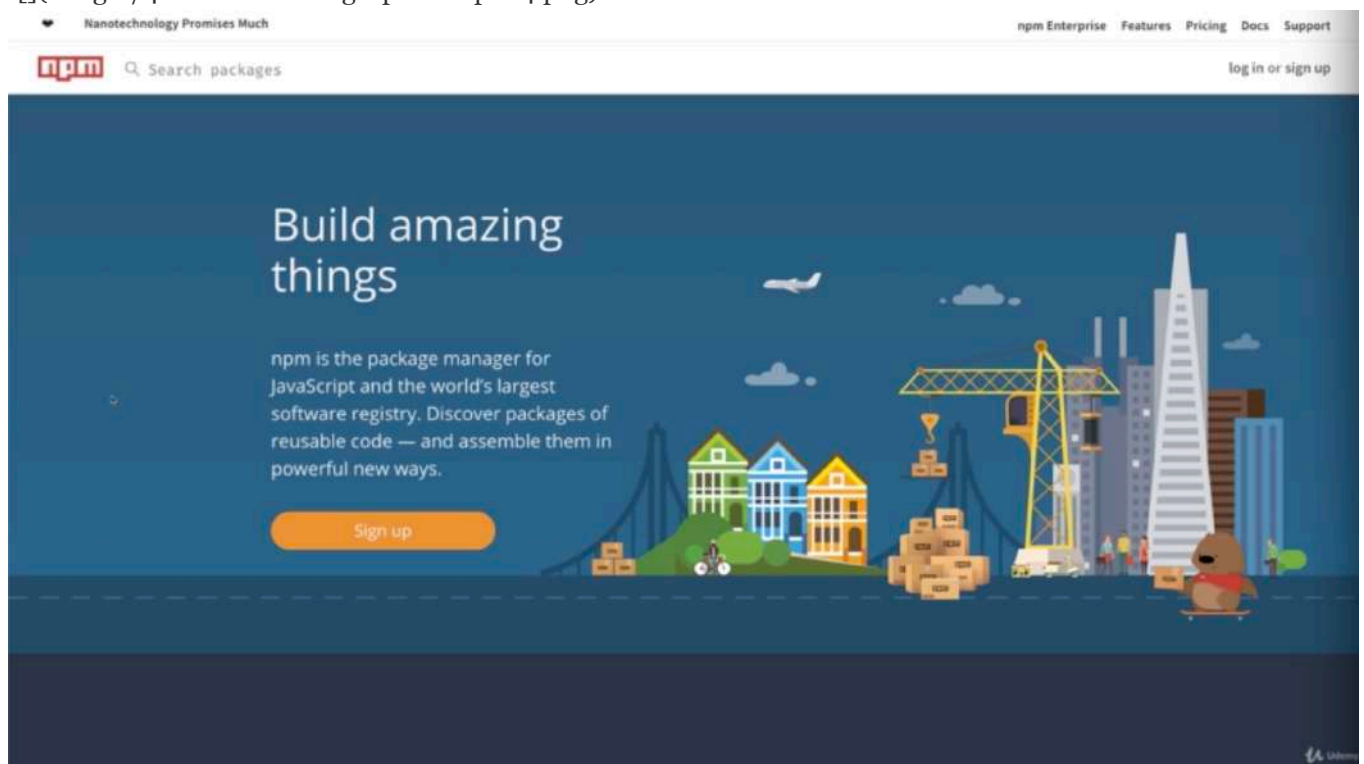
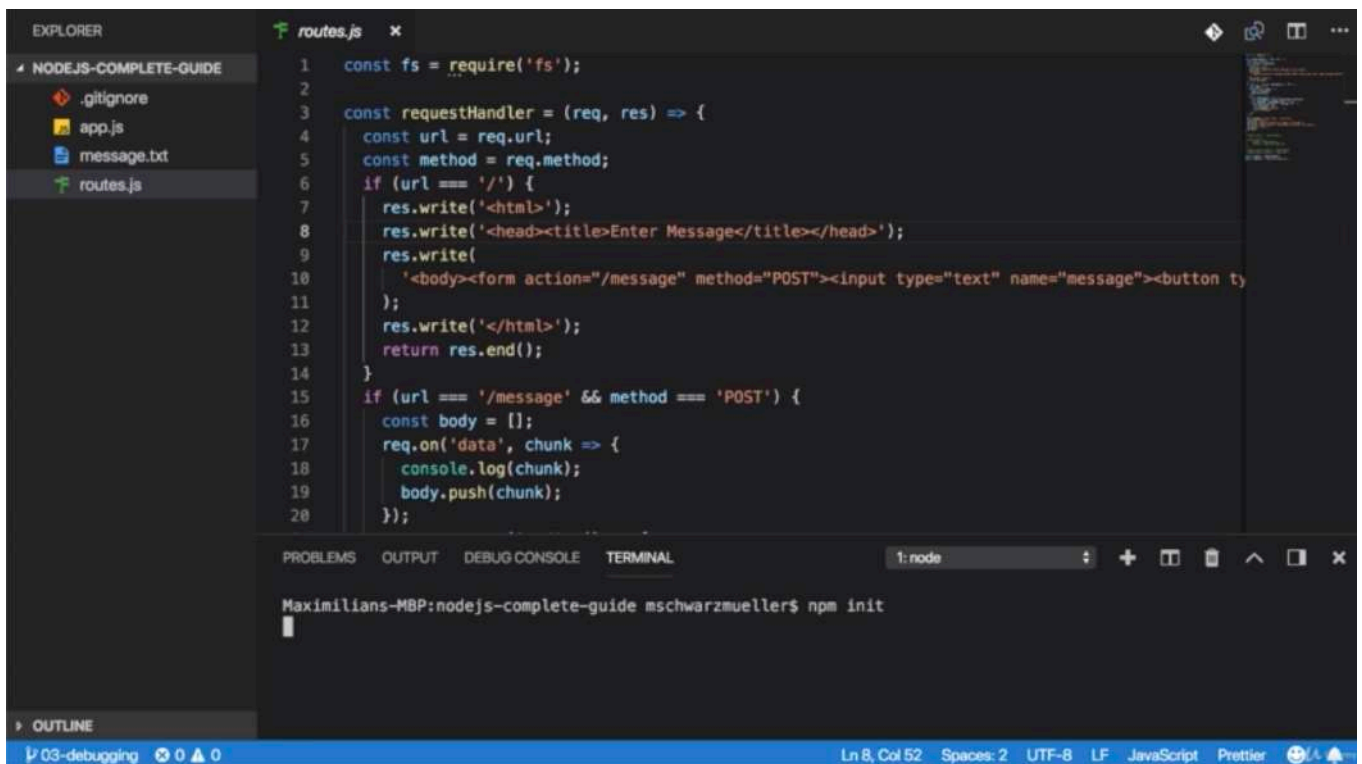


4. Improved Development Workflow And Debugging

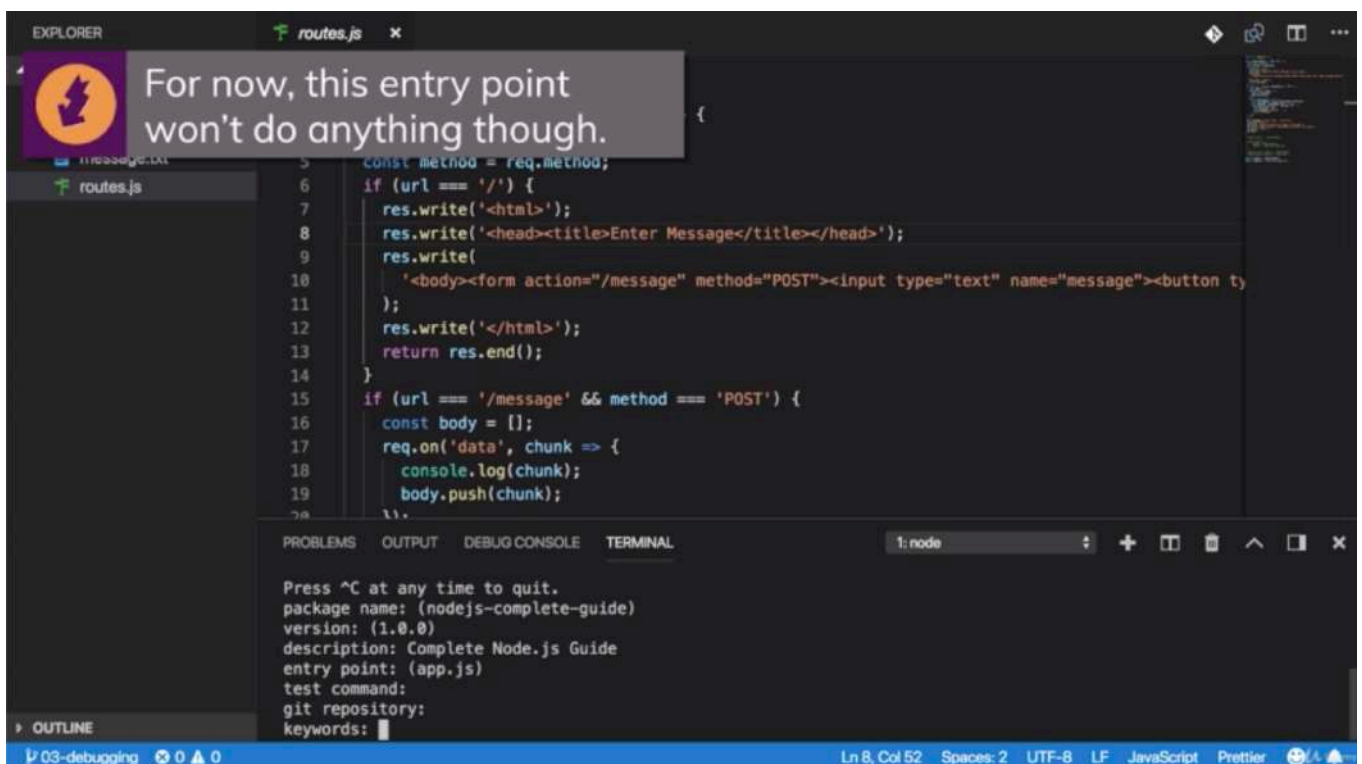
* Chapter 41: Understanding NPM Scripts





```
1 const fs = require('fs');
2
3 const requestHandler = (req, res) => {
4   const url = req.url;
5   const method = req.method;
6   if (url === '/') {
7     res.write('<html>');
8     res.write('<head><title>Enter Message</title></head>');
9     res.write(
10       '<body><form action="/message" method="POST"><input type="text" name="message"><button ty
11     );
12     res.write('</html>');
13     return res.end();
14   }
15   if (url === '/message' && method === 'POST') {
16     const body = [];
17     req.on('data', chunk => {
18       console.log(chunk);
19       body.push(chunk);
20     });
```

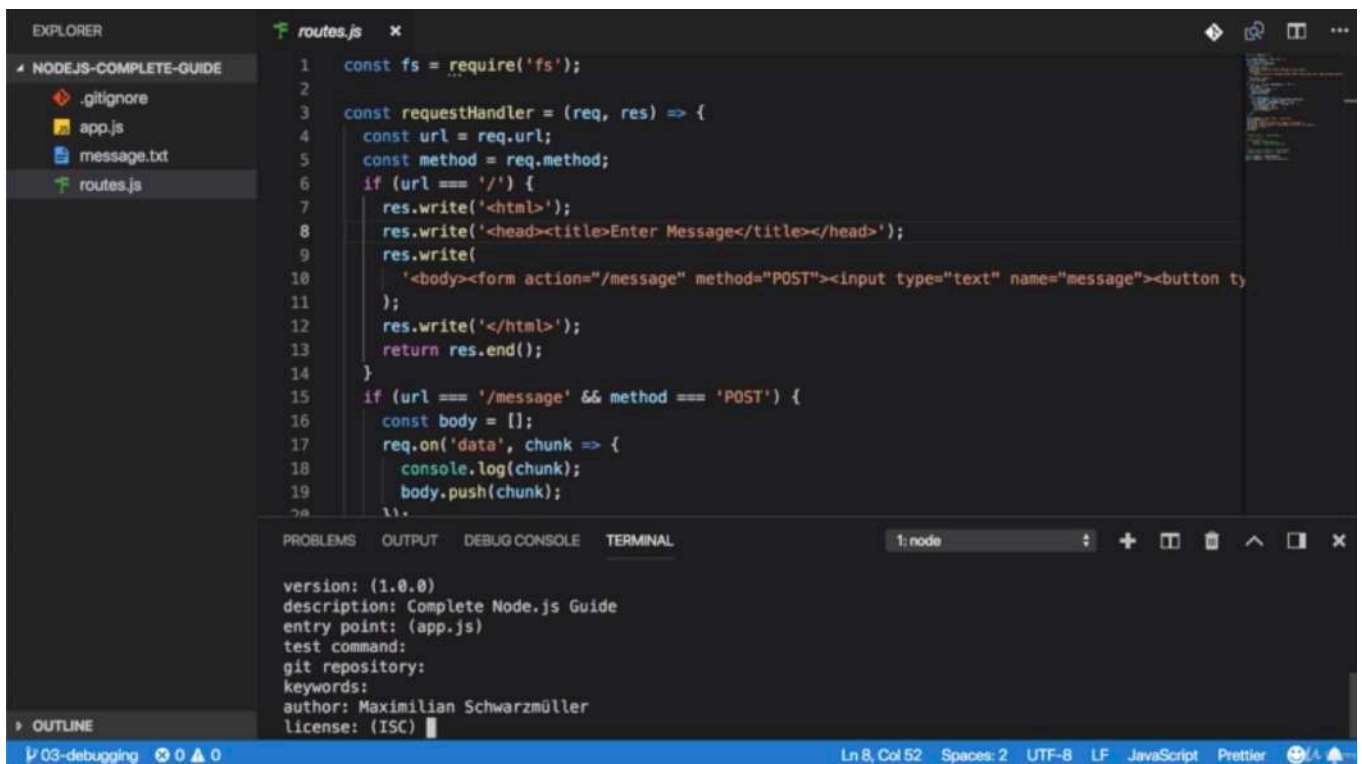
Maximilians-MBP:nodejs-complete-guide mschwarzmueller\$ npm init



For now, this entry point won't do anything though.

```
5   const method = req.method;
6   if (url === '/') {
7     res.write('<html>');
8     res.write('<head><title>Enter Message</title></head>');
9     res.write(
10       '<body><form action="/message" method="POST"><input type="text" name="message"><button ty
11     );
12     res.write('</html>');
13     return res.end();
14   }
15   if (url === '/message' && method === 'POST') {
16     const body = [];
17     req.on('data', chunk => {
18       console.log(chunk);
19       body.push(chunk);
20     });
```

Press ^C at any time to quit.
package name: (nodejs-complete-guide)
version: (1.0.0)
description: Complete Node.js Guide
entry point: (app.js)
test command:
git repository:
keywords:



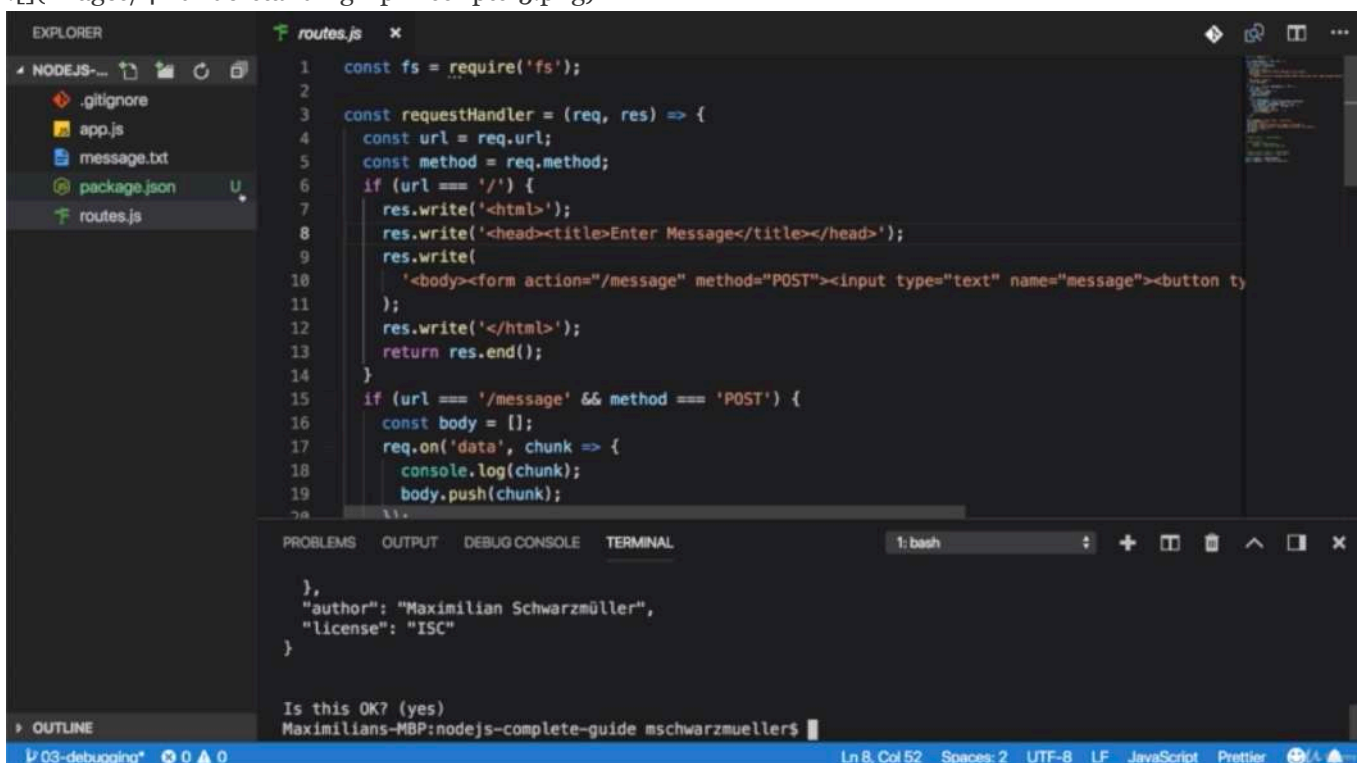
The screenshot shows a VS Code editor with a file explorer on the left containing .gitignore, app.js, message.txt, and routes.js. The main editor displays routes.js with the following code:

```
1 const fs = require('fs');
2
3 const requestHandler = (req, res) => {
4   const url = req.url;
5   const method = req.method;
6   if (url === '/') {
7     res.write('<html>');
8     res.write('<head><title>Enter Message</title></head>');
9     res.write(
10       '<body><form action="/message" method="POST"><input type="text" name="message"><button ty
11     );
12     res.write('</html>');
13     return res.end();
14   }
15   if (url === '/message' && method === 'POST') {
16     const body = [];
17     req.on('data', chunk => {
18       console.log(chunk);
19       body.push(chunk);
20     });
21   }
22 }
```

The terminal at the bottom shows the output of a command:

```
version: (1.0.0)
description: Complete Node.js Guide
entry point: (app.js)
test command:
git repository:
keywords:
author: Maximilian Schwarzmüller
license: (ISC)
```

- you will be prompted with a couple of questions. it will ask you first of all for a name of your package and now you can simply translate this with your project name. now you can pick any name you want, the part in the parenthese here always is the suggetstion, the default it will pick if you don't choose your own name.
 - and you can choose license but if you don't plan on sharing this project publicly, this doesn'y matter.
-

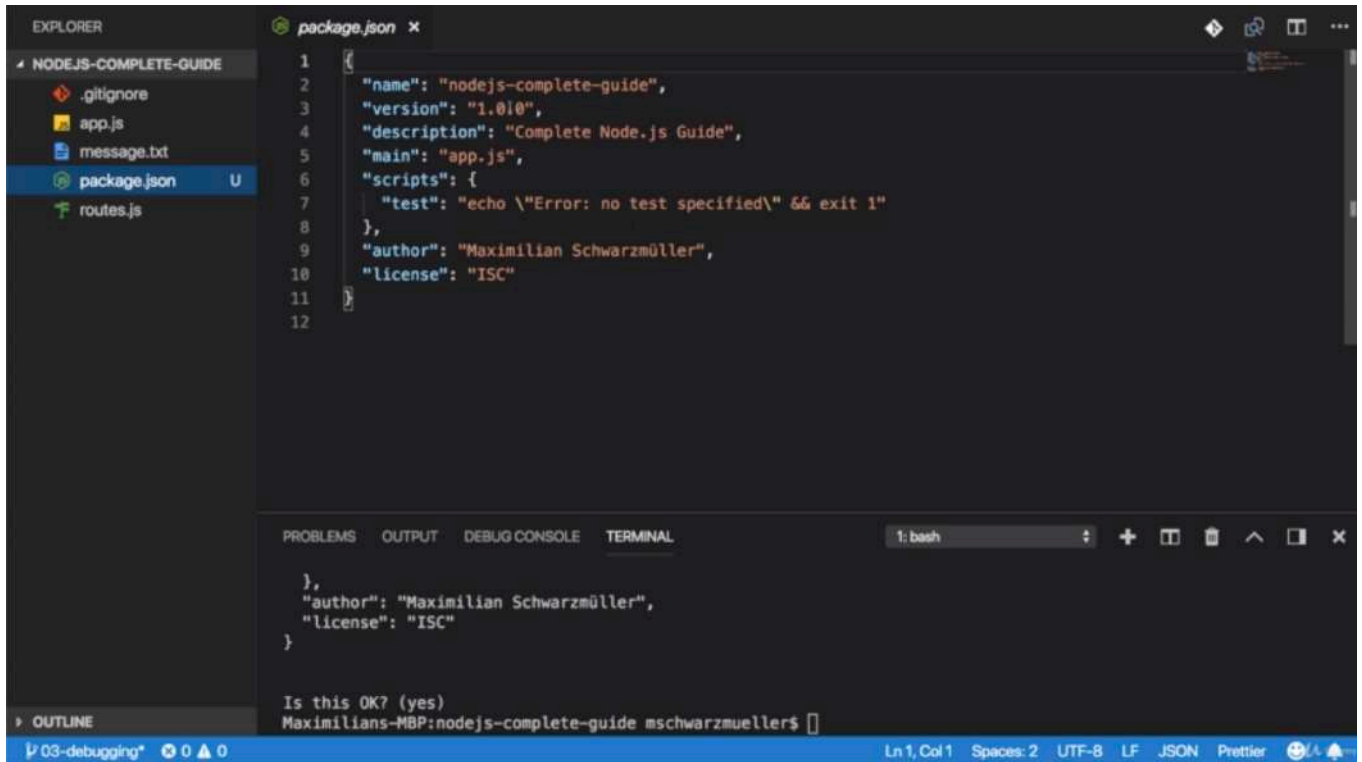


The screenshot shows a VS Code editor with a file explorer on the left containing .gitignore, app.js, message.txt, package.json, and routes.js. The main editor displays routes.js with the same code as the previous screenshot. The terminal at the bottom shows the output of a command:

```
},
"author": "Maximilian Schwarzmüller",
"license": "ISC"
}

Is this OK? (yes)
Maximilians-MBP:nodejs-complete-guide mschwarzmueller$
```

- and you get package.json and there you also see all these settings or configurations you just set up and you can edit them there too.
 - this is using the JSON format which is a special kind of data format which basically looks a lot like javascript objects and it pretty much is based on that
 - the key are always put between double quotation marks and so are the values, except for numbers or arrays or true or false which are not put between these.
-

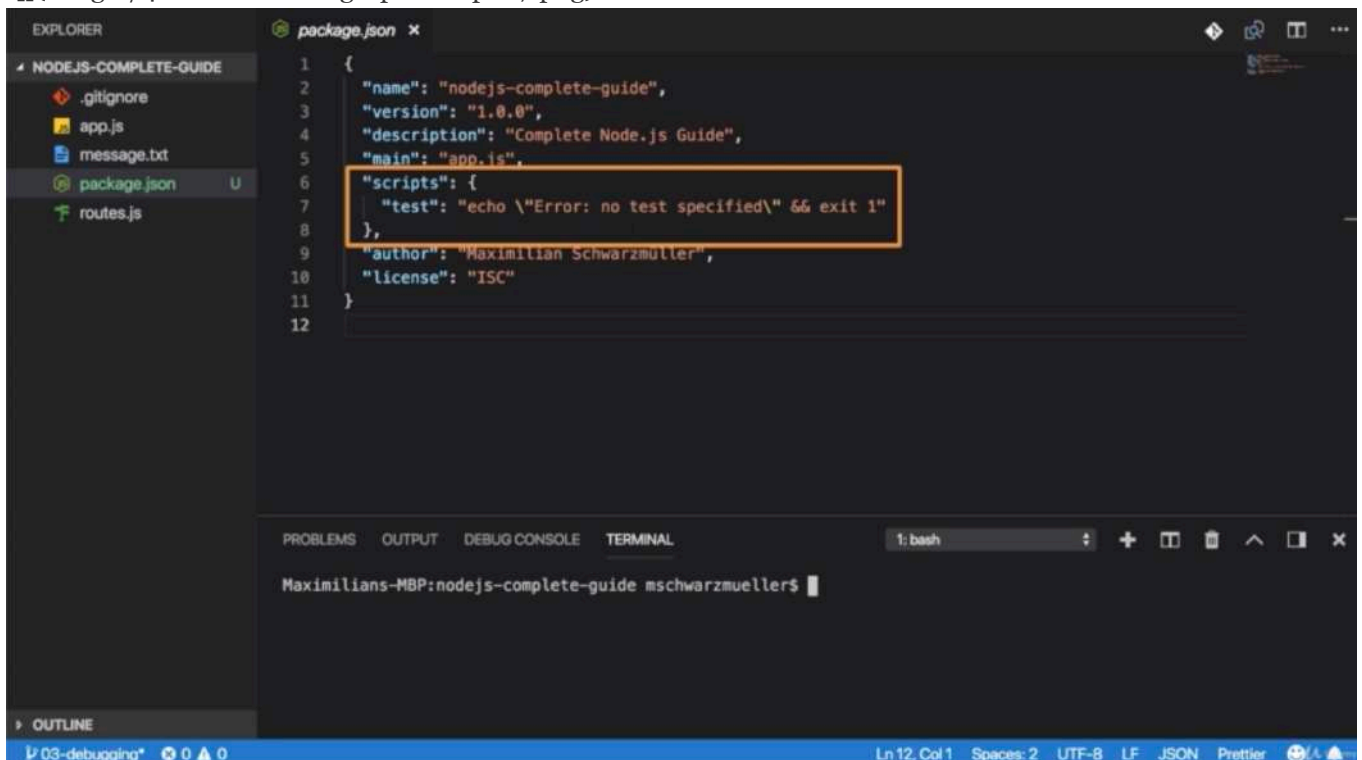


The screenshot shows the VS Code editor with the `package.json` file open. The file contains the following JSON:

```
{
  "name": "nodejs-complete-guide",
  "version": "1.0.0",
  "description": "Complete Node.js Guide",
  "main": "app.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "author": "Maximilian Schwarzmüller",
  "license": "ISC"
}
```

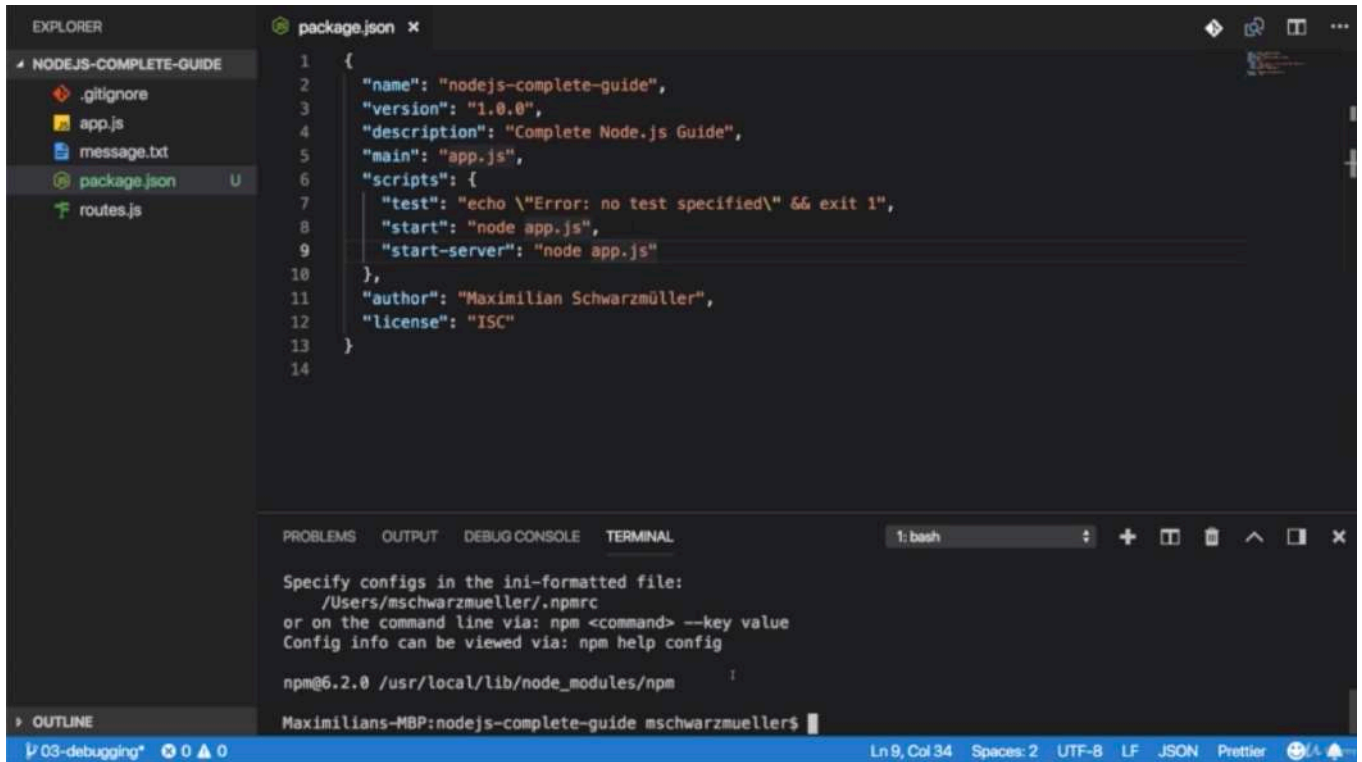
The Explorer sidebar on the left shows the project structure for `NODEJS-COMPLETE-GUIDE`, including `.gitignore`, `app.js`, `message.txt`, `package.json` (selected), and `routes.js`. The Terminal at the bottom shows a `bash` prompt with the command `Maximilians-MBP:nodejs-complete-guide mschwarzmuellers$`.

- what does this configuration file give you? you will see that we got a scripts section there which has one default script that won't do anything for you now.
- and you can add your own scripts here and i will tell you how to execute them too.



This screenshot is similar to the previous one, but the `"scripts": {` section of the `package.json` file is highlighted with an orange box. The JSON content is identical to the previous image.

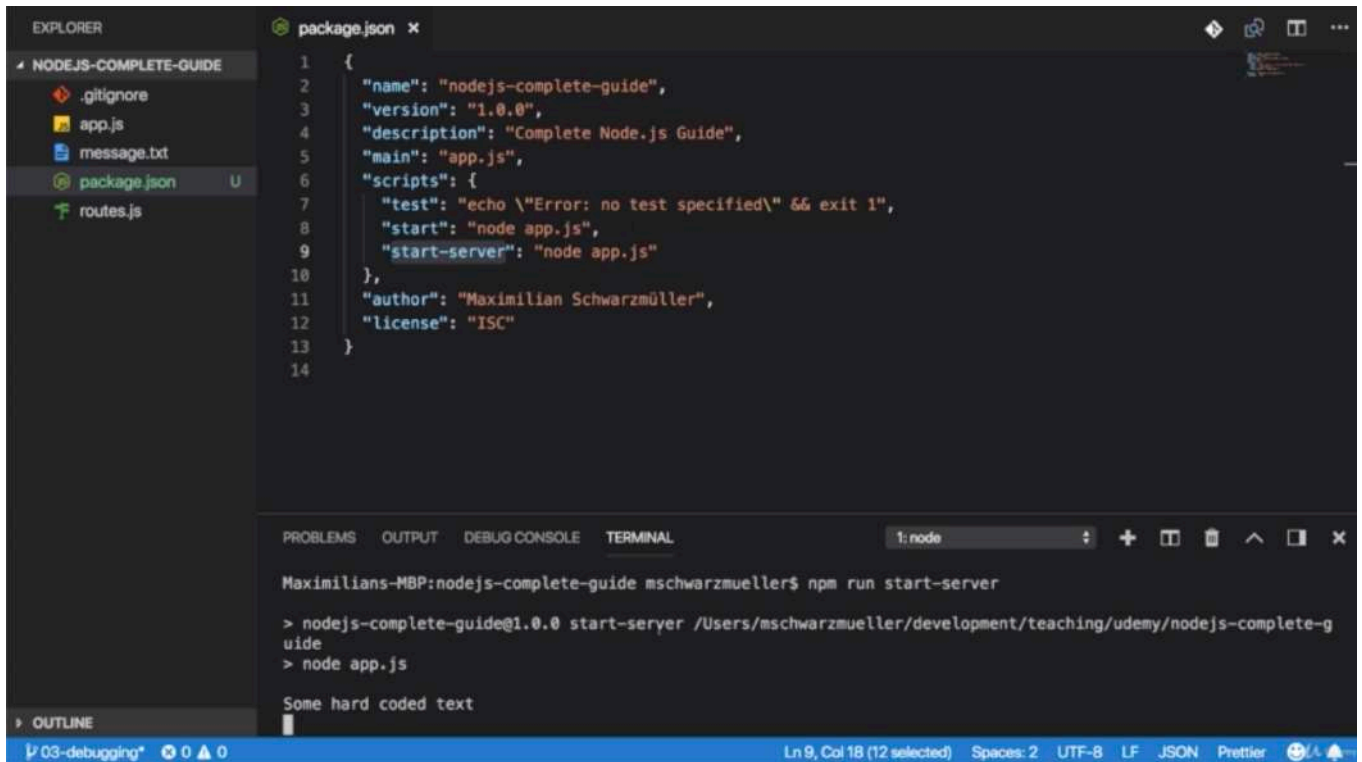
- 'start' is actually a special script name as you will see in a second. then you type a command that should be executed.
- 'start' is reserved name and this will always look for such a start script.
- you can always just run this command instead of running `node app.js`



The screenshot shows a VS Code editor with a file explorer on the left containing files like .gitignore, app.js, message.txt, package.json, and routes.js. The main editor displays the package.json file with the following content:

```
1 {
2   "name": "nodejs-complete-guide",
3   "version": "1.0.0",
4   "description": "Complete Node.js Guide",
5   "main": "app.js",
6   "scripts": {
7     "test": "echo \"Error: no test specified\" && exit 1",
8     "start": "node app.js",
9     "start-server": "node app.js"
10  },
11  "author": "Maximilian Schwarzmüller",
12  "license": "ISC"
13 }
14
```

The terminal window at the bottom shows the output of running 'npm help config', displaying instructions on how to specify configurations in an ini-formatted file or via command line, and the current npm version and path.

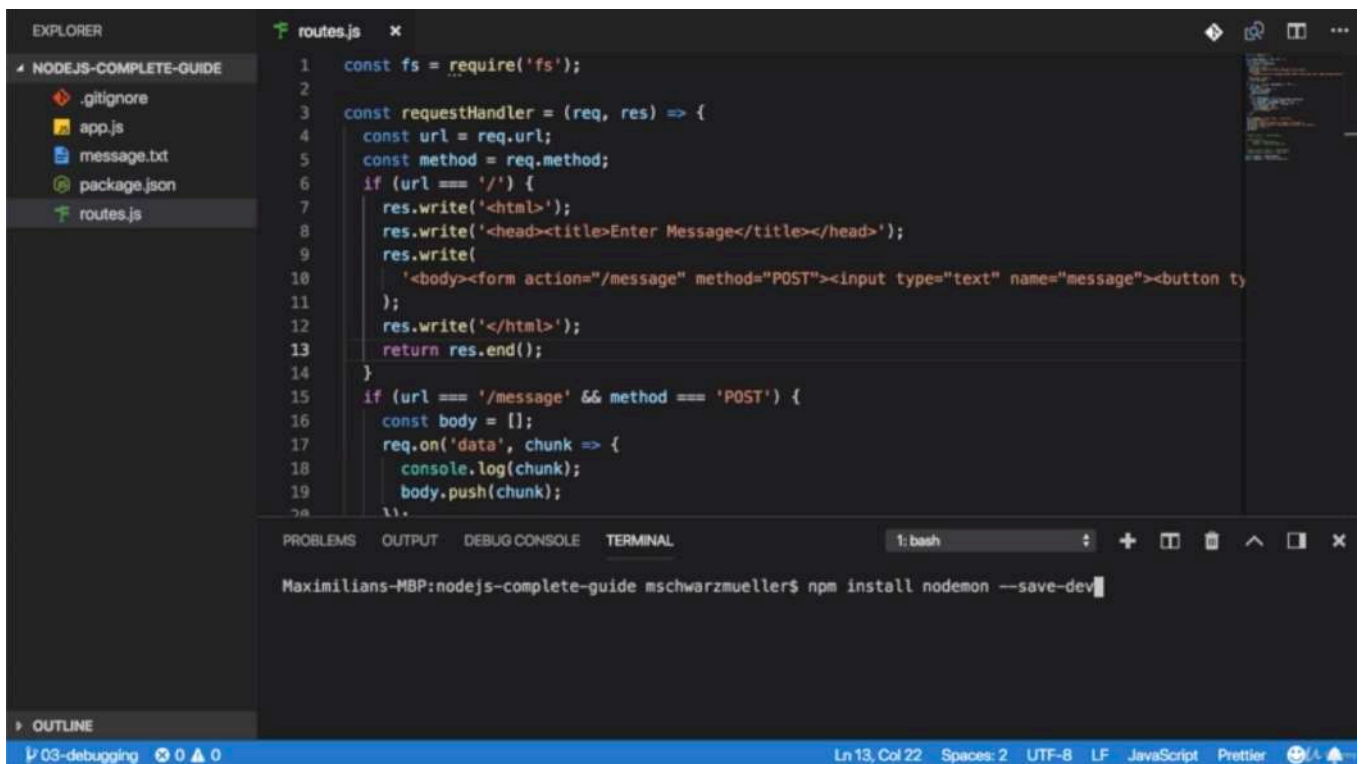


This screenshot shows the same VS Code editor setup, but the terminal window now shows the output of running 'npm run start-server'. The output indicates that the command was not found, which is expected because 'start-server' is not a standard npm script name.

```
Maximilians-MBP:nodejs-complete-guide mschwarzmueller$ npm run start-server
> nodejs-complete-guide@1.0.0 start-server /Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide
> node app.js
Some hard coded text
```

- you get a error that is not a known command and indeed it isn't because just typing the script name will not work. 'start' is just a special case.
- for normal script with their own custom names, you have to run npm run + script name

* Chapter 42: Installing 3rd Party Packages

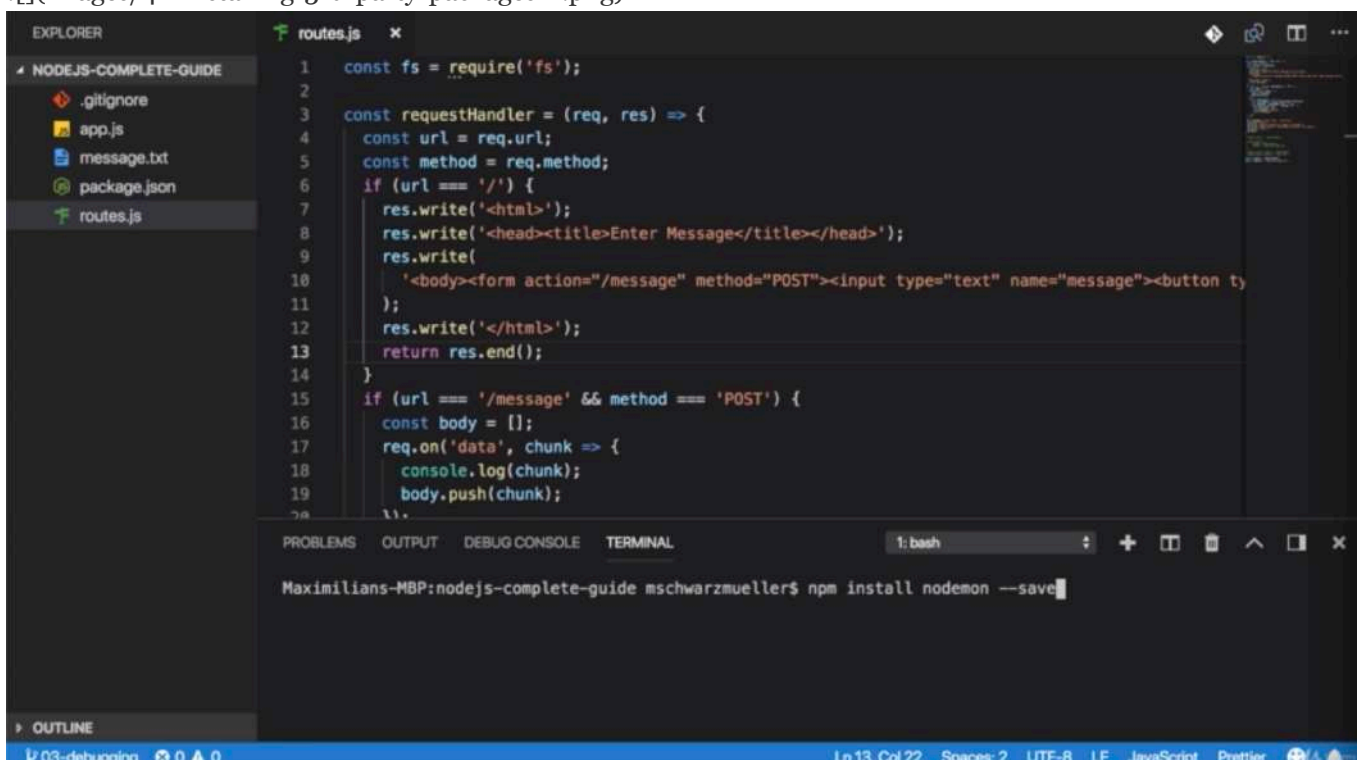


The screenshot shows a VS Code editor with a file explorer on the left containing files like .gitignore, app.js, message.txt, package.json, and routes.js. The main editor displays the content of routes.js, which is a Node.js Express route handler. The terminal at the bottom shows the command `npm install nodemon --save-dev` being executed in a bash shell.

```
1  const fs = require('fs');
2
3  const requestHandler = (req, res) => {
4    const url = req.url;
5    const method = req.method;
6    if (url === '/') {
7      res.write('<html>');
8      res.write('<head><title>Enter Message</title></head>');
9      res.write(
10        '<body><form action="/message" method="POST"><input type="text" name="message"><button ty
11      );
12      res.write('</html>');
13      return res.end();
14    }
15    if (url === '/message' && method === 'POST') {
16      const body = [];
17      req.on('data', chunk => {
18        console.log(chunk);
19        body.push(chunk);
20      });
21    }
22  }
```

Maximilians-MBP:nodejs-complete-guide mschwarzmueller\$ npm install nodemon --save-dev

- we will install one big 3rd package because you typically have such dependencies 3rd party packages.
 - ‘nodemon’ which is auto-restart mechanism
 - you can define how this should be installed because packages which you nstall can be divided into development packages, so packages which mostly help you during development and production dependencies. so packages that helps you for the app as it’s running on a server, for example nodemon would be a development dependency because we only use it during the development process. once we install our app on a real server, we don’t need it there. the real server which is running somewhere in the internet shouldn’t restart.
 - and you can tell npm which kind of dependency this is, this doesn’t make a huge difference and you can omit the setting but it helps you understand which package is used for what.
 - we are indicating that this only adds something we used during development
-



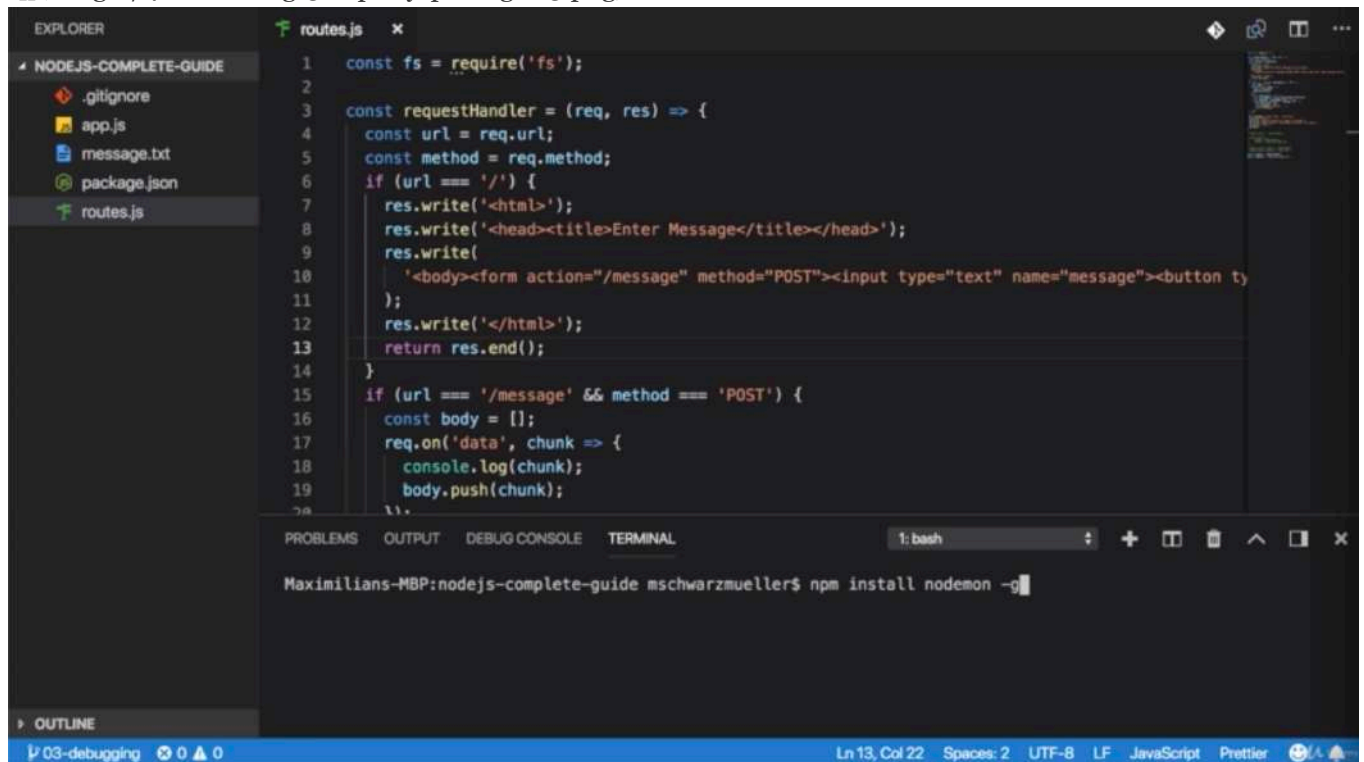
This screenshot is similar to the one above, but the terminal shows the command `npm install nodemon --save` instead of `--save-dev`. The code in routes.js is identical.

```
1  const fs = require('fs');
2
3  const requestHandler = (req, res) => {
4    const url = req.url;
5    const method = req.method;
6    if (url === '/') {
7      res.write('<html>');
8      res.write('<head><title>Enter Message</title></head>');
9      res.write(
10        '<body><form action="/message" method="POST"><input type="text" name="message"><button ty
11      );
12      res.write('</html>');
13      return res.end();
14    }
15    if (url === '/message' && method === 'POST') {
16      const body = [];
17      req.on('data', chunk => {
18        console.log(chunk);
19        body.push(chunk);
20      });
21    }
22  }
```

Maximilians-MBP:nodejs-complete-guide mschwarzmueller\$ npm install nodemon --save

- if you had just save like this, this would install it as a production dependency. so a package which we really use

and use in our code and work with and with this



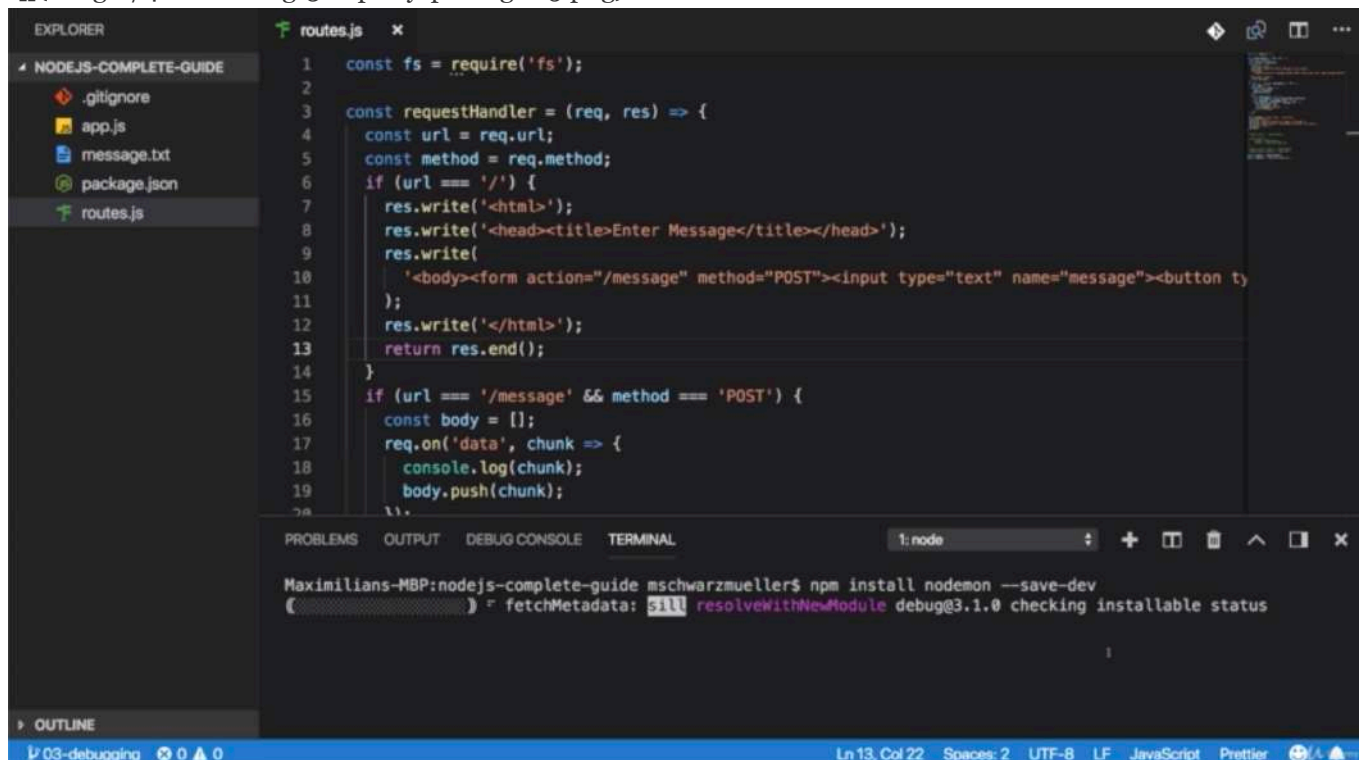
The screenshot shows a VS Code editor with a file explorer on the left containing files like .gitignore, app.js, message.txt, package.json, and routes.js. The main editor displays the content of routes.js, which is a Node.js Express application. The code defines a request handler for the root path that serves an HTML form, and another handler for the /message path that logs incoming POST data. The terminal at the bottom shows the command 'npm install nodemon -g' being executed, with the output 'Maximilians-MBP:nodejs-complete-guide mschwarzmueller\$ npm install nodemon -g'.

```
1 const fs = require('fs');
2
3 const requestHandler = (req, res) => {
4   const url = req.url;
5   const method = req.method;
6   if (url === '/') {
7     res.write('<html>');
8     res.write('<head><title>Enter Message</title></head>');
9     res.write(
10       '<body><form action="/message" method="POST"><input type="text" name="message"><button ty
11     );
12     res.write('</html>');
13     return res.end();
14   }
15   if (url === '/message' && method === 'POST') {
16     const body = [];
17     req.on('data', chunk => {
18       console.log(chunk);
19       body.push(chunk);
20     });
21   }
22 }
```

1: bash

Maximilians-MBP:nodejs-complete-guide mschwarzmueller\$ npm install nodemon -g

- 'g' is that we will not install it in this project but globally on your machine so that you can use it anywhere.



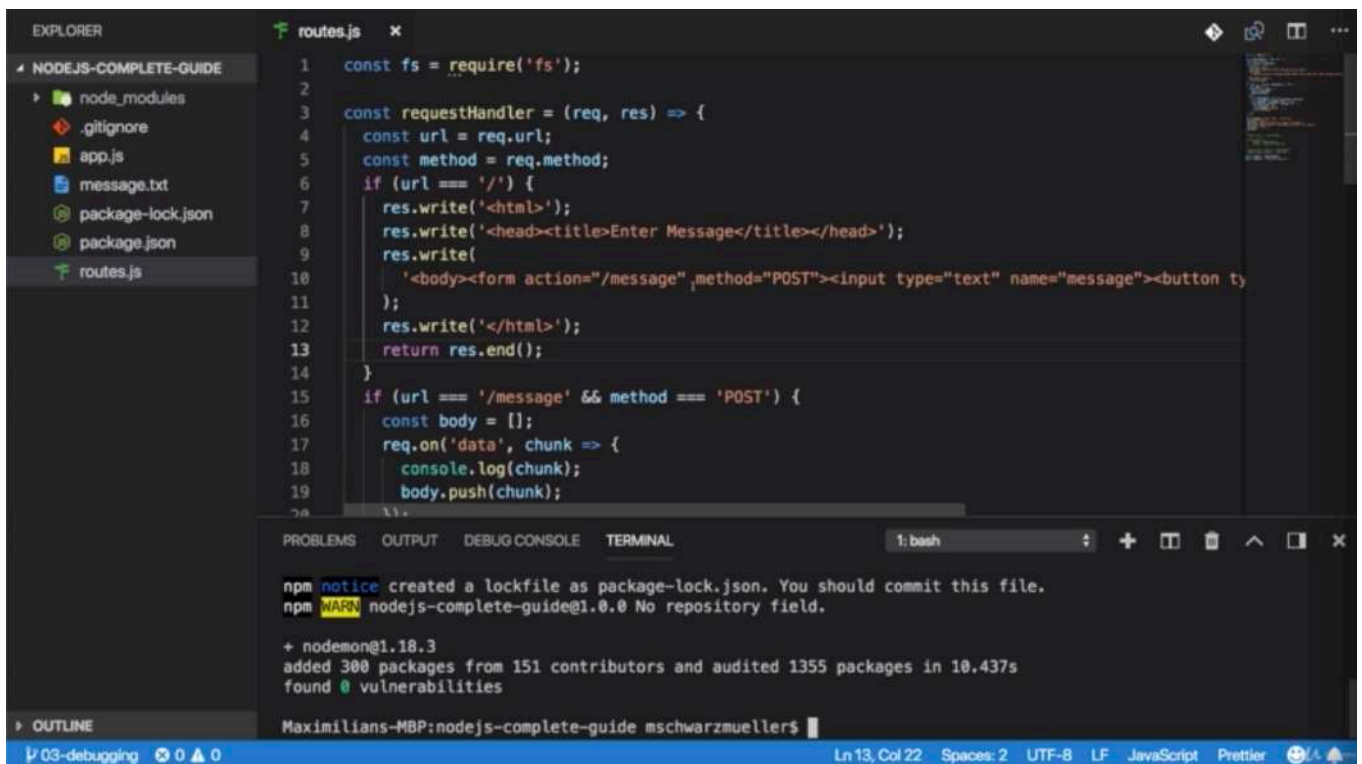
The screenshot shows the same VS Code editor setup as the previous one, but the terminal now shows the command 'npm install nodemon --save-dev'. The output shows the command being executed and the package being installed as a development dependency. The status bar at the bottom indicates 'Ln 13, Col 22'.

```
1 const fs = require('fs');
2
3 const requestHandler = (req, res) => {
4   const url = req.url;
5   const method = req.method;
6   if (url === '/') {
7     res.write('<html>');
8     res.write('<head><title>Enter Message</title></head>');
9     res.write(
10       '<body><form action="/message" method="POST"><input type="text" name="message"><button ty
11     );
12     res.write('</html>');
13     return res.end();
14   }
15   if (url === '/message' && method === 'POST') {
16     const body = [];
17     req.on('data', chunk => {
18       console.log(chunk);
19       body.push(chunk);
20     });
21   }
22 }
```

1: node

Maximilians-MBP:nodejs-complete-guide mschwarzmueller\$ npm install nodemon --save-dev

() = fetchMetadata: sill resolveWithNewModule debug@3.1.0 checking installable status



The screenshot shows the VS Code editor with the Explorer sidebar on the left displaying the file structure of 'NODEJS-COMPLETE-GUIDE'. The main editor area shows the 'routes.js' file with the following code:

```
1 const fs = require('fs');
2
3 const requestHandler = (req, res) => {
4   const url = req.url;
5   const method = req.method;
6   if (url === '/') {
7     res.write('<html>');
8     res.write('<head><title>Enter Message</title></head>');
9     res.write(
10       '<body><form action="/message" method="POST"><input type="text" name="message"><button ty
11     );
12     res.write('</html>');
13     return res.end();
14   }
15   if (url === '/message' && method === 'POST') {
16     const body = [];
17     req.on('data', chunk => {
18       console.log(chunk);
19       body.push(chunk);
20     });
21   }
22 }
```

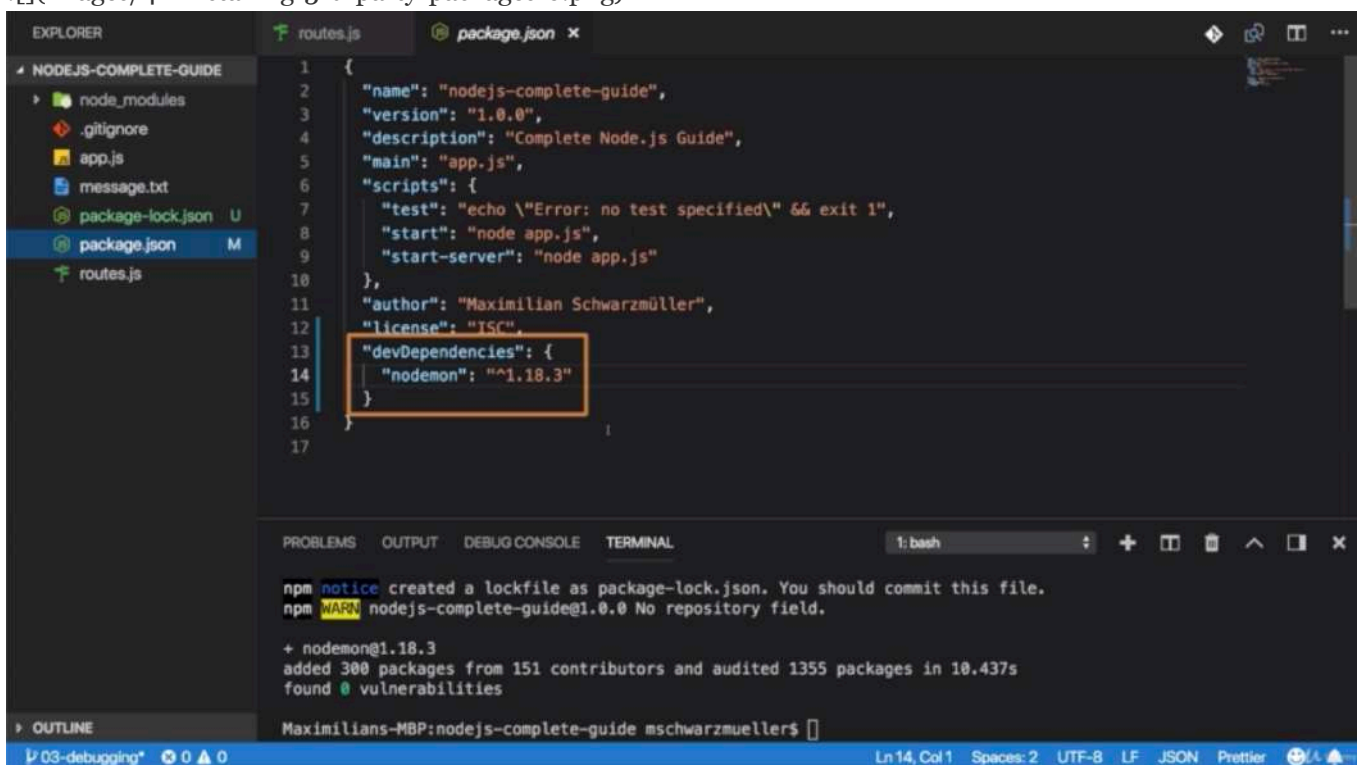
The bottom panel shows the 'TERMINAL' tab with the following output:

```
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN nodejs-complete-guide@1.0.0 No repository field.

+ nodemon@1.18.3
added 300 packages from 151 contributors and audited 1355 packages in 10.437s
found 0 vulnerabilities

Maximilians-MBP:nodejs-complete-guide mschwarzmueller$
```

- after installing, it gives you that node_modules folder, the package-lock.json file and it updated the package.json file.



The screenshot shows the VS Code editor with the Explorer sidebar on the left. The main editor area shows the 'package.json' file with the following code:

```
1 {
2   "name": "nodejs-complete-guide",
3   "version": "1.0.0",
4   "description": "Complete Node.js Guide",
5   "main": "app.js",
6   "scripts": {
7     "test": "echo \"Error: no test specified\" && exit 1",
8     "start": "node app.js",
9     "start-server": "node app.js"
10  },
11   "author": "Maximilian Schwarzmüller",
12   "license": "ISC",
13   "devDependencies": {
14     "nodemon": "^1.18.3"
15  }
16 }
```

The bottom panel shows the 'TERMINAL' tab with the same output as the previous screenshot.

- we see that the new devDependencies section was added and that stands for development dependencies

The screenshot shows a VS Code editor with the Explorer sidebar on the left displaying the file structure of a project named 'NODEJS-COMPLETE-GUIDE'. The files listed are .gitignore, app.js, message.txt, package-lock.json, package.json, and routes.js. The package.json file is open in the editor, showing its contents. The terminal at the bottom shows the output of an npm install command, including a notice about creating a lockfile and a warning about the repository field.

```
1 {
2   "name": "nodejs-complete-guide",
3   "version": "1.0.0",
4   "description": "Complete Node.js Guide",
5   "main": "app.js",
6   "scripts": {
7     "test": "echo \"Error: no test specified\" && exit 1",
8     "start": "node app.js",
9     "start-server": "node app.js"
10  },
11  "author": "Maximilian Schwarzmüller",
12  "license": "ISC",
13  "devDependencies": {
14    "nodemon": "^1.18.3"
15  }
16 }
```

npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN nodejs-complete-guide@1.0.0 No repository field.

+ nodemon@1.18.3
added 300 packages from 151 contributors and audited 1355 packages in 10.437s
found 0 vulnerabilities

- this defines how this package will be updated. if you rerun just npm install, without defining an extra package name because this command standalone will simply go through all your packages mentioned in package.json and install them and it would automatically pick a later version if available.

- where is it installed? that is the node_modules folder.

- this is basically the source code of the package or the build version of the package we installed.

- this package simply happens to have a couple of peer dependencies. so we got a bunch of dependencies in there and their dependencies are also installed, that is why you could end up with quite a big node_modules folder

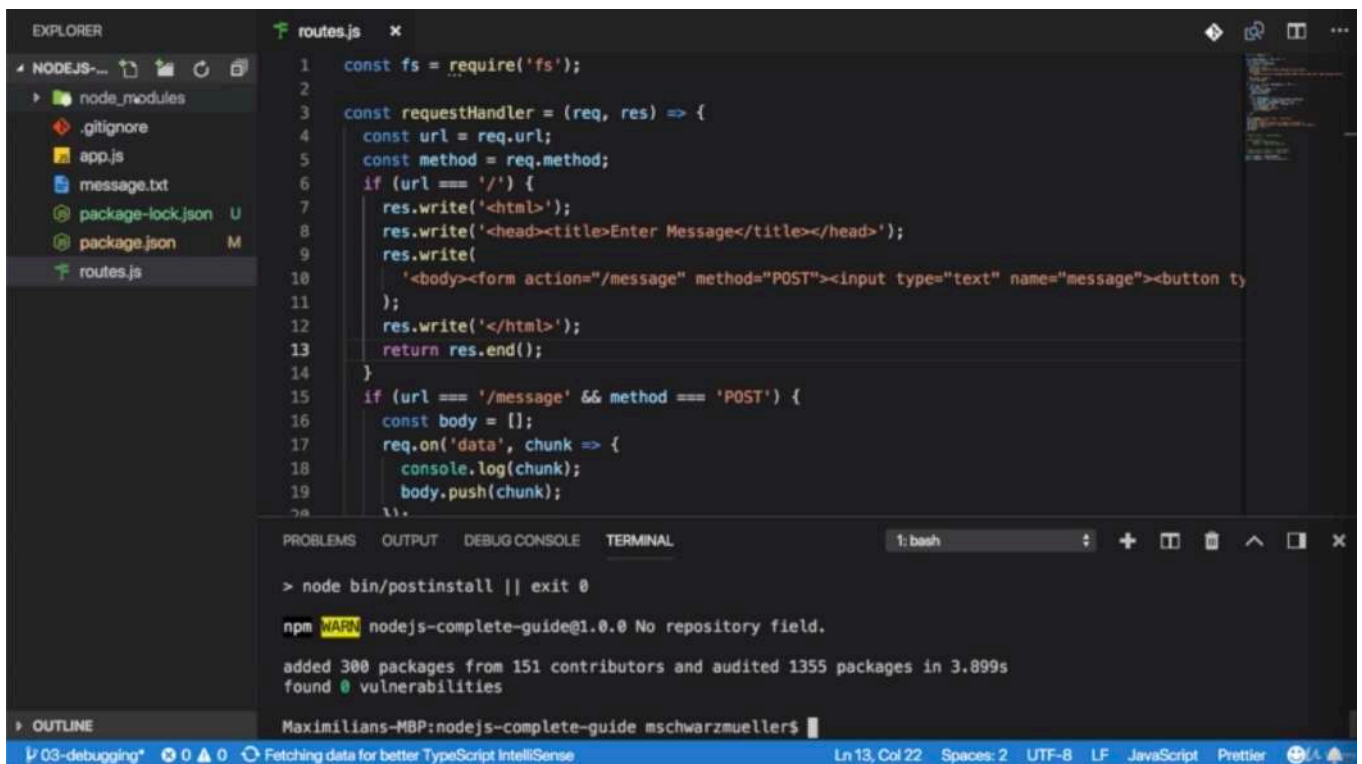
The screenshot shows a VS Code editor with the Explorer sidebar on the left displaying the file structure of a project named 'NODEJS-COMPLETE-GUIDE'. The files listed are .gitignore, app.js, message.txt, package-lock.json, package.json, and routes.js. The routes.js file is open in the editor, showing its contents. The terminal at the bottom shows the output of an npm install command, including a notice about creating a lockfile and a warning about the repository field.

```
1 const fs = require('fs');
2
3 const requestHandler = (req, res) => {
4   const url = req.url;
5   const method = req.method;
6   if (url === '/') {
7     res.write('<html>');
8     res.write('<head><title>Enter Message</title></head>');
9     res.write(
10       '<body><form action="/message" method="POST"><input type="text" name="message"><button ty
11     );
12     res.write('</html>');
13     return res.end();
14   }
15   if (url === '/message' && method === 'POST') {
16     const body = [];
17     req.on('data', chunk => {
18       console.log(chunk);
19       body.push(chunk);
20     });
21   }
22 }
```

Maximilians-MBP:nodejs-complete-guide mschwarzmueller\$ npm install
npm WARN nodejs-complete-guide@1.0.0 No repository field.

audited 1355 packages in 2.266s
found 0 vulnerabilities

Maximilians-MBP:nodejs-complete-guide mschwarzmueller\$ npm install



- but you can always delete that node_modules folder. you can't use that package but you can then rerun 'npm install' if you start working on that project again.

* Chapter 43: Global Features Vs Core Modules Vs 3rd Party Modules

- You can basically differentiate between:
 - Global Features: Keywords like 'const' or 'function' but also some global objects like process
 - Core Node.js Modules: Examples would be the file-system modules('fs'), the path module("path") or the Http module('http')
 - 3rd-party Modules: installed via 'npm install' - you can add any kind of feature to your app via this way.
- Global features are always available, you don't need to import them into the file where you wanna use them
- Core Node.js Modules don't need to be installed (NO 'npm install' is required) but you need to import them when you wanna use features exposed by them.
 - example
 - const fs = require('fs')
 - you can now use the 'fs' object exported by the 'fs' modules
- 3rd-party Modules need to be installed (via 'npm install' in the project folder) and imported
 - example(which you don't need to understand yet - we will cover this later in the course)

```

1 //In terminal / command prompt
2 npm install --save express-session
3
4 //in code file(e.g. app.js)
5 const sessions = require('express-session')

```

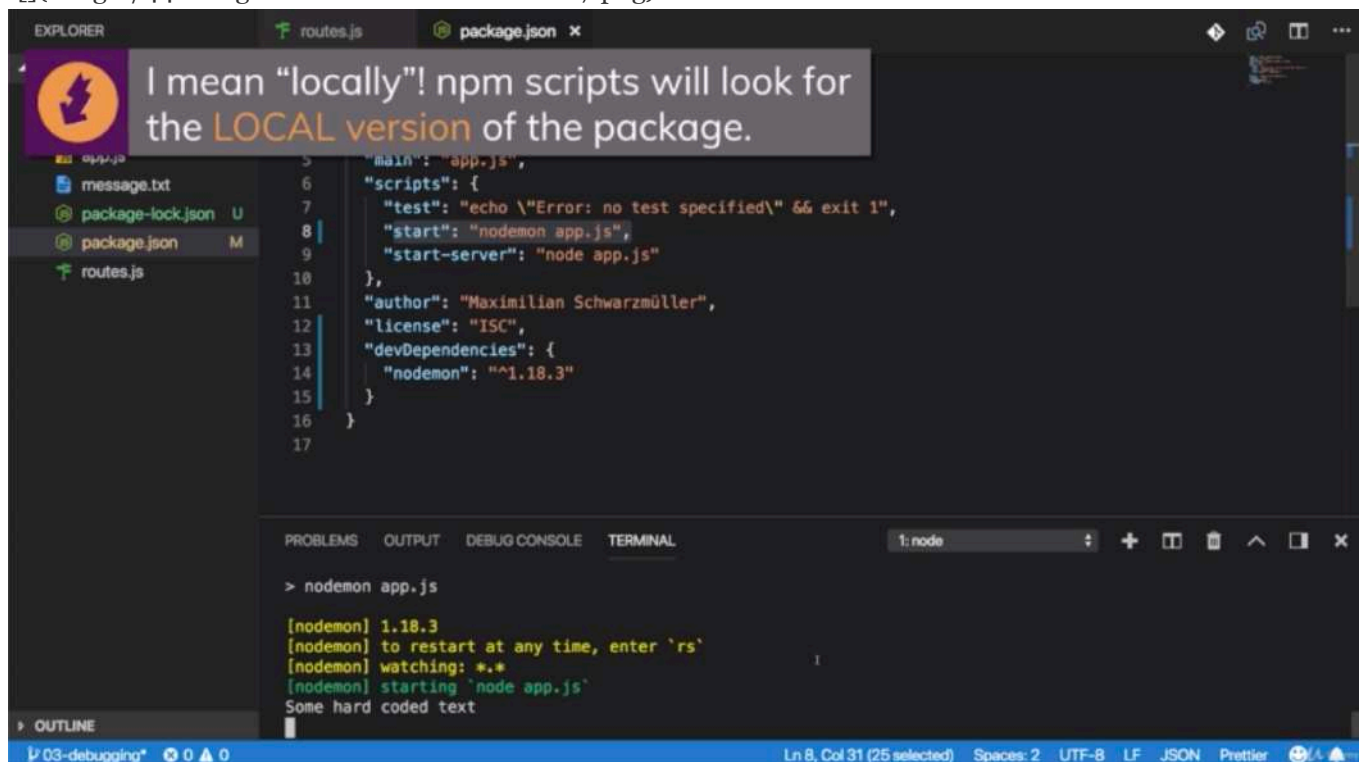
* Chapter 44: Using Nodemon For Auto Restart

- Nodemon is a utility tool and it allows us to run our application, our node application through this package which will run node app.js and will also watch our files for changes and restart the process for us if we do change something.

- so we can simply change start here, so the node app.js command to nodemon app.js.

- this will look for a nodemon tool which it will find in this project because we installed it here

- as a side note here, if you were to run nodemon app.js down here, you would get an error that this command is not found. because it's only installed in this project and not globally on your machine but the terminal will try to find this globally.



The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left shows a file tree with 'app.js', 'message.txt', 'package-lock.json', 'package.json', and 'routes.js'. The main editor area displays the 'package.json' file. A semi-transparent text box is overlaid on the top left of the editor, stating: "I mean 'locally'! npm scripts will look for the LOCAL version of the package." The 'package.json' content is as follows:

```
1 {
2   "name": "03-debugging",
3   "version": "1.0.0",
4   "description": "",
5   "main": "app.js",
6   "scripts": {
7     "test": "echo \\\"Error: no test specified\\\" && exit 1",
8     "start": "nodemon app.js",
9     "start-server": "node app.js"
10  },
11   "author": "Maximilian Schwarzmüller",
12   "license": "ISC",
13   "devDependencies": {
14     "nodemon": "^1.18.3"
15  }
16 }
```

The bottom panel shows the 'TERMINAL' tab with the following output:

```
> nodemon app.js
[nodemon] 1.18.3
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node app.js`
Some hard coded text
```

The status bar at the bottom indicates the file is '03-debugging', has 0 errors and 0 warnings, and is at line 8, column 31 (25 selected). It also shows settings for Spaces: 2, UTF-8, LF, JSON, and Prettier.

```
1 {
2   "name": "nodejs-complete-guide",
3   "version": "1.0.0",
4   "description": "Complete Node.js Guide",
5   "main": "app.js",
6   "scripts": {
7     "test": "echo \"Error: no test specified\" && exit 1",
8     "start": "node app.js",
9     "start-server": "node app.js"
10  },
11  "author": "Maximilian Schwarzmüller",
12  "license": "ISC",
13  "devDependencies": {
14    "nodemon": "^1.18.3"
15  }
16 }
17
```

```
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN nodejs-complete-guide@1.0.0 No repository field.

+ nodemon@1.18.3
added 300 packages from 151 contributors and audited 1355 packages in 18.437s
found 0 vulnerabilities

Maximilians-MBP:nodejs-complete-guide mschwarzmueller$
```

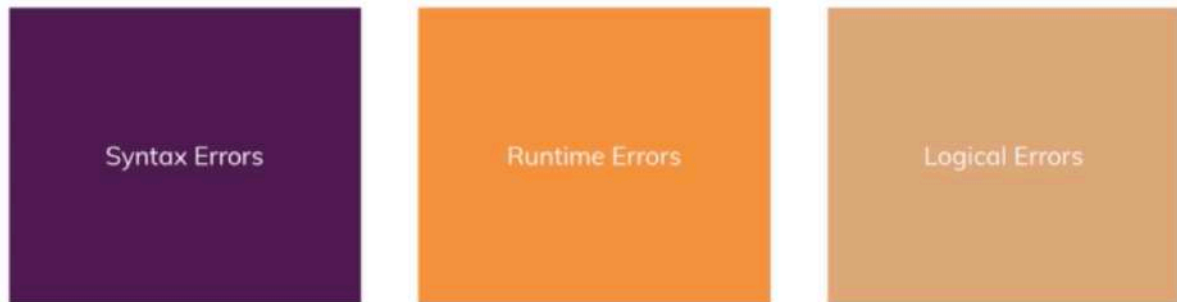
- but here, it will work because this will look globally. i mean 'locally!' npm scripts will look for the LOCAL version of the package. so if you run npm start, this will simply start the node server and output some extra information.
- and if you now go to your routes.js file and edit something and save that file, you see it's restarting.

* Chapter 45: Global & Local npm Package

- In the last lecture, we added 'nodemon' as a local dependency to our project
- The good thing about local dependencies is that you can share projects without the node_modules folder(where they are stored) and you can run 'npm install' in a project to then re-create that node_modules folder. This allows you to share only your source code, hence reducing the size of the shared project vastly.
- The attached course code snippets also are shared in that way, hence you need to run 'npm install' in the extracted packages to be able to run my code
- I showed that 'nodemon app.js' would not work in the terminal or command line because we don't use local dependencies there but global packages.
- You could install 'nodemon' globally if you wanted (this is not required though - because we can just run it locally) 'npm install -g nodemon' would do the trick Specifically the '-g' flag ensures that the package gets added as a global package which you now can use anywhere on your machine, directly from inside the terminal or command prompt.

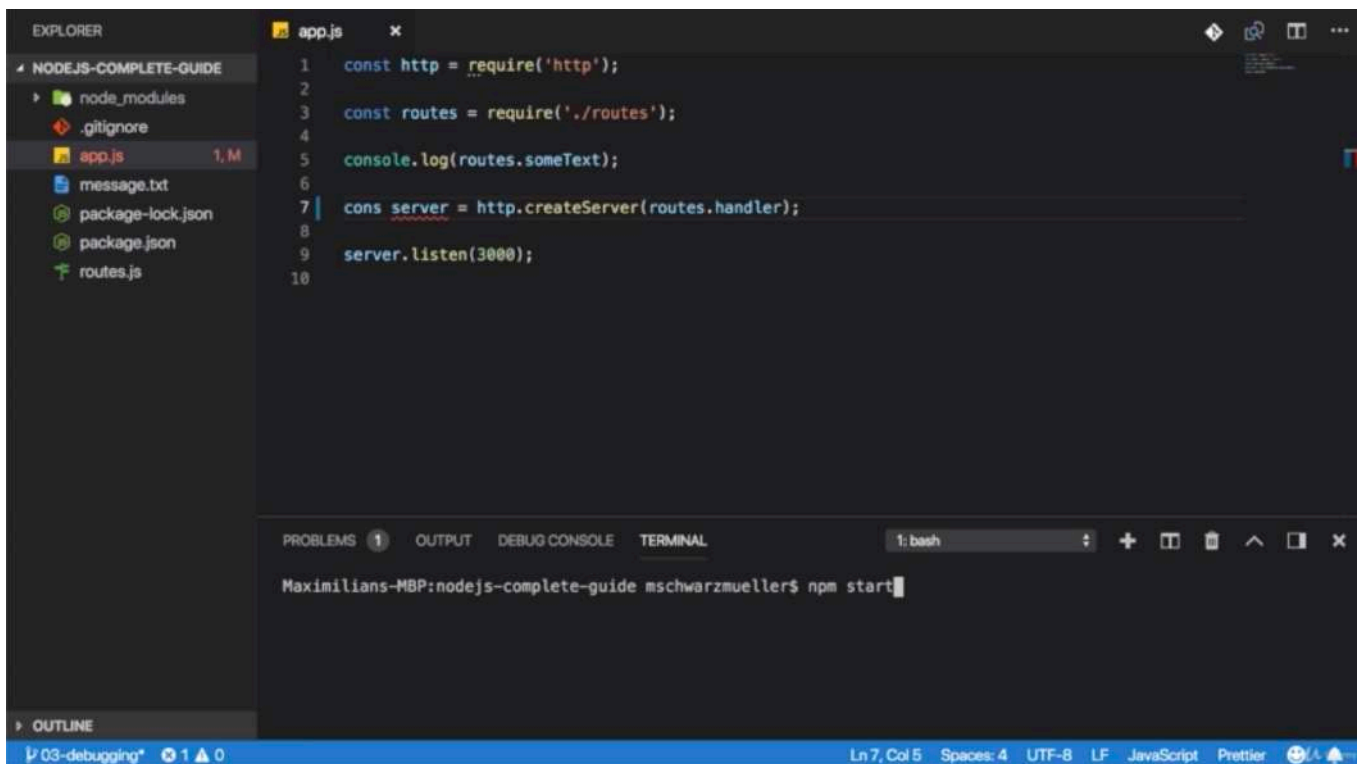
* Chapter 46: Understanding Different Error Types

Types of Errors



- Syntax Errors is that if you have a typo in your code or you forget like a closing curly brace or anything like that. so you have an error which should automatically be thrown when you try to run your project. most of time, these are pretty easy to fix but we will have a look at this and how we find and fix them in a second
- Runtime Errors are the errors which are not typos but where you try to execute some code which will just break when it runs.
- Logical Errors are most difficult ones because you will never see an error message. your app just doesn't work the way it should.

* Chapter 47: Finding & Fixing Syntax Errors



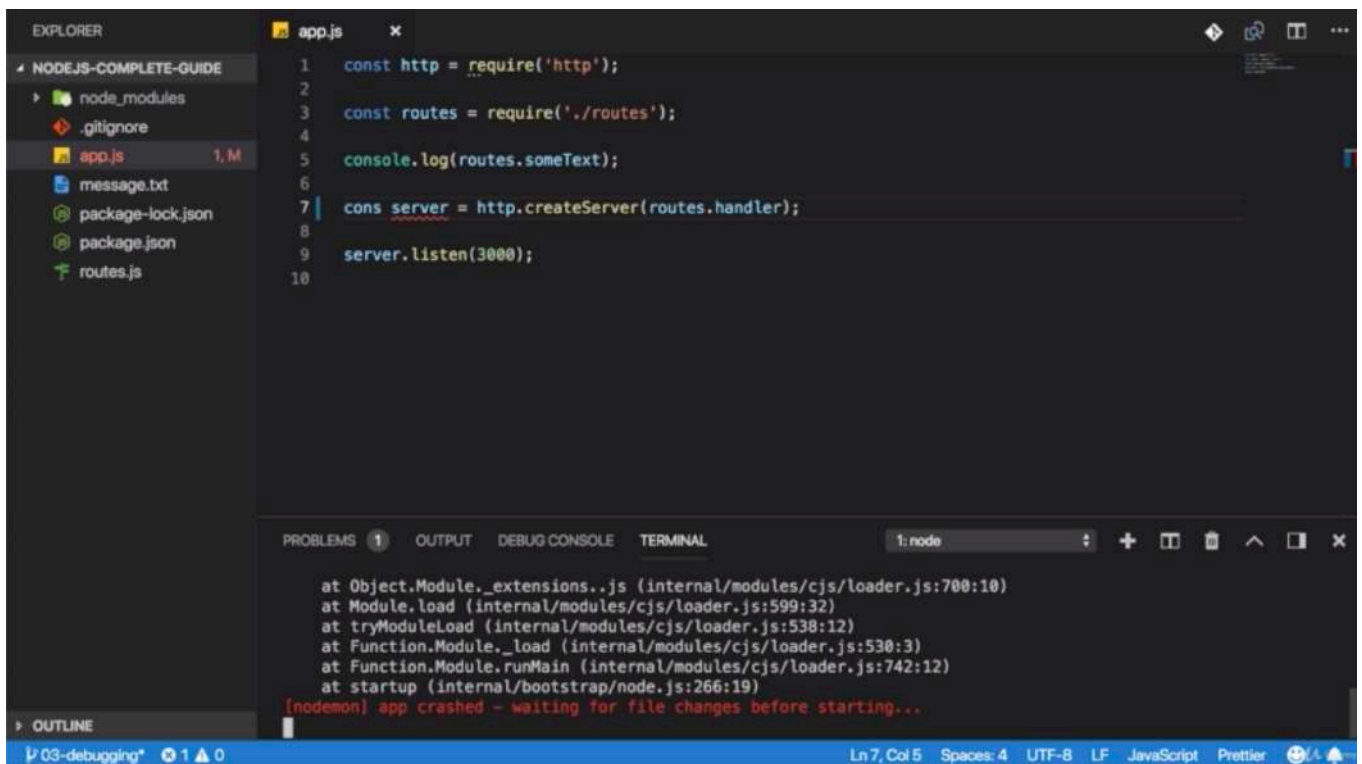
The image shows a VS Code editor window with a dark theme. The Explorer sidebar on the left shows a project named 'NODEJS-COMPLETE-GUIDE' with files: node_modules, .gitignore, app.js (1, M), message.txt, package-lock.json, package.json, and routes.js. The main editor area shows 'app.js' with the following code:

```
1  const http = require('http');
2
3  const routes = require('./routes');
4
5  console.log(routes.someText);
6
7  cons server = http.createServer(routes.handler);
8
9  server.listen(3000);
10
```

The bottom panel shows the 'TERMINAL' tab with a 'bash' shell. The command 'npm start' has been executed, and the output is:

```
Maximilians-MBP:nodejs-complete-guide mschwarzmueller$ npm start
```

The status bar at the bottom indicates 'Ln 7, Col 5', 'Spaces: 4', 'UTF-8', 'LF', 'JavaScript', and 'Prettier'.



The image shows the same VS Code editor window as above, but the terminal now shows an error message. The terminal tab is still '1: bash', but the output is:

```
at Object.Module._extensions..js (internal/modules/cjs/loader.js:700:10)
at Module.load (internal/modules/cjs/loader.js:599:32)
at tryModuleLoad (internal/modules/cjs/loader.js:538:12)
at Function.Module._load (internal/modules/cjs/loader.js:530:3)
at Function.Module.runMain (internal/modules/cjs/loader.js:742:12)
at startup (internal/bootstrap/node.js:266:19)
[nodemon] app crashed - waiting for file changes before starting...
```

The status bar at the bottom indicates 'Ln 7, Col 5', 'Spaces: 4', 'UTF-8', 'LF', 'JavaScript', and 'Prettier'.

```
1 const http = require('http');
2
3 const routes = require('./routes');
4
5 console.log(routes.someText);
6
7 const server = http.createServer(routes.handler);
8
9 server.listen(3000);
10
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL 1: node

/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/app.js:7
const server = http.createServer(routes.handler);

SyntaxError: Unexpected identifier
at new Script (vm.js:73:7)
at createScript (vm.js:245:10)
at Object.runInThisContext (vm.js:297:10)

03-debugging* 1 0 Ln 7, Col 5 Spaces: 4 UTF-8 LF JavaScript Prettier

- we see 'unexpected identifier' const server. so in the end, whilst it doesn't clearly tell us that we forget the t here because it's not smart enough to understand that this is the error here. so in such cases, as dumb as it sounds, you should simply take a closer look at this line and see what could be wrong there.


```
1 const fs = require('fs');
2
3 const requestHandler = (req, res) => {
4   const url = req.url;
5   const method = req.method;
6   if (url === '/') {
7     res.write('<html>');
8     res.write('<head><title>Enter Message</title></head>');
9     res.write(
10       '<body><form action="/message" method="POST"><input type="text" name="message"><button ty
11     );
12     res.write('</html>');
13     return res.end();
14   }
15
16   if (url === '/message' && method === 'POST') {
17     const body = [];
18     req.on('data', chunk => {
19       console.log(chunk);
20       body.push(chunk);
21     });
22   }
23 }
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL 1: bash

at Module.load (internal/modules/cjs/loader.js:599:32)
at tryModuleLoad (internal/modules/cjs/loader.js:538:12)
at Function.Module._load (internal/modules/cjs/loader.js:530:3)
at Function.Module.runMain (internal/modules/cjs/loader.js:742:12)
at startup (internal/bootstrap/node.js:266:19)
(nodemon) app crashed - waiting for file changes before starting...
^C
Maximilians-MBP:nodejs-complete-guide mschwarzmueller\$

03-debugging* 1 0 Ln 14, Col 3 Spaces: 2 UTF-8 LF JavaScript Prettier

```
42 // module.exports = {
43 //   handler: requestHandler,
44 //   someText: 'Some hard coded text'
45 // };
46
47 // module.exports.handler = requestHandler;
48 // module.exports.someText = 'Some text';
49
50 exports.handler = requestHandler;
51 exports.someText = 'Some hard coded text';
```

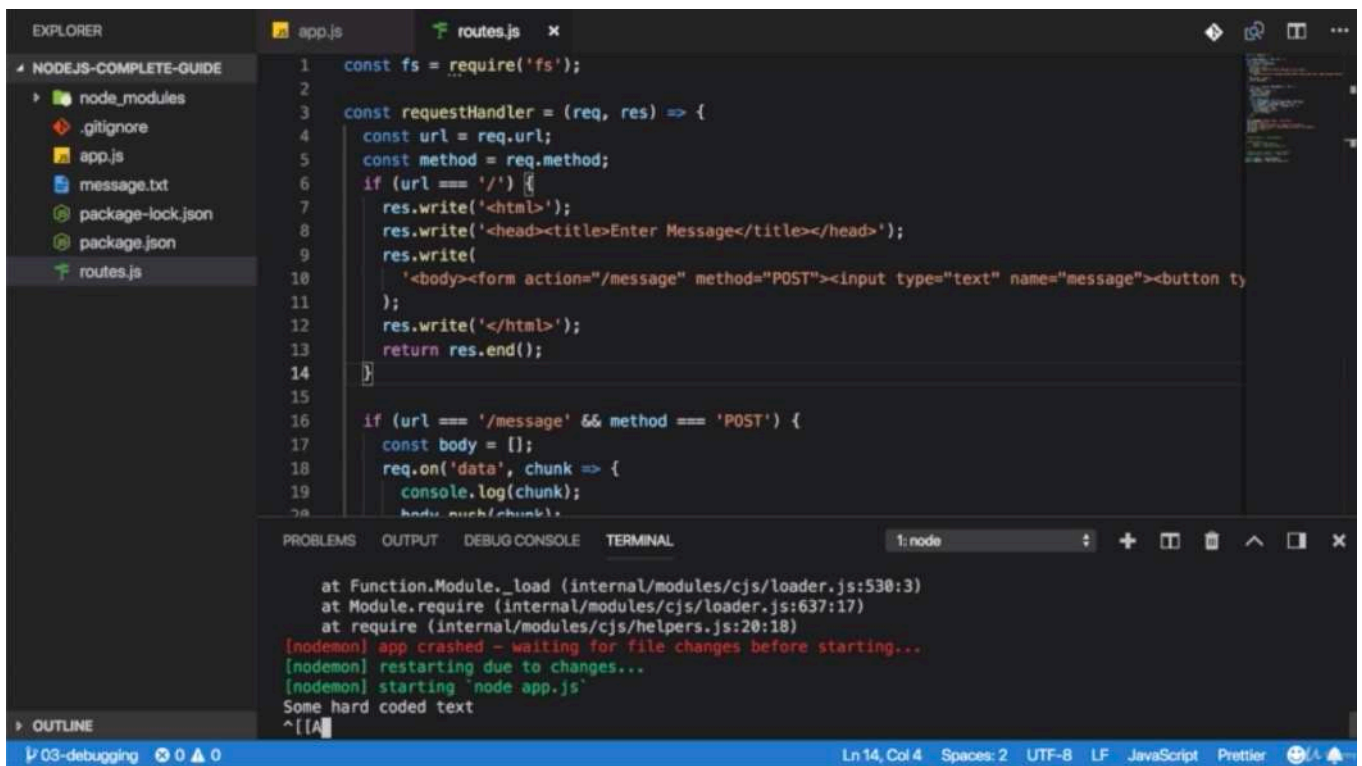
[ts] '}' expected.

at Module.load (internal/modules/cjs/loader.js:599:32)
at tryModuleLoad (internal/modules/cjs/loader.js:538:12)
at Function.Module._load (internal/modules/cjs/loader.js:530:3)
at Function.Module.runMain (internal/modules/cjs/loader.js:742:12)
at startup (internal/bootstrap/node.js:266:19)
[nodemon] app crashed - waiting for file changes before starting...

```
5 const method = req.method;
6 if (url === '/') {
7   res.write('<html>');
8   res.write('<head><title>Enter Message</title></head>');
9   res.write(
10     '<body><form action="/message" method="POST"><input type="text" name="message"><button ty
11   );
12   res.write('</html>');
13   return res.end();
14
15
16 if (url === '/message' && method === 'POST') {
17   const body = [];
18   req.on('data', chunk => {
19     console.log(chunk);
20     body.push(chunk);
21   });
22   return req.on('end', () => {
23     const parsedBody = Buffer.concat(body).toString();
```

SyntaxError: Unexpected token)
at new Script (vm.js:73:7)

[nodemon] watching: *.*
[nodemon] starting `node app.js`
/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/routes.js:52
});
^

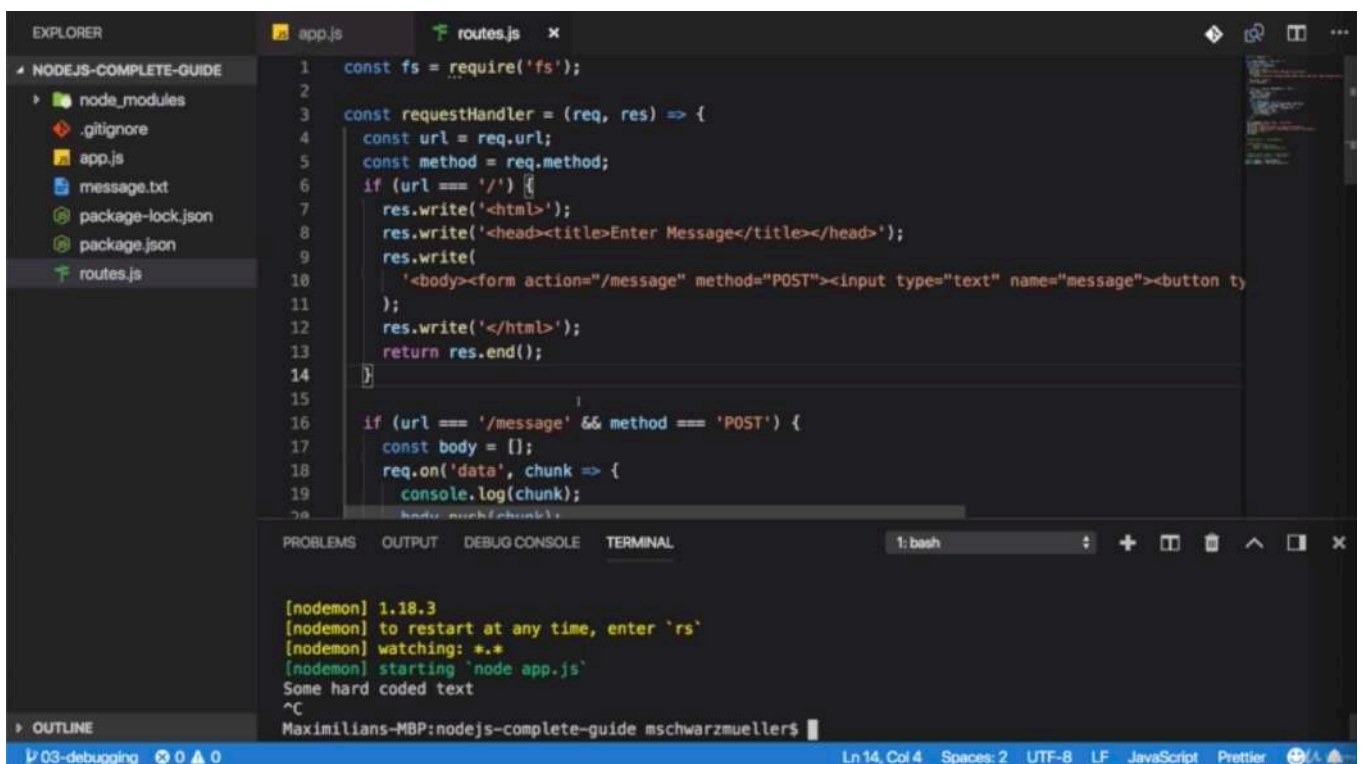


```
1 const fs = require('fs');
2
3 const requestHandler = (req, res) => {
4   const url = req.url;
5   const method = req.method;
6   if (url === '/') {
7     res.write('<html>');
8     res.write('<head><title>Enter Message</title></head>');
9     res.write(
10       '<body><form action="/message" method="POST"><input type="text" name="message"><button ty
11     );
12     res.write('</html>');
13     return res.end();
14   }
15
16   if (url === '/message' && method === 'POST') {
17     const body = [];
18     req.on('data', chunk => {
19       console.log(chunk);
20     });
21     req.on('end', () => {
22       // ...
23     });
24   }
25 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1: node

at Function.Module._load (internal/modules/cjs/loader.js:530:3)
at Module.require (internal/modules/cjs/loader.js:637:17)
at require (internal/modules/cjs/helpers.js:20:18)
[nodemon] app crashed - waiting for file changes before starting...
[nodemon] restarting due to changes...
[nodemon] starting 'node app.js'
Some hard coded text
^[[A



```
1 const fs = require('fs');
2
3 const requestHandler = (req, res) => {
4   const url = req.url;
5   const method = req.method;
6   if (url === '/') {
7     res.write('<html>');
8     res.write('<head><title>Enter Message</title></head>');
9     res.write(
10       '<body><form action="/message" method="POST"><input type="text" name="message"><button ty
11     );
12     res.write('</html>');
13     return res.end();
14   }
15
16   if (url === '/message' && method === 'POST') {
17     const body = [];
18     req.on('data', chunk => {
19       console.log(chunk);
20     });
21     req.on('end', () => {
22       // ...
23     });
24   }
25 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

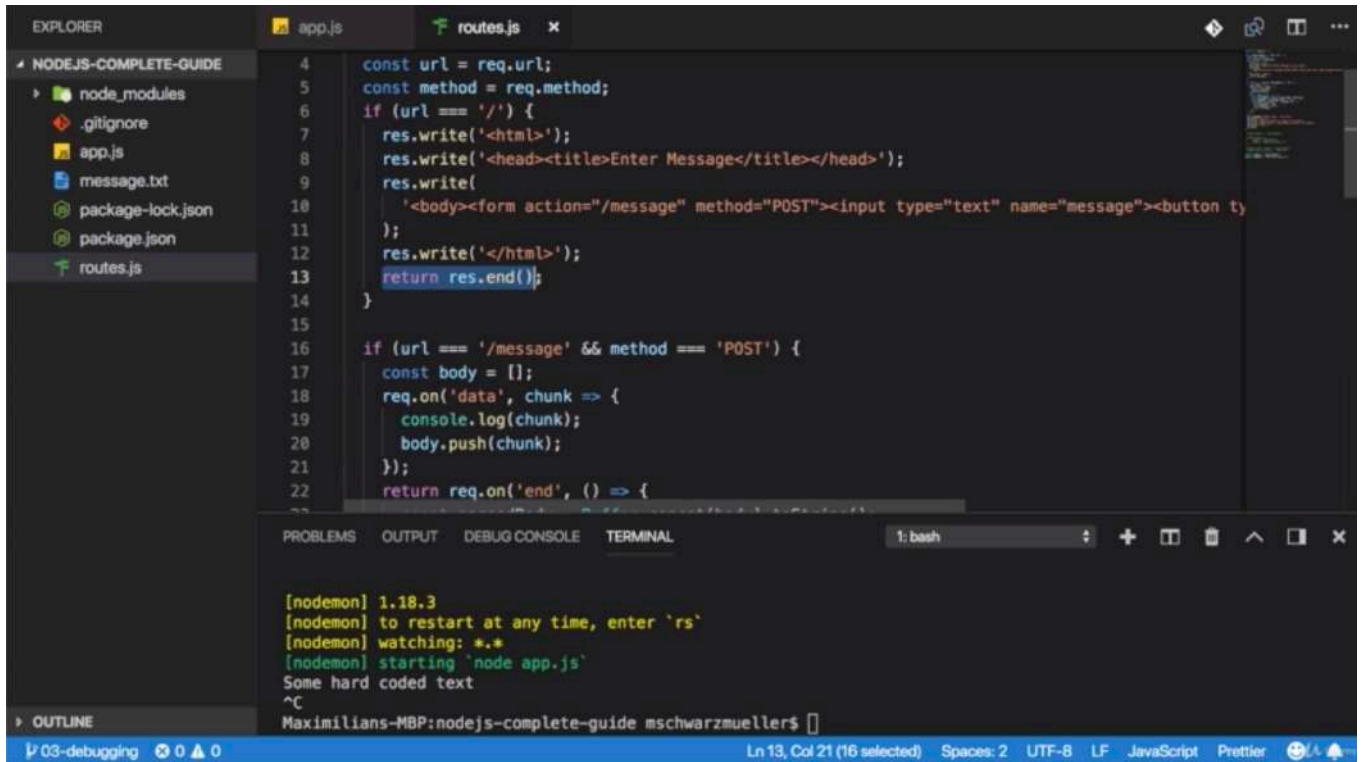
1: bash

[nodemon] 1.18.3
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting 'node app.js'
Some hard coded text
^C
Maximilians-MBP:nodejs-complete-guide mschwarzmueller\$

- if we jump there, we see at the end that the closing curly brace is expected. now it's not expected at this point where it's showing this message but it's expected somewhere in the file. you should then check your block statement like this if statement and see if there are all closed correctly.

- if you run this, you will see that it crashes and there, you also see unexpected token. and it points us at routes.js and then you see the line number too. so it points at line 52 and it shows us the same place as VS Code which is the wrong place. but in such cases, you should really take the IDE help, see that error message and then go through your file and see where could i be missing a character or where do i have an extra closing curly brace or something like that. and eventually you should be able to find this error and therefore fix it.

* Chapter 48: Dealing With Runtime Errors



```
4   const url = req.url;
5   const method = req.method;
6   if (url === '/') {
7     res.write('<html>');
8     res.write('<head><title>Enter Message</title></head>');
9     res.write(
10      '<body><form action="/message" method="POST"><input type="text" name="message"><button ty
11    );
12    res.write('</html>');
13    return res.end();
14  }

16  if (url === '/message' && method === 'POST') {
17    const body = [];
18    req.on('data', chunk => {
19      console.log(chunk);
20      body.push(chunk);
21    });
22    return req.on('end', () => {
```

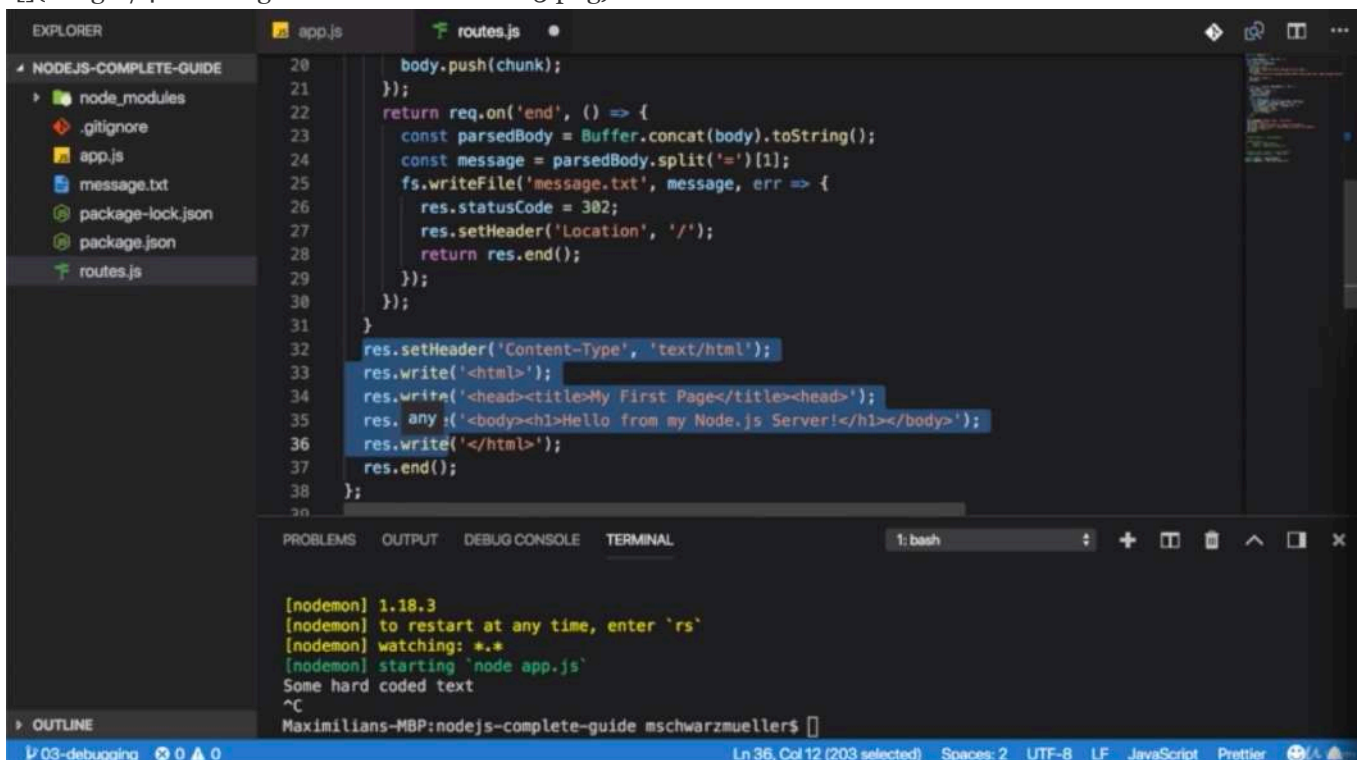
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1: bash

```
[nodemon] 1.18.3
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node app.js`
Some hard coded text
^C
Maximilians-MBP:nodejs-complete-guide mschwarzmuellers$
```

- The Great example of the runtime errors can be shown with the `res.write()` method.

- You have to return to prevent the execution of the code after if statement. otherwise this is something node specific. you would end the request here but the code execution would continue

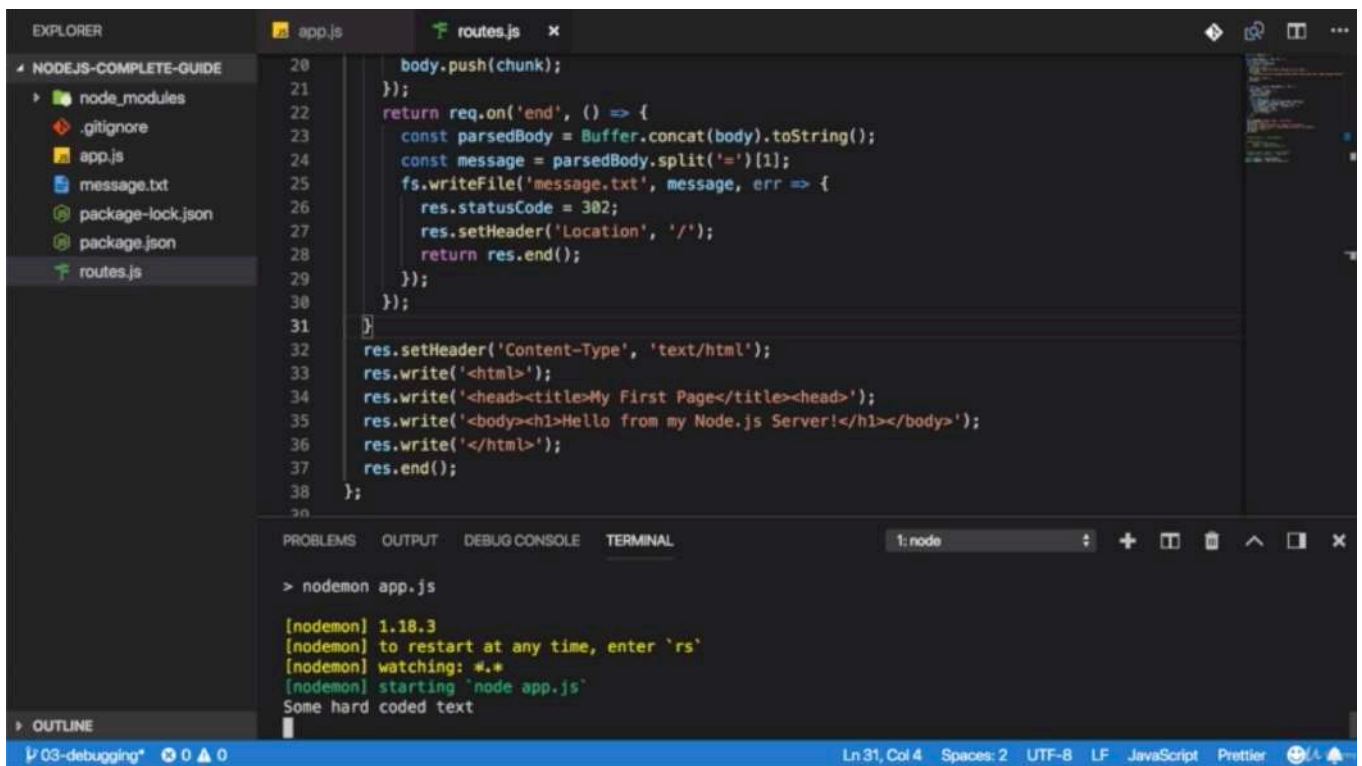


```
20    body.push(chunk);
21  });
22  return req.on('end', () => {
23    const parsedBody = Buffer.concat(body).toString();
24    const message = parsedBody.split('=')[1];
25    fs.writeFile('message.txt', message, err => {
26      res.statusCode = 302;
27      res.setHeader('Location', '/');
28      return res.end();
29    });
30  });
31 }
32 res.setHeader('Content-Type', 'text/html');
33 res.write('<html>');
34 res.write('<head><title>My First Page</title></head>');
35 res.any('<body><h1>Hello from my Node.js Server!</h1></body>');
36 res.write('</html>');
37 res.end();
38 };
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1: bash

```
[nodemon] 1.18.3
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node app.js`
Some hard coded text
^C
Maximilians-MBP:nodejs-complete-guide mschwarzmuellers$
```

```
20 body.push(chunk);
21 });
22 return req.on('end', () => {
23   const parsedBody = Buffer.concat(body).toString();
24   const message = parsedBody.split('=')[1];
25   fs.writeFile('message.txt', message, err => {
26     res.statusCode = 302;
27     res.setHeader('Location', '/');
28     return res.end();
29   });
30 });
31
32 res.setHeader('Content-Type', 'text/html');
33 res.write('<html>');
34 res.write('<head><title>My First Page</title><head>');
35 res.write('<body><h1>Hello from my Node.js Server!</h1></body>');
36 res.write('</html>');
37 res.end();
38 };
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1: node

> nodemon app.js

[nodemon] 1.18.3
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node app.js`
Some hard coded text

- and eventually we would reach this line and this is no syntax error, this is correct but if we run our code, seems to work right.



This page isn't working
localhost didn't send any data.
(See console for details)

Reload

```
20 body.push(chunk);
21 });
22 return req.on('end', () => {
23   const parsedBody = Buffer.concat(body).toString();
24   const message = parsedBody.split('=')[1];
25   fs.writeFile('message.txt', message, err => {
26     res.statusCode = 302;
27     res.setHeader('Location', '/');
28     return res.end();
29   });
30 });
31 }
32 res.setHeader('Content-Type', 'text/html');
33 res.write('<html>');
34 res.write('<head><title>My First Page</title></head>');
```

```
_http_outgoing.js:470
  throw new ERR_HTTP_HEADERS_SENT('set');
        ^

Error [ERR_HTTP_HEADERS_SENT]: Cannot set headers after they are sent to the client
    at ServerResponse.setHeader (_http_outgoing.js:470:11)
    at Server.requestHandler (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/routes.js:32:7)
    at Server.emit (events.js:182:13)
    at parserOnIncoming (_http_server.js:652:12)
    at HTTPParser.parserOnHeadersComplete (_http_common.js:109:17)
[nodemon] app crashed - waiting for file changes before starting...
```

- but if we visit our page, eventually it breaks here and this is the point where it should go back to your code and should find an error message there too.

- this error message is something you shouldn't just take and paste into the Q&A section but actually read it. a lot of the error message are helpful.


```
_http_outgoing.js:470
  throw new ERR_HTTP_HEADERS_SENT('set');
        ^

Error [ERR_HTTP_HEADERS_SENT]: Cannot set headers after they are sent to the client
    at ServerResponse.setHeader (_http_outgoing.js:470:11)
    at Server.requestHandler (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/routes.js:32:7)
    at Server.emit (events.js:182:13)
    at parserOnIncoming (_http_server.js:652:12)
    at HTTPParser.parserOnHeadersComplete (_http_common.js:109:17)
[nodemon] app crashed - waiting for file changes before starting...
```

- at the bottom, you always find uninteresting stuff i would say but you have to scroll to the top of the error message and all of a sudden, it should get more meaningful.


```
20 body.push(chunk);
21 });
22 return req.on('end', () => {
23   const parsedBody = Buffer.concat(body).toString();
24   const message = parsedBody.split('=')[1];
25   fs.writeFile('message.txt', message, err => {
26     res.statusCode = 302;
27     res.setHeader('Location', '/');
28     return res.end();
29   });
30 });
31 }
32 res.setHeader('Content-Type', 'text/html');
33 res.write('<html>');
34 res.write('<head><title>My First Page</title><head>');
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: node

```
[nodemon] watching: *.*
[nodemon] starting `node app.js`
Some hard coded text
_http_outgoing.js:470
  throw new ERR_HTTP_HEADERS_SENT('set');
        ^
Error [ERR_HTTP_HEADERS_SENT]: Cannot set headers after they are sent to the client
    at ServerResponse.setHeader (_http_outgoing.js:470:11)
    at Server.requestHandler (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/routes.js:32:7)
    at Server.emit (events.js:182:13)
    at parserOnIncoming (_http_server.js:652:12)
```

- for example, you see the error code which already indicates that it's something with headers being sent


```
20 body.push(chunk);
21 });
22 return req.on('end', () => {
23   const parsedBody = Buffer.concat(body).toString();
24   const message = parsedBody.split('=')[1];
25   fs.writeFile('message.txt', message, err => {
26     res.statusCode = 302;
27     res.setHeader('Location', '/');
28     return res.end();
29   });
30 });
31 }
32 res.setHeader('Content-Type', 'text/html');
33 res.write('<html>');
34 res.write('<head><title>My First Page</title><head>');
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: node

```
[nodemon] starting `node app.js`
Some hard coded text
_http_outgoing.js:470
  throw new ERR_HTTP_HEADERS_SENT('set');
        ^
Error [ERR_HTTP_HEADERS_SENT]: Cannot set headers after they are sent to the client
    at ServerResponse.setHeader (_http_outgoing.js:470:11)
    at Server.requestHandler (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/routes.js:32:7)
    at Server.emit (events.js:182:13)
    at parserOnIncoming (_http_server.js:652:12)
    at HTTPParser.parserOnHeadersComplete (_http_common.js:109:17)
```

```
EXPLORER  app.js  routes.js x
NODEJS-COMPLETE-GUIDE
node_modules
.gitignore
app.js
message.txt
package-lock.json
package.json
routes.js

20 body.push(chunk);
21 });
22 return req.on('end', () => {
23   const parsedBody = Buffer.concat(body).toString();
24   const message = parsedBody.split('=')[1];
25   fs.writeFile('message.txt', message, err => {
26     res.statusCode = 302;
27     res.setHeader('Location', '/');
28     return res.end();
29   });
30 });
31 }
32 res.setHeader('Content-Type', 'text/html');
33 res.write('<html>');
34 res.write('<head><title>My First Page</title><head>');

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
1: node

[nodemon] starting 'node app.js'
Some hard coded text
_http_outgoing.js:470
  throw new ERR_HTTP_HEADERS_SENT('set');
        ^

Error [ERR_HTTP_HEADERS_SENT]: Cannot set headers after they are sent to the client
    at ServerResponse.setHeader (_http_outgoing.js:470:11)
    at Server.requestHandler (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/routes.js:32:7)
    at Server.emit (events.js:182:13)
    at parserOnIncoming (_http_server.js:652:12)
    at HTTPParser.parserOnHeadersComplete (_http_common.js:109:17)

03-debugging* 0 0 0  Ln 31, Col 4  Spaces: 2  UTF-8  LF  JavaScript  Prettier
```

- and here you find a detailed error message, cannot setHeaders after they are sent to the client. then you see that it was caused by a call to setHeader and unfortunately, the line numbers don't help you.


```
EXPLORER  app.js  routes.js x
NODEJS-COMPLETE-GUIDE
node_modules
.gitignore
app.js
message.txt
package-lock.json
package.json
routes.js

20 body.push(chunk);
21 });
22 return req.on('end', () => {
23   const parsedBody = Buffer.concat(body).toString();
24   const message = parsedBody.split('=')[1];
25   fs.writeFile('message.txt', message, err => {
26     res.statusCode = 302;
27     res.setHeader('Location', '/');
28     return res.end();
29   });
30 });
31 }
32 res.setHeader('Content-Type', 'text/html');
33 res.write('<html>');
34 res.write('<head><title>My First Page</title><head>');

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
1: node

[nodemon] starting 'node app.js'
Some hard coded text
_http_outgoing.js:470
  throw new ERR_HTTP_HEADERS_SENT('set');
        ^

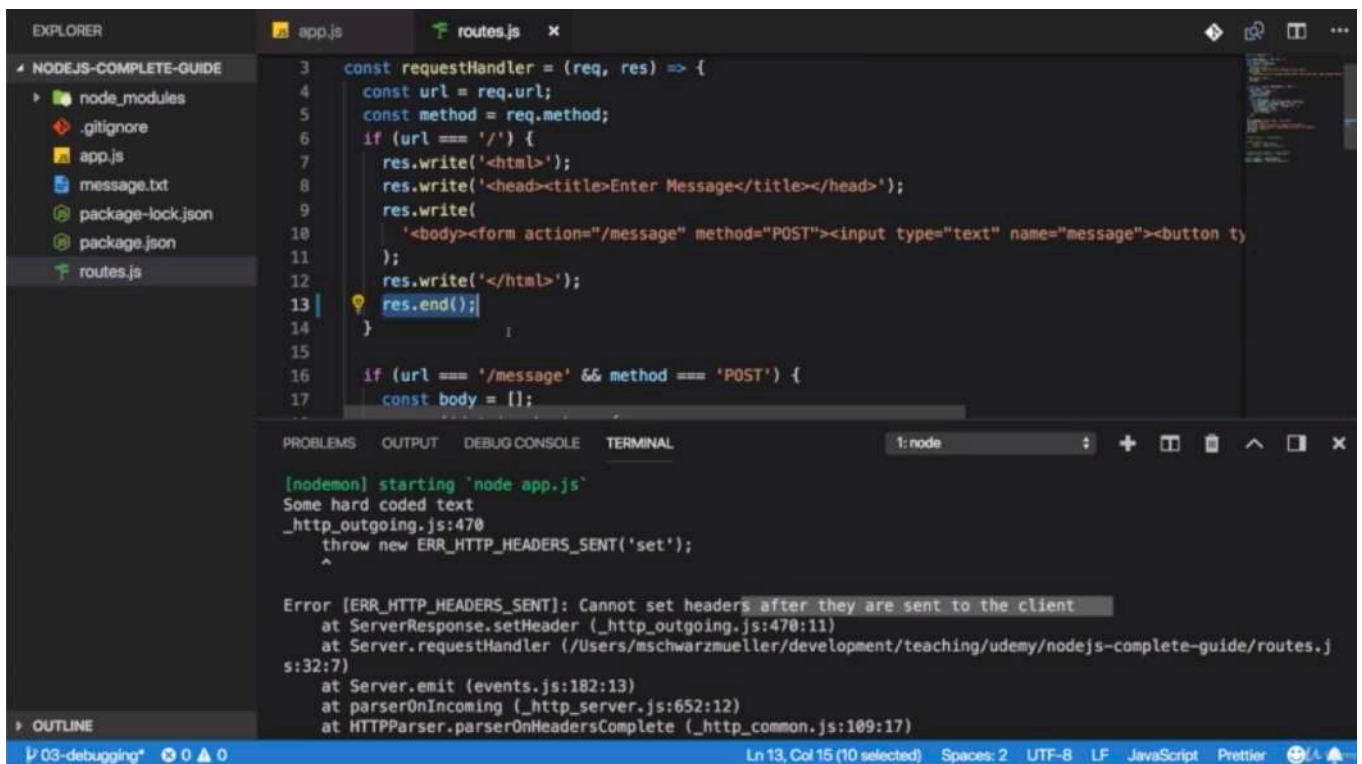
Error [ERR_HTTP_HEADERS_SENT]: Cannot set headers after they are sent to the client
    at ServerResponse.setHeader (_http_outgoing.js:470:11)
    at Server.requestHandler (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/routes.js:32:7)
    at Server.emit (events.js:182:13)
    at parserOnIncoming (_http_server.js:652:12)
    at HTTPParser.parserOnHeadersComplete (_http_common.js:109:17)

03-debugging* 0 0 0  Ln 31, Col 4  Spaces: 2  UTF-8  LF  JavaScript  Prettier
```

- but then it does help you here. it's in the requestHandler and hter it points at the routes.js file line 32. and now this is the point where you can dive in and see

- i'm calling setHeader here, since it's complaining that i can't set them after they are sent to the client

- so i should look well in the code before this statement because it looks like i'm finishing my response there for example.



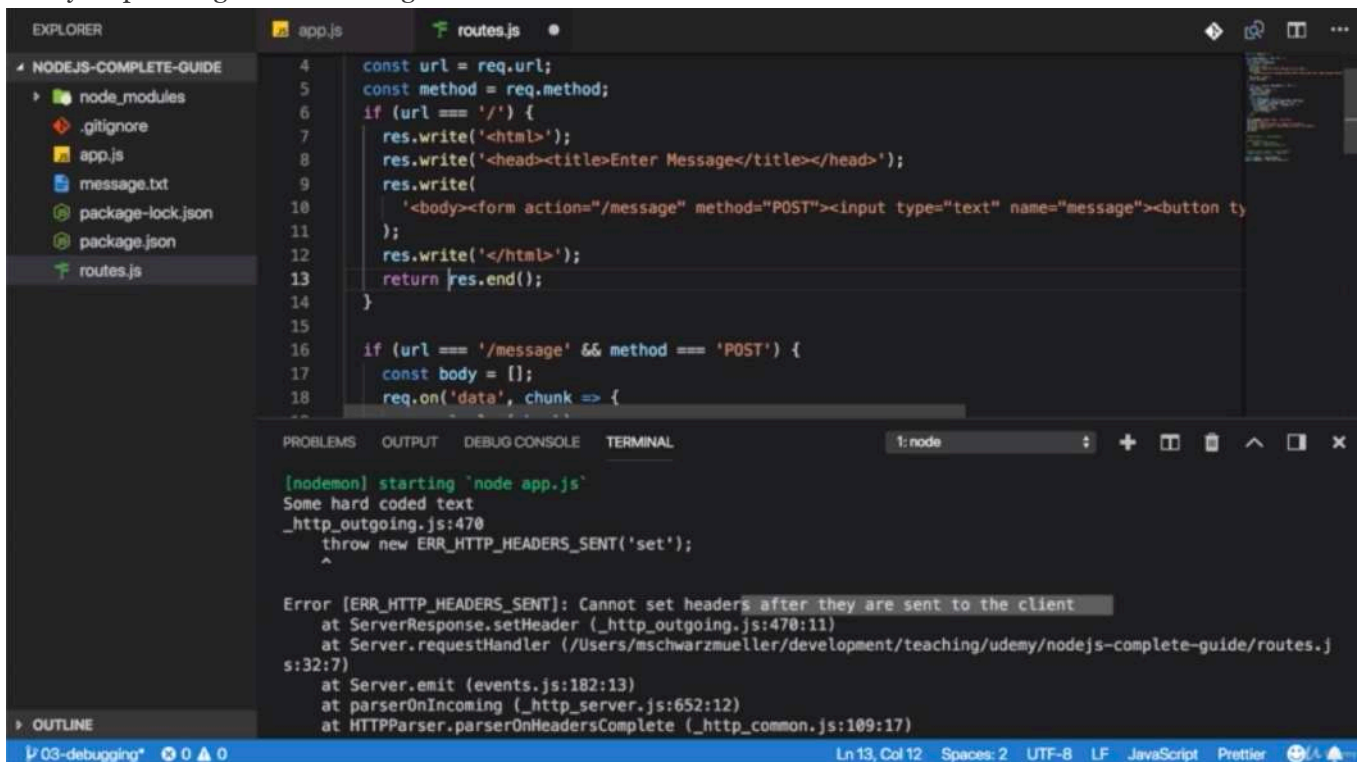
```
3 const requestHandler = (req, res) => {
4   const url = req.url;
5   const method = req.method;
6   if (url === '/') {
7     res.write('<html>');
8     res.write('<head><title>Enter Message</title></head>');
9     res.write(
10      '<body><form action="/message" method="POST"><input type="text" name="message"><button ty
11    );
12    res.write('</html>');
13    res.end();
14  }
15
16  if (url === '/message' && method === 'POST') {
17    const body = [];
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1: node

```
[nodemon] starting `node app.js`
Some hard coded text
_http_outgoing.js:470
  throw new ERR_HTTP_HEADERS_SENT('set');
      ^
Error [ERR_HTTP_HEADERS_SENT]: Cannot set headers after they are sent to the client
    at ServerResponse.setHeader (_http_outgoing.js:470:11)
    at Server.requestHandler (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/routes.js:32:7)
    at Server.emit (events.js:182:13)
    at parserOnIncoming (_http_server.js:652:12)
    at HTTPParser.parserOnHeadersComplete (_http_common.js:109:17)
```

- and indeed at some point, you should find this statement and see i do actually not finish my code execution after this statement which is not parse a problem but it becomes a problem if in the following code, i then work on my response again as i'm doing here



```
4   const url = req.url;
5   const method = req.method;
6   if (url === '/') {
7     res.write('<html>');
8     res.write('<head><title>Enter Message</title></head>');
9     res.write(
10      '<body><form action="/message" method="POST"><input type="text" name="message"><button ty
11    );
12    res.write('</html>');
13    return res.end();
14  }
15
16  if (url === '/message' && method === 'POST') {
17    const body = [];
18    req.on('data', chunk => {
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1: node

```
[nodemon] starting `node app.js`
Some hard coded text
_http_outgoing.js:470
  throw new ERR_HTTP_HEADERS_SENT('set');
      ^
Error [ERR_HTTP_HEADERS_SENT]: Cannot set headers after they are sent to the client
    at ServerResponse.setHeader (_http_outgoing.js:470:11)
    at Server.requestHandler (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/routes.js:32:7)
    at Server.emit (events.js:182:13)
    at parserOnIncoming (_http_server.js:652:12)
    at HTTPParser.parserOnHeadersComplete (_http_common.js:109:17)
```

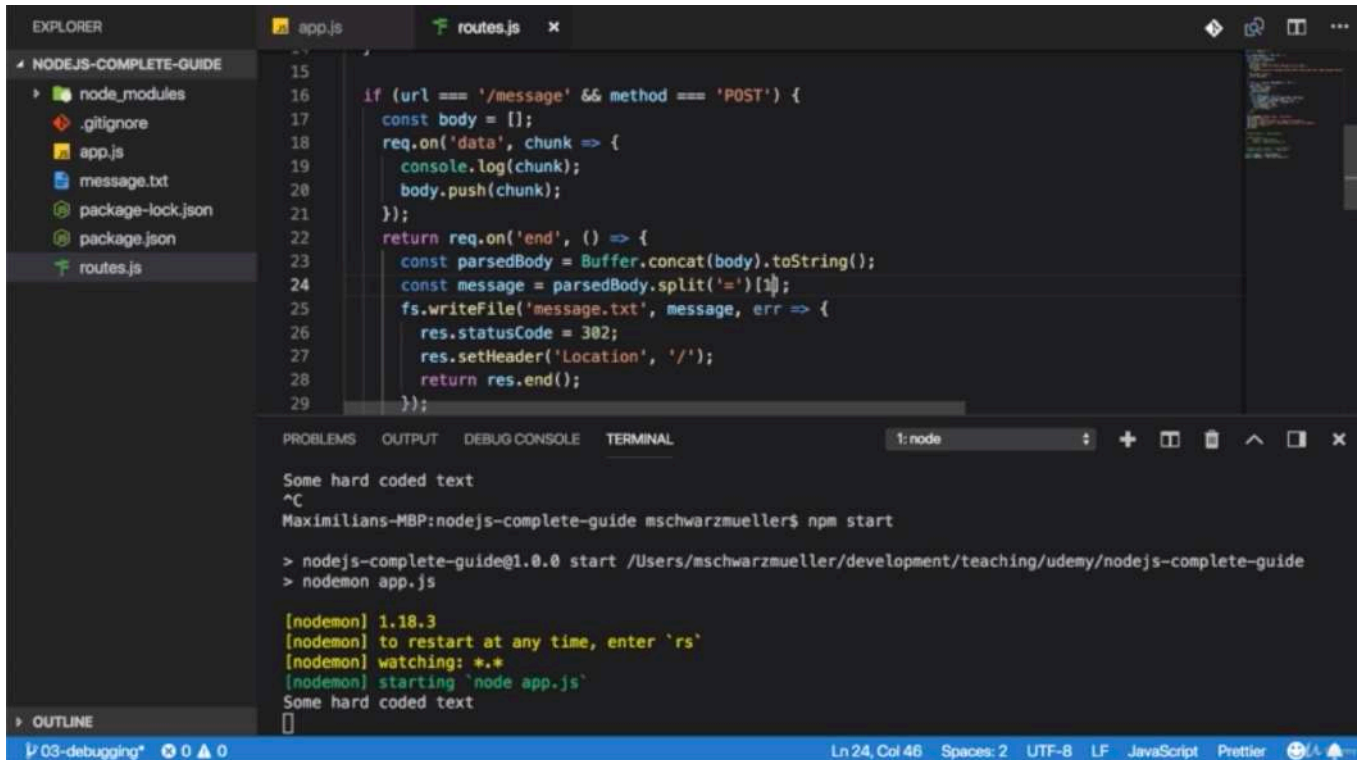

- so this is the point where we can fix this by stoppin the code execution

- or by wrapping this in an extra if statement which is guaranteed to not run

if we make it into this statement

* Chapter 49: Logical Errors

- Logical error is the most difficult one to fix because it will not cause an error message, your app will just not behave the way you expect it to.



```
EXPLORER  app.js  routes.js x
NODEJS-COMPLETE-GUIDE
node_modules
.gitignore
app.js
message.txt
package-lock.json
package.json
routes.js

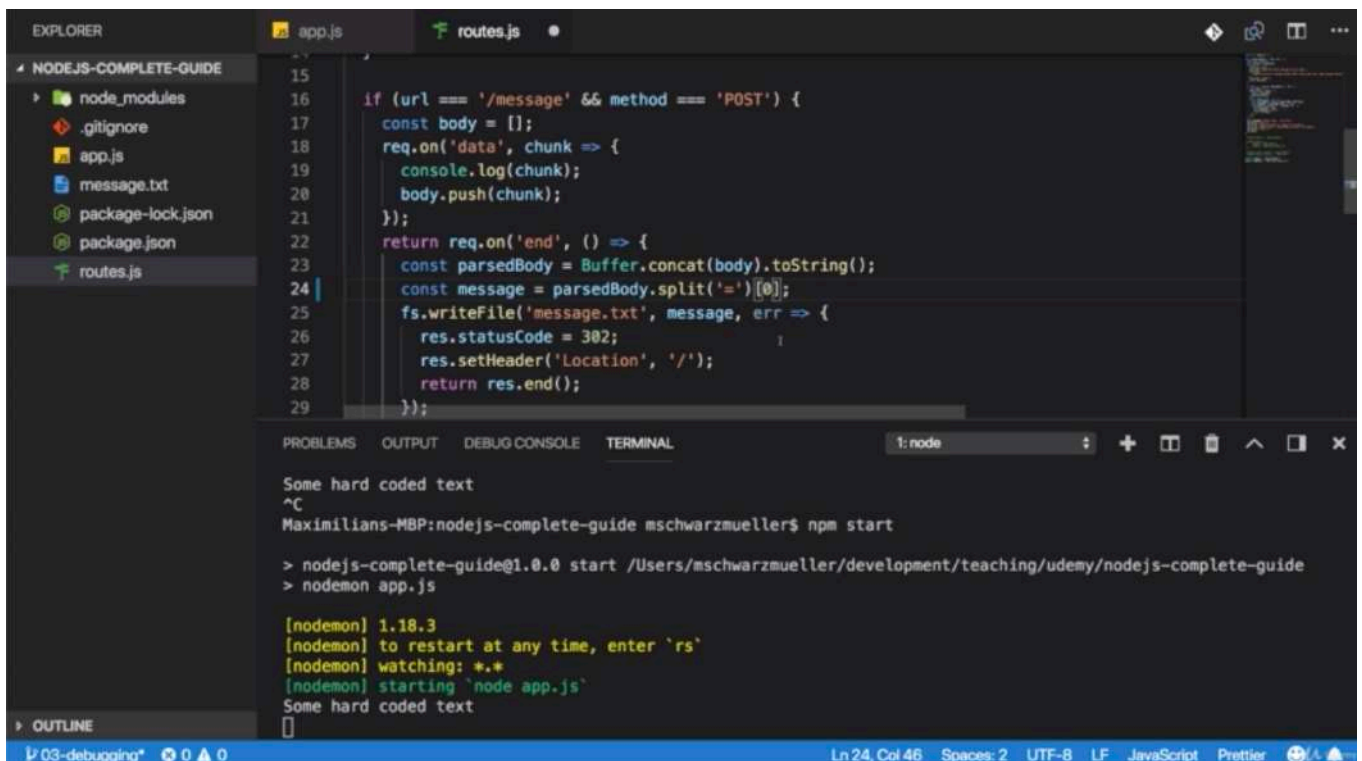
15
16 if (url === '/message' && method === 'POST') {
17   const body = [];
18   req.on('data', chunk => {
19     console.log(chunk);
20     body.push(chunk);
21   });
22   return req.on('end', () => {
23     const parsedBody = Buffer.concat(body).toString();
24     const message = parsedBody.split('=')[1];
25     fs.writeFile('message.txt', message, err => {
26       res.statusCode = 302;
27       res.setHeader('Location', '/');
28       return res.end();
29     });
30   });
31 }

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
1: node

Some hard coded text
^C
Maximilians-MBP:nodejs-complete-guide mschwarzmueller$ npm start

> nodejs-complete-guide@1.0.0 start /Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide
> nodemon app.js

[nodemon] 1.18.3
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node app.js`
Some hard coded text
[]
```



```
EXPLORER  app.js  routes.js
NODEJS-COMPLETE-GUIDE
node_modules
.gitignore
app.js
message.txt
package-lock.json
package.json
routes.js

15
16 if (url === '/message' && method === 'POST') {
17   const body = [];
18   req.on('data', chunk => {
19     console.log(chunk);
20     body.push(chunk);
21   });
22   return req.on('end', () => {
23     const parsedBody = Buffer.concat(body).toString();
24     const message = parsedBody.split('=')[0];
25     fs.writeFile('message.txt', message, err => {
26       res.statusCode = 302;
27       res.setHeader('Location', '/');
28       return res.end();
29     });
30   });
31 }

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
1: node

Some hard coded text
^C
Maximilians-MBP:nodejs-complete-guide mschwarzmueller$ npm start

> nodejs-complete-guide@1.0.0 start /Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide
> nodemon app.js

[nodemon] 1.18.3
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node app.js`
Some hard coded text
[]
```

- for example, let's say i have the first element being stored in message here, and now we are actually storing the wrong element.

EXPLORER

NODEJS-COMPLETE-GUIDE

node_modules

.gitignore

app.js

message.txt

package-lock.json

package.json

routes.js

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

```
if (url === '/message' && method === 'POST') {
  const body = [];
  req.on('data', chunk => {
    console.log(chunk);
    body.push(chunk);
  });
  return req.on('end', () => {
    const parsedBody = Buffer.concat(body).toString();
    const message = parsedBody.split('=')[0];
    fs.writeFile('message.txt', message, err => {
      res.statusCode = 302;
      res.setHeader('Location', '/');
      return res.end();
    });
  });
}
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

1: node

+

□

□

□

□

×

```
> nodejs-complete-guide@1.0.0 start /Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide
> nodemon app.js

[nodemon] 1.18.3
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node app.js`
Some hard coded text
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
Some hard coded text
```

OUTLINE

VS-debugging*

0

▲

0

Ln 24, Col 47

Spaces: 2

UTF-8

LF

JavaScript

Prettier

Test

Send

The screenshot shows a VS Code editor with a project named 'NODEJS-COMPLETE-GUIDE'. The Explorer sidebar on the left shows files: node_modules, .gitignore, app.js, message.txt, package-lock.json, package.json, and routes.js. The main editor area shows 'message.txt' with the content '1 | message'. The bottom panel shows the 'TERMINAL' tab with the following output:

```
> nodejs-complete-guide@1.0.0 start /Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide
> nodemon app.js

[nodemon] 1.18.3
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node app.js`
Some hard coded text
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
Some hard coded text
<Buffer 6d 65 73 73 61 67 65 3d 54 65 73 74>
```

- so if i save this and my server restarts, and type something and go to message.txt, i can see message there.
- this is a logical error because we got no error message. but the app is not behaving the way we want it to
- we know that we used the wrong index because i just changed that
- with node.js debugger, you can fix it.

The screenshot shows the VS Code editor with the 'app.js' file open. The code is as follows:

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

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

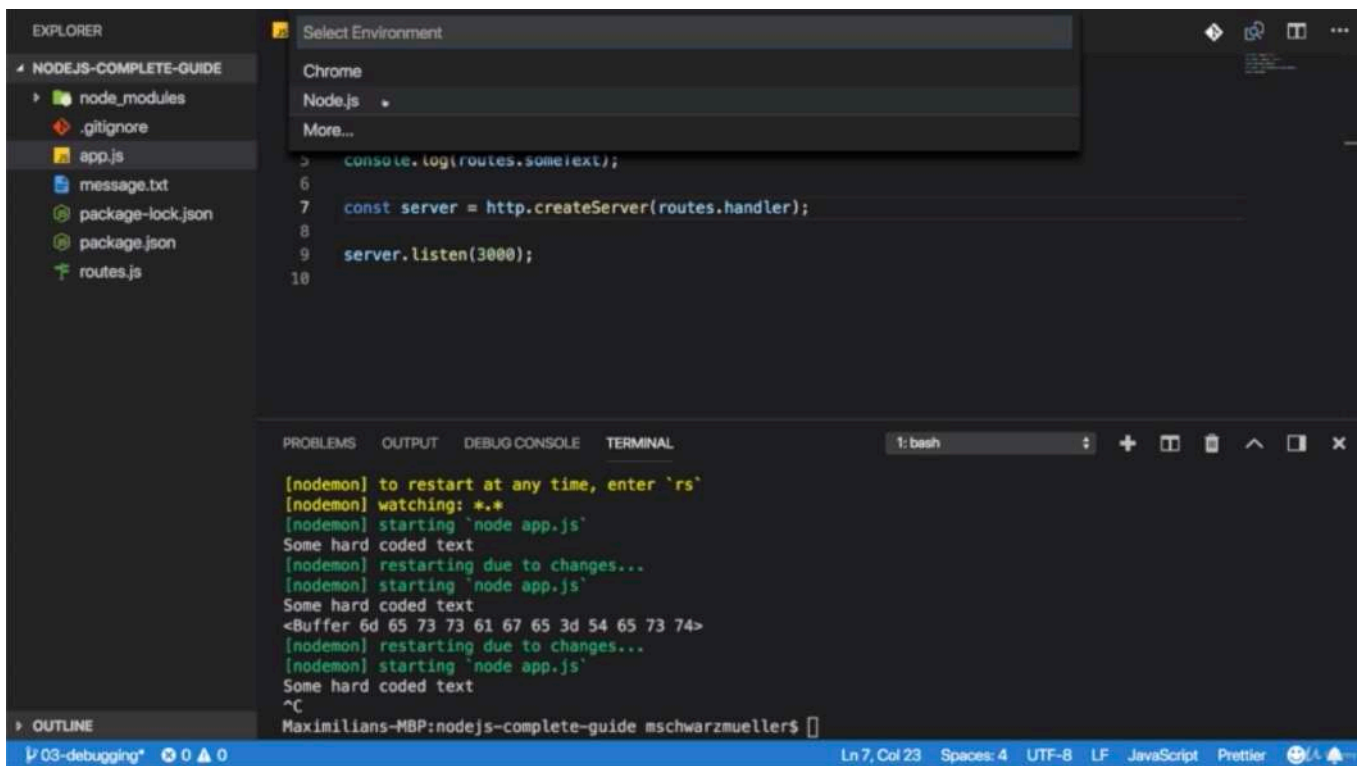
console.log(routes.someText);

const server = http.createServer(routes.handler);

server.listen(3000);
```

The bottom panel shows the 'TERMINAL' tab with the following output:

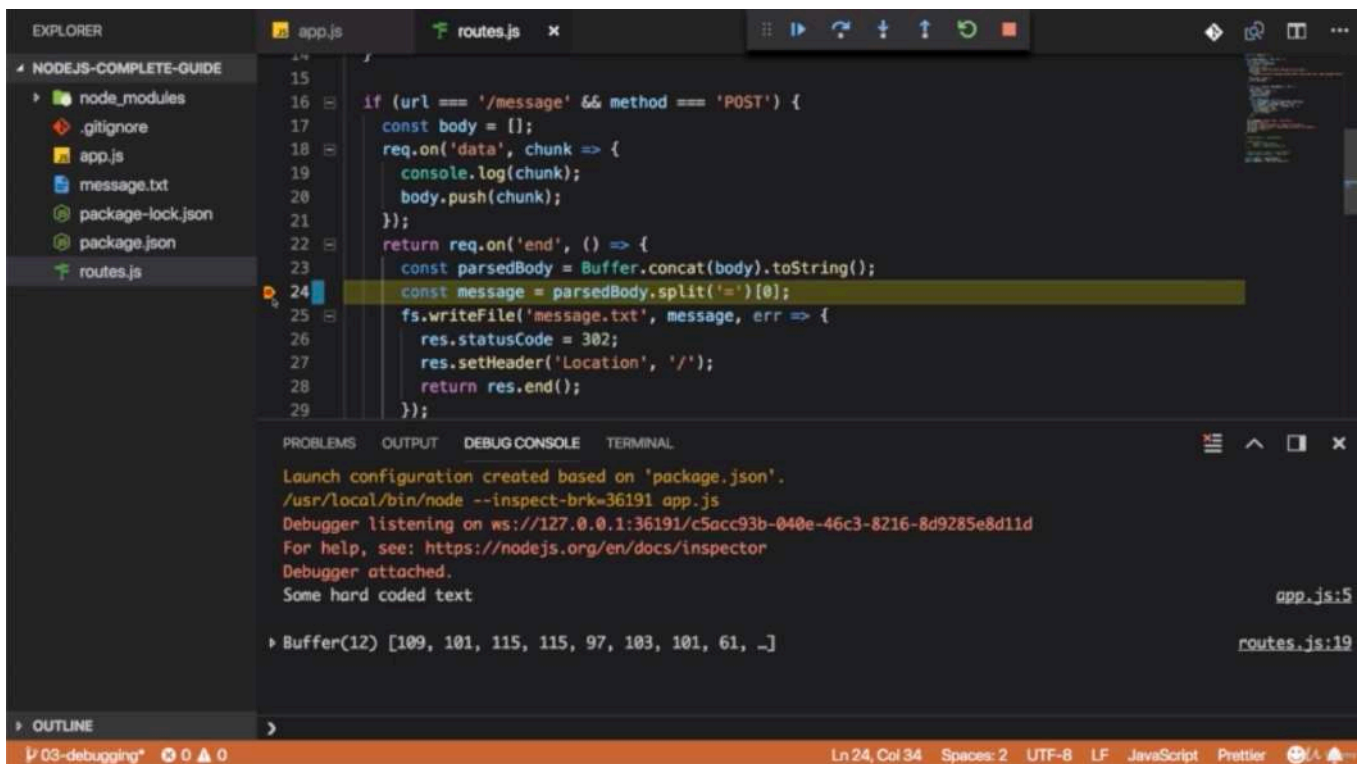
```
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node app.js`
Some hard coded text
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
Some hard coded text
<Buffer 6d 65 73 73 61 67 65 3d 54 65 73 74>
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
Some hard coded text
^C
Maximilians-MBP:nodejs-complete-guide mschwarzmueller$
```



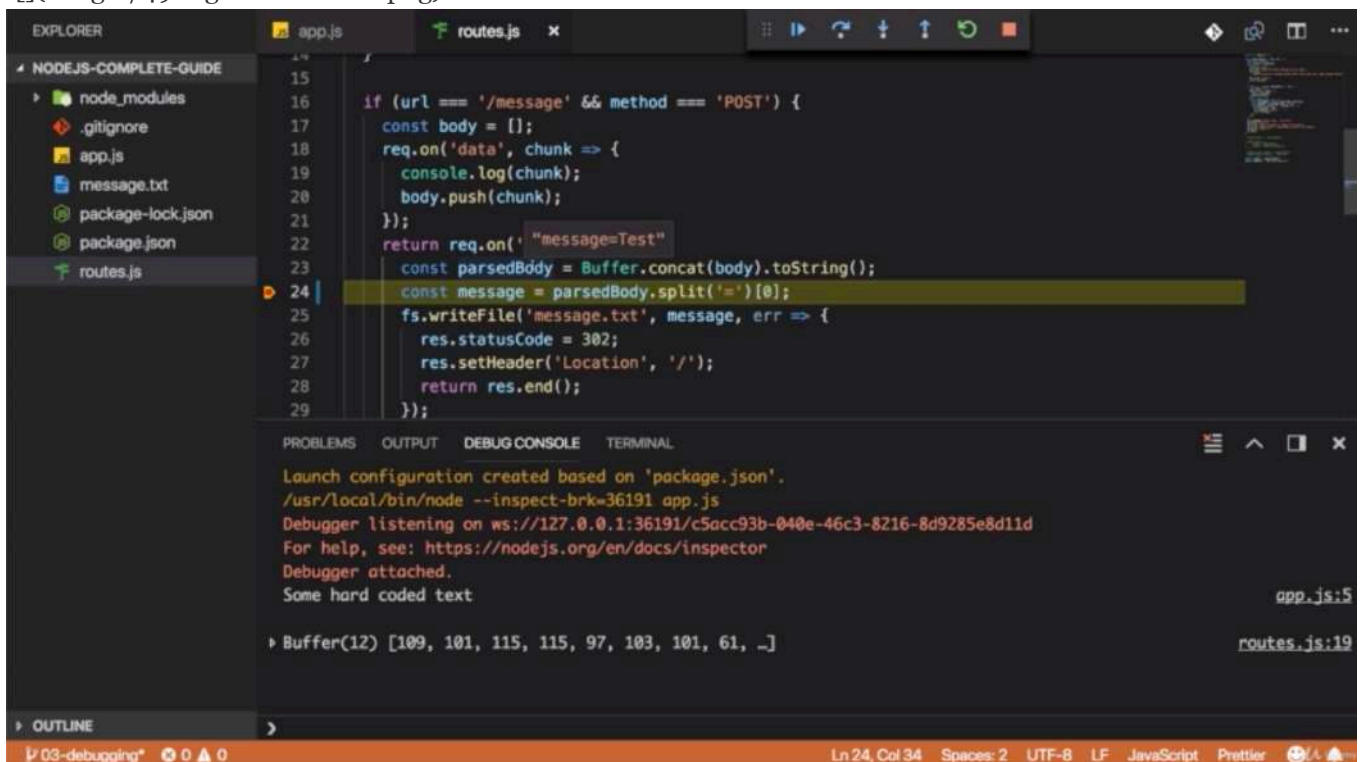
- this means that you can look into your code as it's running, to do that you need to set 'breakpoint'

- set the breakpoint by clicking red point.

Test



- if you not submit some text, it should automatically jump back and mark this line
- this line means the code execution now stopped here, so that you can look inside of it and actually you can now analyze your code in the moment it's running.



- for example, you can hover over your variable to see what's stored inside of them. so you see that the parsedBody is this string which we are splitting in the yellow highlighted line.


```
14
15
16 if (url === '/message' && method === 'POST') {
17   const body = [];
18   req.on('data', chunk => {
19     console.log(chunk);
20     body.push(chunk);
21   });
22   return req.on('end', () => {
23     const parsedBody = Buffer.concat(body).toString();
24     const message = parsedBody.split('=')[0];
25     fs.writeFile('message.txt', message, err => {
26       res.statusCode = 302;
27       res.setHeader('Location', '/');
28       return res.end();
29     });
30   });
31 }
```

Debugger Console:

```
Launch configuration created based on 'package.json'.
/usr/local/bin/node --inspect-brk=36191 app.js
Debugger listening on ws://127.0.0.1:36191/c5acc93b-040e-46c3-8216-8d9285e8d11d
For help, see: https://nodejs.org/en/docs/inspector
Debugger attached.
Some hard coded text
app.js:5
> Buffer(12) [109, 101, 115, 115, 97, 103, 101, 61, ...]
routes.js:19
```

```
14
15
16 if (url === '/message' && method === 'POST') {
17   const body = [];
18   req.on('data', chunk => {
19     console.log(chunk);
20     body.push(chunk);
21   });
22   return req.on('end', () => {
23     const parsedBody = Buffer.concat(b
24     const message = parsedBody.split('
25     fs.writeFile('message.txt', messag
26     res.statusCode = 302;
27     res.setHeader('Location', '/');
28     return res.end();
29   });
30 }
```

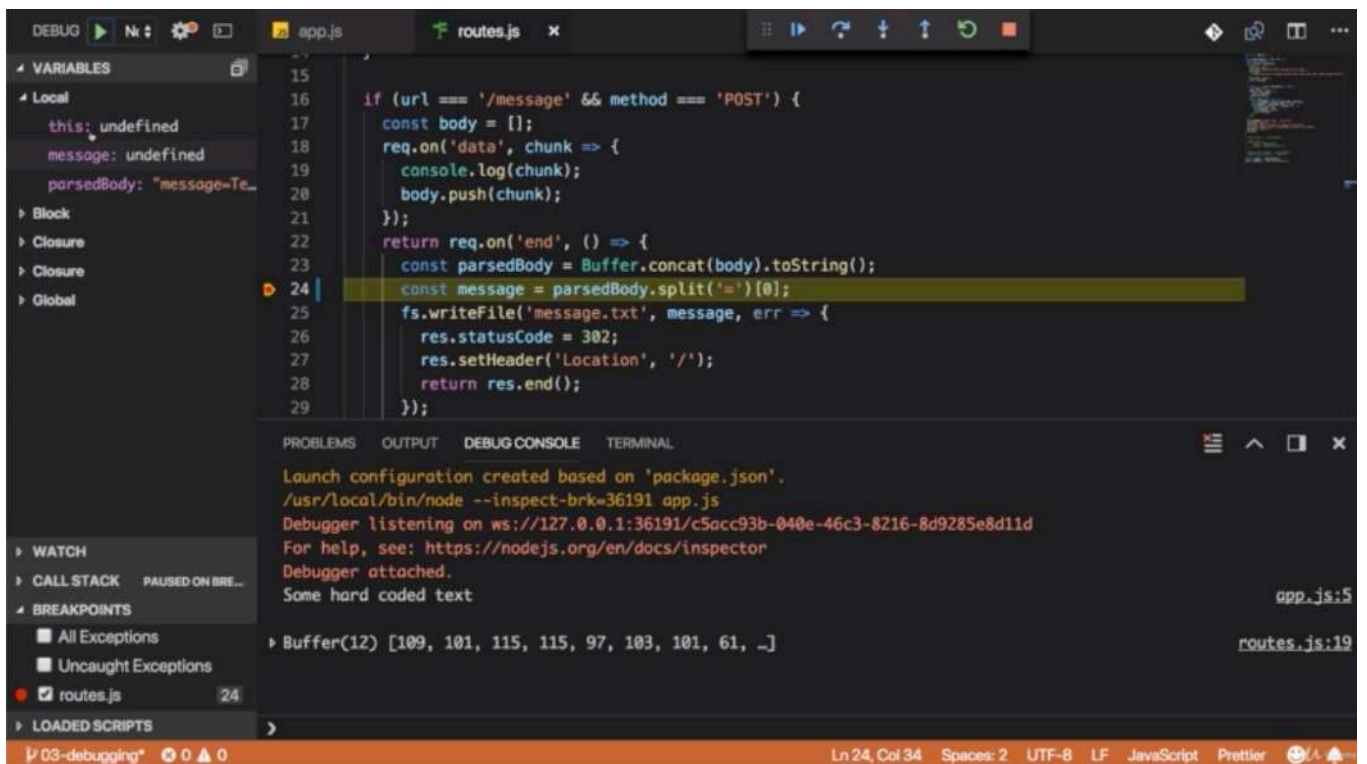
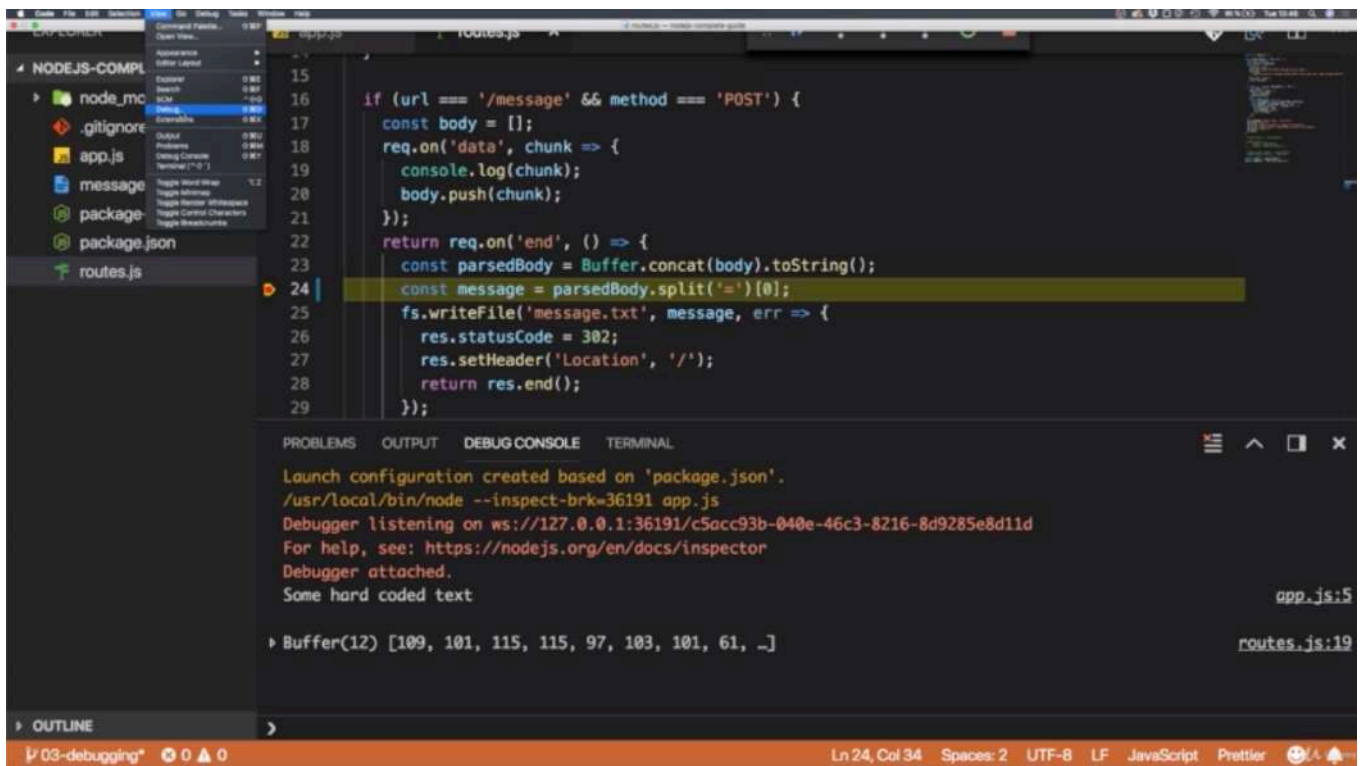
Debugger Console:

```
Launch configuration created based on 'package
/usr/local/bin/node --inspect-brk=36191 app.js
Debugger listening on ws://127.0.0.1:36191/c5ac
For help, see: https://nodejs.org/en/docs/inspe
Debugger attached.
Some hard coded text
app.js:5
> Buffer(12) [109, 101, 115, 115, 97, 103, 101, 61, ...]
routes.js:19
```

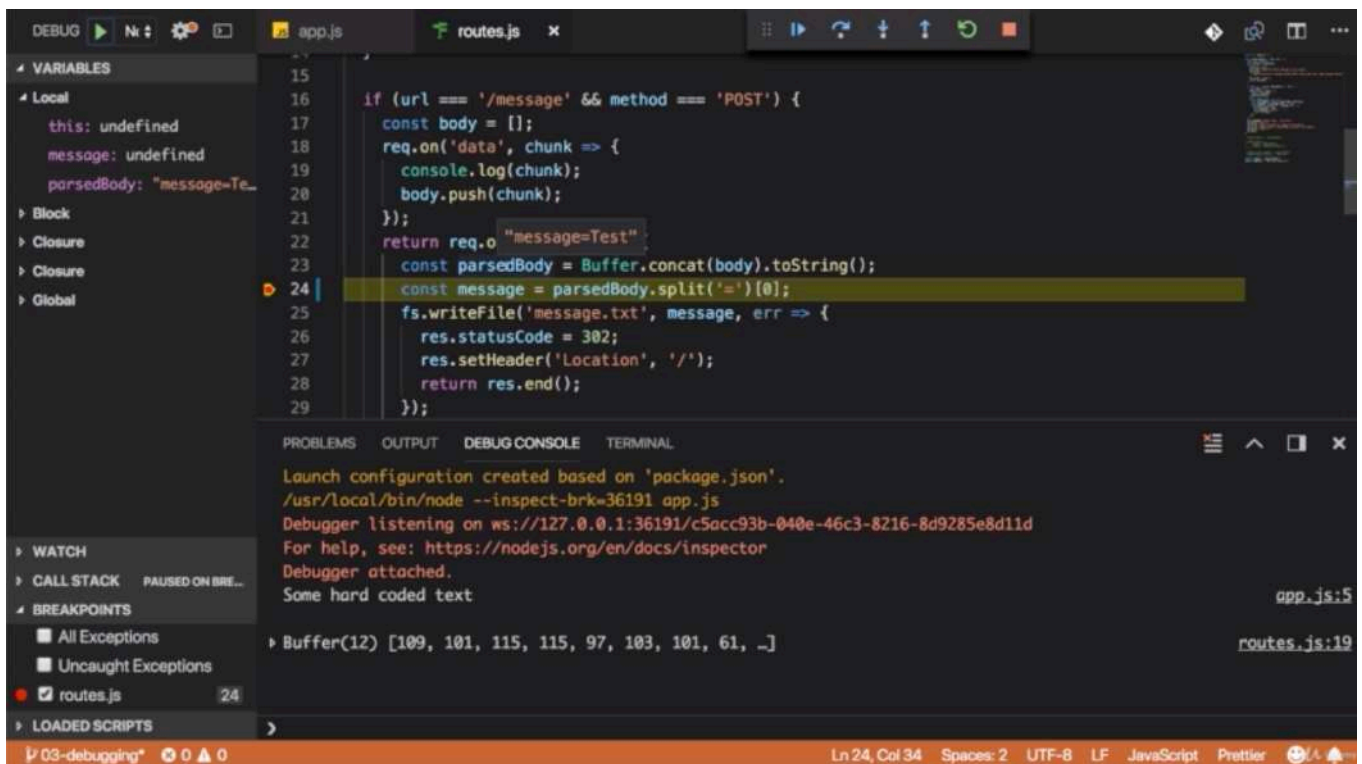
Debugger Console (Expanded):

```
Array(1) [Buffer(12)]
length: 1
__proto__: Array(0) [ , ...]
0: Buffer(12) [109, 101, 115, ...]
buffer: ArrayBuffer(633)
byteLength: 12
byteOffset: 621
length: 12
offset: 621
parent: ArrayBuffer(633)
Symbol(Symbol.toStringTag): undefined
__proto__: Uint8Array {readUIntLE: , readUInt
0: 109
1: 101
2: 115
3: 115
4: 97
5: 103
6: 101
```

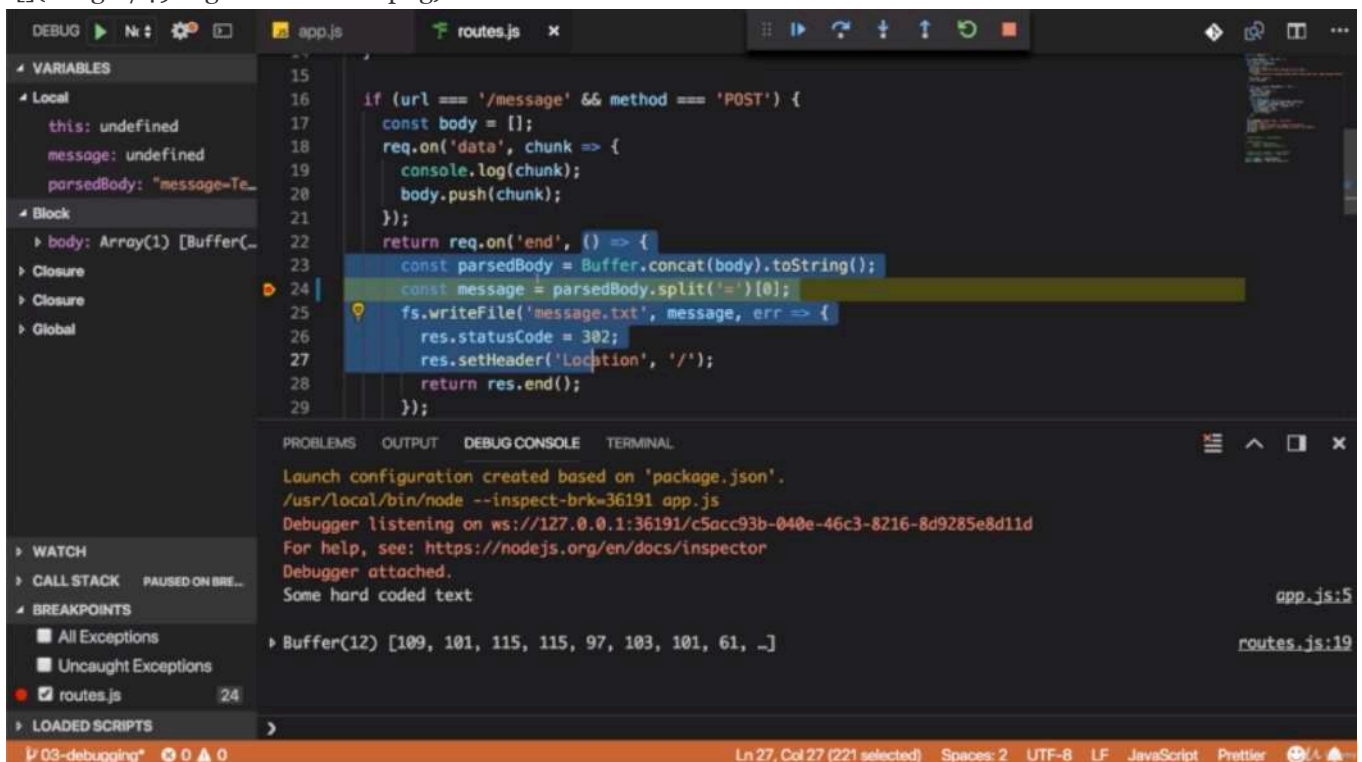
- you see what's inside of the body you passed to concat, that it's an array with a buffer of length 12.



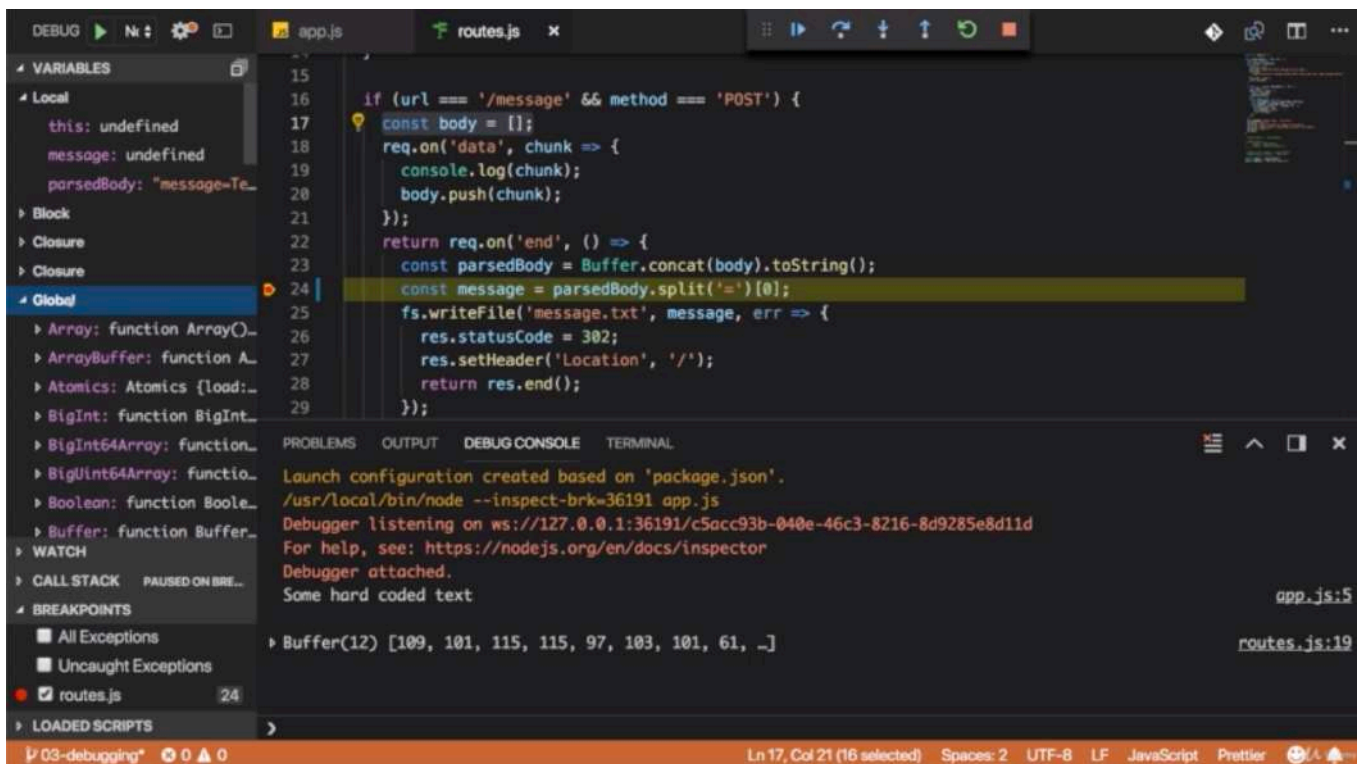
- you can see the key variables you have available in your code right now. message is undefined because we stopped in that line where we would set it but it stops before it executes the line.



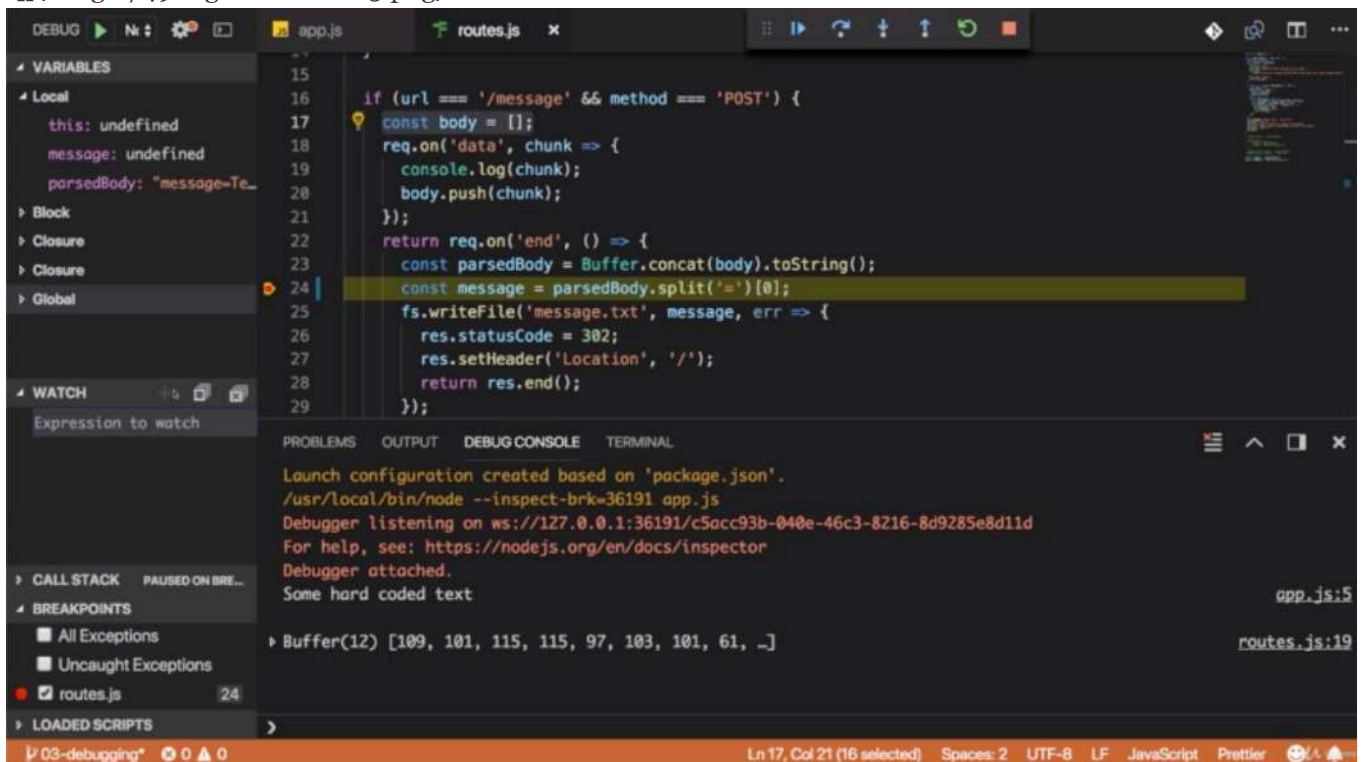
- but the parsedBody hold the value which you can also see if you hover over it



- you don't just see local variable which are available in this function but block variables too, this is the variable which is always available outside of this function.



- and you can also look into global values and the value they currently have stored



DEBUG app.js routes.js

VARIABLES

- Local
 - this: undefined
 - message: undefined
 - parsedBody: "message=Te...
- Block
- Closure
- Closure
- Global

WATCH

message

CALL STACK PAUSED ON BRE...

BREAKPOINTS

- ☐ All Exceptions
- ☐ Uncaught Exceptions
- ☒ routes.js 24

LOADED SCRIPTS

>

03-debugging* 0 0 0

```
15
16 if (url === '/message' && method === 'POST') {
17   const body = [];
18   req.on('data', chunk => {
19     console.log(chunk);
20     body.push(chunk);
21   });
22   return req.on('end', () => {
23     const parsedBody = Buffer.concat(body).toString();
24     const message = parsedBody.split('=')[0];
25     fs.writeFile('message.txt', message, err => {
26       res.statusCode = 302;
27       res.setHeader('Location', '/');
28       return res.end();
29     });
30   });
31 }
```

PROBLEMS **OUTPUT** **DEBUG CONSOLE** **TERMINAL**

Launch configuration created based on 'package.json'.
/usr/local/bin/node --inspect-brk=36191 app.js
Debugger listening on ws://127.0.0.1:36191/c5acc93b-040e-46c3-8216-8d9285e8d11d
For help, see: <https://nodejs.org/en/docs/inspector>
Debugger attached.
Some hard coded text

app.js:5

Buffer(12) [109, 101, 115, 115, 97, 103, 101, 61, ...]

routes.js:19

Ln 17, Col 21 (16 selected) Spaces: 2 UTF-8 LF JavaScript Prettier

DEBUG app.js routes.js

VARIABLES

- Local
 - this: undefined
 - message: undefined
 - parsedBody: "message=Te...
- Block
- Closure
- Closure
- Global

WATCH

message: undefined

CALL STACK PAUSED ON BRE...

BREAKPOINTS

- ☐ All Exceptions
- ☐ Uncaught Exceptions
- ☒ routes.js 24

LOADED SCRIPTS

>

03-debugging* 0 0 0

```
15
16 if (url === '/message' && method === 'POST') {
17   const body = [];
18   req.on('data', chunk => {
19     console.log(chunk);
20     body.push(chunk);
21   });
22   return req.on('end', () => {
23     const parsedBody = Buffer.concat(body).toString();
24     const message = parsedBody.split('=')[0];
25     fs.writeFile('message.txt', message, err => {
26       res.statusCode = 302;
27       res.setHeader('Location', '/');
28       return res.end();
29     });
30   });
31 }
```

PROBLEMS **OUTPUT** **DEBUG CONSOLE** **TERMINAL**

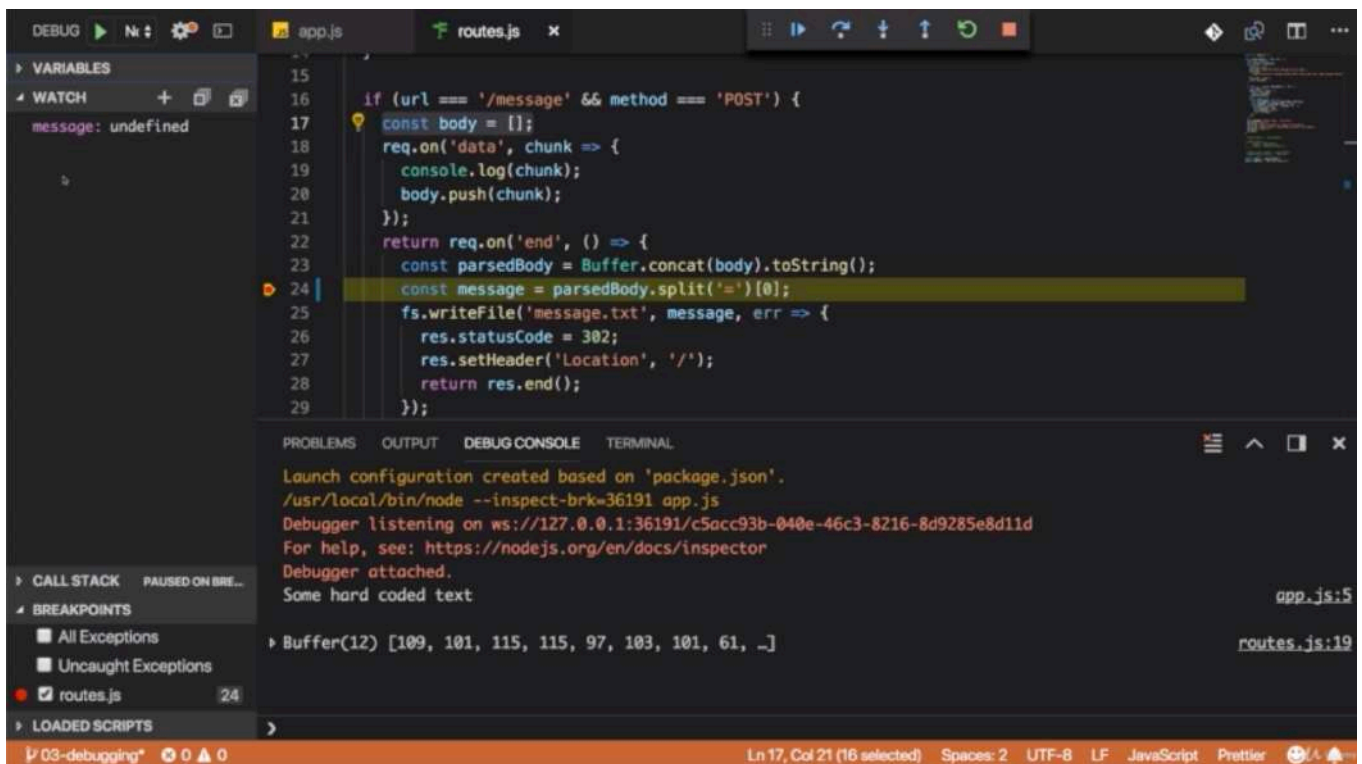
Launch configuration created based on 'package.json'.
/usr/local/bin/node --inspect-brk=36191 app.js
Debugger listening on ws://127.0.0.1:36191/c5acc93b-040e-46c3-8216-8d9285e8d11d
For help, see: <https://nodejs.org/en/docs/inspector>
Debugger attached.
Some hard coded text

app.js:5

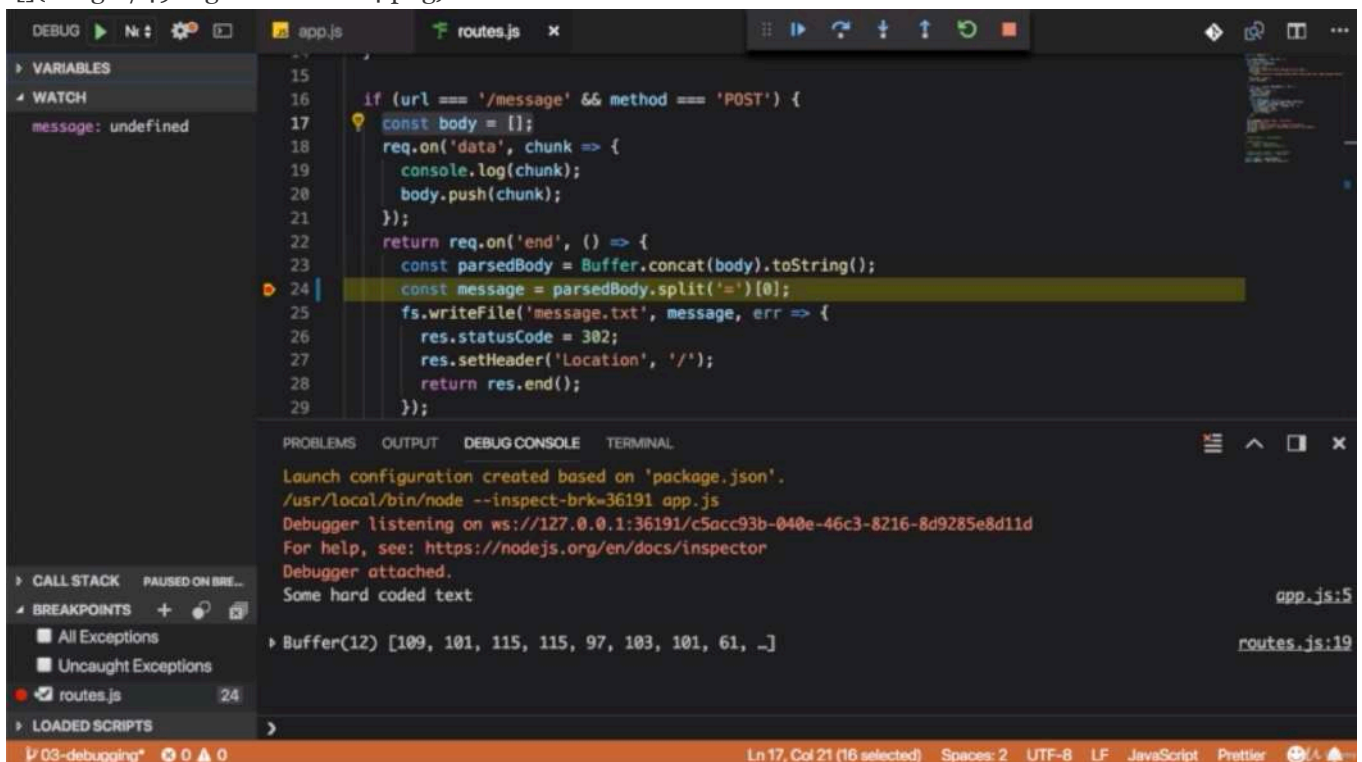
Buffer(12) [109, 101, 115, 115, 97, 103, 101, 61, ...]

routes.js:19

Ln 17, Col 21 (16 selected) Spaces: 2 UTF-8 LF JavaScript Prettier

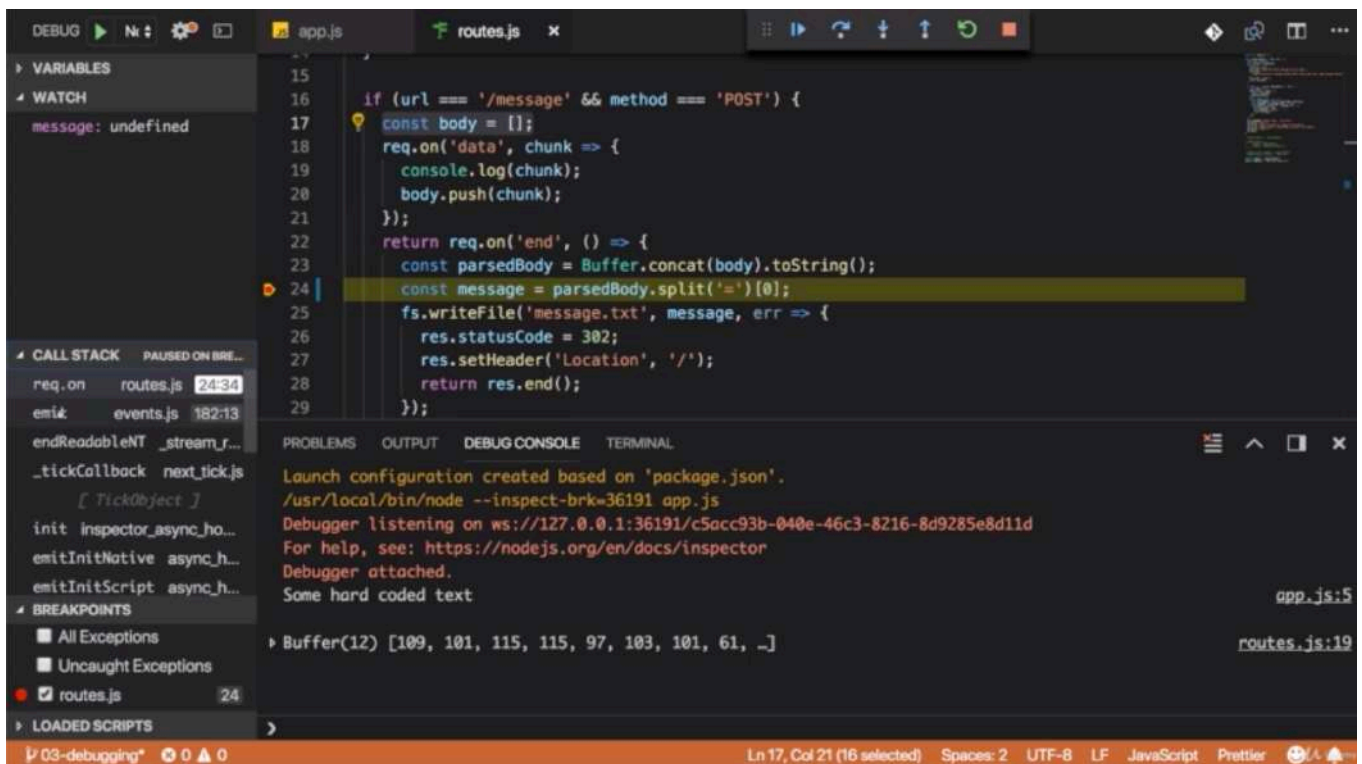


- and you can define watchers here, you can click pluss here, and then watch message and hit enter

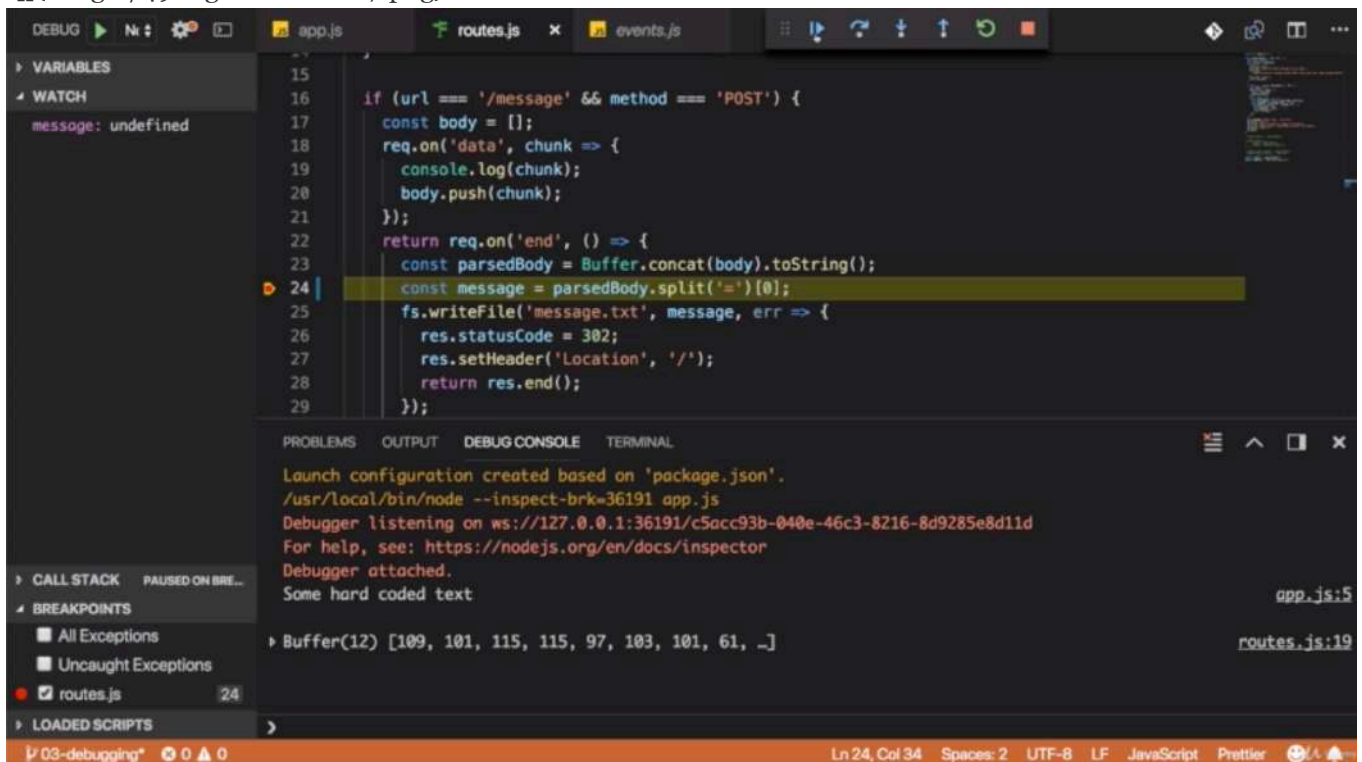


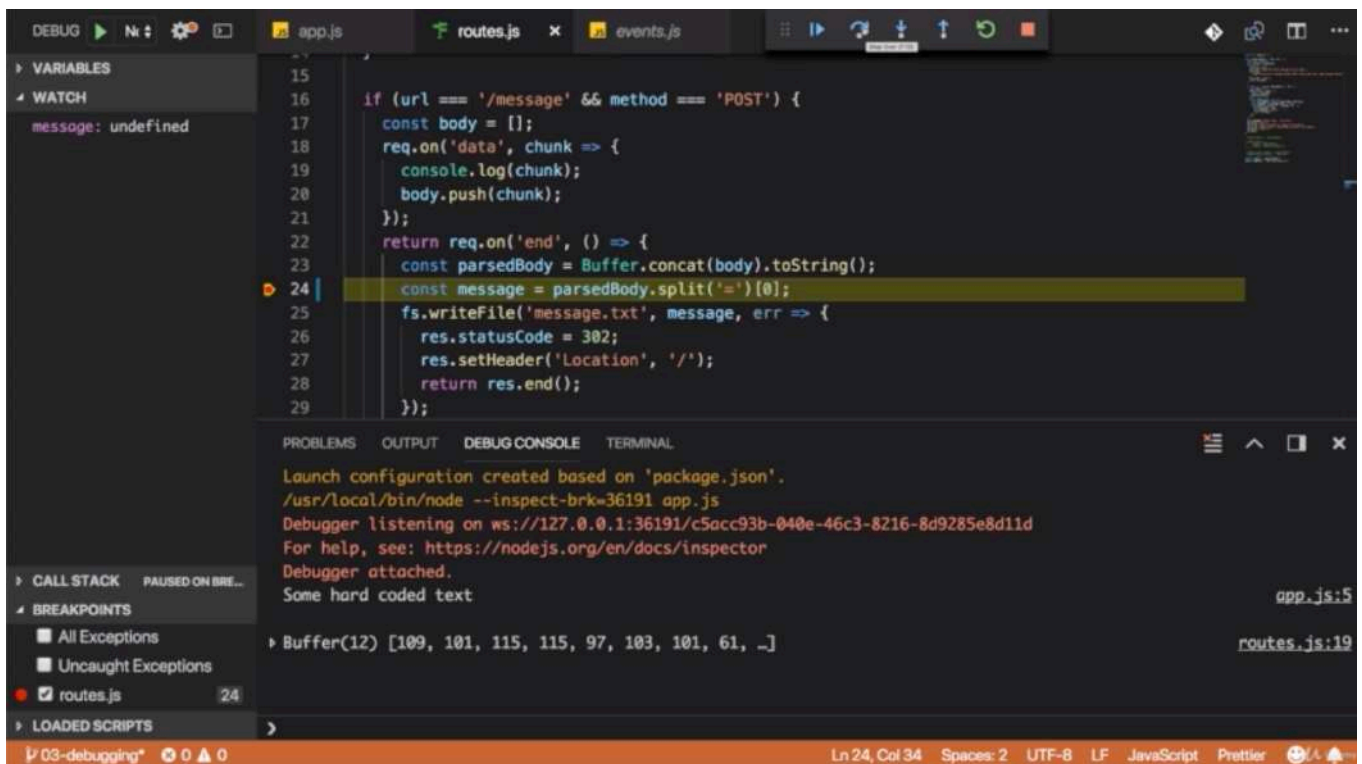
- you see all the breakpoints you set in below of the left side

- if you uncheck them to not stop execution the next time they are reached and so on.

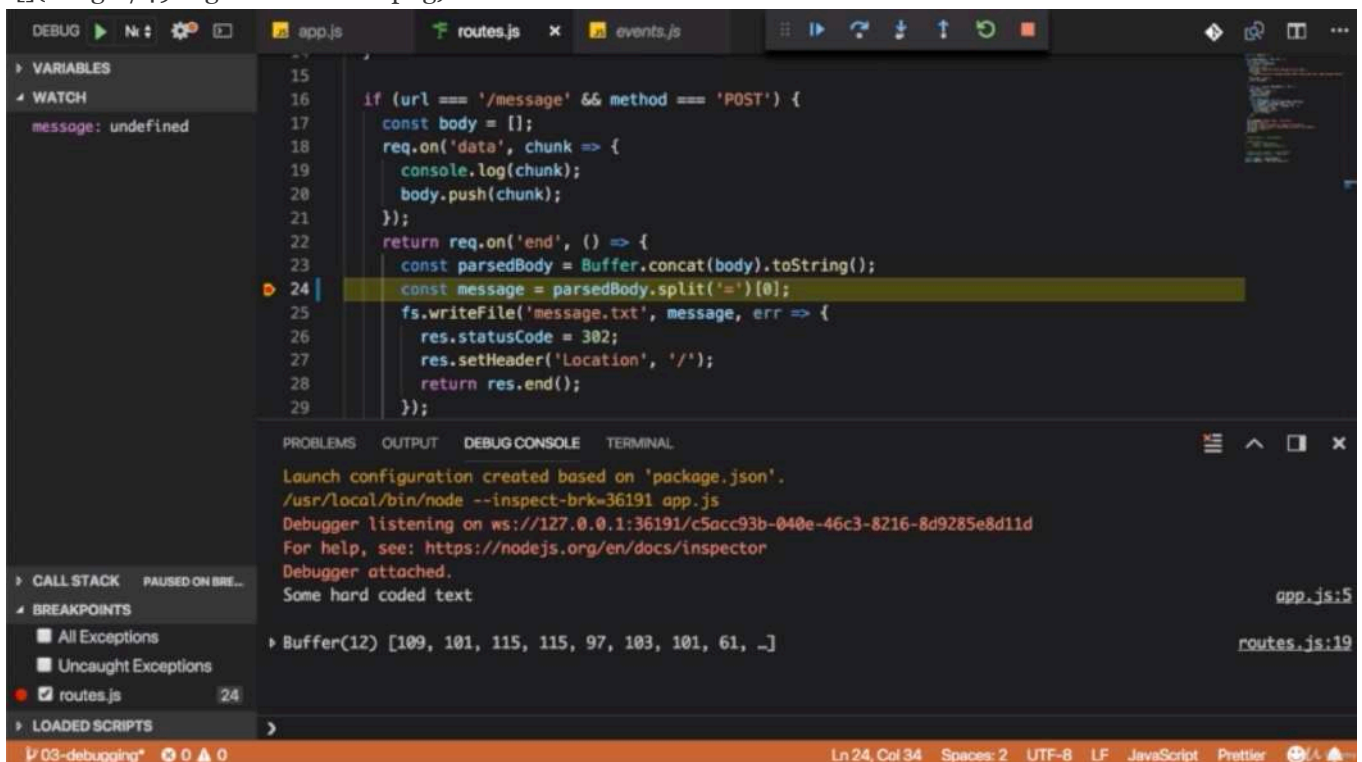


- you also see the call stack which looks very cryptic but what in the end just shows how the process went through your code and you can click on these different parts to see where actually this code which belongs to the code that was executed can be found and not all of that is code you wrote, a lot of that is core node.js code.

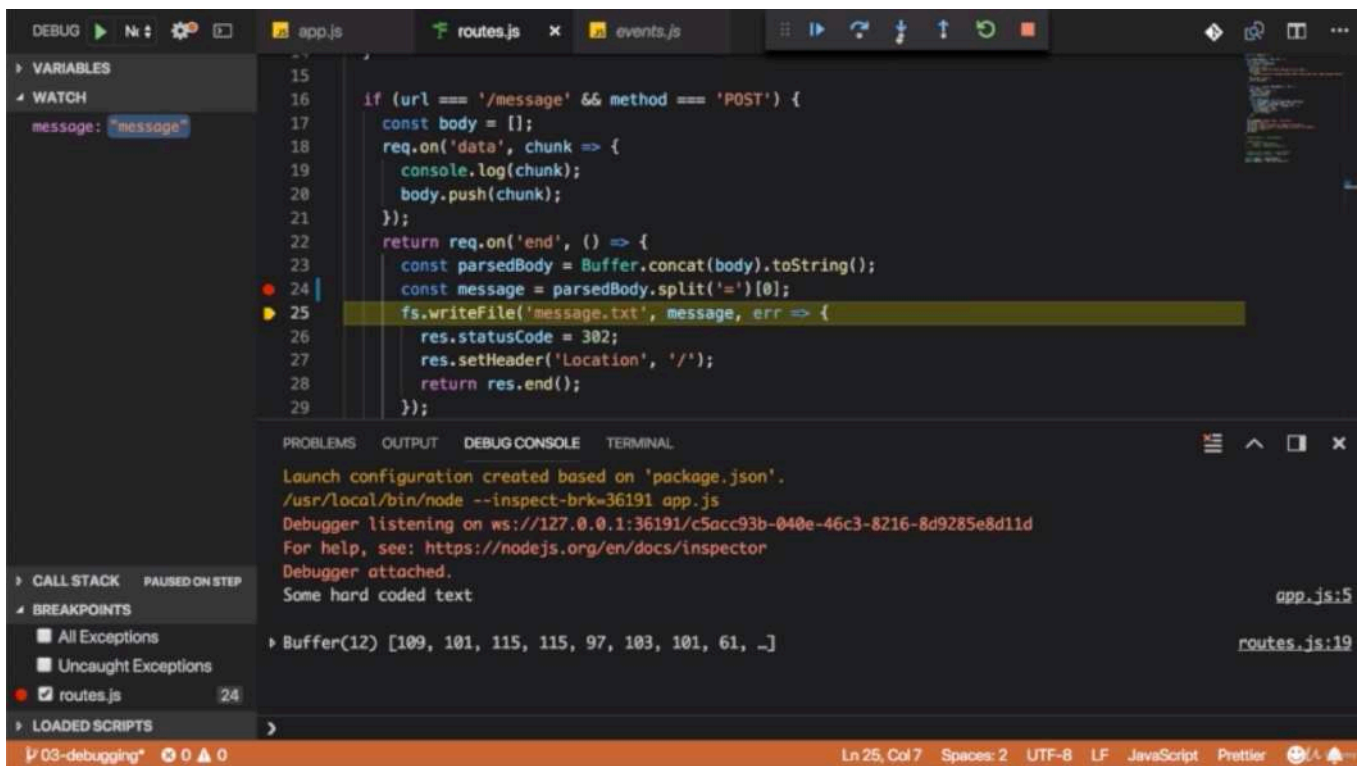




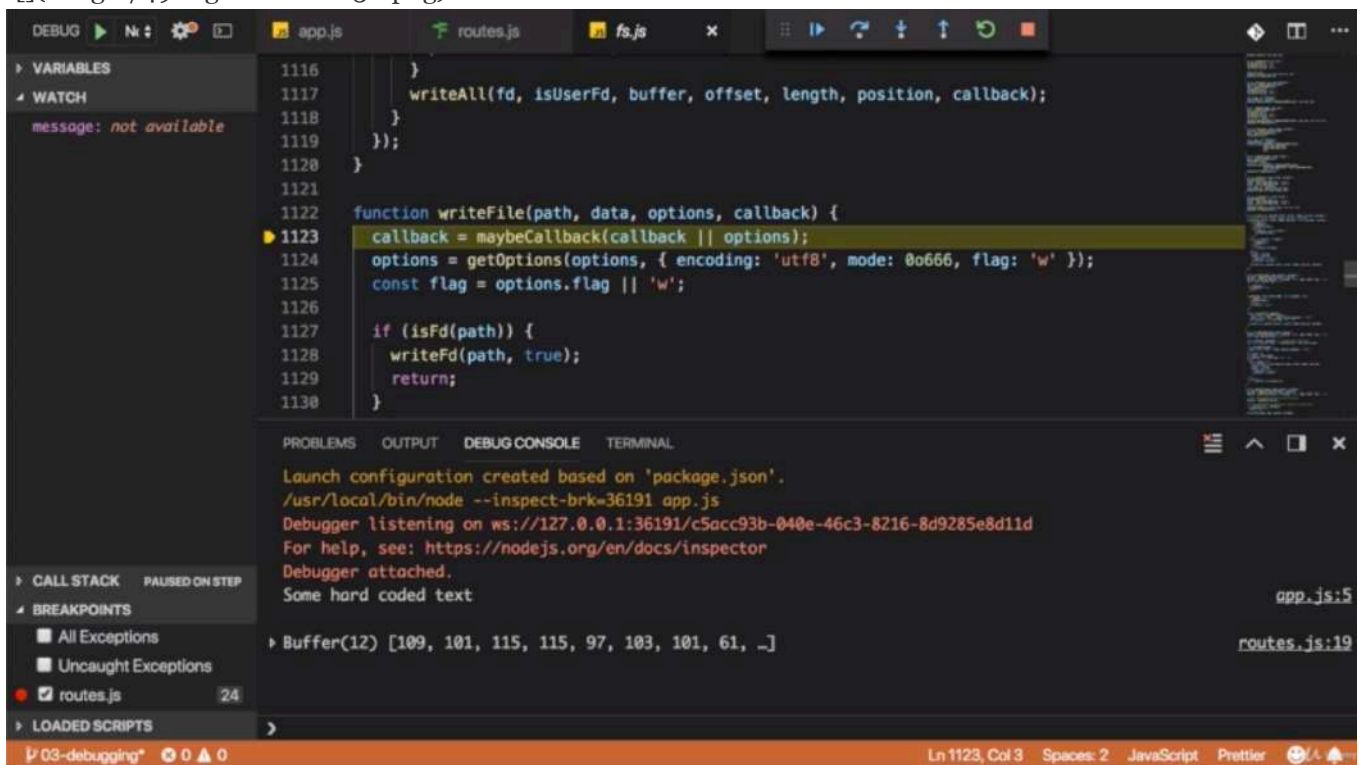
- now to work it and to continue with your code execution, you can resume code execution with the play button but we don't wanna do that instead we wanna step through our code step by step so that we can see when it fails or where it goes wrong.



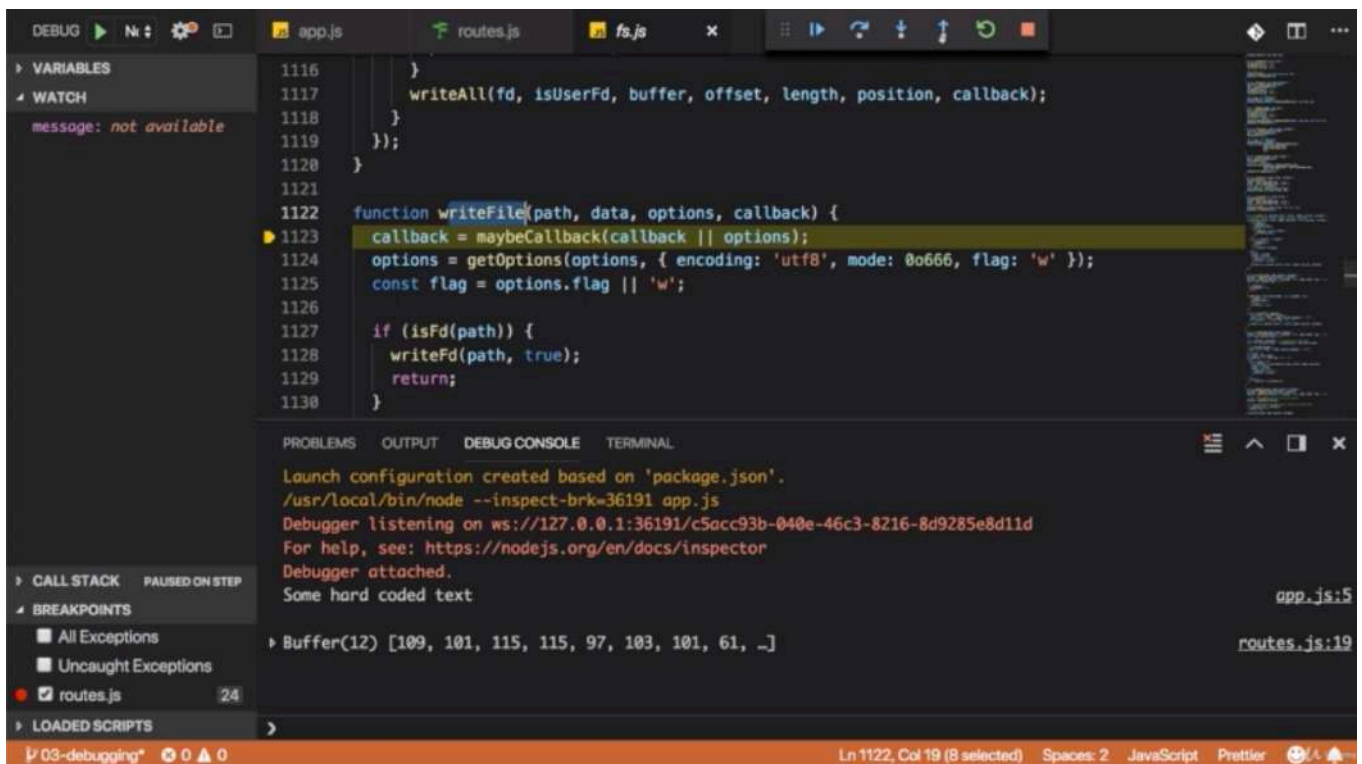
- you can do that with this button. which steps to the next line



- or with this button, which doesn't just step into the next line but actually even goes into functions like this one



- so if you click here and again, now all of sudden, you are in the `writeFile` function defined by node.js



- you can step back by this key.

* Chapter 50: Using The Debugger



- let's send another message here, hit send, and it breaks again because i didn't remove the breakpoint. and let's fix that error we have.

DEBUG app.js routes.js

VARIABLES

WATCH

message: "message"

```
18 req.on('data', chunk => {
19   console.log(chunk);
20   body.push(chunk);
21 });
22 return req.on('end', () => {
23   const parsedBody = Buffer.concat(body).toString();
24   const message = parsedBody.split('=')[0];
25   fs.writeFile('message.txt', message, err => {
26     res.statusCode = 302;
27     res.setHeader('Location', '/');
28     return res.end();
29   });
30 });
31 }
32 res.setHeader('Content-Type', 'text/html');
33 res.write('<html>');
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Debugger: Listening on WS://127.0.0.1:30091/300930-6906-70C3-82A0-0026C600110

For help, see: <https://nodejs.org/en/docs/inspector>

Debugger attached.

Some hard coded text app.js:5

CALL STACK PAUSED ON STEP

BREAKPOINTS

☐ All Exceptions

☐ Uncaught Exceptions

☒ routes.js 24

LOADED SCRIPTS

Ln 25, Col 7 Spaces: 2 UTF-8 LF JavaScript Prettier

DEBUG app.js routes.js

VARIABLES

WATCH

message: "message"

```
18 req.on('data', chunk => {
19   console.log(chunk);
20   body.push(chunk);
21 });
22 return req.on('end', () => {
23   const parsedBody = Buffer.concat(body).toString();
24   const message = parsedBody.split('=')[0];
25   fs.writeFile('message.txt', message, err => {
26     res.statusCode = 302;
27     res.setHeader('Location', '/');
28     return res.end();
29   });
30 });
31 }
32 res.setHeader('Content-Type', 'text/html');
33 res.write('<html>');
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Debugger: Listening on WS://127.0.0.1:30091/300930-6906-70C3-82A0-0026C600110

For help, see: <https://nodejs.org/en/docs/inspector>

Debugger attached.

Some hard coded text app.js:5

CALL STACK PAUSED ON STEP

BREAKPOINTS

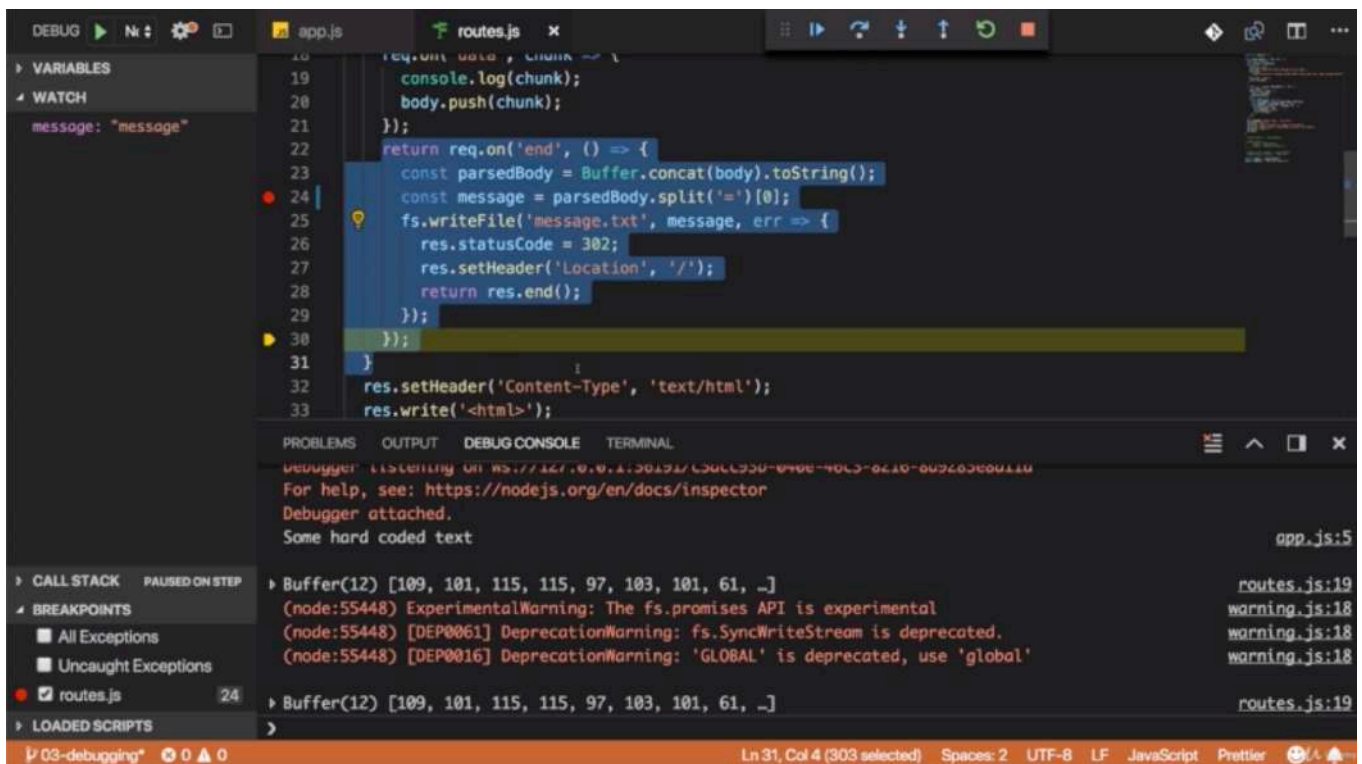
☐ All Exceptions

☐ Uncaught Exceptions

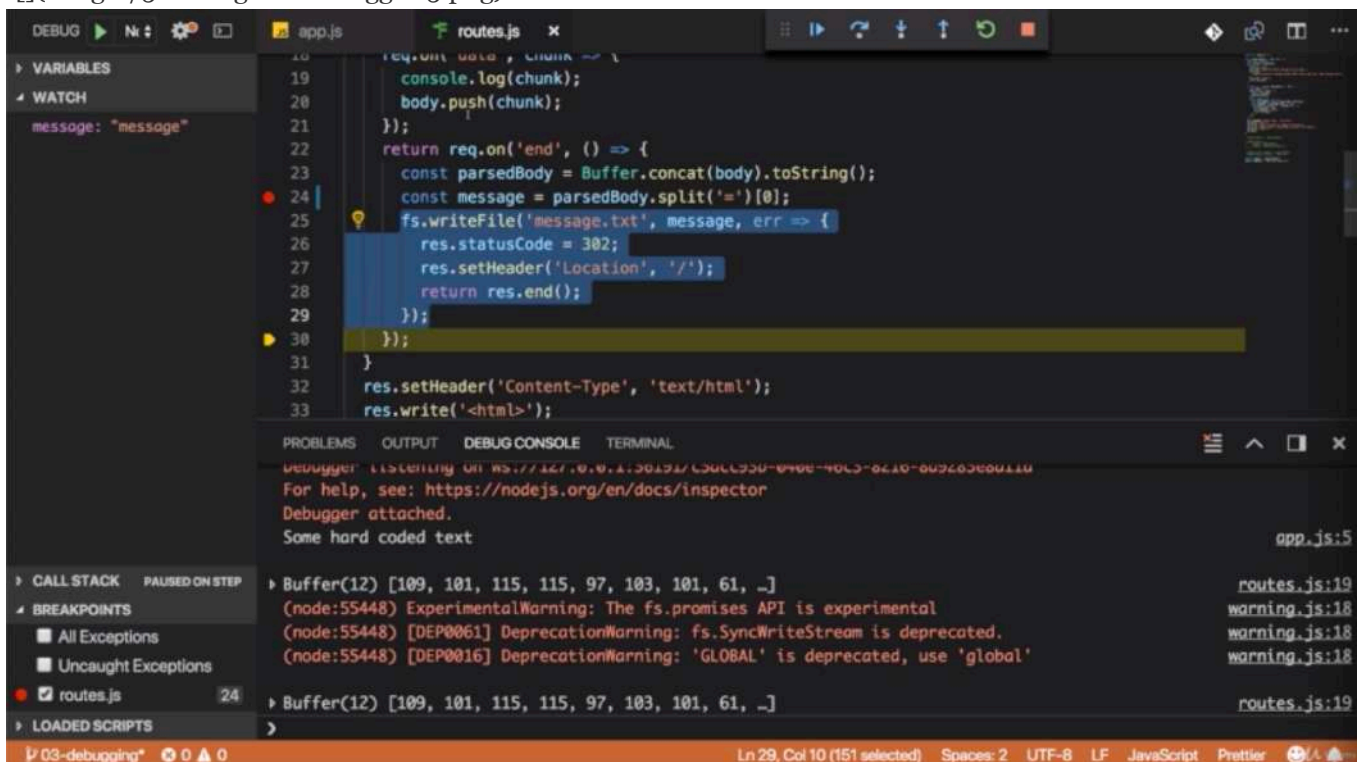
☒ routes.js 24

LOADED SCRIPTS

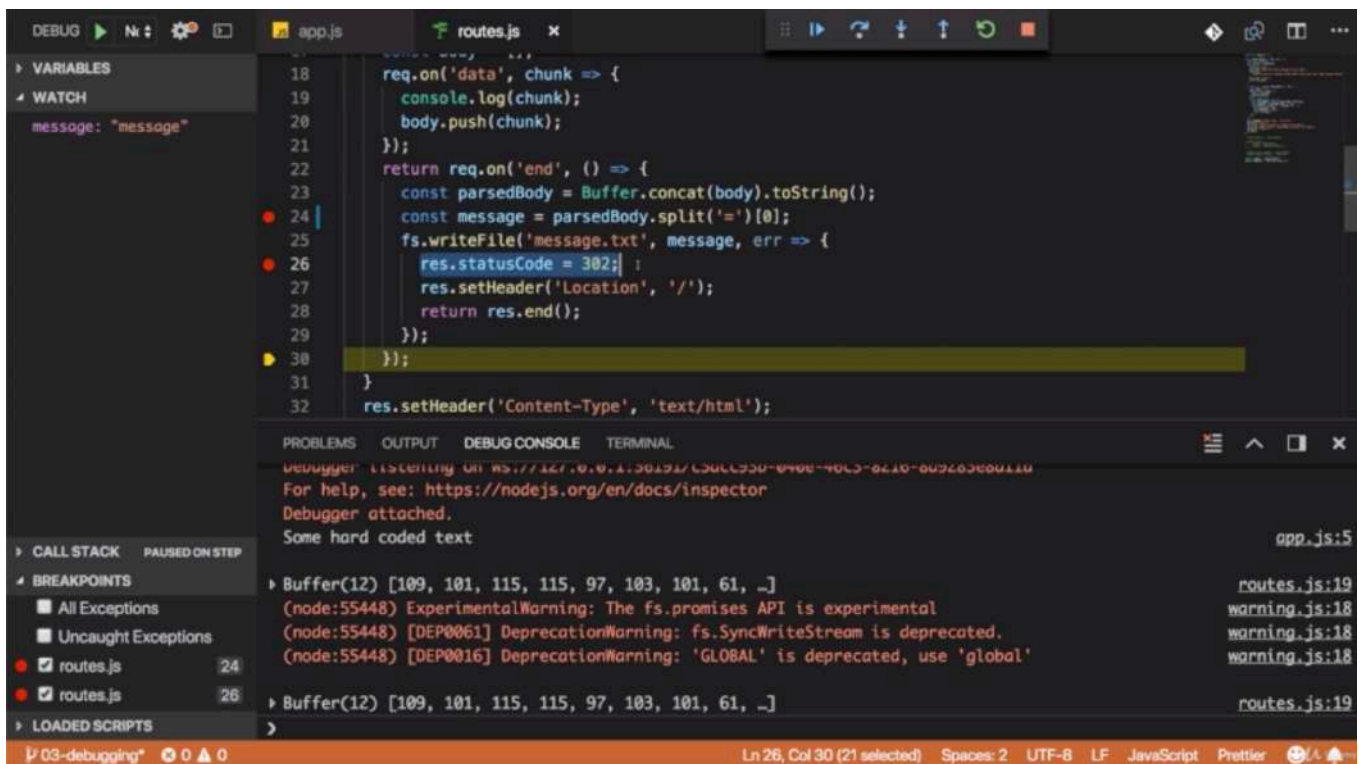
Ln 29, Col 7 (111 selected) Spaces: 2 UTF-8 LF JavaScript Prettier

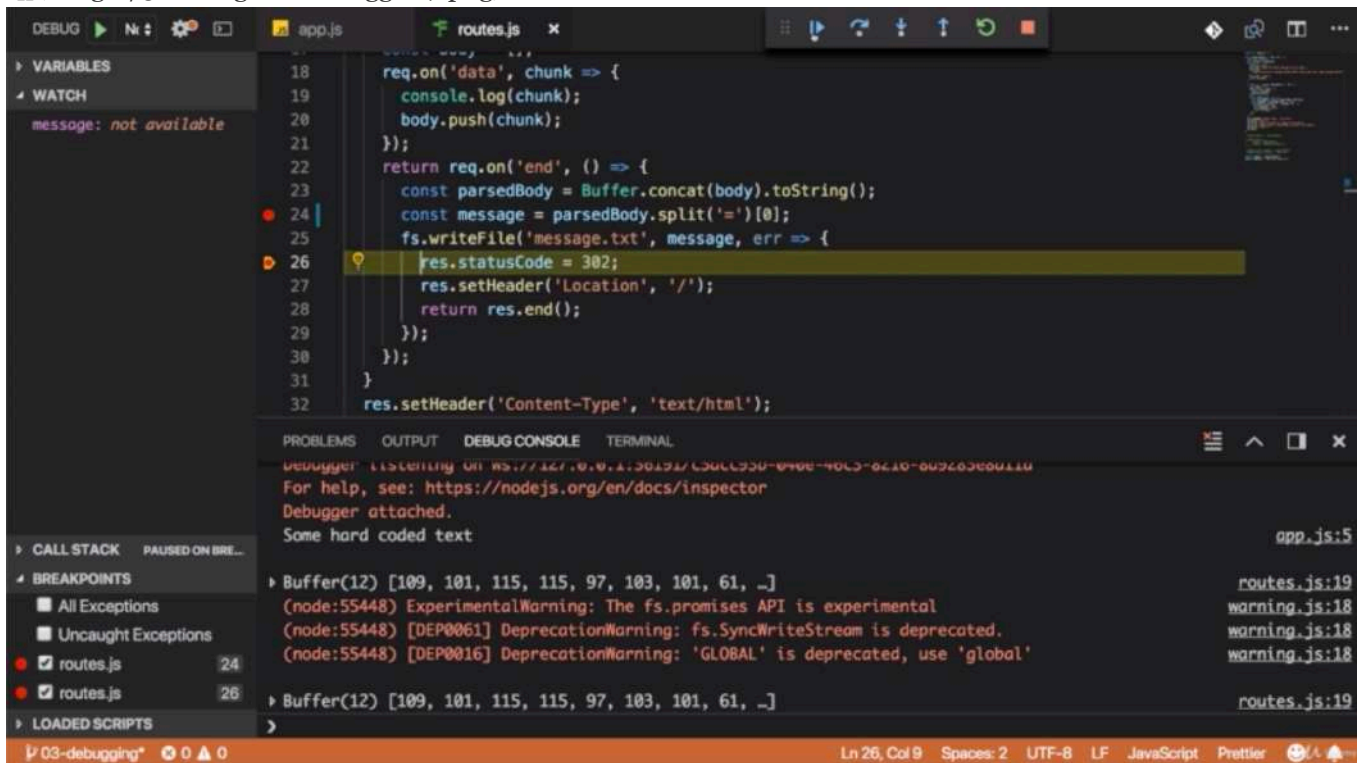


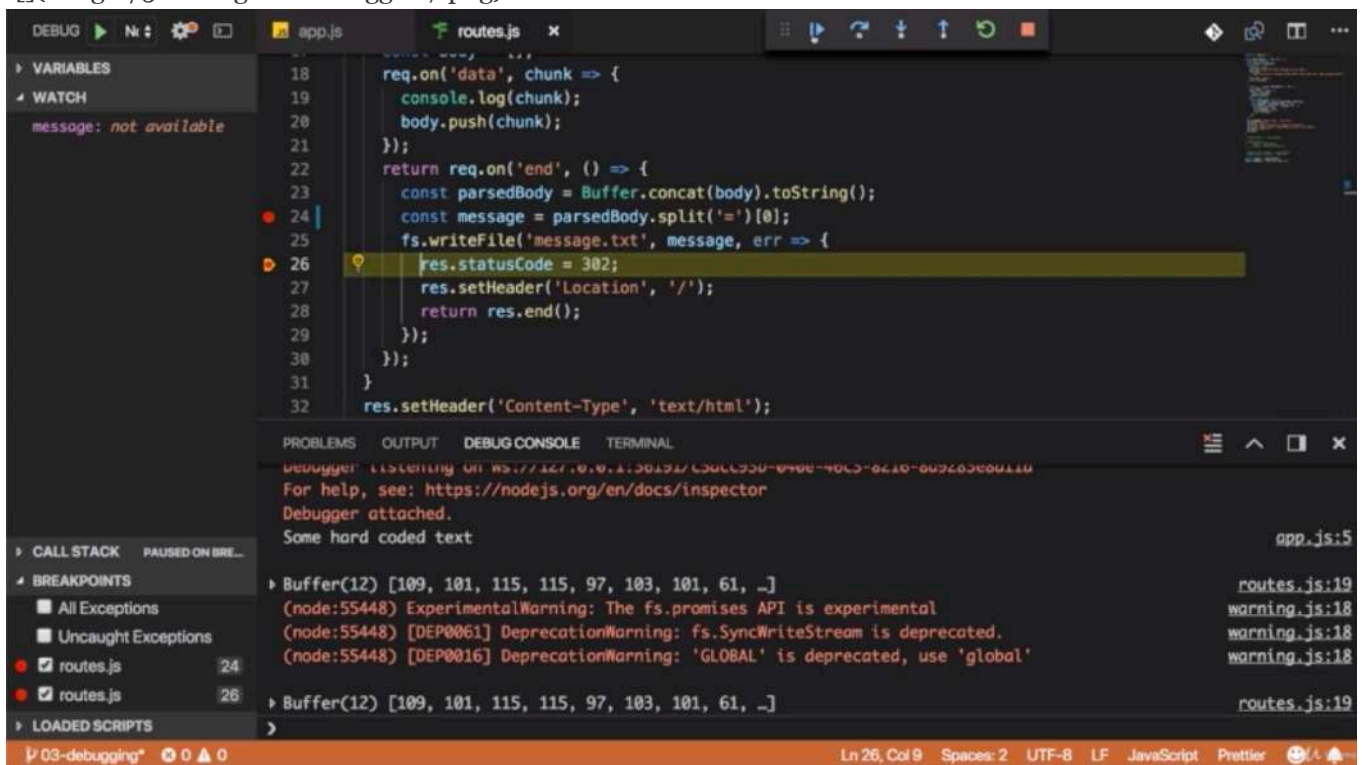
- now it registers this code and here you also see that it doesn't immediately execute this function because if i click that button again, we jump to the end of this function here and that makes a lot of sense because as i mentioned, this function is just registered by node.js to be executed in the future once the file is read.



- the problem we wanna solve is not inside of there though.



- but if you wanna get notified once a part in here is reached, you can simply add a breakpoint there,




resume execution and it will break once it reaches that, so once that callback is triggered because the right file operation is done.

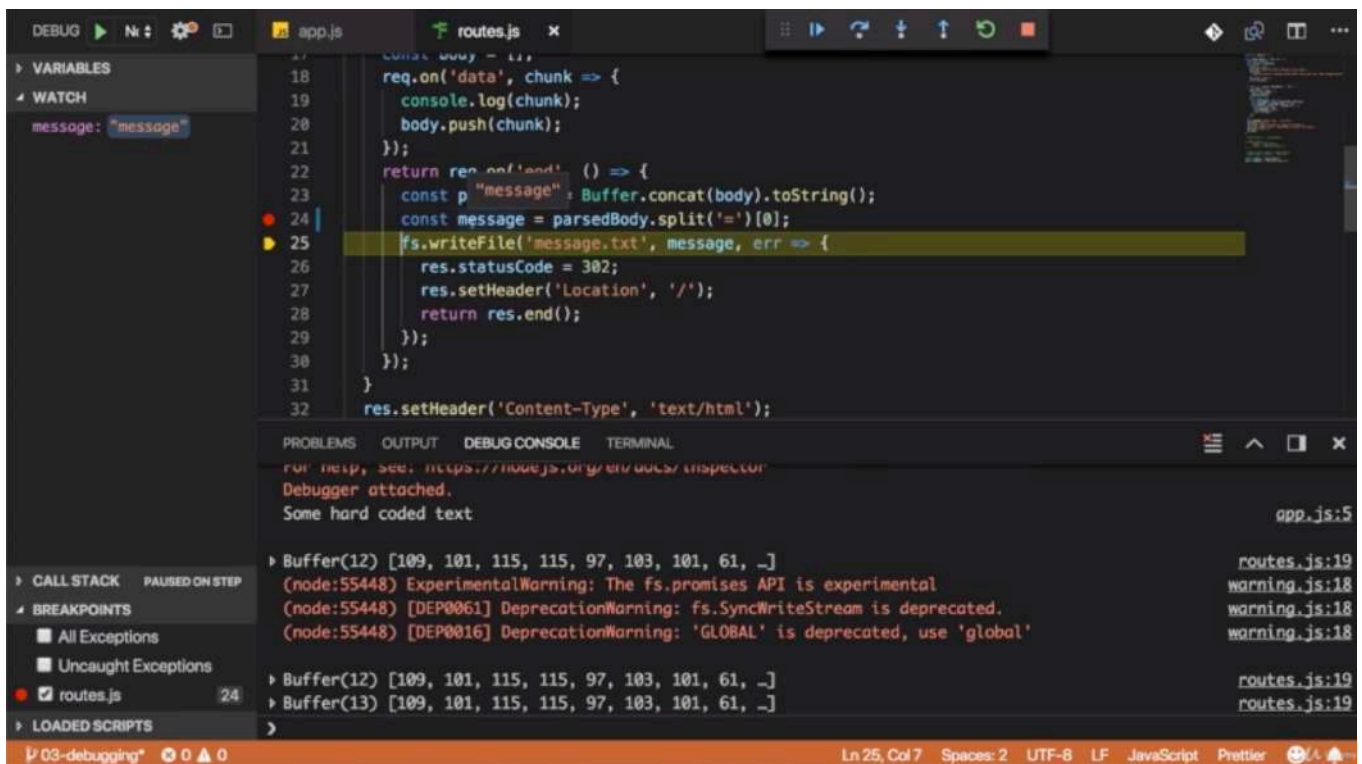
- our problem is the message. message is not available here anymore. so this analysis is not useful to us.



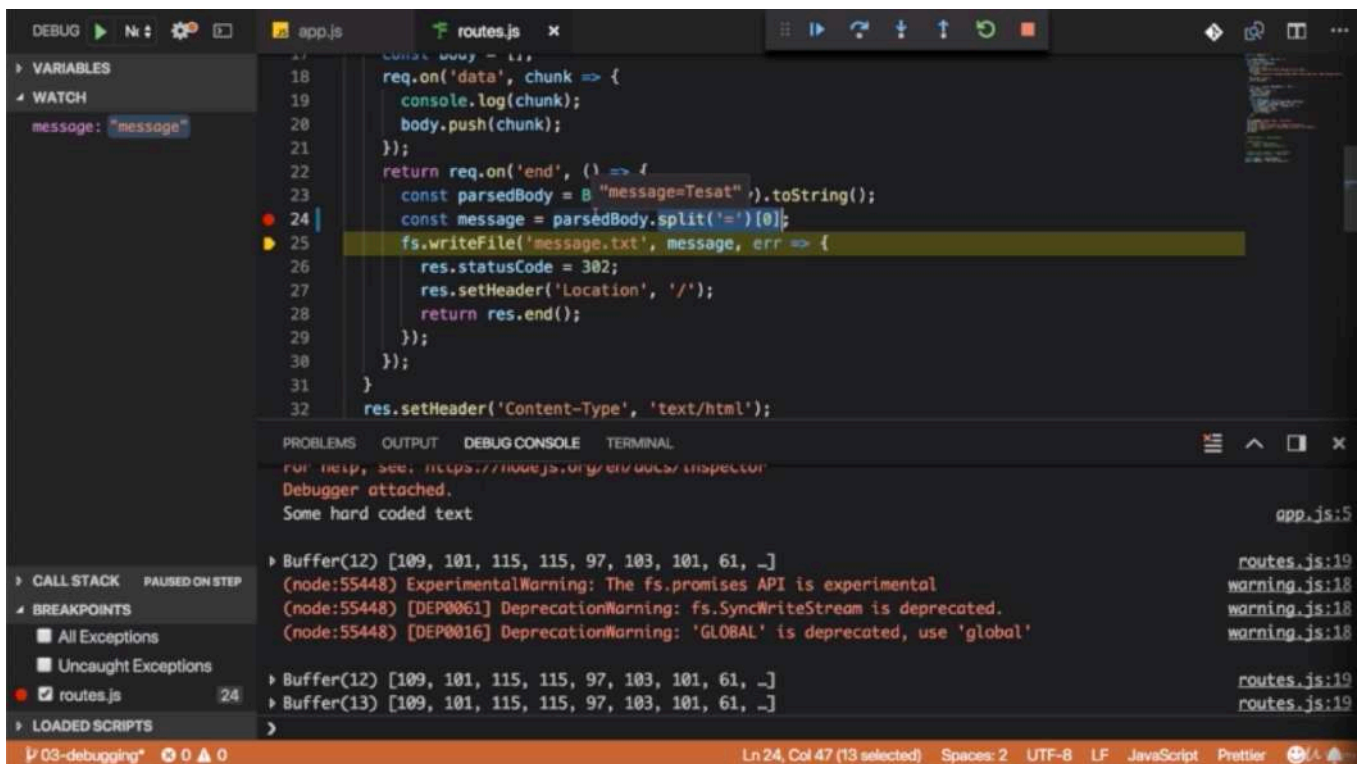




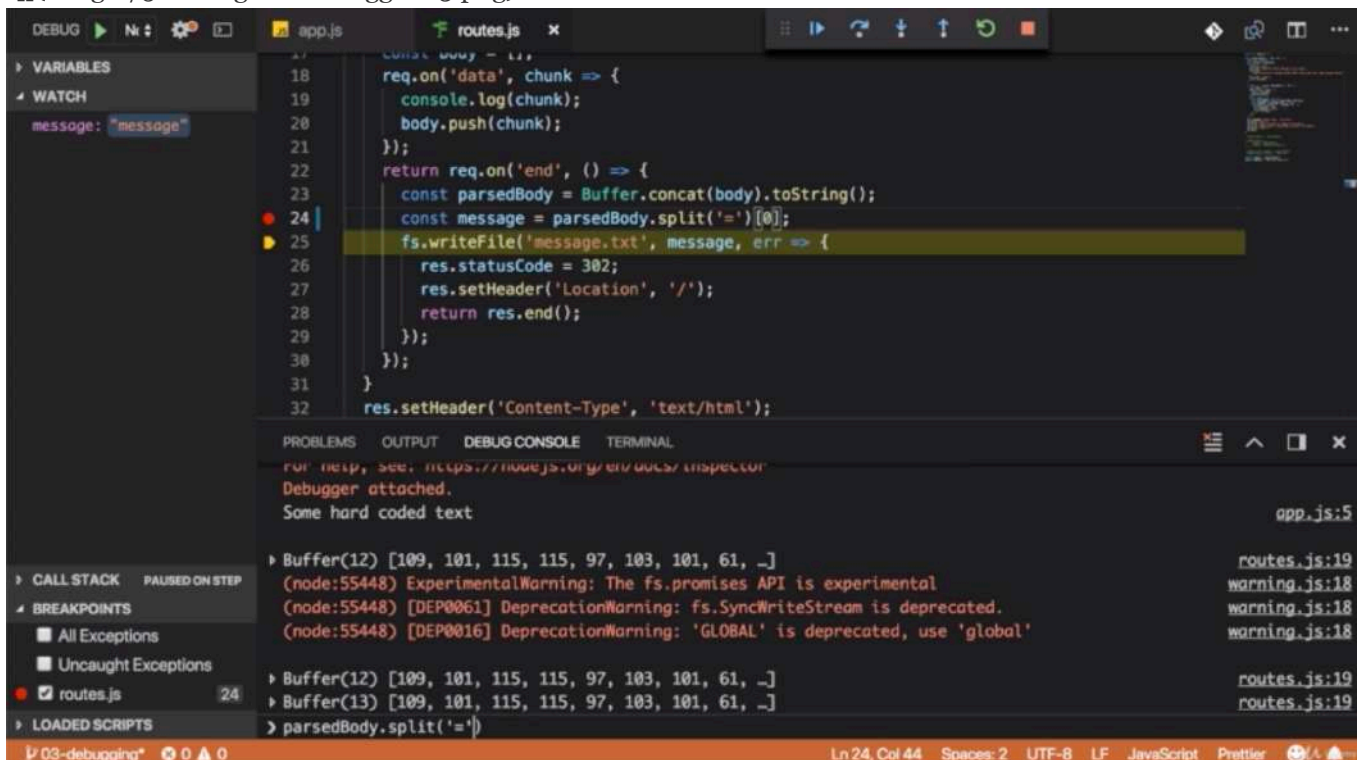
Test Send

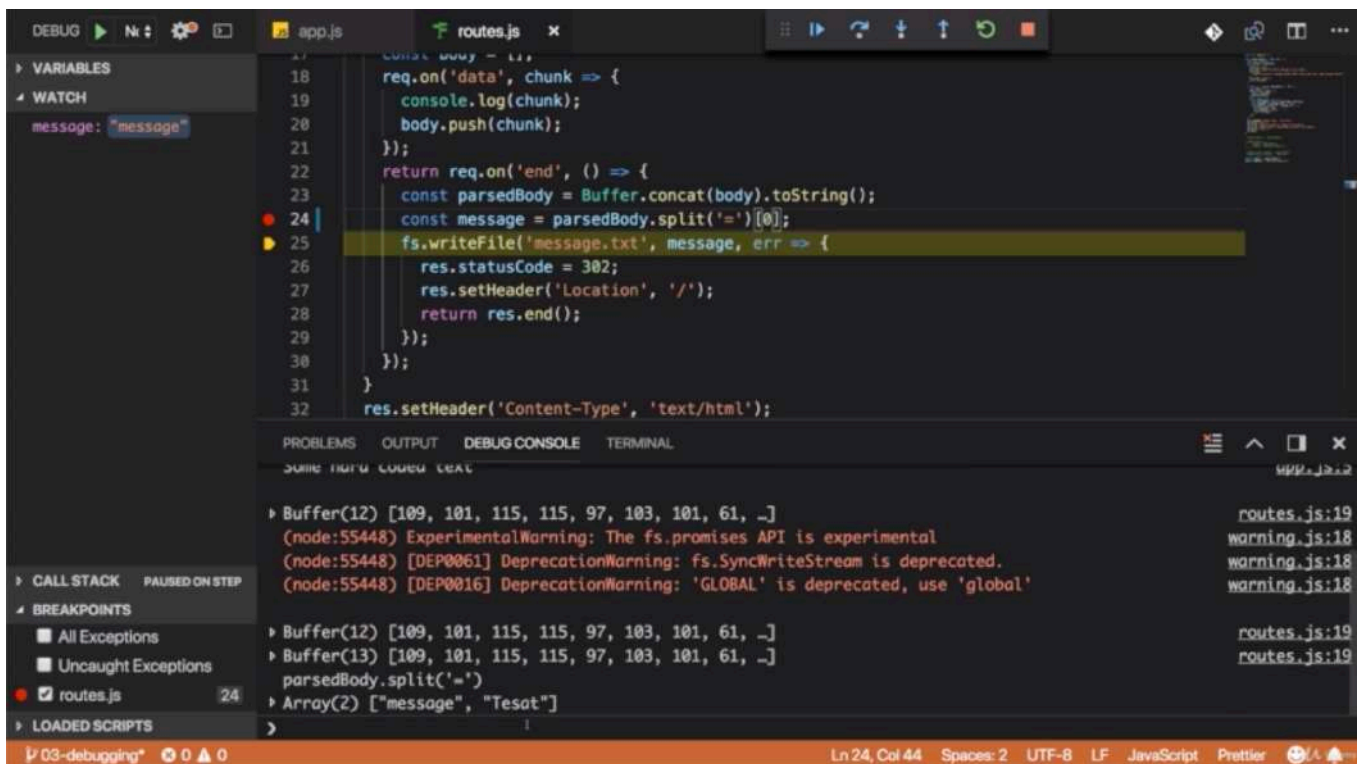


- so let's remove that breakpoint and resume and let's again run this one more time
 - and now let's not resume execution but look into message and we see message is message and that looks wrong.
 - we see that `parsedBody` is message equals and then our value.
-



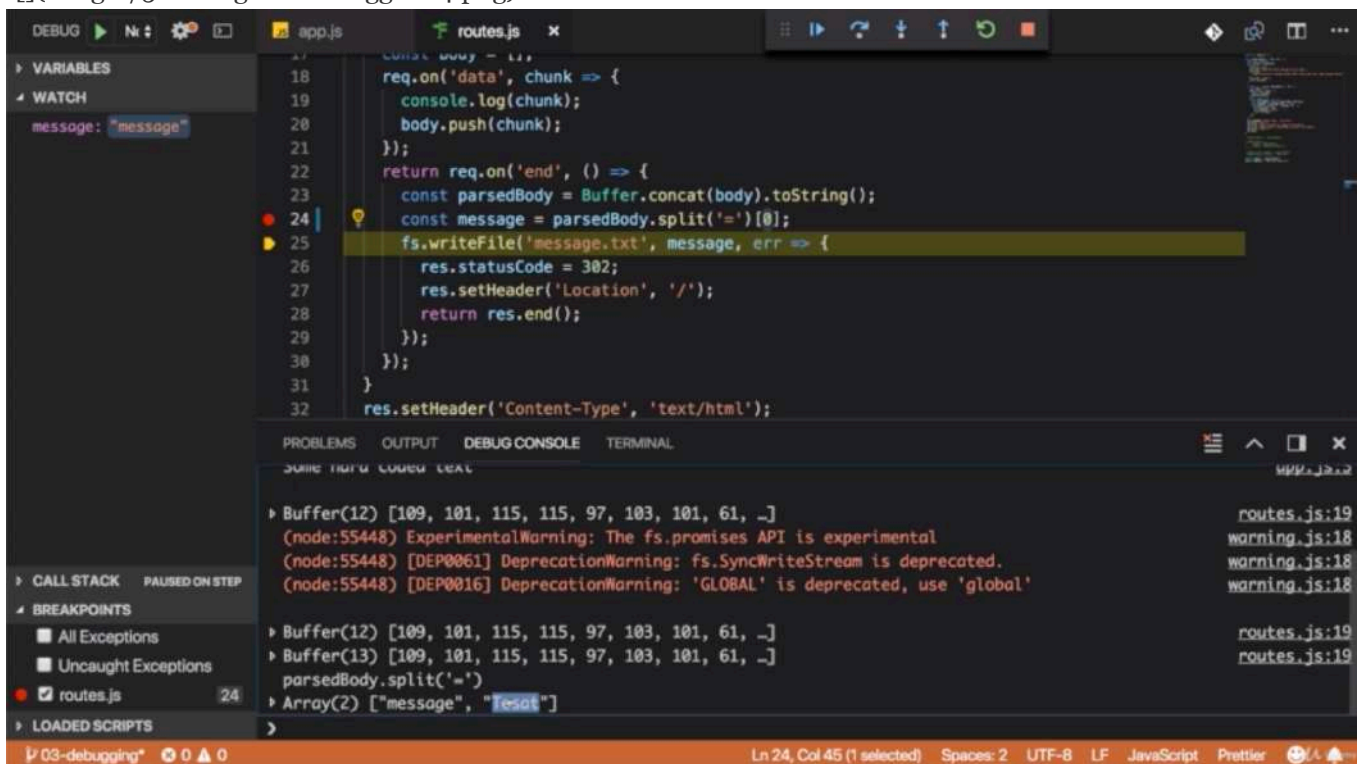
- now since this is what we stored in the message constant, we now can tell pretty clearly that our error has to be stemming from that part because we clearly have our value in here. but then we seem to be extracting the wrong piece



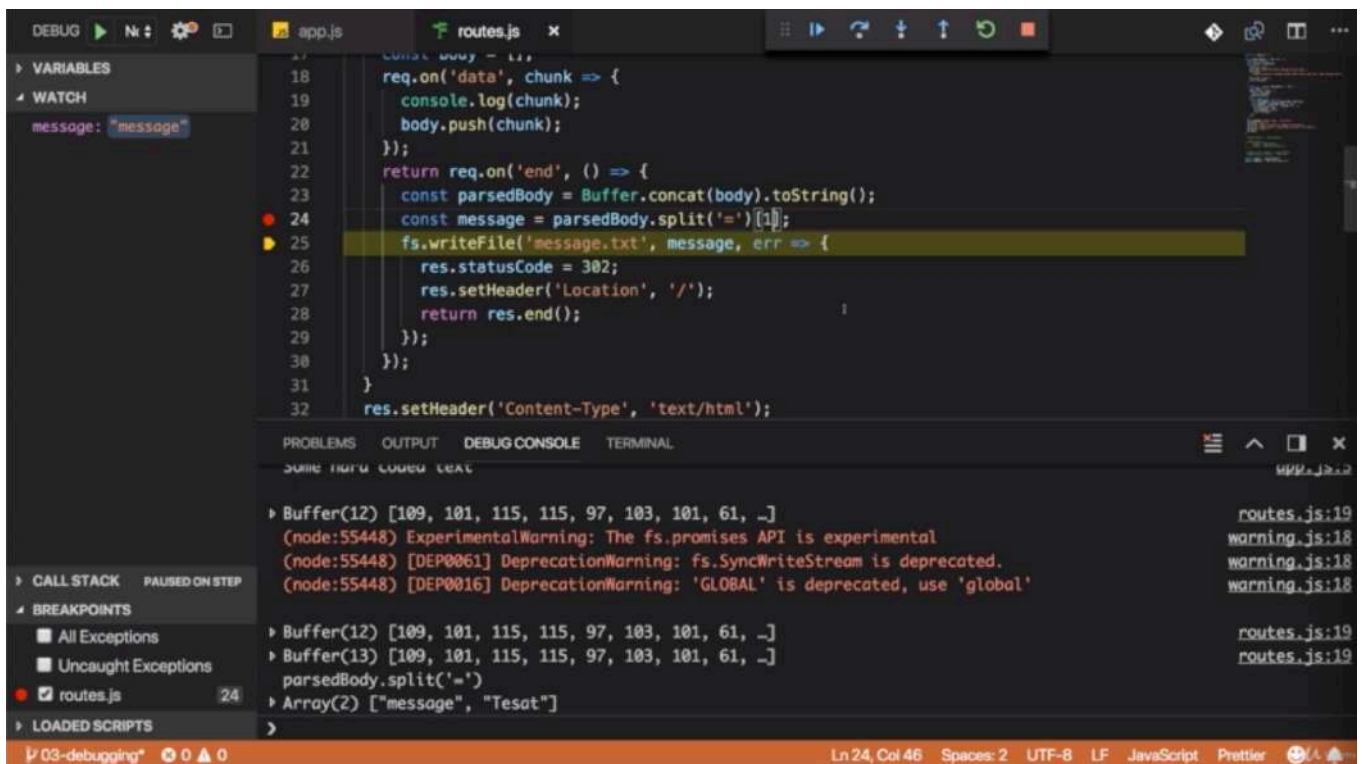


- and you can even use the console here to run some commands for example. so you can see you get an array with message and test because i mistyped test.

- this clearly tells you. split is working. i'm in theory getting all the values i need, so the only thing that can be wrong now is the value i'm extracting here and we see that message. and message holds message which is incorrect(not hold value but hold key). message the first element in the array. so i seem to be extracting the wrong element from the array



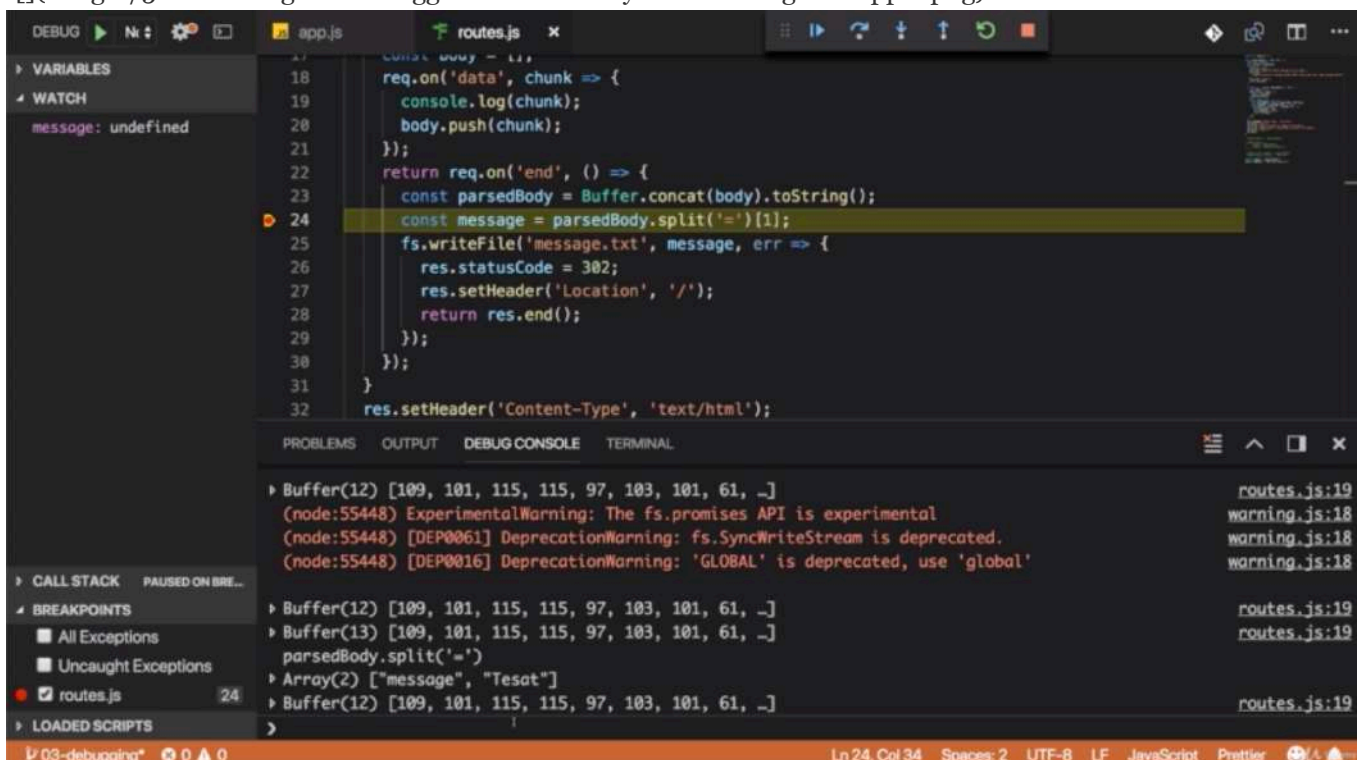
- and indeed we are extracting the first one when we should be extracting the second one here.



- and now we get this error fixed

- you have to keep in mind that node doesn't execute line after line, but it register callbacks which are executed sometime in the future, and which should therefore also have to control with breakpoints if you wanna look into them.

* Chapter 51: Restarting The Debugger Automatically After Editing Our App



- you can execute code here in that debug console at the bottom, so you don't see the console log here, you can also execute code here.

- something which is available in your code at this point of time, in general something which you can find in local or in block here or in global.

- you can also type that down there and hit enter to print its value.

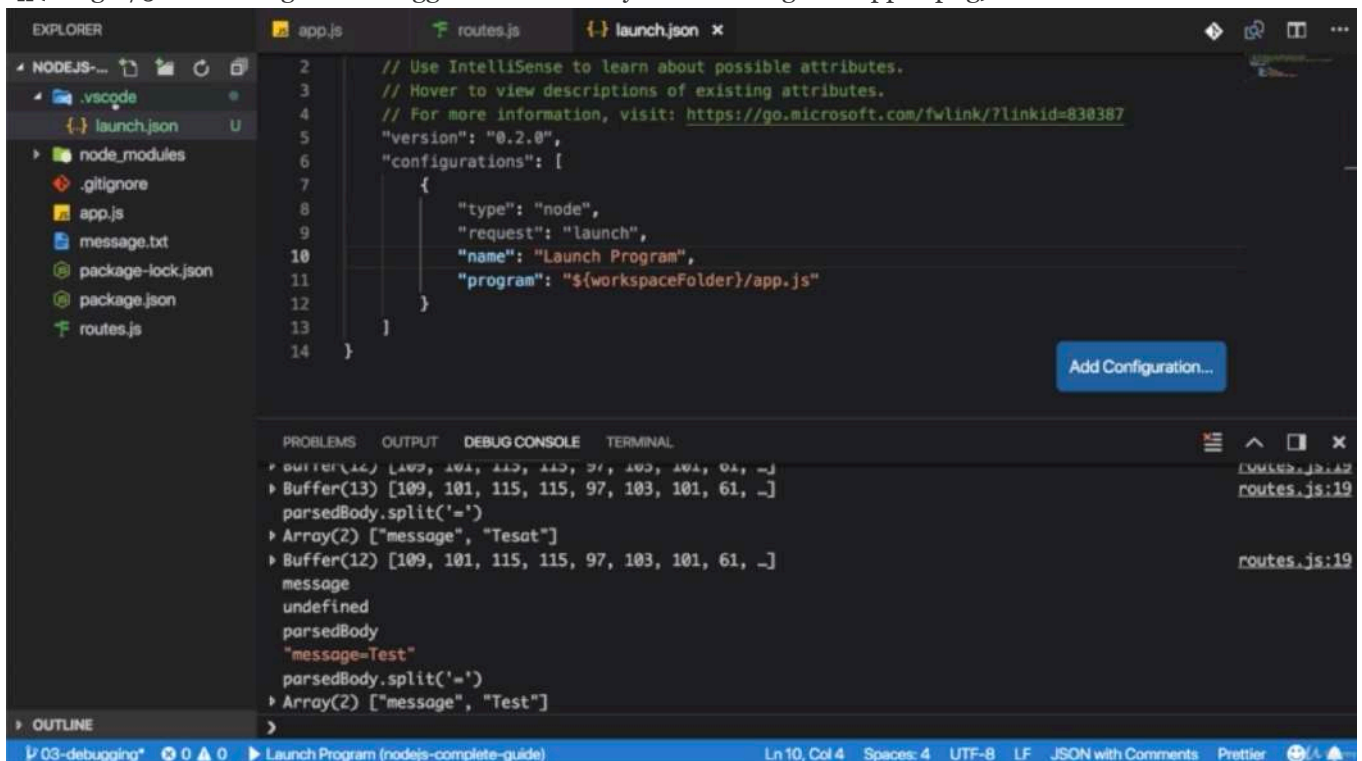
- you cannot just print the value, you can also run operation on it that will not affect your code as it's running but that will allow you to look into it or try out some transformation before you add it to your real code.

- if you add a blank line, our debugger doesn't restart but with nodemon we actually have a package that does allow us to restart. so it would be nice if the debugger would also restart if we change our code. otherwise it just behaves differently than the rest of our app and we have to restart it manually.

- now to restart it, let's go back to the explorer view for a second. then you have to go to debug and then add a configuration for node.js

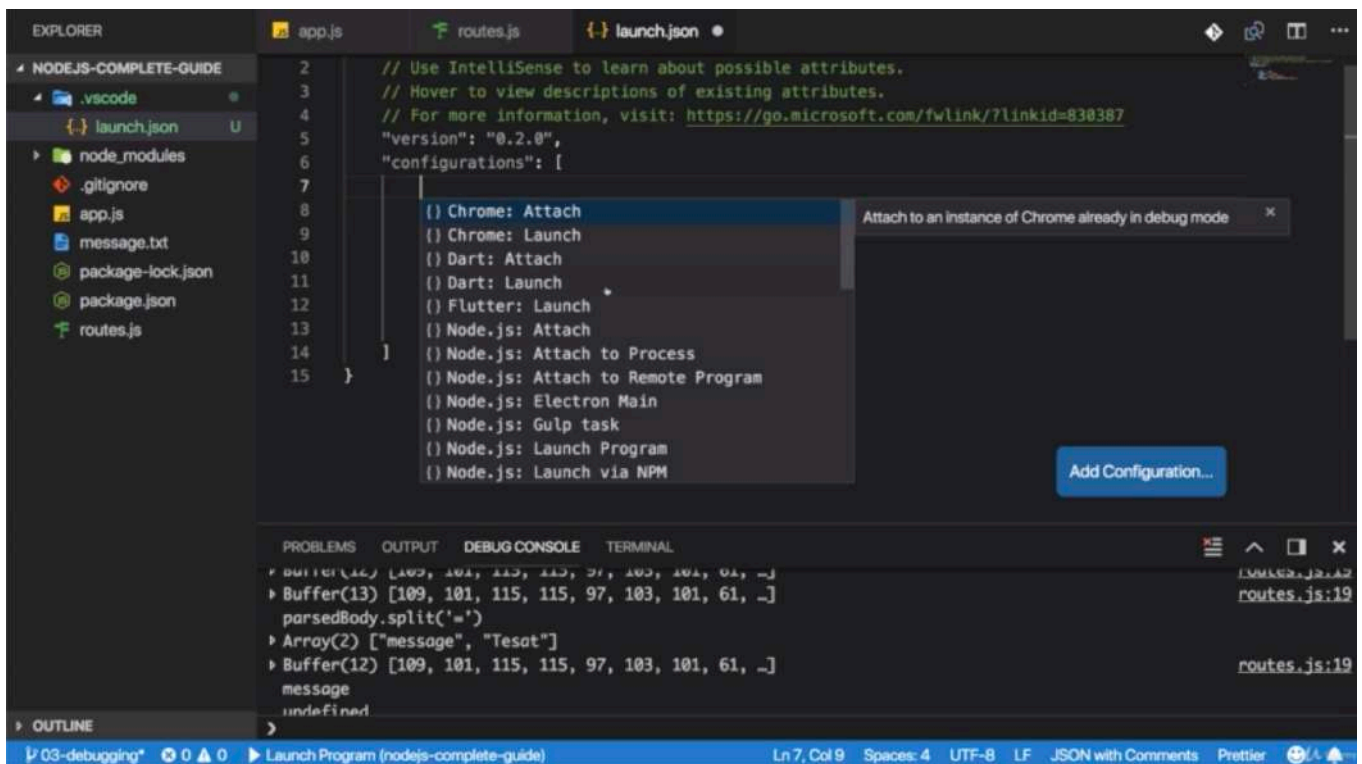
- this adds the .vscode folder with the launch json file.

- this allows you to configure debugging for this project and how it behaves. you can click on add configuration to see some demo settings you can add but you can also just type in there



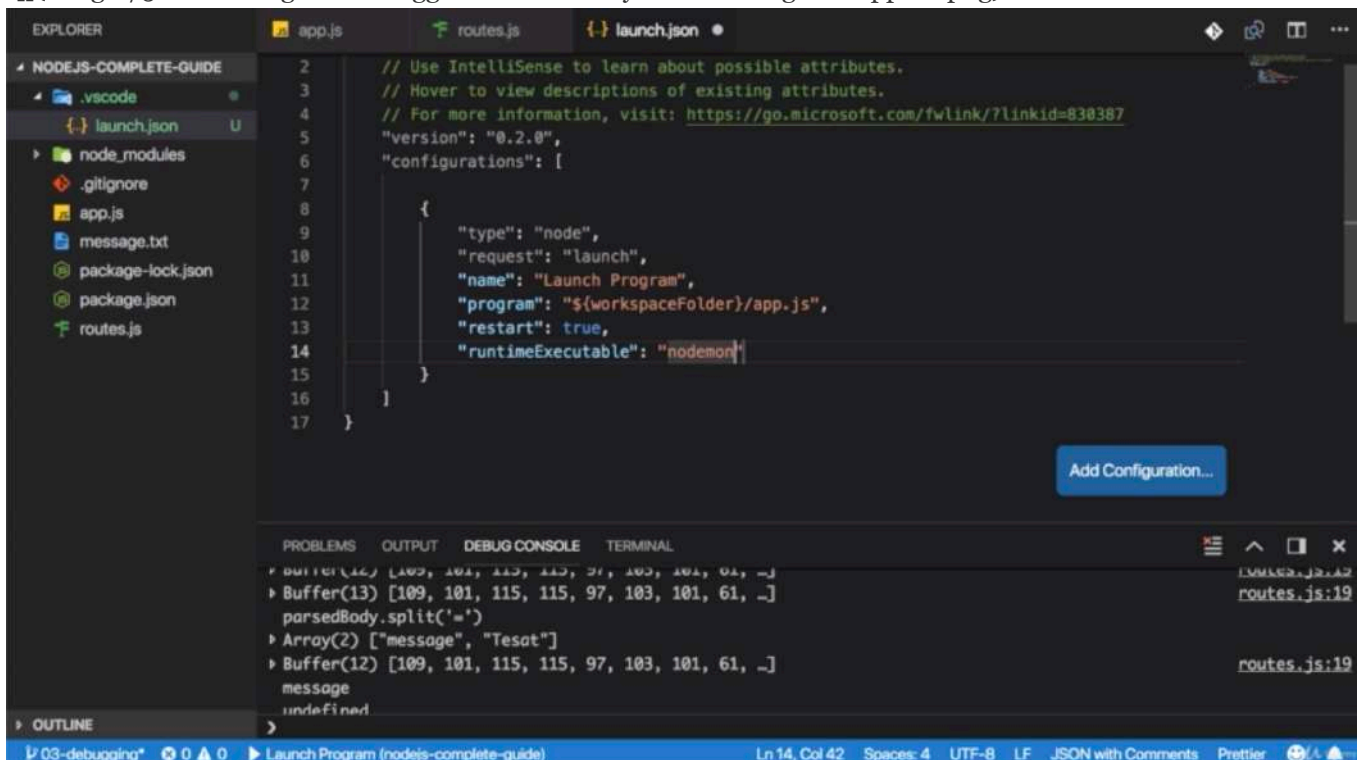
- and one setting you can add is restart and you can set it to true.

- you have to make sure that nodemon is used and for that, you set the runtime executable not to node which would be the default but to nodemon. now it will use nodemon and it will restart the debugger when a change is detected. so that not just the server is restarted but also the debugging process.

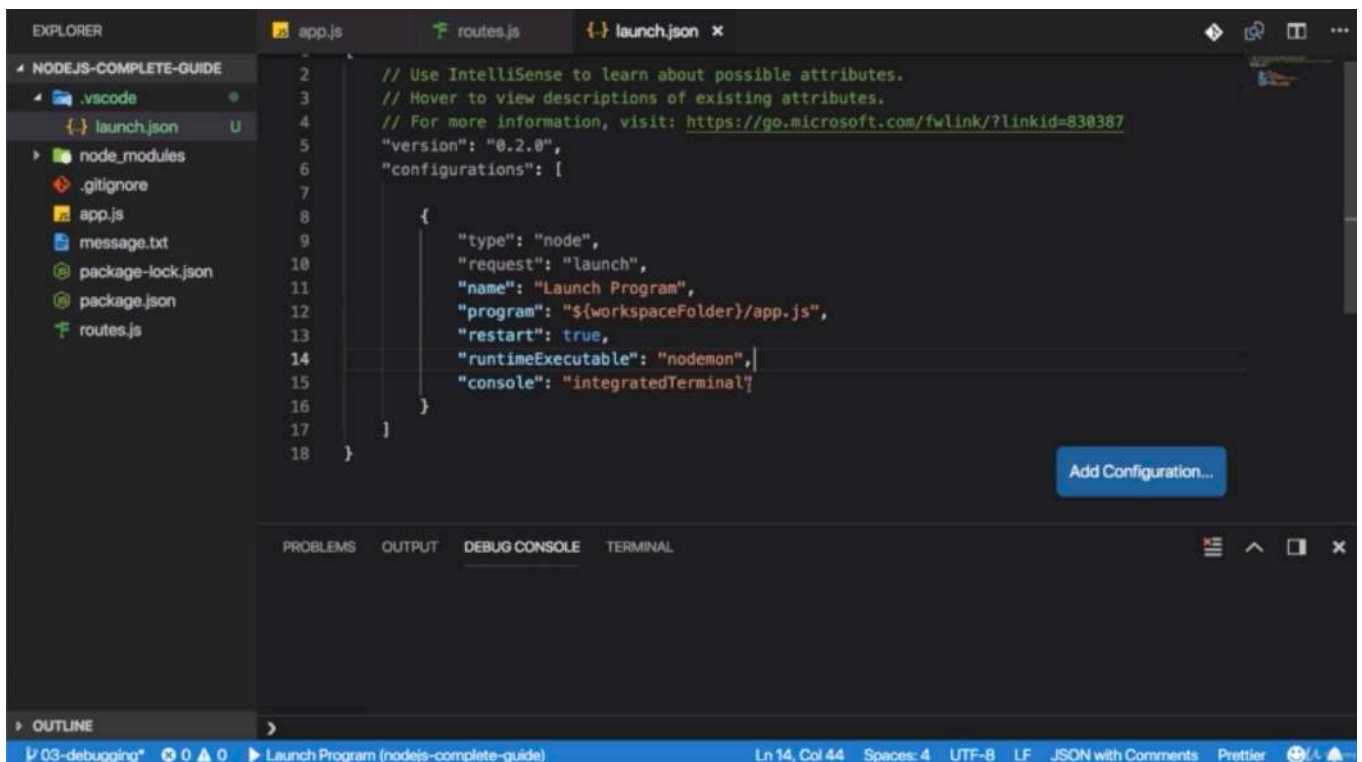
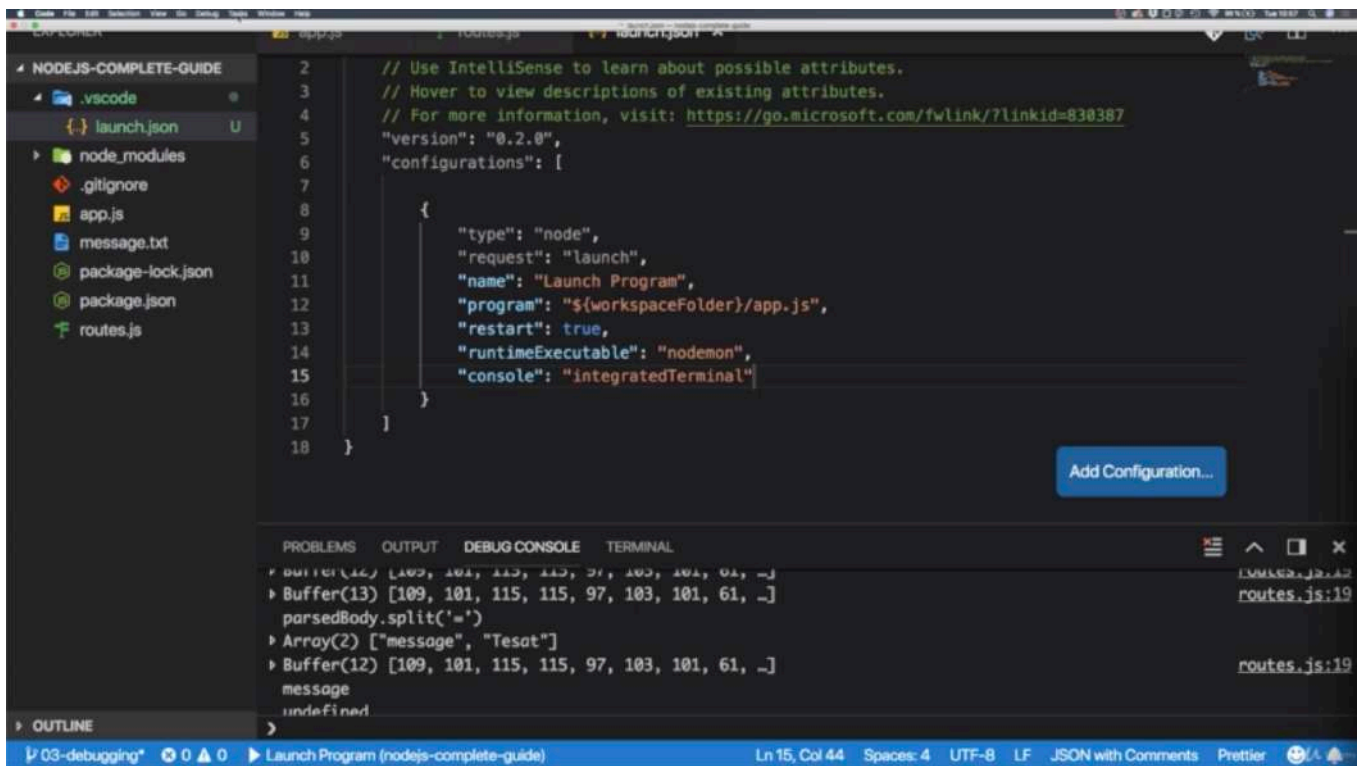


- you can also change the console where things are logged to the integrated terminal which is the normal terminal.

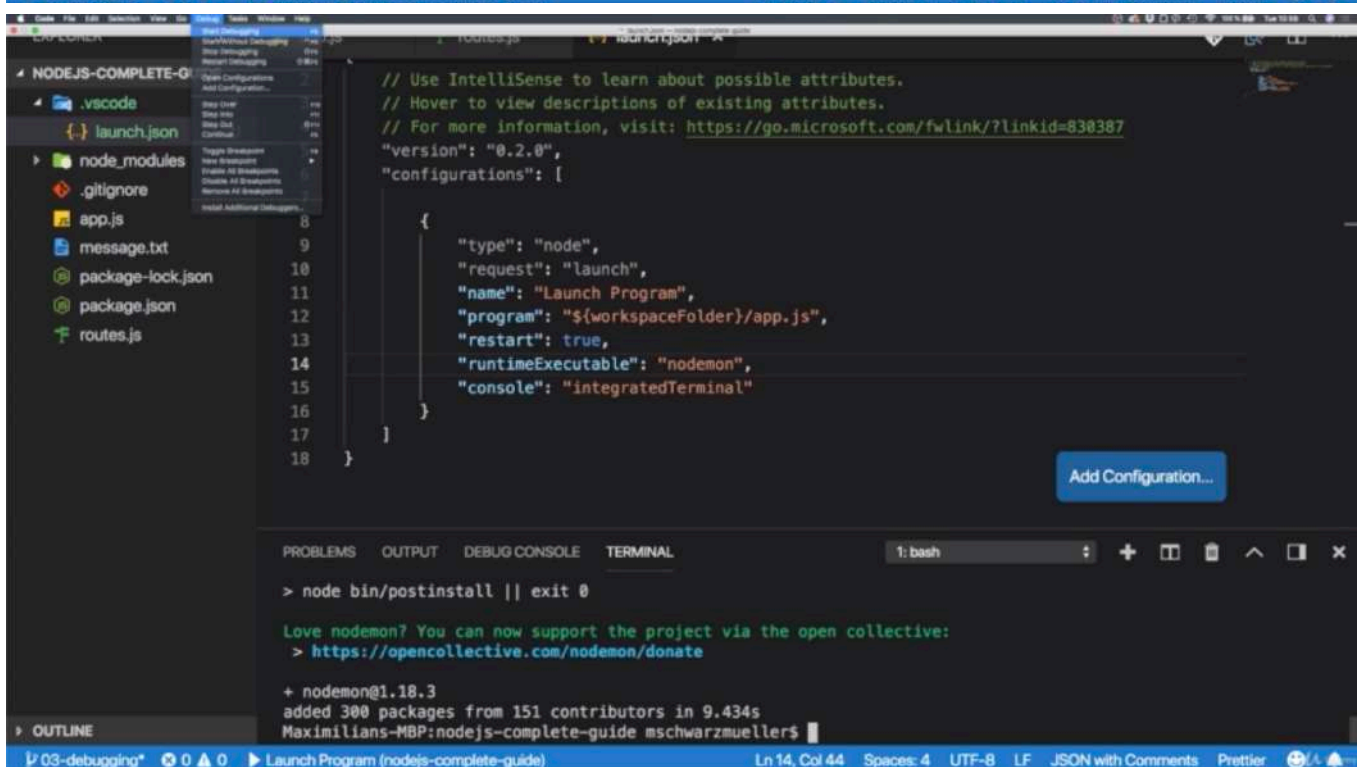
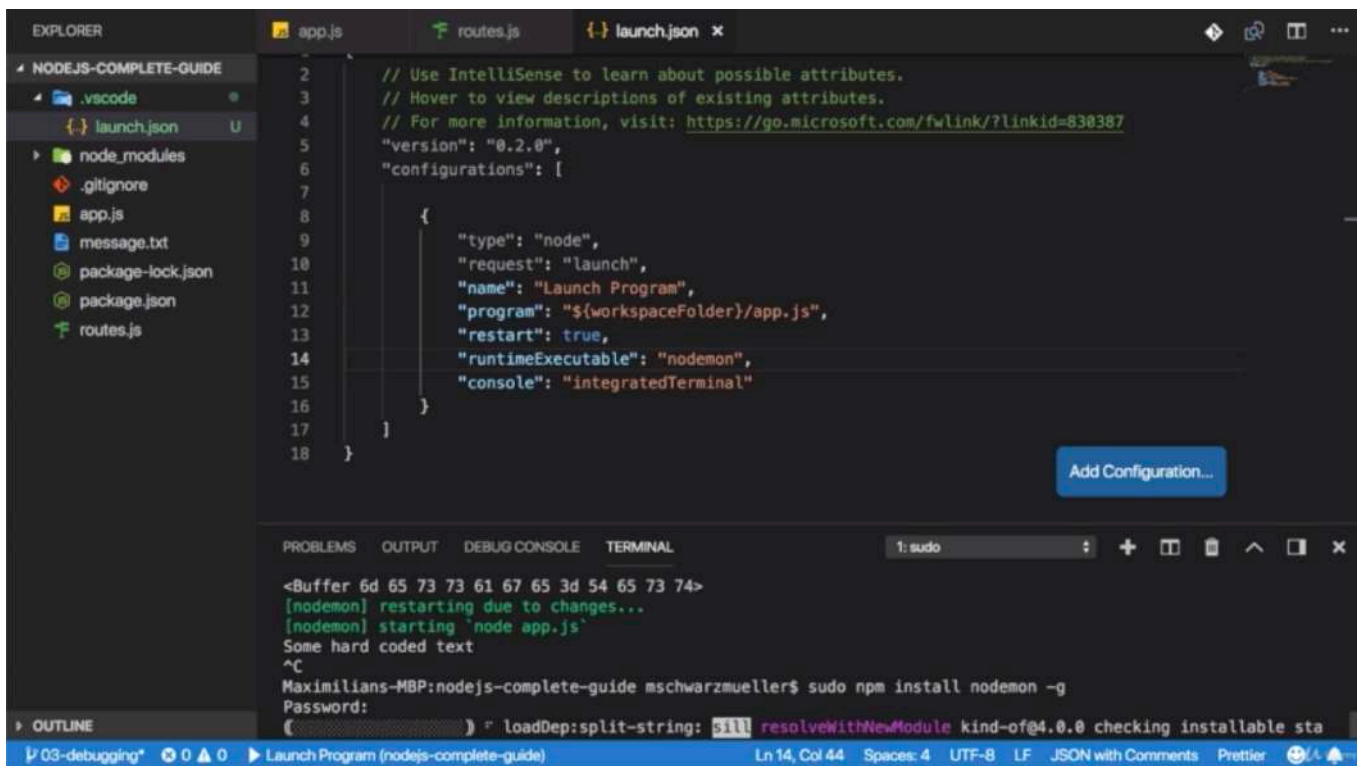
- when you start debugging, it fails though because the reason for this is that it will not use the local nodemon but global nodemon.

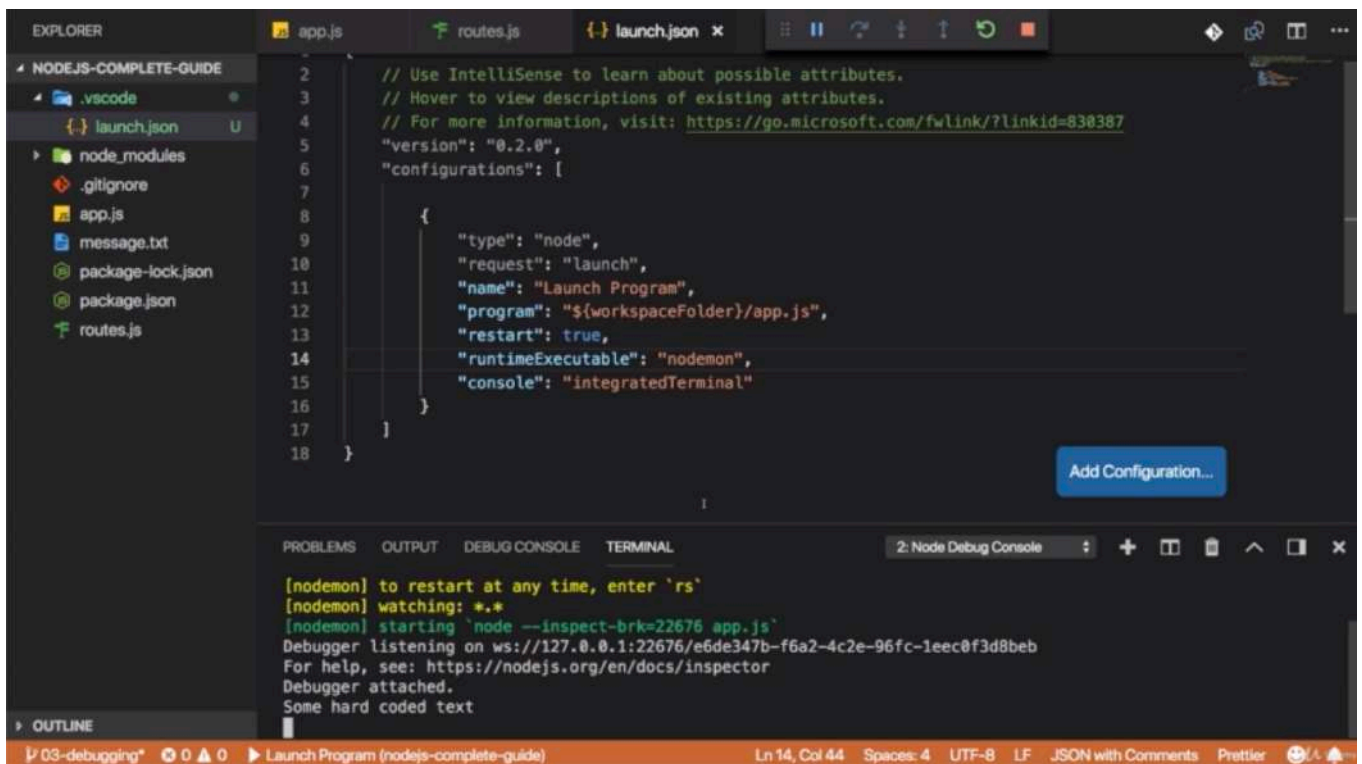


- so to add it globally, you have to run an `install nodemon -g`



- after installing, you can also run this debugger here which will use the global version and you will see it now opens terminal, starts nodemon

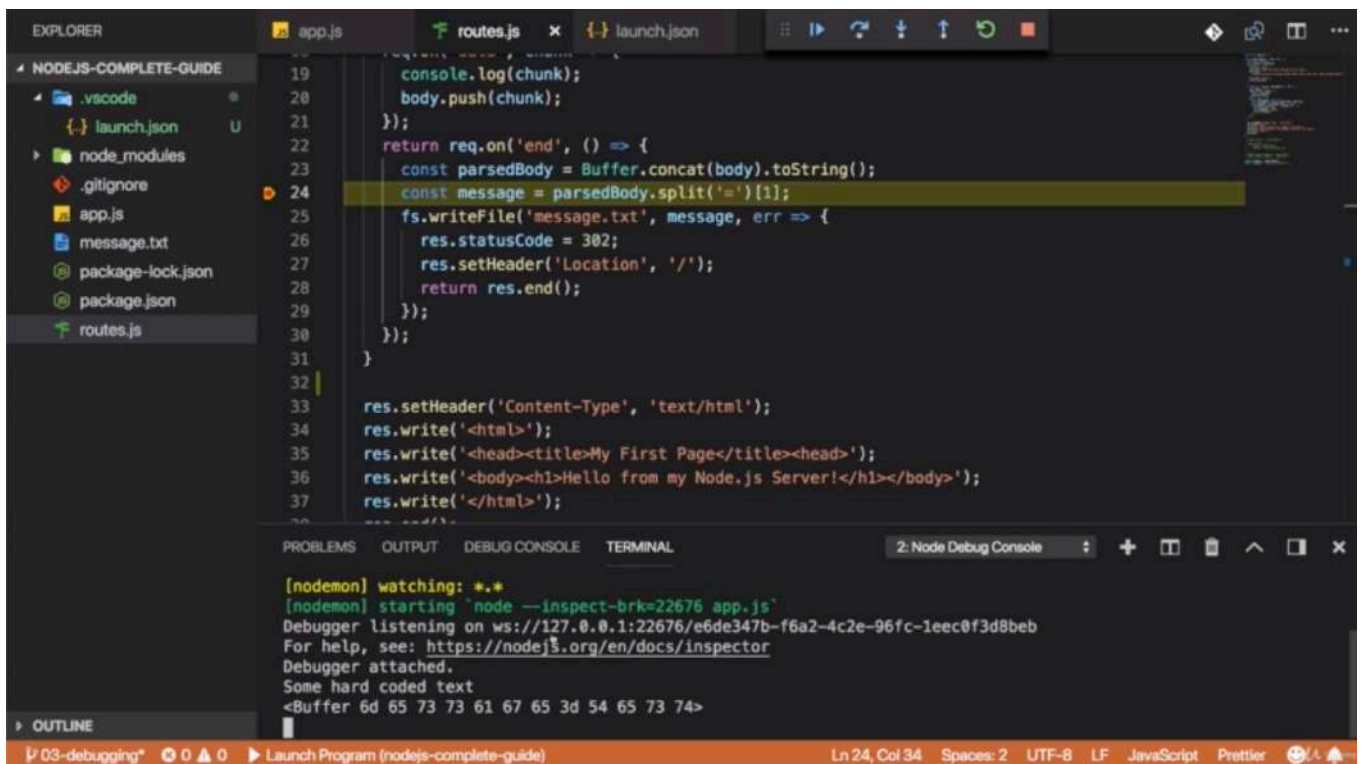




- and if you now add a breakpoint somewhere and submit this again. it works as before. you now see terminal all is logged in here and you can still use the debug console to output message though, so you can still work.

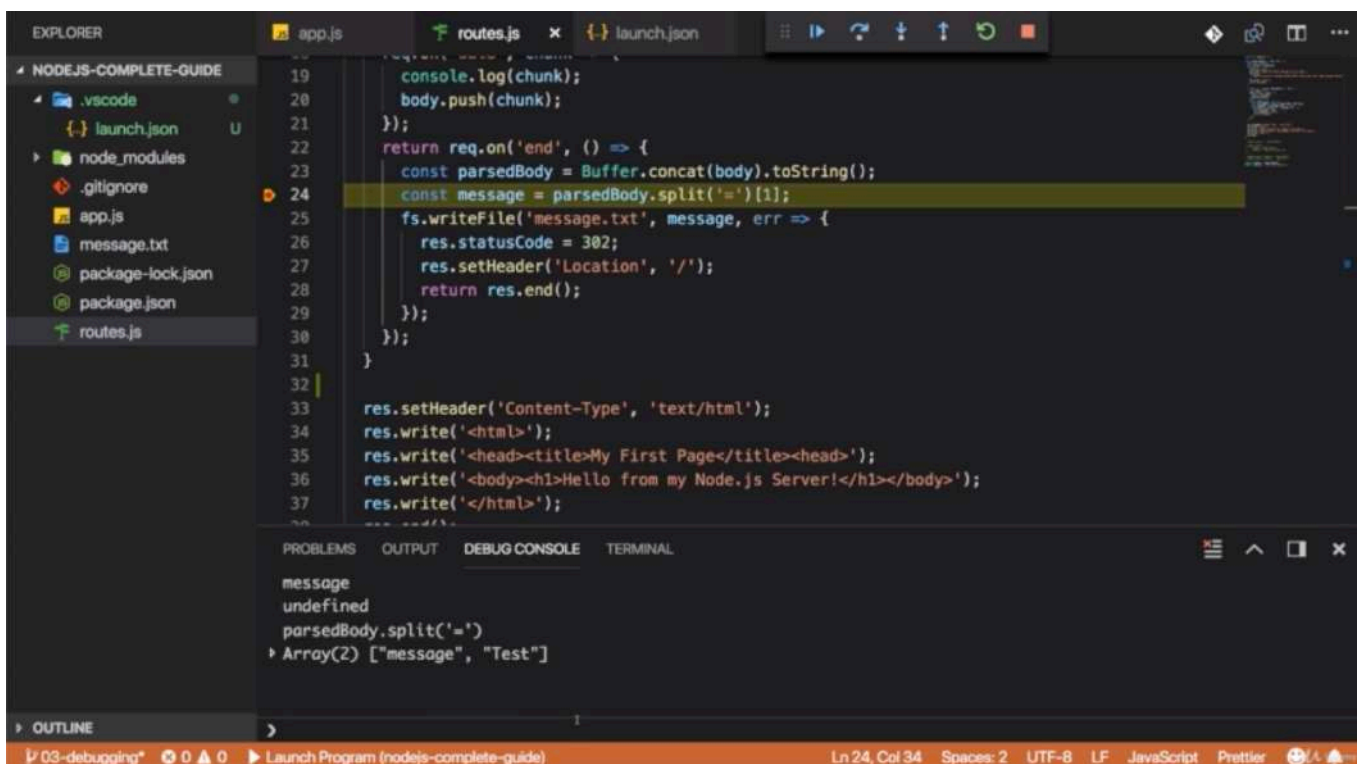


- but in the terminal, you get the normal output and you have to use the terminal because if you now change something, it restart the debugger and node and these are 2 separate processes.



```
19 console.log(chunk);
20 body.push(chunk);
21 });
22 return req.on('end', () => {
23   const parsedBody = Buffer.concat(body).toString();
24   const message = parsedBody.split('=')[1];
25   fs.writeFile('message.txt', message, err => {
26     res.statusCode = 302;
27     res.setHeader('Location', '/');
28     return res.end();
29   });
30 });
31 }
32
33 res.setHeader('Content-Type', 'text/html');
34 res.write('<html>');
35 res.write('<head><title>My First Page</title><head>');
36 res.write('<body><h1>Hello from my Node.js Server!</h1></body>');
37 res.write('</html>');
```

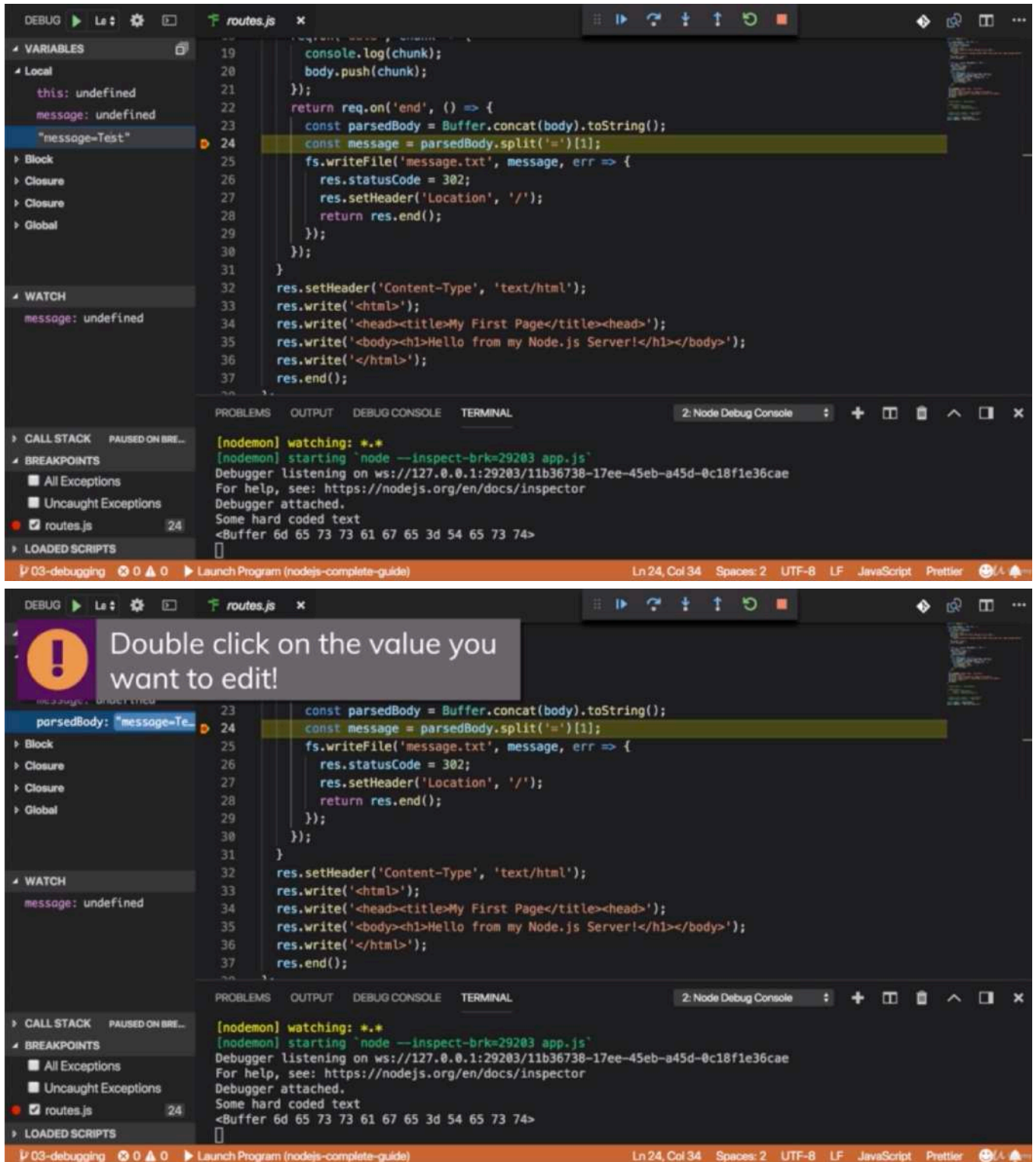
[nodemon] watching: *.*
[nodemon] starting `node --inspect-brk=22676 app.js`
Debugger listening on ws://127.0.0.1:22676/e6de347b-f6a2-4c2e-96fc-1eec0f3d8beb
For help, see: <https://nodejs.org/en/docs/inspector>
Debugger attached.
Some hard coded text
<Buffer 6d 65 73 73 61 67 65 3d 54 65 73 74>



```
message
undefined
parsedBody.split('=')
Array(2) ["message", "Test"]
```

- and if you stop the debugger, nodemon has to quit separately or has to exit separately and you do this by ctrl + C
- and this couldn't be done in the debug console which is why you have to funnel this to the terminal.
- you have to stop that process separately which you can do from the terminal which is why if you are using that nodemon process, you should use the integrated terminal

* Chapter 53: Changing Variables In The Debug Console



* Chapter 54: Wrap Up

Module Summary

npm	3rd Party Packages
<ul style="list-style-type: none"> • npm stands for "Node Package Manager" and it allows you to manage your Node project and its dependencies • You can initialize a project with <code>npm init</code> • npm scripts can be defined in the <code>package.json</code> to give you "shortcuts" to common tasks/ commands 	<ul style="list-style-type: none"> • Node projects typically don't just use core modules and custom code but also third-party packages • You install them via npm • You can differentiate between production dependencies (<code>--save</code>), development dependencies (<code>--save-dev</code>) and global dependencies (<code>-g</code>)
Types of Errors	Debugging
<ul style="list-style-type: none"> • Syntax, runtime and logical errors can break your app • Syntax and runtime errors throw (helpful) error messages (with line numbers!) • Logical errors can be fixed with testing and the help of the debugger 	<ul style="list-style-type: none"> • Use the VS Code Node debugger to step into your code and go through it step by step • Analyze variable values at runtime • Look into (and manipulate) variables at runtime • Set breakpoints cleverly (i.e. respect the async/ event-driven nature)