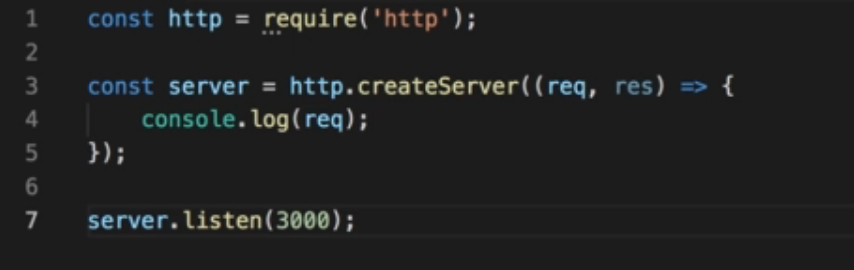
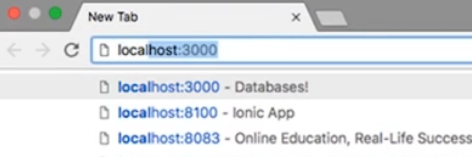
* How to write your own server in node Js

For that you have to get and save the path of the request some where which should be const for constant not changing 

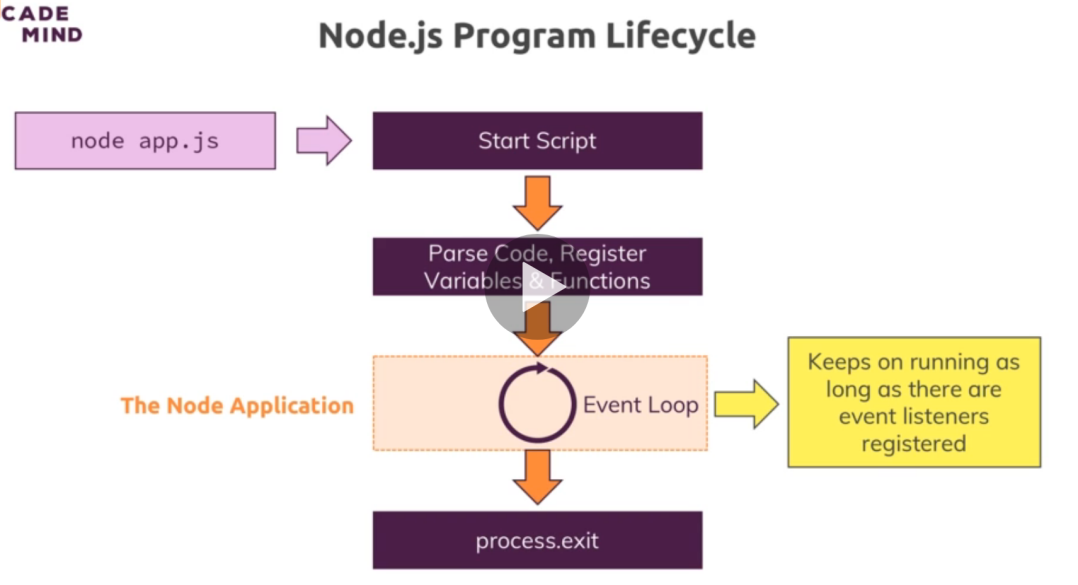
Since the path is not defined therefore we using http only( but the actual path starts with (./http))



Here above we get request from the required(http) which is the local host or our net server and call a method from that http to create Server and print the request of it and after saving in server we listen the server by passing the port value greater then 1000 which is safe and her we do not pass other argument hostname which by default pass the local host and this keeps on listening request ( so when write localhost 3000 ) in our browser the it will listen to the browser. And you can check it by node in vs code

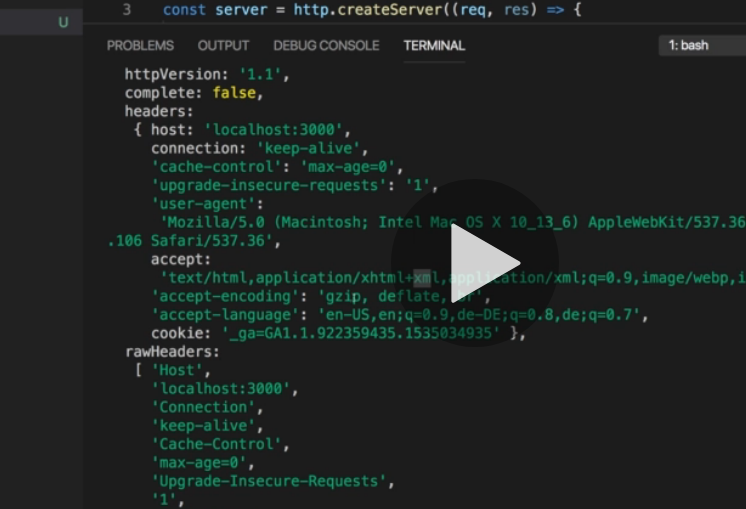


Node life cycle is when we call the node (file) it start scripting then parse and memorize the code and go to event loop and keeps on running until it listen the server it keeps on running so client can see the data so to exit it we use process.exit (when client do it.)

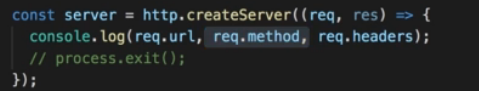


* Request in node js

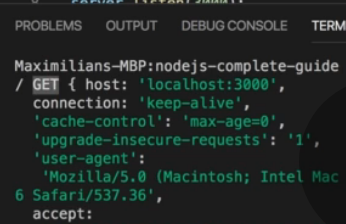
So when we console the request we get whole bunch of data, variables and methods etc



But we want only few to understand those are , request url, method and headers (headers contains the meta data)



So here is the output in terminal

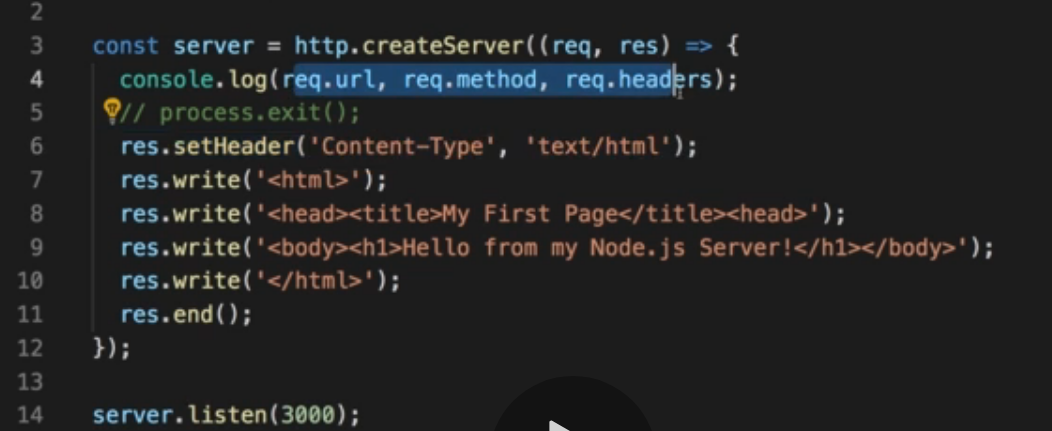


Which is (/) for the url because it is localhost (get) is for method and ( host : 🡪 below) is for header

* Response in Node js

So we can give response to the user/client side in node js can set header by res.setHeader(‘Content-Type, ‘text/html’) // so inside it we have to pass content Type.

We can also write a html code by our own inside the Js by res.write(); and when we have to push the our html code to the client that we have written have to use res.end() to end the response after this we can not write any res.write() html



On both requests and responses, Http headers are added to transport metadata from A to B.

The following article provides a great overview of available headers and their role: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers>

// her we create a local server

const http = require('http'); // get the url request

const fs = require('fs'); // fs enable us to work with the file system

const server = http.createServer((request,response) => { // crete server from teh url req. and its response

    const url = request.url;

    const method = request.method;

    if(url === '/') {

        response.write('<html>');

        response.write('<head><title> response in node </title></head>')

        response.write(`<body>

        <form action="/message" method="POST">

/\* the action in the form will move the page to that/message\*/

        <input type="text" name="message">

/\* name message will be shown on the network  menu doc \*/

        <button type="submit">Send</button>

        </form>

        </body>`)

        response.write('</html>')

        return response.end(); // we have to return the after response end so it will not go down further because we do not write after response end

    }

    if(url === '/message' && method === 'POST') { // it will run only the above condition run

        fs.writeFileSync('message.txt','Dummy'); // it will create the file with of .txt

        response.statusCode = 302; // it sends for redirection 302 will redirect

        response.setHeader('Location','/') // it will locate the heder back to (/) home page

        return response.end();

    }

    console.log(request); // it keeps on running in the event loop

    // process.exit() // it is used to quit the server

    response.setHeader('Content-Type','text/html');

    response.write('<html>');  // it will write the in the response in form of html

    response.write('<head><title> response in node </title></head>')

    response.write('<body> this line written by response.write inside the node</body>')

    response.write('</html>')

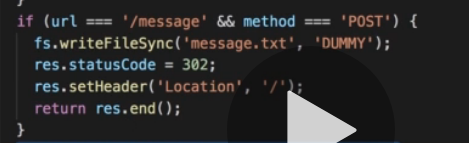
    response.end();

})

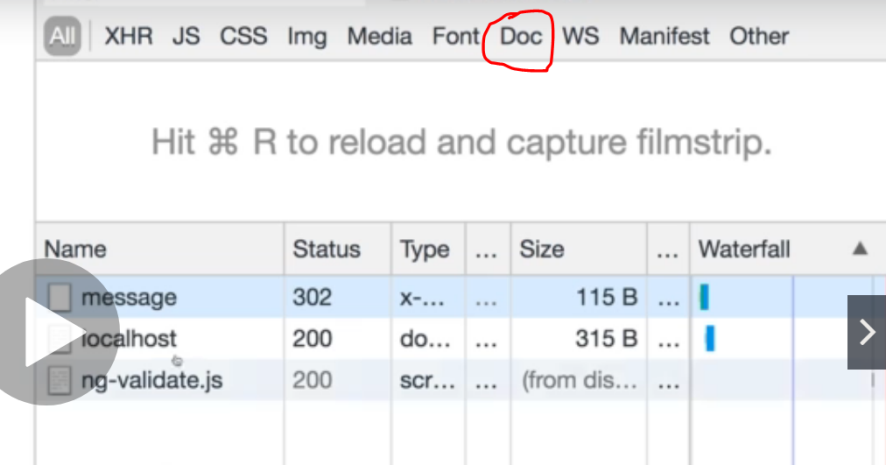
server.listen(3000); // it will listen the server continuously with the port of 3000

// and the host name as local host by default if we do not pass so

// so when we write localhost 3000 in our browser it will console in our node



The above code set the status as 302and move page to back /

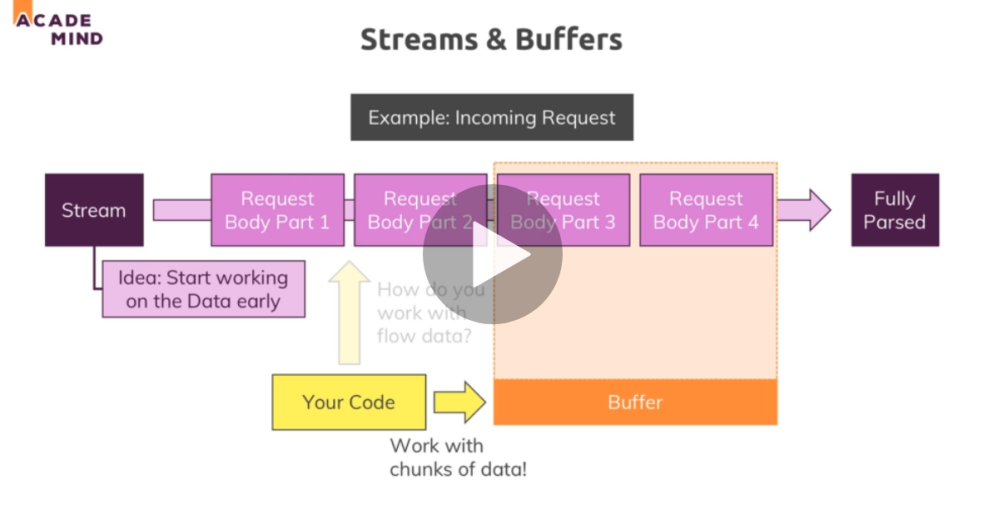


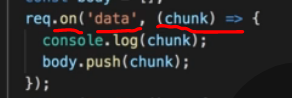
* Streams and buffer in node Js and the data

We are using streams and buffer so we can get the data that we stored in message above

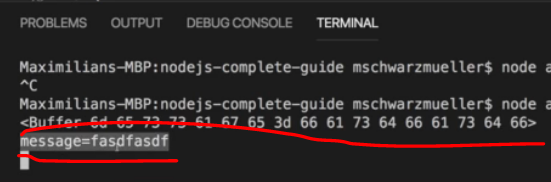
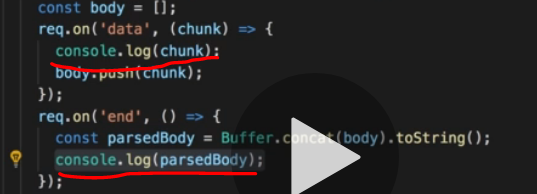
Streams are the objects which lead you to read data from a source or write data to a destination in continues manner. We can say stream as live stream or streaming the movie and not downloading it. Stream will provide us data continuesly while we watching it

Buffer is the temporary chunk of data that is being transfer from one place to another in small amount without waiting the whole data to load.



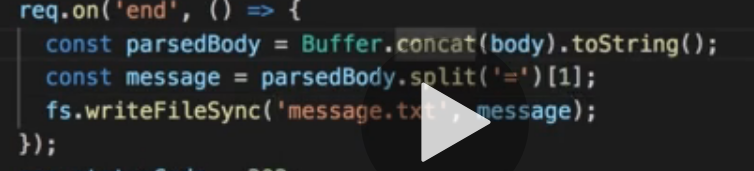
So to get the data have to listen the data first by event listener which is written as .a()  here the event is data and the we run the method on it when ever it pass the data and pushing in body array so we can use with buffer.

So we have to write another event that is (end) so when all the data passed at the end we can use the buffer as bus stop and convert it to string so that we can get the data as string



So the first red line is the console of first while (2 is the console of 2nd line)

The data is saved in key value pair which is message as key and = fasd----f as value

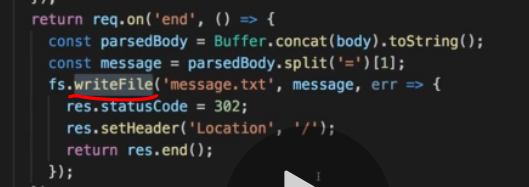


So above we storing our data at 1 index of split into message and writing a file containing message in our system.

* Event driven code and blocking and non blocking code

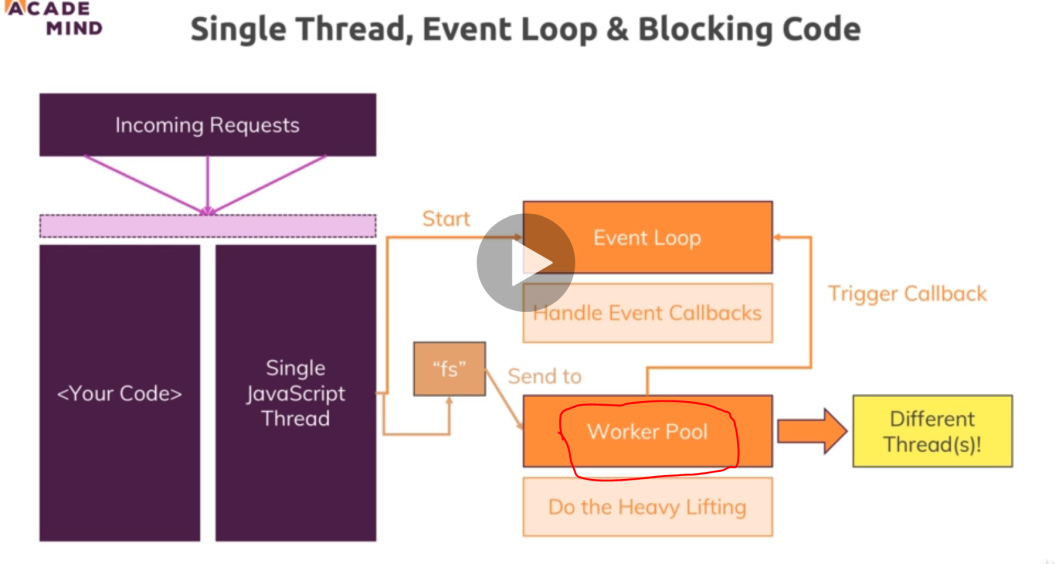
So event listener store that into registry because it is async in nature and run after all the code executes

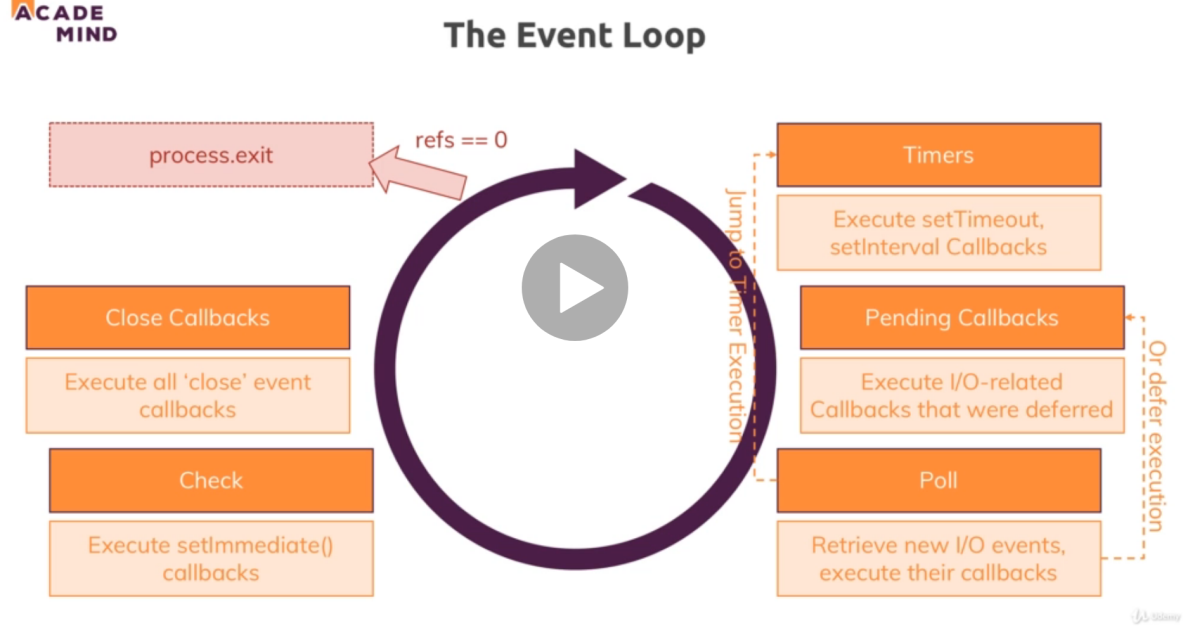
The differernce between .writeFileSync and .writeFile is that .writeFileSync is Syncronus and if it is large file to write then it will block the code below so it is good to use .writeFile because it is async and run after the code completion.



Node js just like java Script single threaded execute one line at a time and the event loop run the sync function first and then Async function after completing all line of code so the set time out will run from the call back queue when the call stack is empty

But in node all the methods and object made by (“fs”) will run inside the worker pool so that will do the heavy work of file



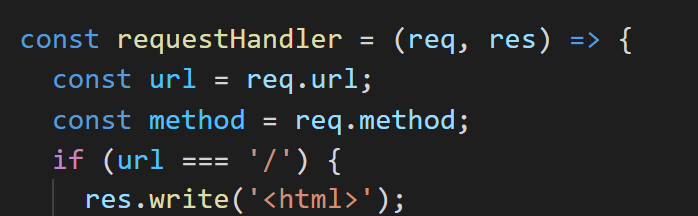


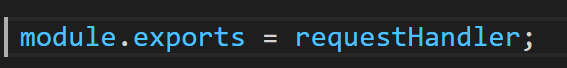
So event loop work on all this and run continues until (process exits)

Poll takes the i/o callbacks and send postponed callbacks to pending callbacks the timers will run simultaneously and jumping from poll to timers and timers to poll according to which done first

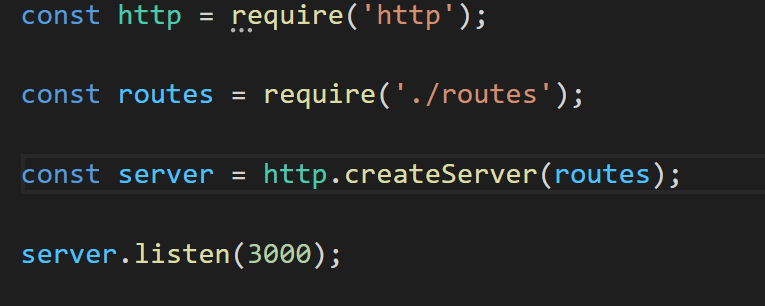
We can connect to Node js files with module.exports in

Second file

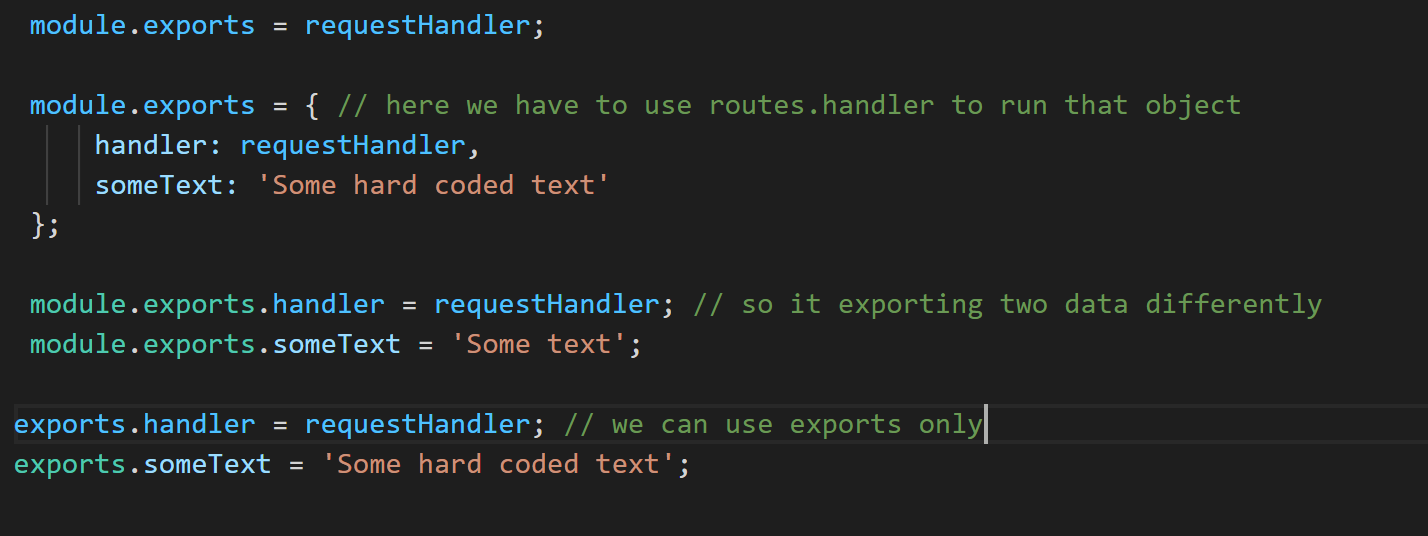




First file is

 so when we pass routes then it passes the request and response inside as module .export

We can also write module export as



Useful Resources & Links

Attached, you find the source code for this section.

Useful resources:

* Official Node.js Docs: <https://nodejs.org/en/docs/guides/>
* Full Node.js Reference (for all core modules): <https://nodejs.org/dist/latest/docs/api/>
* More about the Node.js Event Loop: <https://nodejs.org/en/docs/guides/event-loop-timers-and-nexttick/>
* Blocking and Non-Blocking Code: <https://nodejs.org/en/docs/guides/dont-block-the-event-loop/>

**Use of npm in Node JS**  
npm is used to install 3rd party packages in our project and those packages are available in npm repository

And we can install a package using **npm install** which will get install from npm repository and if we write –g it will installed globally

So when you write npm install it will install the node module folder