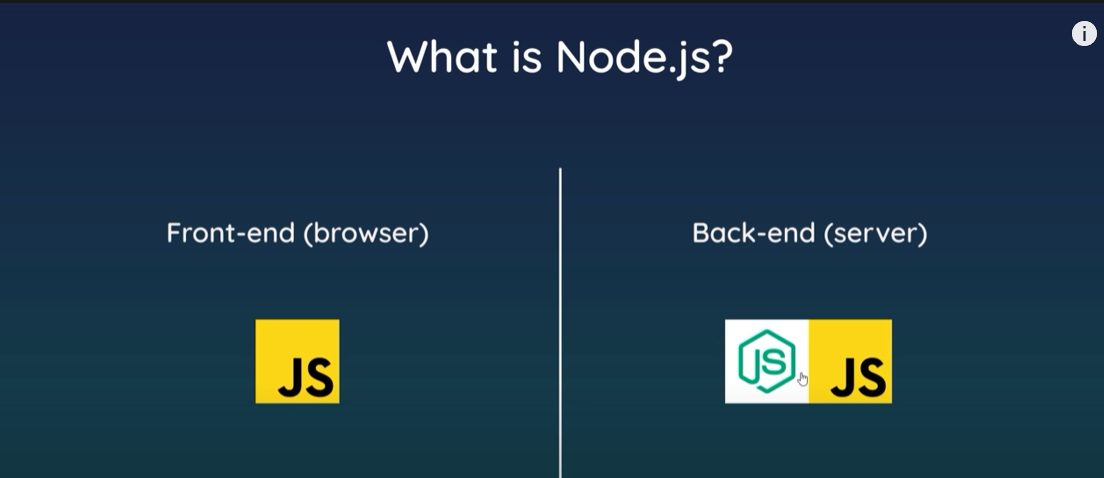
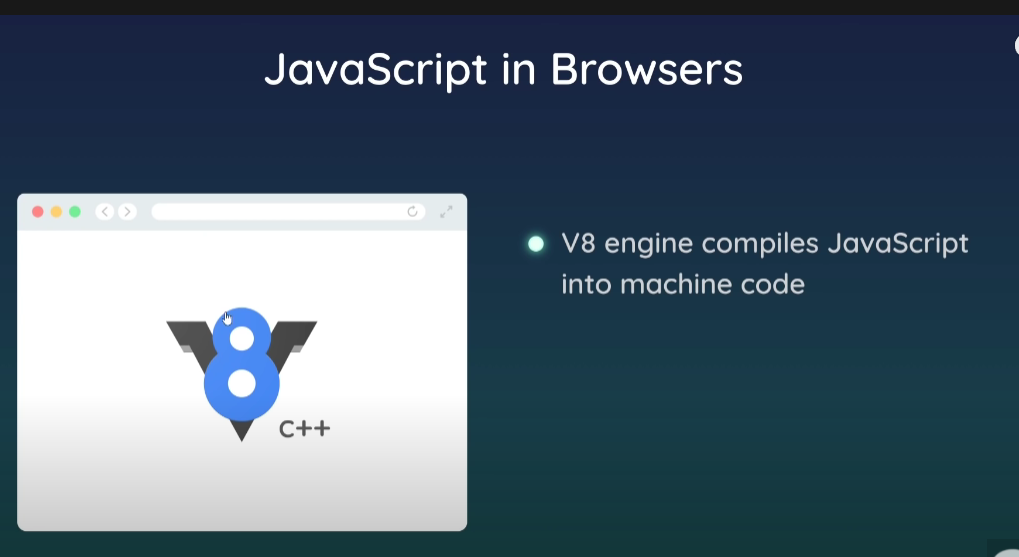
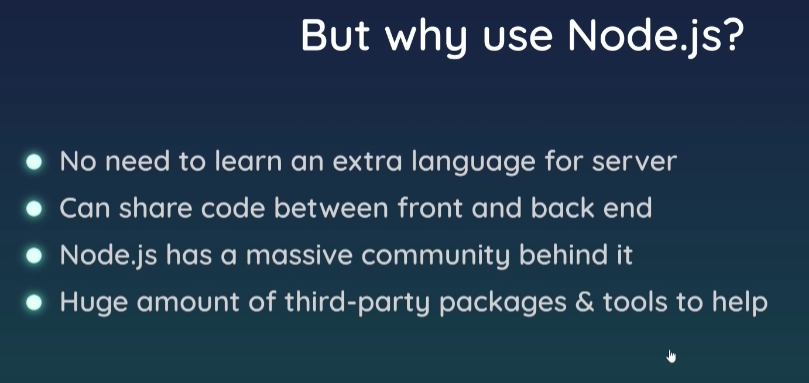
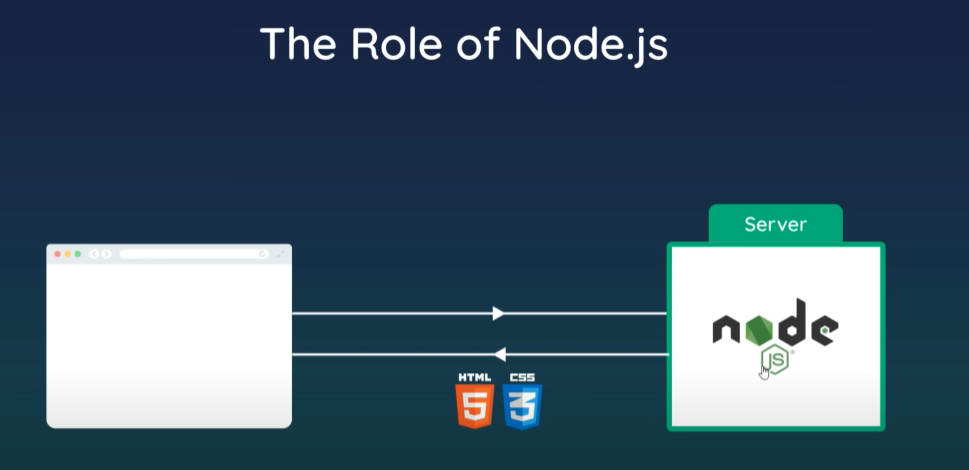
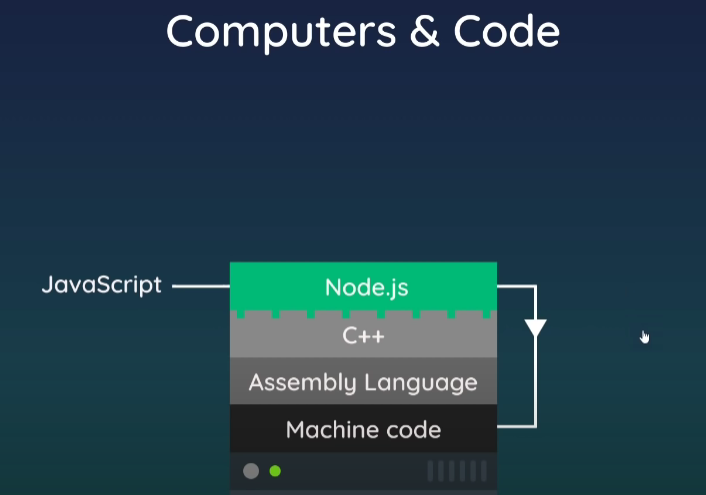
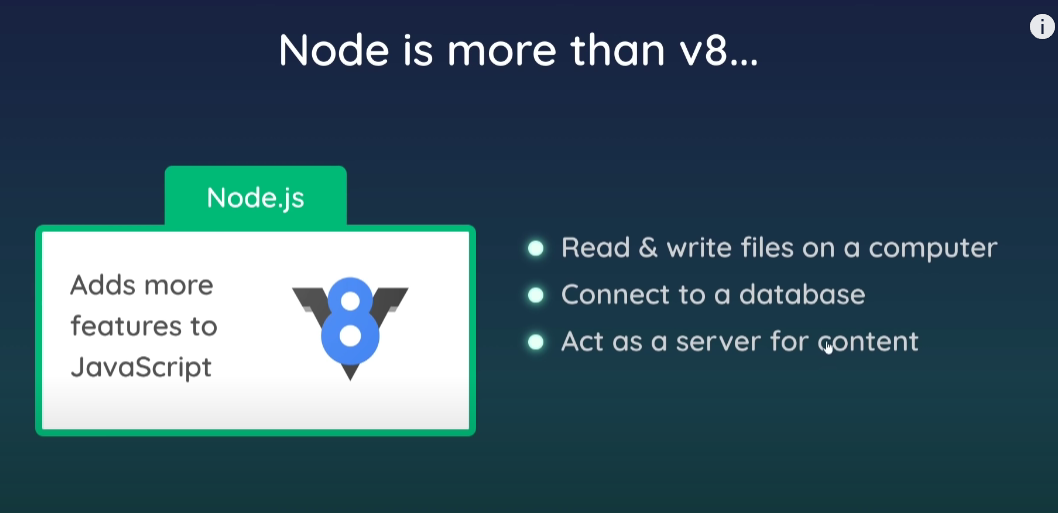
# Node.JS and Express.JS 🡪



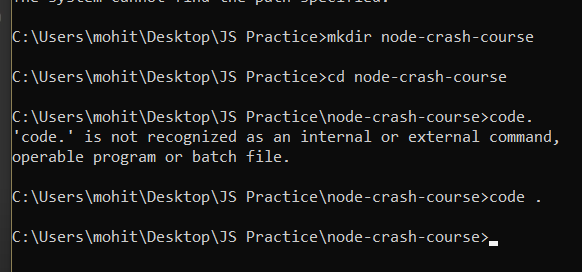




Node.js basically can run JS on our computer and not just on the browser.



Creating and opening a directory in code 🡪

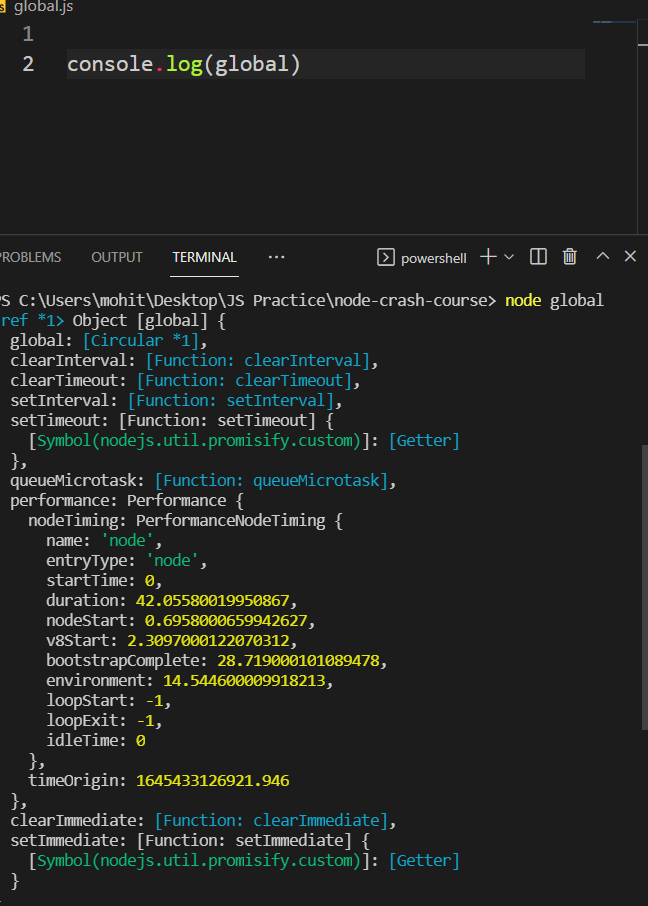
****

Basic Code 🡪



Global 🡪

Some of the objects are global ie they are present in global scope and we can use them anywhere.



Eg setTimeout() is a global function that is given. Use setTimeout() or global.setTimeout()

Same with setInterval()

\_\_dirname 🡪

This variable will give u the current directory where we are present.

C:\Users\mohit\Desktop\JS Practice\node-crash-course> node global

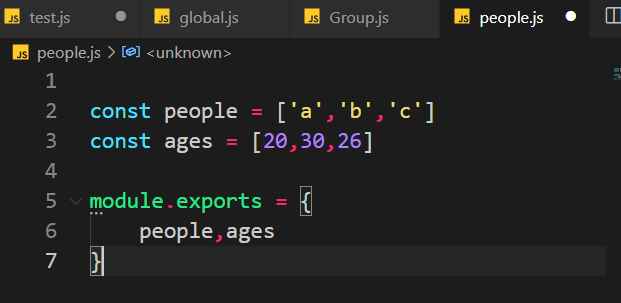
\_\_filename 🡪

This variable will give u the current directory but also the current filename.

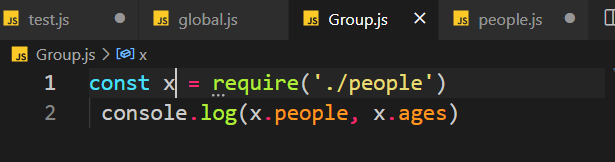
C:\Users\mohit\Desktop\JS Practice\node-crash-course\global.js

Modules and Exports 🡪

How to do that 🡪

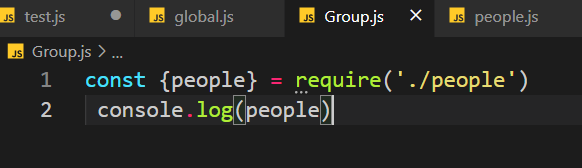


We can export anything using the module.exports as shown.



Take the imported values into as variable and then you can use the . to access individual parts as shown.

Another way 🡪



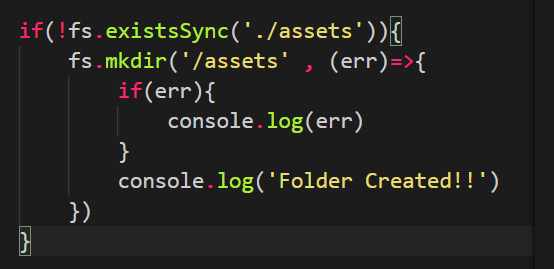
But here you have only imported people and not ages.

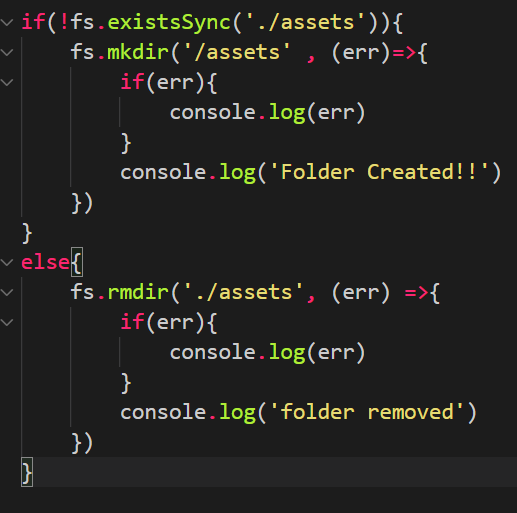
Many other core modules are present in the node which u can use by using require.

Read and write files 🡪

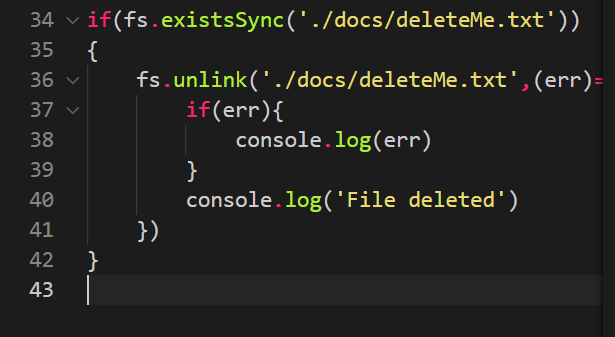


Creating a directory 🡪

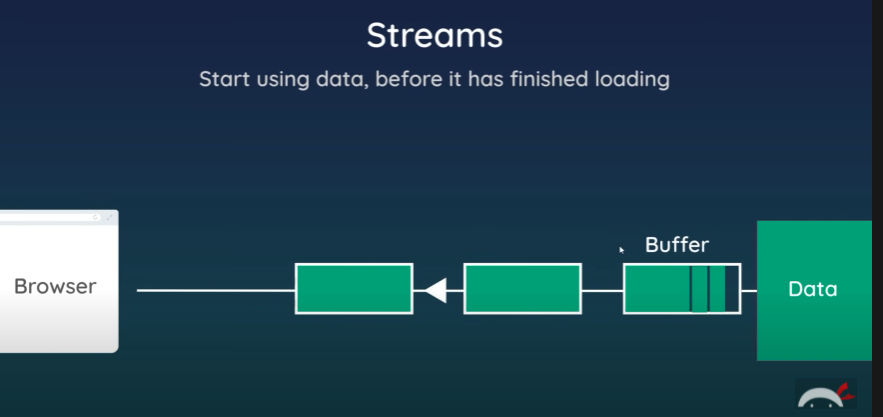




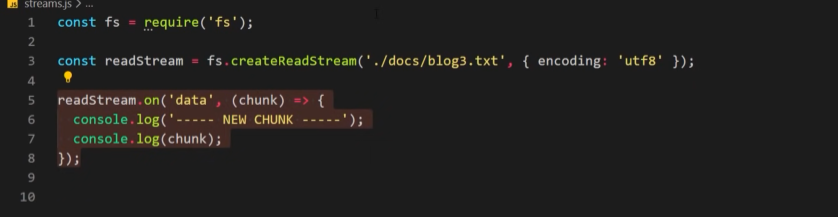
Delete a file 🡪



Streams 🡪 to read long data. We can start using data without fully loading the data.



Read stream 🡪



Write stream 🡪



Or use the below code to do the same things in just a single line of code 🡪



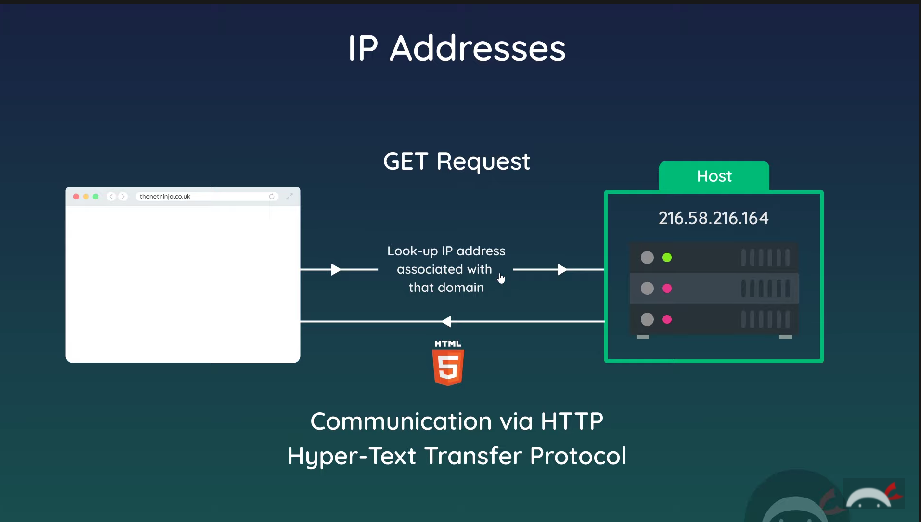
Client and Server 🡪

IP Addresses 🡪



Where you store you files for the website, you need to have its IP address and IP address is masked with a domain name. So you basically use a domain name.





Whenever we go to any link or website etc a get req is sent. HTTP protocol is used for this.

Creating Server 🡪

const http = require('http')

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

    console.log('request made')

})

So, server has been created and it takes 2 params, req as the request that comes and res which is the response which is sent back.

This callback function which logs request made will be called every time a request is made.

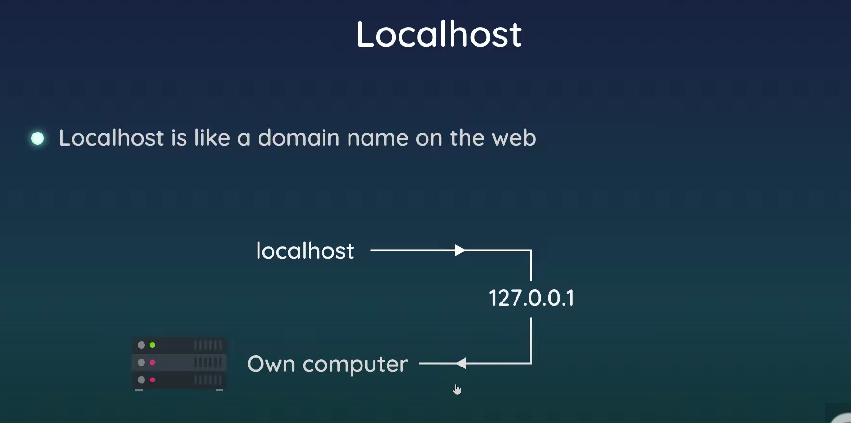
server.listen(3000, 'localhost', ()=>{

    console.log('listening for reqs at port 3000')

})

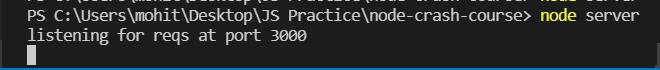
with this code the server is listening to requests at port number 3000

About localhost 🡪

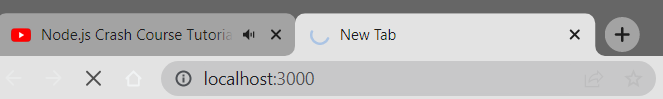




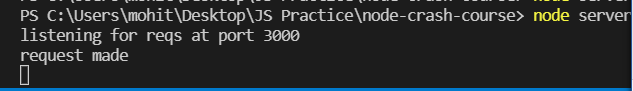
Every program will use a port number.



In Browser 🡪

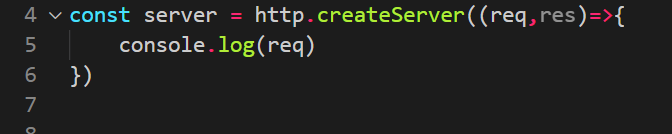


In Code 🡪

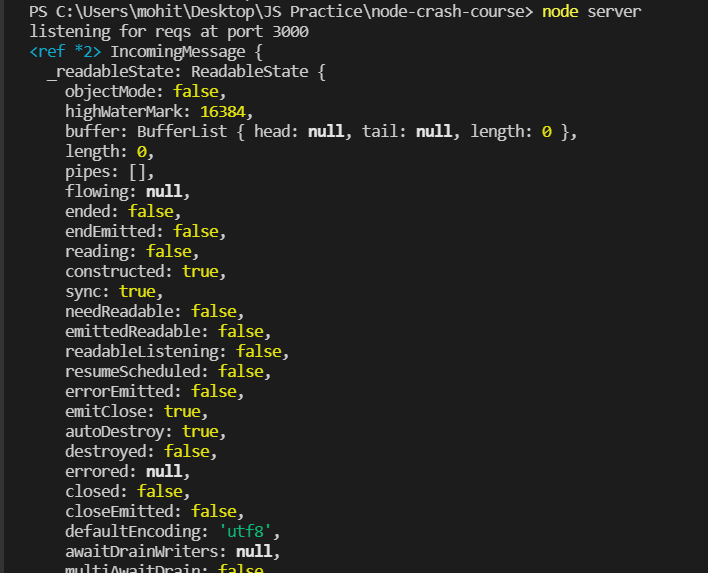


So, request has been made and we have listened to the request but we have not provided a response.

If I do below 🡪



Then req object is printed to the console when the request is made 🡪



Req.url 🡪 gives you the URL of the request.

Req.method 🡪 gives you the method of the req e.g. get or post etc.

How to set response 🡪

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

    console.log(req.url, req.method)

    //set header content type

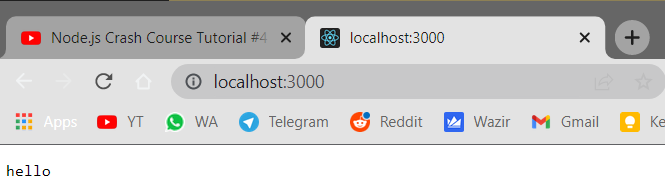
    res.setHeader('Content-Type', 'text/plain')

    res.write('hello')

    res.end()

})

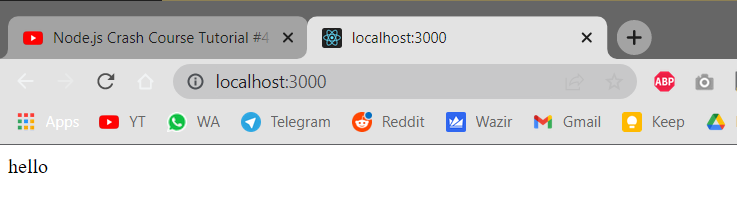
In the browser 🡪



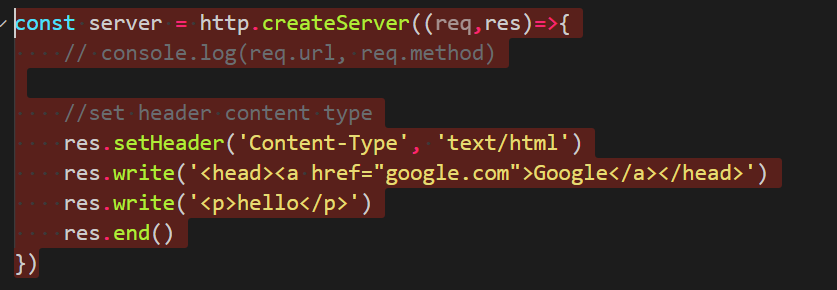
You can also send html as response as 🡪



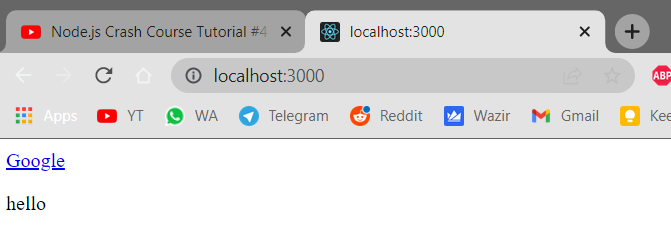
In browser 🡪



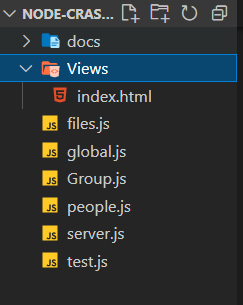
Code 🡪

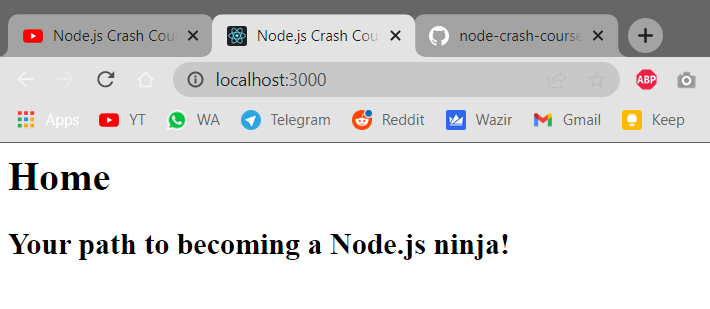
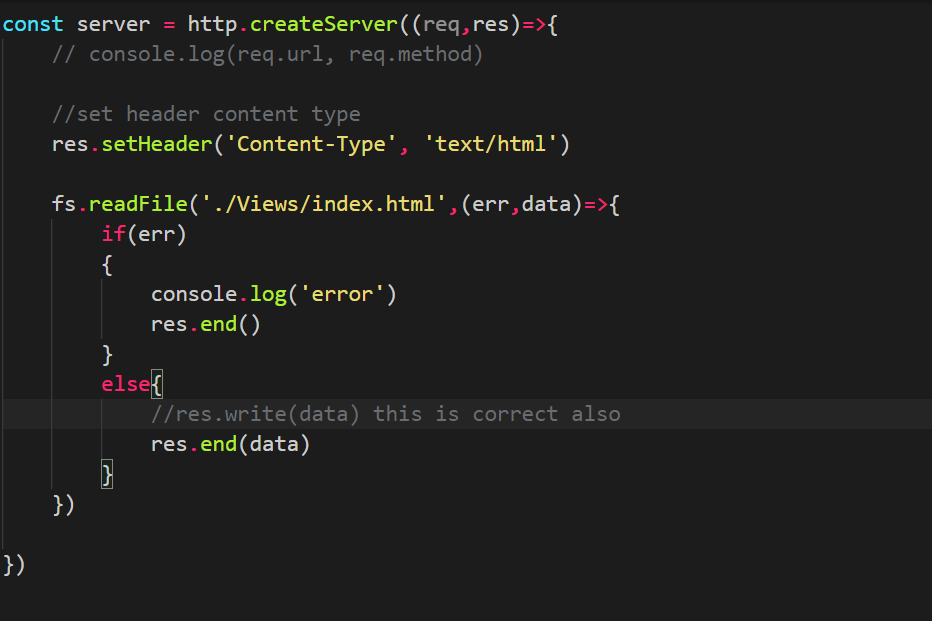


In browser 🡪

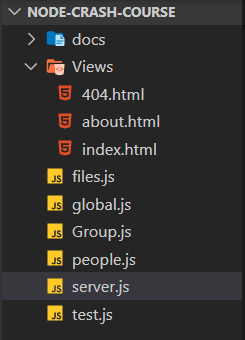


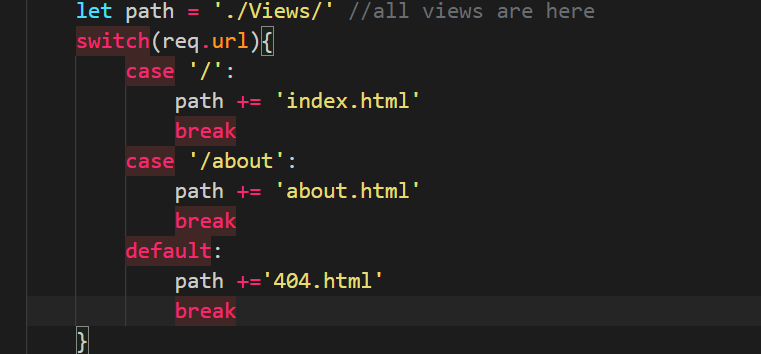
How to send a full html file as response 🡪

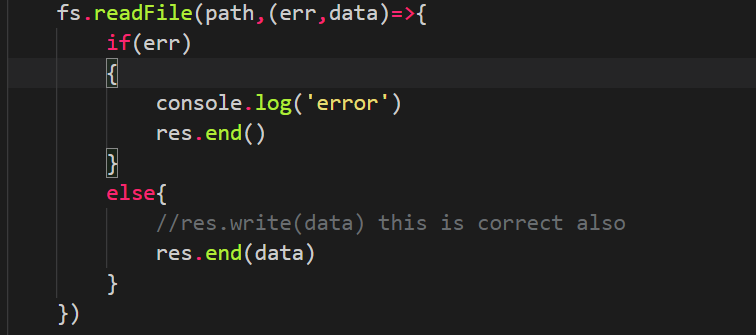


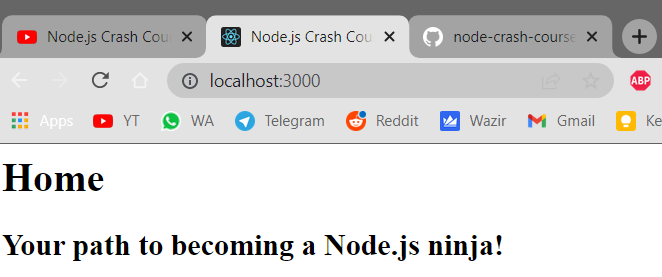
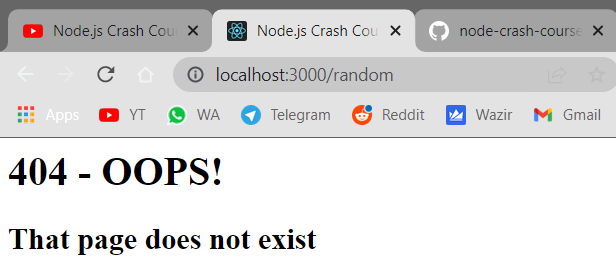


Basic Routing 🡪

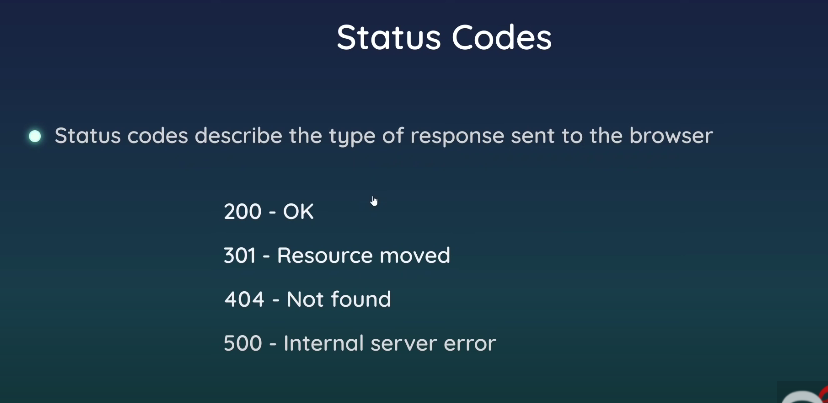


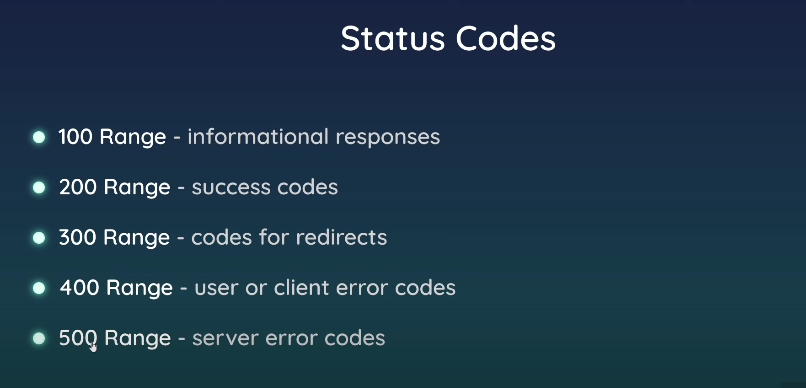


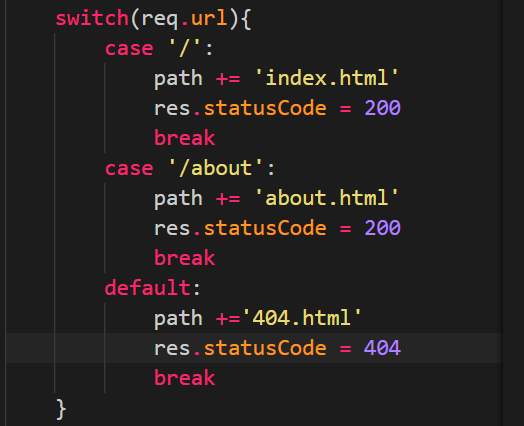


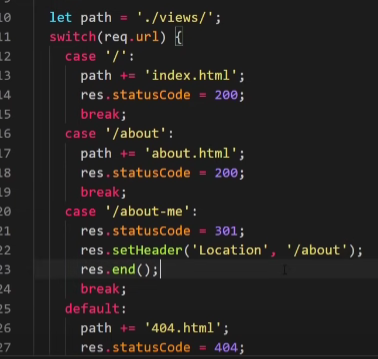
Status Codes 🡪







Redirects 🡪



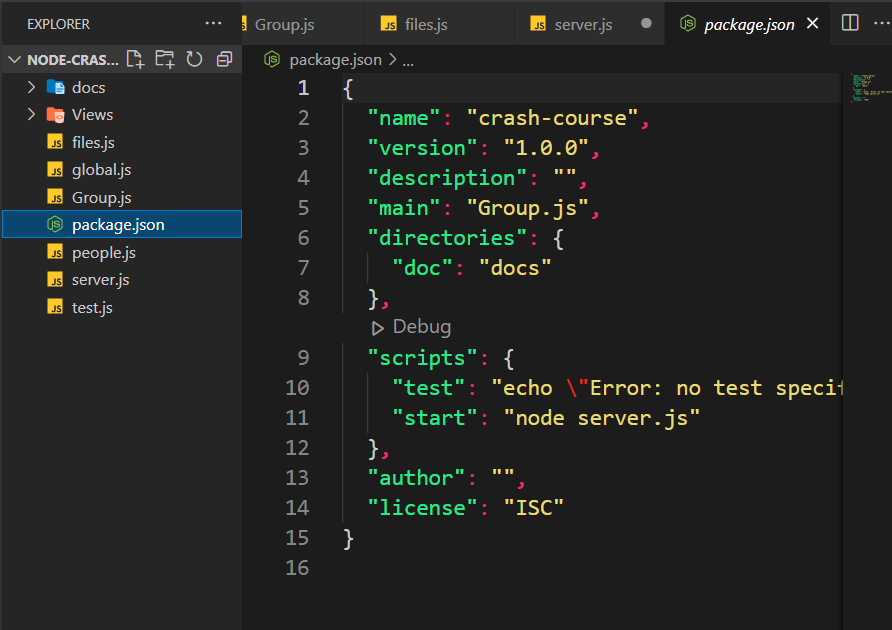
NPM 🡪

Install package as 🡪

npm install -g nodemon

replace nodeman with your package name.

Package.json file 🡪



Package-JSON

The package. json file is **the heart of any Node project**. It records important metadata about a project which is required before publishing to NPM, and also defines functional attributes of a project that npm uses to install dependencies, run scripts, and identify the entry point to our package.

Package-lock.json 🡪

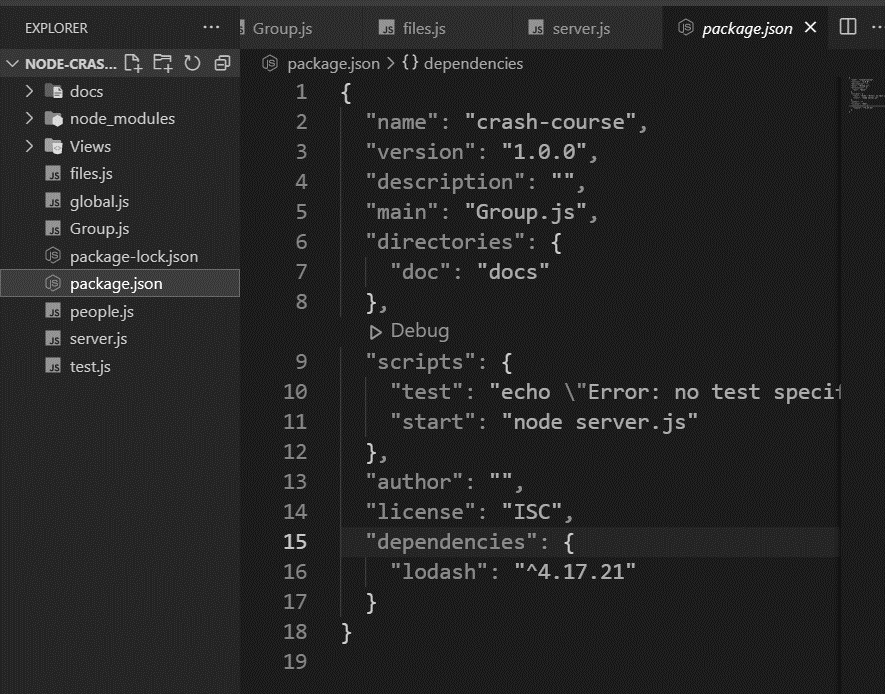
The goal of package-lock.json file is to keep track of the exact version of every package that is installed so that a product is 100% reproducible in the same way even if packages are updated by their maintainers

Lodash 🡪

It is an NPM library or package which gives you many utility functions which you can use.

Dependency 🡪

All the packages that you have used in your code. They are listed in package.json and inside that they are listed in dependencies.



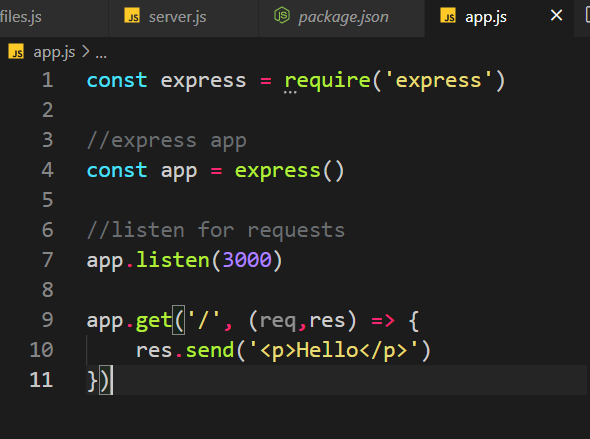
You can resolve all the dependencies by using 🡪

Npm i or npm install

This will install all the packages that are there in dependencies in packge.json.

**EXPRESS 🡪**

Makes building backend very easy.



Get() is used for get requests. Res.send() means what response we want to send back.

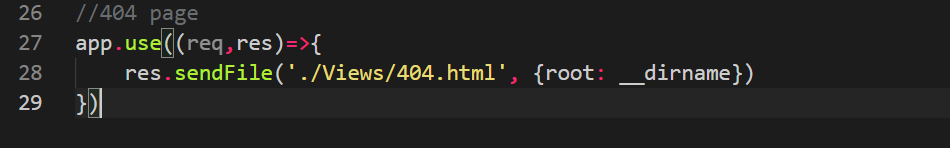
How to send a html file instead of code 🡪



How to redirect 🡪



404 page 🡪



If any of the above route is found then we go to that, if we don’t find it then we will go to these 404 routes. So this must always be the last one bc if it is above others then that will match with 404 and 404 will show and it will not go further then that.

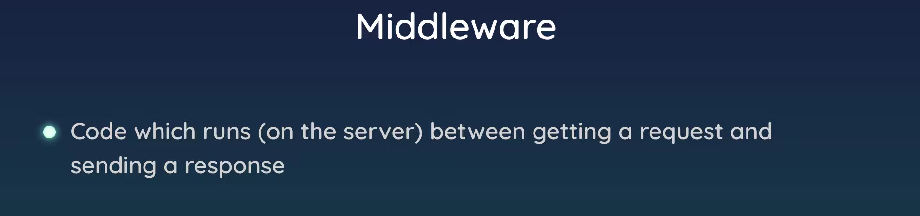
**EJS 🡪**

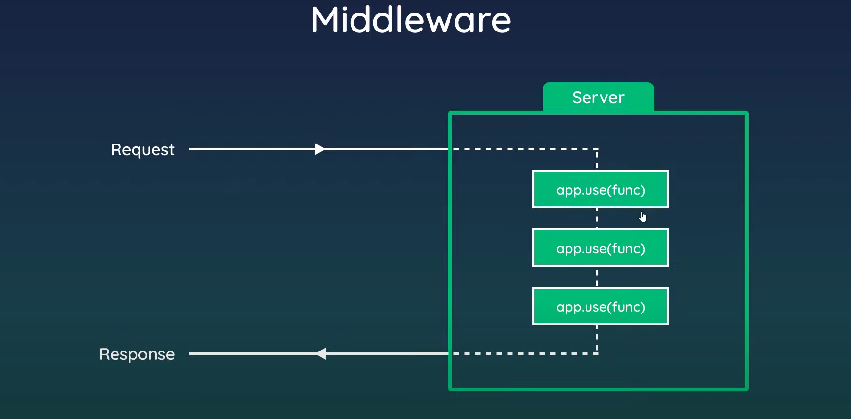
It is a view engine.

Set up 🡪

//register view engine

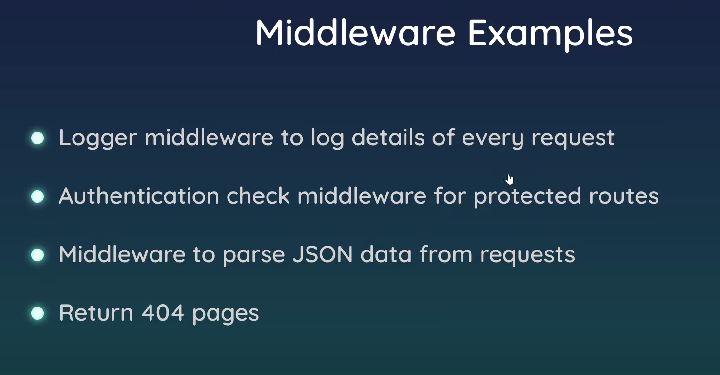
app.set('view engine', 'ejs')

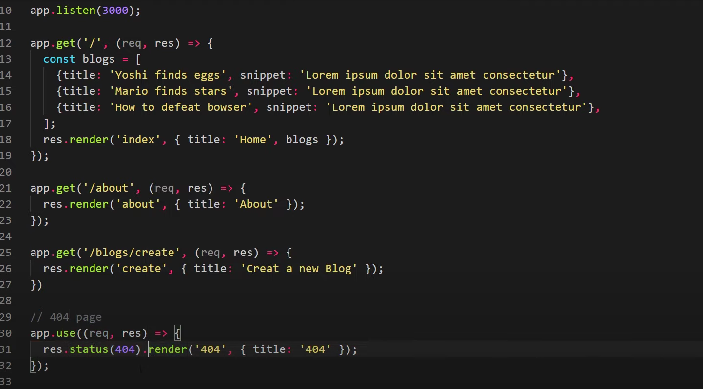




Technically get handlers etc are also middleware.

Uses 🡪

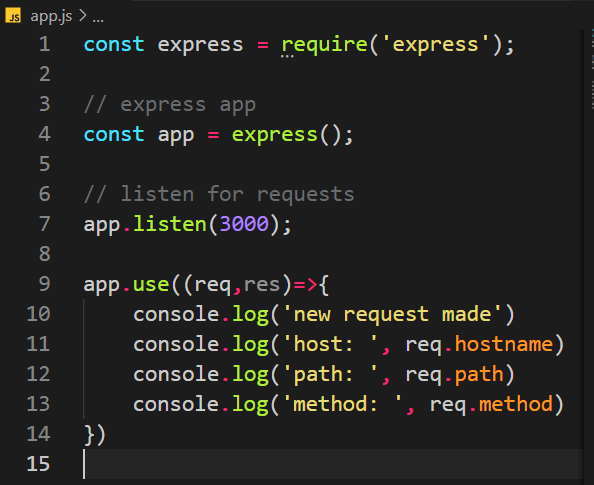




Here app.use() which is being used for 404 is a middleware. This code will run for every request but only if a response has not been sent yet.

If the code reaches the use() method then it will run and if the code doesn’t then it will not run.

Example 🡪



Here this code will run for every request that is made, but here the browser hangs because browser does not know where to go next.

Use next() 🡪

app.use((req,res,next)=>{

    console.log('new request made')

    console.log('host: ', req.hostname)

    console.log('path: ', req.path)

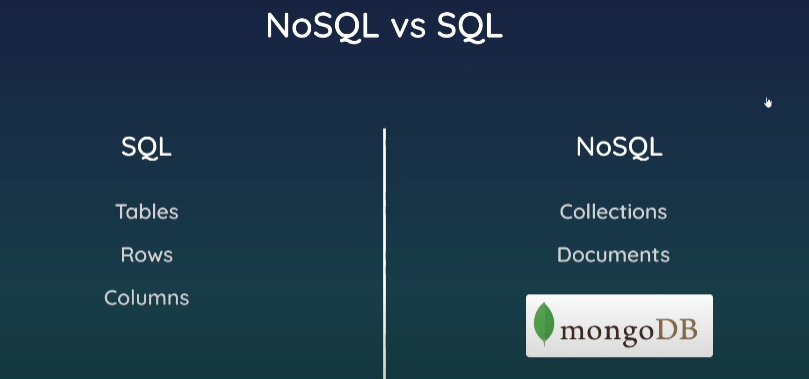
    console.log('method: ', req.method)

    next();

})

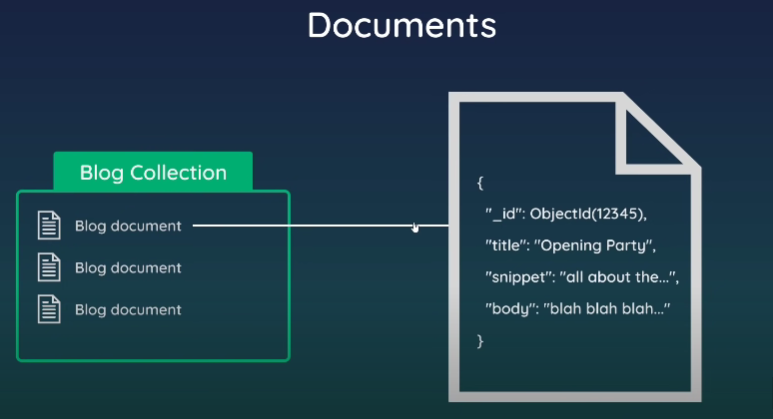
Make sure you reach the middleware; response must not have been sent already.

MongoDB 🡪

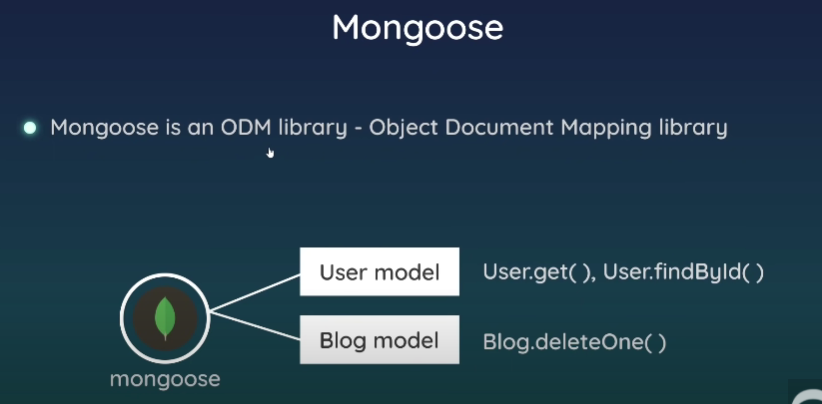


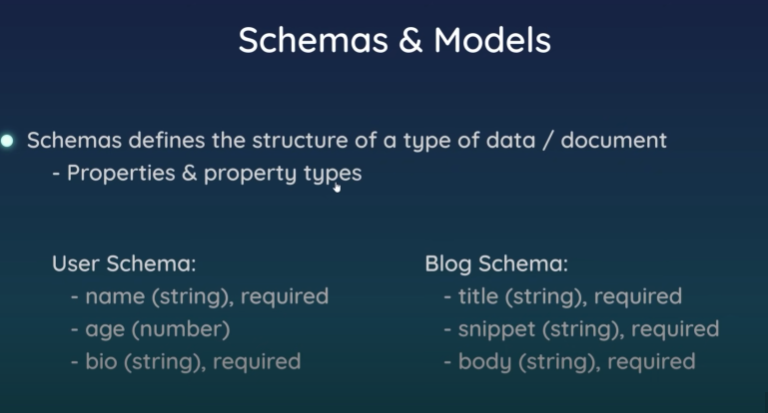


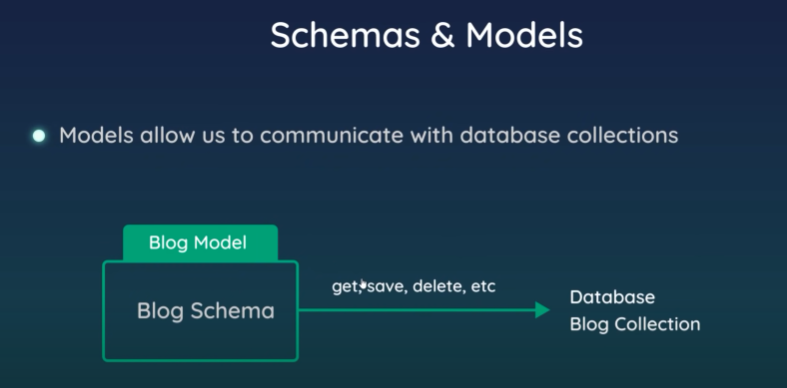
Collections are similar to tables in RDBMS.



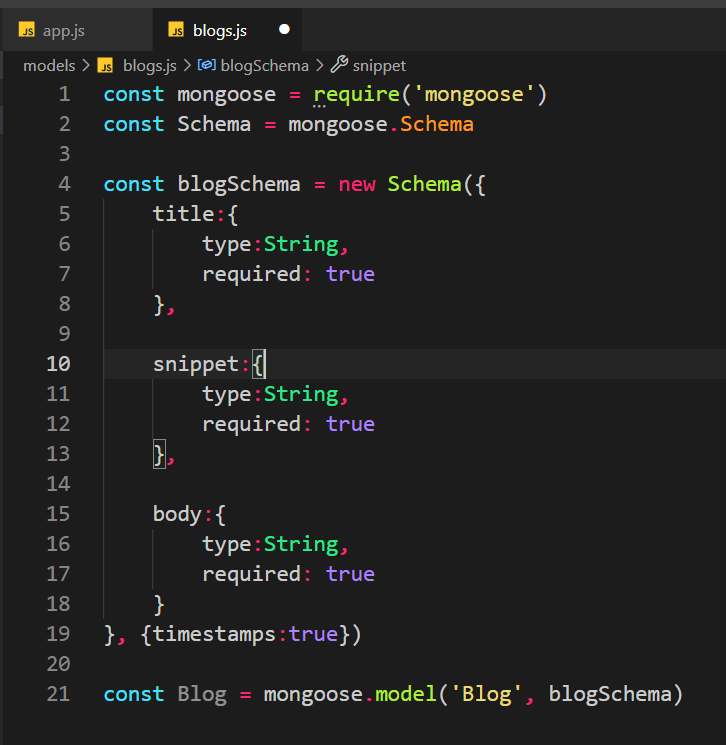
Data looks similar to JSON. They have key values pairs.







You have to create a schema to map a collection in the mongoDB, then create a model based on that. 🡪

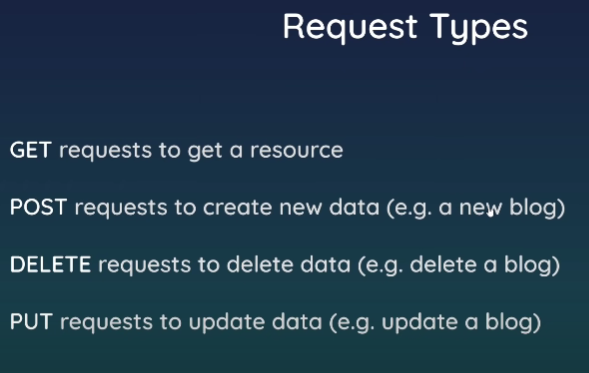


First argument in model() 🡪

Blog, this means the blogs collection in the mongoDB. Remember that s is added automatically.

Second argument means the schema it is based on which is blogSchema that we created above.

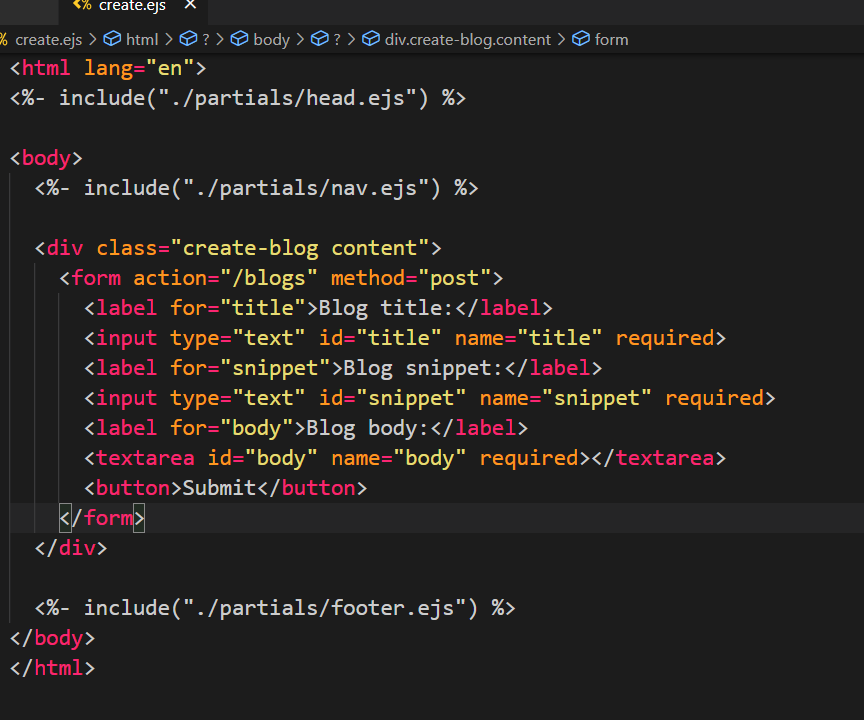
Requests 🡪





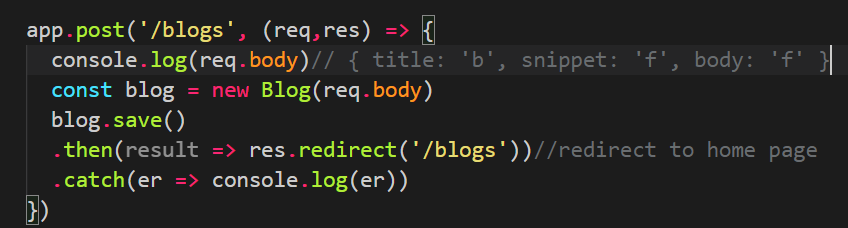
For Post requests 🡪

HTML form 🡪



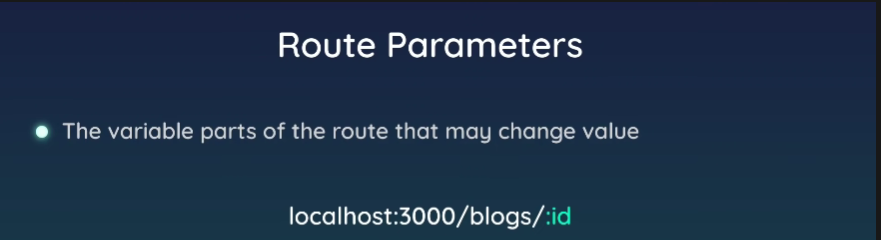
Use this middleware in app.js🡪





This middleware helps in getting data from the form.

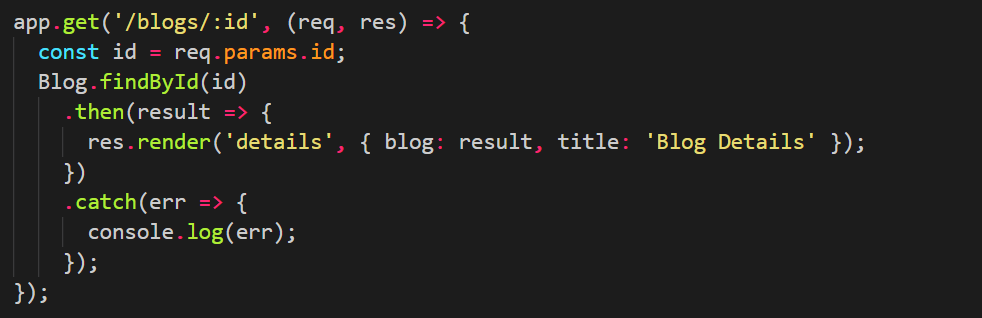
Req.body contains all the data of the form like title etc.



Id can change. So, it is a parameter.



We need to extract this to send the correct data.

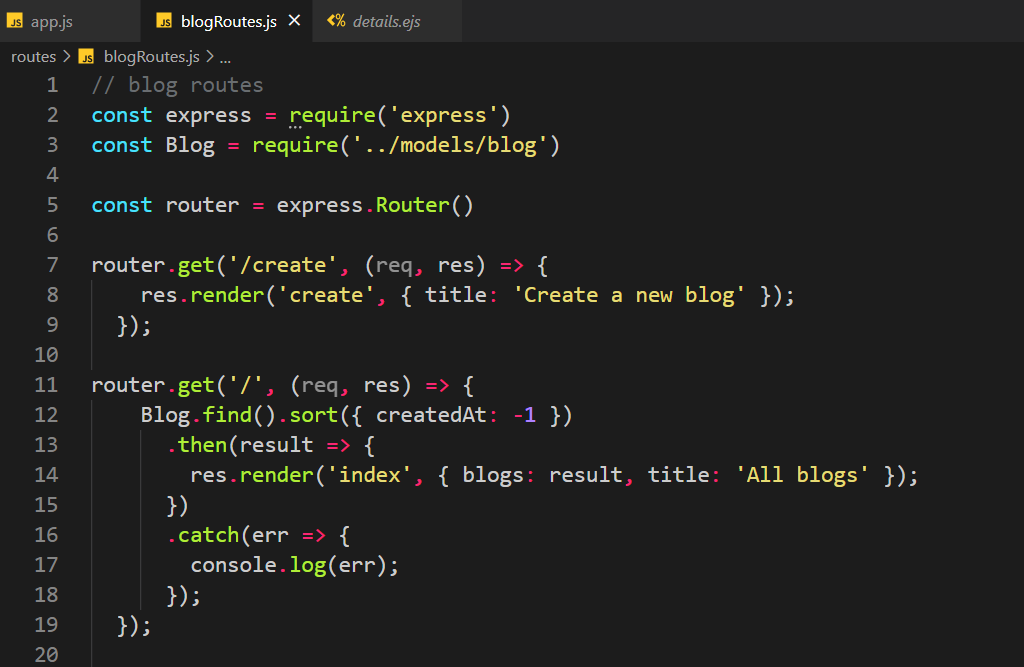


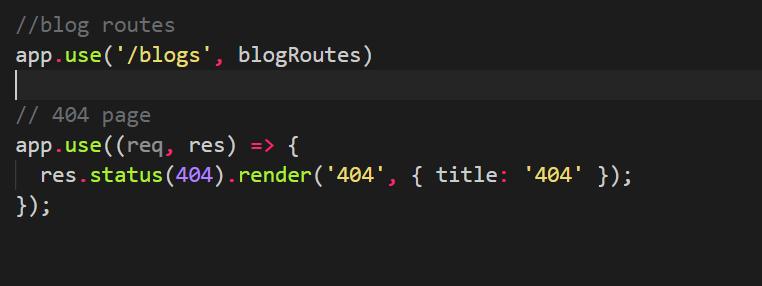
Express Router and MVC 🡪

Express Router 🡪

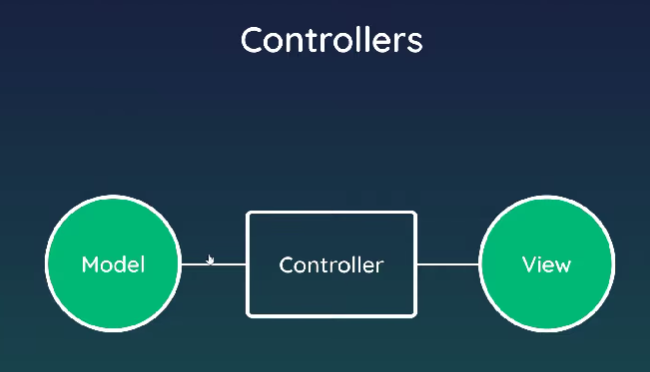
To maintain the routes efficiently and correctly.

* To Split the routes into different files and manage them
* Groups of routes are put together which are related to each other.
* Easier to update the router and everything modular.



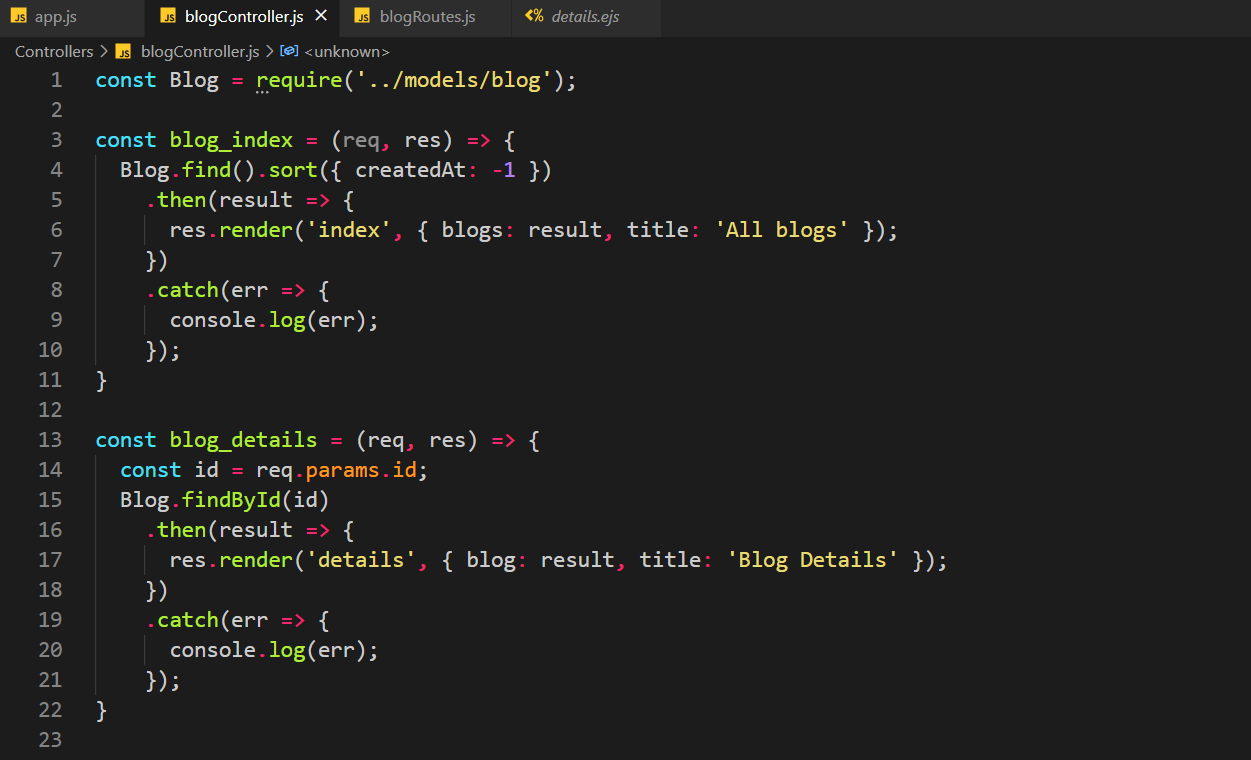


MVC 🡪



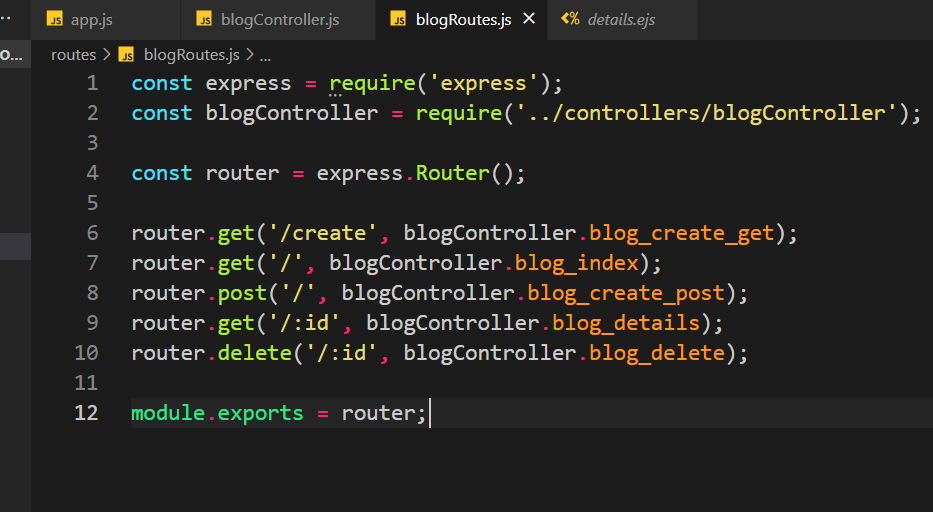
Create a controller that has all the functions that should be executed on get post etc requests.

Controller 🡪





And import that in routes and use as 🡪



Easier to read and update code.

Complete Project 🡪

