**Node JS:**

**Note:**

Node JS is **server environment** not programing language.

Node JS is mostly used for **API’s** but we can also create websites.

**History and More:**

First release in **May 27, 2009**.

Current version is **17**.

Written in **C, C++, Java Script**.

Node JS use **V8** engine of chrome and **V8** made in **C and C++.**

**Node JS / Java Script:**

Node JS and Java Script have same syntax.

|  |  |
| --- | --- |
| **Node JS** | **Java Script** |
| .Can connect to Server | . Can’t connect to server. |
| . Can Connect to DB. |  |
| . Run on the Server Side | . Run on the Browser. |

**Install and Setup Node JS.**

Download **Node JS** from nodejs.org**.**

Install **NPM** and **Node.**

**First Script with Node:**

Just simple script.

Var a = 20;

Var c= 30;

Var d =40;

Console.log(a + c + d);

Ans = 90

**Interview Questions:**

On browser **console.log** is a browser function.

On CMD **console.log** is not part of js but it’s module of node.

**Console.log** is used for debugging.

Types is:

Console.log.

Console.warn.

Console.error.

**Fundamentals of Node js:**

**Conditions, loops and arrays:**

You must need to know basics of JS for start node.

**How can we use other file code in first one:**

In second file use syntax for export file or code.

**Module.exports={**

**x:10, y:20**

**z:function(){**

**return 10;**

**}}**

In first file use code for import file.

**Const some-name = require(‘file-path’);**

Console.log(some-name);

**Interview Question:**

Where we can used **filter function;**

Used for find specific value from array. Always run on array.

Used is: **arr.filter((item)=>{**

**return item === 3**

**})**

**Core Module in Node JS.**

**What is core module?**

Some basic features is by default made in any language like console.log() is called core module.

And **global module** is that we not need to import for use that module like we don’t need to import console for use.

**Make basic server and print its output on server.**

**1: Import Http;**

Const http = require(‘http’);

**http** is used for handle request and response in node JS.

**2: http. createServer().listen(port-number);**

http.createServer().listen(4500);

here **createServer()** function accept **function as a parameter.**

**Like:**

http.createServer( (req, resp) =>{

resp.write(“hello this is response”);

resp.end();

} ).listen(4500);

**Run this file:**

Use **cmd command. (**node ./index.js**)**

**Package.json.**

**What is package.json?**

It keep your project details related to coding.

**How to make it?**

Run command **npm init.**

**Package-lock.json:**

Package-lock.json keeps the all details of packages. Like version, integrity etc.

**Interview Questions:**

**Node JS is single threading or multiple threading:**

Node JS is single threading mean its run **single command** at a time.

**Nodemon | time saving module.**

**What is Nodemon?**

Use for continuous run your node project.

**How to Install it.**

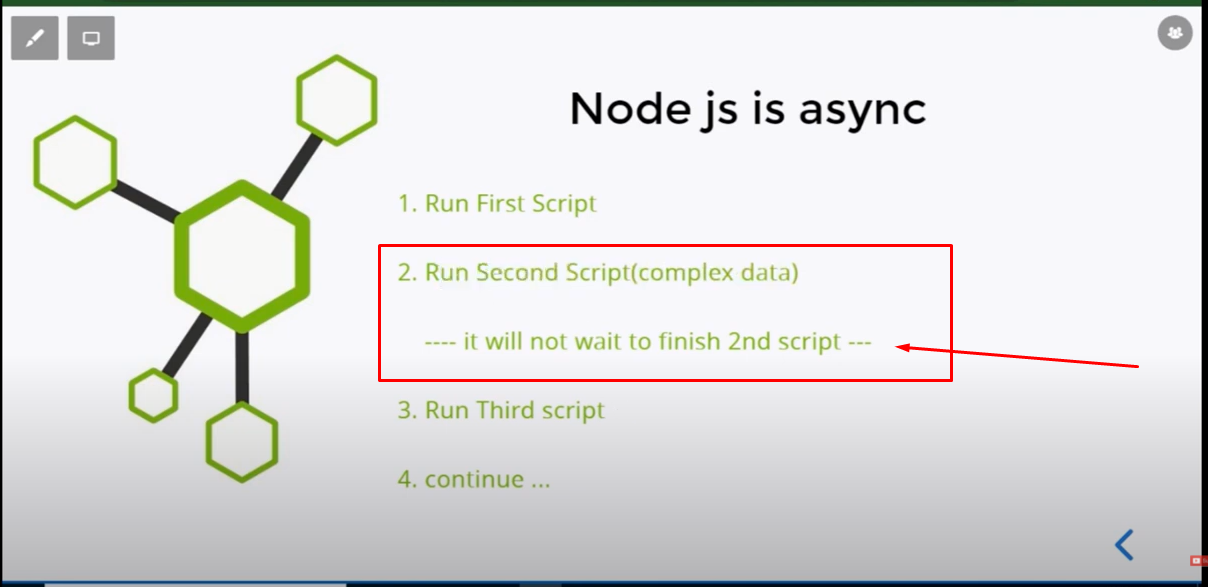
Command (**npm I nodemon -g**) –g mean for install globally.

**Interview Questions:**

**Node js is async or sync language?**

Node JS is async language. It’s never wait.

(In asynchronous operations, on the other hand, you can move to another task before the previous one finishes).



**Make a Simple API.**

**Data in same file:**

Here we use **writeHead** function for write status and application type.

const http =  require('http');

http.createServer((req , resp)=>{

    resp.writeHead(200, {'Content-Type': 'application\json'});

    resp.write(JSON.stringify({name:'Usman', email:'usmanshan@gmail.com'}));

    resp.end();

}).listen(8000)

**Data from another file:**

**1st file is:**

const data = require('./data');

const http =  require('http');

http.createServer((req , resp)=>{

    resp.writeHead(200, {'Content-Type': 'application\json'});

    resp.write(JSON.stringify(data));

    resp.end();

}).listen(8000)

**2nd File is:**

const data = [

    {name:'Usman', email:'usmanshan@gmail.com'},

    {name:'Shan', email:'shan@gmail.com'},

    {name:'Mohsin', email:'mohsin@gmail.com'}

];

module.exports = data;

**Interview Questions:**

201 is and 200, 400, 500 is different types of status.

**Input from Command Line:**

**How can we pass input through cmd:**

**(**$ node ./index.js 100) here 100 is an input.

**How can we get input from cmd:**

Console.log(process.argv):

It gives three input and third one is that that we pass through cmd input.

$ node ./index.js hello

[

'C:\\Program Files\\nodejs\\node.exe',

'E:\\nodeJs\\projects\\first\\index.js',

'hello'

]

**Show File List:**

**Make files in folder.**

For create files or modified files we need to import **file system in index.js file.**

Const fs = require(‘fs’);

**Create new file using fs:**

Fs.writeFileSync(‘file-name-with-extension’, ‘text inside file’);

**Create multiple files:**

Import **path module** in index.js file.

Const path = require(‘path’);

**Get the path of current Directory:**

Const dirPath = path.join(\_\_dirname);

**Return:** E:\nodeJs\projects

**Add folder with current getting path:**

Const dirPath = path.join(\_\_dirname, ‘folder-name’);

**Return:** E:\nodeJs\projects\folder-name

**Get Files:**

Fs.readdir(directory-path, (err , files)=>{

Console.log(files);

})

**Fs.readdir()** return arrays of all files in given directory.

**Crud with Files System:**

**Read file:**

Fs.readFile(filePath, ,(err, item)=>{

Consle.log(item);

})

Fs.readFile(filePath, **‘utf8’** ,(err, item)=>{

Consle.log(item);

})

**Update or Append File:**

Fs .appendFile(filePath, ‘text for append in this file’, (err)=>{

If(err){

Console.log(‘file Is updated’);

}

});

**Rename File:**

Fs.rename(filePath, ‘${dirPath}/fruit.txt’, (err)=>{

If(err){

Console.log(‘file Is renamed);

}

})

**Delete File:**

fs.unlinkSync(‘${dirPath}/fruit.txt’);

**interview Questions:**

**What is Buffer?**

Buffer is called temporary memory location.

**Asynchronous Basics:**

**Handle Asynchronous Behaviors:**

We need **promises** for those logics or functions who take more time to expectation.

Let a = 10;

Let b =20;

Let waitingData = new Promise( (resolve , reject) => {

setTimeout( () =>{

b=30;

resolve(30);

} , 2000)

} )

waitingData.then( (data) =>{

b = data;

Console.log(a+data);

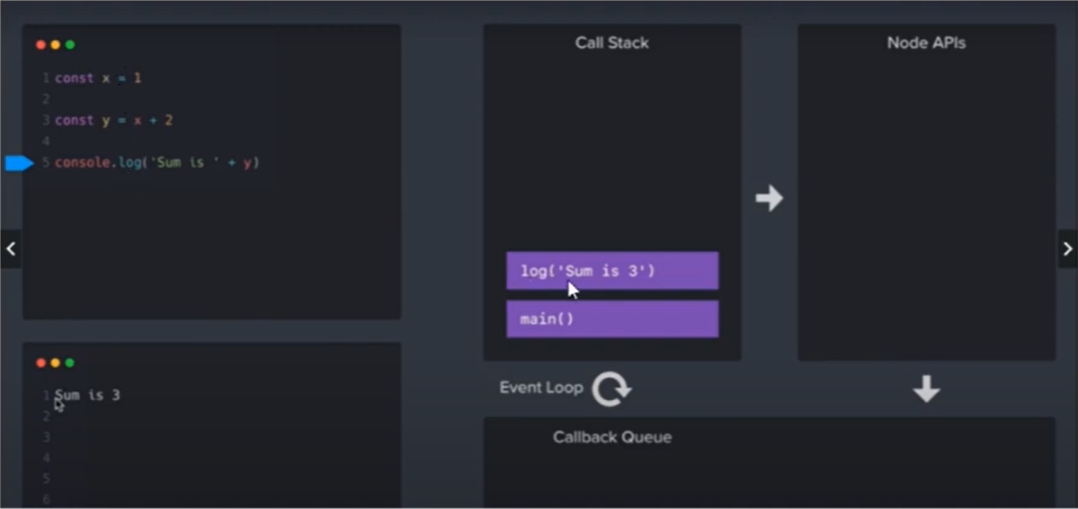
})

**How Node JS work:**

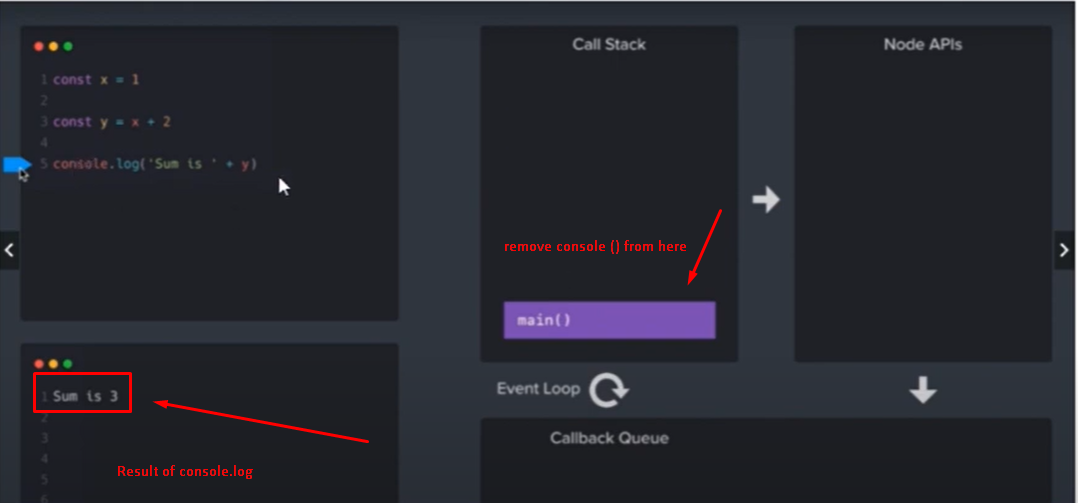
**1:Call Stack.**

When we start execute our code then every function registered in call stack. In simple word call stack registered our functions before execute. Mean which function will call **first and last**.

**Main()** is by default in call stack. In blew example **.log** is a also function so it’s register with main function.



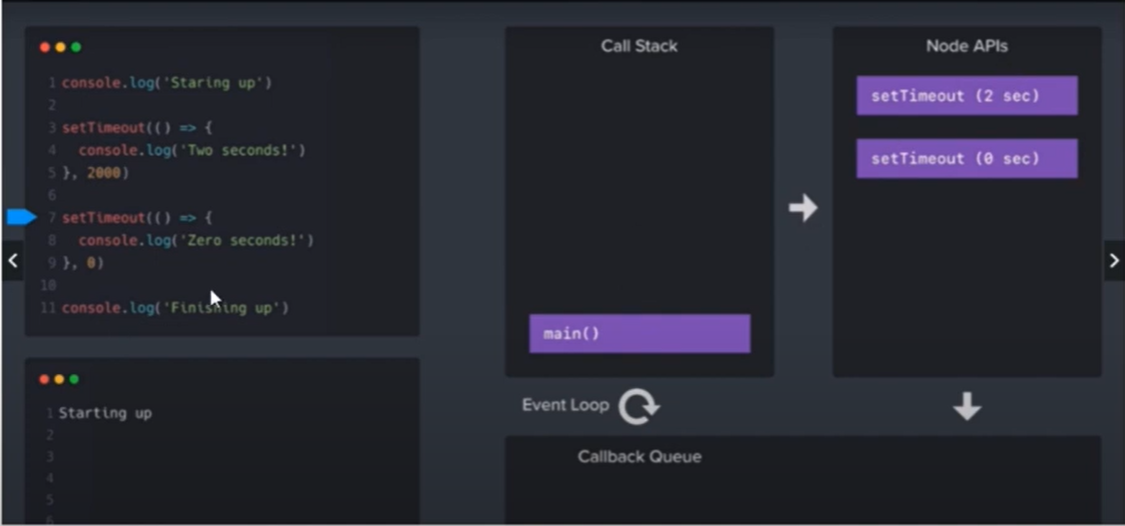
When function execution is end then it automatically remove from call stack. And main function is remove from call stack when no more function is remaining in call stack.



**2: Node APIs.**

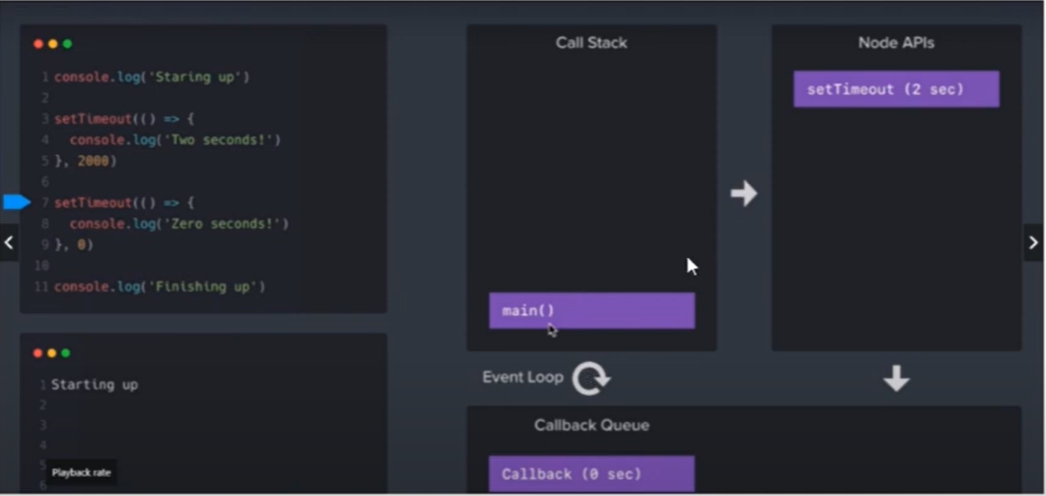
Anything that we inherit from **C, C++** it first registered in **Node APIs.** Here **setTimeOut()** is an C++ inherit then it will registered in **Node APIs.**

Node JS is 92% JS based and 8% C and C++;

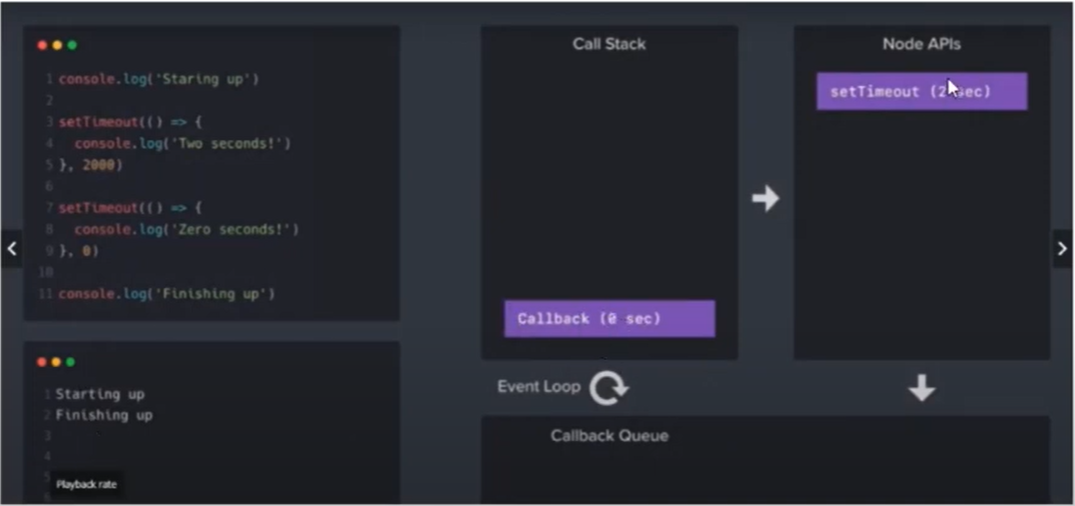


**3:Callback Queue.**

**Node APIs** send less time functions in **Callback Queue.**



**Node APIs** need empty stack so it comes in stack when stack is empty in upper example when both console the first one and the last one is executed then no any function remaining in **Call Stack** then main function also remove from it after that **Node APIs** registered in **Call Stack.**



**What is Express JS:**

Express JS is a framework of **Node JS**.

**Install Express:**

Command ( npm install express );

**Import Express in index.js;**

Const express = require(‘express’);

Const app = express();

**Get Method of Express;**

Get method provides us **routes;**

**For home page**

App.get(‘’, (req , resp) =>{

Resp.send(‘hello this is home page’);

} )

**About us page:**

App.get(‘/about’ , (req, resp) => {

Resp.send(“hello this is about page”);

})

**Assign Port for run this code;**

App.listen(5000);

**Render HTML and JSON:**

**Apply HTML:**

App.get(‘’, (req , resp) =>{

Resp.send(‘<h1>hello this is home page<h1>’);

} )

**Render JSON:**

App.get(‘’, (req , resp) =>{

Resp.send(

Name:’Usman’,

Email:’usman@gmail.com’

);

} )

We can also use **` `** for render html in browser page.

**Make HTML pages in folder and display in node applications:**

**Create Folder and html files in it.**

**Load folder in index.js.**

Const path = require(‘path’);

Const publicPath = path.join(\_\_dirname, ‘folder\_name’);

//now we successfully access the folder where html files exist.

App.use(express.static(publicPath));

// now we can access the file paste in publicPath folder

//by localhost:portNo/fileName.html

**Path() module** help you to access the folder of your project.

**Static method** help you to load static pages or static content.

**Remove Extensions from URL’s:**

App.get(‘’, (req, resp)=>{

Resp.sendFile(`${publicPath}/index.html`);

//here public path mean folder path where files exist.

})

**404 Page:**

App.get(‘\*’, (req, resp)=>{

Resp.sendFile(`${publicPath}/error.html`);

//here we use \* mean if url not match then 404 page show

})

**How can we load file in get methods.**

We can load file in get method by using **sendFile()** as use in above examples.

**Template Engine:**

Use for make dynamically pages and we need to install it as same as we install other npm packages it’s also npm package.

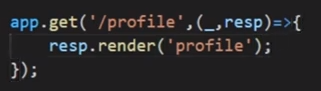
**Install ejs template engine.**

1: 1st we need to set engine.

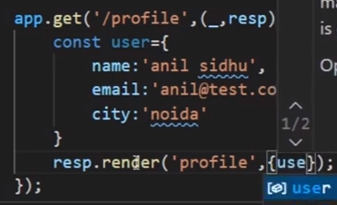


2: Make new folder with name **views** and make new page with name **profile.ejs** its extension should be **ejs.**

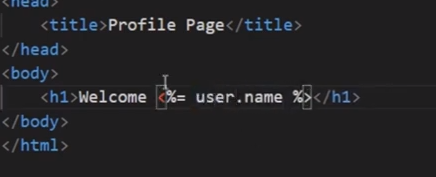
3: call profile page in **index.js** file by using **render** method .



4: Send data to profile page.



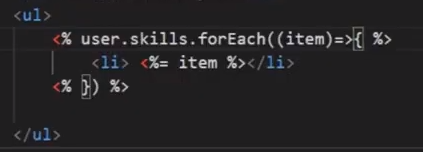
5: Get data in profile page.



**Dynamic Pages:**

We make it using template engine like **ejs** as show in above example.

**How can make loops in dynamic pages:**



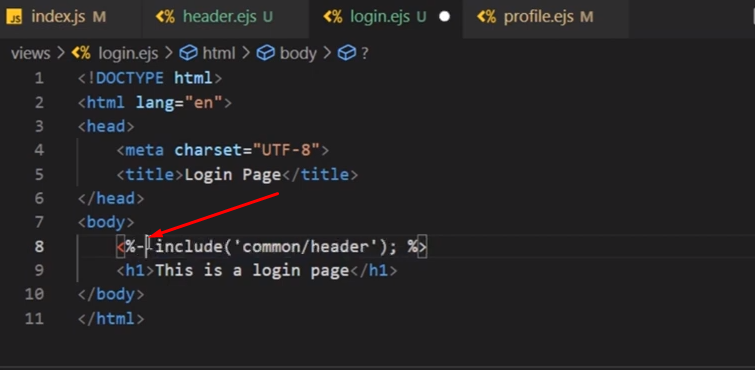
**Common Headers in different files:**

Create header file with extension **ejs. Header.ejs.** and put html code in here.

You can only use this file in only **.ejs** extensions pages.

**Use Header In our File:**

Must use – when you import header in your file.



**Middleware:**

**What is middleware:**

Functions which can use with **routes** with the help of **middleware** we can access and modify **request and response.** Also can use for authentications and filter request and response.

**Make Middleware:**

Next => basically use as a function proceed route that you call.

Const reqFilter = (req, resp, next)=>{

Console.log(“requFilter”);

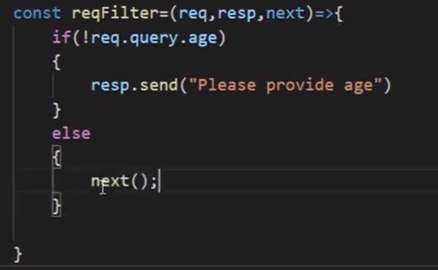
Next();

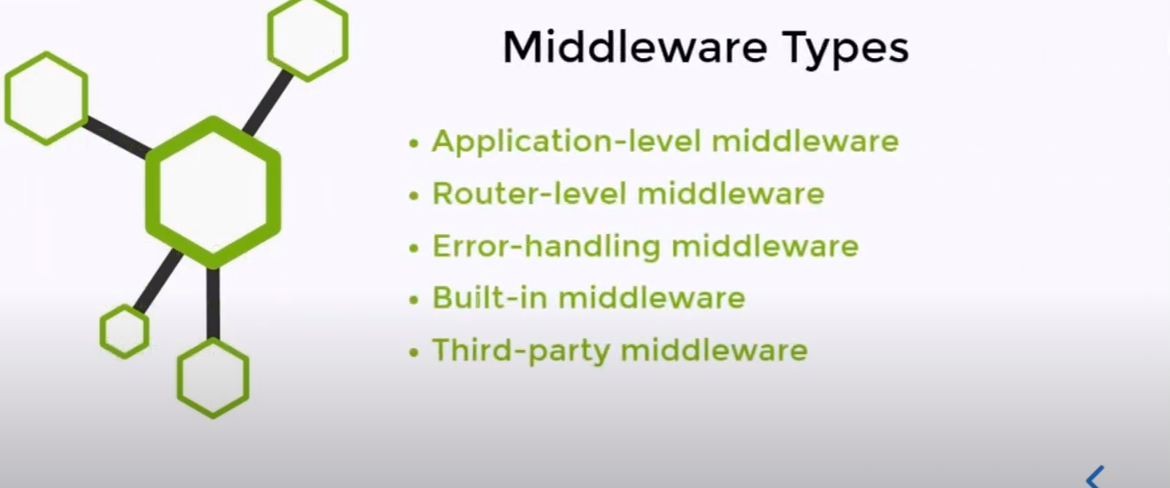
}

App.use(reqFilter);

**Basic Use.**

Check user age from route and get age by **req.query.age.**





**Route Level Middleware.**

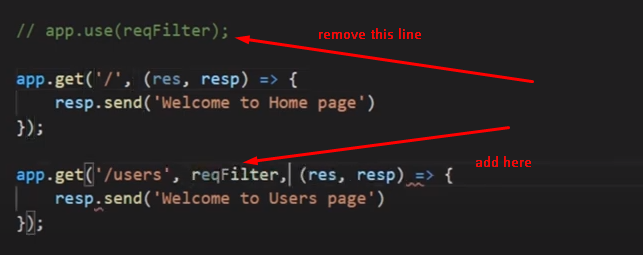
Use for apply middleware on specific or wanted routes.

Middleware which can work on single route and as well as group of routes or on all routes is called **route level middleware.** The above middleware that we use **is called application level middleware.** Mean which run on all routes of applications.

**How can we implement Route level middleware.**

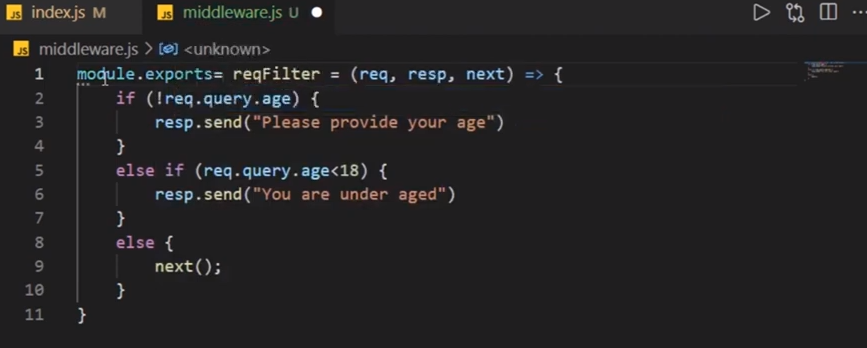
1: remove application middleware.

2: and add where you want to use.

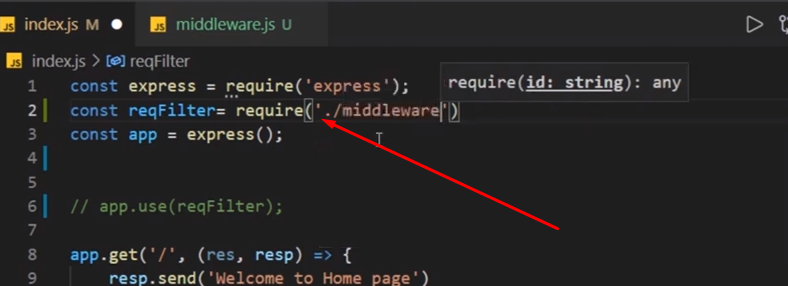


**Make middleware in different file:**

New file

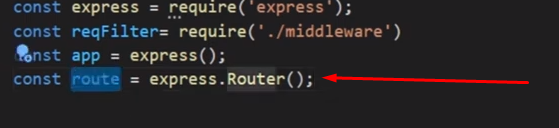


File where we want to use it.

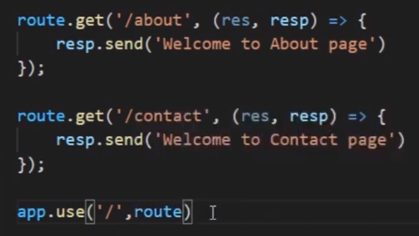


**Add on groups of routs:**

1: Import router Middleware.



2: Apply middleware on routes.



3: Also add this line on top:



**OS Module in Node JS.**

Operating system module its use where you want to know information about your system. Like want to see system ram or etc.

It’s used if we want to add different conditions on windows and mac.

**Import OS module:**

Const os = require(“os”);

Console.log(os.arch());

// gives your system architecture like 32 or 64 bit

Console.log(os.freemem()/(1024\*1024\*1024));

//gives how much ram is free in your system it gives result in bites so we need to divide it.

Console.log(os.totalmem());

Console.log(os.hostname());

//givers system hostname;

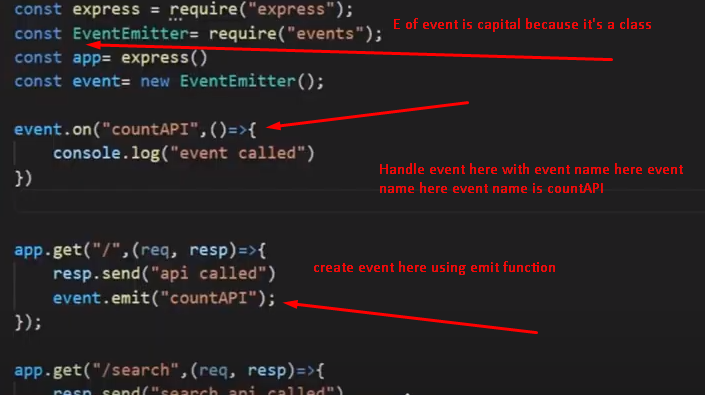
Console.log(os.platform());

//tells which platform we use window or mac.

Console.log(os.userInfor());

**Events and Event Emitter in Node JS.**

Almost everything in node JS is event or event emitter.

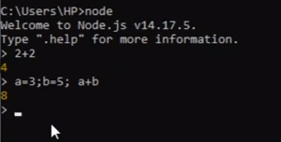


**REPL in Node JS:**

**(**Read, Evaluations, Print, Loop**);**

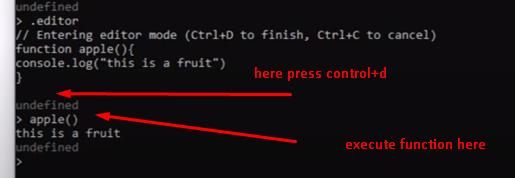
**REPL** is an command line tool of node JS with the help of this you can run code of node JS or javascript on command line.

When you open **cmd** and write **node** the you enter in **REPL.**



We can use command **.editor** for enter in editor mode of cmd.

Now press **control+d** for stop coding. Then you can execute functions.



**Node JS connect with MySQL:**

**Install MySQL Package in Project:**

npm install mysql

**Use MySQL:**

Load mysql => const mysql = require(“mysql”);

Connection => const con = mysql.createConnection({

host: ‘localhost’,

user:’root’,

password:’’,

database:’db-name’

})

**Check Connection is successful:**

con.connect( (err) =>{

if(err){

Console.log(“error”);

}

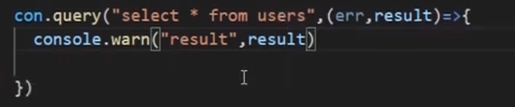
else{

console.lo(“connected”);

}

})

**Fetch data from database:**



**Node JS GET API with MySQL:**

**1: Create config.js file:**

Create config file on root folder.

And create connection in it.

const mysql = require("mysql");

const con = mysql.createConnection({

    host:'localhost',

    user:'roor',

    password:'',

    database:'nodefirst'

})

con.connect((err)=>{

    if(err){

        console.log("Connection error with Database");

    }

})

module.exports = con;

**2: Load need files and module in index.js.**

After Create config file import config and other related files in **index.js** file and run query that you want.

const express = require('express');

const con = require("./config");

const app = express();

app.get("/user", (req, resp)=>{

    con.query("select \* from user", (err, result)=>{

        if(err){

            resp.send([{status:false,messgae:"some error"}]);

        }

        else{

            resp.send(result);

        }

    })

});

app.listen(8000);

**Node JS POST API with MySQL:**

**Steps:**

1: Make route same as we created up for get result with **post method**.

app.post("/user", (req, resp)=>{

});

2: Insert Data.

First we insert static data.

app.post("/user", (req, resp)=>{

    const data = {name:"Aqeel", email:"aqeel@gmail.com", address:"Okara"};

    con.query("Insert INTO user SET ?" , data , (error , result , fields) =>{

        if(error) throw error;

        resp.send(result);

    } )

});

3: Insert dynamic data from postman:

First we need to Parse data into json formate:

app.use(express.json());

Now we need to get data from postman by using **request** parameter.

const express = require('express');

const con = require("./config");

const app = express();

app.use(express.json());

app.post("/user", (req, resp)=>{

    const data = req.body;

    con.query("Insert INTO user SET ?" , data , (error , result , fields) =>{

        if(error) throw error;

        //resp.send({status:true, messgae:"user created successfully"});

        resp.send(result);

    } )

});

app.listen(8000);

**Node JS PUT API with MySQL:**

Put API is used for update.

**Steps:**

Create route for this with **put method.**

app.put("/user", (req, resp)=>{

})

Update by static data:

app.put("/user", (req, resp)=>{

    const data = ["Mohsin", "mohsin@gmail.com", "lahore", 5];

    con.query("Update user SET name = ?, email = ? , address = ? Where id = ?" , data , (error , result , fields) => {

        if(error) throw error;

        //resp.send({status:true, messgae:"user updated successfully"});

        resp.send(result);

    })

})

**For dynamic data:**

For dynamic data first we need to send id in url and get id from url.

And for data first of all we need to **parse data** as we done in **post method.**

app.use(express.json());

After data fetch data and id.

app.put("/user/:id", (req, resp)=>{

    const data = [req.body.name, req.body.email, req.body.address, req.params.id];

    con.query("Update user SET name = ?, email = ? , address = ? Where id = ?" , data , (error , result , fields) => {

        if(error) throw error;

        //resp.send({status:true, messgae:"user updated successfully"});

        resp.send(result);

    })

})

Here first we pass id in URL and then get id by **req.param.id** and get data by **req.body.name.**

**Node JS PUT API with MySQL:**

Same upper steps and route with **delete method.**

app.delete("/user/:id", (req, resp)=>{

    con.query("DELETE FROM user WHERE id = " +req.params.id , (error , result , fields) => {

        if(error) throw error;

        //resp.send({status:true, messgae:"user Deleted successfully"});

        resp.send(result);

    })

})