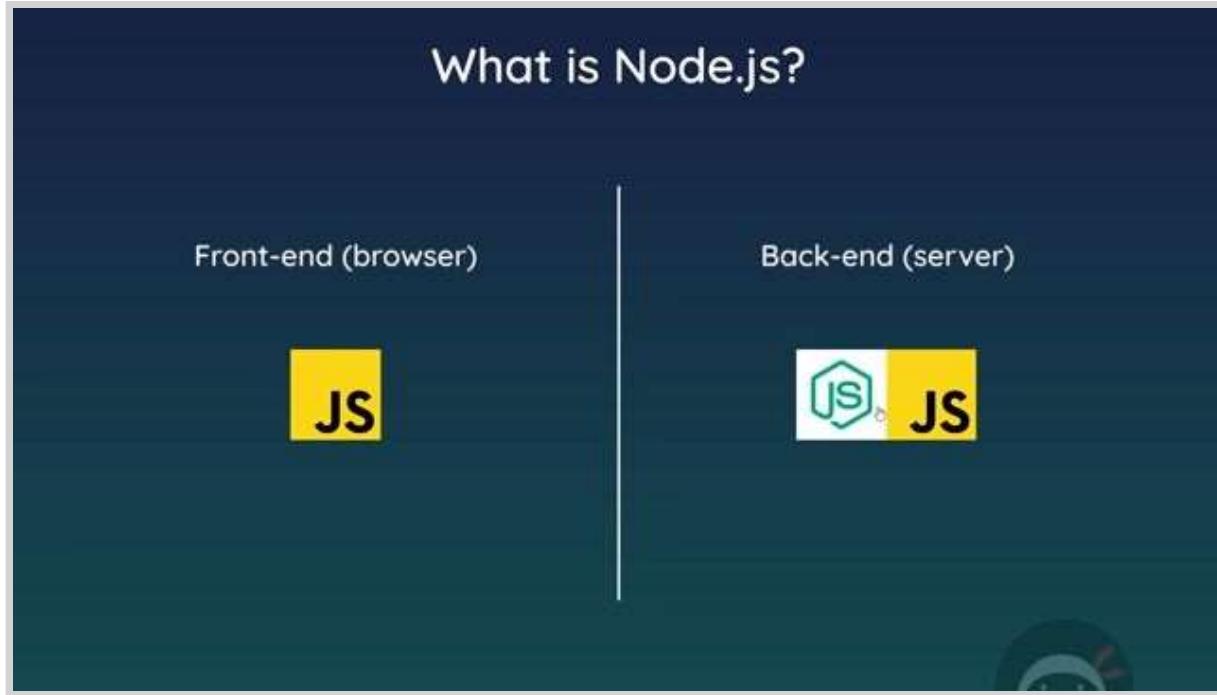




## Video1:

[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](#)

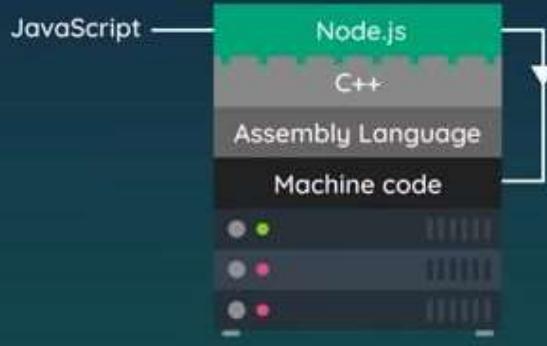


03:53



05:02

## Computers & Code



**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

### The Role of Node.js



**benefits :**

06:32

## But why use Node.js?

- No need to learn an extra language for server
- Can share code between front and back end
- Node.js has a massive community behind it
- Huge amount of third-party packages & tools to help

07:13

## In this course...

- How to install Node & use it to run JavaScript
- How to read & write files on your computer
- How to create a server using Node.js to create a website
- How to create an Express app / website
- How to use MongoDB (a NoSQL database)
- How to use template engines to easily create HTML views
- Put everything together to make a simple blog site

**EJS => for dynamic content**

09:07

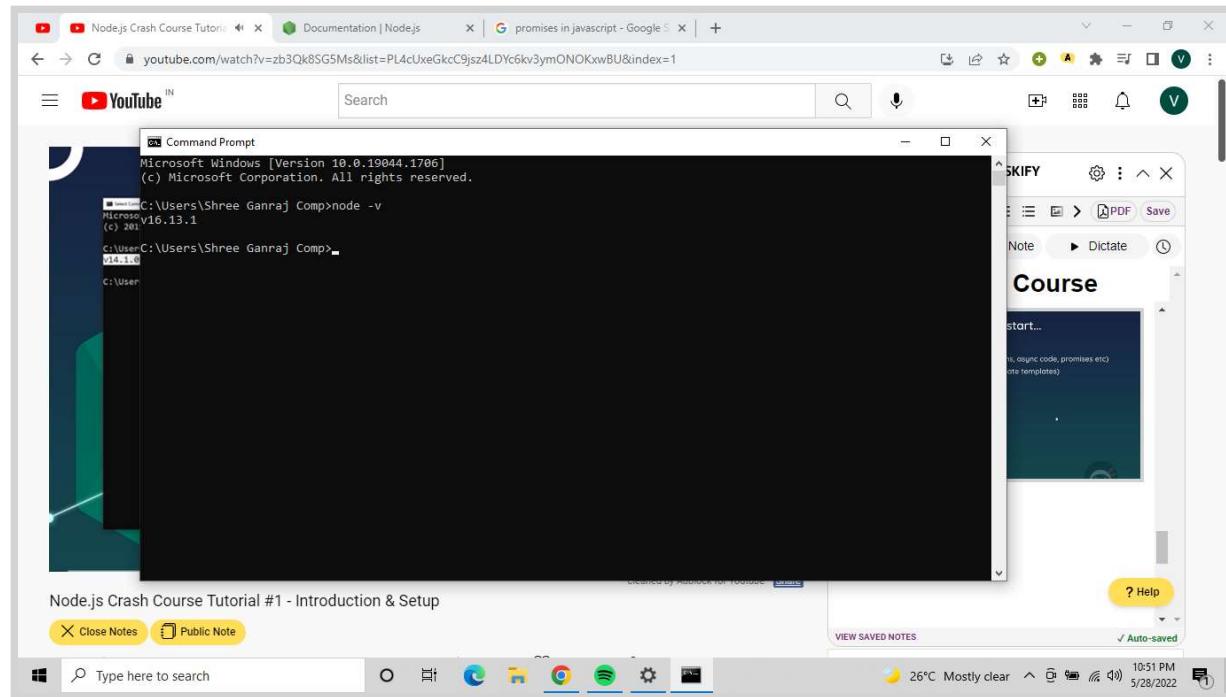
## Before you start...

- The foundations of JavaScript (functions, async code, promises etc)
- HTML & CSS (we'll be using these to create templates)



## Installing nodejs:

<https://nodejs.org/en/>



```
Command Prompt - node
Microsoft Windows [Version 10.0.19044.1706]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Shree Ganraj Comp>node -v
v16.13.1

C:\Users\Shree Ganraj Comp>node
Welcome to Node.js v16.13.1.
Type ".help" for more information.
> 5+5
10
> var name = 'Vaidehi'
undefined
> name
'Vaidehi'
>
```

```
Command Prompt
Microsoft Windows [Version 10.0.19044.1706]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Shree Ganraj Comp>node -v
v16.13.1

C:\Users\Shree Ganraj Comp>node
Welcome to Node.js v16.13.1.
Type ".help" for more information.
> 5+5
10
> var name = 'Vaidehi'
undefined
> name
'Vaidehi'
>
(To exit, press Ctrl+C again or Ctrl+D or type .exit)
>

C:\Users\Shree Ganraj Comp>cd Documents

C:\Users\Shree Ganraj Comp\Documents>mkdir node-crash-course
C:\Users\Shree Ganraj Comp\Documents>cd node-crash-course
C:\Users\Shree Ganraj Comp\Documents\node-crash-course>code .
```

## Running File through Nodejs:

The screenshot shows two separate instances of Visual Studio Code. Each instance has an 'EXPLORER' sidebar on the left and a 'TERMINAL' tab open at the bottom. In the top instance, the terminal window displays:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node test
valdehi
```

In the bottom instance, the terminal window displays:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node test
valdehi
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course>
```

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

## Video2: Nodejs basics

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a folder named "NODE-CRASH-COURSE" containing a file named "test.js".
- Editor:** Displays the content of "test.js":

```
1 const name = 'Vaidehi';
2 console.log(name);
3
4 const greet = (name) => {
5   console.log(`Hello, ${name}`);
6 }
7
8 greet('Vaidehi');
9 greet('Varada');
```

- Terminal:** Shows a Windows PowerShell window with the following output:

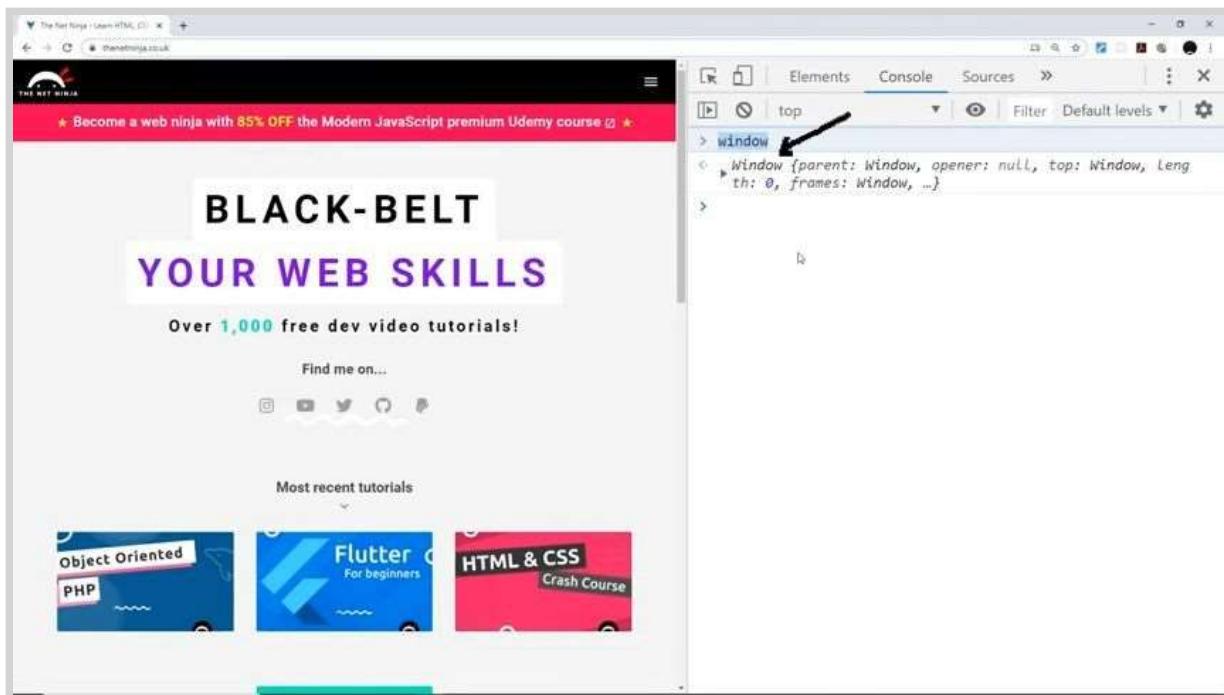
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node test
Vaidehi
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node test
Vaidehi
Hello, Vaidehi
Hello, Varada
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course>
```
- Bottom Bar:** Includes icons for Run Tests, Live Share, and a search bar.

## The global object:

02:47



**03:06**

The screenshot shows a web browser window with the URL <https://thenetninja.co.uk/>. The main content is a landing page for a course titled "BLACK-BELT YOUR WEB SKILLS". It features a banner stating "Over 1,000 free dev video tutorials!". Below this is a "Find me on..." section with links to various social media platforms. Further down, there's a "Most recent tutorials" section with three cards: "Object Oriented PHP", "Flutter For beginners", and "HTML & CSS Crash Course". On the right side of the browser window, the developer tools are open, specifically the "Console" tab. A command has been entered: `window.setTimeout(() => { alert('hello') }, 3000)`. The browser's status bar at the bottom shows the file path "C:\Users\Shree Ganraj Comp\Documents\node-crash-course\global.js" and the line number "Ln 1, Col 21".

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

**`setTimeout()` is the global object**

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

The screenshot shows a Visual Studio Code interface with a Node.js project titled "node-crash-course". The "global.js" file is open, containing the single line of code `console.log(global);`. In the terminal panel, the command `node global` is run, and the output is a large, detailed object dump of the global object in Node.js. The object includes properties like "global", "clearInterval", "clearTimeout", "setInterval", "setTimeout", "queueMicrotask", and "performance", along with their respective methods and values. The status bar at the bottom of the screen shows the file path "C:\Users\Shree Ganraj Comp\Documents\node-crash-course\global.js", line "Ln 1, Col 21", and other development details.

The screenshot displays two separate terminal sessions within the Visual Studio Code interface.

**Top Terminal Session:**

```
globaljs > ...
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
```

**Bottom Terminal Session:**

```
timeOrigin: 16537600093911.255
),
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>
```

**Code Editor:**

The code editor shows two files: `test.js` and `global.js`. The `global.js` file contains the code for the top terminal session, and the `test.js` file contains the code for the bottom terminal session.

```
globaljs > NODE-CRASH-COURSE
JS global.js
JS test.js
```

```
globaljs > 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);
```

```

File Edit Selection View Go Run Terminal Help
globaljs - node-crash-course - Visual Studio Code
EXPLORER ... JS Get Started JS test.js JS globaljs
NODE-CRASH-COURSE
JS globaljs
JS test.js

globaljs > [object Int] > setInterval() callback
1   console.log(global);
2
3
4   //global Object
5   global.setTimeout(() => {
6     console.log('in the timeout');
7     clearInterval(int);
8   }, 3000); //after 3 sec we get the msg
9
10 //now setInterval=>
11 const int = setInterval(() => {
12   console.log('in the interval');
13 }, 1000);

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
loopStart: -1,
loopExit: -1,
idleTime: 0
),
timeOrigin: 1653768356827.465
),
clearImmediate: [Function: clearImmediate],
setImmediate: [Function: setImmediate] {
  [Symbol(nodejs.util.promisify.custom)]: [Getter]
}
}
in the interval
in the interval
in the timeout
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course>

Type here to search  PowerShell + Live Share
Ln 11, Col 32 Spaces: 4 UTF-8 CRLF Go Live Prettier 11:22 PM 26°C Mostly clear 5/28/2022

```

dirname = absolute path of current folder

filename = absolute path of folder with file name

```

File Edit Selection View Go Run Terminal Help
globaljs - node-crash-course - Visual Studio Code
EXPLORER ... JS Get Started JS test.js JS globaljs
NODE-CRASH-COURSE
JS globaljs ...
1   console.log(global);
2
3
4   //global Object
5   global.setTimeout(() => {
6     console.log('in the timeout');
7     clearInterval(int);
8   }, 3000); //after 3 sec we get the msg
9
10 //now setInterval=>
11 const int = setInterval(() => {
12   console.log('in the interval');
13 }, 1000);
14
15 console.log(__filename);
16 console.log(__dirname);

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
idleTime: 0
),
timeOrigin: 1653768609593.313
),
clearImmediate: [Function: clearImmediate],
setImmediate: [Function: setImmediate] {
  [Symbol(nodejs.util.promisify.custom)]: [Getter]
}
}
C:\Users\Shree Ganraj Comp\Documents\node-crash-course\global.js
C:\Users\Shree Ganraj Comp\Documents\node-crash-course
in the interval
in the interval
in the timeout
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course>

Type here to search  PowerShell + Live Share
Ln 16, Col 24 Spaces: 4 UTF-8 CRLF Go Live Prettier 11:26 PM 26°C Mostly clear 5/28/2022

```

**document. nahi use kru shkt coz to global object nahiye...node madhe**

09:13

The screenshot shows a Visual Studio Code interface. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The title bar indicates "global.js - node-crash-course - Visual Studio Code". The Explorer sidebar on the left shows a project structure with "OPEN EDITORS" containing "global.js" and "NODE-CRASH-COURSE" containing "global.js". The main editor area displays the following code:

```

1 // Global object
2
3 // console.log(global);
4
5 // setTimeout(() => {
6 //   console.log('in the timeout');
7 //   clearInterval(int);
8 // }, 3000);
9
10 // const int = setInterval(() => {
11 //   console.log('in the interval');
12 // }, 1000);
13
14 // console.log(__dirname);
15 // console.log(__filename);
16
17 console.log(document.querySelector);

```

The terminal below shows the following error output:

```

ReferenceError: document is not defined
    at Object.<anonymous> (C:\Users\Shaun\Documents\Tuts\node-crash-course\global.js:17:13)
    at Module._compile (internal/modules/cjs/loader.js:1126:30)
    at Object.Module._extensions..js (internal/modules/cjs/loader.js:1136:18)
    at Module.load (internal/modules/cjs/loader.js:1040:32)
    at Function.Module._load (internal/modules/cjs/loader.js:929:14)
    at Function.executeUserEntryPoint [as runMain] (internal/modules/run_main.js:71:12)
    at internal/main/run_main_module.js:17:47
PS C:\Users\Shaun\Documents\Tuts\node-crash-course>

```

## Modules and Require:

reusable banvu shkto

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

The screenshot shows a Visual Studio Code interface. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and a tab for "people.js - node-crash-course - Visual Studio Code". The Explorer sidebar on the left shows a project structure with "NODE-CRASH-COURSE" containing "global.js", "modules.js", "people.js", and "test.js". The main editor area displays the following code:

```

1 const people = ['vaidehi', 'varada', 'pranjali', 'yash', 'shubham']; // array of strings
2 console.log(people);
3

```

The screenshot displays two instances of Visual Studio Code side-by-side, both showing the same workspace named "node-crash-course".

**Left Instance (Syntax Error):**

- Explorer:** Shows files: global.js, modules.js, people.js, test.js.
- Terminal:** Contains the following text:

```
timeOrigin: 1653760600503.313
},
clearImmediate: [Function: clearImmediate],
setImmediate: [Function: setImmediate] {
  [Symbol(nodejs.util.promisify.custom)]: [Getter]
}
}
C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node modules
[ 'valdehi', 'varada', 'pranjali', 'yash', 'shubham' ]
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course>
```
- Status Bar:** Shows Ln 2, Col 45, Spaces: 4, UTF-8, CRLF, Go Live, Prettier.

**Right Instance (Successful Execution):**

- Explorer:** Shows files: global.js, modules.js, people.js, test.js.
- Terminal:** Contains the following text:

```
setImmediate: [Function: setImmediate] {
  [Symbol(nodejs.util.promisify.custom)]: [Getter]
}
}
C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node modules
[ 'valdehi', 'varada', 'pranjali', 'yash', 'shubham' ]
C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node modules
[ 'valdehi', 'varada', 'pranjali', 'yash', 'shubham' ]
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course>
```
- Status Bar:** Shows Ln 4, Col 18, Spaces: 4, UTF-8, CRLF, Go Live, Prettier.

**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

The screenshot shows a Visual Studio Code interface with the following details:

- File Structure (Explorer):** Shows files in the "NODE-CRASH-COURSE" folder: modules.js, people.js.
- Code Editor (modules.js):**

```
1 const xyz = require('./people');
2
3 console.log(xyz);
4 console.log(people);
```
- Terminal:**

```
C:\Users\Shaun\Documents\Tuts\node-crash-course\modules.js:4
  console.log(people);

ReferenceError: people is not defined
    at Object.<anonymous> (C:\Users\Shaun\Documents\Tuts\node-crash-course\modules.js:4:13)
    at Module._compile (internal/modules/cjs/loader.js:1176:30)
    at Object.Module._extensions..js (internal/modules/cjs/loader.js:1196:10)
    at Module.load (internal/modules/cjs/loader.js:1040:32)
```

**to access people inside modules=>  
we have to export**

The screenshot shows a Visual Studio Code interface with the following details:

- File Structure (Explorer):** Shows files in the "NODE-CRASH-COURSE" folder: global.js, modules.js, people.js, test.js.
- Code Editor (people.js):**

```
1 const people = ['vaidehi', 'varada', 'pranjali', 'yash', 'shubham']; // array of strings
2
3
4 module.exports = 'hiee';
5
```
- Terminal:**

```
C:\Users\Shree Ganraj Comp\Documents\node-crash-course\global.js
C:\Users\Shree Ganraj Comp\Documents\node-crash-course
in the interval
in the interval
in the timeout
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node modules
[ 'vaidehi', 'varada', 'pranjali', 'yash', 'shubham' ]
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node modules
[ 'vaidehi', 'varada', 'pranjali', 'yash', 'shubham' ]
()
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node modules
[ 'vaidehi', 'varada', 'pranjali', 'yash', 'shubham' ]
hiee
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course>
```

A screenshot of Visual Studio Code showing a file named `people.js` in the editor. The code exports a single array of strings:

```
JS people.js > ...
1 const people = ['vaidehi', 'varada', 'pranjali', 'yash', 'shubham']; // array of strings
2 console.log(people);
3
4 //module.exports = 'hiee';
5 module.exports = people;
6
```

The terminal below shows the output of running the script:

```
in the interval
in the interval
in the timeout
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node modules
[ 'vaidehi', 'varada', 'pranjali', 'yash', 'shubham' ]
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node modules
[ 'vaidehi', 'varada', 'pranjali', 'yash', 'shubham' ]
[]
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node modules
[ 'vaidehi', 'varada', 'pranjali', 'yash', 'shubham' ]
hiee
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node modules
[ 'vaidehi', 'varada', 'pranjali', 'yash', 'shubham' ]
[ 'vaidehi', 'varada', 'pranjali', 'yash', 'shubham' ]
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course>
```

**that's how we're exporting single thing  
now how we can export multiple things??**

=>

**14:50**

A screenshot of Visual Studio Code showing a file named `modules.js` in the editor. The code exports two variables:

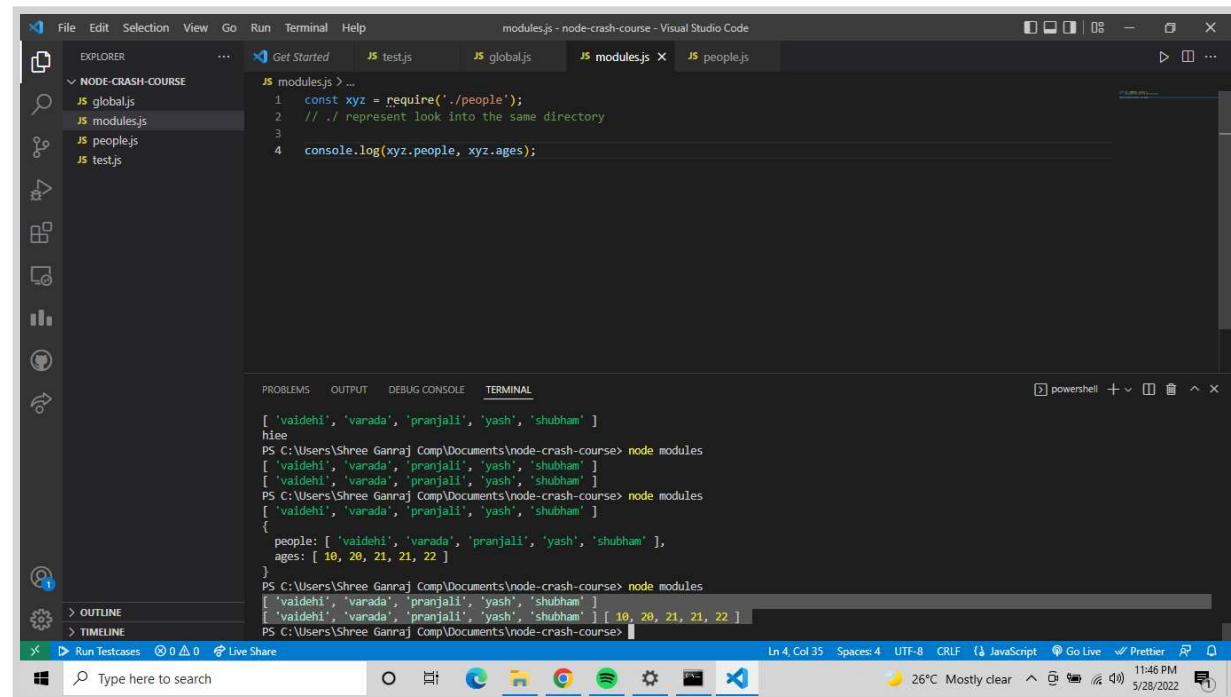
```
modules.js
5
6 module.exports = [
7   people: people,
8 ];
9 const people: string[];
```

A screenshot of Visual Studio Code showing a file named `people.js` in the editor. The code exports an object with properties `people` and `ages`:

```
people.js - node-crash-course - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER ... Get Started JS test.js JS global.js JS modules.js JS people.js
JS people.js > ...
1 const people = ['vaidehi', 'varada', 'pranjali', 'yash', 'shubham']; // array of strings
2 const ages = [18,20,21,21,22];
3 console.log(people);
4
5 //module.exports = 'hiee';
6 //module.exports = people;
7
8 module.exports = { //object with properties
9   people: people,
10   ages: ages
11 }
12
13
```

The terminal below shows the output of running the script:

```
in the interval
in the interval
in the timeout
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node modules
[ 'vaidehi', 'varada', 'pranjali', 'yash', 'shubham' ]
[]
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node modules
[ 'vaidehi', 'varada', 'pranjali', 'yash', 'shubham' ]
[ 'vaidehi', 'varada', 'pranjali', 'yash', 'shubham' ]
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node modules
[ 'vaidehi', 'varada', 'pranjali', 'yash', 'shubham' ]
{
  people: [ 'vaidehi', 'varada', 'pranjali', 'yash', 'shubham' ],
  ages: [ 18, 20, 21, 21, 22 ]
}
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course>
```



The screenshot shows a VS Code interface with the following details:

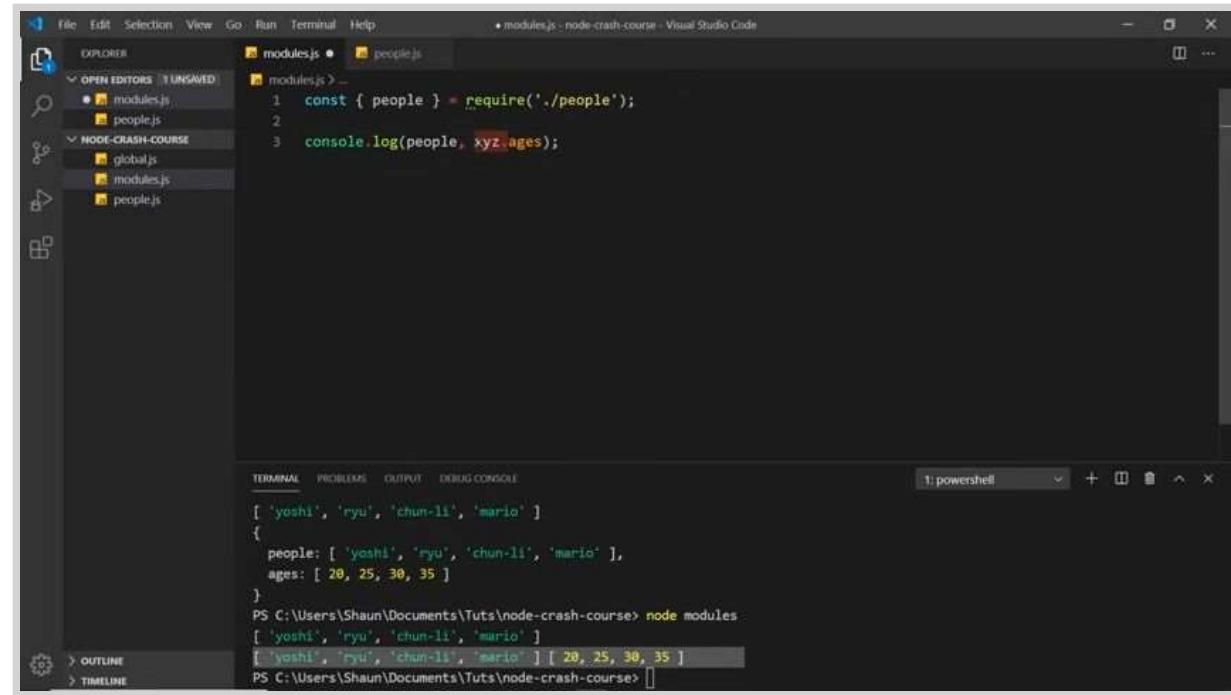
- File Explorer:** Shows a folder named "NODE-CRASH-COURSE" containing files: global.js, modules.js, people.js, and test.js.
- Editor:** The "modules.js" file is open, displaying the following code:
 

```
JS modules.js > ...
1 const xyz = require('./people');
2 // ./ represent look into the same directory
3
4 console.log(xyz.people, xyz.ages);
```
- Terminal:** The terminal window shows the output of running the script:
 

```
[ 'vaidehi', 'varada', 'pranjali', 'yash', 'shubham' ]
hee
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node modules
[ 'vaidehi', 'varada', 'pranjali', 'yash', 'shubham' ]
[ 'vaidehi', 'varada', 'pranjali', 'yash', 'shubham' ]
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node modules
[ 'vaidehi', 'varada', 'pranjali', 'yash', 'shubham' ]
{
  people: [ 'vaidehi', 'varada', 'pranjali', 'yash', 'shubham' ],
  ages: [ 10, 20, 21, 21, 22 ]
}
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node modules
[ 'vaidehi', 'varada', 'pranjali', 'yash', 'shubham' ]
[ 'vaidehi', 'varada', 'pranjali', 'yash', 'shubham' ] [ 10, 20, 21, 21, 22 ]
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course>
```
- Bottom Status Bar:** Shows the current time as 11:46 PM, the date as 5/28/2022, and the weather as 26°C Mostly clear.

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

16:21



The screenshot shows a VS Code interface with the following details:

- File Explorer:** Shows a folder named "NODE-CRASH-COURSE" containing files: global.js, modules.js, people.js, and test.js. The "modules.js" file is currently selected.
- Editor:** The "modules.js" file is open, displaying the following code:
 

```
modules.js • people.js
modules.js > ...
1 const { people } = require('./people');
2
3 console.log(people, xyz.ages);
```
- Terminal:** The terminal window shows the output of running the script:
 

```
[ 'yoshi', 'ryu', 'chun-li', 'mario' ]
{
  people: [ 'yoshi', 'ryu', 'chun-li', 'mario' ],
  ages: [ 20, 25, 30, 35 ]
}
PS C:\Users\Shaun\Documents\Tuts\node-crash-course> node modules
[ 'yoshi', 'ryu', 'chun-li', 'mario' ]
[ 'yoshi', 'ryu', 'chun-li', 'mario' ] [ 20, 25, 30, 35 ]
PS C:\Users\Shaun\Documents\Tuts\node-crash-course>
```
- Bottom Status Bar:** Shows the current time as 11:46 PM, the date as 5/28/2022, and the weather as 26°C Mostly clear.

17:00

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows files in the "NODE-CRASH-COURSE" folder: global.js, modules.js, people.js.
- Terminal:** Running in powershell, showing the execution of "node modules". The output shows two runs of the code, each printing an array of names and ages.

```

at Function.executeUserEntryPoint [as runMain] (internal/modules/run_main.js:71:12)
at internal/main/run_main_module.js:17:47
PS C:\Users\Shaun\Documents\Tuts\node-crash-course> node modules
[ 'yoshi', 'ryu', 'chun-li', 'mario' ]
[ 'yoshi', 'ryu', 'chun-li', 'mario' ]
PS C:\Users\Shaun\Documents\Tuts\node-crash-course> node modules
[ 'yoshi', 'ryu', 'chun-li', 'mario' ]
[ 'yoshi', 'ryu', 'chun-li', 'mario' ] [ 20, 25, 30, 35 ]
PS C:\Users\Shaun\Documents\Tuts\node-crash-course>
  
```

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows files in the "NODE-CRASH-COURSE" folder: global.js, modules.js, people.js, test.js.
- Terminal:** Running in powershell, showing the execution of "node modules". The output shows two runs of the code, each printing an array of names and ages.

```

PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node modules
[ 'vaidehi', 'varada', 'pranjali', 'yash', 'shubham' ]
[ 'vaidehi', 'varada', 'pranjali', 'yash', 'shubham' ]
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node modules
[ 'vaidehi', 'varada', 'pranjali', 'yash', 'shubham' ]
{
  people: [ 'vaidehi', 'varada', 'pranjali', 'yash', 'shubham' ],
  ages: [ 10, 20, 21, 21, 22 ]
}
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node modules
[ 'vaidehi', 'varada', 'pranjali', 'yash', 'shubham' ]
[ 'vaidehi', 'varada', 'pranjali', 'yash', 'shubham' ] [ 10, 20, 21, 21, 22 ]
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node modules
[ 'vaidehi', 'varada', 'pranjali', 'yash', 'shubham' ]
[ 'vaidehi', 'varada', 'pranjali', 'yash', 'shubham' ] [ 10, 20, 21, 21, 22 ]
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course>
  
```

build in modules astat kahi

js ki "os";

The screenshot shows a Visual Studio Code interface. The left sidebar has an 'EXPLORER' view with a folder named 'NODE-CRASH-COURSE' containing files: 'global.js', 'modules.js', 'people.js', and 'test.js'. The main 'CODE' area displays a JavaScript file named 'modules.js' with the following code:

```

JS modules.js > ...
1 const {people, ages} = require('../people');
2 // ./ represent look into the same directory
3
4 console.log(people, ages);
5
6 const os = require('os');
7 console.log(os);
8
9 console.log(os); //info about os
10

```

The 'TERMINAL' tab at the bottom shows the command line output:

```

SIGINCH: 28
},
Priority: [Object: null prototype] {
  PRIORITY_LOW: 19,
  PRIORITY_BELOW_NORMAL: 10,
  PRIORITY_NORMAL: 0,
  PRIORITY_ABOVE_NORMAL: -7,
  PRIORITY_HIGH: -14,
  PRIORITY_HIGHEST: -20
}
},
EOL: '\r\n',
devNull: '\\\\.\\\\nul'
}
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course>

```

The status bar at the bottom right indicates: Ln 10, Col 1 Spaces: 4 UTF-8 CRLF Go Live Prettier 12:04 AM 26°C Mostly clear 5/29/2022

18:23

The screenshot shows a Visual Studio Code interface. The left sidebar has an 'EXPLORER' view with a folder named 'NODE-CRASH-COURSE' containing files: 'global.js', 'modules.js', 'people.js', and 'test.js'. The main 'CODE' area displays a JavaScript file named 'modules.js' with the following code:

```

JS modules.js > ...
1 const {people, ages} = require('../people');
2 // ./ represent look into the same directory
3
4 console.log(people, ages);
5
6 const os = require('os');
7 console.log(os);
8
9 console.log(os.platform(), os.homedir()); //info about os
10

```

The 'TERMINAL' tab at the bottom shows the command line output:

```

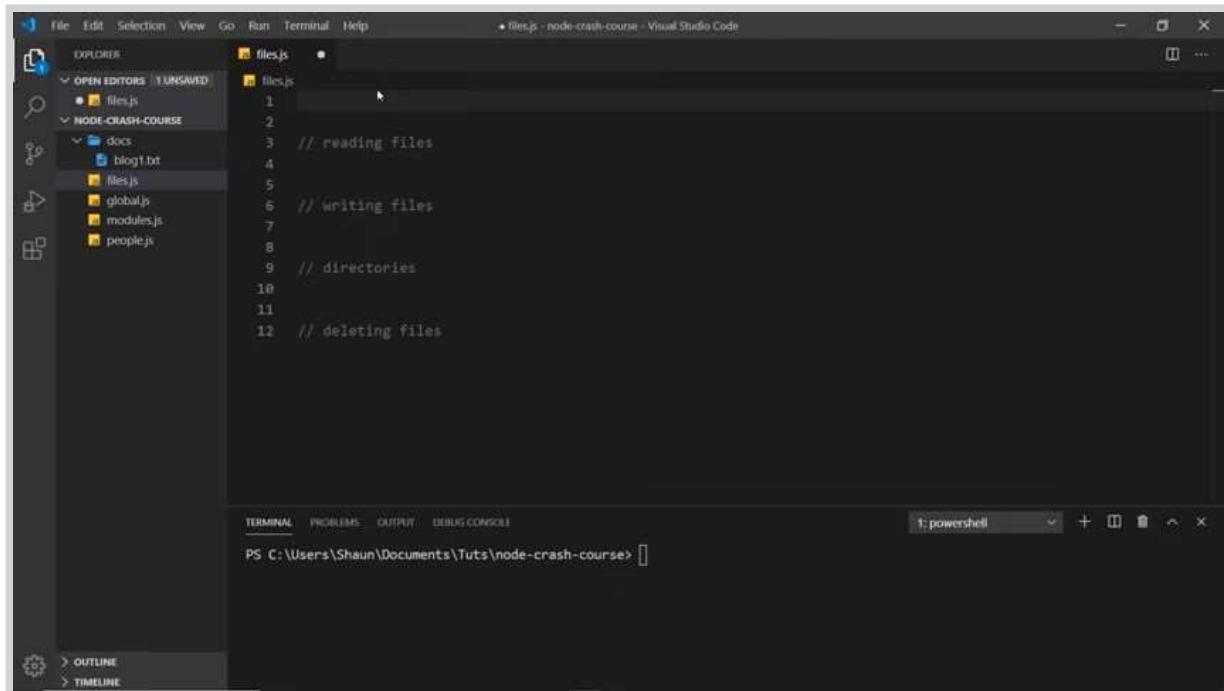
),
Priority: [Object: null prototype] {
  PRIORITY_LOW: 19,
  PRIORITY_BELOW_NORMAL: 10,
  PRIORITY_NORMAL: 0,
  PRIORITY_ABOVE_NORMAL: -7,
  PRIORITY_HIGH: -14,
  PRIORITY_HIGHEST: -20
}
),
EOL: '\r\n',
devNull: '\\\\.\\\\nul'
}
win32 C:\Users\Shree Ganraj Comp
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course>

```

The status bar at the bottom right indicates: Ln 9, Col 41 Spaces: 4 UTF-8 CRLF Go Live Prettier 12:05 AM 26°C Mostly clear 5/29/2022

**The file system...modules use kru shkto using node..  
fs module which is used for file system**

19:33



```

File Edit Selection View Go Run Terminal Help
files.js - node-crash-course - Visual Studio Code

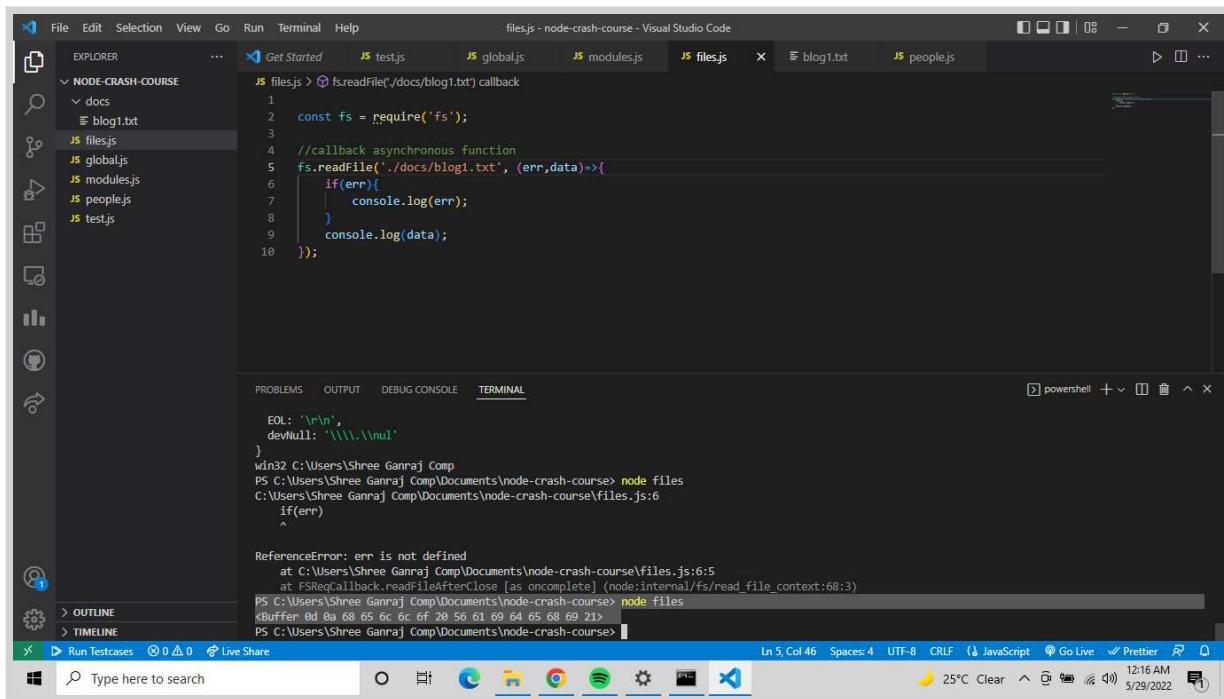
EXPLORER OPEN EDITORS 1 UNSAVED
NODE-CRASH-COURSE
docs
  blog1.txt
  files.js
  global.js
  modules.js
  people.js

files.js
1
2
3 // reading files
4
5
6 // writing files
7
8 // directories
9
10
11
12 // deleting files

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE
PS C:\Users\Shaun\Documents\Tuts\node-crash-course> []

OUTLINE TIMELINE

```



```

File Edit Selection View Go Run Terminal Help
files.js - node-crash-course - Visual Studio Code

EXPLORER ...
NODE-CRASH-COURSE
docs
  blog1.txt
JS files.js
JS global.js
JS modules.js
JS files.js x
JS blog1.txt
JS people.js

files.js > ⚡ fs.readFile('./docs/blog1.txt') callback
1
2 const fs = require('fs');
3
4 //callback asynchronous function
5 fs.readFile('./docs/blog1.txt', (err,data) => {
6   if(err){
7     console.log(err);
8   }
9   console.log(data);
10 });

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
EOL: '\r\n',
devNull: '\\\\.\\\\nul'
}
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 6f 20 56 61 69 64 65 68 69 21>
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course>

Run Tests 0 ▲ 0 Live Share Type here to search
Ln 5, Col 46 Spaces: 4 UTF-8 CRLF Go Live Prettier 25°C Clear 12:16 AM 5/29/2022

```

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a project structure under "NODE-CRASH-COURSE" with files: blog1.txt, filesjs, globaljs, modules.js, people.js, test.js.
- Code Editor:** The "files.js" tab is active, displaying the following code:
 

```
1 const fs = require('fs');
2 //callback asynchronous function
3 fs.readFile('./docs/blog1.txt', (err,data)=>{
4   if(err){
5     console.log(err);
6   }
7   //console.log(data); //buffer is a package of data being sent when we read this file
8   console.log(data.toString());
9 });
10 
```
- Terminal:** Shows the output of running the script:
 

```
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 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 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!
```
- Status Bar:** Shows "Ln 9, Col 7" and "12:18 AM 5/29/2022".

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Same as the first screenshot.
- Code Editor:** The "files.js" tab is active, displaying the following code:
 

```
1 const fs = require('fs');
2 //callback asynchronous function
3 fs.readFile('./docs/blog1.txt', (err,data)=>{
4   if(err){
5     console.log(err);
6   }
7   //console.log(data); //buffer is a package of data being sent when we read this file
8   console.log(data.toString());
9 };
10
11 // as javascript asynchronous wr work krto..tr
12 //file madhla data read karayla vel lagto So, aadhi hi khalchi line execute houn jail mean time madhe
13
14 console.log('last line....');
```
- Terminal:** Shows the output of running the script:
 

```
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 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 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!
```
- Status Bar:** Shows "Ln 14, Col 102" and "12:21 AM 5/29/2022".

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left shows a project structure with files like 'docs/blog1.txt', 'global.js', 'modules.js', 'people.js', 'test.js', and 'files.js'. The 'files.js' file is open in the editor, displaying code related to file operations. The Terminal tab at the bottom shows the command-line output of running the script, which includes logs and the creation of a file named 'blog1.txt'.

```

File Edit Selection View Go Run Terminal Help
files.js - node-crash-course - Visual Studio Code
EXPLORER ... JS test.js JS global.js JS modules.js JS files.js x JS blog1.txt JS people.js ...
JS files.js > fs.writeFile('docs/blog1.txt', 'hello world') callback
11   // as javascript asynchronous wr work krtto..tr
12   console.log(data.toString());
13 });
14
15 // as javascript asynchronous wr work krtto..tr
16 //file madhla data read karayla vel lagto So, aadhi hi khalchi line execute houn jail mean time madhe
17 console.log('last line....');
18
19
20 //write File
21 fs.writeFile('../docs/blog1.txt', 'hello world', ()=>{
22   | console.log("file is written");
23 })

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```

<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> node files
last line....
file is written
hello world
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> []

```

Ln 22, Col 37 Spaces: 4 UTF-8 CRLF ↳ JavaScript ⚡ Go Live ✅ Prettier 🔍 12:25 AM 5/29/2022

This screenshot shows the same Visual Studio Code environment as the previous one, but it includes a new file named 'blog2.txt' in the project structure. The terminal output shows that the script has created both 'blog1.txt' and 'blog2.txt' files, each containing the string 'hello world'.

```

File Edit Selection View Go Run Terminal Help
files.js - node-crash-course - Visual Studio Code
EXPLORER ... JS test.js JS global.js JS modules.js JS files.js x JS blog1.txt JS people.js ...
JS files.js > ...
11   console.log(data.toString());
12 }
13
14
15 // as javascript asynchronous wr work krtto..tr
16 //file madhla data read karayla vel lagto So, aadhi hi khalchi line execute houn jail mean time madhe
17 console.log('last line....');
18
19
20 //write File
21 fs.writeFile('../docs/blog1.txt', 'hello world', ()=>{
22   | console.log("file is written");
23 })
24
25 //blog2 exist navhti tr new create krun lihun takle apoap tyane
26 fs.writeFile('../docs/blog2.txt', 'hello Vaidehi', ()=>{
27   | console.log("file is written");
28 })

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```

PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node files
last line....
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> []

```

Ln 25, Col 65 Spaces: 4 UTF-8 CRLF ↳ JavaScript ⚡ Go Live ✅ Prettier 🔍 12:26 AM 5/29/2022

## //directories:

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

26:16

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure under "NODE-CRASH-COURSE". Files listed include files.js, blog1.txt, blog2.txt, files.js (selected), global.js, modules.js, and people.js.
- Code Editor:** Displays the contents of files.js. The code uses the fs module to write to files and create directories.
- Terminal:** Shows the command "node files" being run in a PowerShell terminal. The output indicates that files were written and a folder was created.

```

files.js > fs.mkdir('./assets') callback
14 // fs.writeFileSync('./docs/blog1.txt', 'Hello, world', () => {
15 //   console.log('file was written');
16 // });
17
18 // fs.writeFileSync('./docs/blog2.txt', 'Hello, again', () => {
19 //   console.log('file was written');
20 // });
21
22 // directories
23 fs.mkdir('./assets', (err) => {
24   if (err) {
25     console.log(err);
26   }
27   console.log('folder created');
28 })
29
30
31 // deleting files

```

```

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

```

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure under "NODE-CRASH-COURSE". Files listed include assets, blog1.txt, blog2.txt, files.js (selected), global.js, modules.js, people.js, and test.js.
- Code Editor:** Displays the contents of files.js. The code uses the fs module to write to files and create directories.
- Terminal:** Shows the command "node files" being run in a PowerShell terminal. The output indicates that files were written and a folder was created.

```

JS files.js > ...
20 //write File
21 fs.writeFileSync('./docs/blog1.txt', 'hello world', ()=>{
22   |   console.log("file is written");
23 })
24
25 //blog2 exist navhti tr new create krun lihun takle apoap tyane
26 fs.writeFileSync('./docs/blog2.txt', 'hello Vaidehi', ()=>{
27   |   console.log("file is written");
28 })
29
30
31 //directory,folder create krne
32 fs.mkdir('./assets', (err)=>{
33   |   if(err){
34   |     |   console.log(err);
35   |   }
36   |   console.log('folder created');
37 })
38

```

```

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>

```

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows a project structure under "NODE-CRASH-COURSE" with files: blog1.txt, blog2.txt, files.js, global.js, modules.js, people.js, and test.js.
- Code Editor:** The "files.js" file is open, containing the following code:
 

```

23  })
24
25 //blog2 exist navhti tr new create krun lihun takle apoap tyane
26 fs.writeFileSync('./docs/blog2.txt', 'Hello Vaidehi', ()=>{
27   console.log("file is written");
28 }
29
30
31 //directory,folder create krne
32 //ani jr he code parat llihila ani run kela tr yeil folder already exists
33 if(!fs.existsSync("./assets")){
34
35   fs.mkdir("./assets", (err)=>{
36     if(err){
37       console.log(err);
38     }
39     console.log('folder created');
40   });
41 }
42
43
      
```
- Terminal:** The terminal shows the output of running the script:
 

```

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>
      
```
- Bottom Bar:** Includes icons for Run Testcases, Live Share, and a search bar.

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows a project structure under "NODE-CRASH-COURSE" with files: blog1.txt, blog2.txt, files.js, global.js, modules.js, people.js, and test.js.
- Code Editor:** The "files.js" file is open, containing the following code:
 

```

28 })
29
30
31 //directory,folder create krne
32 //ani jr he code parat llihila ani run kela tr yeil folder already exists
33 if(!fs.existsSync("./assets")){
34
35   fs.mkdir("./assets", (err)=>{
36     if(err){
37       console.log(err);
38     }
39     console.log('folder created');
40   });
41 }
42
43 else{
44   fs.rmdir("./assets", (err)=>{
45     if(err){
46       console.log(err);
47     }
48     console.log('folder deleted');
      
```
- Terminal:** The terminal shows the output of running the script:
 

```

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>
      
```
- Bottom Bar:** Includes icons for Run Testcases, Live Share, and a search bar.

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows a project structure under "NODE-CRASH-COURSE" with files: blog1.txt, blog2.txt, files.js, global.js, modules.js, people.js, and test.js.
- Code Editor:** The "files.js" file is open, containing Node.js code for file operations. The code creates a folder named "assets" if it doesn't exist, and then deletes it. It also logs "hello world" and "file is written" messages.
- Terminal:** The terminal shows the output of running the script: "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".
- Bottom Bar:** Includes a search bar, taskbar icons, and system status like weather (25°C), time (12:35 AM), and date (5/29/2022).

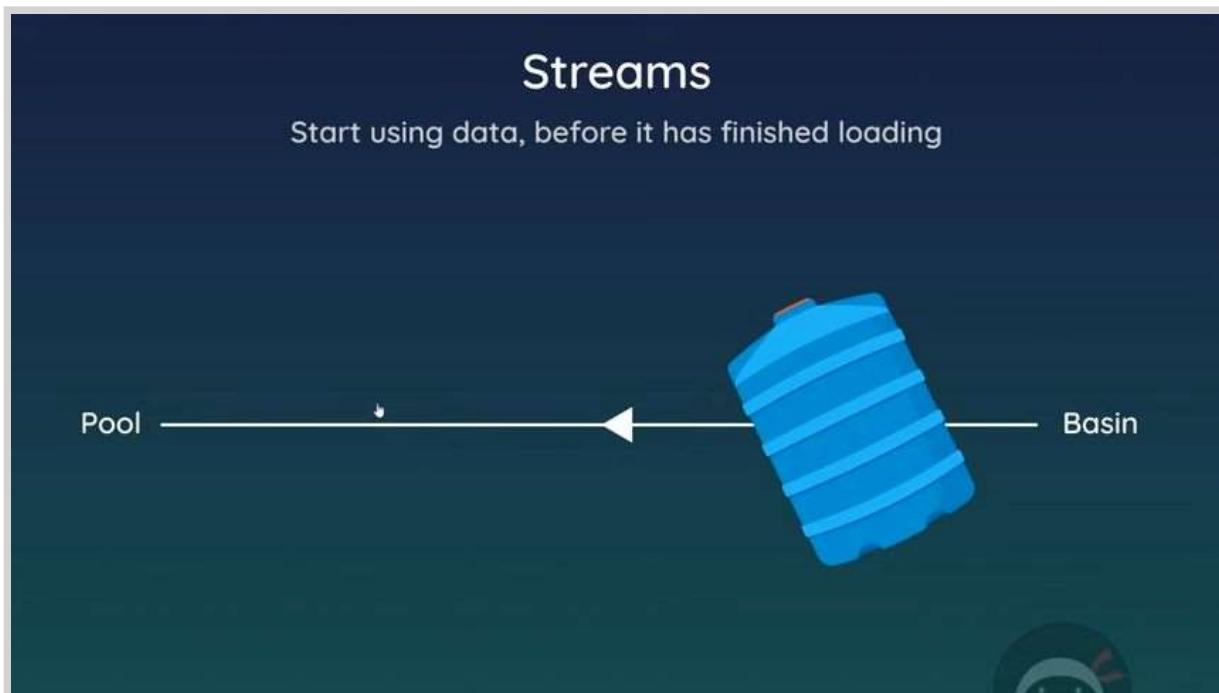
The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows a project structure under "NODE-CRASH-COURSE" with files: blog1.txt, blog2.txt, files.js, global.js, modules.js, people.js, and test.js.
- Code Editor:** The "files.js" file is open, containing Node.js code for file operations. It attempts to delete a file named "deleteme.txt" in the "docs" directory. It logs "file is written", "hello world", and "file deleted...".
- Terminal:** The terminal shows the output of running the script: "file is written", "hello world", "PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node files", "last line....", "folder deleted", "file deleted...", "file is written", "file is written", "hello world".
- Bottom Bar:** Includes a search bar, taskbar icons, and system status like weather (25°C), time (12:39 AM), and date (5/29/2022).

## streams & Buffers:

streams => live streams

32:50



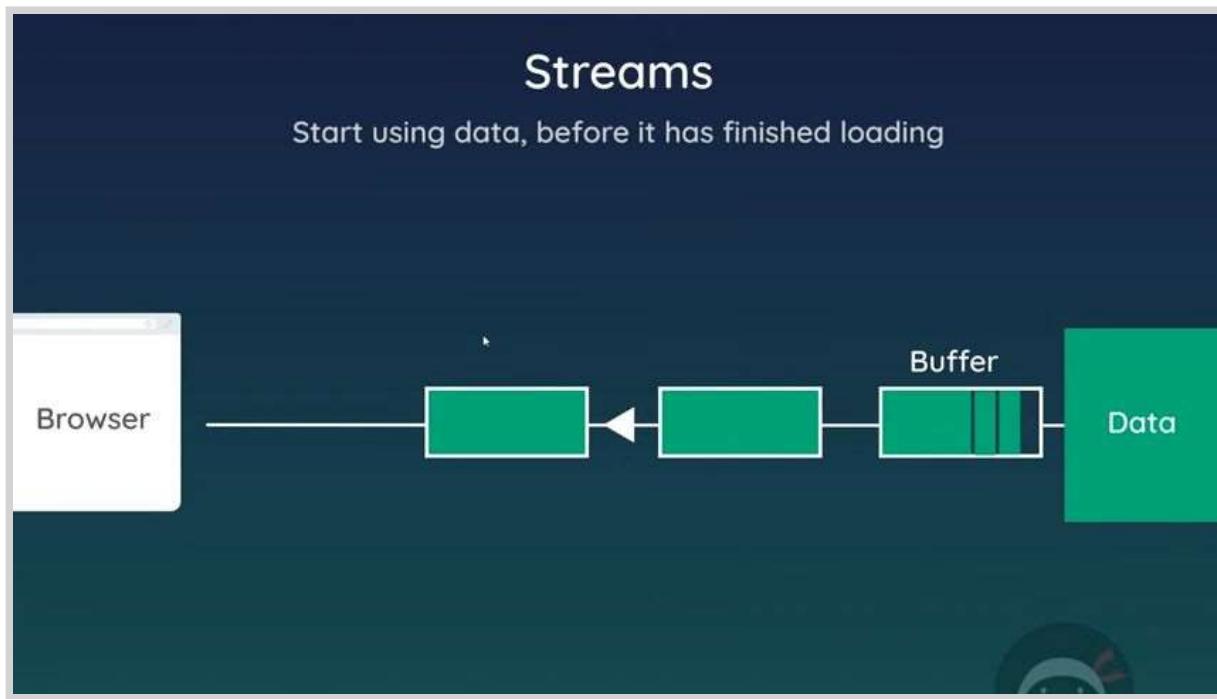
**to have a stream => so that aapn pani bhartanach pool use kru shkto**

33:01



**if huge amount of file is there=>  
small chunks madhe browser la detooy so that vel nko lagayla**

33:38



## //read streams and write streams:

```

File Edit Selection View Go Run Terminal Help streams.js - node-crash-course - Visual Studio Code
EXPLORER ... started JS test.js JS global.js JS modules.js JS files.js blog1.txt JS people.js JS streams.js blog3.txt ...
NODE-CRASH-COURSE
docs
blog1.txt
blog2.txt
blog3.txt
files.js
global.js
modules.js
people.js
streams.js
test.js

streams.js > ⚡ readStream.on('data') callback
1
2 const fs = require('fs');
3
4 const readStream = fs.createReadStream('./docs/blog3.txt');
5
6 readStream.on('data', (chunk)=>{
7   console.log('-----New Chunk-----');
8   console.log(chunk);
9 }) //listening to data event (event listener)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Copyright (C) Microsoft Corporation. All rights reserved.
Try the new cross-platform PowerShell https://aka.ms/pscore6
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node streams
-----New Chunk-----
<Buffer 54 68 69 73 20 70 61 67 65 20 68 61 73 20 4d 69 63 72 6f 73 6f 66 74 20 45 78 63 6f 6e 70 77 61 6d 70 6c 77 6f 70 74 73 20 74 69 c
1 74 ... 2884 more bytes>
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> []
Take Survey Remind Me later Don't Show Again
Ln 7, Col 45 Spaces: 4 UFT-8 CRLF JavaScript Go Live Prettier 9:29 AM 5/29/2022
Type here to search

```

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows a folder named "NODE-CRASH-COURSE" containing files: blog1.txt, blog2.txt, blog3.txt, files.js, global.js, modules.js, people.js, streams.js, and test.js.
- Code Editor:** The active file is "streams.js". The code reads a file named "blog3.txt" using the "fs.createReadStream" method and logs each chunk to the console.
- Terminal:** The terminal tab is open, showing notes about Microsoft Excel sample datasets.
- Taskbar:** Shows icons for Run Tests, Live Share, and a search bar.
- System Tray:** Displays weather (28°C Sunny), time (9:31 AM), and date (5/29/2022).

```

1
2 const fs = require('fs');
3
4 const readStream = fs.createReadStream('./docs/blog3.txt', {encoding:'utf8'}); //second argument is optional
5
6 readStream.on('data',(chunk)=>{
7   console.log('-----New Chunk-----');
8   console.log(chunk.toString());
9 }) //listening to data event (event listener)

```

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows a folder named "NODE-CRASH-COURSE" containing files: blog1.txt, blog2.txt, blog3.txt, blog4.txt, files.js, global.js, modules.js, people.js, streams.js, and test.js.
- Code Editor:** The active file is "streams.js". The code reads a file named "blog3.txt" and writes its contents to another file named "blog4.txt" using "readStream" and "writeStream" methods.
- Terminal:** The terminal tab is open, showing notes about Microsoft Excel sample datasets.
- Taskbar:** Shows icons for Run Tests, Live Share, and a search bar.
- System Tray:** Displays weather (28°C Sunny), time (9:34 AM), and date (5/29/2022).

```

1
2 const fs = require('fs');
3
4 const readStream = fs.createReadStream('./docs/blog3.txt', {encoding:'utf8'}); //second argument is optional
5 const writeStream = fs.createWriteStream('./docs/blog4.txt');
6 readStream.on('data',(chunk)=>{
7   console.log('-----New Chunk-----');
8   console.log(chunk.toString());
9   writeStream.write('\nNew Chunk\n');
10  writeStream.write(chunk);
11 }) //listening to data event (event listener)

```

instead of this use pipe => streams  
everytime we getting chunk it will pipe ...that

The screenshot shows a Visual Studio Code interface. The left sidebar displays a project structure for 'NODE-CRASH-COURSE' with files like 'docs/blog1.txt', 'files.js', 'modules.js', 'people.js', and 'streams.js'. The main code editor window shows a file named 'streams.js' with the following content:

```

1
2 const fs = require('fs');
3
4 const readStream = fs.createReadStream('./docs/blog3.txt', {encoding:'utf8'});
5 const writeStream = fs.createWriteStream('./docs/blog4.txt'); //second argument is optional
6 // readStream.on('data',(chunk)=>{
7 //   console.log('-----New Chunk-----');
8 //   console.log(chunk.toString());
9 //   writeStream.write(`\nNew Chunk\n`);
10 //   writeStream.write(chunk);
11 // }) //listening to data event (event listener)
12
13 //piping
14 readStream.pipe(writeStream);

```

The terminal tab at the bottom shows command-line history:

```

PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node streams
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node streams
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course>

```

The status bar at the bottom right indicates: Ln 11, Col 51 Spaces: 4 UTF-8 JavaScript Go Live Prettier 9:37 AM 28°C Sunny 5/29/2022

**duplex streams => r-w both**

## Video3:

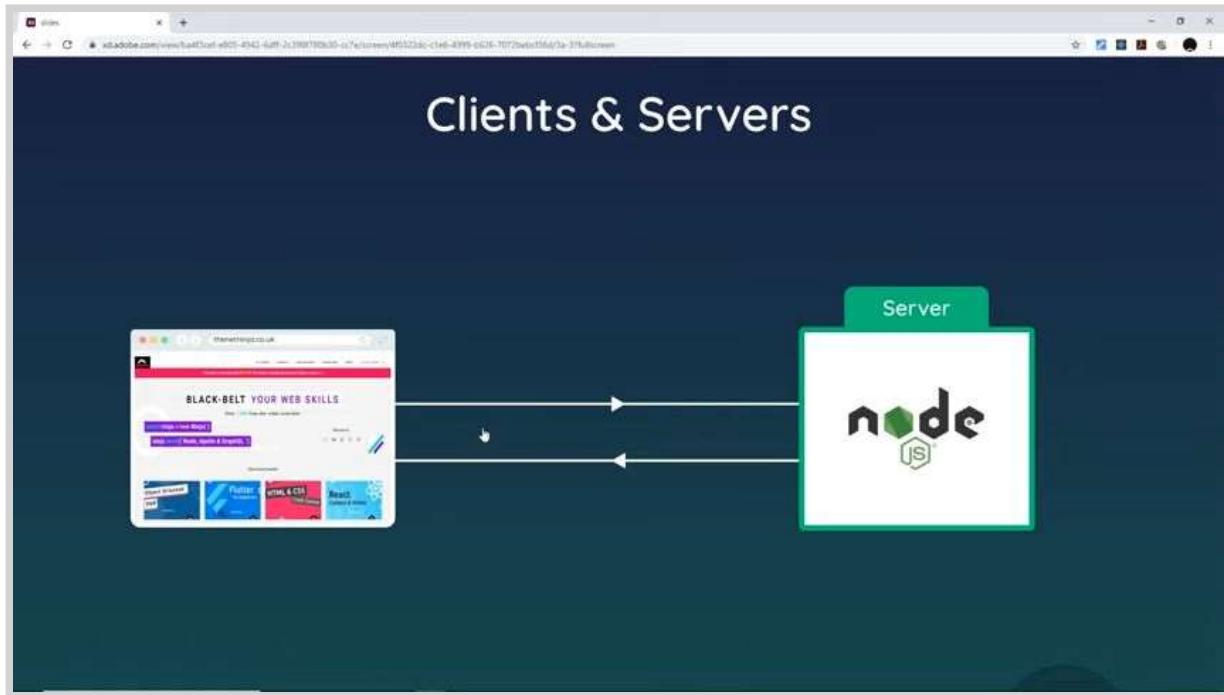
**Clients & Servers=>**

**server => listens incoming request from browser**

**how this happens??**

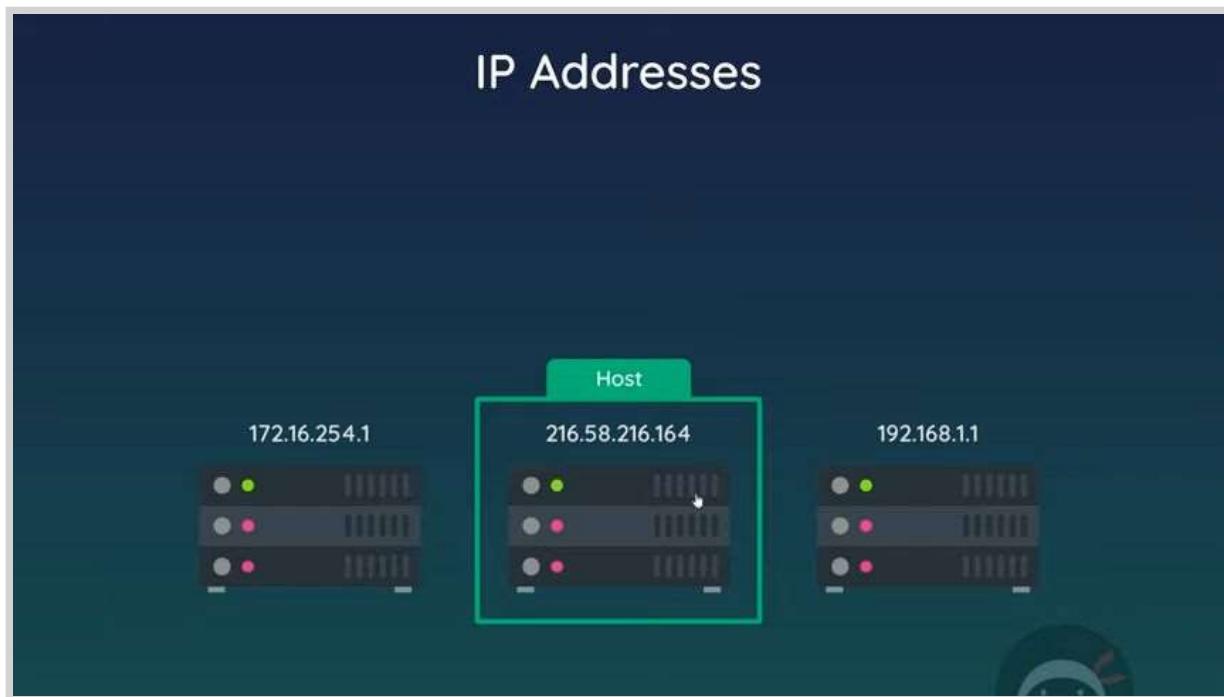
**=> when we write some URL on browser and press enter that sends request to the server**

**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

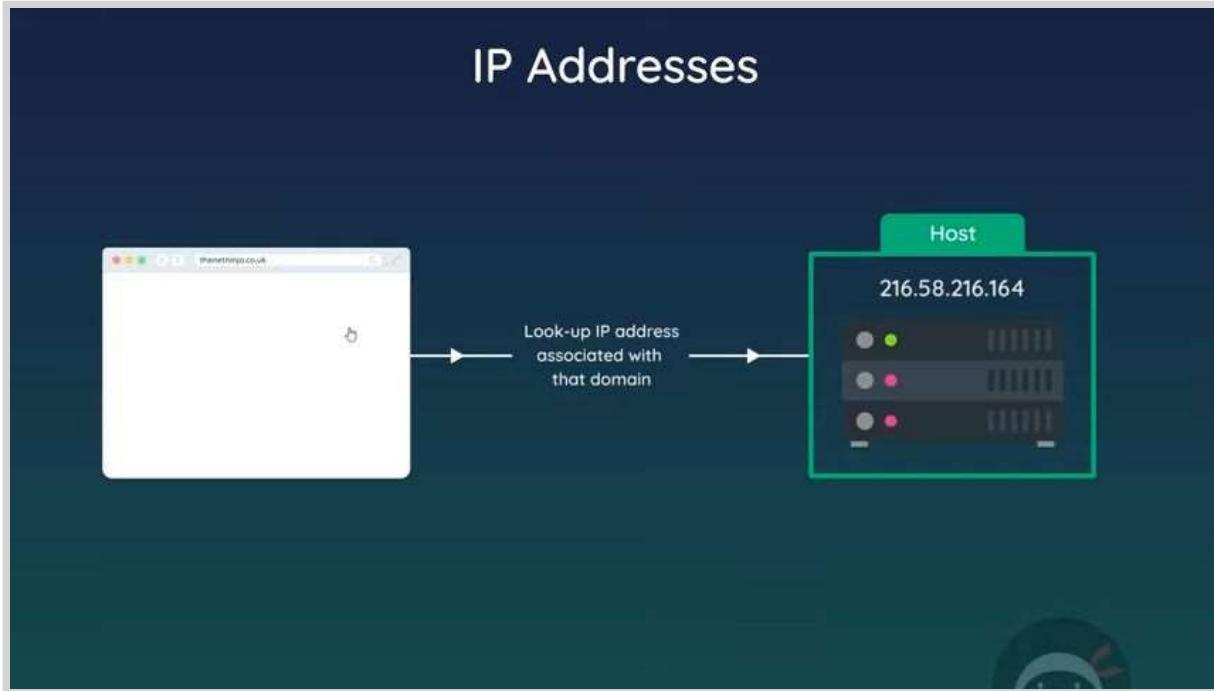


**if we want to connect to some computer then we should know the IP address of that**

**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..**

A screenshot of Visual Studio Code showing a file named 'server.js' in the center editor tab. The code is as follows:

```

const http = require('http');
//http.createServer(); // which actually creates a server
//as an argument it takes callback func
//request and response object
const server = http.createServer((req, res) => {
  console.log('request made!');
});
server.listen(3000, 'localhost', () => {
  console.log('listening for requests on port 3000');
});
  
```

The left sidebar shows a project structure with files like 'modules.js', 'files.js', 'global.js', 'people.js', 'streams.js', and 'test.js'. The bottom status bar shows the path 'C:\Users\Shree Ganraj\Comp\Documents\node-crash-course>' and the terminal output.

## //localhost:

local host is like domain name on the web

eg: google.com

but localhost has a special IP address as to 127.0.0.1

09:21



## Port Number:

specific channel on ur computer (jyala server sobt communicate karayachy)

through which internet communication is made

....localhost nanntr we type port number..

10:42

# Port Numbers

- Port number are like 'doors' into a computer



File Edit Selection View Go Run Terminal Help serverjs - node-crash-course - Visual Studio Code

EXPLORER NODE-CRASH-COURSE docs files.js global.js modules.js people.js streams.js test.js server.js

```
JS server.js > server.listen('localhost') callback
1 const http = require('http');
2
3 //http.createServer(); // which actually creates a server
4
5 //as an argument it takes callback func
6 //request and response object
7 //get, post request and also it gives where that request belong to
8 const server = http.createServer((req, res)=>{
9   console.log('request made!');
10 });
11
12 server.listen(3000, 'localhost', ()=>{
13   console.log('listening for requests on port 3000');
14 });

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
```

Excel format. There are dataset examples for property insurance data, food sales records, work orders, and other Excel datasets.

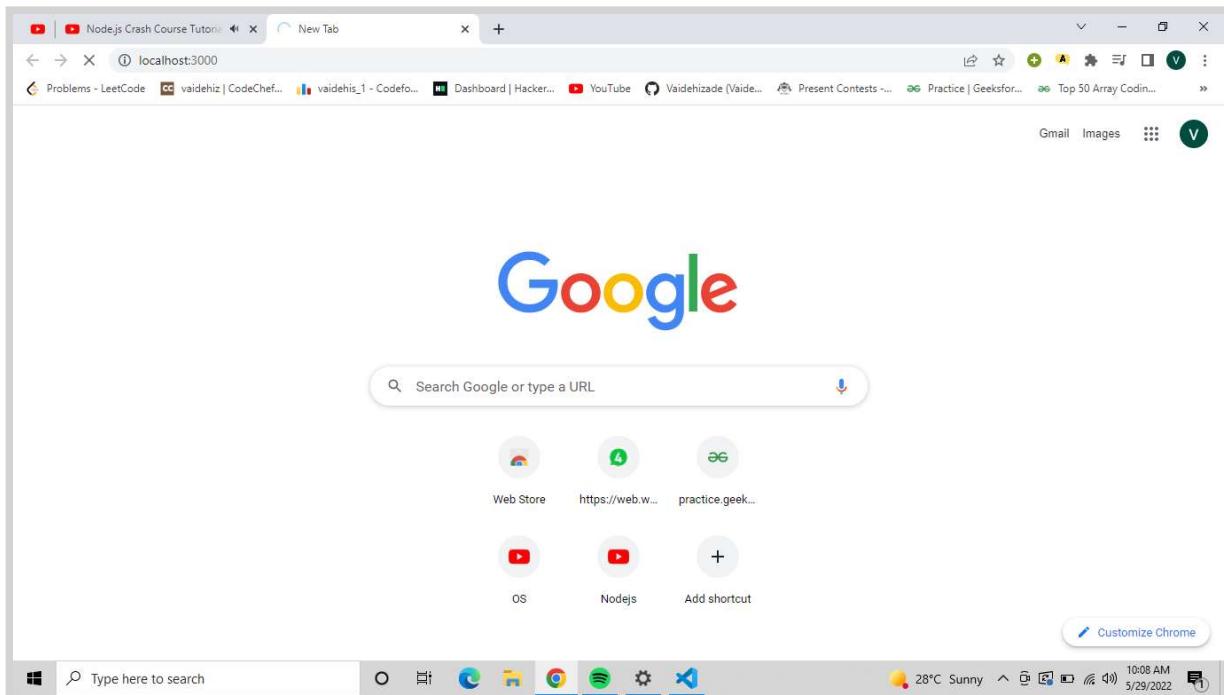
Note: You may not resell this sample data, in whole or in part, or include it in paid product. This page has Microsoft Excel sample datasets that you can freely use for testing, Excel training and demos, student practice, and other learning activities. There is a table with office supply sales sample data, to copy and paste into your Excel workbook. Or download one of the many sample data files in Excel format. There are dataset examples for property insurance data, food sales records, work orders, and other Excel datasets.

Note: You may not resell this sample data, in whole or in part, or include it in paid product  
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node streams  
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node streams  
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node server  
listening for requests on port 3000

Ln 13, Col 56 Spaces: 4 UTF-8 CRLF ↗ JavaScript ⚡ Go Live ✅ Prettier 🔍 10:07 AM 28°C Sunny 5/29/2022

Type here to search

ata applyala server la request karaychiy

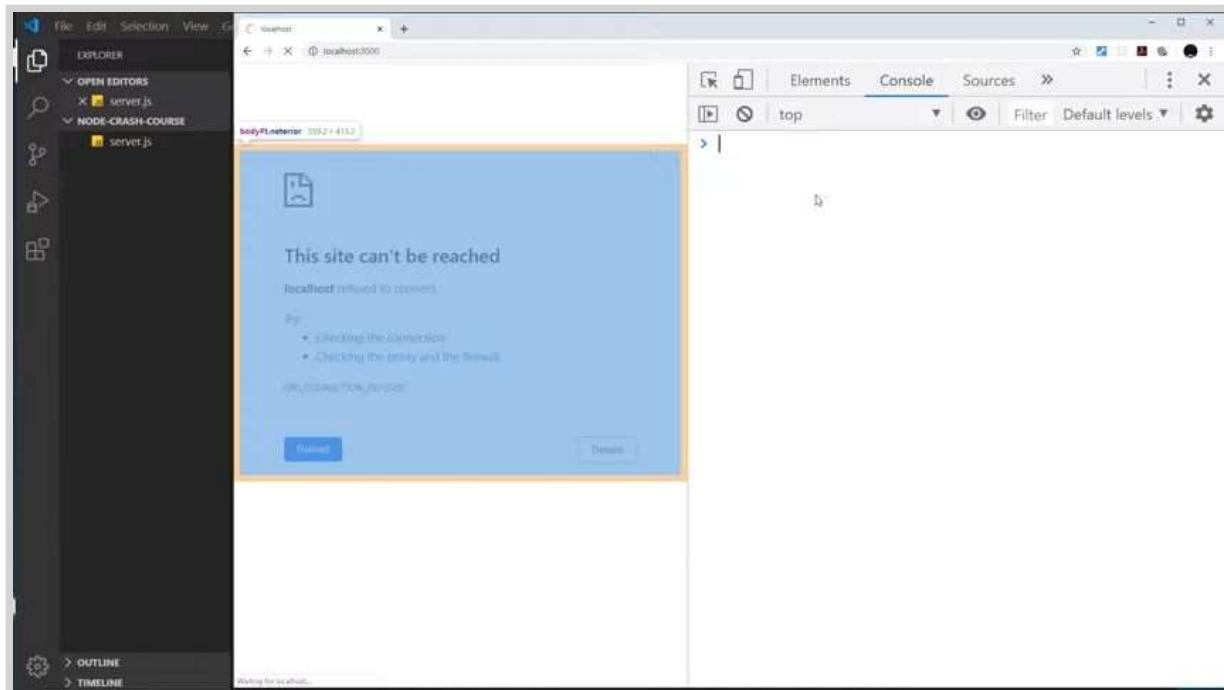


**waiting for server to respond ...**

**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**



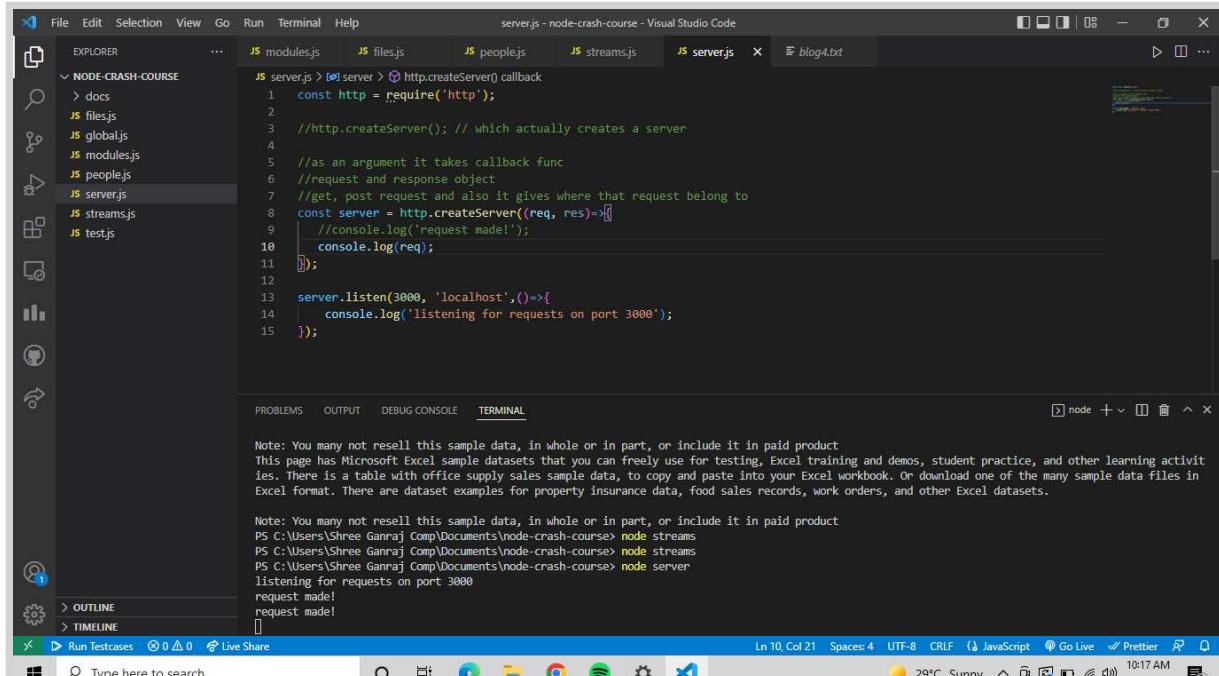
**how we can get response back...?**

**Video4:**

## Requests and Responses:

=> so far we have created our server

request object contains info about request send by user



```

File Edit Selection View Go Run Terminal Help
server.js - node-crash-course - Visual Studio Code
EXPLORER JS modules.js JS files.js JS people.js JS streams.js JS server.js x blog4.txt
NODE-CRASH-COURSE > docs > server > http.createServer() callback
1 const http = require('http');
2
3 //http.createServer(); // which actually creates a server
4
5 //as an argument it takes callback func
6 //request and response object
7 //get, post request and also it gives where that request belong to
8 const server = http.createServer((req, res)=>{
9   //console.log('request made!');
10  console.log(req);
11 });
12
13 server.listen(3000, 'localhost', ()=>{
14   console.log('listening for requests on port 3000');
15 });

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

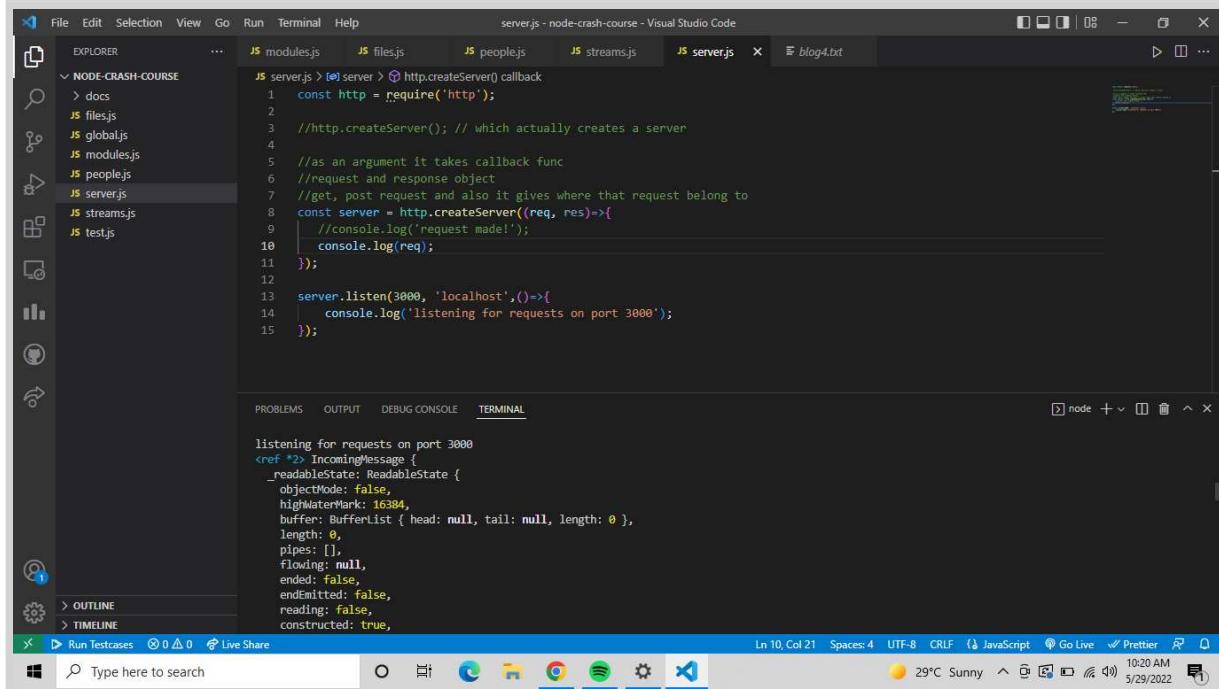
Note: You may not resell this sample data, in whole or in part, or include it in paid product  
This page has Microsoft Excel sample datasets that you can freely use for testing, Excel training and demos, student practice, and other learning activities. There is a table with office supply sales sample data, to copy and paste into your Excel workbook. Or download one of the many sample data files in Excel format. There are dataset examples for property insurance data, food sales records, work orders, and other Excel datasets.

Type here to search

File Edit Selection View Go Run Terminal Help server.js - node-crash-course - Visual Studio Code

PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node streams  
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node streams  
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node server  
listening for requests on port 3000  
request made!  
request made!

Ln 10, Col 21 Spaces: 4 UTF-8 CRLF JavaScript Go Live Prettier 10:17 AM 29°C Sunny 5/29/2022



```

File Edit Selection View Go Run Terminal Help server.js - node-crash-course - Visual Studio Code
EXPLORER JS modules.js JS files.js JS people.js JS streams.js JS server.js x blog4.txt
NODE-CRASH-COURSE > docs > server > http.createServer() callback
1 const http = require('http');
2
3 //http.createServer(); // which actually creates a server
4
5 //as an argument it takes callback func
6 //request and response object
7 //get, post request and also it gives where that request belong to
8 const server = http.createServer((req, res)=>{
9   //console.log('request made!');
10  console.log(req);
11 });
12
13 server.listen(3000, 'localhost', ()=>{
14   console.log('listening for requests on port 3000');
15 });

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

listening for requests on port 3000  
<ref>>> IncomingMessage {  
 \_readableState: ReadableState {  
 objectMode: false,  
 highWaterMark: 16384,  
 buffer: BufferList { head: null, tail: null, length: 0 },  
 length: 0,  
 pipes: [],  
 flowing: null,  
 ended: false,  
 endEmitted: false,  
 reading: false,  
 constructed: true,

Type here to search

File Edit Selection View Go Run Terminal Help server.js - node-crash-course - Visual Studio Code

Ln 10, Col 21 Spaces: 4 UTF-8 CRLF JavaScript Go Live Prettier 10:20 AM 29°C Sunny 5/29/2022



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



```

File Edit Selection View Go Run Terminal Help
serverjs - node-crash-course - Visual Studio Code
EXPLORER JS modules.js JS files.js JS peoplejs JS streams.js JS server.js x blog4.txt
NODE-CRASH-COURSE
> docs
JS files.js
JS global.js
JS modules.js
JS people.js
JS server.js
JS streams.js
JS test.js
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
TERMINAL
:accept-encoding': 'gzip, deflate, br',
:accept-language': 'en-US,en;q=0.9'
},
[Symbol(kHeadersCount)]: 30,
[Symbol(kTrailers)]: null,
[Symbol(kTrailersCount)]: 0,
[Symbol(RequestTimeout)]: undefined
}
P5 C:\Users\Shree Ganraj Comp\Documents\node-crash-course> node server
listening for requests on port 3000
/ GET
/about GET
Ln 10, Col 37 Spaces: 4 UTF-8 CRLF Go Live Prettier
29°C Sunny 10:28 AM 5/29/2022
Run Tests Live Share Type here to search

```

## #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

```

NODE-CRASH-COURSE
server.js
3 const server = http.createServer((req, res) => {
4   console.log(req.url, req.method);
5
6   // set header content type
7   res.setHeader('Content-Type', 'text/plain');
8
9   res.write('hello, ninjas');
10  res.end();
11 });

```

hello Vaidehi

localhost:3000

Request URL: http://localhost:3000/

Request Method: GET

Status Code: 200 OK

Remote Address: 127.0.0.1:3000

Referrer Policy: strict-origin-when-cross-origin

Content-Type: text/plain

2 requests | 360 B transferred

Console | What's New

Highlights from the Chrome 102 update

New preview feature: Performance insights panel  
Get actionable and use-case-driven insights on your website's performance.

New emulation shortcuts in the Styles pane  
Emulate light themes, dark themes, and more with the emulation shortcuts.

new

hello Vaidehi

localhost:3000

Request URL: http://localhost:3000/

Request Method: GET

Status Code: 200 OK

Remote Address: 127.0.0.1:3000

Referrer Policy: strict-origin-when-cross-origin

Content-Type: text/plain

Date: Sun, 29 May 2022 05:04:54 GMT

Keep-Alive: timeout=5

Transfer-Encoding: chunked

2 requests | 360 B transferred

Console | What's New

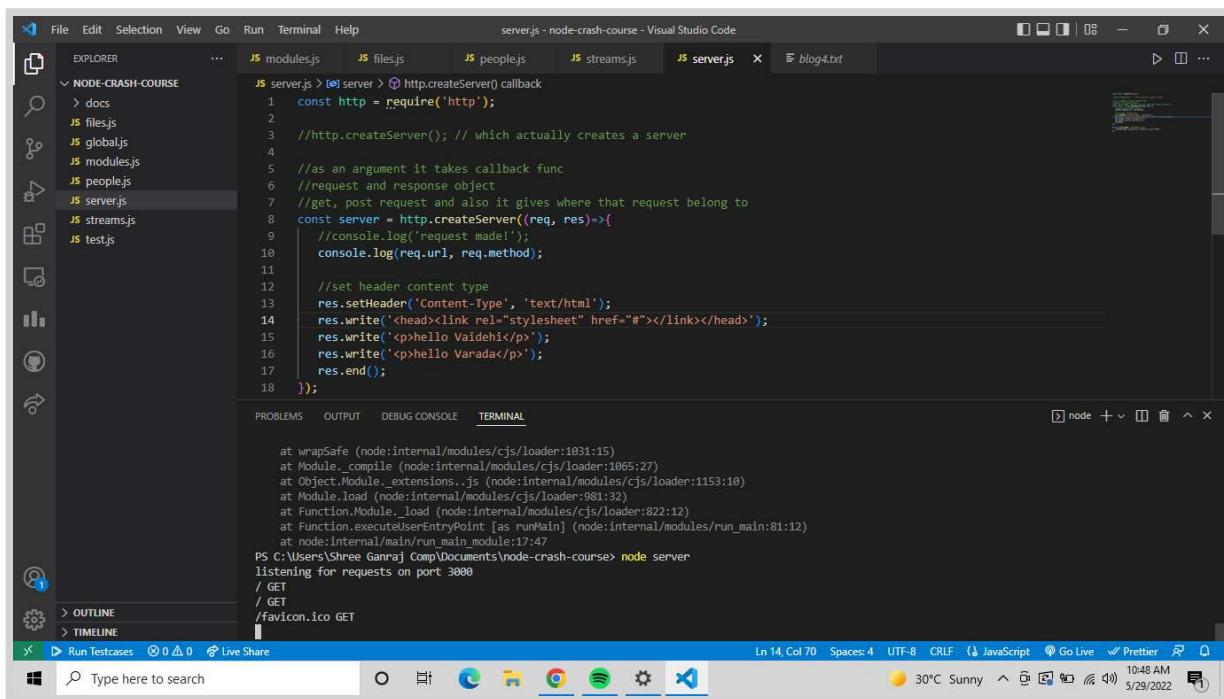
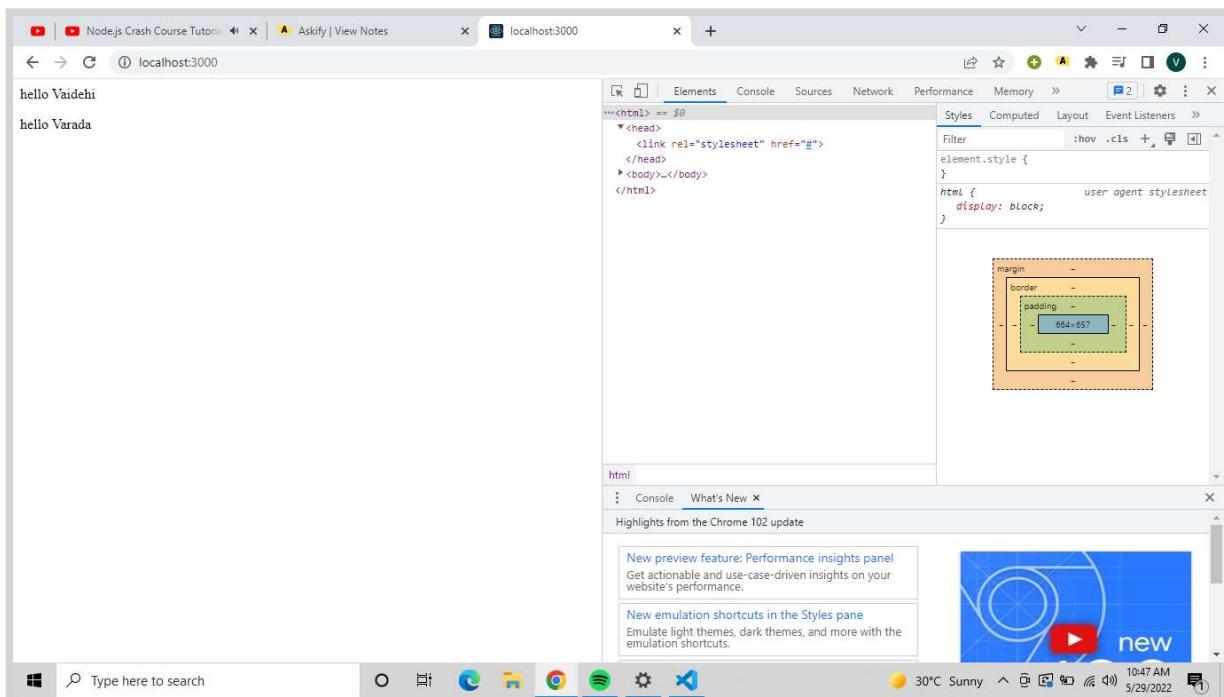
Highlights from the Chrome 102 update

New preview feature: Performance insights panel  
Get actionable and use-case-driven insights on your website's performance.

New emulation shortcuts in the Styles pane  
Emulate light themes, dark themes, and more with the emulation shortcuts.

new

**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 ...**

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left lists files in the 'NODE-CRASH-COURSE' folder, including 'index.html', 'modules.js', 'files.js', 'people.js', 'streams.js', and 'server.js'. The 'index.html' file is open in the main editor, displaying the following code:

```

<!DOCTYPE html>
<html>
  <head><title>Node.js Crash Course</title></head>
  <body>
    <h1>My First Heading</h1>
    <p>My first paragraph.</p>
  </body>
</html>

```

The terminal tab at the bottom shows the command line output:

```

PS C:\Users\Shree Ganraj\Comp\Documents\node-crash-course> node server
listening for requests on port 3000
/ GET
/ favicon.ico GET

```

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left lists files in the 'NODE-CRASH-COURSE' folder, including 'index.html', 'modules.js', 'files.js', 'people.js', 'streams.js', and 'server.js'. The 'server.js' file is open in the main editor, displaying the following code:

```

const http = require('http');
const fs = require('fs');

// http.createServer().listen() // which actually creates a server

```

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left lists files in the 'NODE-CRASH-COURSE' folder, including 'index.html', 'modules.js', 'files.js', 'people.js', 'streams.js', and 'server.js'. The 'server.js' file is open in the main editor, displaying the following code:

```

const http = require('http');
const fs = require('fs');

http.createServer((req, res) => {
  if (req.url === '/') {
    fs.readFile('./views/index.html', (err, data) => {
      if (err) {
        console.log(err);
        res.end();
      } else {
        res.write(data);
        res.end();
      }
    });
  }
}).listen(3000, 'localhost', () => {
  console.log('Listening for requests on port 3000');
})

```

The terminal tab at the bottom shows the command line output:

```

PS C:\Users\Shree Ganraj\Comp\Documents\node-crash-course> node server
listening for requests on port 3000
/ GET
/ favicon.ico GET
PS C:\Users\Shree Ganraj\Comp\Documents\node-crash-course> node server
listening for requests on port 3000
/ GET
/ favicon.ico GET

```

**more easy way=>**

12:34



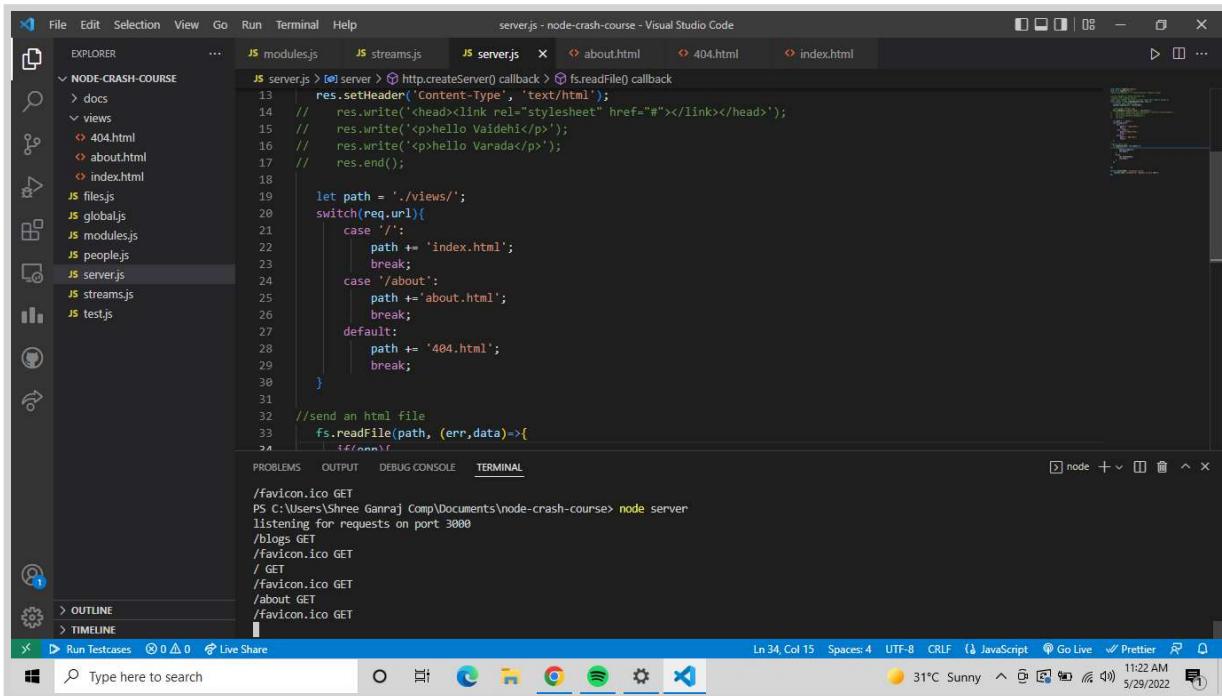
```

10 // send HTML file
11   fs.readFile('./views/index.html', (err, data) => {
12     if (err) {
13       console.log(err);
14       res.end();
15     } else {
16       //res.write(data);
17       res.end(data);
18     }
19   })

```

## #Basic Routing:

### with different url , send different html pages



The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows files in the "NODE-CRASH-COURSE" folder, including modules.js, streams.js, server.js, about.html, 404.html, and index.html.
- Code Editor:** The active file is server.js, which contains the following code:
 

```

res.setHeader('Content-Type', 'text/html');
// res.write(<head><link rel="stylesheet" href="#"></link></head>');
// res.write('<p>Hello Vaidehi</p>');
// res.end();

let path = './views/';
switch(req.url){
  case '/':
    path += 'index.html';
    break;
  case '/about':
    path += 'about.html';
    break;
  default:
    path += '404.html';
    break;
}

//send an html file
fs.readFile(path, (err,data)=>{
  if(err)
    res.end();
})

```
- Terminal:** Shows command-line output from running the node server script, including requests for favicon.ico and blogs.
- Status Bar:** Shows the current time (11:22 AM), date (5/29/2022), weather (31°C Sunny), and other system information.

## #Status Codes:

19:01



The slide has a dark blue background and features the following content:

## Status Codes

- Status codes describe the type of response sent to the browser

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      18 let path = './views/';
JS files.js    19 switch(req.url){
JS global.js   20     case '/':
JS modules.js 21         path += 'index.html';
JS people.js   22         res.statusCode = 200;
JS server.js   23         break;
JS streams.js  24     case '/about':
JS test.js     25         path += 'about.html';
JS                 26         res.statusCode = 200;
JS                 27         break;
JS                 28     default:
JS                 29         path += '404.html';
JS                 30         res.statusCode = 404;
JS                 31         break;
JS                 32
JS                 33 }
```

The screenshot shows a browser window with two tabs: "Node.js Crash Course Tutorial #4" and "Askify | View Notes". The main content area displays the text "My First Heading" and "My first paragraph.". The developer tools are open, specifically the Network tab. A request for "localhost" resulted in a 200 status code. A request for "favicon.ico" resulted in a 404 status code. The Network tab includes a waterfall chart and a table of requests.

Name	Status	Type	Initiator	Size	Time	Waterfall
localhost	200	document	Other	331 B	351 ms	
favicon.ico	404	text/html	Other	323 B	8 ms	

The screenshot shows a browser window with two tabs: "Node.js Crash Course Tutorial #4" and "Askify | View Notes". The main content area displays the text "Oops...404! page does not exist". The developer tools are open, specifically the Network tab. A request for "blogs" resulted in a 404 status code. The Network tab includes a waterfall chart and a table of requests.

Name	Status	Type	Initiator	Size	Time	Waterfall
blogs	404	document	Other	323 B	32 ms	

**#How to do redirects:**  
**i want to redirect from about to about -me**

```

File Edit Selection View Go Run Terminal Help
serverjs - node-crash-course - Visual Studio Code
EXPLORER JS modules.js JS streams.js JS serverjs X about.html 404.html index.html
NODE-CRASH-COURSE
docs
views
404.html
about.html
index.html
JS files.js
JS global.js
JS modules.js
JS people.js
JS server.js
JS streams.js
JS test.js
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Listening for requests on port 3000
/about GET
/about-me GET
[Error: EISDIR: illegal operation on a directory, read] {
  errno: -4068,
  code: 'EISDIR',
  syscall: 'read'
}
about GET
Ln 32, Col 23 Spaces: 4 UTF-8 CRLF Go Live Prettier
32°C Sunny 11:33 AM 5/29/2022

```

so if khup mothe pages asle tr express(thirty party ) aplyal  
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...

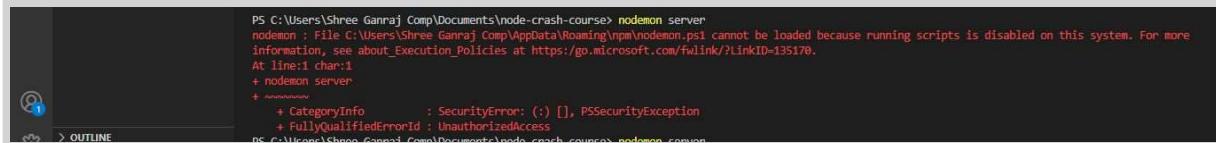
nodemon mule automatically server madhe changes save hot  
jatat ani te ...chalat rahta

It automatically restarts the server

//we re installing package globally:

npm install -g nodemon

I had got an error:



```
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> nodemon server
nodemon : File C:\Users\Shree Ganraj Comp\AppData\Roaming\npm\nodemon.ps1 cannot be loaded because running scripts is disabled on this system. For more
information, see about_Execution_Policies at https://go.microsoft.com/fwlink/?LinkId=135170.
At line:1 char:1
+ nodemon server
+ ~~~~~
+ CategoryInfo          : SecurityError: () [], PSSecurityException
+ FullyQualifiedErrorId : UnauthorizedAccess
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course>
```

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:

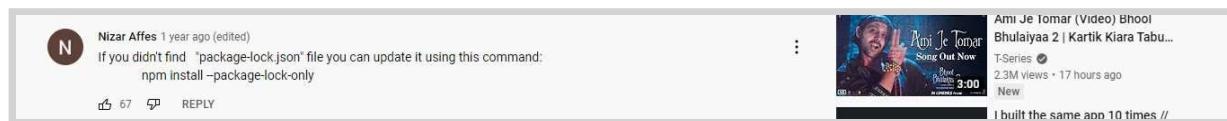
=>Package file is json file....

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)



Nizar Afes 1 year ago (edited)  
If you didn't find "package-lock.json" file you can update it using this command:  
npm install --package-lock-only

67 REPLY

Ami Je Tomar (Video) Bhool Bhulaiya 2 | Kartik Kiara Tabu...  
T-Series 2.3M views • 17 hours ago  
New

I built the same app 10 times //

#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)

```

File Edit Selection View Go Run Terminal Help
serverjs - node-crash-course - Visual Studio Code
EXPLORER JS modules.js JS server.js package.json
NODE-CRASH-COURSE ...
docs > node_modules
views
JS files.js
JS global.js
JS modules.js
{} package-lock.json
{} package.json
JS people.js
JS server.js
JS streams.js
JS test.js
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Is this OK? (yes) y
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> npm install --package-lock-only
up to date, audited 1 package in 384ms
found 0 vulnerabilities
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course> npm install lodash
added 1 package, and audited 2 packages in 3s
found 0 vulnerabilities
PS C:\Users\Shree Ganraj Comp\Documents\node-crash-course>
Ln 3, Col 29 Spaces: 4 UTF-8 CRLF Go Live Prettier
Run Testcases 0 △ 0 Live Share 32°C Sunny 12:29 PM 5/29/2022
Type here to search
O 🔍

```

lodash provides utility methods:

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

```

File Edit Selection View Go Run Terminal Help
serverjs - node-crash-course - Visual Studio Code
EXPLORER JS modules.js JS server.js package.json
NODE-CRASH-COURSE ...
docs > node_modules
views
JS files.js
JS global.js
JS modules.js
{} package-lock.json
{} package.json
JS people.js
JS server.js
JS streams.js
JS test.js
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
[nodemon] 2.0.16
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
listening for requests on port 3000
/about GET
8
/ GET
11
/ GET
15
Ln 14, Col 1 (53 selected) Spaces: 4 UTF-8 CRLF Go Live Prettier
Run Testcases 0 △ 0 Live Share 32°C Sunny 12:32 PM 5/29/2022
Type here to search
O 🔍

```

12:59

```

File Edit Selection View Go Run Terminal Help
server.js - node-crash-course - Visual Studio Code
EXPLORER OPEN EDITORS server.js
NODE-CRASH-COURSE
node_modules\...
views
404.html
about.html
index.html
package-lock.json
package.json
server.js
server.js > (W) server > @ http.createServer() callback > (E) greet > @ _once() callback
1 const http = require('http');
2 const fs = require('fs');
3 const _ = require('lodash');
4
5 const server = http.createServer((req, res) => {
6
7   // lodash
8   const num = _.random(0, 20);
9   console.log(num);
10
11  const greet = _.once(() => {
12    console.log('hello');
13  });
14
15  greet();
16  greet();
17
18  // set header content type
19  res.setHeader('Content-Type', 'text/html');

18
12
[nodemon] restarting due to changes...
[nodemon] starting 'node server.js'
listening for requests on port 3000
1
hello

```

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

=>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..

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure with files like modules.js, server.js, app.js, files.js, global.js, modules.js, package-lock.json, package.json, people.js, server.js, streams.js, and test.js.
- Code Editor:** The app.js file is open, displaying the code provided at the top of the page. It defines an Express app, sets up a root route to return 'Hello World', and listens on port 3000.
- Terminal:** The terminal shows the output of running the application with nodemon. It includes the command used (nodemon app), the node version (node v16.13.1), and the log message: '[nodemon] 2.0.16 [nodemon] to restart at any time, enter `rs` [nodemon] watching path(s): \*.\* [nodemon] watching extensions: js,mjs,json [nodemon] starting `node app.js` [nodemon] restarting due to changes... [nodemon] starting `node app.js`'.
- Status Bar:** Shows the current file (app.js), line 13, column 54, spaces 4, and other system information like weather (34°C Mostly sunny) and date (5/29/2022).

//Routing HTML:

## => how to send html file using express

```

File Edit Selection View Go Run Terminal Help
app.js - node-crash-course - Visual Studio Code
EXPLORER JS modules.js JS server.js JS app.js package.json
JS app.js > app.get('about') callback
8
9 //now how we actually respond to those requests
10
11 // '/'=> root of the domain
12 // req, res object
13 //this is the code how we can respond to specific URL
14 app.get('/', (req, res)=>{
15   //res.send('<p>home page</p>');
16   res.sendFile('./views/index.html', {root: __dirname});
17 });
18
19 app.get('/about', (req, res)=>{
20   //res.send('<p>about page</p>');
21   res.sendFile('./views/about.html', {root: __dirname});
22 });

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```

[nodemon] restarting due to changes...
[nodemon] starting "node app.js"
[nodemon] restarting due to changes...
[nodemon] starting "node app.js"
[nodemon] restarting due to changes...
[nodemon] starting "node app.js"
[nodemon] restarting due to changes...
[nodemon] starting "node app.js"
[nodemon] restarting due to changes...
[nodemon] starting "node app.js"
[nodemon] restarting due to changes...
[nodemon] starting "node app.js"
[nodemon] restarting due to changes...
[nodemon] starting "node app.js"

```

Ln 21, Col 59 Spaces: 4 UTF-8 CRLF ↳ JavaScript ⚡ Go Live ✅ Prettier ⌂ 4:05 PM 5/29/2022

## //Redirects and 404Pages=>in express

```

File Edit Selection View Go Run Terminal Help
app.js - node-crash-course - Visual Studio Code
EXPLORER JS modules.js JS server.js JS app.js about.html
JS app.js ...
11 // '/'=> root of the domain
12 // req, res object
13 //this is the code how we can respond to specific URL
14 app.get('/', (req, res)=>{
15   //res.send('<p>home page</p>');
16   res.sendFile('./views/index.html', {root: __dirname});
17 });
18
19 app.get('/about', (req, res)=>{
20   //res.send('<p>about page</p>');
21   res.sendFile('./views/about.html', {root: __dirname});
22 });
23
24 //redirects
25 app.get('/about-me', (req,res)=>{
26   res.redirect('../about');
27 });

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```

[nodemon] restarting due to changes...
[nodemon] starting "node app.js"
[nodemon] restarting due to changes...
[nodemon] starting "node app.js"
[nodemon] restarting due to changes...
[nodemon] starting "node app.js"
[nodemon] restarting due to changes...
[nodemon] starting "node app.js"
[nodemon] restarting due to changes...
[nodemon] starting "node app.js"
[nodemon] restarting due to changes...
[nodemon] starting "node app.js"

```

Ln 25, Col 10 Spaces: 4 UTF-8 CRLF ↳ JavaScript ⚡ Go Live ✅ Prettier ⌂ 4:16 PM 5/29/2022

## //express automatically sets the status code

//404 Pages=>

```

File Edit Selection View Go Run Terminal Help
appjs - node-crash-course - Visual Studio Code
EXPLORER JS modules.js JS server.js JS app.js X about.html
NODE-CRASH-COURSE
docs
node_modules
views
404.html
about.html
index.html
JS app.js
JS files.js
JS global.js
JS package-lock.json
JS package.json
JS people.js
JS server.js
JS streams.js
JS test.js

14
15 // the below are get handlers
16 app.get('/', (req, res)=>{
17   //res.sendFile('../views/index.html', {root: __dirname});
18   res.sendFile('../views/index.html', {root: __dirname});
19 });
20
21 app.get('/about', (req, res)=>{
22   //res.sendFile('../about page</p>');
23   res.sendFile('../views/about.html', {root: __dirname});
24 });
25
26 //redirects
27 app.get('/about-me', (req,res)=>{
28   res.redirect('../about');
29 });
30
31 //app.use() // this function is used to create middleware and
32 app.use((req,res)=>{
33   res.status(404).sendFile('../views/404.html',{root: __dirname});
34 });
35 //use function is going to fire every single request coming in but only if the request reaches this point in the code

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

[nodemon] starting `node app.js`  
[nodemon] restarting due to changes...  
[nodemon] starting `node app.js`  
[nodemon] restarting due to changes...  
[nodemon] starting `node app.js`

Ln 33, Col 67 Spaces: 4 UTF-8 CRLF ↗ JavaScript Go Live Prettier 4:27 PM 5/29/2022

Type here to search

**//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

```

File Edit Selection View Go Run Terminal Help
appjs - node-crash-course - Visual Studio Code
EXPLORER JS modules.js JS server.js JS app.js X about.ejs 404.ejs index.ejs
JS app.js > app.use() callback.
22 // the below are get handlers
23 app.get('/', (req, res)=>{
24   //res.send('<p>home page</p>');
25   //res.sendFile('./views/index.html', {root: __dirname});
26   res.render('index');
27 });
28
29 app.get('/about', (req, res)=>{
30   //res.send('<p>about page</p>');
31   //res.sendFile('./views/about.html', {root: __dirname});
32   res.render('about');
33 });
34
35 //redirects
36 app.get('/about-me', (req,res)=>{
37   res.redirect('/about');
38 });
39
40 //app.use() // this function is used to create middleware and
41 app.use((req,res)=>{
42   //res.status(404).sendFile('./views/404.html',{root: __dirname});
43   res.status(404).render('404');
44 });
45 //use function is going to fire every single request coming in but only if the request reaches this point in the code

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

[nodemon] watching path(s): \*.\*  
[nodemon] watching extensions: js,mjs,json  
[nodemon] starting `node app.js`

Ln 43, Col 35 Spaces: 4 UTF-8 CRLF Go Live Prettier 6:32 PM 30°C Partly sunny 5/30/2022

Type here to search

```

File Edit Selection View Go Run Terminal Help
create.ejs - node-crash-course - Visual Studio Code
EXPLORER JS modules.js JS server.js JS app.js create.ejs X about.ejs 404.ejs index.ejs
JS app.js > create.ejs > HTML > body > div.create-blog-content > form > button
14   <div class="site-title">
15     <a href="/">h1>Blog Vaidehi</h1></a>
16     <p>Vaidehi's new site</p>
17   </div>
18   <ul>
19     <li><a href="/Blogs">Blogs</a></li>
20     <li><a href="/about">About</a></li>
21     <li><a href="/blogs/create">New Blogs</a></li>
22   </ul>
23
24 <div class="create-blog content">
25   <form>
26     <label for="title">Blog title:</label>
27     <input type="text" id="title" required>
28     <label for="snippet">Blog snippet:</label>
29     <input type="text" id="snippet" required>
30     <label for="body">Blog body:</label>
31     <textarea id="body" required></textarea>
32     <button>Submit</button>
33   </form>
34
35 </div>
36
37 </body>
38 </html>

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

[nodemon] starting `node app.js`  
[nodemon] restarting due to changes...  
[nodemon] starting `node app.js`

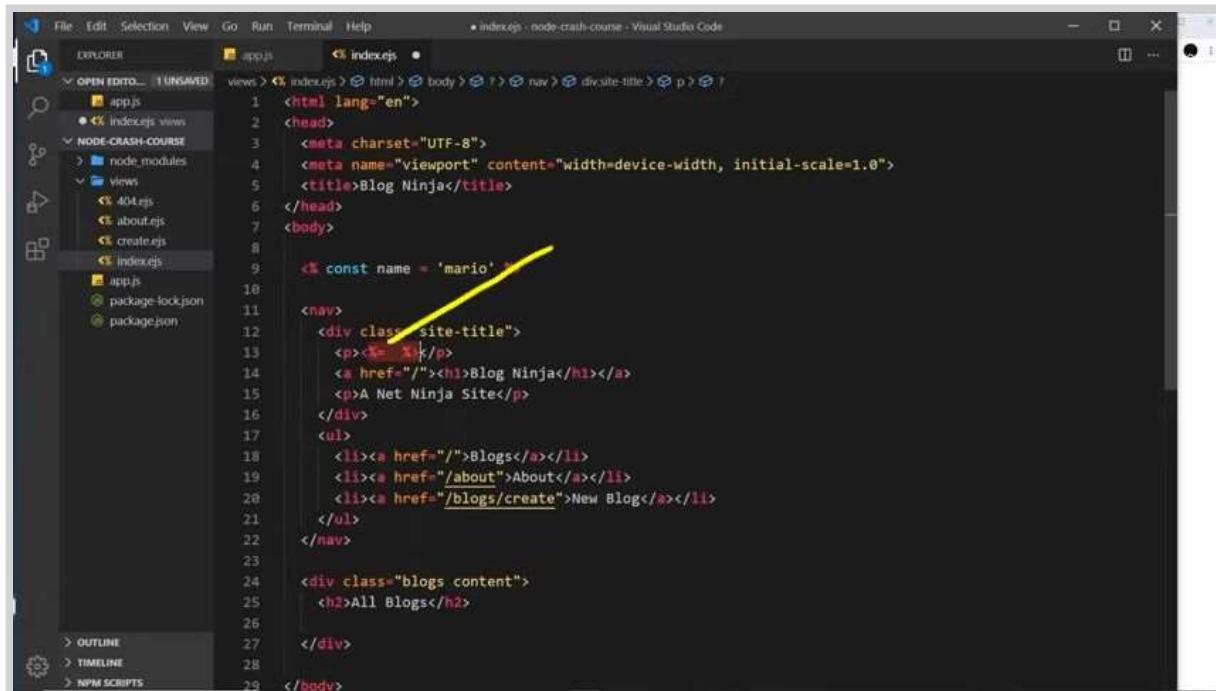
Ln 33, Col 40 Spaces: 4 UTF-8 CRLF Go Live Prettier 6:41 PM 30°C Partly sunny 5/30/2022

Type here to search

## Passing data into views:

inside % we can do any kind of JS(dynamic)  
server wr js yeil...

12:28



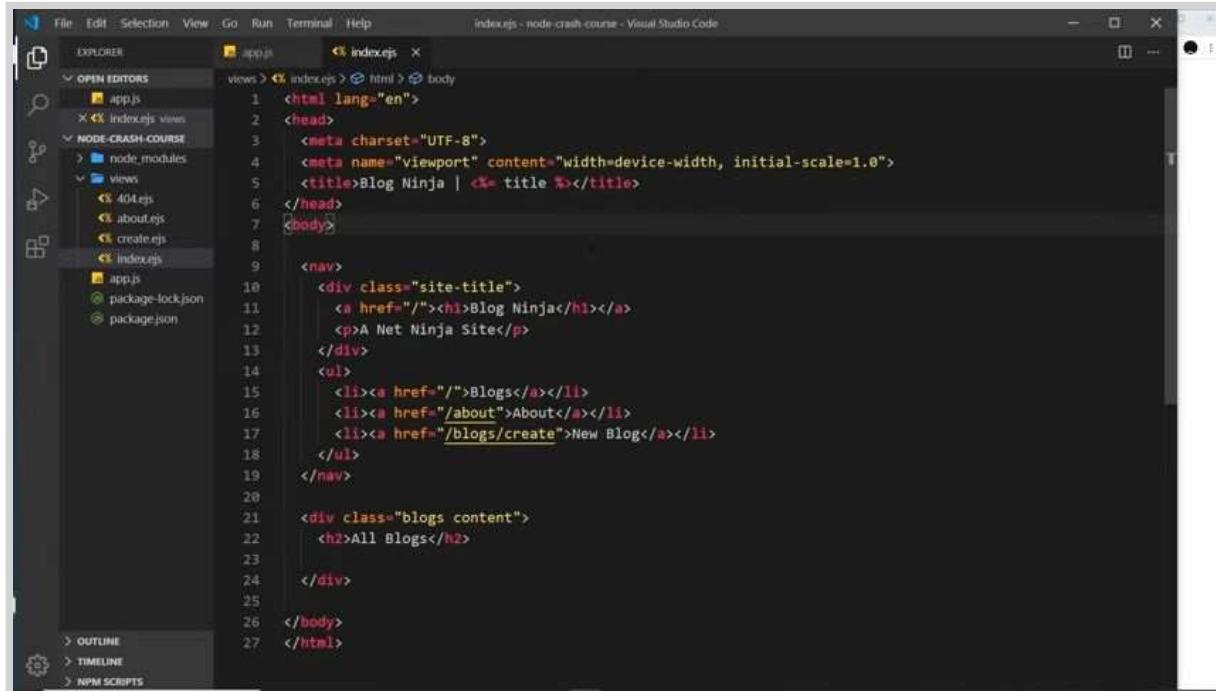
```

File Edit Selection View Go Run Terminal Help
OPEN EDITORS index.ejs
views > index.ejs > html > body > head > meta > title > p > 
1 <html lang="en">
2   <head>
3     <meta charset="UTF-8">
4     <meta name="viewport" content="width=device-width, initial-scale=1.0">
5     <title>Blog Ninja</title>
6   </head>
7   <body>
8     <% const name = 'mario' %>
9       <nav>
10         <div class="site-title">
11           <p>*&nbsp;*&nbsp;</p>
12           <a href="/"><h1>Blog Ninja</h1></a>
13           <p>A Net Ninja Site</p>
14         </div>
15         <ul>
16           <li><a href="/">Blogs</a></li>
17           <li><a href="/about">About</a></li>
18           <li><a href="/blogs/create">New Blog</a></li>
19         </ul>
20       </nav>
21       <div class="blogs content">
22         <h2>All Blogs</h2>
23       </div>
24     </body>
25   </html>

```

but we want to pass data ...  
 so we can pass it using render method  
 by making the object...

14:32



```

File Edit Selection View Go Run Terminal Help
OPEN EDITORS index.ejs
views > index.ejs > html > body
1 <html lang="en">
2   <head>
3     <meta charset="UTF-8">
4     <meta name="viewport" content="width=device-width, initial-scale=1.0">
5     <title>Blog Ninja | <% title %></title>
6   </head>
7   <body>
8     <nav>
9       <div class="site-title">
10         <a href="/"><h1>Blog Ninja</h1></a>
11         <p>A Net Ninja Site</p>
12       </div>
13       <ul>
14         <li><a href="/">Blogs</a></li>
15         <li><a href="/about">About</a></li>
16         <li><a href="/blogs/create">New Blog</a></li>
17       </ul>
18     </nav>
19     <div class="blogs content">
20       <h2>All Blogs</h2>
21     </div>
22   </body>
23 </html>

```

**if we want to output the array of blogs**

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure with files like `index.ejs`, `app.js`, `server.js`, and `modules.js`.
- Editor:** Displays the `index.ejs` file content:

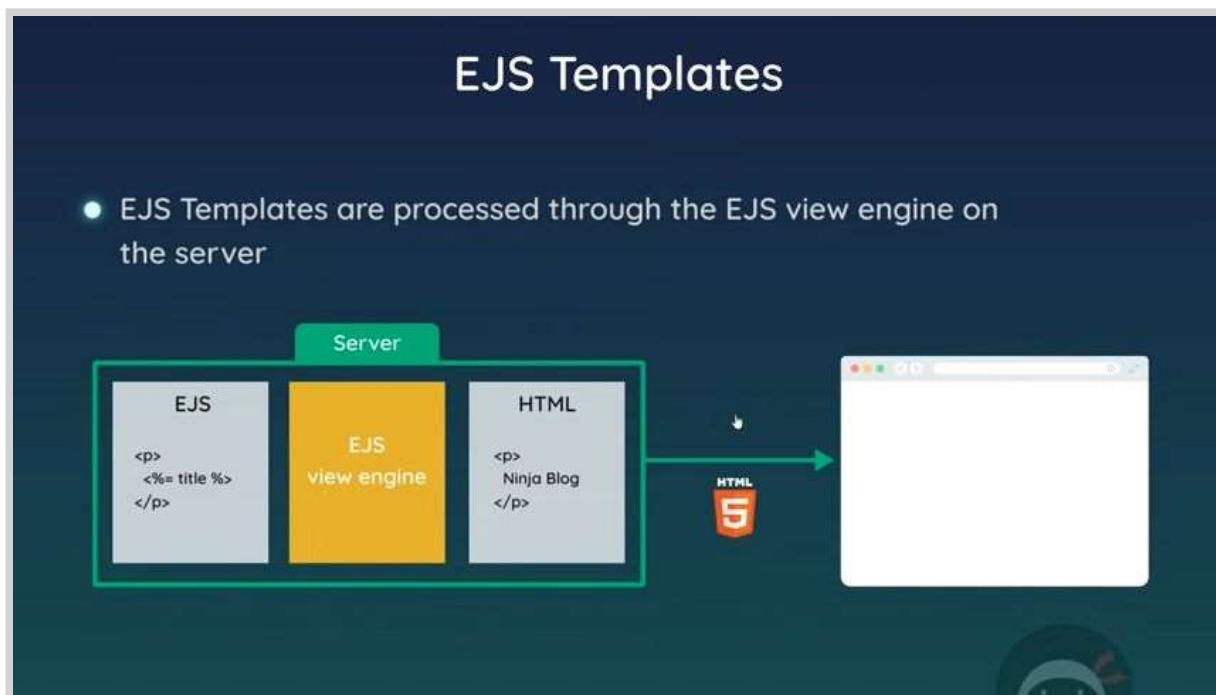
```
</div>
<ul>
  <li><a href="/">Blogs</a></li>
  <li><a href="/about">About</a></li>
  <li><a href="/blogs/create">New Blogs</a></li>
</ul>

<div class="blogs content">
  <h2>All Blogs</h2>

  <% if(blogs.length > 0){ %>
    <% blogs.forEach(blog =>{ %>
      <h3 class="title"><%= blog.title %></h3>
      <p class="snippet"><%= blog.snippet %></p>
      <% } %>
    <% } %>
  </div>
</body>
</html>
```
- Bottom Status Bar:** Shows the terminal output: `[nodemon] restarting due to changes...` repeated several times, indicating the application is restarting.
- Bottom Navigation:** Includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL, along with icons for Run Testcases, Live Share, and a search bar.
- Bottom Right:** Shows the node.js icon in the taskbar, indicating it is currently running.

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

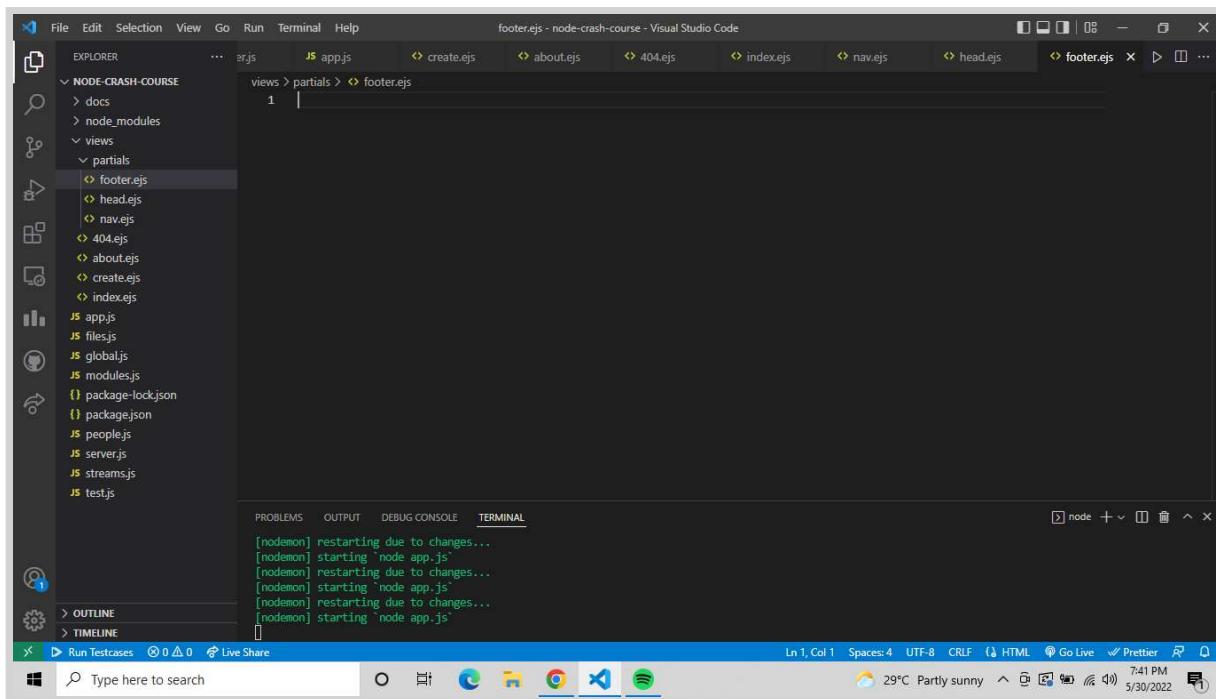


This whole process is called server side rendering..

## 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)



```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>Blog with Vaidehi %</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="">
  </head>
  <body>
    <% include('../partials/nav') %>

    <div class="not-found content">
      <h2>OOPS, page not found!</h2>
    </div>
  </body>
</html>

```

**TERMINAL**

```
[nodemon] restarting due to changes...
[nodemon] starting 'node app.js'
[nodemon] restarting due to changes...
[nodemon] starting 'node app.js'
[nodemon] restarting due to changes...
[nodemon] starting 'node app.js'
```

7:58 PM 5/30/2022

```

<!DOCTYPE html>
<html>
  <head>
    <% include('../partials/head.ejs') %>
  <body>
    <% include('../partials/nav.ejs') %>

    <div class="create-blog content">
      <form>
        <label for="title">Blog title:</label>
        <input type="text" id="title" required>
        <label for="snippet">Blog snippet:</label>
        <input type="text" id="snippet" required>
        <label for="body">Blog body:</label>
        <textarea id="body" required></textarea>
        <button type="submit">Submit</button>
      </form>
    </div>
    <% include('../partials/footer.ejs') %>
  </body>
</html>

```

**TERMINAL**

```
[nodemon] restarting due to changes...
[nodemon] starting 'node app.js'
[nodemon] restarting due to changes...
[nodemon] starting 'node app.js'
[nodemon] restarting due to changes...
[nodemon] starting 'node app.js'
```

8:09 PM 5/30/2022

//adding CSS:  
serving static files using middle wares:

## Video8: Middlewares:

00:03

# Middleware

- Code which runs (on the server) between getting a request and sending a response

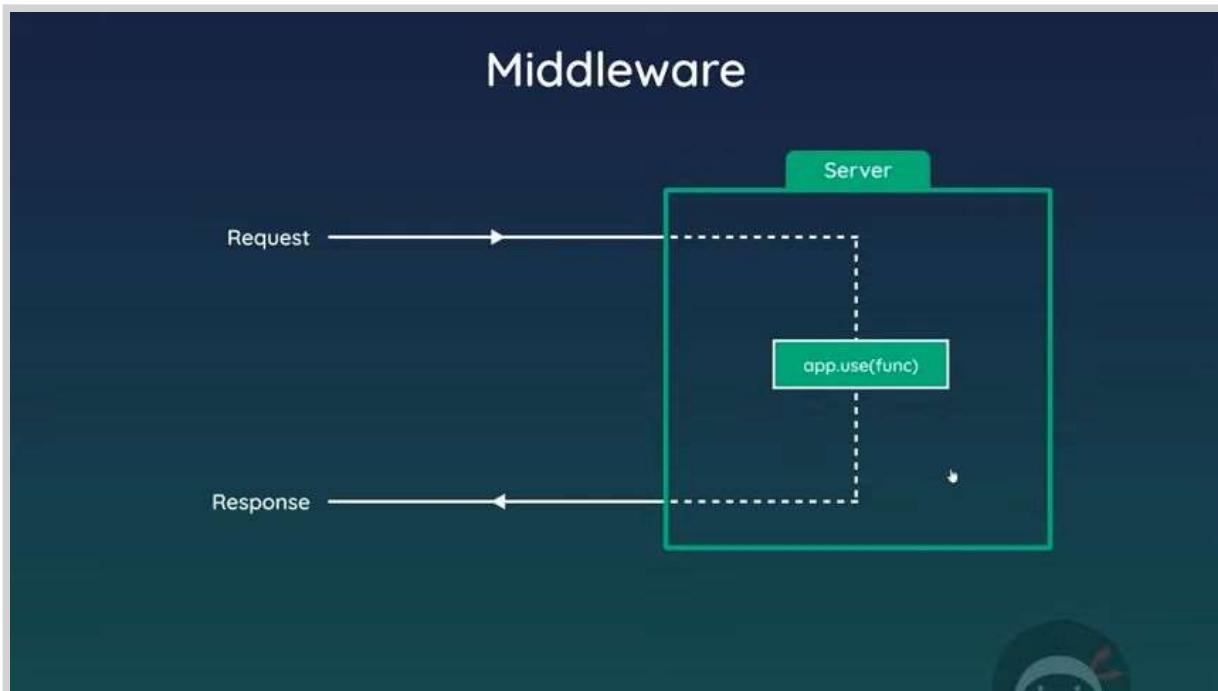
00:21

# Middleware

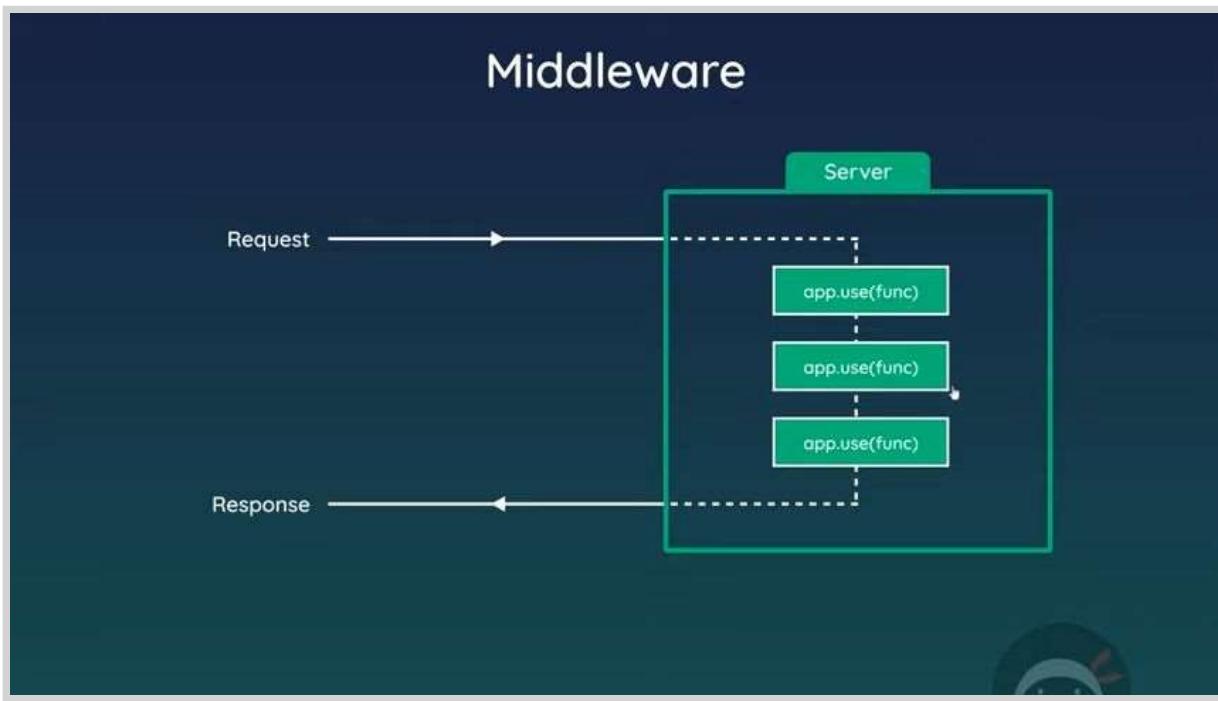


**use method is generally used as a middleware..**

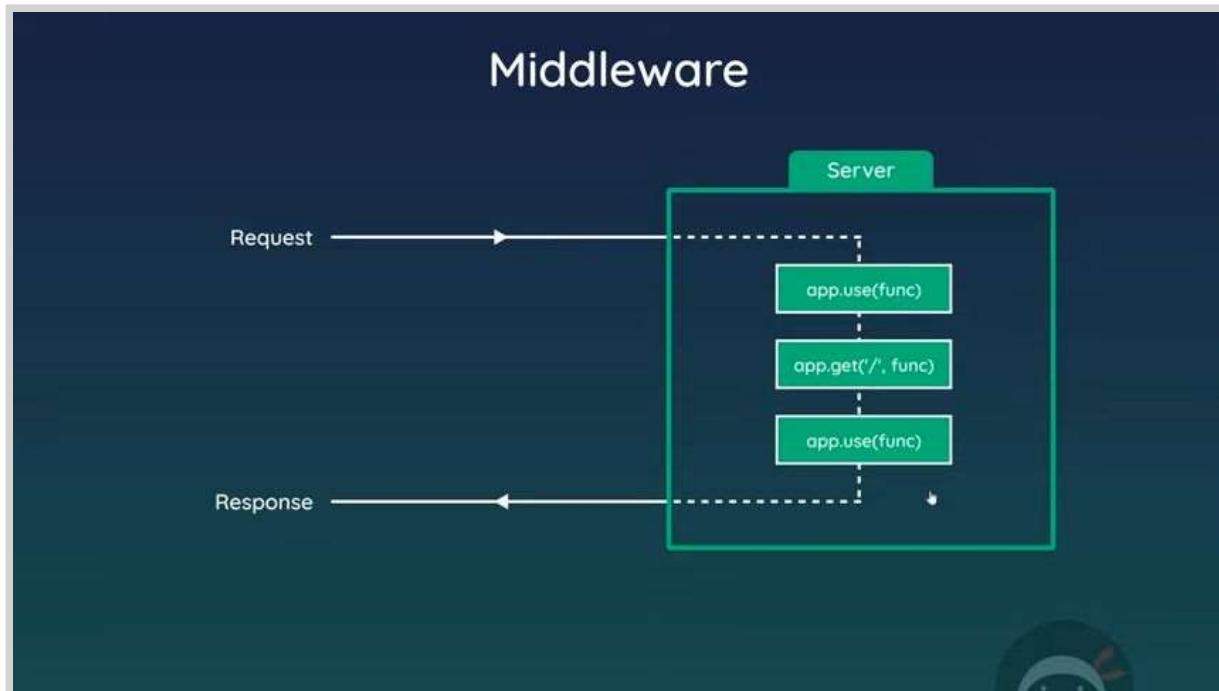
00:34



00:45



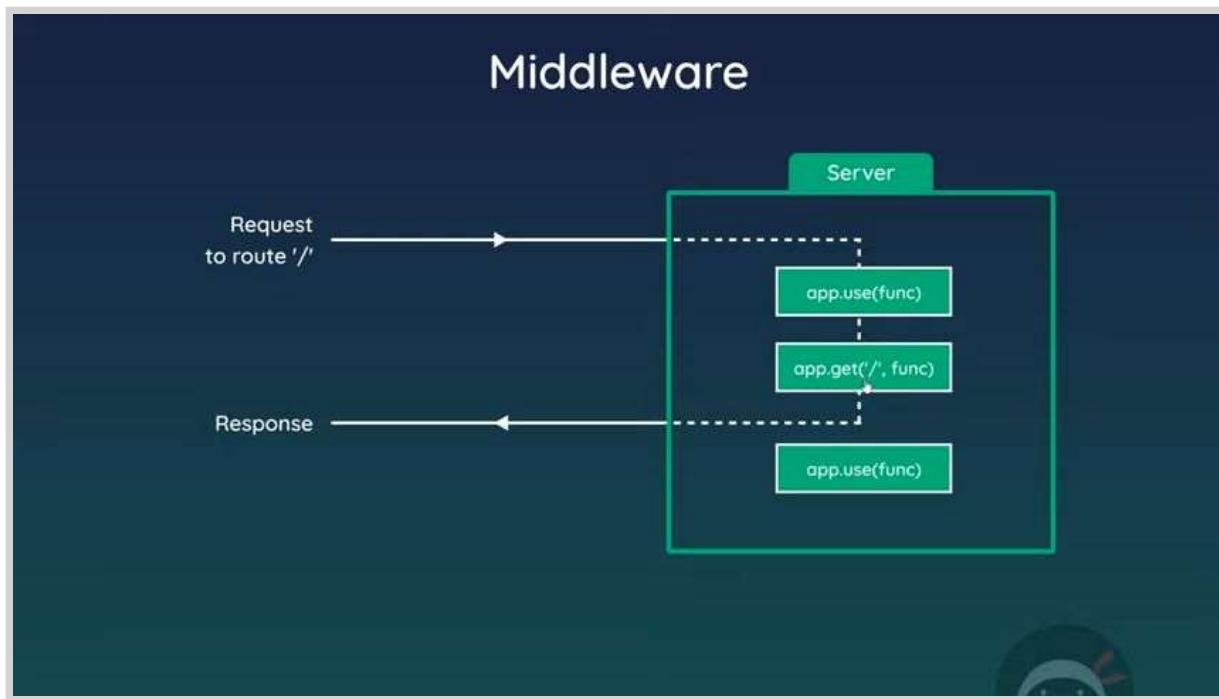
01:13



**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

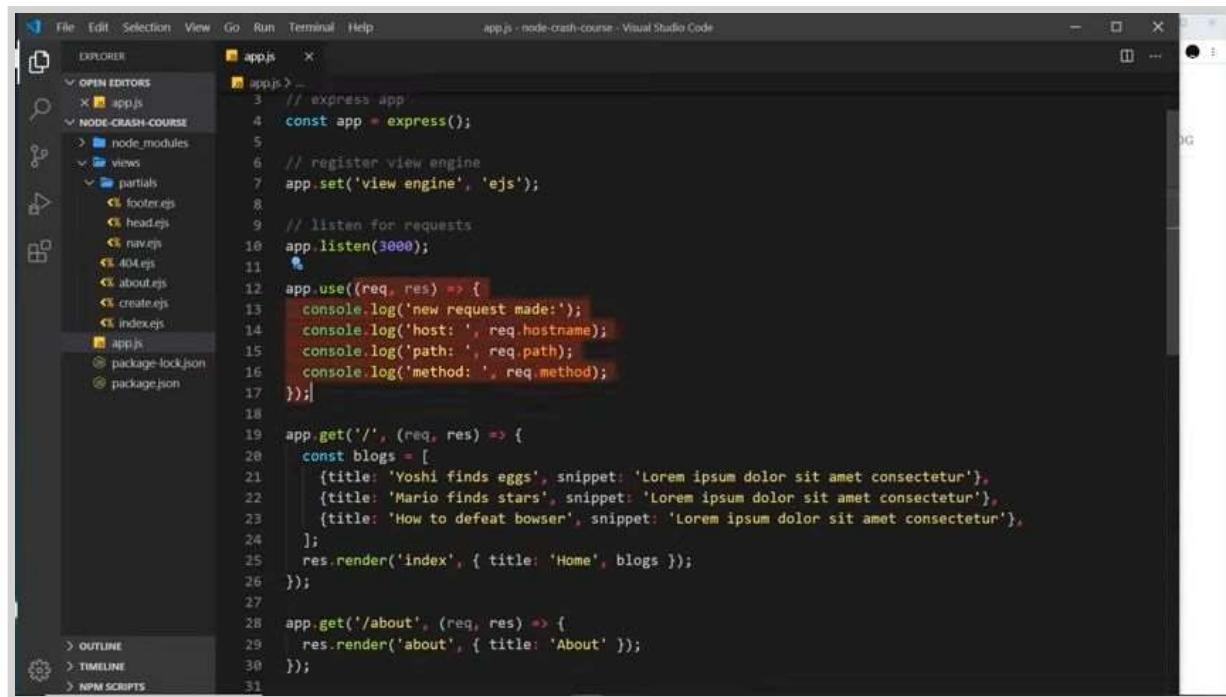
## Middleware Examples

- Logger middleware to log details of every request
- Authentication check middleware for protected routes
- Middleware to parse JSON data from requests
- Return 404 pages

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

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

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

04:42


The screenshot shows the Visual Studio Code interface with the file 'app.js' open. The code defines an Express application and adds a middleware function that logs request details to the console. It also includes route handlers for '/' and '/about'.

```

1 // express app
2 const app = express();
3
4 app.set('view engine', 'ejs');
5
6 app.listen(3000);
7
8 app.use((req, res) => {
9   console.log('new request made:');
10  console.log('host: ', req.hostname);
11  console.log('path: ', req.path);
12  console.log('method: ', req.method);
13});
14
15 app.get('/', (req, res) => {
16   const blogs = [
17     {title: 'Yoshi finds eggs', snippet: 'Lorem ipsum dolor sit amet consectetur'},
18     {title: 'Mario finds stars', snippet: 'Lorem ipsum dolor sit amet consectetur'},
19     {title: 'How to defeat bowser', snippet: 'Lorem ipsum dolor sit amet consectetur'}
20   ];
21   res.render('index', { title: 'Home', blogs });
22 });
23
24 app.get('/about', (req, res) => {
25   res.render('about', { title: 'About' });
26 });
27
28
29
30
31

```

The screenshot shows a development environment with the following components:

- VS Code Editor:** The main window displays the file `app.js` with the following code snippet:
 

```

14 app.listen(3000); //bydefault localhost
15
16 //now how we actually respond to those requests
17
18 // '/'> root of the domain
19 // req, res object
20 //this is the code how we can respond to specific URL
21
22 //this is gonna fire for every single request
23 app.use((req,res)=>{
24   console.log('new request made');
25   console.log('host: ',req.hostname);
26   console.log('path: ', req.path);
27   console.log('method: ',req.method);
28 })
29
30
31 // the below are get handlers
32 app.get('/', (req, res)=>{
33   //res.send('<p>home page</p>');
34   //res.sendFile('./views/index.html', {root: __dirname});
35
36 const blogs = [
37   {title:'Movie Date', snippet:'I remember when we went for movie first time...Honestly,I was little bit nervous al
      
```
- Terminal:** Shows the command `node` and the output:
 

```

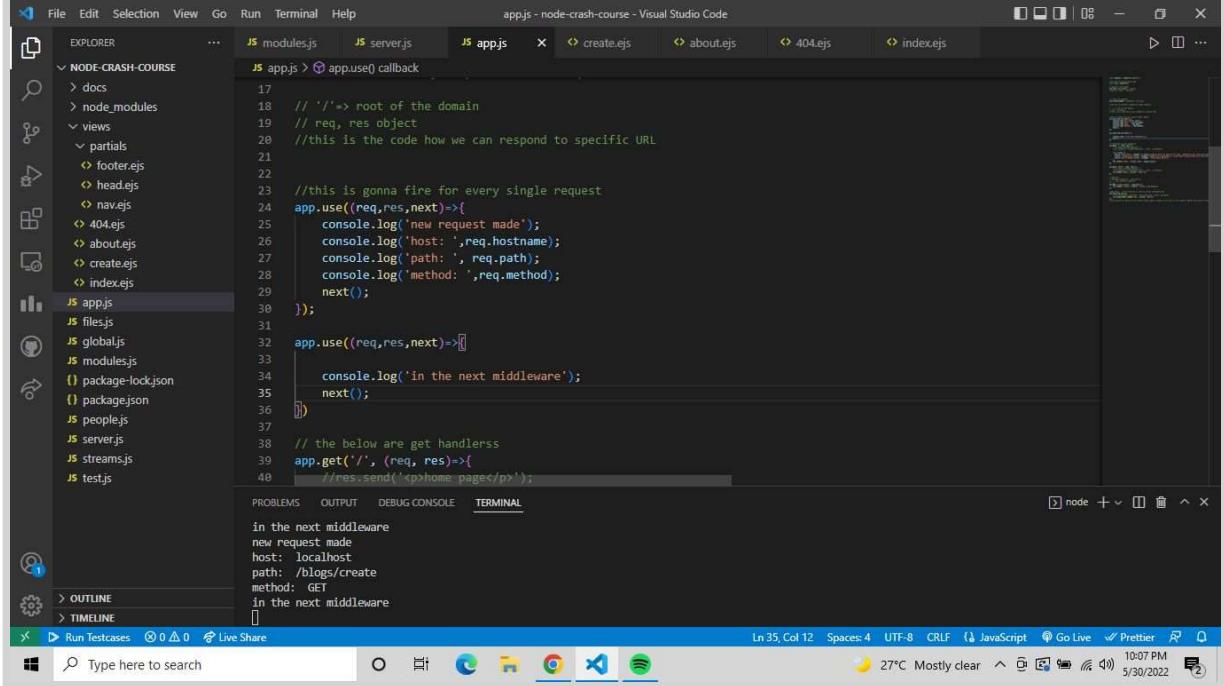
path: /
method: GET
new request made
host: localhost
path: /
method: GET
      
```
- Browser:** A Microsoft Edge window is open at `localhost:3000`, displaying a blog creation form titled "Blog with Vaidehi create a new Blog". The form has fields for "Blog title:", "Blog snippets:", and "Blog body:". A red arrow points from the browser's status bar (which shows "Partly cloudy" and the date/time) towards the browser window.
- Taskbar:** The taskbar at the bottom of the screen shows the Windows Start button, a search bar, and icons for various applications like File Explorer, Task View, and Spotify.

**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**

# Using next( )

now ata browser nahi hang hoty



The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure for "NODE-CRASH-COURSE" with files like modules.js, server.js, app.js, create.ejs, about.ejs, 404.ejs, index.ejs, footer.ejs, head.ejs, nav.ejs, 404.ejs, about.ejs, create.ejs, and index.ejs.
- Code Editor:** The active file is app.js, which contains the following code:

```

17 // '/' -> root of the domain
18 // req, res object
19 //this is the code how we can respond to specific URL
20
21 //this is gonna fire for every single request
22 app.use((req,res,next)=>{
23   console.log('new request made');
24   console.log(`host: ${req.hostname}`);
25   console.log(`path: ${req.path}`);
26   console.log(`method: ${req.method}`);
27   next();
28 });
29
30 app.use((req,res,next)=>{
31   console.log('in the next middleware');
32   next();
33 });
34
35
36
37 // the below are get handlers
38 app.get('/', (req, res)=>{
39   res.send('<p>home page</p>');
40 });

```
- Terminal:** Shows the output of the application running, indicating a new request made, host, path, method, and the message "in the next middleware".
- Bottom Bar:** Includes icons for Run Testcases, Live Share, a search bar, and various system and development status indicators.

## #3rd Party Middleware

tons of middleware created already in express:

morgan DT  
1.10.0 • Public • Published 2 years ago

[Readme](#) [Explore BETA](#) [5 Dependencies](#) [7,819 Dependents](#) [26 Versions](#)

## morgan

HTTP request logger middleware for node.js  
Named after **Dexter**, a show you should not watch until completion.

### API

```
var morgan = require('morgan')
```

#### morgan(format, options)

Create a new morgan logger middleware function using the given `format` and `options`. The

Version: 1.10.0 License: MIT

Install: `npm i morgan`

Repository: [github.com/expressjs/morgan](https://github.com/expressjs/morgan)

Homepage: [github.com/expressjs/morgan#readme](https://github.com/expressjs/morgan#readme)

Weekly Downloads: 3,293,604

26°C Mostly clear 10:16 PM 5/30/2022

File Edit Selection View Go Run Terminal Help

app.js - node-crash-course - Visual Studio Code

EXPLORER JS modules.js JS server.js JS app.js package.json create.ejs about.ejs 404.ejs index.ejs JS files.js JS global.js JS modules.js {} package-lock.json {} package.json JS people.js JS server.js JS streams.js JS test.js

```
JS app.js > ...
14 //listen for requests
15 app.listen(3000); //by default localhost
16
17 //now how we actually respond to those requests
18
19 // '/> root of the domain
20 // req, res object
21 //this is the code how we can respond to specific URL
22
23 //this is gonna fire for every single request
24
25 app.use(morgan('dev')) //may be a string of predefined name
26 app.use((req,res,next)>{
27   console.log('new request made');
28   console.log('host: ',req.hostname);
29   console.log('path: ', req.path);
30   console.log('method: ',req.method);
31   next();
32 });
33 }
34
35 app.use((req,res,next)>{
36   console.log('in the next middleware');
37 })
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

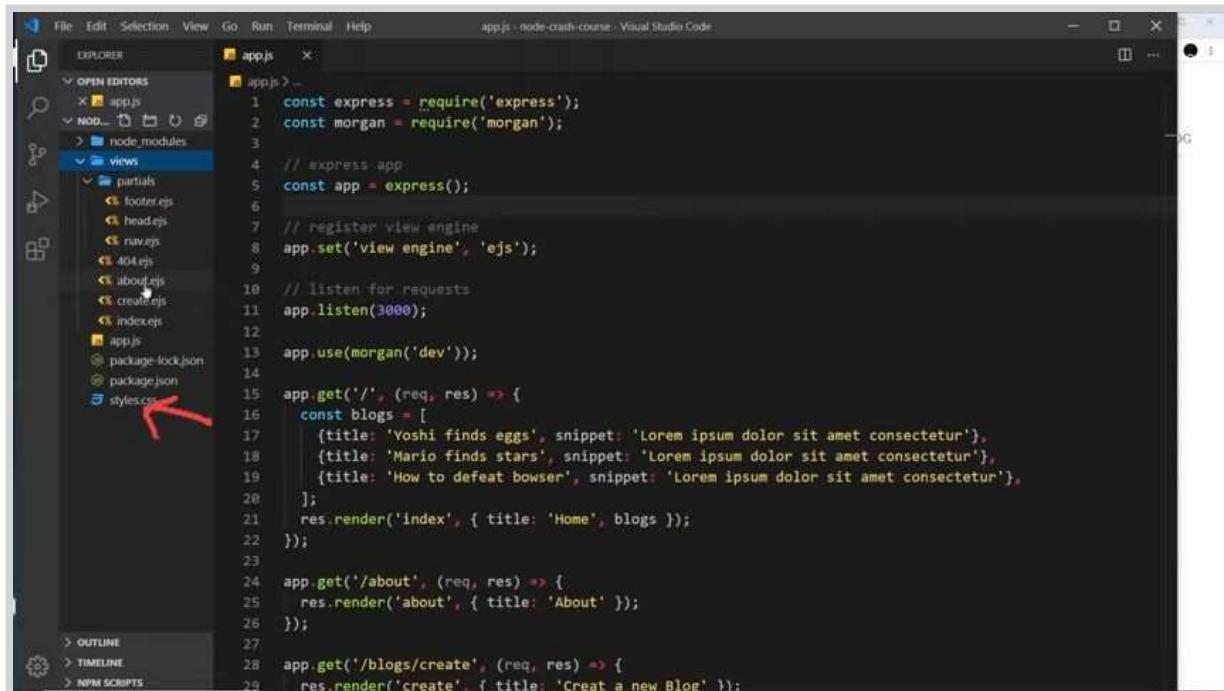
new request made  
host: localhost  
path: /blogs/create  
method: GET  
in the next middleware  
GET /blogs/create 304 19.513 ms - -

node 10:25 PM 5/30/2022

//static files:

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

12:27



```

File Edit Selection View Go Run Terminal Help
app.js - node-crash-course - Visual Studio Code

EXPLORER
OPEN EDITORS
  app.js
  NODE- CRASH-COURSE
    node_modules
      views
        partials
          footer.ejs
          head.ejs
          nav.ejs
          404.ejs
          about.ejs
          create.ejs
          index.ejs
          app.js
          package-lock.json
          package.json
          styles.css
        styles.css

OUTLINE
TIMELINE
NPM SCRIPTS

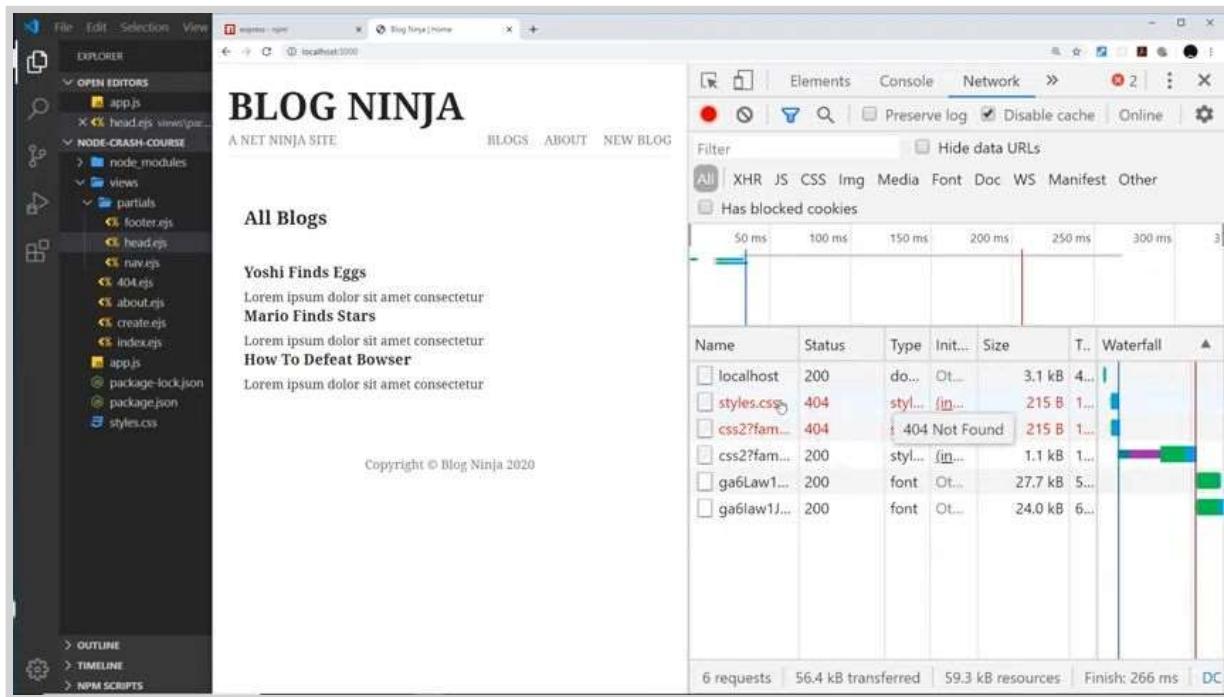
app.js
1 const express = require('express');
2 const morgan = require('morgan');
3
4 // express app
5 const app = express();
6
7 // register view engine
8 app.set('view engine', 'ejs');
9
10 // listen for requests
11 app.listen(3000);
12
13 app.use(morgan('dev'));
14
15 app.get('/', (req, res) => {
16   const blogs = [
17     {title: 'Yoshi finds eggs', snippet: 'Lorem ipsum dolor sit amet consectetur'},
18     {title: 'Mario finds stars', snippet: 'Lorem ipsum dolor sit amet consectetur'},
19     {title: 'How to defeat bowser', snippet: 'Lorem ipsum dolor sit amet consectetur'},
20   ];
21   res.render('index', { title: 'Home', blogs });
22 });
23
24 app.get('/about', (req, res) => {
25   res.render('about', { title: 'About' });
26 });
27
28 app.get('/blogs/create', (req, res) => {
29   res.render('create', { title: 'Create a new Blog' });
30 });

```

this wouldn't work for the browser...

see , style.css gives 404 error

13:02



Name	Status	Type	Init...	Size	T...	Waterfall
localhost	200	do...	Ot...	3.1 kB	4...	
styles.css	404	styl...	(in...	215 B	1...	
css2?fam...	404	styl...	(in...	215 B	1...	
css2?fam...	200	styl...	(in...	1.1 kB	1...	
ga6Law1...	200	font	Ot...	27.7 kB	5...	
ga6law1j...	200	font	Ot...	24.0 kB	6...	

//so middleware static banvu:

app.use(express.static('dirname'));

15:44

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure with files like `app.js`, `head.ejs`, `styles.css`, and various `ejs` files.
- Code Editor:** The `head.ejs` file is open, containing the following code:
 

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Blog Ninja | <% title %</title>
  <link rel="stylesheet" href="/styles.css">
</head>
```
- Terminal:** Shows the command `node app.js` running.
- Status Bar:** Displays the current file path (`head.ejs - node-crash-course - Visual Studio Code`), line count (Ln 64, Col 20), and other system information.

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure with files like `app.js`, `head.ejs`, `styles.css`, and various `ejs` files.
- Code Editor:** The `styles.css` file is open, containing the following code:
 

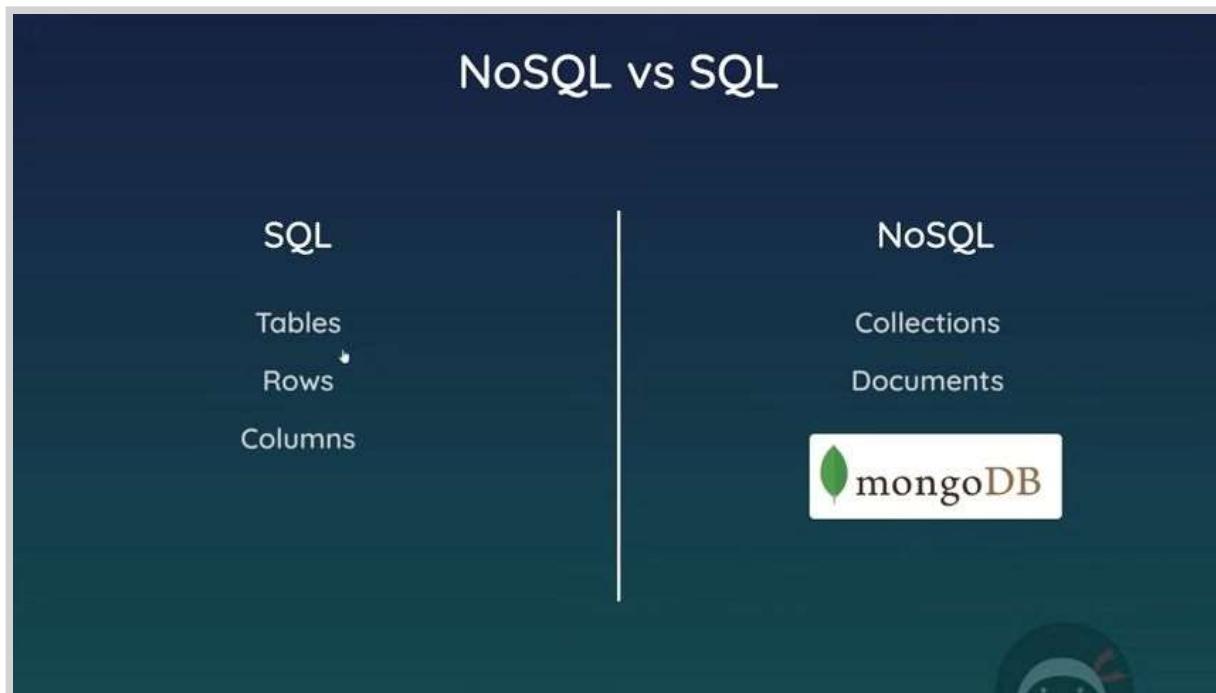
```
@import url('https://fonts.googleapis.com/css2?family=Noto+Serif:wght@400;700&display=swap');
```
- Terminal:** Shows the command `node app.js` running, with output indicating the application is restarting due to changes.
- Status Bar:** Displays the current file path (`styles.css - node-crash-course - Visual Studio Code`), line count (Ln 84, Col 20), and other system information.

## Video9:

//MongoDB

=>

00:32



//how it works:

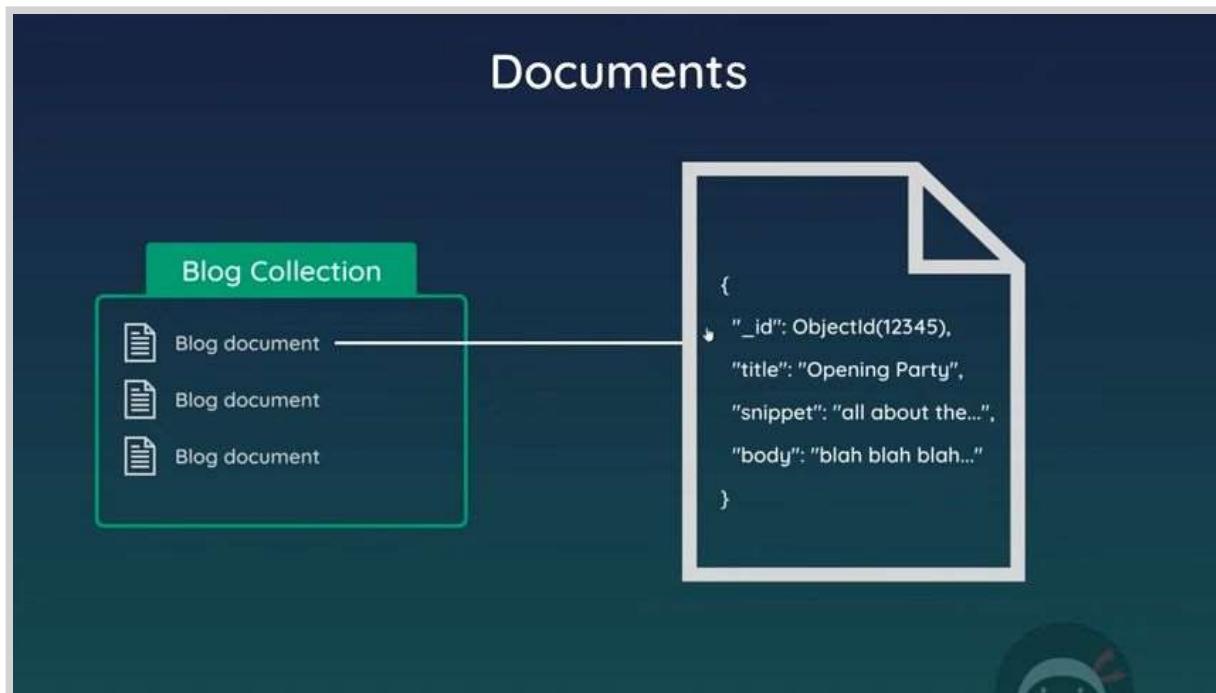
01:04



//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#cluster>

S

The screenshot shows the MongoDB Atlas interface. On the left, a sidebar lists 'DEPLOYMENT' (highlighted), 'Connect to Atlas' (status: 40%), 'PROJECT 0', and 'ACCESS'. Under 'ACCESS', there's a 'Custom Roles' section with a table:

Authentication Method	MongoDB Roles	Resources	Actions
SCRAM	readWriteAnyDatabase@admin	All Resources	<a href="#">EDIT</a> <a href="#">DELETE</a>

At the bottom of the interface, there's a 'Get Started' button and a status message: 'System Status: All Good'.

The screenshot shows the MongoDB Atlas interface. On the left, a sidebar lists 'DEPLOYMENT' (highlighted), 'Database' (highlighted), 'Data Lake', 'DATA SERVICES', 'Triggers', 'Data API (PREVIEW)', 'SECURITY', 'Database Access', 'Network Access', 'Advanced', and 'New On Atlas'. Under 'Database', it says 'VAIDHEI'S ORG - 2022-05-30 > PROJECT 0' and 'Database Deployments'. The main area features a 'Create a database' section with a 'Build a Database' button. A note at the bottom says: 'Once your database is up and running, live migrate an existing MongoDB database into Atlas with our [Live Migration Service](#)'. At the bottom, there's a 'Get Started' button.

VAIDELHI'S ORG - 2022-05-30 > PROJECT 0 > DATABASES

**nodetuts**

VERSION 5.0.8 REGION AWS Mumbai (ap-south-1)

Collections

DATABASES: 0 COLLECTIONS: 0

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

VAIDELHI'S ORG - 2022-05-30 > PROJECT 0 > DATABASES

**node-tuts.blocs**

STORAGE SIZE: 4KB TOTAL DOCUMENTS: 0 INDEXES TOTAL SIZE: 4KB

Find Indexes Schema Anti-Patterns Aggregation Search Indexes

INSERT DOCUMENT

QUERY RESULTS: 0

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.

**Database User Privileges**

Read and write to any database 1 SELECTED

**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**  
Select one **built-in role** for this user.  
Read and write to any database 1 SELECTED

**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.

The screenshot shows the MongoDB Atlas dashboard for a project named "VAIDELHI'S ORG - 2022-05-30 > PROJECT 0". A progress bar indicates the deployment is at 40%. The "Connect to Atlas" section lists several steps: "Build your first cluster" (green checkmark), "Create your first database user" (green checkmark), "Add IP Address to your Access List" (yellow circle), "Load Sample Data (Optional)" (yellow circle), and "Connect to your cluster" (green checkmark). Below this is a summary table with columns: VERSION, REGION, CLUSTER TIER, TYPE, BACKUPS, LINKED REALM APP, and ATLAS SEARCH. The cluster details are: VERSION 5.0.8, REGION AWS / Mumbai (ap-south-1), CLUSTER TIER MO Sandbox (General), TYPE Replica Set - 3 nodes, BACKUPS Inactive, LINKED REALM APP None Linked, and ATLAS SEARCH Create Index. To the right, there's a "FREE" and "SHARED" upgrade offer and a "Upgrade" button.

The screenshot shows a modal dialog titled "Connect to nodetuts" on the MongoDB Atlas dashboard. It has three tabs: "Setup connection security" (selected), "Choose a connection method", and "Connect". A message states: "You need to secure your MongoDB Atlas cluster before you can use it. Set which users and IP addresses can access your cluster now." Below this, a yellow box says: "You can't connect yet. Set up your firewall access below." Step 1: "Add a connection IP address" with options: "Add Your Current IP Address" (button), "Add a Different IP Address", and "Allow Access from Anywhere". Step 2: "Create a Database User" with a note: "A MongoDB user has been added to this project. Not yours? Create one in the MongoDB Users tab." A yellow box below says: "You'll need your MongoDB user's credentials in the next step." At the bottom are "Close" and "Choose a connection method" buttons.

The screenshot shows the Visual Studio Code editor with a file named "app.js" open. The code is as follows:

```

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_vaideli:Cpu@123@nodetuts.m6nky.mongodb.net/?retryWrites=true&w=majority";

//register view engine
app.set('view engine', 'ejs');
//app.set('view', 'myviews');

```

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

06:53

## Mongoose, Models & Schemas

06:58

### Mongoose

- Mongoose is an ODM library - Object Document Mapping library



07:59

## Schemas & Models

- Schemas defines the structure of a type of data / document
  - Properties & property types

User Schema:

- name (string), required
- age (number)
- bio (string), required

Blog Schema:

- title (string), required
- snippet (string), required
- body (string), required

09:06

## Schemas & Models

- Models allow us to communicate with database collections



**so we're using mongoose to connect to the database:  
mongoose is a third party package  
now we can use mongoose object to connect to the database...**

The screenshot shows a Visual Studio Code interface. On the left is the Explorer sidebar with a tree view of files: NODE-CRASH-COURSE (docs, node\_modules, public, views), partials (footer.ejs, head.ejs, nav.ejs, 404.ejs, about.ejs, create.ejs, index.ejs), and JS files (app.js, files.js, global.js, modules.js, package-lock.json, package.json, people.js, server.js, streams.js, test.js). The app.js file is selected. The main editor area contains the following code:

```

const express = require('express');
const morgan = require('morgan');
const mongoose = require('mongoose');

//we need to setup express app
const app = express();

//connect to mongodb
const dbURL = 'mongodb+srv://net_vaidehi:CPU@1234@nodetuts.m6nky.mongodb.net/node-tuts?retryWrites=true&w=majority';
mongoose.connect();

```

The terminal below shows the command `npm install mongoose` being run, which added 28 packages and audited 108 packages in 5s. It also shows a warning about vulnerabilities.

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

This screenshot shows the same Visual Studio Code environment as above, but with a different terminal output. The terminal shows an error message from node.js regarding the use of options like `useCreateIndex` and `useFindAndModify` in connection strings. It points to a specific line in the mongoose library's index.js file where these options are not supported.

```

at Mongoose._promiseOrCallback (C:\Users\Shree Ganraj\Comp\Documents\node-crash-course\node_modules\mongoose\lib\index.js:1181:10)
    [Symbol(errorLabels)]: Set(0) {}
}
[nodemon] app crashed - waiting for file changes before starting...
[nodemon] restarting due to changes...
[nodemon] starting node app.js
C:\Users\Shree Ganraj\Comp\Documents\node-crash-course\node_modules\mongoose\lib\connection_string.js:280
    throw new Error(`_MongoParseError('${optionWord} ${Array.from(unsupportedOptions).join(', ')} ${isOrAre} not supported`);

```

solution:

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

**error:**

The screenshot shows a Visual Studio Code interface. The left sidebar displays a file tree for a 'NODE-CRASH-COURSE' project, containing files like 'app.js', 'head.ejs', 'footer.ejs', and 'styles.css'. The main editor area shows a portion of 'app.js' with code related to MongoDB connection. The bottom right corner of the screen shows a terminal window with the following error message:

```
MongooseServerSelectionError: connection <monitor> to 52.66.254.200:27017 closed
at NativeConnection.Connection.openUri (C:\Users\Shree Ganraj Comp\Documents\node-crash-course\node_modules\mongoose\lib\connection.js:807:32)
at C:\Users\Shree Ganraj Comp\Documents\node-crash-course\node_modules\mongoose\lib\index.js:343:10
at C:\Users\Shree Ganraj Comp\Documents\node-crash-course\node_modules\mongoose\lib\helpers\promiseOrCallback.js:32:5
at new Promise (<anonymous>)
at promiseOrCallback (C:\Users\Shree Ganraj Comp\Documents\node-crash-course\node_modules\mongoose\lib\helpers\promiseOrCallback.js:31:18)
at Mongoose._promiseOrCallback (C:\Users\Shree Ganraj Comp\Documents\node-crash-course\node_modules\mongoose\lib\index.js:1181:10)
at Mongoose.connect (C:\Users\Shree Ganraj Comp\Documents\node-crash-course\node_modules\mongoose\lib\index.js:342:29)
at Object.<anonymous> (C:\Users\Shree Ganraj Comp\Documents\node-crash-course\app.js:10:10)
at Module._compile (node:internal/modules/cjs/loader:1101:14)
at Object.Module._extensions..js (node:internal/modules/cjs/loader:1153:10)
at Reason: TopologyDescription {
  type: 'ReplicaSetNoPrimary',
  servers: Map(3) {
    'nodetuts-shard-00-02.m6nky.mongodb.net:27017' => ServerDescription {
      _hostAddress: HostAddress {
```

The terminal also shows the current date and time: 5/31/2022 10:06 AM.

## solution:

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

//when connection is complete it returns promises:

The screenshot shows a Visual Studio Code interface. The left sidebar displays a file tree for a 'NODE-CRASH-COURSE' project, containing files like 'app.js', 'head.ejs', 'footer.ejs', and 'styles.css'. The main editor area shows a portion of 'app.js' with code for connecting to MongoDB using the 'mongoose' module. The bottom right corner of the screen shows a terminal window with the following output:

```
connected to db
[nodemon] restarting due to changes...
[nodemon] starting "node app.js"
[nodemon] restarting due to changes...
connected to db
[nodemon] starting "node app.js"
connected to db
[nodemon] restarting due to changes...
[nodemon] starting "node app.js"
connected to db
[]
```

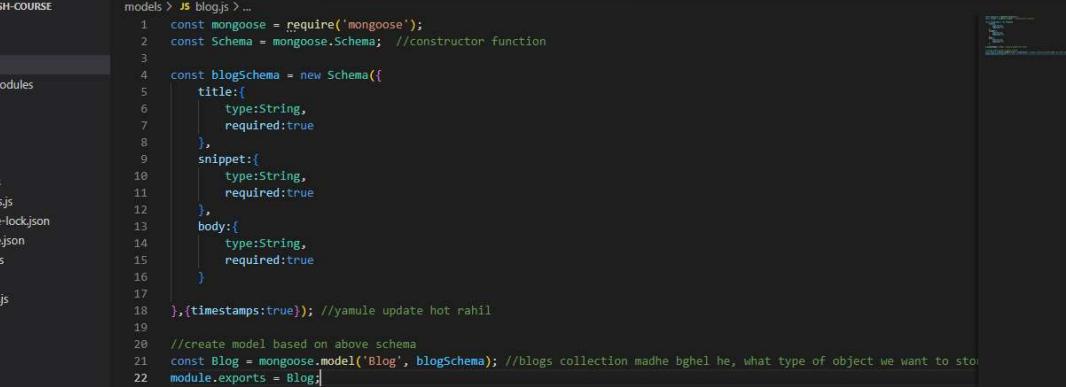
The terminal also shows the current date and time: 5/31/2022 10:10 AM.

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

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows a tree view of the project structure under "NODE-CRASH-COURSE". The "app.js" file is currently selected.
- Code Editor:** Displays the content of "app.js". The code initializes an Express app, connects to MongoDB, and sets up EJS as the view engine.
- Terminal:** Shows the output of running the application, indicating it's connected to the database and restarting due to changes.
- Bottom Status Bar:** Provides information about the current file (Ln 13, Col 35), workspace settings (Spaces: 2, UTF-8, CRLF), and active extensions (JavaScript, Go Live, Prettier).

//now create a schemas for our blog modules



```
const mongoose = require('mongoose');
const Schema = mongoose.Schema; //constructor function

const blogSchema = new Schema({
  title: {
    type: String,
    required: true
  },
  snippet: {
    type: String,
    required: true
  },
  body: {
    type: String,
    required: true
  }
}, {timestamps: true}); //yamule update hot rahil

//create model based on above schema
const Blog = mongoose.model('Blog', blogSchema); //blogs collection madhe bghel he, what type of object we want to store
module.exports = Blog;
```

19:28

## Getting & Saving Data

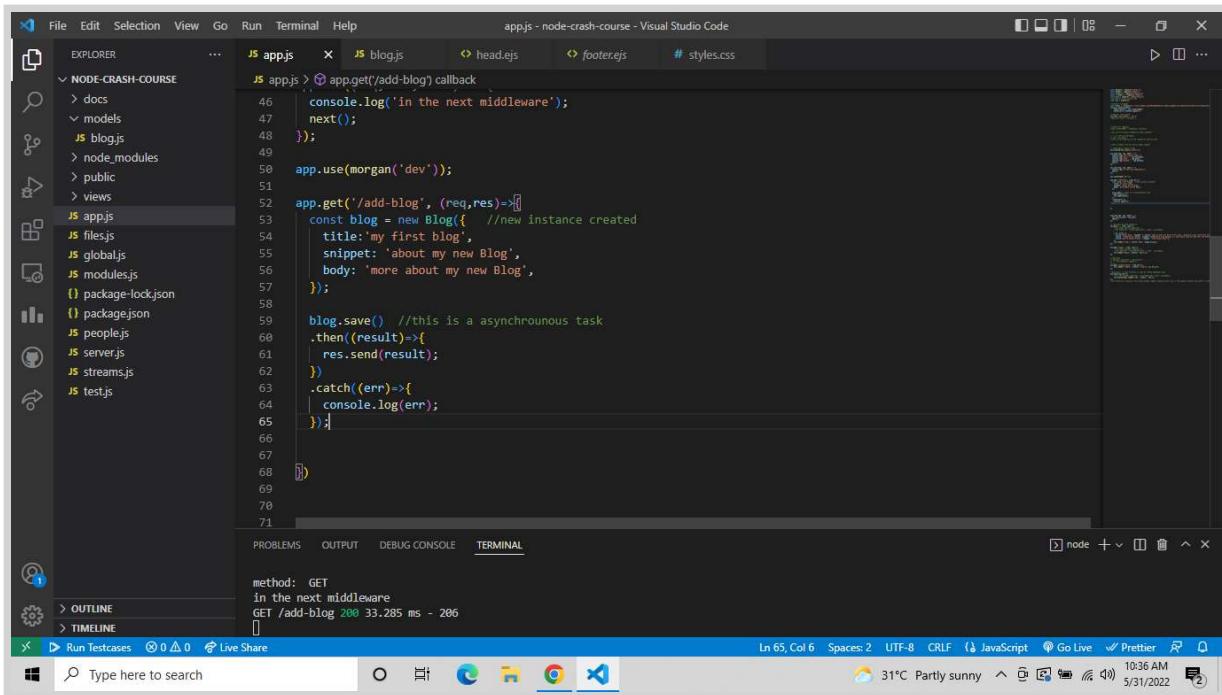
**//mongoose and mongo sandbox routes:  
just out to test the interaction with database**



```

js blog.js
> node_modules
> public
> views
4  const { result } = require('lodash');
5  const Blog = require('./models/blog');
6 //we need to setup express app
7  const app = express();

```



```

File Edit Selection View Go Run Terminal Help app.js - node-crash-course - Visual Studio Code

EXPLORER JS app.js JS blog.js head.ejs footer.ejs # styles.css
NODE-CRASH-COURSE
docs
models
JS blog.js
node_modules
public
views
JS app.js
JS files.js
JS global.js
JS modules.js
{} package-lock.json
{} package.json
JS people.js
JS server.js
JS streams.js
JS test.js

app.get('/add-blog', (req,res)=>{
  const blog = new Blog({
    title: 'my first blog',
    snippet: 'about my new Blog',
    body: 'more about my new Blog'
  });

  blog.save() //this is a asynchronous task
    .then(result=>{
      res.send(result);
    })
    .catch(err=>{
      console.log(err);
    });
});

method: GET
in the next middleware
GET /add-blog 200 33.285 ms - 206

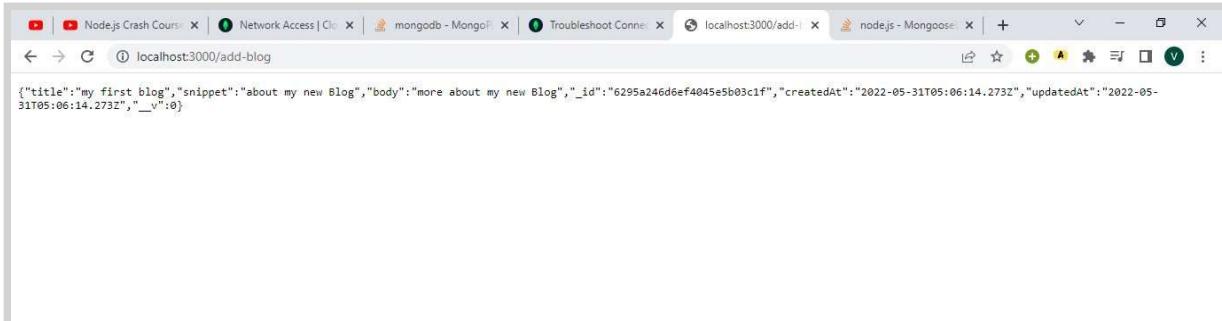
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Ln 65, Col 6 Spaces: 2 UTF-8 CRLF ↗ JavaScript ↗ Go Live ↗ Prettier ↗

Type here to search

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



```

{
  "title": "my first blog",
  "snippet": "about my new Blog",
  "body": "more about my new Blog",
  "_id": "6295a246d6ef4045e5b03c1f",
  "createdAt": "2022-05-31T05:06:14.273Z",
  "updatedAt": "2022-05-31T05:06:14.273Z",
  "__v": 0
}

```

The screenshot shows the MongoDB Atlas interface. On the left sidebar, under the 'Database' section, 'nodetuts' is selected. In the main area, the 'Collections' tab is active, showing the 'nodetuts' database with a single collection named 'blogs'. The table below shows one document with the following details:

Collection Name	Documents	Documents Size	Documents Avg	Indexes	Index Size	Index Avg
blogs	1	158B	158B	1	20KB	20KB

The screenshot shows the MongoDB Atlas interface with the 'Find' tab selected for the 'blogs' collection in the 'nodetuts' database. The results table shows one document with the following fields:

_id	objectId	title	snippet	body	createdAt	updatedAt
<code>_id: ObjectId("6295114217e3dd75b4986762")</code>		"my first blog"	"about my new Blog"	"more about my new Blog"	2022-05-31T05:06:14.273+00:00	2022-05-31T05:06:14.273+00:00

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

```

modules.js
package-lock.json
package.json
people.js
server.js
streams.js
test.js

    68 });
    69
    70 app.get('/all-blogs', (req,res)=>{
    71   Blog.find() //as this method is also asynchronous therefore we use (then) method
    72   .then((result)=>[
    73     | res.send(result); //sending response to the browser
    74     |
    75     | .catch((err)=>{
    76     |   | console.log(err);
    77     | });
    78   ]);
    79

```

```
[{"_id": "6295a246d6ef4045e5b03c1f", "title": "my first blog", "snippet": "about my new Blog", "body": "more about my new Blog", "createdAt": "2022-05-31T05:06:14.273Z", "updatedAt": "2022-05-31T05:06:14.273Z", "_v": 0}, {"_id": "6295a3dafc362b5b1882a4fb", "title": "my new blog 1", "snippet": "about my new Blog", "body": "more about my new Blog", "createdAt": "2022-05-31T05:12:58.406Z", "updatedAt": "2022-05-31T05:12:58.406Z", "_v": 0}]
```

29:38

```
37 });
38
39 app.get('/all-blogs', (req, res) => {
40   Blog.find()
41     .then((result) => {
42       res.send(result);
43     })
44     .catch((err) => {
45       console.log(err);
46     });
47 });
48
49 app.get('/single-blog', (req, res) => {
50   Blog.findById('5eb415667fcf2d6448fc162a')
51     .then((result) => {
52       res.send(result);
53     })
54     .catch((err) => {
55       console.log(err);
56     });
57 })
```

The terminal shows the application starting and receiving requests:

```
[nodemon] starting `node app.js`
GET /add-blog 200 785.166 ms - 203
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
GET /all-blogs 200 777.339 ms - 407
```

**//I don't want this kind of routes...****i want to integrate this routes with my main routess...so**30:45

## Outputting Documents in Views

```

File Edit Selection View Go Run Terminal Help appjs - node-crash-course - Visual Studio Code
EXPLORER JS app.js JS blog.js head.ejs footer.ejs # styles.css
NODE-CRASH-COURSE
docs
models
JS blog.js
node_modules
public
views
JS app.js
JS files.js
JS global.js
modules.js
package-lock.json
package.json
JS people.js
server.js
streams.js
test.js

JS app.js > app.get('blogs') callback
98 // the below are get handlers
99 app.get('/', (req, res)=>{
100   //res.send('<p>home page</p>');
101   //res.sendFile('./views/index.html', {root: __dirname});
102
103   // const blogs =[

104   // {title:'Movie Date', snippet:'I remember when we went for movie first time...Honestly,I was little bit nervous.'},
105   // {title: 'Competitive Coding', snippet:'Competitive programming is a mind sport usually held over the Internet.'},
106   // {title: 'How to defeat Brower', snippet: 'Lorem epson delor.'},
107   // ];
108
109   // res.render('index', {title:'Home', blogs:blogs});
110   res.redirect('/blogs');

111 app.get('/about', (req, res)=>{
112   //res.send('<p>about page</p>');
113   //res.sendFile('./views/about.html', {root: __dirname});
114   res.render('about', {title: 'About'});
115 });
116
117 //redirects
118 // app.get('/about-me', (req,res)=>{
119 //   res.redirect('/about');
120 // });

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
[nodemon] restarting due to changes...
[nodemon] starting 'node app.js'
new request made:
host: localhost
path: /blogs
method: GET
in the next middleware
GET /blogs 200 35.410 ms - 1114

```

```

File Edit Selection View Go Run Terminal Help appjs - node-crash-course - Visual Studio Code
EXPLORER JS app.js JS blog.js head.ejs footer.ejs # styles.css
NODE-CRASH-COURSE
docs
models
JS blog.js
node_modules
public
views
JS app.js
JS files.js
JS global.js
modules.js
package-lock.json
package.json
JS people.js
server.js
streams.js
test.js

JS app.js > app.get('blogs') callback
119 // app.get('/about-me', (req,res)=>
120 //   res.redirect('/about');
121 // );
122
123 //blog_routes
124 app.get('/blogs',(req,res)=>{
125   Blog.find().sort([['createdAt: -1']]);
126   .then((result) =>{
127     res.render('index', {title: 'All Blogs', blogs:result});
128   })
129   .catch((err)>{
130     console.log(err);
131   });
132 });
133
134 app.get('/blogs/create', (req,res)=>{
135   res.render('create', {title: 'create a new Blog'});
136 });
137
138 //app.use() // this function is used to create middleware and
139 app.use((req,res)=>{
140   //res.sendFile('./views/404.html',{root: __dirname});
141   res.status(404).render('404', {title: '404'});
142 });

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
[nodemon] restarting due to changes...
[nodemon] starting 'node app.js'
new request made:
host: localhost
path: /blogs
method: GET
in the next middleware
GET /blogs 200 35.410 ms - 1114

```

## 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

## Request Types

GET requests to get a resource

POST requests to create new data (e.g. a new blog)

DELETE requests to delete data (e.g. delete a blog)

PUT requests to update data (e.g. update a blog)

//

01:42

## Request Types

localhost:3000/blogs	↓	GET
localhost:3000/blogs/create		GET
localhost:3000/blogs		POST

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

**//route item:**

02:44

## Request Types

localhost:3000/blogs	GET
localhost:3000/blogs/create	GET
localhost:3000/blogs	POST
localhost:3000/blogs/:id	GET
localhost:3000/blogs/:id	DELETE

02:57

## Request Types

localhost:3000/blogs	GET
localhost:3000/blogs/create	GET
localhost:3000/blogs	POST
localhost:3000/blogs/:id	GET
localhost:3000/blogs/:id	DELETE
localhost:3000/blogs/:id	PUT

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

03:51

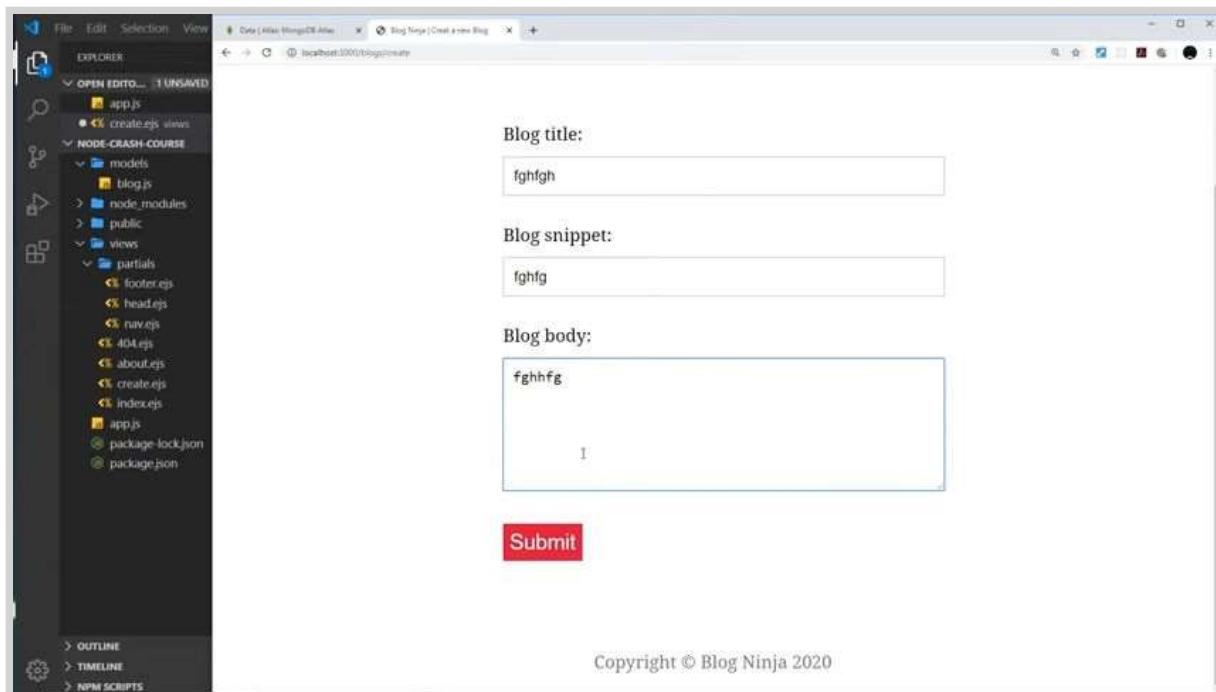
## Request Types

localhost:3000/blogs	GET
localhost:3000/blogs/create	GET
localhost:3000/blogs	POST
localhost:3000/blogs/:id	GET
localhost:3000/blogs/:id	DELETE

### //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**

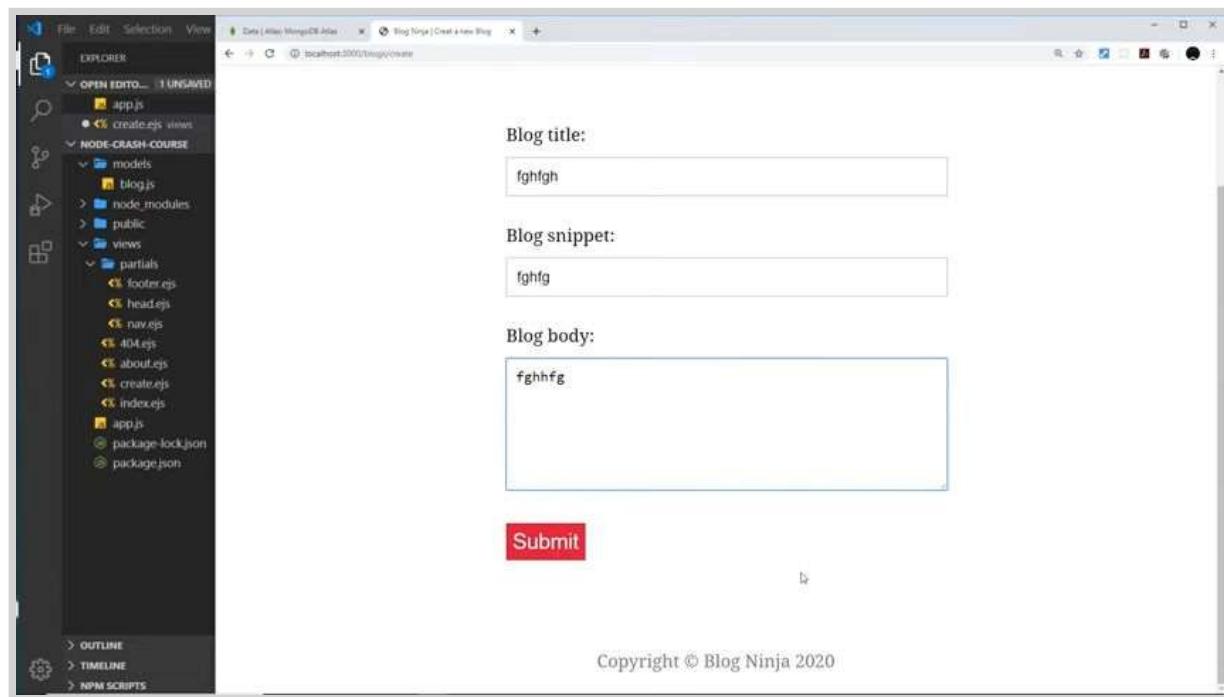
04:16

**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

```

File Edit Selection View Go Run Terminal Help
OPEN EDITOR 1 UNSAVED
views > create.ejs > HTML > body > div.create-blog.content > form
1 <html lang="en">
2   <% include('../partials/head.ejs') %>
3   <body>
4     <% include('../partials/nav.ejs') %>
5
6   <div class="create-blog content">
7     <form action="/blogs">
8       <label for="title">Blog title:</label>
9       <input type="text" id="title" required>
10      <label for="snippet">Blog snippet:</label>
11    </form>
12  </div>
13</body>
14</html>
  
```

//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

```

File Edit Selection View Go Run Terminal Help
OPEN EDITOR 1 UNSAVED
views > create.ejs > HTML > body > div.create-blog.content > form
1 <html lang="en">
2   <% include('../partials/head.ejs') %>
3   <body>
4     <% include('../partials/nav.ejs') %>
5
6   <div class="create-blog content">
7     <form action="/blogs" method="POST">
8       <label for="title">Blog title:</label>
9       <input type="text" id="title" required>
10      <label for="snippet">Blog snippet:</label>
11    </form>
12  </div>
13</body>
14</html>
  
```

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

**tr name attribute add karava lagel**

06:08

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows the project structure with files like `app.js`, `create.ejs`, `index.ejs`, and `views`.
- Code Editor (Center):** Displays the `create.ejs` file content. The code is an EJS template for creating a blog post. It includes partials for header, navigation, and footer, and a form for title, snippet, body, and submission.
- Status Bar (Bottom):** Shows "OUTLINE", "TIMELINE", and "NPM SCRIPTS".

```
<html lang="en">
  <% include('../partials/head.ejs') %>
<body>

  <% include('../partials/nav.ejs') %>

  <div class="create-blog_content">
    <form action="/blogs" method="POST">
      <label for="title">Blog title:</label>
      <input type="text" id="title" name="title" required>
      <label for="snippet">Blog snippet:</label>
      <input type="text" id="snippet" name="snippet" required>
      <label for="body">Blog body:</label>
      <textarea id="body" required name="body"></textarea>
      <button>Submit</button>
    </form>
  </div>

  <% include('../partials/footer.ejs') %>

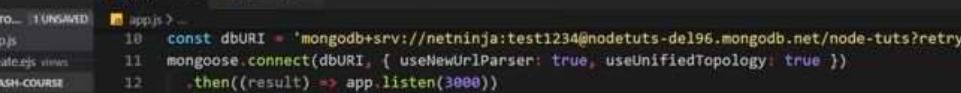
</body>
</html>
```

## //handling post request

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

adding middleware...

**07:35**



```
const dbURI = 'mongodb+srv://netninja:test1234@node-tuts-de196.mongodb.net/node-tuts?retryWrites=true&w=majority';
mongoose.connect(dbURI, { useNewUrlParser: true, useUnifiedTopology: true })
  .then((result) => app.listen(3000))
  .catch((err) => console.log(err));

// register view engine
app.set('view engine', 'ejs');

// middleware & static files
app.use(express.static('public'));
app.use(express.urlencoded({ extended: true }));
app.use(morgan('dev'));

// routes
```

```

File Edit Selection View Go Run Terminal Help
appjs - node-crash-course - Visual Studio Code
EXPLORER JS app.js x create.ejs JS blog.js head.ejs # styles.css
NODE-CRASH-COURSE
> docs
> models
> node_modules
> public
> views
> partials
> 404.ejs
> about.ejs
> create.ejs
> index.ejs
JS app.js
JS files.js
JS global.js
JS modules.js
{ package-lock.json
{ package.json
JS people.js
JS server.js
JS streams.js
JS test.js
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
path: /blogs
method: POST
in the next middleware
{
  title: 'Movie date',
  snippet: 'most precious day of my life..',
  body: '.....dhdhdb dbc'
}
Ln 36, Col 44 Spaces: 2 UTF-8 CRLF Go Live Prettier
Run Tests 0 0 Live Share
Type here to search
O C G X S
33°C Sunny 12:47 PM 5/31/2022

```

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

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

10:57

```

views
  partials
    footer.ejs
    head.ejs
    nav.ejs
    404.ejs
    about.ejs
    create.ejs
    index.ejs
    app.js
    package-lock.json
    package.json
42
43 app.post('/blogs', (req, res) => {
44   const blog = new Blog(req.body);
45
46   blog.save()
47     .then((result) => {
48       res.redirect('/blogs');
49     })
50     .catch((err) => {
51       console.log(err);
52     })
}
TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE
1: node
POST /blogs - - ms -
undefined
[nodemon] restarting due to changes...
[nodemon] starting 'node app.js'
GET / 302 3.127 ms - 56
GET /blogs 304 765.215 ms -
[nodemon] restarting due to changes...
[nodemon] starting 'node app.js'
[]

OUTLINE TIMELINE NPM SCRIPTS

```

STORAGE SIZE: 36KB TOTAL DOCUMENTS: 3 INDEXES TOTAL SIZE: 36KB

**nodeltuts.blogs**

**Find** Indexes Schema Anti-Patterns Aggregation Search Indexes

**FILTER** { field: 'value' } **OPTIONS** **Apply** **Reset**

**INSERT DOCUMENT**

**DOCUMENT**

```
_id: ObjectId("6295c39708d0f91c3b25153e")
title: "CP"
snippet: "Algorithms for Competitive Programming"
body: "The goal of this project is to translate the wonderful resource http://..."
createdat: 2022-05-31T07:28:23.050+00:00
updatedat: 2022-05-31T07:28:23.050+00:00
__v: 0
```

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

**id is variable**

12:08

## Route Parameters

- The variable parts of the route that may change value

localhost:3000/blogs/:id

12:24



14:12

```

File Edit Selection View Go Run Terminal Help index.ejs - node-crash-course - Visual Studio Code
EXPLORER index.ejs create.ejs blog.js head.ejs styles.css
NODE-CRASH-COURSE
docs
models
node_modules
public
views
partials
404.ejs
about.ejs
create.ejs
index.ejs
app.js
files.js
global.js
modules.js
package-lock.json
package.json
people.js
server.js
streams.js
test.js
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
in the next middleware
POST /blogs 302 78.562 ms - 56
new request made:
host: localhost
path: /blogs
method: GET
in the next middleware
GET /blogs 200 50.436 ms - 1284
Ln 16, Col 25 Spaces: 4 UTF-8 CRLF Go Live Prettier
Type here to search
35°C Sunny 2:10 PM 5/31/2022

```

```

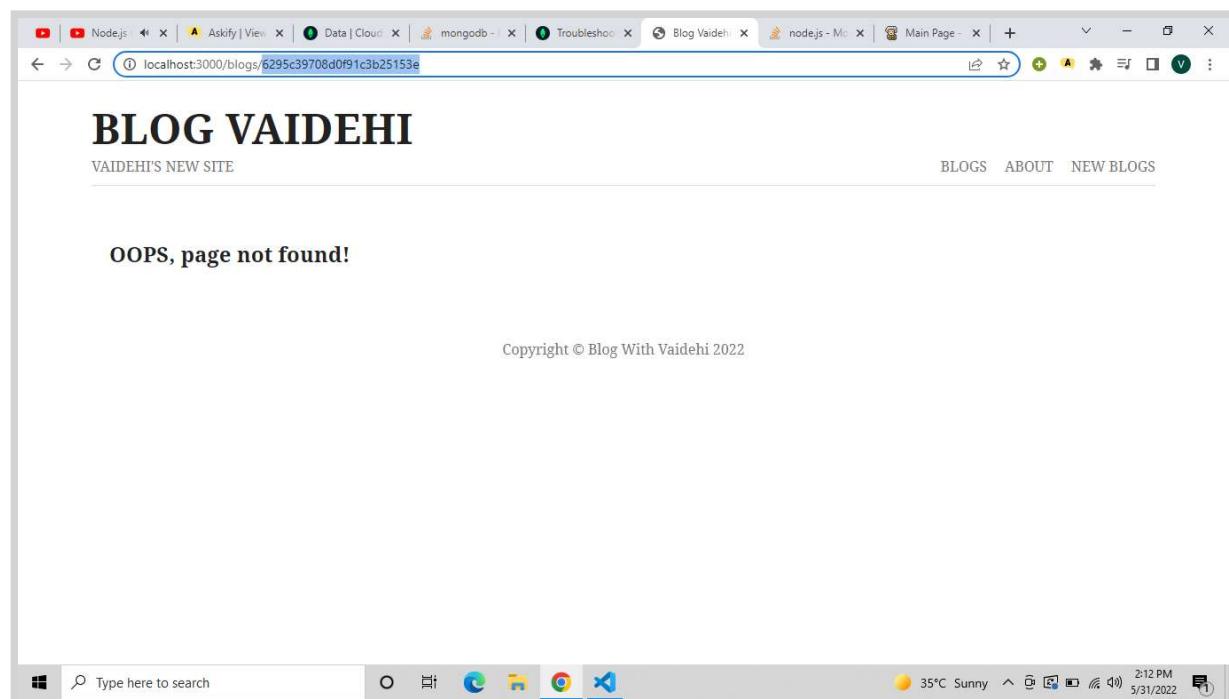
<body>
  <% include('../partials/nav') %>

  <div class="blogs content">
    <h2>All Blogs</h2>

    <% if(blogs.length > 0){ %>
      <% blogs.forEach(blog =>{ %>
        <a class="single" href="/blogs/">%= blog._id %</a>
        <h3 class="title">%= blog.title %</h3>
        <p class="snippet">%= blog.snippet %</p>
      </a%>
      <% } %>
    <% } else{ %>
      <p>There are no blogs to display....</p>
    <% } %>
  </div>
  <% include('../partials/footer.ejs') %>
</body>
</html>

```

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



## //but browser will got hung

The screenshot shows a Visual Studio Code interface with multiple files open in the Explorer sidebar: app.js, index.ejs, create.ejs, blog.js, head.ejs, and styles.css. The app.js file contains code for a blog application, including routes for creating and retrieving blogs. The terminal tab shows a command-line session where a user has run the application and issued a GET request to '/blogs/:id'. The response shows the browser is still loading, indicating a hangs. The status bar at the bottom right shows the date and time as 5/31/2022, 2:12 PM.

```

File Edit Selection View Go Run Terminal Help app.js - node-crash-course - Visual Studio Code
EXPLORER NODE-CRASH-COURSE
docs
models
blog.js
node_modules
public
views
partials
404.ejs
about.ejs
create.ejs
index.ejs
app.js
files.js
global.js
modules.js
{} package-lock.json
{} package.json
people.js
server.js
server.js
streams.js
test.js
app.js > @app.get('/blogs/:id') callback
135 app.post('/blogs', (req,res)=>{
136   const blog = new Blog(req.body);
137   blog.save()
138     .then(result=>{
139       res.redirect('/blogs');
140     })
141     .catch(err)=>{
142       console.log(err);
143     }
144   })
145 }
146 app.get('/blogs/:id' , (req,res)=>[
147   const id = req.params.id;
148   console.log(id);
149 ])
150
151 app.get('/blogs/create' , (req,res)=>{
152   res.render('create', {title: 'create a new Blog'});
153 }
154 );
155
156 //app.use() // this function is used to create middleware and
157
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
6295c39708d0f91c3b25153e
GET /blogs/6295c39708d0f91c3b25153e -- ms --
new request made:
host: localhost
path: /blogs/6295a3dafc362b5b1882a4fb
method: GET
In the next middleware
6295a3dafc362b5b1882a4fb
[]

Ln 149, Col 19 Spaces: 2 UFT-8 CRLF ↳ JavaScript ⚡ Go Live ✨ Prettier ⌂
Type here to search

```

## //models madhe

17:15

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows the project structure with files like `app.js`, `index.ejs`, `blog.js`, `404.ejs`, `about.ejs`, `create.ejs`, and `index.ejs`.
- Editor (Center):** Displays the `app.js` file content, which includes routes for listing blogs, viewing a blog by ID, and creating a new blog.
- Terminal (Bottom):** Shows the output of running the application with nodemon, including log messages and the URL `Seb415f1f2649478dc1e3bbB`.

//nantr views

The screenshot shows a Visual Studio Code interface with the following details:

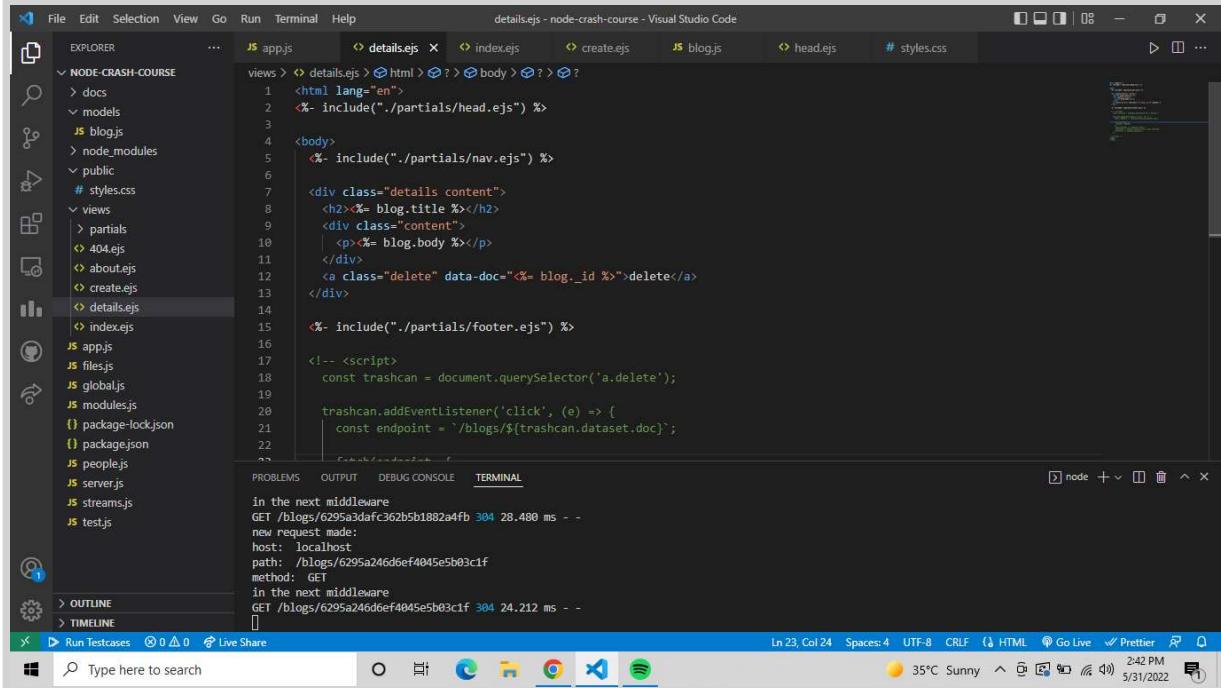
- File Explorer:** Shows the project structure with files like app.js, blog.js, index.ejs, etc., and folders like NODE-CRASH-COURSE, public, and views.
- Editor:** The file styles.css is open, containing the following code:

```
public > # styles.css > .blogs a:hover h3
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
```

/\* index styles \*/

```
.blogs a{
  display: block;
  margin: 40px 0;
  padding-left: 30px;
  border-left: 6px solid #crimson;
}
.blogs a:hover h3 {
  color: #crimson;
```
- Terminal:** Shows two log entries from the node.js application:

```
in the next middleware
GET /blogs/6295a3dxfc362b5b1882a4fb 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 -
```
- Bottom Bar:** Includes icons for Run Testcases, Live Share, a search bar, and system status indicators (CPU, RAM, battery, network).



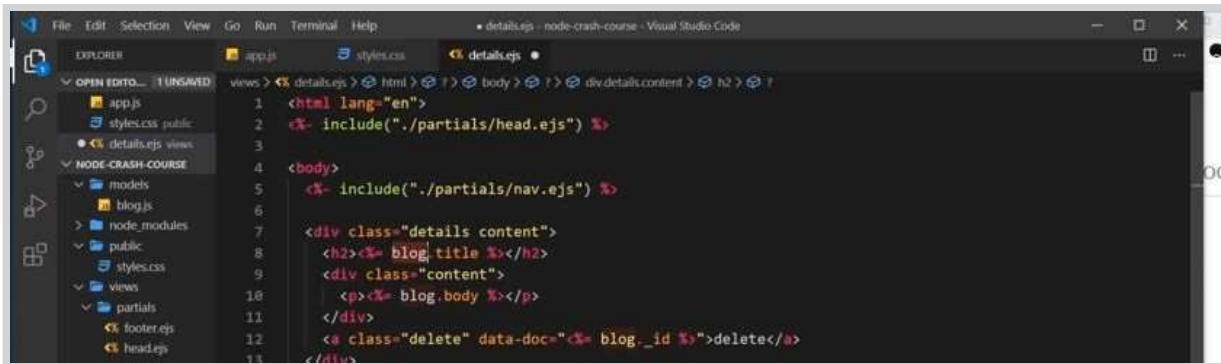
```

File Edit Selection View Go Run Terminal Help
details.ejs - node-crash-course - Visual Studio Code
EXPLORER JS app.js details.ejs index.ejs create.ejs JS blog.js head.ejs # styles.css
NODE-CRASH-COURSE
  > docs
  > models
    JS blog.js
  > node_modules
  > public
    # styles.css
  > views
    > partials
      404.ejs
      about.ejs
      create.ejs
      < details.ejs
      < index.ejs
    JS app.js
    JS files.js
    JS global.js
    JS modules.js
    {} package-lock.json
    {} package.json
    JS people.js
    JS server.js
    JS streams.js
    JS test.js
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
in the next middleware
GET /blogs/6295a3d9fc362b5b1882a4fb 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 -
Ln 23, Col 24 Spaces:4 UTF-8 CRLF ⓘ HTML ⓘ Go Live ⓘ Prettier ⓘ
node + × ⓘ
Type here to search ⓘ 35°C Sunny ⓘ 2:42 PM ⓘ 5/31/2022 ⓘ

```

**#Delete Requests:**  
**to delete a specific blog from frontend**  
**//u have to send a delete request**

21:36



```

File Edit Selection View Go Run Terminal Help
details.ejs - node-crash-course - Visual Studio Code
EXPLORER JS app.js styles.css details.ejs
OPEN EDITOR... 1 UNSAVED
NODE-CRASH-COURSE
  > docs
  > models
    JS blog.js
  > node_modules
  > public
    # styles.css
  > views
    > partials
      footer.ejs
      header.ejs
      < details.ejs
      < index.ejs
    JS app.js
    JS files.js
    JS global.js
    JS modules.js
    {} package-lock.json
    {} package.json
    JS people.js
    JS server.js
    JS streams.js
    JS test.js
views < details.ejs > < html > ? > body > ? > div.details.content > < h2 > ? >
1 < html lang="en">
2 <%- include("./partials/head.ejs") %>
3
4 <body>
<%- include("./partials/nav.ejs") %>
5 <div class="details content">
6   <h2><%= blog.title %></h2>
7   <div class="content">
8     <p><%= blog.body %></p>
9   </div>
10  <a class="delete" data-doc="<% blog._id %>">delete</a>
11
12  <%- include("./partials/footer.ejs") %>
13
14  <!-- <script>
15    const trashcan = document.querySelector('a.delete');
16
17    trashcan.addEventListener('click', (e) => {
18      const endpoint = `/blogs/${trashcan.dataset.doc}`;
19
20      fetch(endpoint, {
21        method: 'DELETE'
22      })
23    });
24  </script>
25
26  <!-- <script>
27    const trashcan = document.querySelector('a.delete');
28
29    trashcan.addEventListener('click', (e) => {
30      const endpoint = `/blogs/${trashcan.dataset.doc}`;
31
32      fetch(endpoint, {
33        method: 'DELETE'
34      })
35    });
36  </script>
37
38  <!-- <script>
39    const trashcan = document.querySelector('a.delete');
40
41    trashcan.addEventListener('click', (e) => {
42      const endpoint = `/blogs/${trashcan.dataset.doc}`;
43
44      fetch(endpoint, {
45        method: 'DELETE'
46      })
47    });
48  </script>
49
50  <!-- <script>
51    const trashcan = document.querySelector('a.delete');
52
53    trashcan.addEventListener('click', (e) => {
54      const endpoint = `/blogs/${trashcan.dataset.doc}`;
55
56      fetch(endpoint, {
57        method: 'DELETE'
58      })
59    });
60  </script>
61
62  <!-- <script>
63    const trashcan = document.querySelector('a.delete');
64
65    trashcan.addEventListener('click', (e) => {
66      const endpoint = `/blogs/${trashcan.dataset.doc}`;
67
68      fetch(endpoint, {
69        method: 'DELETE'
70      })
71    });
72  </script>
73
74  <!-- <script>
75    const trashcan = document.querySelector('a.delete');
76
77    trashcan.addEventListener('click', (e) => {
78      const endpoint = `/blogs/${trashcan.dataset.doc}`;
79
80      fetch(endpoint, {
81        method: 'DELETE'
82      })
83    });
84  </script>
85
86  <!-- <script>
87    const trashcan = document.querySelector('a.delete');
88
89    trashcan.addEventListener('click', (e) => {
90      const endpoint = `/blogs/${trashcan.dataset.doc}`;
91
92      fetch(endpoint, {
93        method: 'DELETE'
94      })
95    });
96  </script>
97
98  <!-- <script>
99    const trashcan = document.querySelector('a.delete');
100
101   trashcan.addEventListener('click', (e) => {
102     const endpoint = `/blogs/${trashcan.dataset.doc}`;
103
104     fetch(endpoint, {
105       method: 'DELETE'
106     })
107   });
108
109  </script>
110
111  <!-- <script>
112    const trashcan = document.querySelector('a.delete');
113
114    trashcan.addEventListener('click', (e) => {
115      const endpoint = `/blogs/${trashcan.dataset.doc}`;
116
117      fetch(endpoint, {
118        method: 'DELETE'
119      })
120    });
121
122  </script>
123
124  <!-- <script>
125    const trashcan = document.querySelector('a.delete');
126
127    trashcan.addEventListener('click', (e) => {
128      const endpoint = `/blogs/${trashcan.dataset.doc}`;
129
130      fetch(endpoint, {
131        method: 'DELETE'
132      })
133    });
134
135  </script>
136
137  <!-- <script>
138    const trashcan = document.querySelector('a.delete');
139
140    trashcan.addEventListener('click', (e) => {
141      const endpoint = `/blogs/${trashcan.dataset.doc}`;
142
143      fetch(endpoint, {
144        method: 'DELETE'
145      })
146    });
147
148  </script>
149
150  <!-- <script>
151    const trashcan = document.querySelector('a.delete');
152
153    trashcan.addEventListener('click', (e) => {
154      const endpoint = `/blogs/${trashcan.dataset.doc}`;
155
156      fetch(endpoint, {
157        method: 'DELETE'
158      })
159    });
160
161  </script>
162
163  <!-- <script>
164    const trashcan = document.querySelector('a.delete');
165
166    trashcan.addEventListener('click', (e) => {
167      const endpoint = `/blogs/${trashcan.dataset.doc}`;
168
169      fetch(endpoint, {
170        method: 'DELETE'
171      })
172    });
173
174  </script>
175
176  <!-- <script>
177    const trashcan = document.querySelector('a.delete');
178
179    trashcan.addEventListener('click', (e) => {
180      const endpoint = `/blogs/${trashcan.dataset.doc}`;
181
182      fetch(endpoint, {
183        method: 'DELETE'
184      })
185    });
186
187  </script>
188
189  <!-- <script>
190    const trashcan = document.querySelector('a.delete');
191
192    trashcan.addEventListener('click', (e) => {
193      const endpoint = `/blogs/${trashcan.dataset.doc}`;
194
195      fetch(endpoint, {
196        method: 'DELETE'
197      })
198    });
199
200  </script>
201
202  <!-- <script>
203    const trashcan = document.querySelector('a.delete');
204
205    trashcan.addEventListener('click', (e) => {
206      const endpoint = `/blogs/${trashcan.dataset.doc}`;
207
208      fetch(endpoint, {
209        method: 'DELETE'
210      })
211    });
212
213  </script>
214
215  <!-- <script>
216    const trashcan = document.querySelector('a.delete');
217
218    trashcan.addEventListener('click', (e) => {
219      const endpoint = `/blogs/${trashcan.dataset.doc}`;
220
221      fetch(endpoint, {
222        method: 'DELETE'
223      })
224    });
225
226  </script>
227
228  <!-- <script>
229    const trashcan = document.querySelector('a.delete');
230
231    trashcan.addEventListener('click', (e) => {
232      const endpoint = `/blogs/${trashcan.dataset.doc}`;
233
234      fetch(endpoint, {
235        method: 'DELETE'
236      })
237    });
238
239  </script>
240
241  <!-- <script>
242    const trashcan = document.querySelector('a.delete');
243
244    trashcan.addEventListener('click', (e) => {
245      const endpoint = `/blogs/${trashcan.dataset.doc}`;
246
247      fetch(endpoint, {
248        method: 'DELETE'
249      })
250    });
251
252  </script>
253
254  <!-- <script>
255    const trashcan = document.querySelector('a.delete');
256
257    trashcan.addEventListener('click', (e) => {
258      const endpoint = `/blogs/${trashcan.dataset.doc}`;
259
260      fetch(endpoint, {
261        method: 'DELETE'
262      })
263    });
264
265  </script>
266
267  <!-- <script>
268    const trashcan = document.querySelector('a.delete');
269
270    trashcan.addEventListener('click', (e) => {
271      const endpoint = `/blogs/${trashcan.dataset.doc}`;
272
273      fetch(endpoint, {
274        method: 'DELETE'
275      })
276    });
277
278  </script>
279
280  <!-- <script>
281    const trashcan = document.querySelector('a.delete');
282
283    trashcan.addEventListener('click', (e) => {
284      const endpoint = `/blogs/${trashcan.dataset.doc}`;
285
286      fetch(endpoint, {
287        method: 'DELETE'
288      })
289    });
290
291  </script>
292
293  <!-- <script>
294    const trashcan = document.querySelector('a.delete');
295
296    trashcan.addEventListener('click', (e) => {
297      const endpoint = `/blogs/${trashcan.dataset.doc}`;
298
299      fetch(endpoint, {
300        method: 'DELETE'
301      })
302    });
303
304  </script>
305
306  <!-- <script>
307    const trashcan = document.querySelector('a.delete');
308
309    trashcan.addEventListener('click', (e) => {
310      const endpoint = `/blogs/${trashcan.dataset.doc}`;
311
312      fetch(endpoint, {
313        method: 'DELETE'
314      })
315    });
316
317  </script>
318
319  <!-- <script>
320    const trashcan = document.querySelector('a.delete');
321
322    trashcan.addEventListener('click', (e) => {
323      const endpoint = `/blogs/${trashcan.dataset.doc}`;
324
325      fetch(endpoint, {
326        method: 'DELETE'
327      })
328    });
329
330  </script>
331
332  <!-- <script>
333    const trashcan = document.querySelector('a.delete');
334
335    trashcan.addEventListener('click', (e) => {
336      const endpoint = `/blogs/${trashcan.dataset.doc}`;
337
338      fetch(endpoint, {
339        method: 'DELETE'
340      })
341    });
342
343  </script>
344
345  <!-- <script>
346    const trashcan = document.querySelector('a.delete');
347
348    trashcan.addEventListener('click', (e) => {
349      const endpoint = `/blogs/${trashcan.dataset.doc}`;
350
351      fetch(endpoint, {
352        method: 'DELETE'
353      })
354    });
355
356  </script>
357
358  <!-- <script>
359    const trashcan = document.querySelector('a.delete');
360
361    trashcan.addEventListener('click', (e) => {
362      const endpoint = `/blogs/${trashcan.dataset.doc}`;
363
364      fetch(endpoint, {
365        method: 'DELETE'
366      })
367    });
368
369  </script>
370
371  <!-- <script>
372    const trashcan = document.querySelector('a.delete');
373
374    trashcan.addEventListener('click', (e) => {
375      const endpoint = `/blogs/${trashcan.dataset.doc}`;
376
377      fetch(endpoint, {
378        method: 'DELETE'
379      })
380    });
381
382  </script>
383
384  <!-- <script>
385    const trashcan = document.querySelector('a.delete');
386
387    trashcan.addEventListener('click', (e) => {
388      const endpoint = `/blogs/${trashcan.dataset.doc}`;
389
390      fetch(endpoint, {
391        method: 'DELETE'
392      })
393    });
394
395  </script>
396
397  <!-- <script>
398    const trashcan = document.querySelector('a.delete');
399
400    trashcan.addEventListener('click', (e) => {
401      const endpoint = `/blogs/${trashcan.dataset.doc}`;
402
403      fetch(endpoint, {
404        method: 'DELETE'
405      })
406    });
407
408  </script>
409
410  <!-- <script>
411    const trashcan = document.querySelector('a.delete');
412
413    trashcan.addEventListener('click', (e) => {
414      const endpoint = `/blogs/${trashcan.dataset.doc}`;
415
416      fetch(endpoint, {
417        method: 'DELETE'
418      })
419    });
420
421  </script>
422
423  <!-- <script>
424    const trashcan = document.querySelector('a.delete');
425
426    trashcan.addEventListener('click', (e) => {
427      const endpoint = `/blogs/${trashcan.dataset.doc}`;
428
429      fetch(endpoint, {
430        method: 'DELETE'
431      })
432    });
433
434  </script>
435
436  <!-- <script>
437    const trashcan = document.querySelector('a.delete');
438
439    trashcan.addEventListener('click', (e) => {
440      const endpoint = `/blogs/${trashcan.dataset.doc}`;
441
442      fetch(endpoint, {
443        method: 'DELETE'
444      })
445    });
446
447  </script>
448
449  <!-- <script>
450    const trashcan = document.querySelector('a.delete');
451
452    trashcan.addEventListener('click', (e) => {
453      const endpoint = `/blogs/${trashcan.dataset.doc}`;
454
455      fetch(endpoint, {
456        method: 'DELETE'
457      })
458    });
459
460  </script>
461
462  <!-- <script>
463    const trashcan = document.querySelector('a.delete');
464
465    trashcan.addEventListener('click', (e) => {
466      const endpoint = `/blogs/${trashcan.dataset.doc}`;
467
468      fetch(endpoint, {
469        method: 'DELETE'
470      })
471    });
472
473  </script>
474
475  <!-- <script>
476    const trashcan = document.querySelector('a.delete');
477
478    trashcan.addEventListener('click', (e) => {
479      const endpoint = `/blogs/${trashcan.dataset.doc}`;
480
481      fetch(endpoint, {
482        method: 'DELETE'
483      })
484    });
485
486  </script>
487
488  <!-- <script>
489    const trashcan = document.querySelector('a.delete');
490
491    trashcan.addEventListener('click', (e) => {
492      const endpoint = `/blogs/${trashcan.dataset.doc}`;
493
494      fetch(endpoint, {
495        method: 'DELETE'
496      })
497    });
498
499  </script>
500
501  <!-- <script>
502    const trashcan = document.querySelector('a.delete');
503
504    trashcan.addEventListener('click', (e) => {
505      const endpoint = `/blogs/${trashcan.dataset.doc}`;
506
507      fetch(endpoint, {
508        method: 'DELETE'
509      })
510    });
511
512  </script>
513
514  <!-- <script>
515    const trashcan = document.querySelector('a.delete');
516
517    trashcan.addEventListener('click', (e) => {
518      const endpoint = `/blogs/${trashcan.dataset.doc}`;
519
520      fetch(endpoint, {
521        method: 'DELETE'
522      })
523    });
524
525  </script>
526
527  <!-- <script>
528    const trashcan = document.querySelector('a.delete');
529
530    trashcan.addEventListener('click', (e) => {
531      const endpoint = `/blogs/${trashcan.dataset.doc}`;
532
533      fetch(endpoint, {
534        method: 'DELETE'
535      })
536    });
537
538  </script>
539
540  <!-- <script>
541    const trashcan = document.querySelector('a.delete');
542
543    trashcan.addEventListener('click', (e) => {
544      const endpoint = `/blogs/${trashcan.dataset.doc}`;
545
546      fetch(endpoint, {
547        method: 'DELETE'
548      })
549    });
550
551  </script>
552
553  <!-- <script>
554    const trashcan = document.querySelector('a.delete');
555
556    trashcan.addEventListener('click', (e) => {
557      const endpoint = `/blogs/${trashcan.dataset.doc}`;
558
559      fetch(endpoint, {
560        method: 'DELETE'
561      })
562    });
563
564  </script>
565
566  <!-- <script>
567    const trashcan = document.querySelector('a.delete');
568
569    trashcan.addEventListener('click', (e) => {
570      const endpoint = `/blogs/${trashcan.dataset.doc}`;
571
572      fetch(endpoint, {
573        method: 'DELETE'
574      })
575    });
576
577  </script>
578
579  <!-- <script>
580    const trashcan = document.querySelector('a.delete');
581
582    trashcan.addEventListener('click', (e) => {
583      const endpoint = `/blogs/${trashcan.dataset.doc}`;
584
585      fetch(endpoint, {
586        method: 'DELETE'
587      })
588    });
589
590  </script>
591
592  <!-- <script>
593    const trashcan = document.querySelector('a.delete');
594
595    trashcan.addEventListener('click', (e) => {
596      const endpoint = `/blogs/${trashcan.dataset.doc}`;
597
598      fetch(endpoint, {
599        method: 'DELETE'
600      })
601    });
602
603  </script>
604
605  <!-- <script>
606    const trashcan = document.querySelector('a.delete');
607
608    trashcan.addEventListener('click', (e) => {
609      const endpoint = `/blogs/${trashcan.dataset.doc}`;
610
611      fetch(endpoint, {
612        method: 'DELETE'
613      })
614    });
615
616  </script>
617
618  <!-- <script>
619    const trashcan = document.querySelector('a.delete');
620
621    trashcan.addEventListener('click', (e) => {
622      const endpoint = `/blogs/${trashcan.dataset.doc}`;
623
624      fetch(endpoint, {
625        method: 'DELETE'
626      })
627    });
628
629  </script>
630
631  <!-- <script>
632    const trashcan = document.querySelector('a.delete');
633
634    trashcan.addEventListener('click', (e) => {
635      const endpoint = `/blogs/${trashcan.dataset.doc}`;
636
637      fetch(endpoint, {
638        method: 'DELETE'
639      })
640    });
641
642  </script>
643
644  <!-- <script>
645    const trashcan = document.querySelector('a.delete');
646
647    trashcan.addEventListener('click', (e) => {
648      const endpoint = `/blogs/${trashcan.dataset.doc}`;
649
650      fetch(endpoint, {
651        method: 'DELETE'
652      })
653    });
654
655  </script>
656
657  <!-- <script>
658    const trashcan = document.querySelector('a.delete');
659
660    trashcan.addEventListener('click', (e) => {
661      const endpoint = `/blogs/${trashcan.dataset.doc}`;
662
663      fetch(endpoint, {
664        method: 'DELETE'
665      })
666    });
667
668  </script>
669
670  <!-- <script>
671    const trashcan = document.querySelector('a.delete');
672
673    trashcan.addEventListener('click', (e) => {
674      const endpoint = `/blogs/${trashcan.dataset.doc}`;
675
676      fetch(endpoint, {
677        method: 'DELETE'
678      })
679    });
680
681  </script>
682
683  <!-- <script>
684    const trashcan = document.querySelector('a.delete');
685
686    trashcan.addEventListener('click', (e) => {
687      const endpoint = `/blogs/${trashcan.dataset.doc}`;
688
689      fetch(endpoint, {
690        method: 'DELETE'
691      })
692    });
693
694  </script>
695
696  <!-- <script>
697    const trashcan = document.querySelector('a.delete');
698
699    trashcan.addEventListener('click', (e) => {
700      const endpoint = `/blogs/${trashcan.dataset.doc}`;
701
702      fetch(endpoint, {
703        method: 'DELETE'
704      })
705    });
706
707  </script>
708
709  <!-- <script>
710    const trashcan = document.querySelector('a.delete');
711
712    trashcan.addEventListener('click', (e) => {
713      const endpoint = `/blogs/${trashcan.dataset.doc}`;
714
715      fetch(endpoint, {
716        method: 'DELETE'
717      })
718    });
719
720  </script>
721
722  <!-- <script>
723    const trashcan = document.querySelector('a.delete');
724
725    trashcan.addEventListener('click', (e) => {
726      const endpoint = `/blogs/${trashcan.dataset.doc}`;
727
728      fetch(endpoint, {
729        method: 'DELETE'
730      })
731    });
732
733  </script>
734
735  <!-- <script>
736    const trashcan = document.querySelector('a.delete');
737
738    trashcan.addEventListener('click', (e) => {
739      const endpoint = `/blogs/${trashcan.dataset.doc}`;
740
741      fetch(endpoint, {
742        method: 'DELETE'
743      })
744    });
745
746  </script>
747
748  <!-- <script>
749    const trashcan = document.querySelector('a.delete');
750
751    trashcan.addEventListener('click', (e) => {
752      const endpoint = `/blogs/${trashcan.dataset.doc}`;
753
754      fetch(endpoint, {
755        method: 'DELETE'
756      })
757    });
758
759  </script>
760
761  <!-- <script>
762    const trashcan = document.querySelector('a.delete');
763
764    trashcan.addEventListener('click', (e) => {
765      const endpoint = `/blogs/${trashcan.dataset.doc}`;
766
767      fetch(endpoint, {
768        method: 'DELETE'
769      })
770    });
771
772  </script>
773
774  <!-- <script>
775    const trashcan = document.querySelector('a.delete');
776
777    trashcan.addEventListener('click', (e) => {
778      const endpoint = `/blogs/${trashcan.dataset.doc}`;
779
780      fetch(endpoint, {
781        method: 'DELETE'
782      })
783    });
784
785  </script>
786
787  <!-- <script>
788    const trashcan = document.querySelector('a.delete');
789
790    trashcan.addEventListener('click', (e) => {
791      const endpoint = `/blogs/${trashcan.dataset.doc}`;
792
793      fetch(endpoint, {
794        method: 'DELETE'
795      })
796    });
797
798  </script>
799
800  <!-- <script>
801    const trashcan = document.querySelector('a.delete');
802
803    trashcan.addEventListener('click', (e) => {
804      const endpoint = `/blogs/${trashcan.dataset.doc}`;
805
806      fetch(endpoint, {
807        method: 'DELETE'
808      })
809    });
810
811  </script>
812
813  <!-- <script>
814    const trashcan = document.querySelector('a.delete');
815
816    trashcan.addEventListener('click', (e) => {
817      const endpoint = `/blogs/${trashcan.dataset.doc}`;
818
819      fetch(endpoint, {
820        method: 'DELETE'
821      })
822    });
823
824  </script>
825
826  <!-- <script>
827    const trashcan = document.querySelector('a.delete');
828
829    trashcan.addEventListener('click', (e) => {
830      const endpoint = `/blogs/${trashcan.dataset.doc}`;
831
832      fetch(endpoint, {
833        method: 'DELETE'
834      })
835    });
836
837  </script>
838
839  <!-- <script>
840    const trashcan = document.querySelector('a.delete');
841
842    trashcan.addEventListener('click', (e) => {
843      const endpoint = `/blogs/${trashcan.dataset.doc}`;
844
845      fetch(endpoint, {
846        method: 'DELETE'
847      })
848    });
849
850  </script>
851
852  <!-- <script>
853    const trashcan = document.querySelector('a.delete');
854
855    trashcan.addEventListener('click', (e) => {
856      const endpoint = `/blogs/${trashcan.dataset.doc}`;
857
858      fetch(endpoint, {
859        method: 'DELETE'
860      })
861    });
862
863  </script>
864
865  <!-- <script>
866    const trashcan = document.querySelector('a.delete');
867
868    trashcan.addEventListener('click', (e) => {
869      const endpoint = `/blogs/${trashcan.dataset.doc}`;
870
871      fetch(endpoint, {
872        method: 'DELETE'
873      })
874    });
875
876  </script>
877
878  <!-- <script>
879    const trashcan = document.querySelector('a.delete');
880
881    trashcan.addEventListener('click', (e) => {
882      const endpoint = `/blogs/${trashcan.dataset.doc}`;
883
884      fetch(endpoint, {
885        method: 'DELETE'
886      })
887    });
888
889  </script>
890
891  <!-- <script>
892    const trashcan = document.querySelector('a.delete');
893
894    trashcan.addEventListener('click', (e) => {
895      const endpoint = `/blogs/${trashcan.dataset.doc}`;
896
897      fetch(endpoint, {
898        method: 'DELETE'
899      })
900    });
901
902  </script>
903
904  <!-- <script>
905    const trashcan = document.querySelector('a.delete');
906
907    trashcan.addEventListener('click', (e) => {
908      const endpoint = `/blogs/${trashcan.dataset.doc}`;
909
910      fetch(endpoint, {
911        method: 'DELETE'
912      })
913    });
914
915  </script>
916
917  <!-- <script>
918    const trashcan = document.querySelector('a.delete');
919
920    trashcan.addEventListener('click', (e) => {
921      const endpoint = `/blogs/${trashcan.dataset.doc}`;
922
923      fetch(endpoint, {
924        method: 'DELETE'
925      })
926    });
927
928  </script>
929
930  <!-- <script>
931    const trashcan = document.querySelector('a.delete');
932
933    trashcan.addEventListener('click', (e) => {
934      const endpoint = `/blogs/${trashcan.dataset.doc}`;
935
936      fetch(endpoint, {
937        method: 'DELETE'
938      })
939    });
940
941  </script>
942
943  <!-- <script>
944    const trashcan = document.querySelector('a.delete');
945
946    trashcan.addEventListener('click', (e) => {
947      const endpoint = `/blogs/${trashcan.dataset.doc}`;
948
949      fetch(endpoint, {
950        method: 'DELETE'
951      })
952    });
953
954  </script>
955
956  <!-- <script>
957    const trashcan = document.querySelector('a.delete');
958
959    trashcan.addEventListener('click', (e) => {
960      const endpoint = `/blogs/${trashcan.dataset.doc}`;
961
962      fetch(endpoint, {
963        method: 'DELETE'
964      })
965    });
966
967  </script>
968
969  <!-- <script>
970    const trashcan = document.querySelector('a.delete');
971
972    trashcan.addEventListener('click', (e) => {
973      const endpoint = `/blogs/${trashcan.dataset.doc}`;
974
975      fetch(endpoint, {
976        method: 'DELETE'
977      })
978    });
979
980  </script>
981
982  <!-- <script>
983    const trashcan = document.querySelector('a.delete');
984
985    trashcan.addEventListener('click', (e) => {
986      const endpoint = `/blogs/${trashcan.dataset.doc}`;
987
988      fetch(endpoint, {
989        method: 'DELETE'
990      })
991    });
992
993  </script>
994
995  <!-- <script>
996    const trashcan = document.querySelector('a.delete');
997
998    trashcan.addEventListener('click', (e) => {
999      const endpoint = `/blogs/${trashcan.dataset.doc}`;
1000
1001      fetch(endpoint, {
1002        method: 'DELETE'
1003      })
1004    });
1005
1006  </script>
1007
1008  <!-- <script>
1009    const trashcan = document.querySelector('a.delete');
1010
1011    trashcan.addEventListener('click', (e) => {
1012      const endpoint = `/blogs/${trashcan.dataset.doc}`;
1013
1014      fetch(endpoint, {
1015        method: 'DELETE'
1016      })
1017    });
1018
1019  </script>
1020
1021  <!-- <script>
1022    const trashcan = document.querySelector('a.delete');
1023
1024    trashcan.addEventListener('click', (e) => {
1025      const endpoint = `/blogs/${trashcan.dataset.doc}`;
1026
1027      fetch(endpoint, {
1028        method: 'DELETE'
1029      })
1030    });
1031
1032  </script>
1033
1034  <!-- <script>
1035    const trashcan = document.querySelector('a.delete');
1036
1037    trashcan.addEventListener('click', (e) => {
1038      const endpoint = `/blogs/${trashcan.dataset.doc}`;
1039
1040      fetch(endpoint, {
1041        method: 'DELETE'
1042      })
1043    });
1044
1045  </script>
1046
1047  <!-- <script>
1048    const trashcan = document.querySelector('a.delete');
1049
1050    trashcan.addEventListener('click', (e) => {
1051      const endpoint = `/blogs/${trashcan.dataset.doc}`;
1052
1053      fetch(endpoint, {
1054        method: 'DELETE'
1055      })
1056    });
1057
1058  </script>
1059
1060  <!-- <script>
1061    const trashcan = document.querySelector('a.delete');
1062
1063    trashcan.addEventListener('click', (e) => {
1064      const endpoint = `/blogs/${trashcan.dataset.doc}`;
1065
1066      fetch(endpoint, {
1067        method: 'DELETE'
1068      })
1069    });
1070
1071  </script>
1072
1073  <!-- <script>
1074    const trashcan = document.querySelector('a.delete');
1075
1076    trashcan.addEventListener('click', (e) => {
1077      const endpoint = `/blogs/${trashcan.dataset.doc}`;
1078
1079      fetch(endpoint, {
1080        method: 'DELETE'
1081      })
1082    });
1083
1084  </script>
1085
1086  <!-- <script>
1087    const trashcan = document.querySelector('a.delete');
1088
1089    trashcan.addEventListener('click', (e) => {
1090      const endpoint = `/blogs/${trashcan.dataset.doc}`;
1091
1092      fetch(endpoint, {
1093        method: 'DELETE'
1094      })
1095    });
1096
1097  </script>
1098
1099  <!-- <script>
1100    const trashcan = document.querySelector('a.delete');
1101
1102    trashcan.addEventListener('click', (e) => {
1103      const endpoint = `/blogs/${trashcan.dataset.doc}`;
1104
1105      fetch(endpoint, {
1106        method: 'DELETE'
1107      })
1108    });
1109
1110  </script>
1111
1112  <!-- <script>
1113    const trashcan = document.querySelector('a.delete');
1114
1115    trashcan.addEventListener('click', (e) => {
1116      const endpoint = `/blogs/${trashcan.dataset.doc}`;
1117
1118      fetch(endpoint, {
1119        method: 'DELETE'
1120      })
1121    });
1122
1123  </script>
1124
1125  <!-- <script>
1126    const trashcan = document.querySelector('a.delete');
1127
1128    trashcan.addEventListener('click', (e) => {
1129      const endpoint = `/blogs/${trashcan.dataset.doc}`;
1130
1131      fetch(endpoint, {
1132        method: 'DELETE'
1133      })
1134    });
1135
1136  </script>
1137
1138  <!-- <script>
1139    const trashcan = document.querySelector('a.delete');
1140
1141    trashcan.addEventListener('click', (e) => {
1142      const endpoint = `/blogs/${trashcan.dataset.doc}`;
1143
1144      fetch(endpoint, {
1145        method: 'DELETE'
1146      })
1147    });
1148
1149  </script>
1150
1151  <!-- <script>
1152    const trashcan = document.querySelector('a.delete');
1153
1154    trashcan.addEventListener('click', (e) => {
1155      const endpoint = `/blogs/${trashcan.dataset.doc}`;
1156
1157      fetch(endpoint, {
1158        method: 'DELETE'
1159      })
1160    });
1161
1162  </script>
1163
1164  <!-- <script>
1165    const trashcan = document.querySelector('a.delete');
1166
1167    trashcan.addEventListener('click', (e) => {
1168      const endpoint = `/blogs/${trashcan.dataset.doc}`;
1169
1170      fetch(endpoint, {
1171        method: 'DELETE'
1172      })
1173    });
1174
1175  </script>
1176
1177  <
```

```

File Edit Selection View Go Run Terminal Help
details.ejs - node-crash-course - Visual Studio Code
EXPLORER JS app.js views details.ejs index.ejs create.ejs blog.js head.ejs styles.css
NODE-CRASH-COURSE
docs
models
blog.js
node_modules
public
styles.css
views
partials
404.ejs
about.ejs
create.ejs
details.ejs
index.ejs
app.js
files.js
global.js
modules.js
package-lock.json
package.json
people.js
server.js
streams.js
test.js
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
in the next middleware
GET /blogs 304 17.070 ms -
new request made:
host: localhost
path: /blogs/6295c39708d0f91c3b25153e
method: GET
in the next middleware
GET /blogs/6295c39708d0f91c3b25153e 304 20.343 ms -
Ln 30, Col 5 Spaces:4 UTF-8 CRLF ⓘ HTML ⓘ Go Live ⓘ Prettier ⓘ
Run Tests 0 ⚡ 0 Live Share 35°C Sunny 248 PM 5/31/2022
Type here to search

```

The screenshot shows the Visual Studio Code interface with the details.ejs file open. The code handles a DELETE request to delete a blog post by its ID. It uses a script block to add an event listener for the 'click' event on an 'a.delete' element. When triggered, it constructs an endpoint URL and performs a fetch request to delete the document from the database. The terminal shows log messages indicating middleware execution and a successful 304 response.

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

```

File Edit Selection View Go Run Terminal Help
details.ejs - node-crash-course - Visual Studio Code
EXPLORER JS app.js views details.ejs index.ejs create.ejs blog.js head.ejs styles.css
NODE-CRASH-COURSE
docs
models
blog.js
node_modules
public
styles.css
views
partials
404.ejs
about.ejs
create.ejs
details.ejs
index.ejs
app.js
files.js
global.js
modules.js
package-lock.json
package.json
people.js
server.js
streams.js
test.js
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
//this is the AJAX request so server wr render nahi kru shkt(we can't use render)
Ln 23, Col 87 Spaces:4 UTF-8 CRLF ⓘ HTML ⓘ Go Live ⓘ Prettier ⓘ
Run Tests 0 ⚡ 0 Live Share 35°C Sunny 253 PM 5/31/2022
Type here to search

```

The screenshot shows the Visual Studio Code interface with the details.ejs file open. A comment is added to the code explaining that it is an AJAX request, so the server cannot render the page (we can't use render). The terminal shows log messages indicating nodemon restarting due to changes.

31:23

```

File Edit Selection View Go Run Terminal Help
OPEN EDITORS app.js styles.css details.ejs
EXPLORER app.js styles.css public NODE-CRASH-COURSE
models blog.js node_modules
public styles.css views
partials footer.ejs header.ejs nav.ejs
404.ejs about.ejs create.ejs details.ejs index.ejs
app.js package-lock.json package.json
OUTLINE
TIMELINE
NPM SCRIPTS
3 <body>
4   <% include("./partials/nav.ejs") %>
5
6   <div class="details content">
7     <h2><% blog.title %></h2>
8     <div class="content">
9       <p><% blog.body %></p>
10    </div>
11    <a class="delete" data-doc="<% blog._id %>">delete</a>
12  </div>
13
14  <% include("./partials/footer.ejs") %>
15
16  <script>
17    const trashcan = document.querySelector('a.delete');
18
19    trashcan.addEventListener('click', (e) => {
20      const endpoint = '/blogs/${trashcan.dataset.doc}';
21
22      fetch(endpoint, {
23        method: 'DELETE'
24      })
25      .then((response) => response.json())
26      .then((data) => console.log(data))
27      .catch((err) => console.log(err));
28    })
29  </script>
30
31 </body>

```

32:06

BLOG NINJA  
A NET NINJA SITE

BLOGS ABOUT NEW BLOG

**new blog 2** [delete](#)

more about my new blog

Copyright © Blog Ninja 2020

Console top

```

Seb415F...:44
redirect: "/blogs"
redirect: "/blogs"
> __proto__: Object

```

## Video11: Express Router & MVC:

=>

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

**//MVC Basics:**

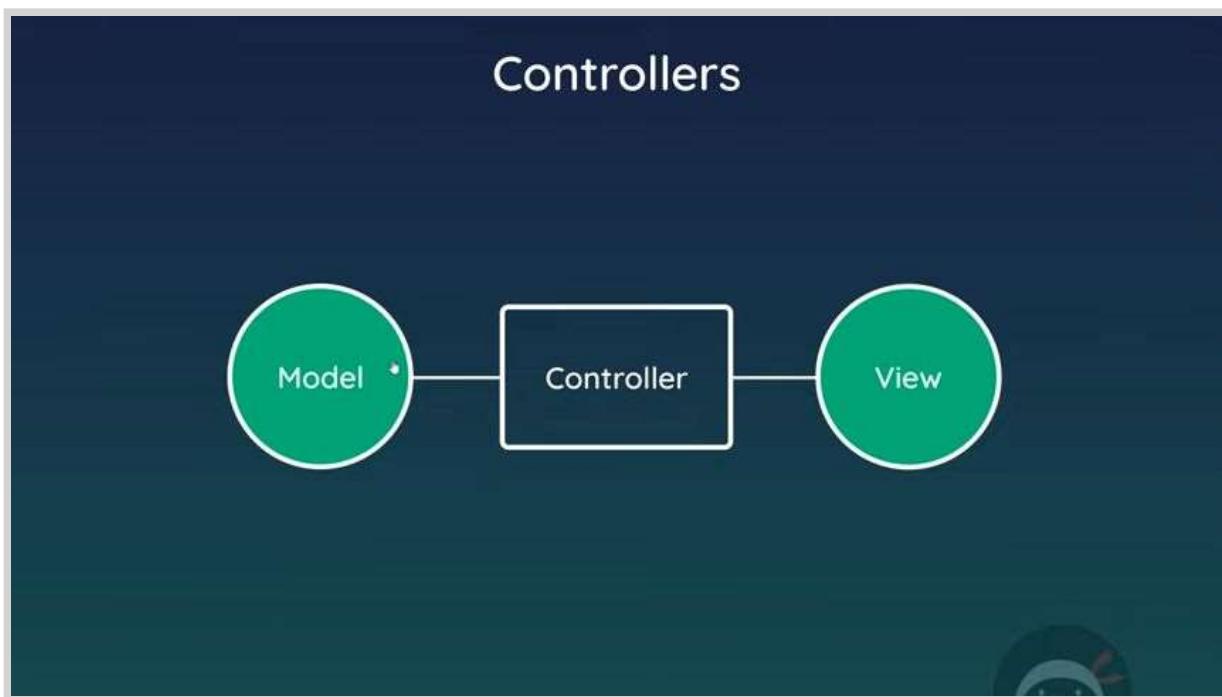
09:59

## MVC Basics

- Stands for Model, View, Controller
- MVC is a way of structuring our code & files
- Keeps code more modular, reusable & easier to read

**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

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure under "NODE-CRASH-COURSE". The "routes" folder contains "blogRoutes.js". Other files like "blogController.js", "app.js", and "package.json" are also listed.
- Code Editor:** The "blogRoutes.js" file is open, showing the following code:
 

```
1 // blog_index, blog_details, blog_create_get, blog_create_post, blog_delete
2
3 const express = require('express');
4 const blogController = require('../controllers/blogController');
5
6 const router = express.Router();
7
8 router.get('/', (req, res) => {
9
10   router.post('/', (req, res) => {
11     const blog = new Blog(req.body);
12
13     blog.save()
14       .then((result) => {
15         res.redirect('/blogs');
16       })
17       .catch((err) => {
18         console.log(err);
19       })
20   });
21
22 router.get('/create', (req, res) => {
23   res.render('create', { title: 'Create a new Blog' });
24 });
25
26 router.get('/:id', (req, res) => {
27   const id = req.params.id;
28   Blog.findById(id)
29     .then(result => {
```

14:55

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure under "NODE-CRASH-COURSE". The "routes" folder contains "blogRoutes.js". Other files like "blogController.js", "app.js", and "package.json" are also listed.
- Code Editor:** The "blogRoutes.js" file is open, showing the continuation of the code from the previous screenshot:
 

```
30       .then(result => {
31         res.render('details', { blog: result });
32       })
33     })
34   })
35 });
36
37 module.exports = router;
```

15:13

The screenshot shows the Visual Studio Code interface with the title "blogController.js - node-crash-course - Visual Studio Code". The Explorer sidebar on the left shows the project structure under "NODE-CRASH-COURSE": controllers (blogRoutes.js, blogController.js), models, node\_modules, public, routes (blogRoutes.js), views (app.js), package-lock.json, and package.json. The "blogController.js" file is open in the editor, containing the following code:

```

1 const Blog = require('../models/blog');
2 // blog_index, blog_details, blog_create_get, blog_create_post, blog_delete
3
4 const blog_index = (req, res) => {
5   Blog.find().sort({ createdAt: -1 })
6     .then((result) => {
7       res.render('index', { title: 'All Blogs', blogs: result });
8     })
9     .catch((err) => {
10       console.log(err);
11     })
12 }
13
14 module.exports = {
15   blog_index
16 }

```

15:25

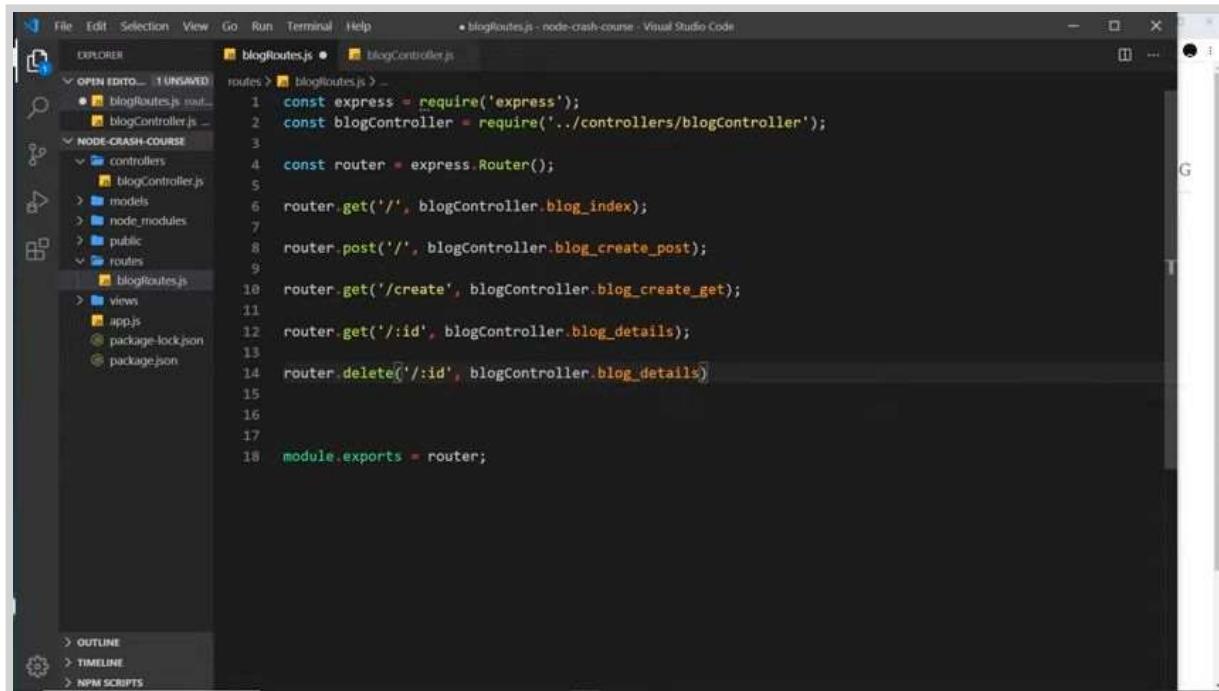
The screenshot shows the Visual Studio Code interface with the title "blogRoutes.js - node-crash-course - Visual Studio Code". The Explorer sidebar on the left shows the project structure under "NODE-CRASH-COURSE": controllers (blogRoutes.js, blogController.js), models, node\_modules, public, routes (blogRoutes.js), views (app.js), package-lock.json, and package.json. The "blogRoutes.js" file is open in the editor, containing the following code:

```

1 const express = require('express');
2 const blogController = require('../controllers/blogController');
3
4 const router = express.Router();
5
6 router.get('/', blogController.blog_index);
7
8 router.post('/', (req, res) => {
9   const blog = new Blog(req.body);
10
11   blog.save()
12     .then((result) => {
13       res.redirect('/blogs');
14     })
15     .catch((err) => {
16       console.log(err);
17     })
18 }
19
20 router.get('/create', (req, res) => {
21   res.render('create', { title: 'Create a new Blog' });
22 })
23
24 router.get('/:id', (req, res) => {
25   const id = req.params.id;
26   Blog.findById(id)
27     .then(result => {
28       res.render('details', { blog: result, title: 'Blog Details' });
29     })
30 }

```

18:31



```
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_delete);

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)

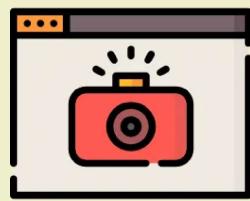
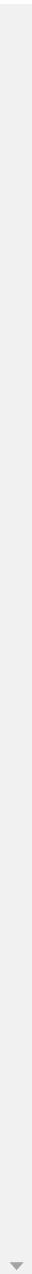












**Now you can use Askify in any websites**

[See How](#)