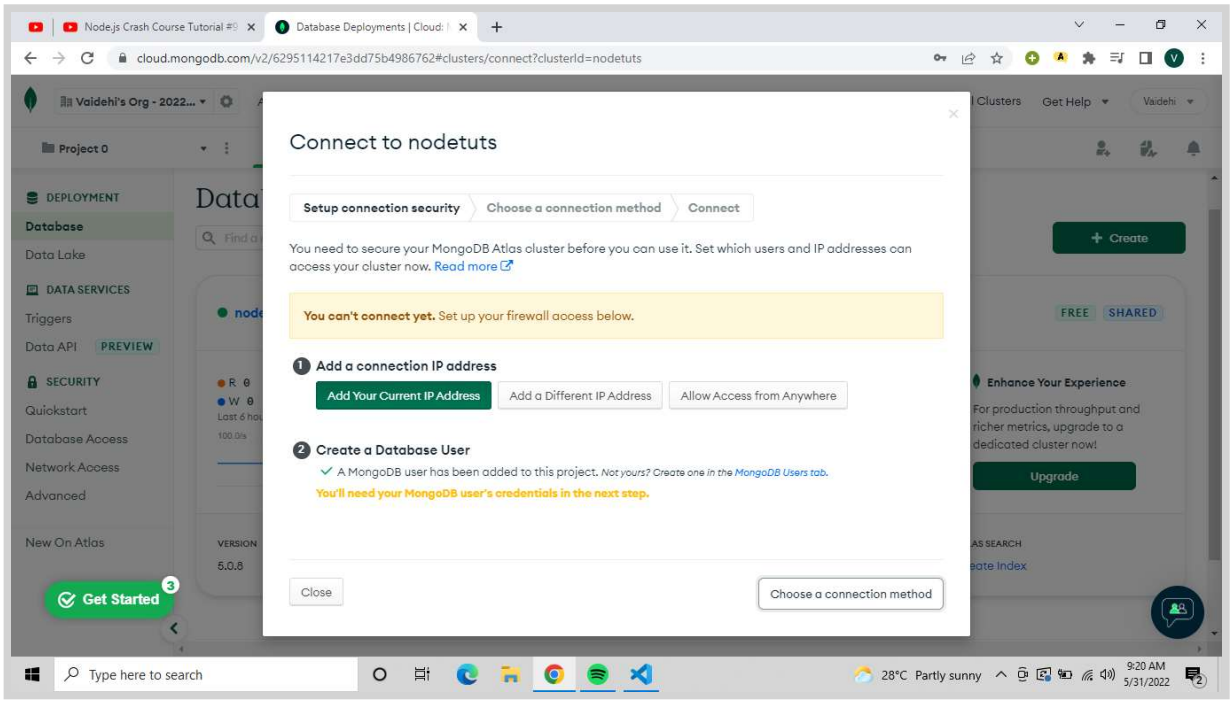

Node.js Crash Course Tutorial #5
Data | Cloud: MongoDB Cloud
cloud.mongodb.com/v2/6295114217e3dd75b4986762#metrics/replicaSet/629512ea3d31b82bbc36aed3/explorer
Vaidehi's Org - 2022...
Access Manager
Billing
All Clusters
Get Help
Vaidehi
Project 0
Atlas
Realm
Charts
DEPLOYMENT
Database
Data Lake
DATA SERVICES
Triggers
Data API
PREVIEW
SECURITY
Quickstart
Database Access
Network Access
Advanced
VAIDEHI'S ORG - 2022-05-30 > PROJECT 0 > DATABASES
nodetuts
VERSION
5.0.8
REGION
AWS Mumbai (ap-south-1)
Overview
Real Time
Metrics
Collections
Search
Profiler
Performance Advisor
Online Archive
Cmd Line Tool
DATABASES: 0 COLLECTIONS: 0
VISUALIZE YOUR DATA
REFRESH
Explore Your Data
- Find: run queries and interact with documents
- Indexes: build and manage indexes
- Aggregation: test aggregation pipelines
- Search: build search indexes
Load a Sample Dataset
Add My Own Data
Get Started
Type here to search
27°C Mostly sunny
9:13 AM
5/31/2022

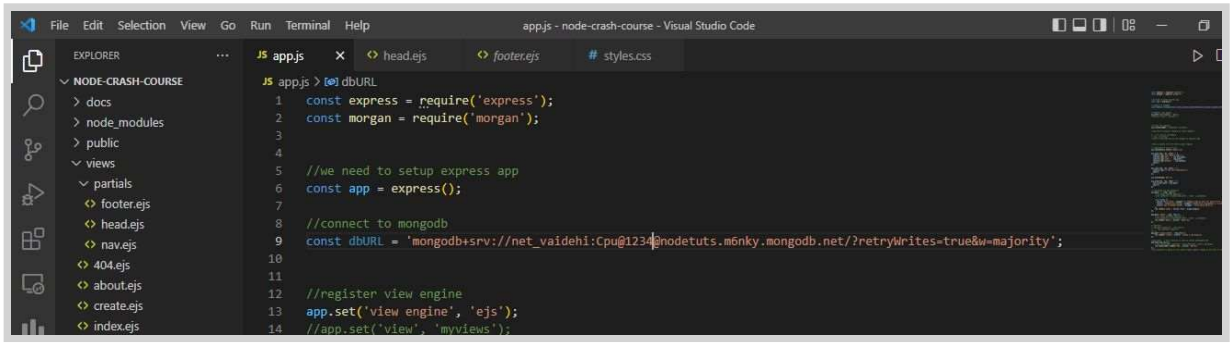

Node.js Crash Course Tutorial #5
Data | Cloud: MongoDB Cloud
cloud.mongodb.com/v2/6295114217e3dd75b4986762#metrics/replicaSet/629512ea3d31b82bbc36aed3/explorer/node-tuts/blocgs/find
Vaidehi's Org - 2022...
Access Manager
Billing
All Clusters
Get Help
Vaidehi
Project 0
Atlas
Realm
Charts
DEPLOYMENT
Database
Data Lake
DATA SERVICES
Triggers
Data API
PREVIEW
SECURITY
Quickstart
Database Access
Network Access
Advanced
VAIDEHI'S ORG - 2022-05-30 > PROJECT 0 > DATABASES
nodetuts
VERSION
5.0.8
REGION
AWS Mumbai (ap-south-1)
Overview
Real Time
Metrics
Collections
Search
Profiler
Performance Advisor
Online Archive
Cmd Line Tool
DATABASES: 1 COLLECTIONS: 1
VISUALIZE YOUR DATA
REFRESH
+ Create Database
Search Namespaces
node-tuts
blocgs
node-tuts.blocgs
STORAGE SIZE: 4KB
TOTAL DOCUMENTS: 0
INDEXES TOTAL SIZE: 4KB
Find
Indexes
Schema Anti-Patterns
Aggregation
Search Indexes
INSERT DOCUMENT
FILTER
{ field: 'value' }
OPTIONS
Apply
Reset
QUERY RESULTS: 0
Get Started
Type here to search
Polluted air
9:14 AM
5/31/2022


Node.js Crash Course Tutorial #5
Database Access | Cloud: Mongo
cloud.mongodb.com/v2/6295114217e3dd75b4986762#security/database/users
Add New Database User
Create a database user to grant an application or user, access to databases and collections in your clusters in this Atlas project. Granular access control can be configured with default privileges or custom roles. You can grant access to an Atlas project or organization using the corresponding Access Manager
Authentication Method
Password
Certificate
AWS IAM
(MongoDB 4.4 and up)
MongoDB uses SCRAM as its default authentication method.
Password Authentication
net_vaidehi
Cpu@1234
HIDE
This password contains special characters which will be URL-encoded.
Autogenerate Secure Password
Copy
Database User Privileges
Get Started
Type here to search
28°C Partly sunny
9:15 AM
5/31/2022


Node.js Crash Course Tutorial #5
Database Access | Cloud: Mongo
cloud.mongodb.com/v2/6295114217e3dd75b4986762#security/database/users
Password Authentication
net_vaidehi
Cpu@1234
HIDE
This password contains special characters which will be URL-encoded.
Autogenerate Secure Password
Copy
Database User Privileges
Configure role based access control by assigning database users a mix of one built-in role, multiple custom roles, and multiple specific privileges. A user will gain access to all actions within the roles assigned to them, not just the actions those roles share in common. You must choose at least one role or privilege. Learn more about roles.
Built-in Role
1 SELECTED
Select one built-in role for this user.
Read and write to any database
Custom Roles
Select your pre-defined custom role(s). Create a custom role in the Custom Roles tab.
Specific Privileges
Select multiple privileges and what database and collection they are associated with.
Leaving collection blank will grant this role for all collections in the database.
Get Started
Type here to search
28°C Partly sunny
9:16 AM
5/31/2022


Node.js Crash Course Tutorial #5
Database Deployments | Cloud:
cloud.mongodb.com/v2/6295114217e3dd75b4986762#clusters
Vaidehi's Org - 2022...
Access Manager
Billing
All Clusters
Get Help
Vaidehi
Project 0
Atlas
Realm
Charts
DEPLOYMENT
VAIDEHI'S ORG - 2022-05-30 > PROJECT 0
Connect to Atlas
Follow this checklist to get started.
40%
Build your first cluster
Create your first database user
Add IP Address to your Access List
Load Sample Data (Optional)
Connect to your cluster
No thanks
Create
View Monitoring
Browse Collections
FREE
SHARED
Connections 12.0
In 124.8 B/s
Out 751.5 B/s
Data Size 0.0 B
Enhance Your Experience
For production throughput and richer metrics, upgrade to a dedicated cluster now!
Upgrade
Get Started
VERSION
REGION
CLUSTER TIER
TYPE
BACKUPS
LINKED REALM APP
ATLAS SEARCH
5.0.8
AWS / Mumbai (ap-south-1)
M0 Sandbox (General)
Replica Set - 3 nodes
Inactive
None Linked
Create Index
Type here to search
28°C Partly sunny
9:19 AM
5/31/2022

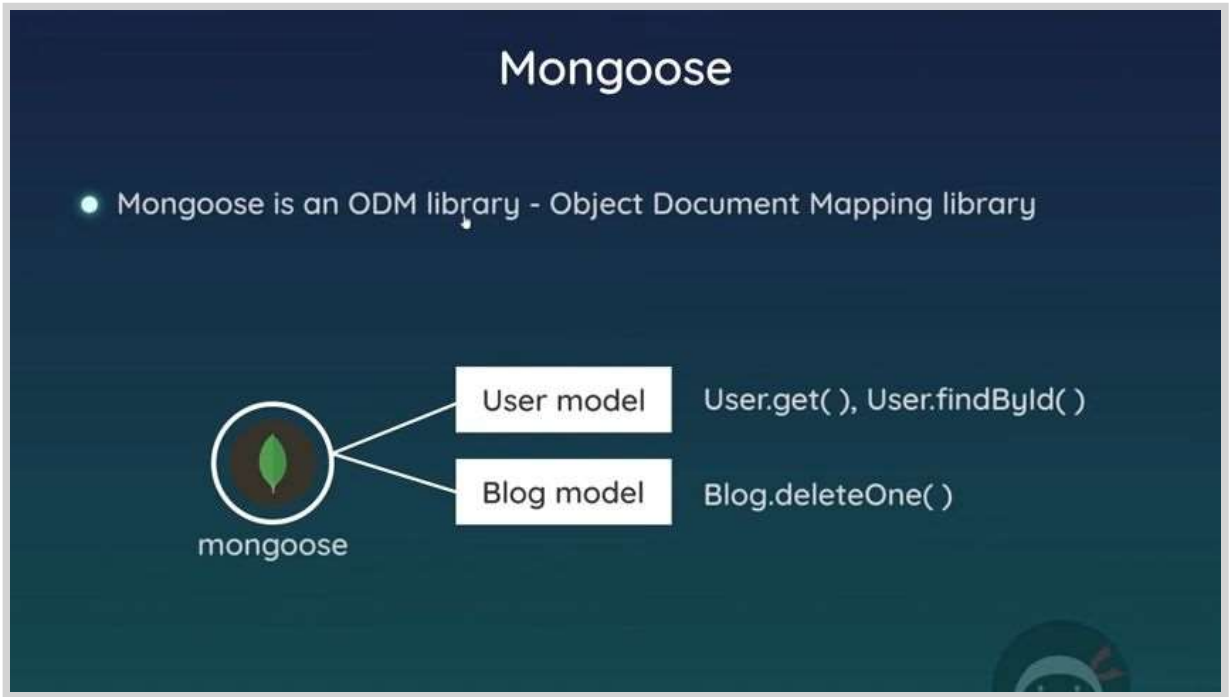

Node.js Crash Course Tutorial #5
Database Deployments | Cloud:
cloud.mongodb.com/v2/6295114217e3dd75b4986762#clusters/connect?clusterId=nodetuts
Connect to nodetuts
Setup connection security
Choose a connection method
Connect
You need to secure your MongoDB Atlas cluster before you can use it. Set which users and IP addresses can access your cluster now. Read more
You can't connect yet. Set up your firewall access below.
1 Add a connection IP address
Add Your Current IP Address
Add a Different IP Address
Allow Access from Anywhere
2 Create a Database User
A MongoDB user has been added to this project. Not yours? Create one in the MongoDB Users tab.
You'll need your MongoDB user's credentials in the next step.
Close
Choose a connection method
Get Started
Type here to search
28°C Partly sunny
9:20 AM
5/31/2022


File Edit Selection View Go Run Terminal Help
app.js - node-crash-course - Visual Studio Code
EXPLORER
NODE-CRASH-COURSE
docs
node_modules
public
views
partials
footer.ejs
head.ejs
nav.ejs
404.ejs
about.ejs
create.ejs
index.ejs
app.js
head.ejs
footer.ejs
styles.css
const express = require('express');
const morgan = require('morgan');
//we need to setup express app
const app = express();
//connect to mongodb
const dbURL = 'mongodb+srv://net_vaidehi:Cpu@1234@nodetuts.m6nky.mongodb.net/?retryWrites=true&w=majority';
//register view engine
app.set('view engine', 'ejs');

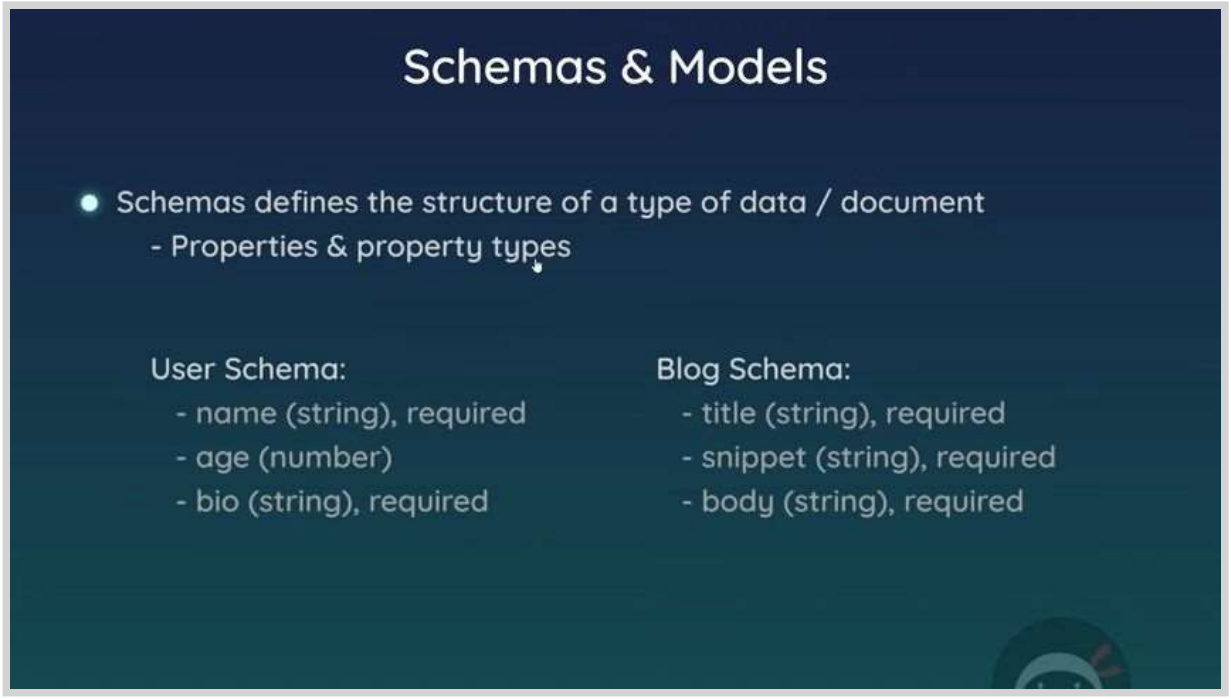
# //plain mongodb API package we can use but , use of Mongoose is easy...

06:53

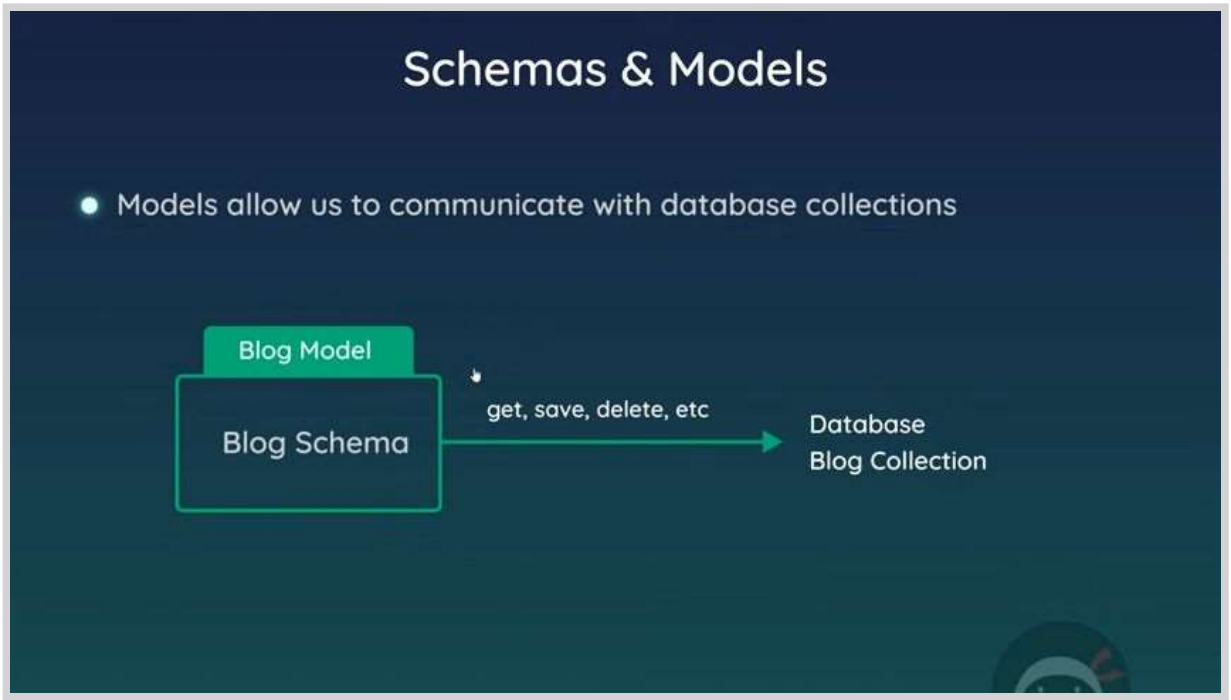

06:58




07:59



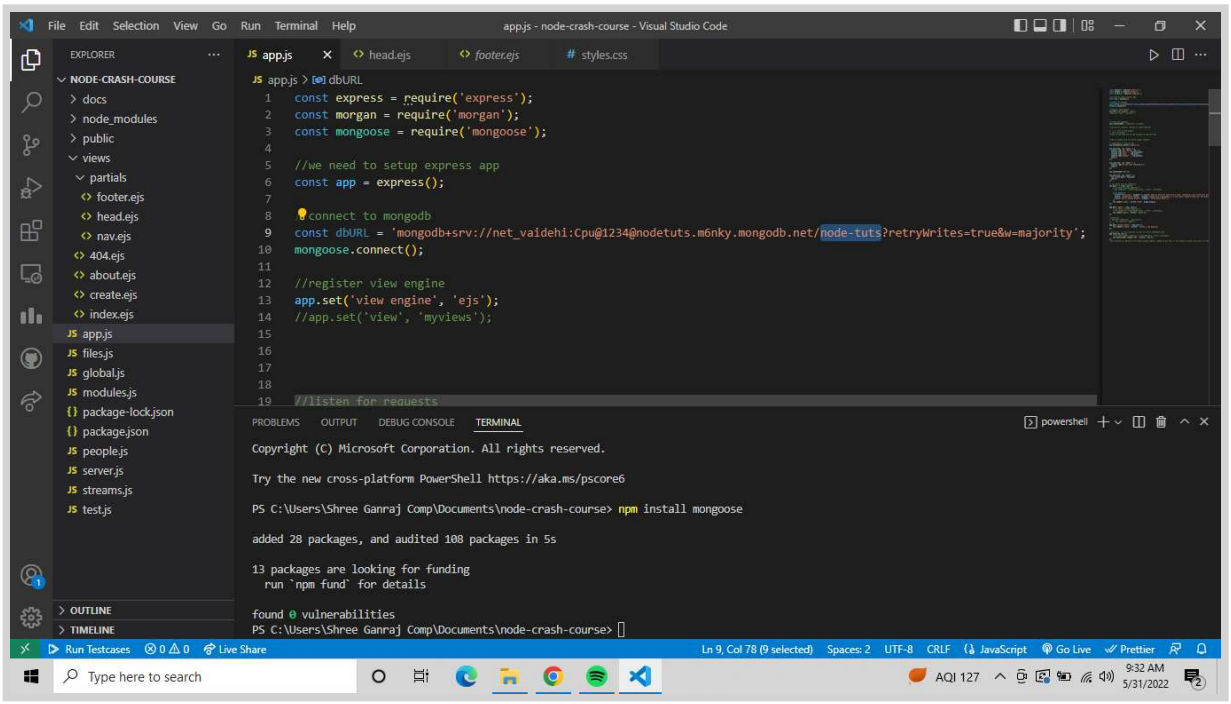

09:06



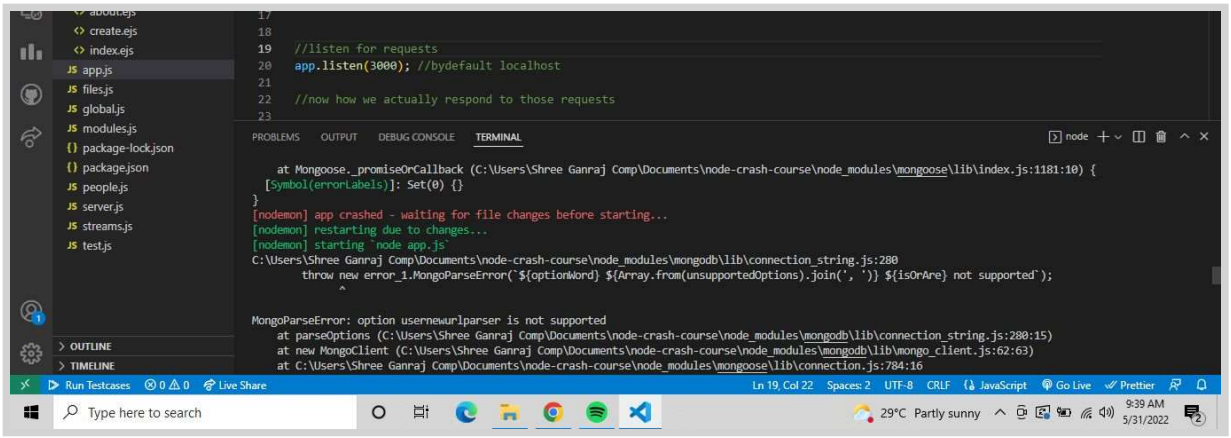

**so we re using moongose to connect to the database:**

**moongose is a third party package**

**now we can use mongoose object to connect to the database...**



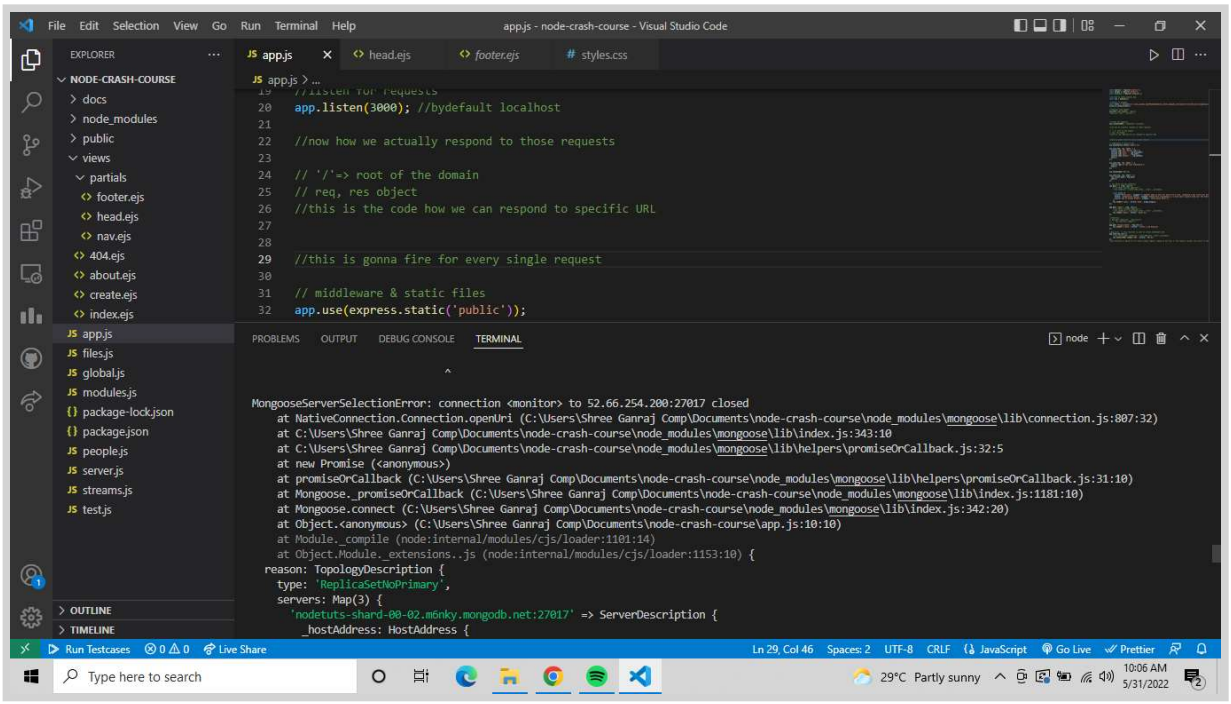

https://stackoverflow.com/questions/68958221/mongoparseerror-options-usecreateindex-usefindandmodify-are-not-supported/68962378#68962378




solution:

**https://www.mongodb.com/docs/atlas/troubleshoot-connection/#special-characters-in-connection-string-password**

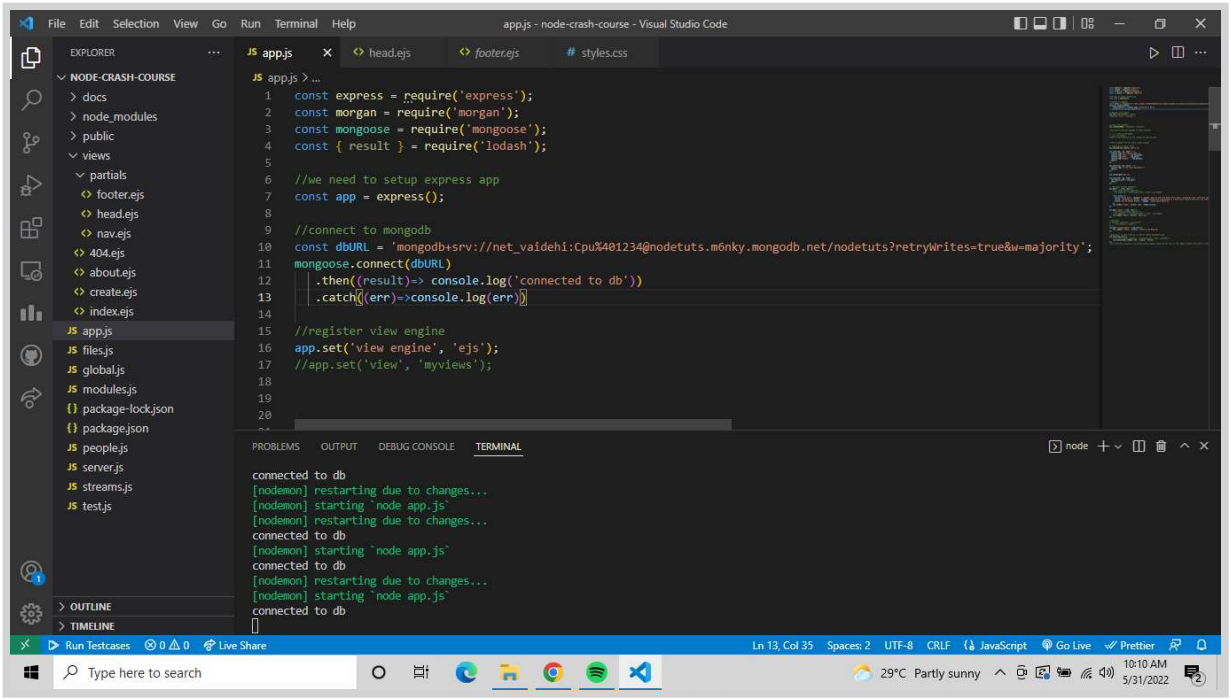
**error:**



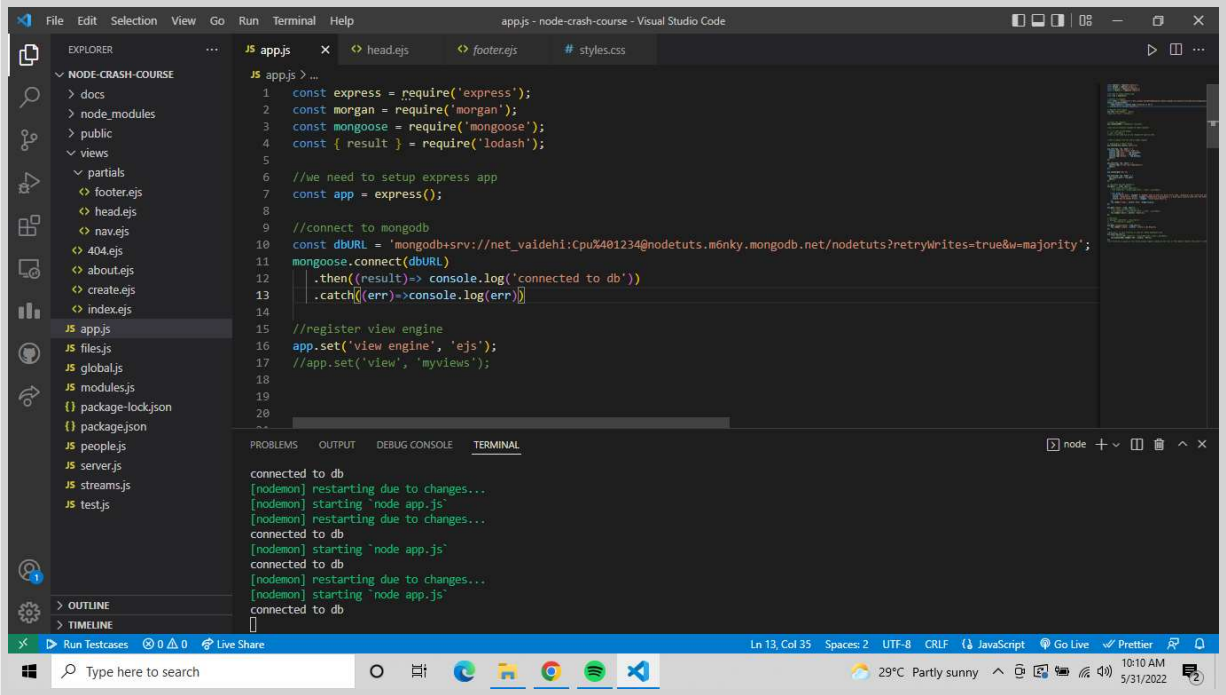

**solution:**

https://stackoverflow.com/questions/60431996/mongooseerror-mongooseserverselectionerror-connection-monitor-to-52-6-250-2

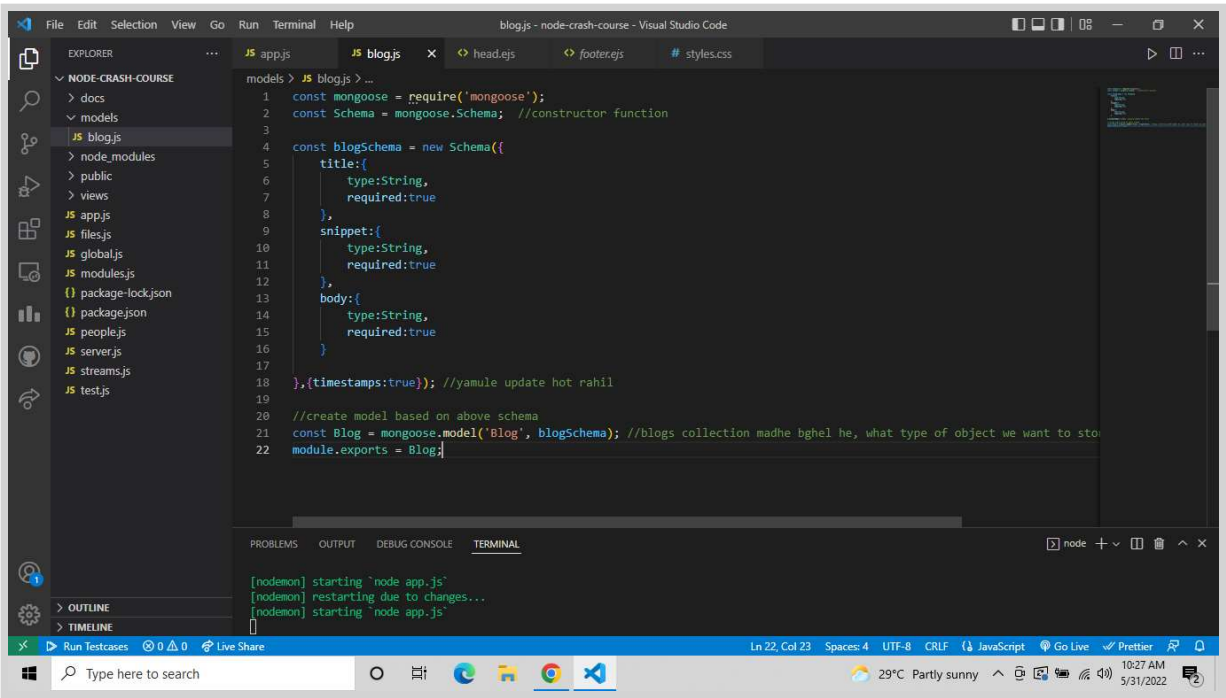
**//when connection is complete it returns promises:**




**so after this connection is completed, only then we should listen to requests..**



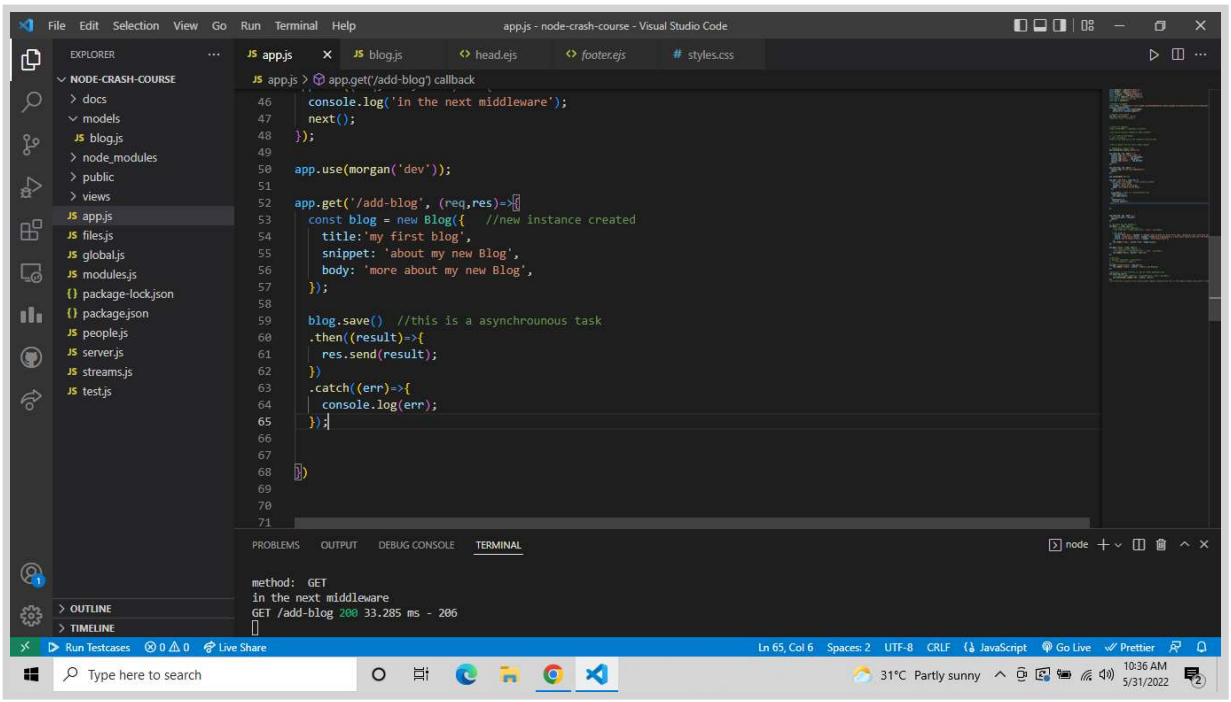

## //now create a schemas for our blog modules



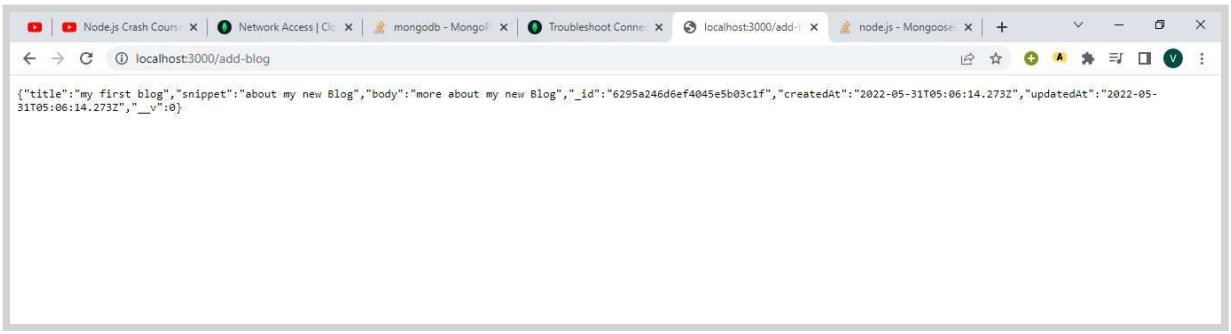

[19:28]

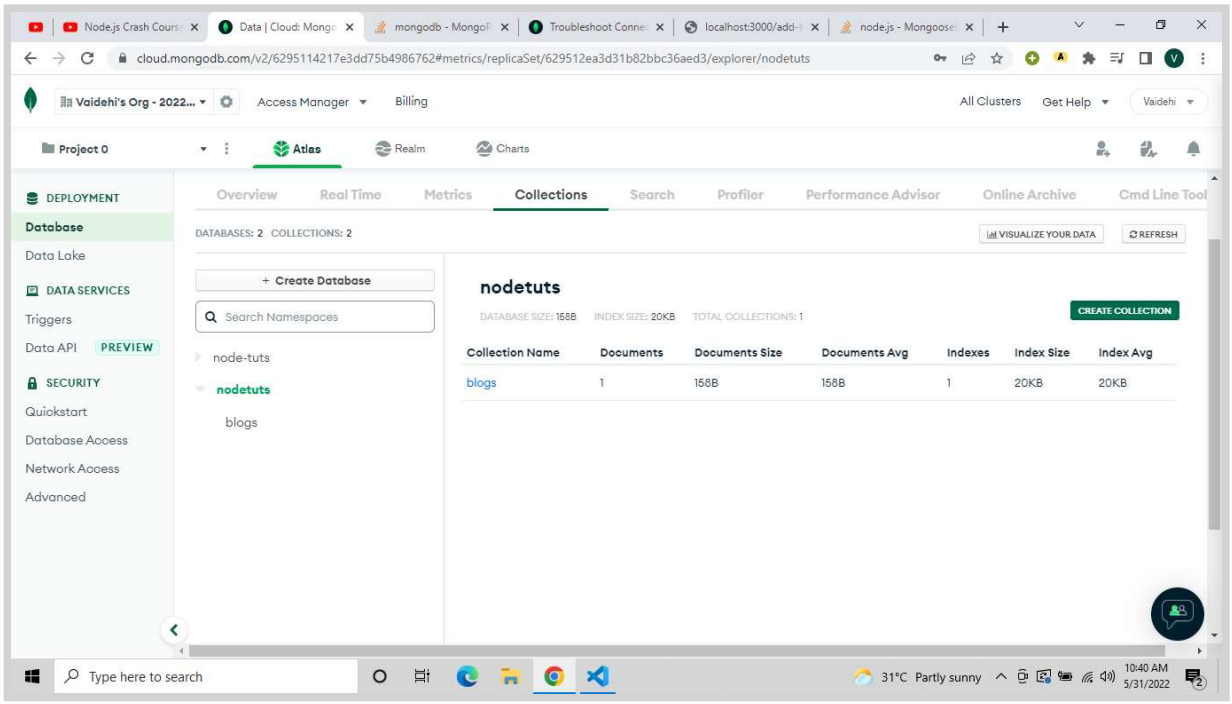

## //mongoose and mongo sandbox routes:

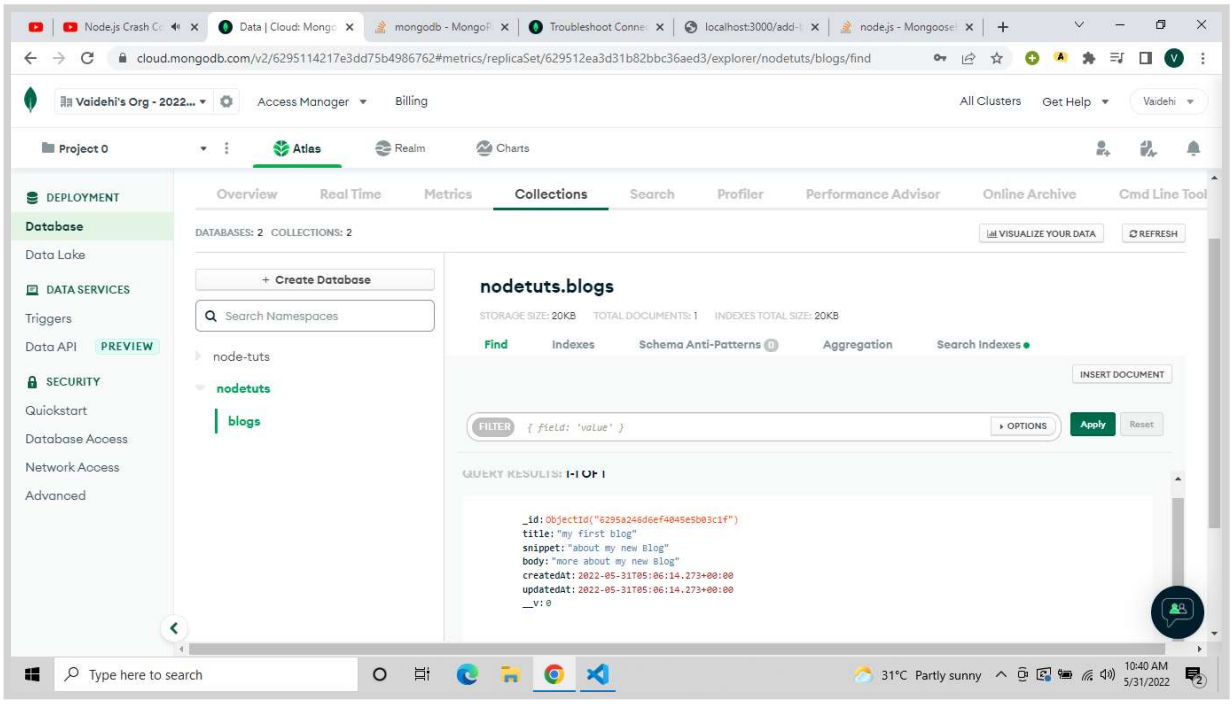
## just out to test the interaction with database

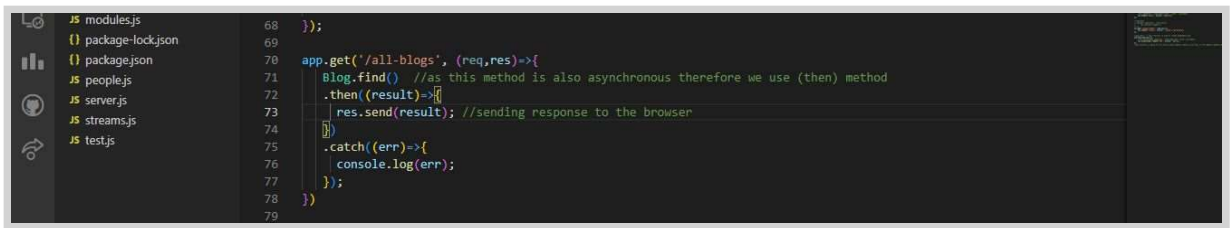

# //This is the response we get , it looks like javascript object...

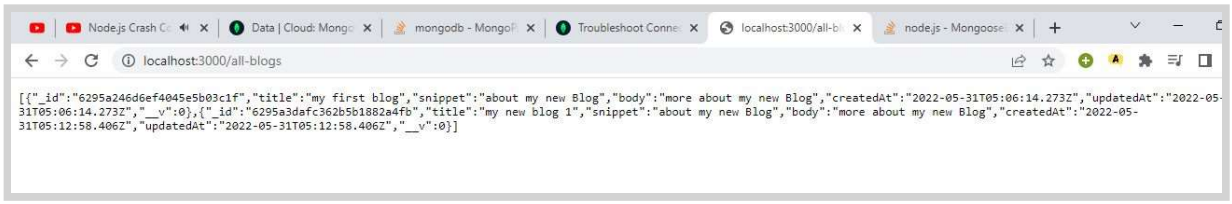




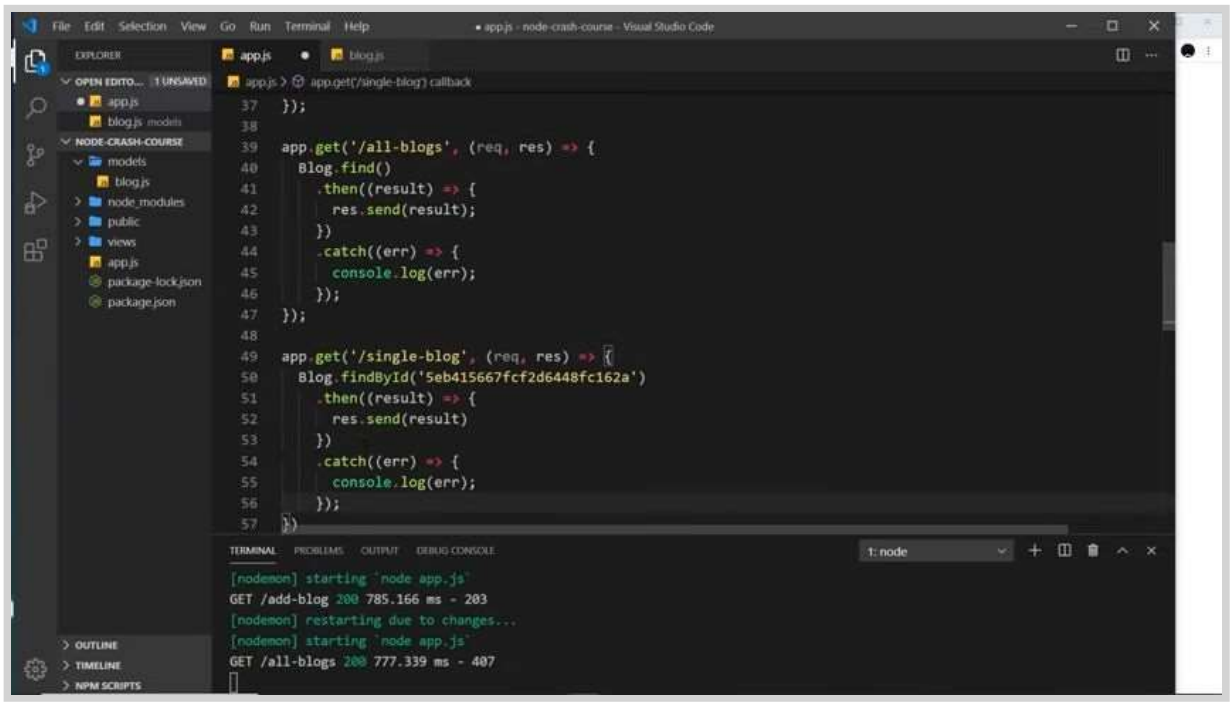




# //if we want to retrieve all the blogs from collection:





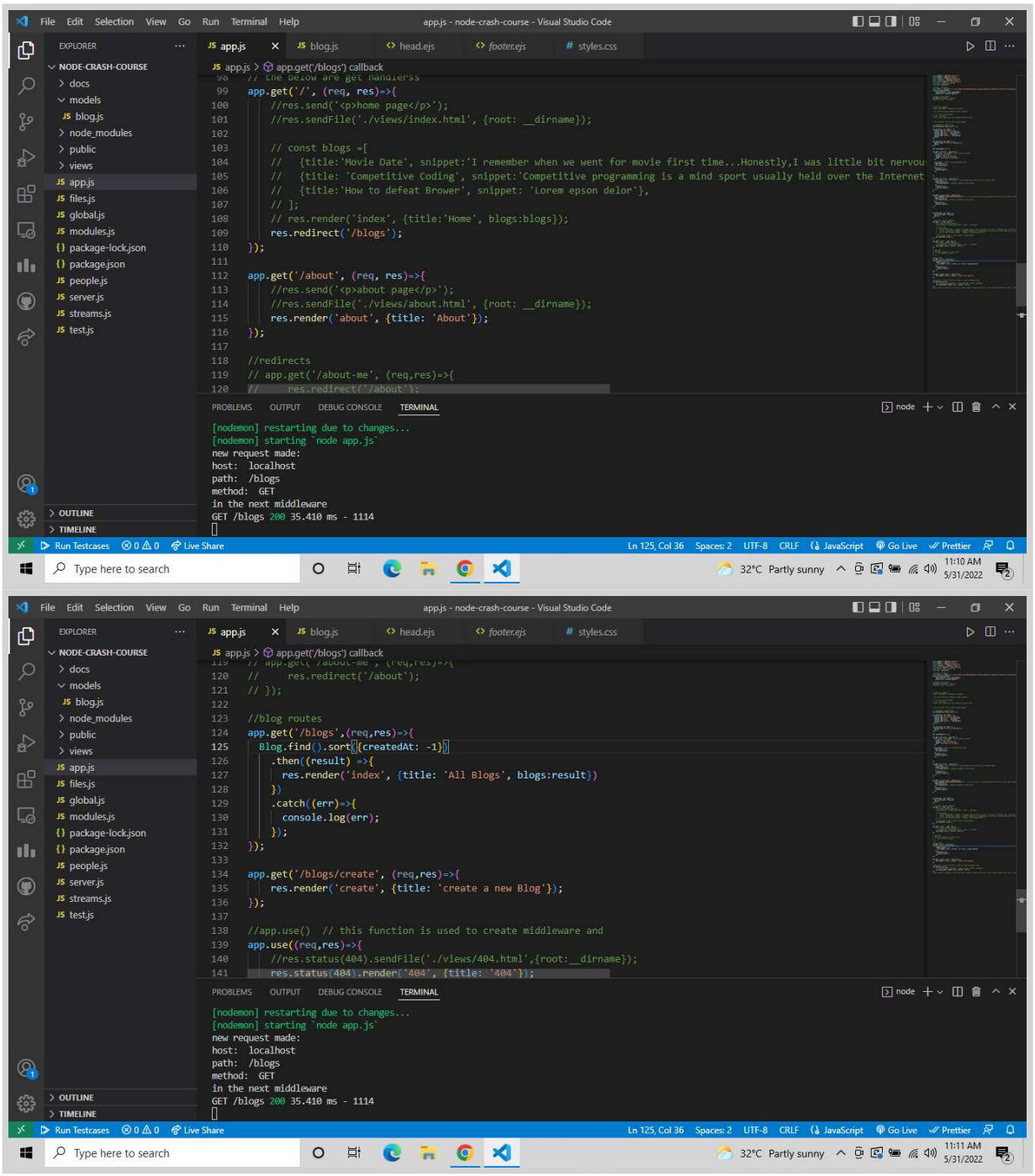

[29:38](#)




## //I don't want this kind of routes...

## i want to integrate this routes with my main routess...so

[30:45](#)




## video10:

## //ataparynt aapn GET request ch banvli aahe..

**In this node.js tutorial we'll set up some route handlers for different request types. We'll also look at how to handle post requests and delete requests.**

00:32


//

01:42


## we can use same routes for different types of request

## //route item:

02:44


02:57

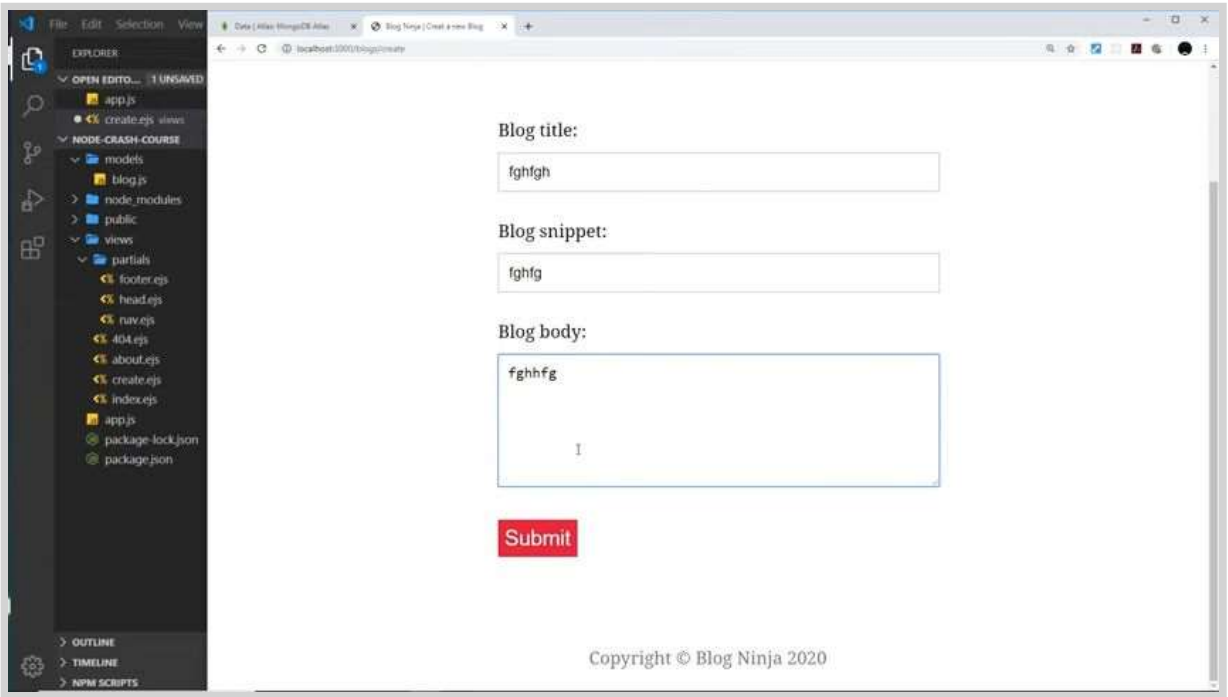

## //we not gonna do PUT(update blogs):

03:51


## //POST Requests:

## after clicking on new blog we need to fire post request after submitting this form

04:20




## //Handling the post request, take all of that data add that data to instance and save data to database //

04:04

new block he'd add in a load of junk and then you click on submit to add this new exciting blog to the websites now at this point we need to fire a post request to the server with all of this data included in the post request then

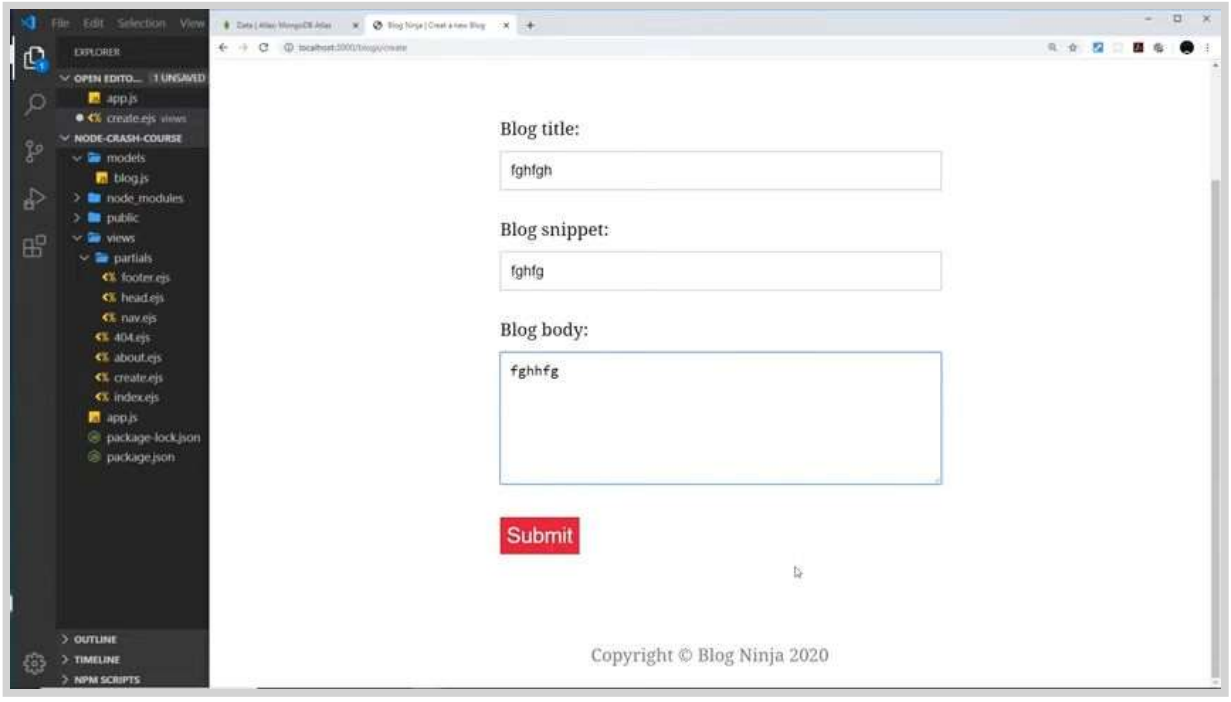
04:16

data included in the post request then on the server we can handle that post request we can take all of that data create a new instance of a blog document using the blog model we created add this data to that instance and then save that

04:22

request we can take all of that data create a new instance of a blog document using the blog model we created add this data to that instance and then save that document to the database so that's the sequence of events that needs to occur

04:47



04:34

**sequence of events that needs to occur so the first thing we need to do is send a post request from this webform when we click on submit now there's a couple of ways we can do that we can do that using the fetch API or some other asynchronous web app**

**action=> where we want to send this request to**

05:22


**//now when we click on submit ,...this is (line no 8)gonna send u the post request to this url (action="/blogs")**

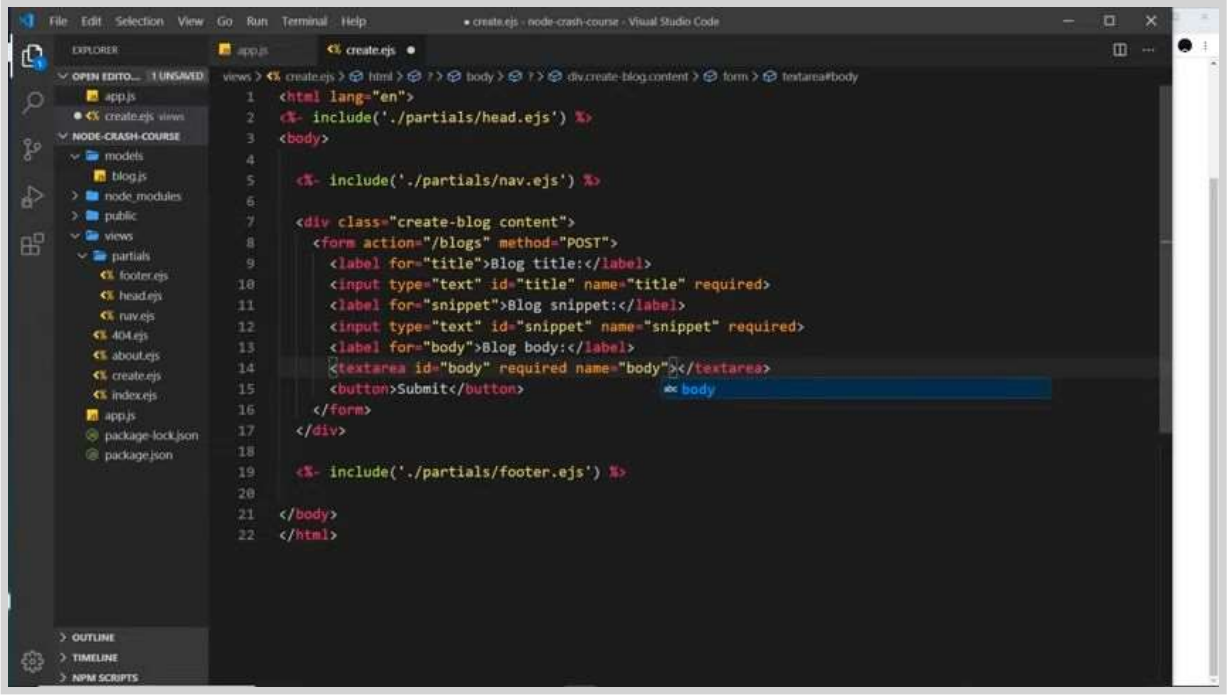
**and now we need to handle that on server**

05:37


**Label Pasna Textarea parynt ....send karaychay post request sobt...**

**tr name attribute add karava lagel**

06:08



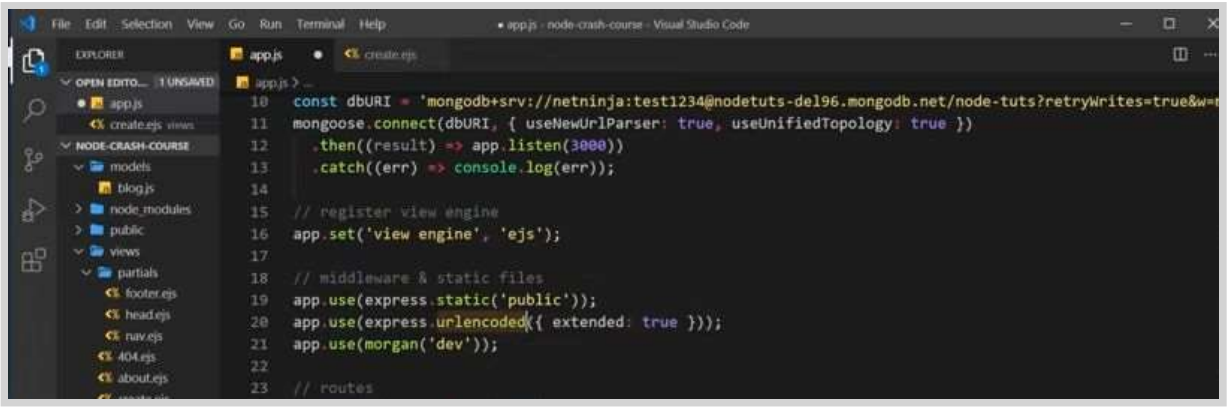

## //handling post request

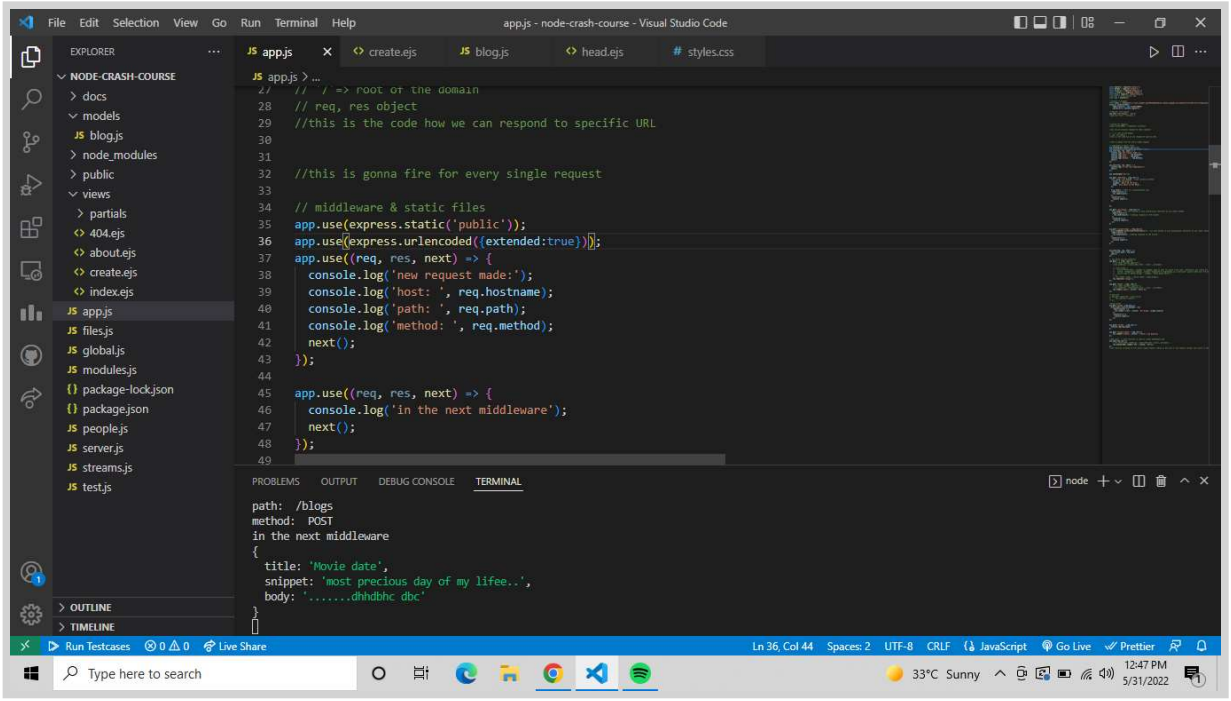
## app.post('/blogs', (req,res)=>{

## })

## adding middleware...

[07:35](#)

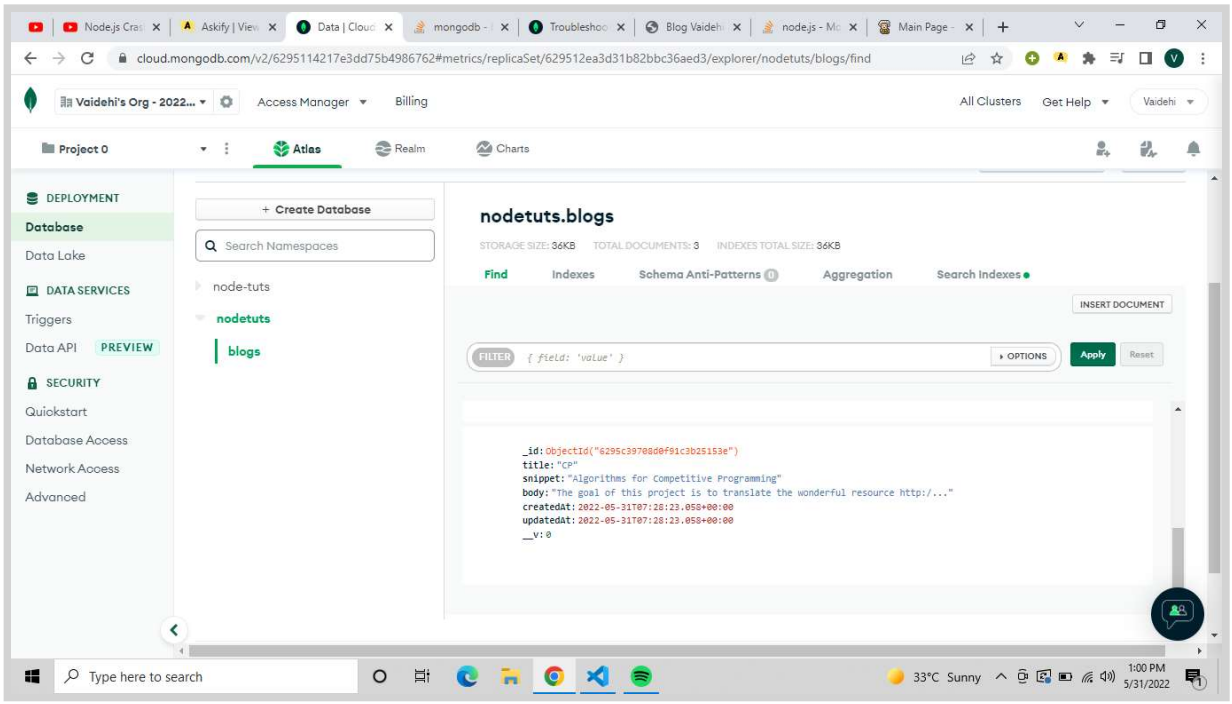





## //firstly create new instance of blog and then save it to database:

## //jya jya methods asynchronous aahet , tya promise return krtat...

[10:57]



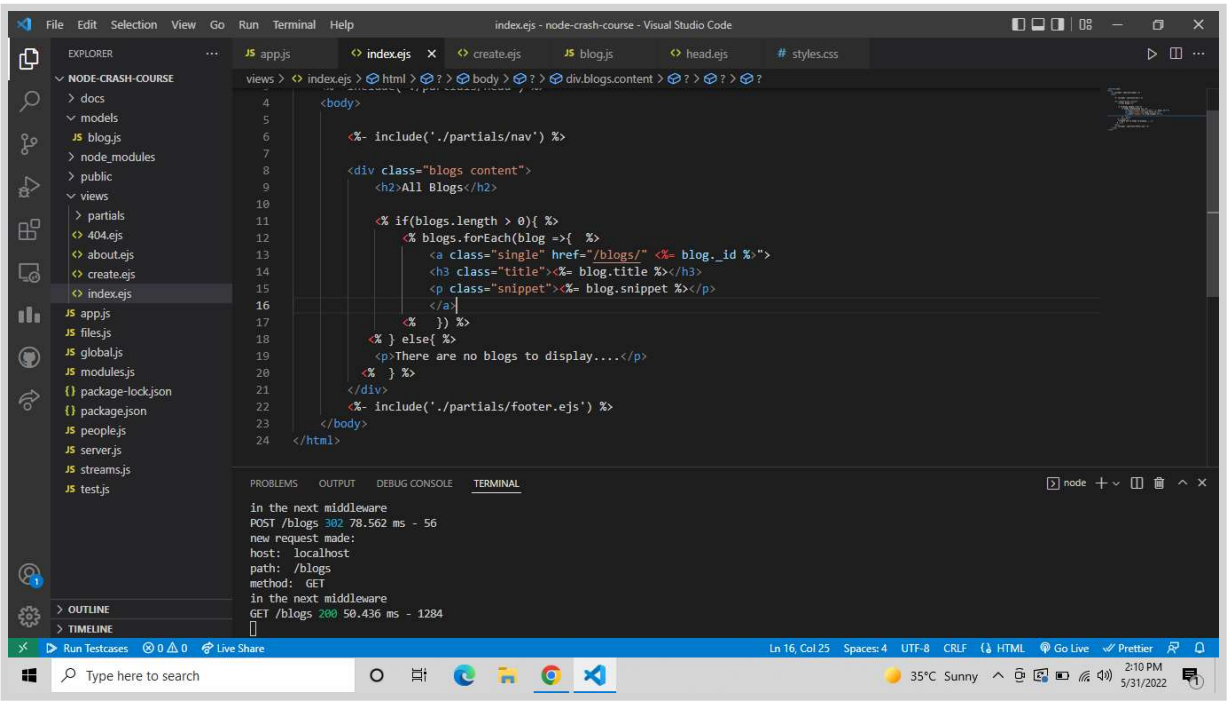

## //Route parameters we need to learn before, get request for showing a single blog and delete request

## id is variable

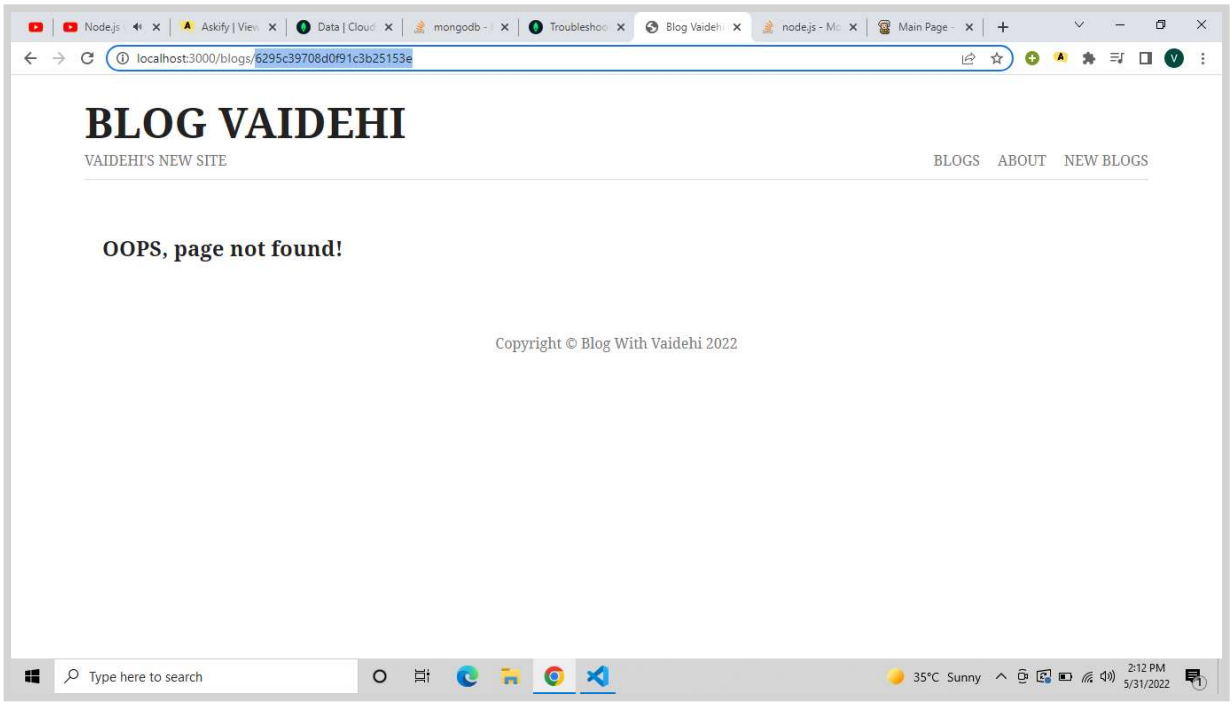
[12:08]


[12:24]

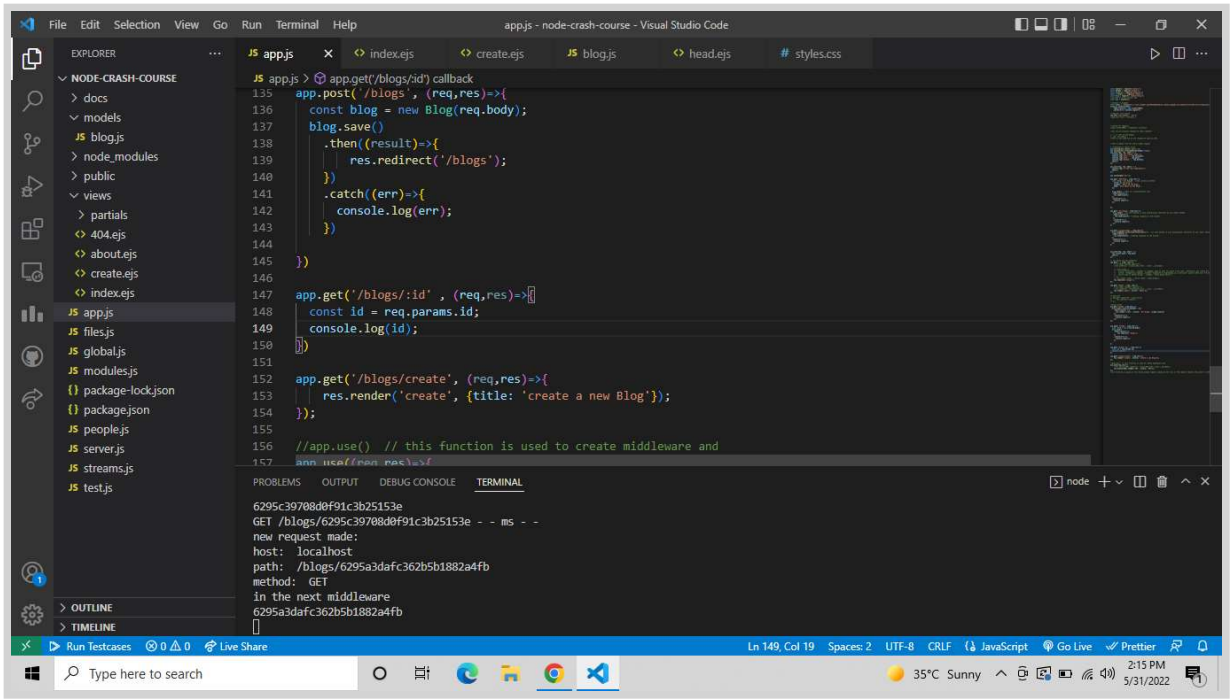

14:12




**this is the get request and we need to extract this see the image below:**

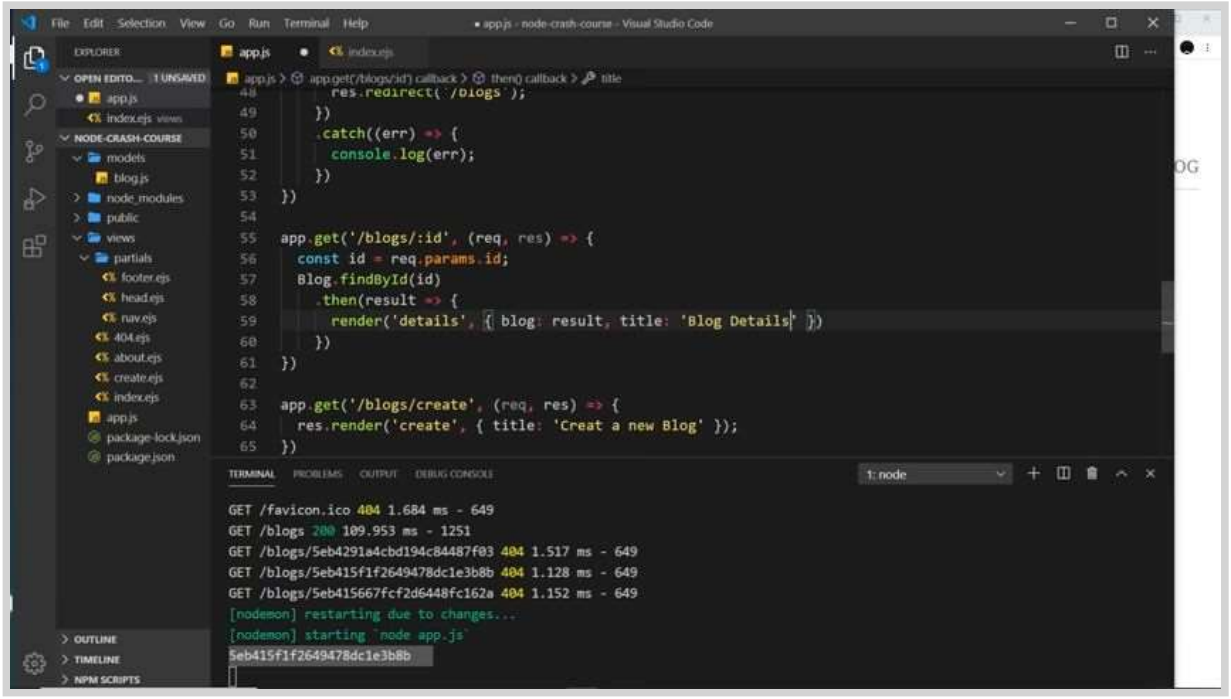



## //but browser will got hung

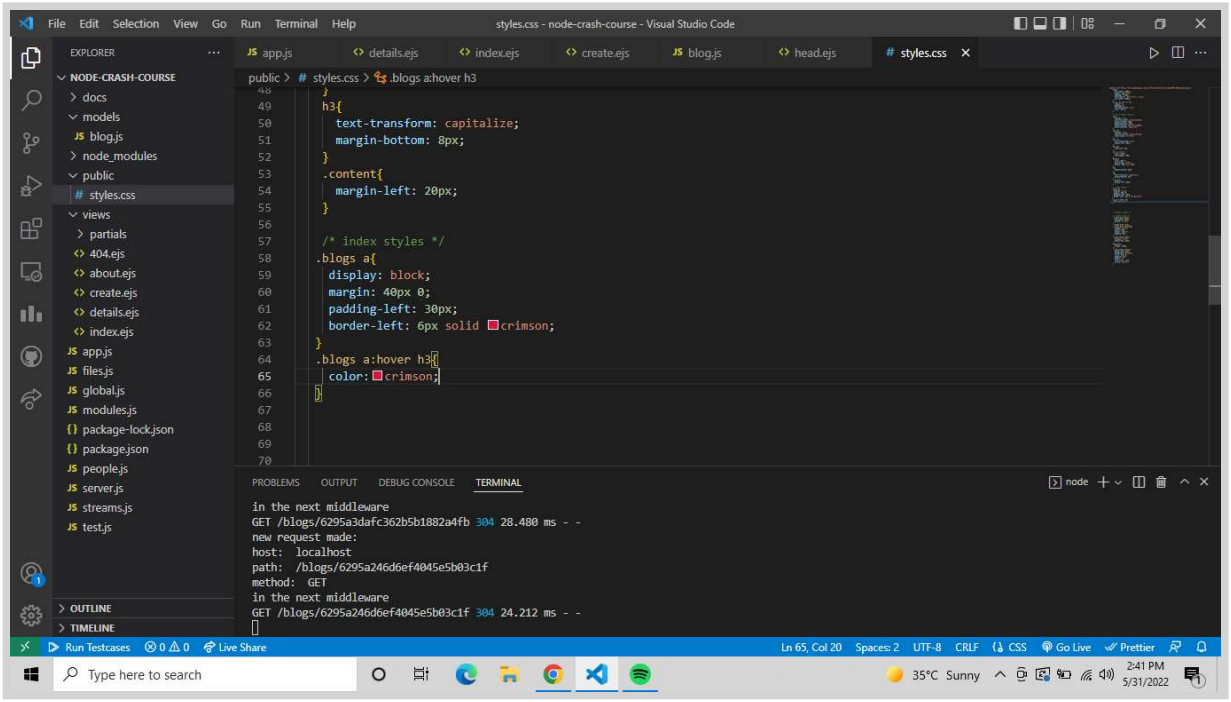


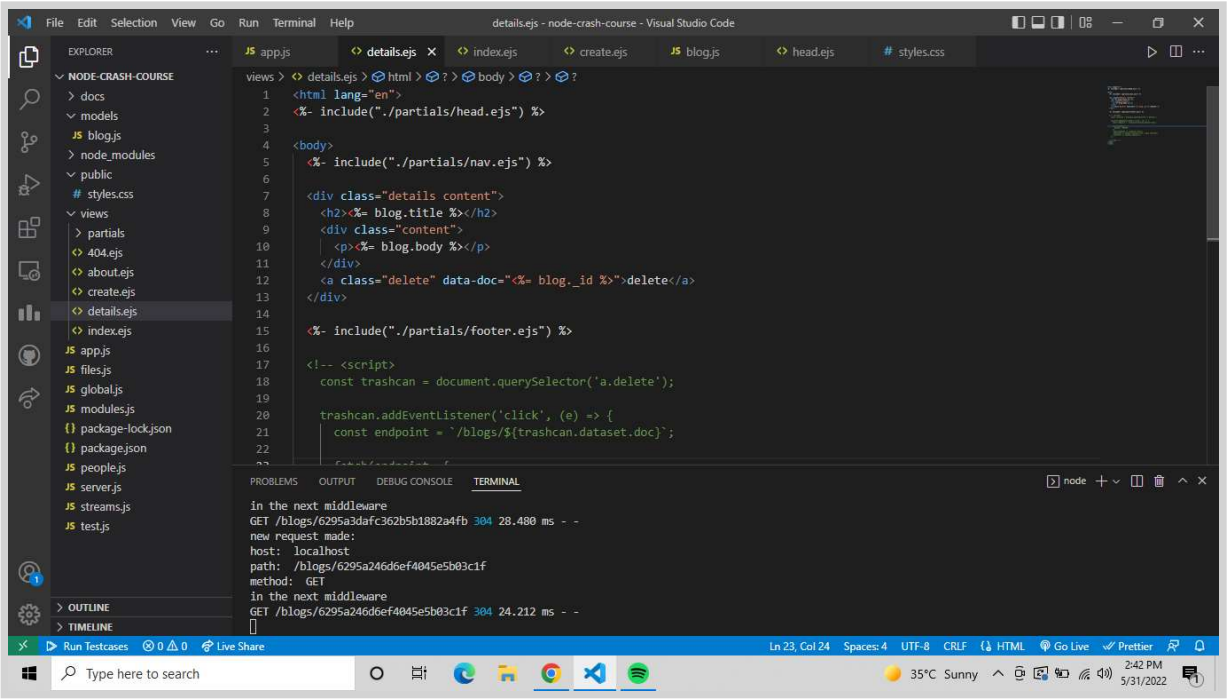

## //models madhe

[17:15]


app.get('/blogs/:id', (req, res) => {
const id = req.params.id;
Blog.findById(id)
.then(result => {
render('details', { blog: result, title: 'Blog Details' })
})
})
app.get('/blogs/create', (req, res) => {
res.render('create', { title: 'Creat a new Blog' });
})
GET /favicon.ico 404 1.684 ms - 649
GET /blogs 200 109.953 ms - 1251
GET /blogs/5eb4291a4cbd194c84487f03 404 1.517 ms - 649
GET /blogs/5eb415f1f2649478dc1e3b8b 404 1.128 ms - 649
GET /blogs/5eb415667fcf2d6448fc162a 404 1.152 ms - 649
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
5eb415f1f2649478dc1e3b8b

# //nantr views


.blogs a{
display: block;
margin: 40px 0;
padding-left: 30px;
border-left: 6px solid crimson;
}
.blogs a:hover h3{
color: crimson;
}
in the next middleware
GET /blogs/6295a3dafc362b5b1882a4fb 304 28.480 ms - -
new request made:
host: localhost
path: /blogs/6295a246d6ef4045e5b03c1f
method: GET
in the next middleware
GET /blogs/6295a246d6ef4045e5b03c1f 304 24.212 ms - -

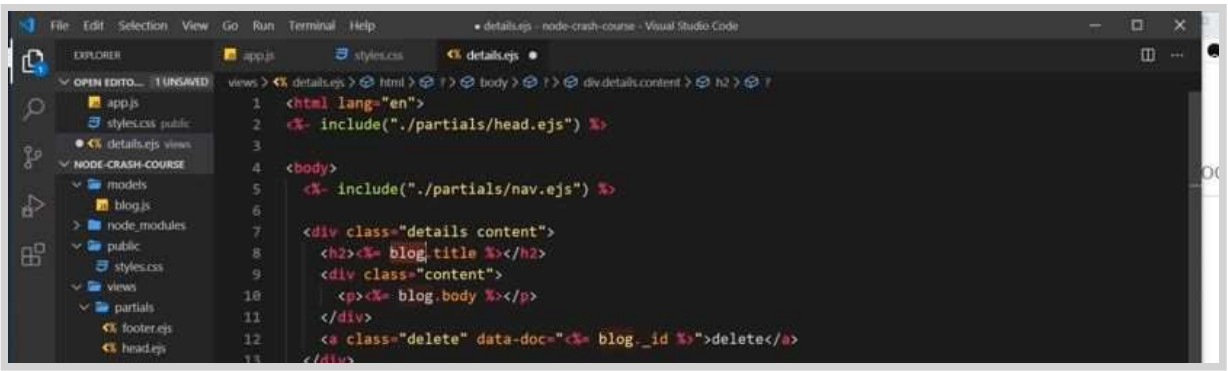



# #Delete Requests:

# to delete a specific blog from frontend

# //u have to send a delete request

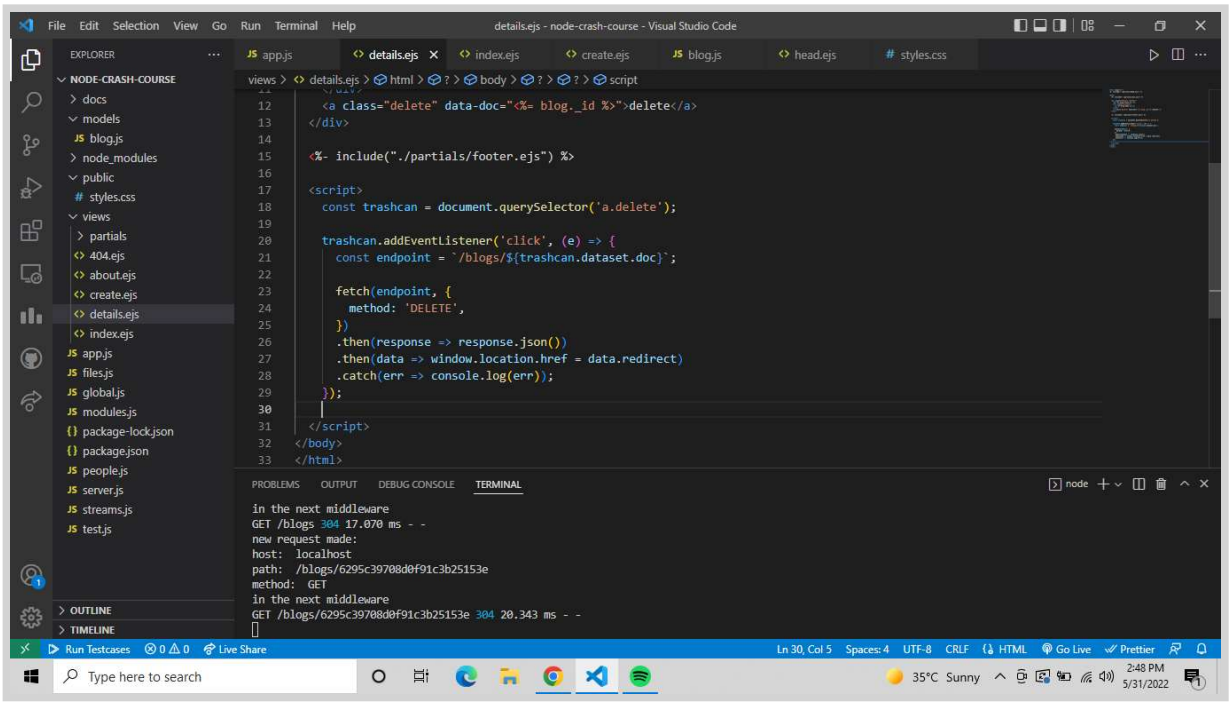
[21:36]




# script tag madhe je pn lihito te browser wr run hota...server wr nahi..

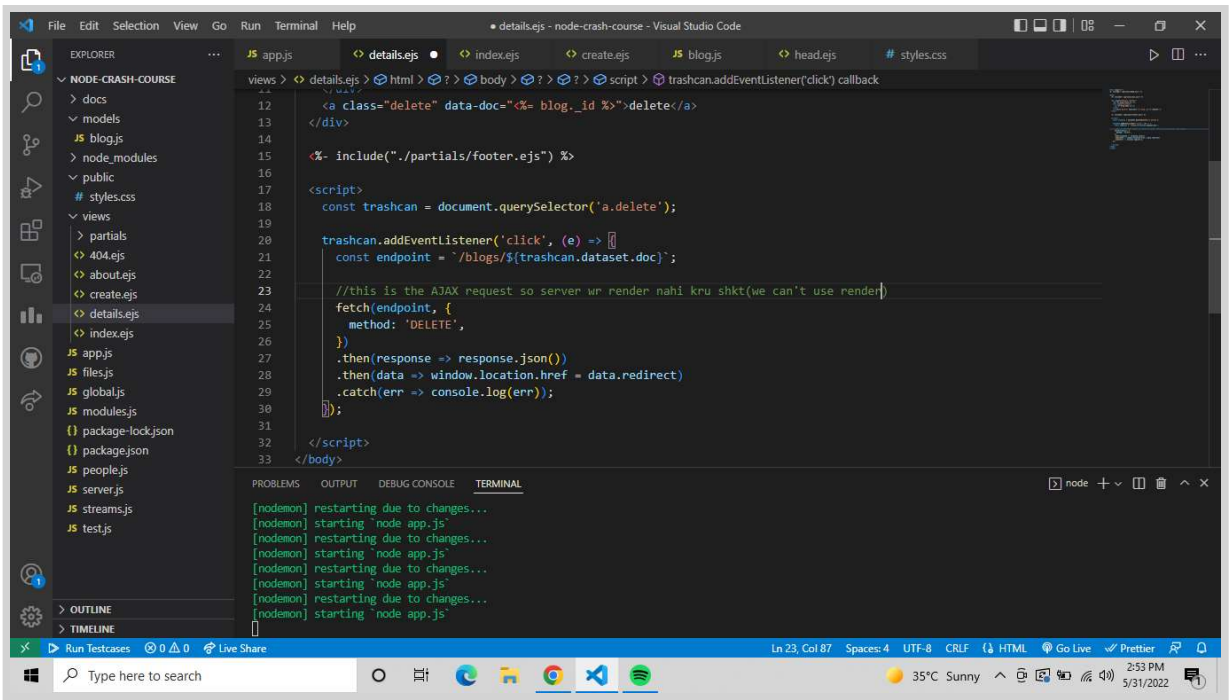
# querySelector =>to grab that event...

# we send delete request to the Endpoint..

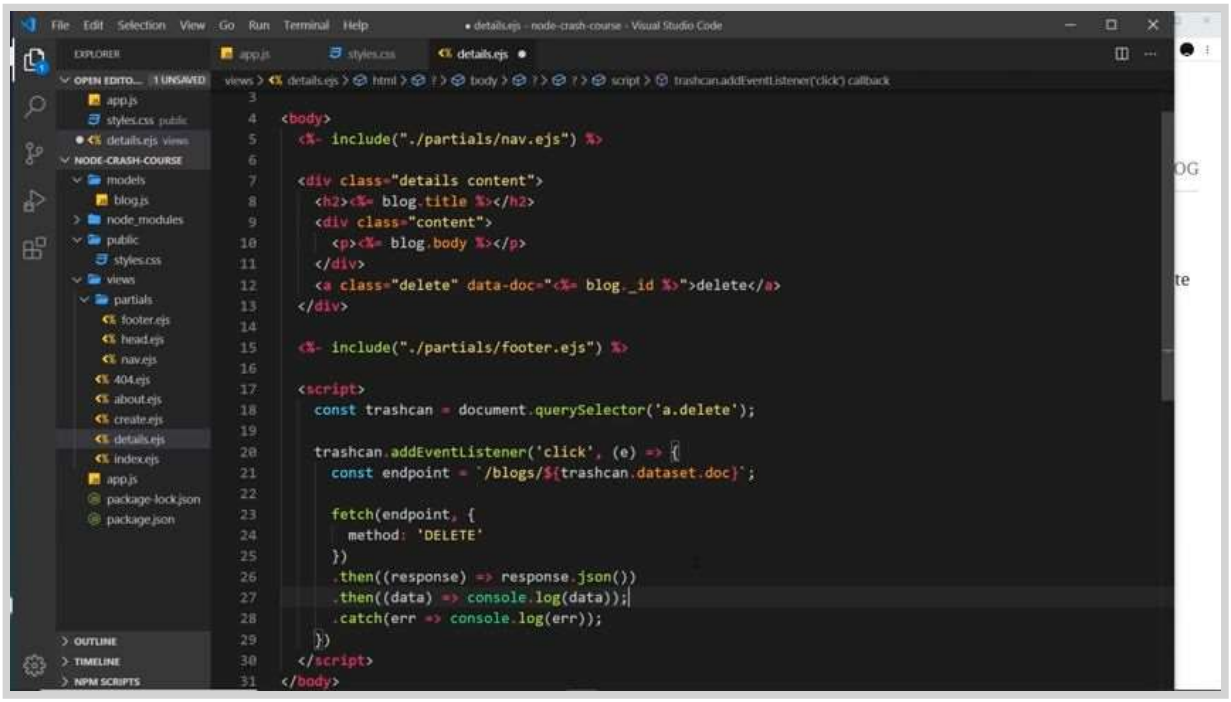



29:00

**is gonna be a URL to where we want to redirect to and then that's going to be done from the front end because we can't do it on the server because this is an ajax request a hope that makes sense it might seem long-winded and around the houses and but this is the way we're going to**



31:23




32:06


# Video11:

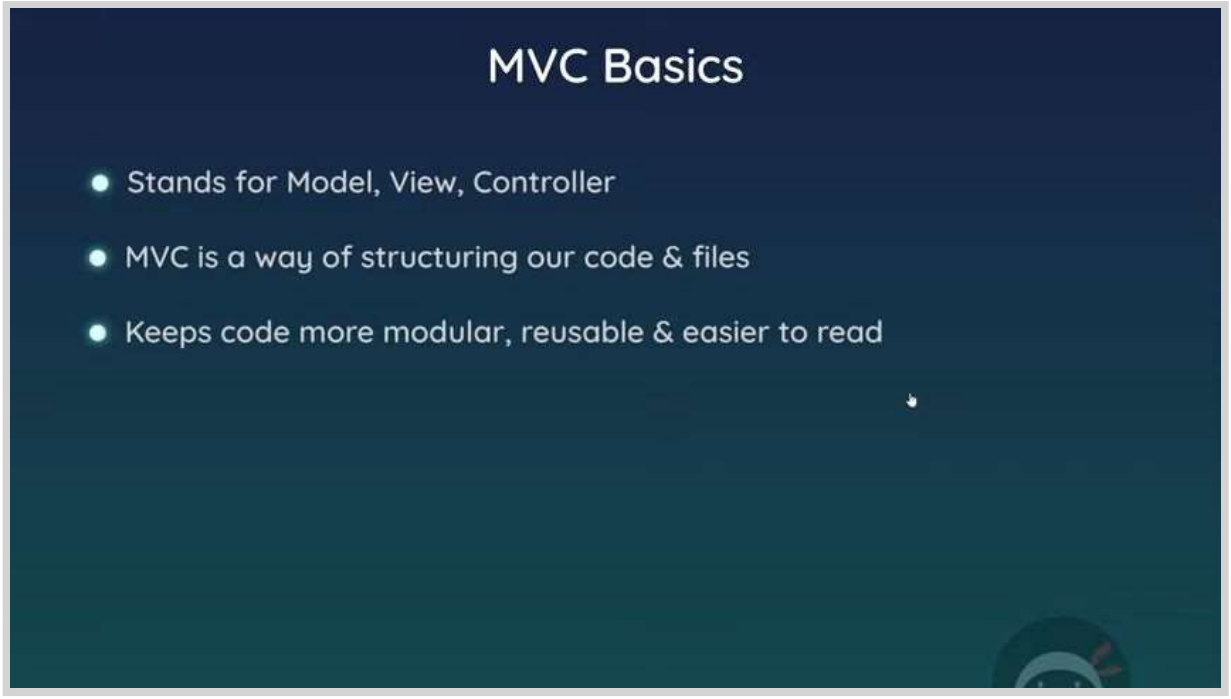
# Express Router & MVC:

=>

## split codes in different own route filess...

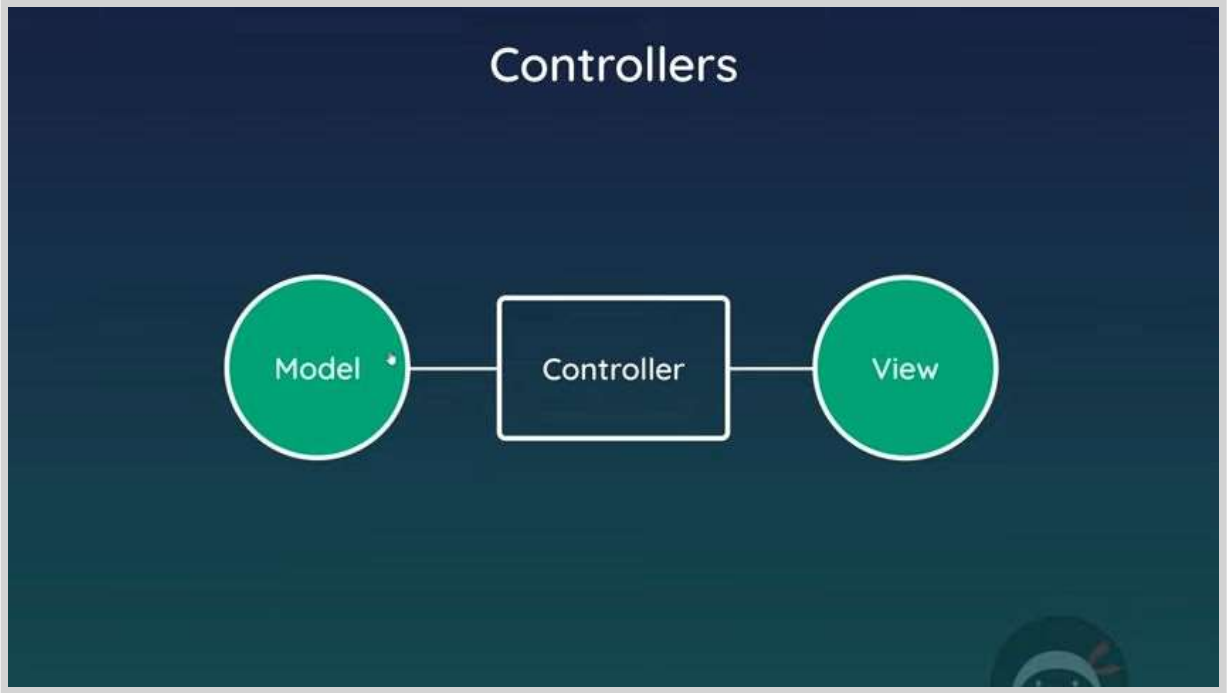
## //MVC Basics:

09:59




## links between model and view

10:42




11:14

**job to do so our route file matches incoming requests and it passes those requests to the correct controller function a controller communicates with the appropriate model to get data into a view and then the view then renders that**

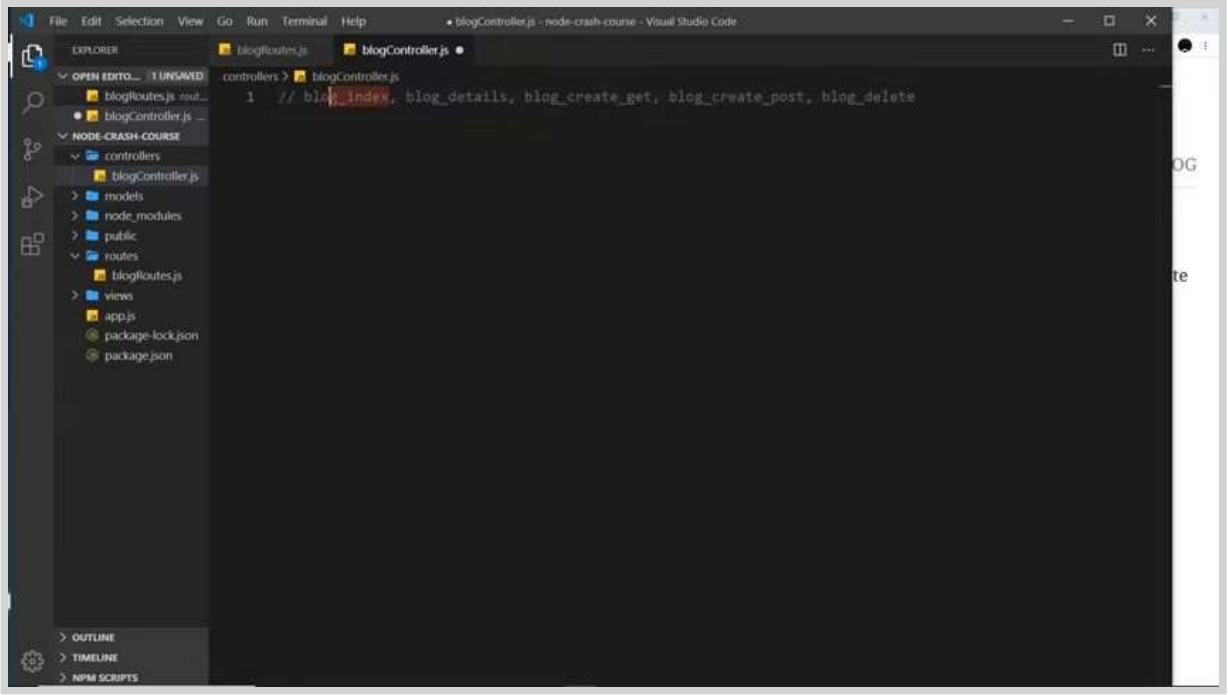
11:22

**function a controller communicates with the appropriate model to get data into a view and then the view then renders that data into its template and it gets sent to the browser so all we're basically going to do is extract our route handler**

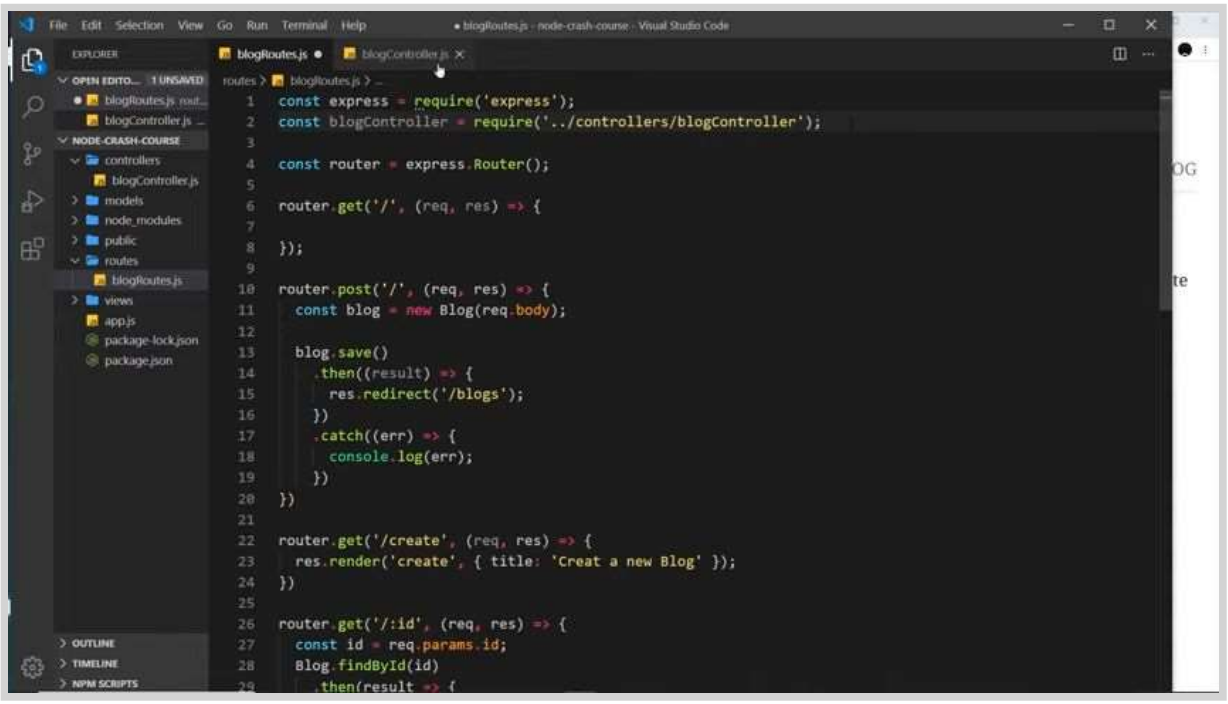
11:30

**data into its template and it gets sent to the browser so all we're basically going to do is extract our route handler functions into a separate controller file to make everything a little easier to read and understand**

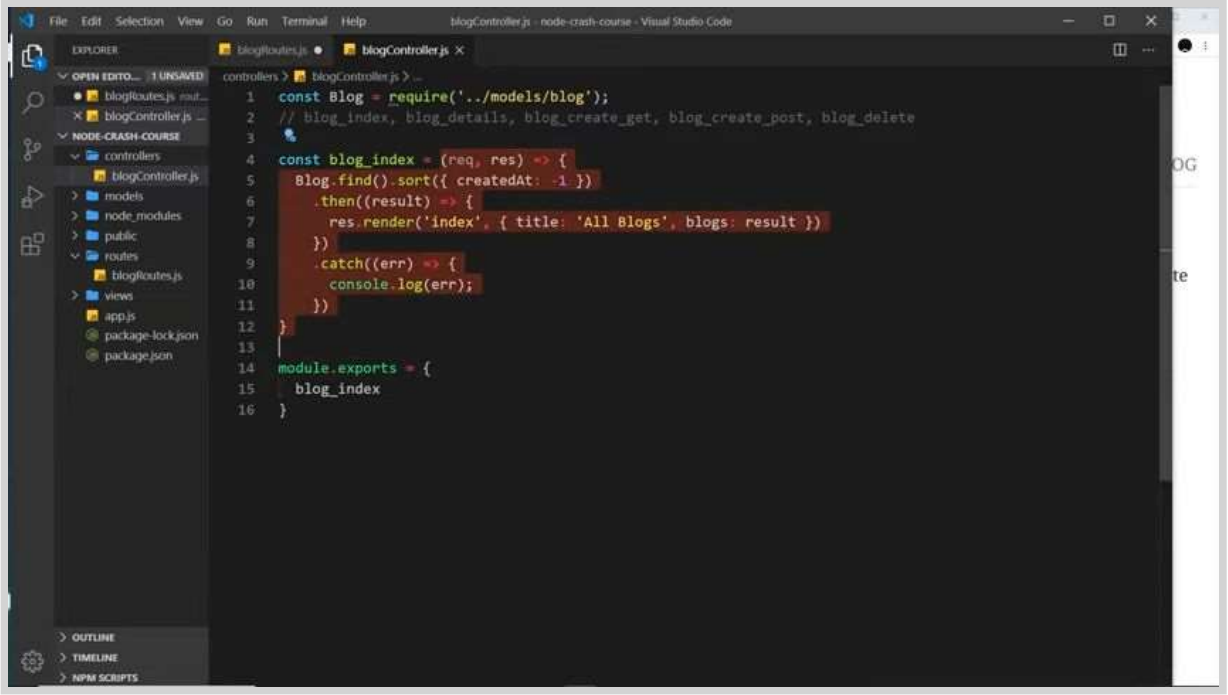
12:45


File Edit Selection View Go Run Terminal Help
blogController.js - node-crash-course - Visual Studio Code
EXPLORER
blogRoutes.js
blogController.js
controllers > blogController.js
1 // blog_index, blog_details, blog_create_get, blog_create_post, blog_delete
OPEN EDITO... 1 UNSAVED
NODE-CRASH-COURSE
controllers
models
node_modules
public
routes
views
app.js
package-lock.json
package.json
OUTLINE
TIMELINE
NPM SCRIPTS

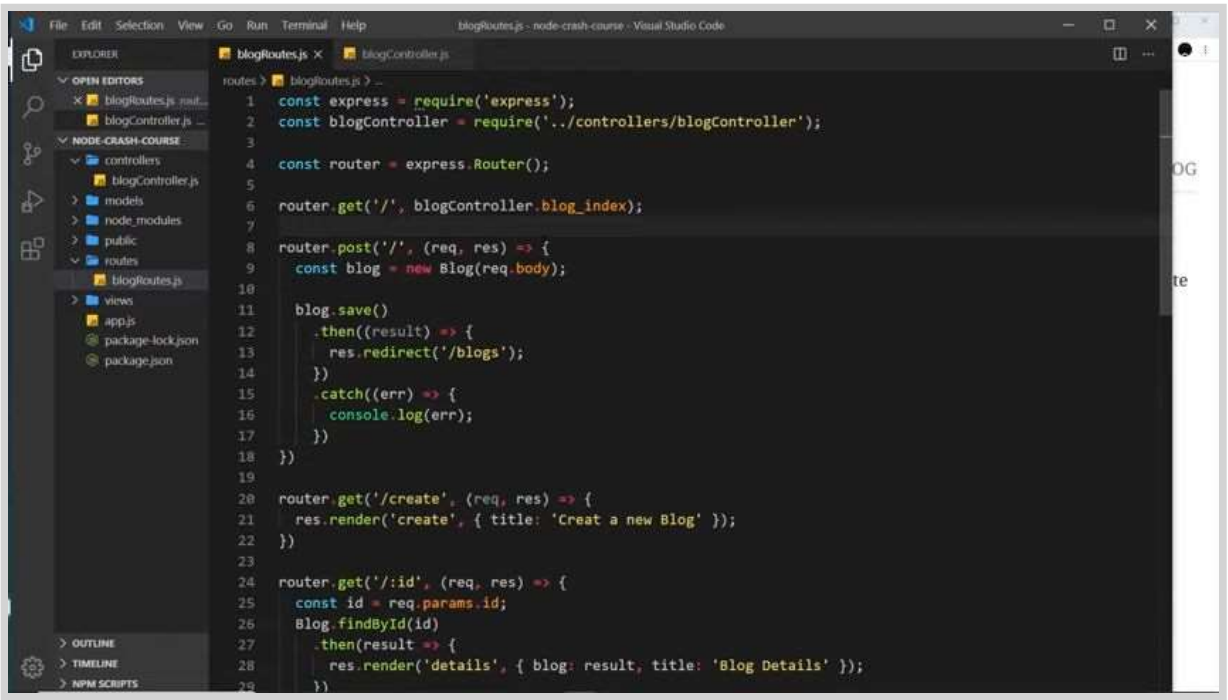
[14:55]


File Edit Selection View Go Run Terminal Help
blogRoutes.js - node-crash-course - Visual Studio Code
EXPLORER
blogRoutes.js
blogController.js
routes > blogRoutes.js
const express = require('express');
const blogController = require('../controllers/blogController');
const router = express.Router();
router.get('/', (req, res) => {
});
router.post('/', (req, res) => {
const blog = new Blog(req.body);
blog.save()
.then((result) => {
res.redirect('/blogs');
})
.catch((err) => {
console.log(err);
})
})
router.get('/create', (req, res) => {
res.render('create', { title: 'Creat a new Blog' });
})
router.get('/:id', (req, res) => {
const id = req.params.id;
Blog.findById(id)
.then(result => {
OUTLINE
TIMELINE
NPM SCRIPTS

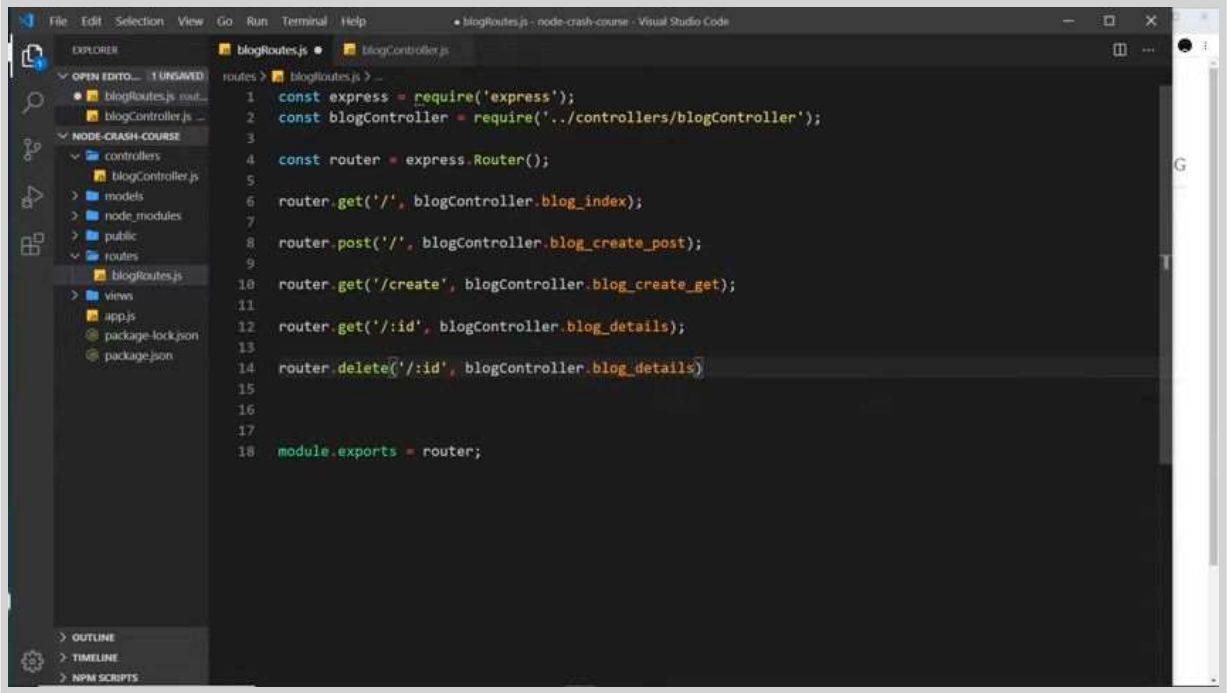
[15:13]


File Edit Selection View Go Run Terminal Help
blogController.js - node-crash-course - Visual Studio Code
EXPLORER
blogRoutes.js
blogController.js
NODE-CRASH-COURSE
controllers
blogController.js
models
node_modules
public
routes
blogRoutes.js
views
app.js
package-lock.json
package.json
OUTLINE
TIMELINE
NPM SCRIPTS
const Blog = require('../models/blog');
// blog_index, blog_details, blog_create_get, blog_create_post, blog_delete
const blog_index = (req, res) => {
Blog.find().sort({ createdAt: -1 })
.then((result) => {
res.render('index', { title: 'All Blogs', blogs: result })
})
.catch((err) => {
console.log(err);
})
}
module.exports = {
blog_index
}

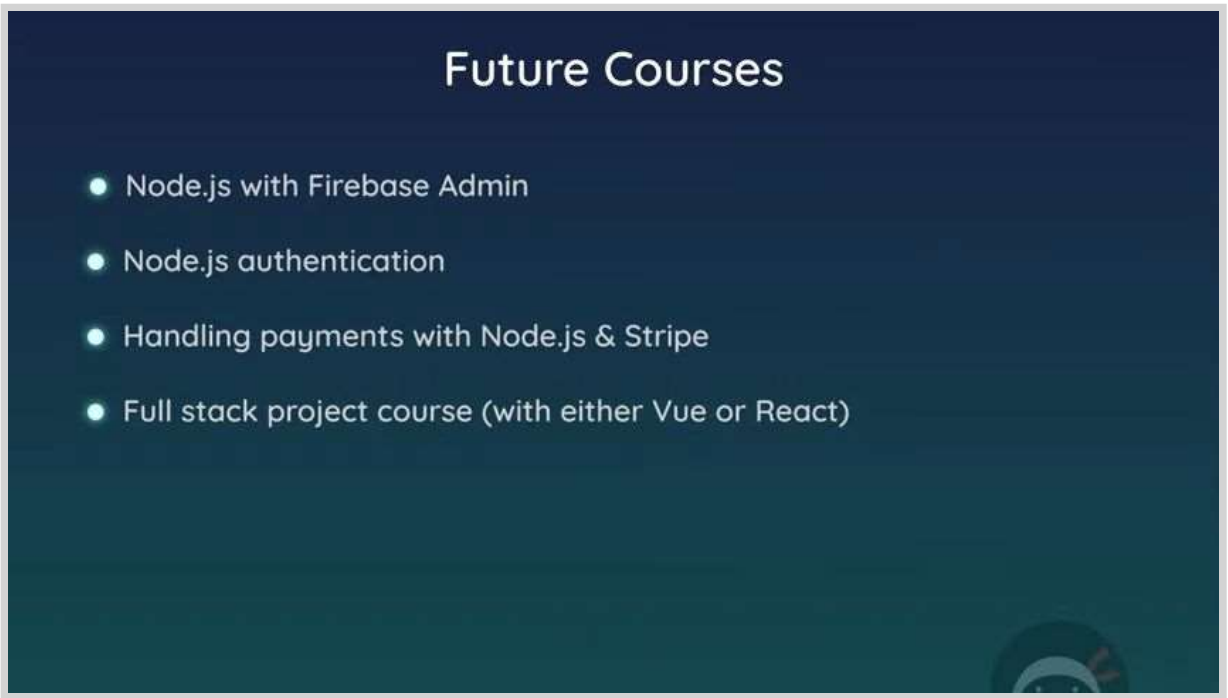
## 15:25


File Edit Selection View Go Run Terminal Help
blogRoutes.js - node-crash-course - Visual Studio Code
EXPLORER
blogRoutes.js
blogController.js
OPEN EDITORS
NODE-CRASH-COURSE
controllers
blogController.js
models
node_modules
public
routes
blogRoutes.js
views
app.js
package-lock.json
package.json
OUTLINE
TIMELINE
NPM SCRIPTS
const express = require('express');
const blogController = require('../controllers/blogController');
const router = express.Router();
router.get('/', blogController.blog_index);
router.post('/', (req, res) => {
const blog = new Blog(req.body);
blog.save()
.then((result) => {
res.redirect('/blogs');
})
.catch((err) => {
console.log(err);
})
})
router.get('/create', (req, res) => {
res.render('create', { title: 'Creat a new Blog' });
})
router.get('/:id', (req, res) => {
const id = req.params.id;
Blog.findById(id)
.then(result => {
res.render('details', { blog: result, title: 'Blog Details' });

## 18:31


File Edit Selection View Go Run Terminal Help
blogRoutes.js - node-crash-course - Visual Studio Code
EXPLORER
blogRoutes.js
blogController.js
NODE-CRASH-COURSE
controllers
blogController.js
models
node_modules
public
routes
blogRoutes.js
views
app.js
package-lock.json
package.json
OUTLINE
TIMELINE
NPM SCRIPTS
const express = require('express');
const blogController = require('../controllers/blogController');
const router = express.Router();
router.get('/', blogController.blog_index);
router.post('/', blogController.blog_create_post);
router.get('/create', blogController.blog_create_get);
router.get('/:id', blogController.blog_details);
router.delete('/:id', blogController.blog_details)
module.exports = router;

[03:03]


Future Courses
Node.js with Firebase Admin
Node.js authentication
Handling payments with Node.js & Stripe
Full stack project course (with either Vue or React)

‘

‘

‘

‘

‘