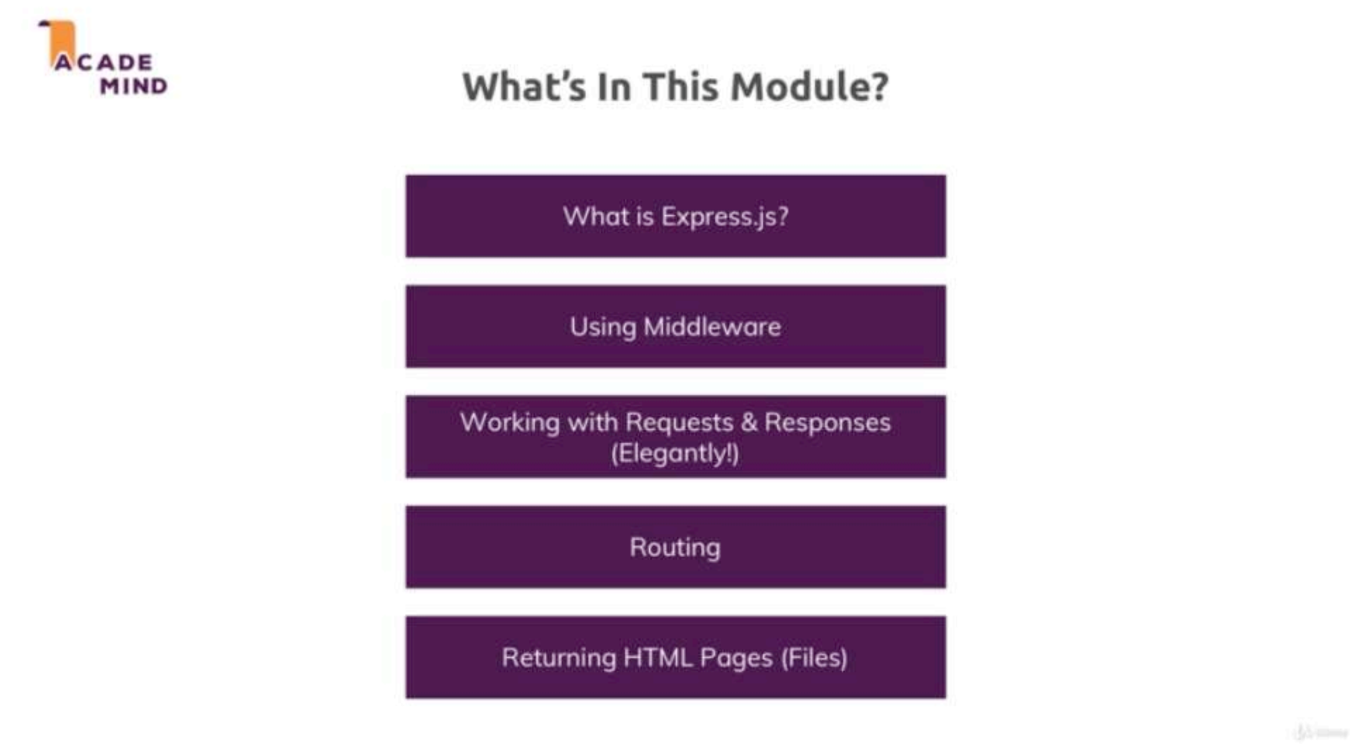
# 5. Working With Express.js

## * Chapter 56: Module Introduction

![](images/56-module-introduction-1.png)



What's In This Module?

What is Express.js?

Using Middleware

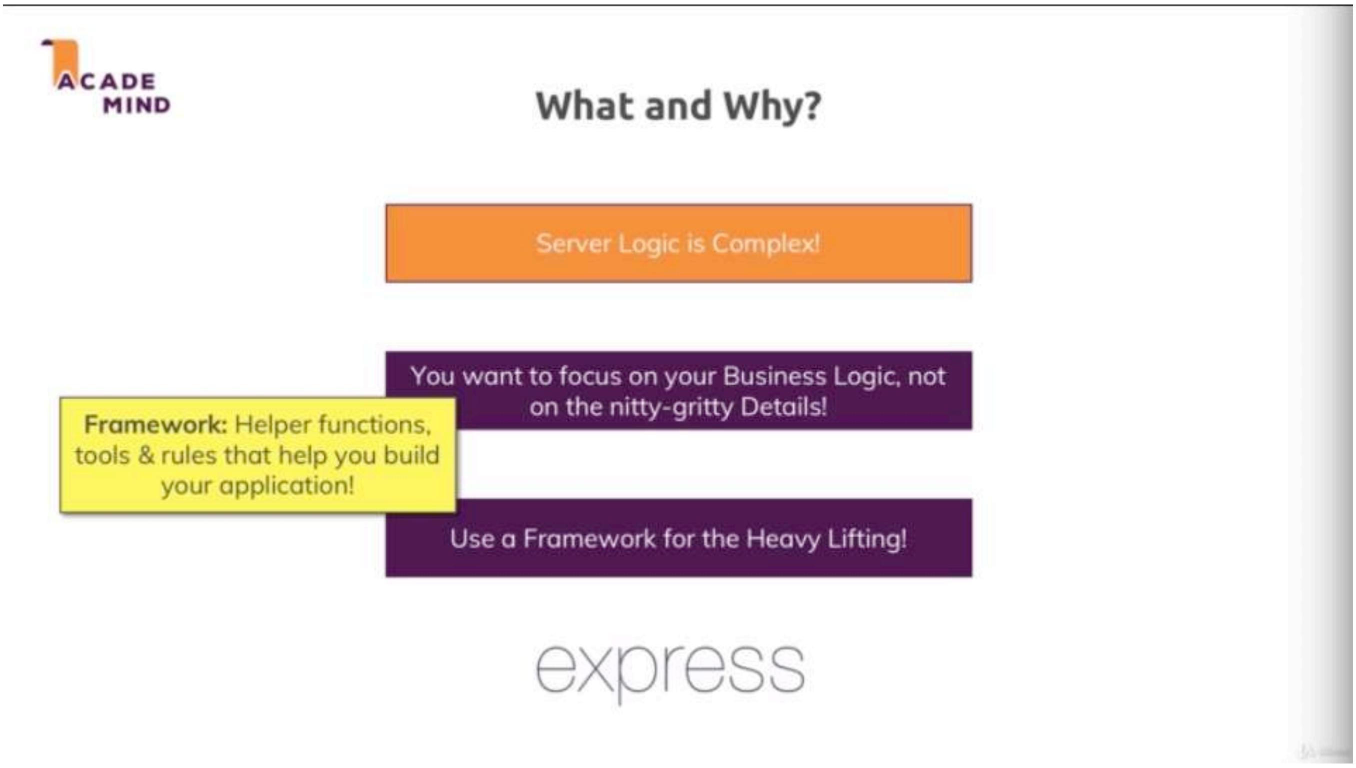
Working with Requests & Responses (Elegantly!)

Routing

Returning HTML Pages (Files)

## * Chapter 57: What Is Express.js?

![](images/57-what-is-express.js-1.png)
![](images/57-what-is-express.js-2.png)
![](images/57-what-is-express.js-3.png)



## What and Why?

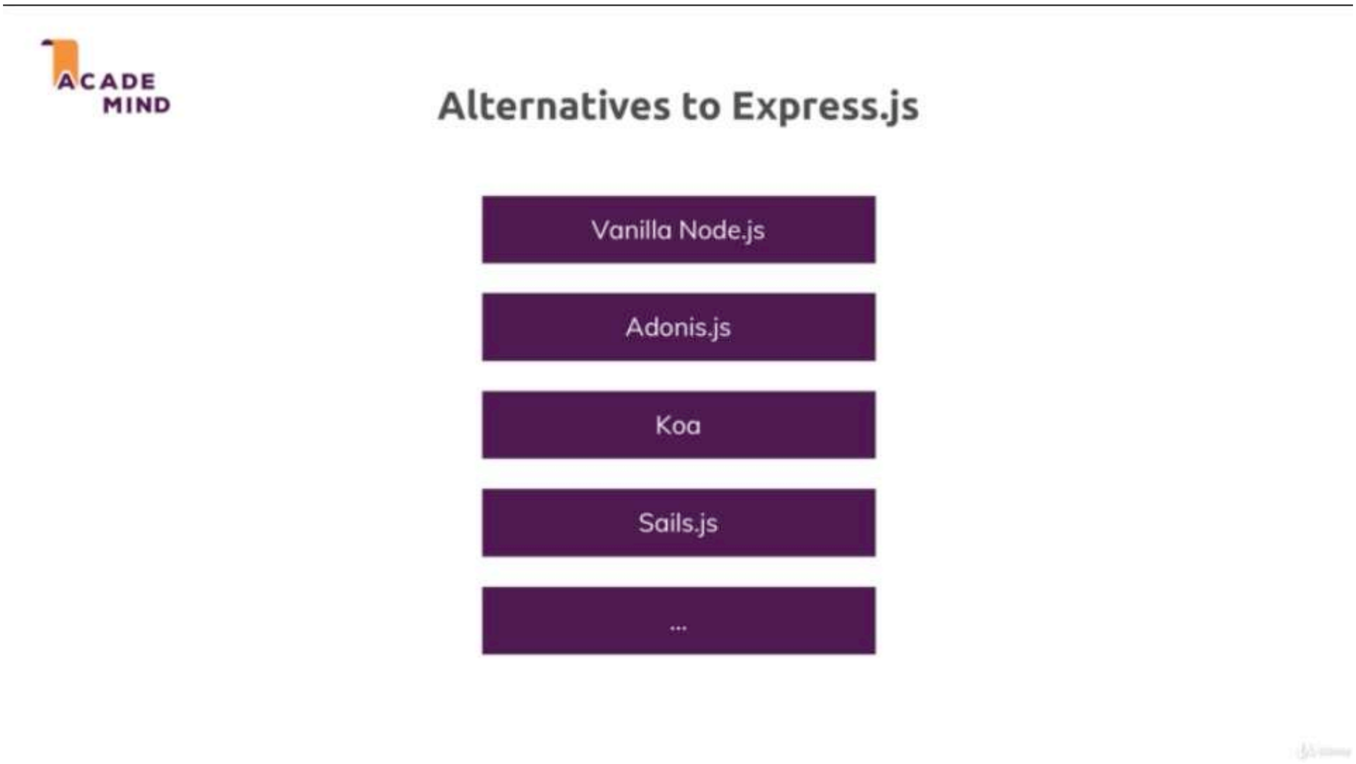
Server Logic is Complex!

You want to focus on your Business Logic, not on the nitty-gritty Details!

**Framework:** Helper functions, tools & rules that help you build your application!

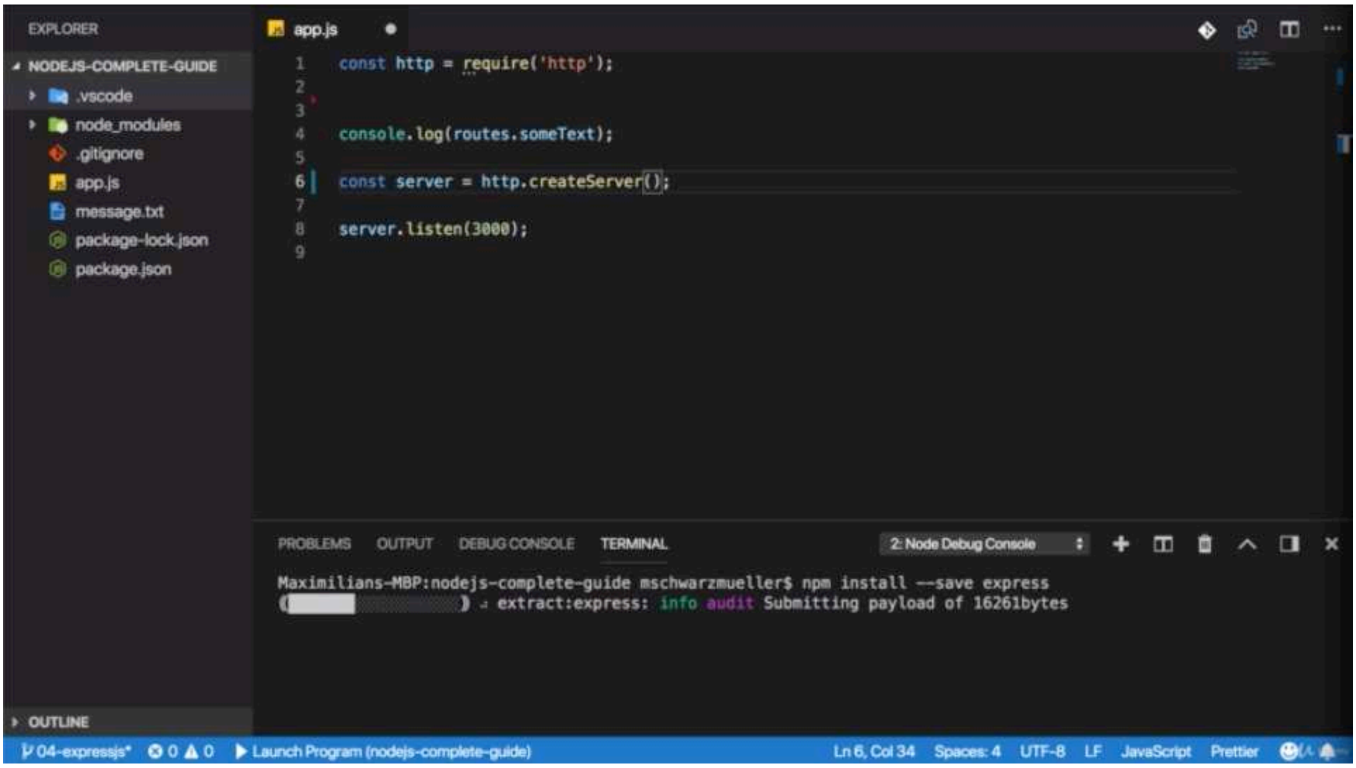
Use a Framework for the Heavy Lifting!

express



## Alternatives to Express.js

- Vanilla Node.js
- Adonis.js
- Koa
- Sails.js
- ...



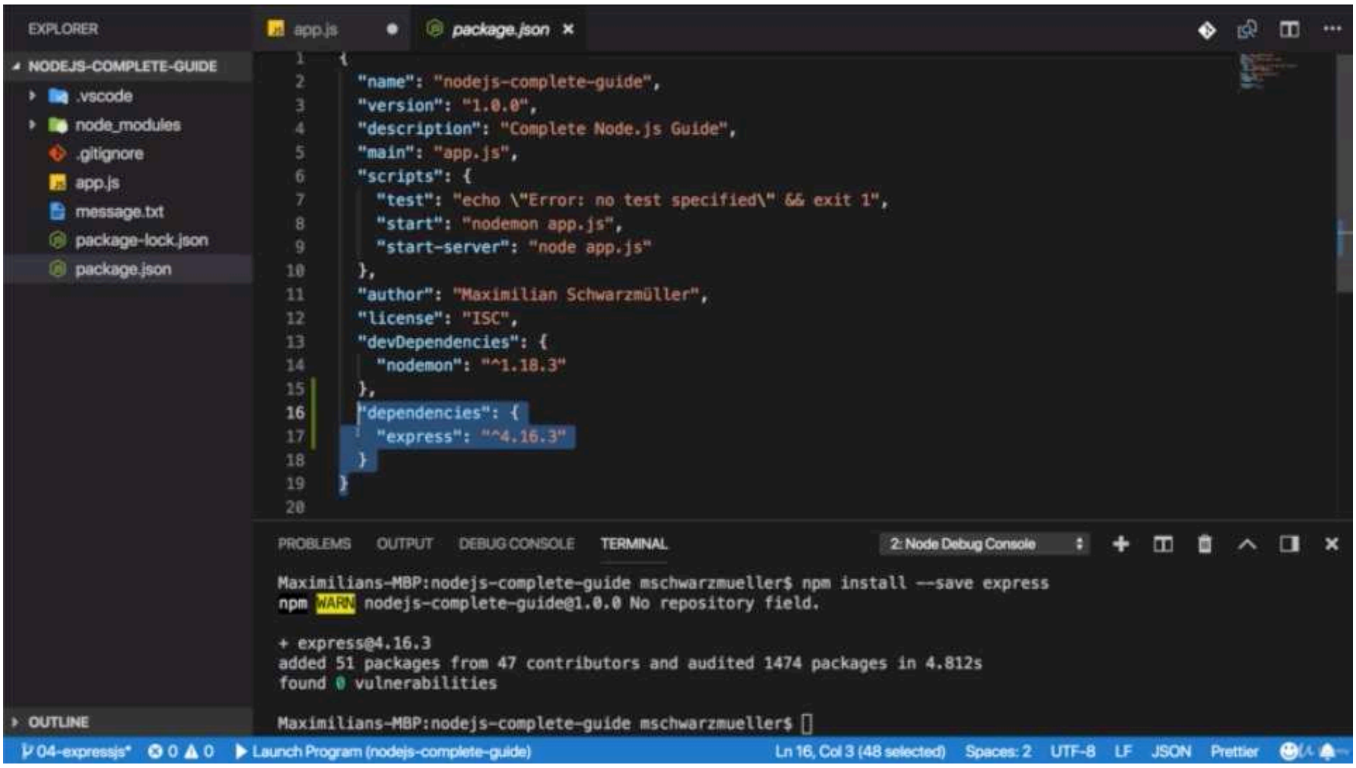
```
const http = require('http');


console.log(routes.someText);

const server = http.createServer();

server.listen(3000);
```

```
Maximilians-MBP:nodejs-complete-guide mschwarzmueller$ npm install --save express
(            ) ⠂ extract:express: info audit Submitting payload of 16261bytes
```

# * Chapter 58: Installing Express.js

1. update
- app.js

![](images/58-installing-express-js-1.png)

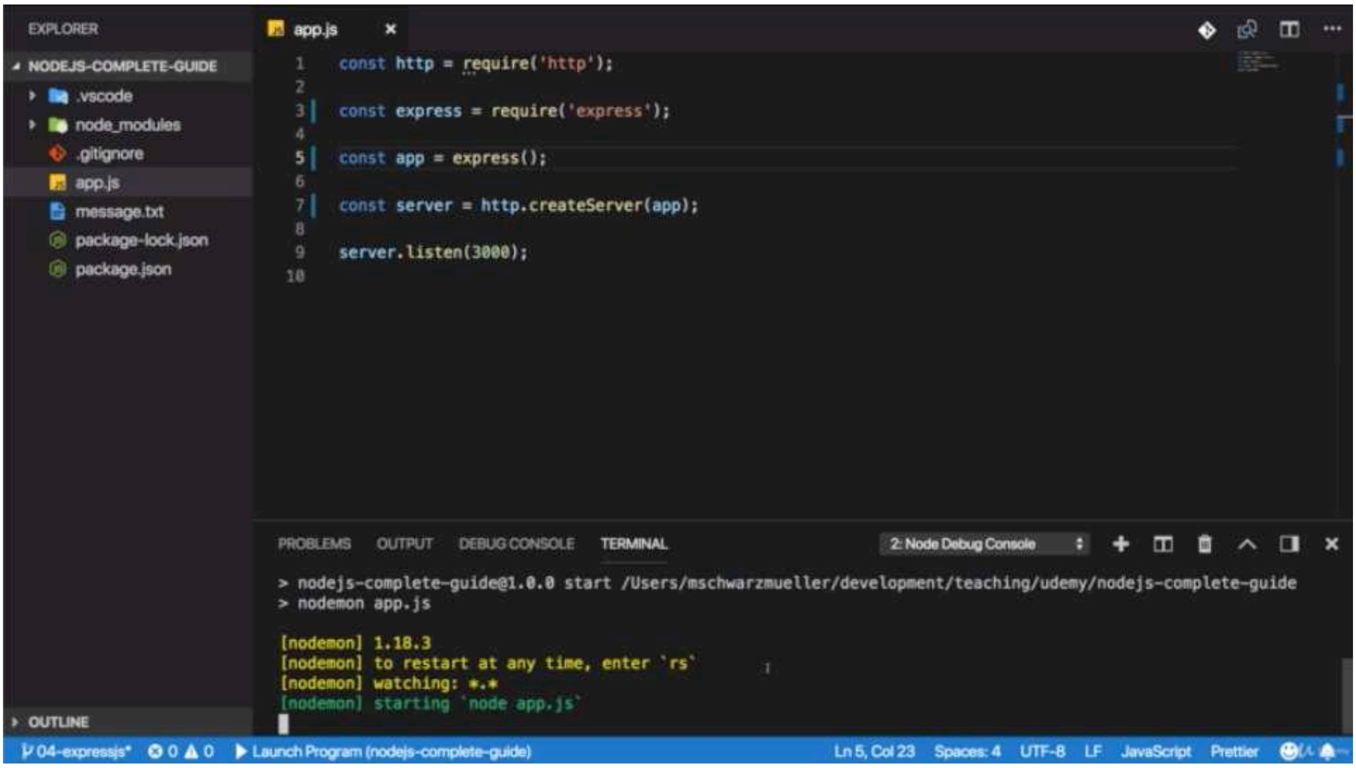


```
{
  "name": "nodejs-complete-guide",
  "version": "1.0.0",
  "description": "Complete Node.js Guide",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "nodemon app.js",
    "start-server": "node app.js"
  },
  "author": "Maximilian Schwarzmüller",
  "license": "ISC",
  "devDependencies": {
    "nodemon": "^1.18.3"
  },
  "dependencies": {
    "express": "^4.16.3"
  }
}
```

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

+ express@4.16.3
added 51 packages from 47 contributors and audited 1474 packages in 4.812s
found 0 vulnerabilities

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

- why not —save-dev but —save? because this will be a production dependency. we don’t just use that as a tool during development. it will be an integral part of the application we ship and therefore it definitely also has to be installed on any server or any computer where we run our aplication once we deploy it. it’s a major piece of our application.

![](images/58-installing-express-js-2.png)



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

const express = require('express');

const app = express();

const server = http.createServer(app);

server.listen(3000);
```

```
> 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`
```

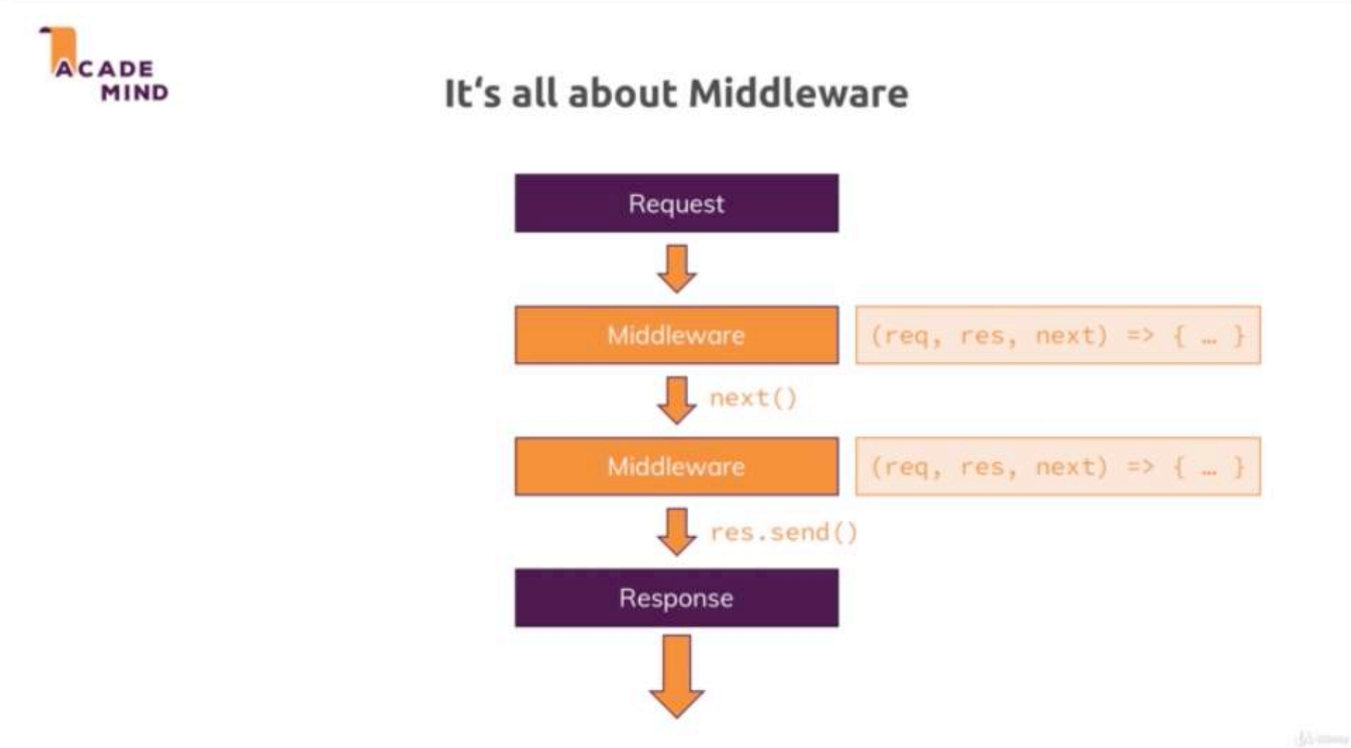
- after npm start, you will actually have a running server.

```
//app.js

const http = require('http');

/**this will initalize new object
*/
const express = require('express');

const app = express()

/** 'app' happens to be a valid request handler
 * so you can pass 'app' here to create server.
 *
 * app sets up a certain way of handling incoming requests
 * that defines or that is a key characteristic of express.js
*/
const server = http.createServer(app);

server.listen(3000);
```

# * Chapter 59: Adding Middleware

1. update
- app.js

![](images/59-adding-middleware-1.png)



It's all about Middleware

Request

Middleware `(req, res, next) => { … }`

`next()`

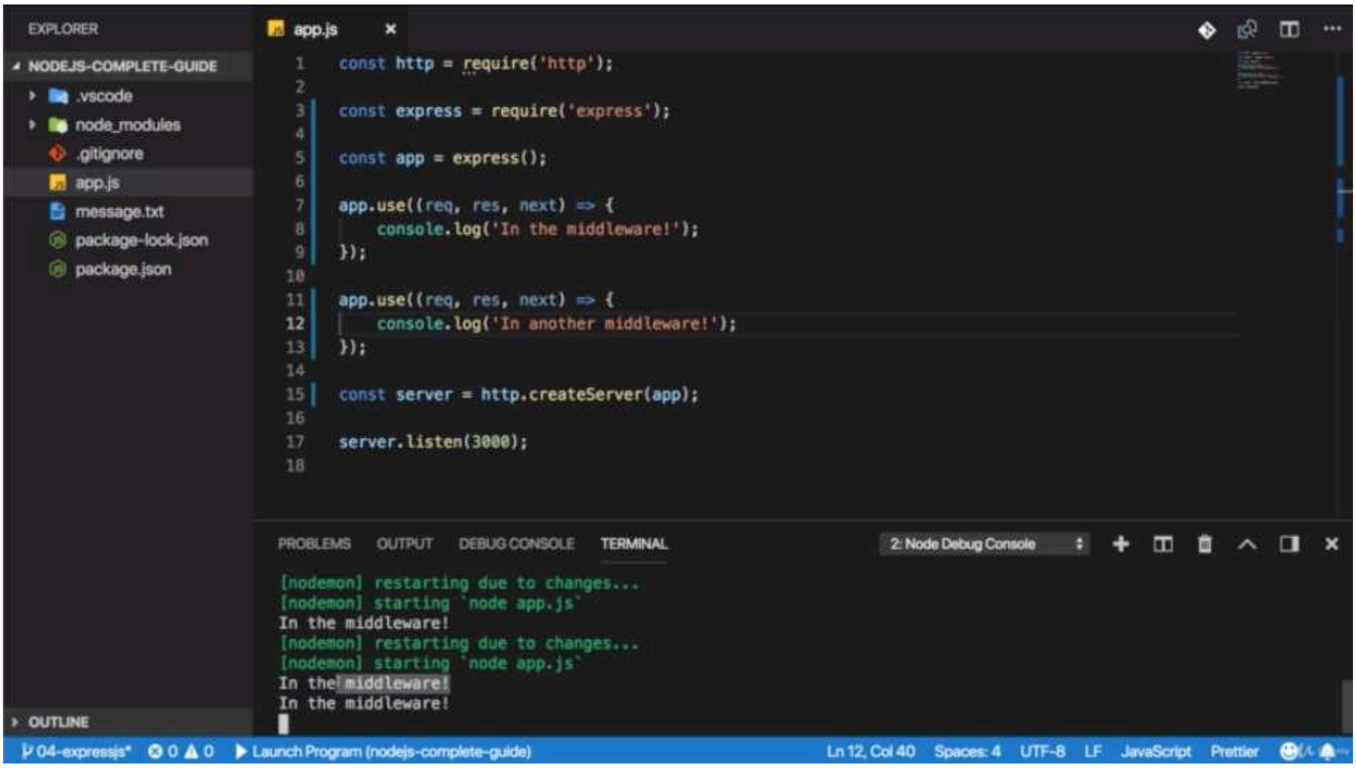
Middleware `(req, res, next) => { … }`

`res.send()`

Response

- Express.js is all about middleware which means an incoming request is automatically funneled through a bunch of function by express.js
- so instead of just having one request handler, you will actually have a possibility of hooking in multiple functions which the request will go through until you send a response.
- This allows you to split your code into multiple blocks or pieces instead of having one huge functionthat does everythingand this is the pluggable nature of expressjs, where you can easily add other third party packageswhich simply happen to give you such middleware functions that you can plug into expressjs and addcertain functionalities
![](images/59-adding-middleware-2.png)

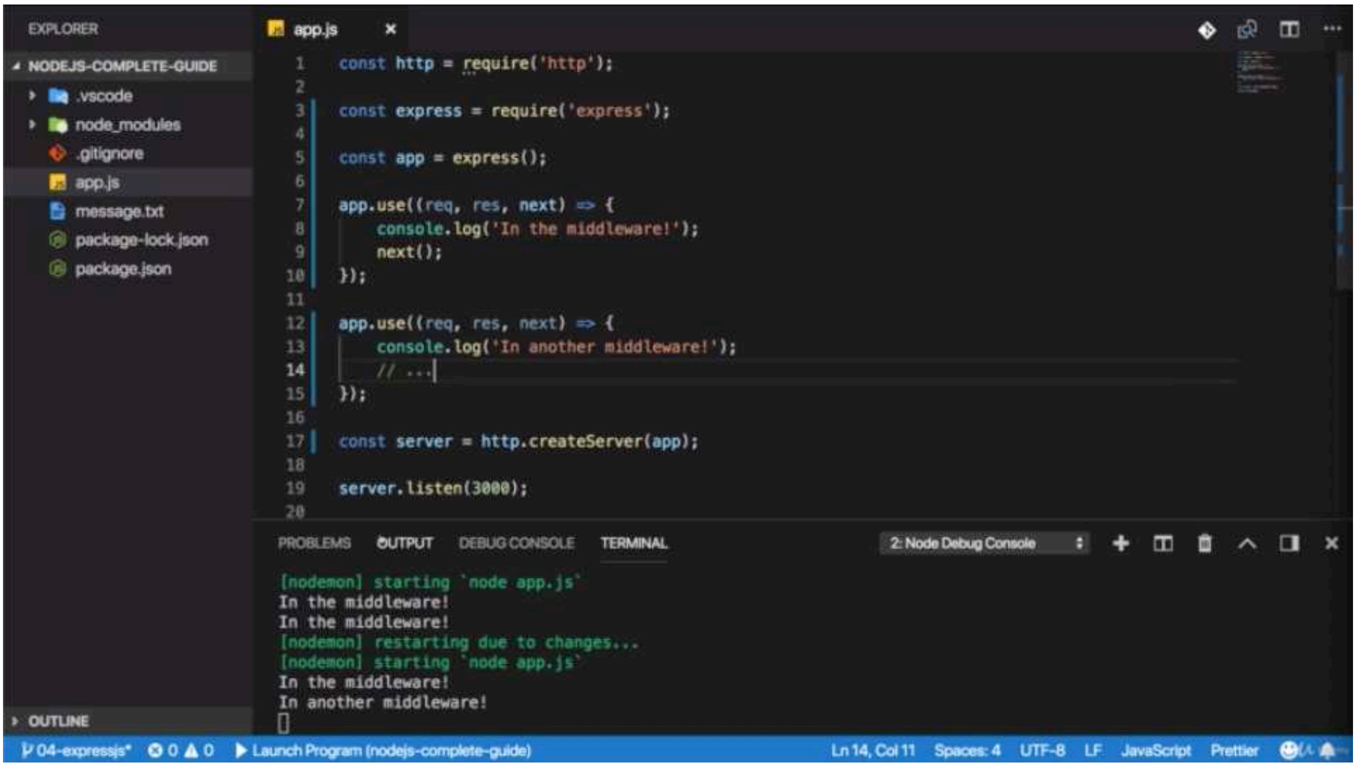
- this spinner will keep on spinning. so we don't get a response. because we got no logic.
![](images/59-adding-middleware-3.png)



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

const express = require('express');

const app = express();

app.use((req, res, next) => {
    console.log('In the middleware!');
});

app.use((req, res, next) => {
    console.log('In another middleware!');
});

const server = http.createServer(app);

server.listen(3000);
```

```
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
In the middleware!
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
In the middleware!
In the middleware!
```

- but i don’t see in another middleware. because we need to type ’next()’ to get next middleware. ![](images/59-adding-middleware-4.png)



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

const express = require('express');

const app = express();

app.use((req, res, next) => {
    console.log('In the middleware!');
    next();
});

app.use((req, res, next) => {
    console.log('In another middleware!');
    // ...
});

const server = http.createServer(app);

server.listen(3000);
```

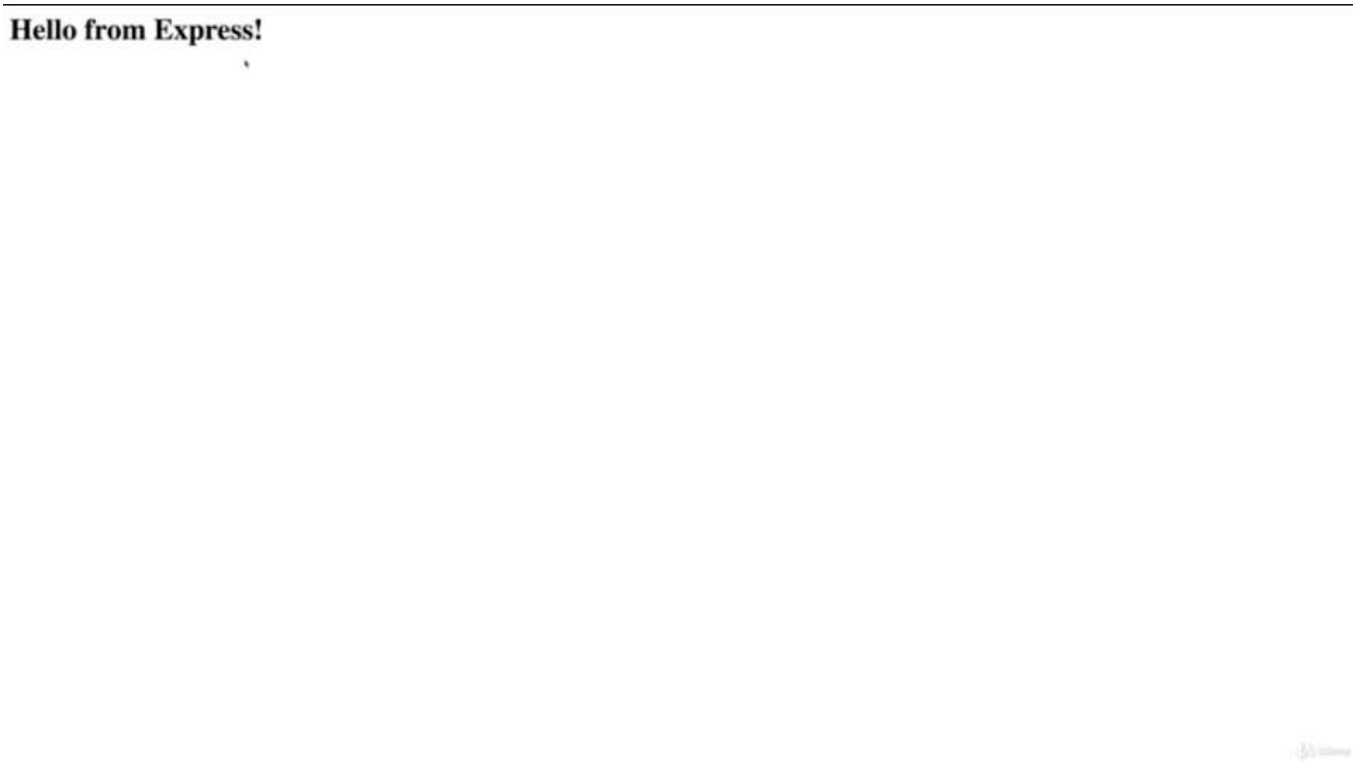
```
[nodemon] starting `node app.js`
In the middleware!
In the middleware!
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
In the middleware!
In another middleware!
```

```
//app.js

const http = require('http');

const express = require('express');

const app = express()

/**'use()' allows us to add a new middleware function
 * use() allows us this function to be executed for every incoming request
 *
 * 'next' is a function that will be passed to this function by Express.js
*/
app.use((req, res, next) => {
    /**this function you are receiving has to be executed
     * to allow the request to travel onto the next middleware */
    console.log('In the middleware')
    /**if we don't call 'next', we should actually send back a response
     * because otherwise the request can't continue
     * so it will never reach a place where we might send a response
     */
    next();
})

app.use((req, res, next) => {
    console.log('In another middleware')
    //...
})

const server = http.createServer(app);

server.listen(3000);
```

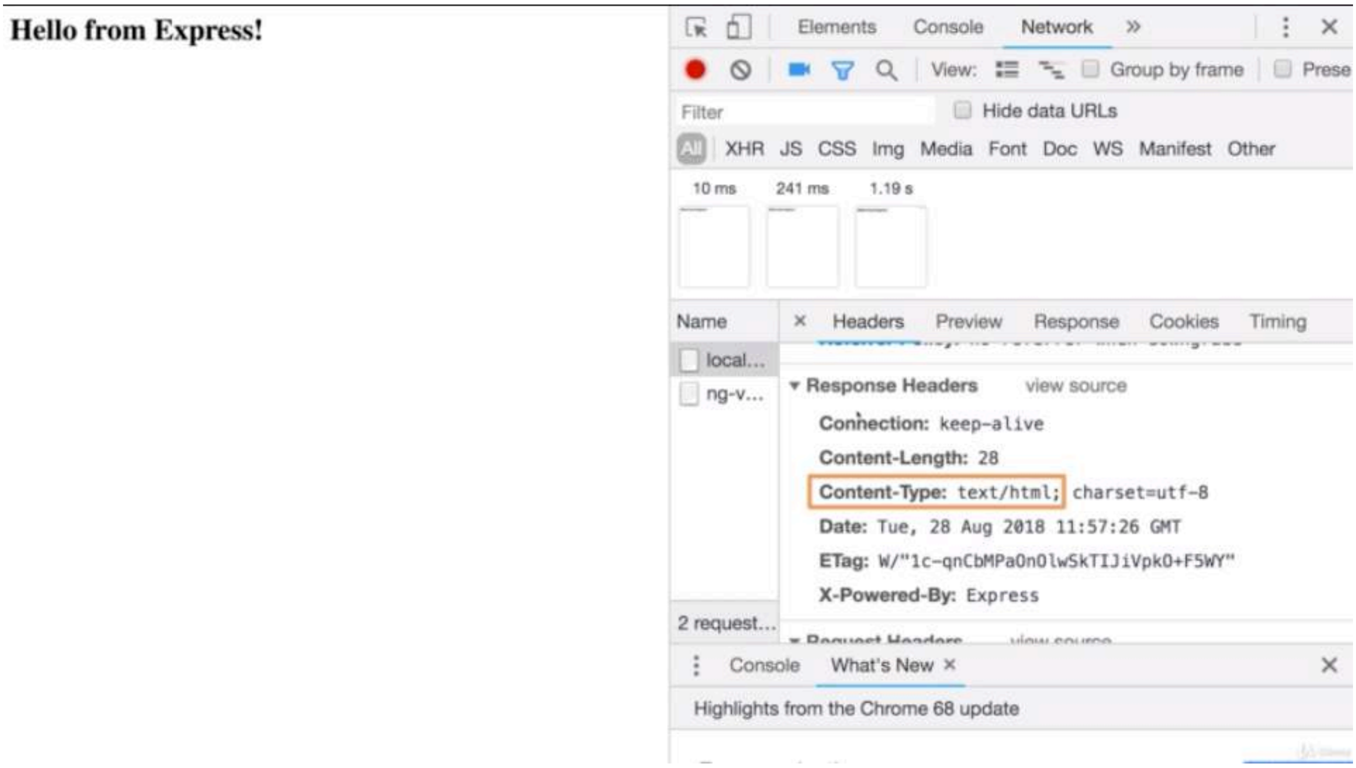
# * Chapter 60: How Middleware Works

1. update
- app.js

![](images/60-how-middleware-works-1.png)
![](images/60-how-middleware-works-2.png)



Hello from Express!



**Hello from Express!**

▾ Response Headers

Connection: keep-alive
Content-Length: 28
Content-Type: text/html; charset=utf-8
Date: Tue, 28 Aug 2018 11:57:26 GMT
ETag: W/"1c-qnCbMPaOnOlwSkTIJiVpkO+F5WY"
X-Powered-By: Express

- under header, the content-type is automatically set to text/html here. this is another feature provided by Express.

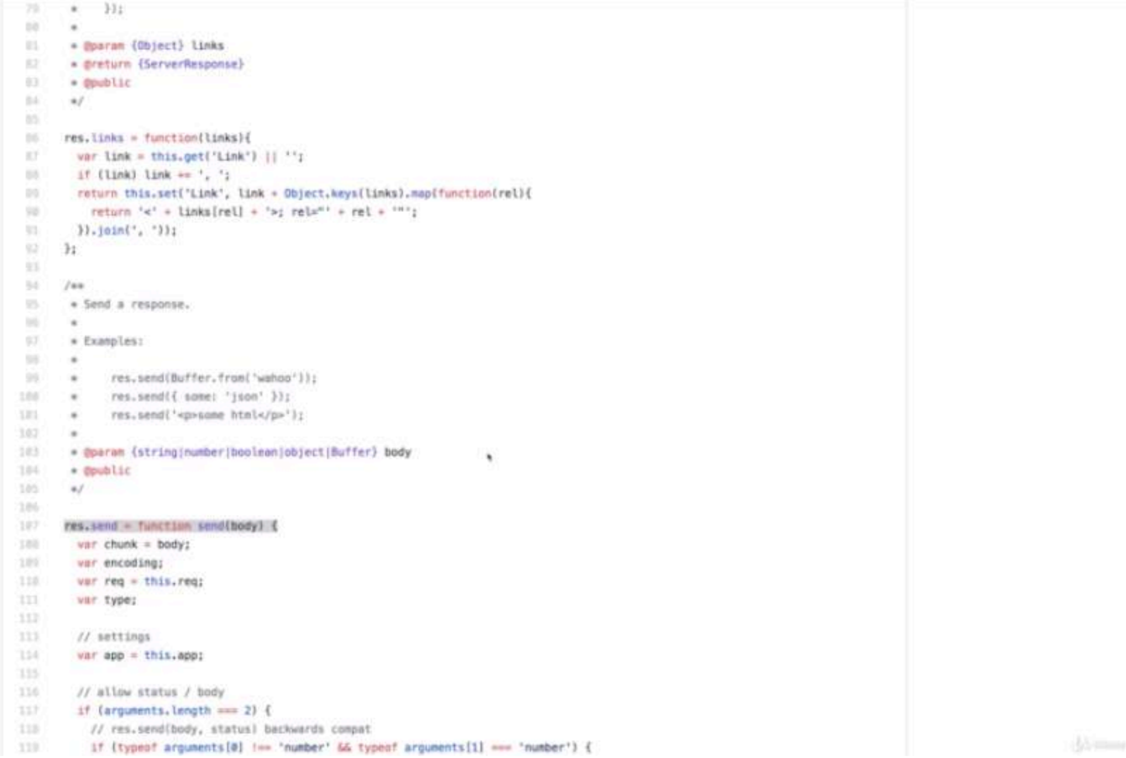
```
const http = require('http');

const express = require('express');

const app = express()

app.use((req, res, next) => {
    console.log('In the middleware')
    next();
})

app.use((req, res, next) => {
    console.log('In another middleware')
    /**'send()' allows us to send well a response
     * this allows us to attach a body which is of type any.
     *
     * this is particularly easier once we start sending back real files or the content of files
    */
    res.send('<h1>Hello from Express!</h1>')
})

const server = http.createServer(app);

server.listen(3000);
```

# * Chapter 61: Express.js - Looking Behind The Scenes

1. update

- app.js

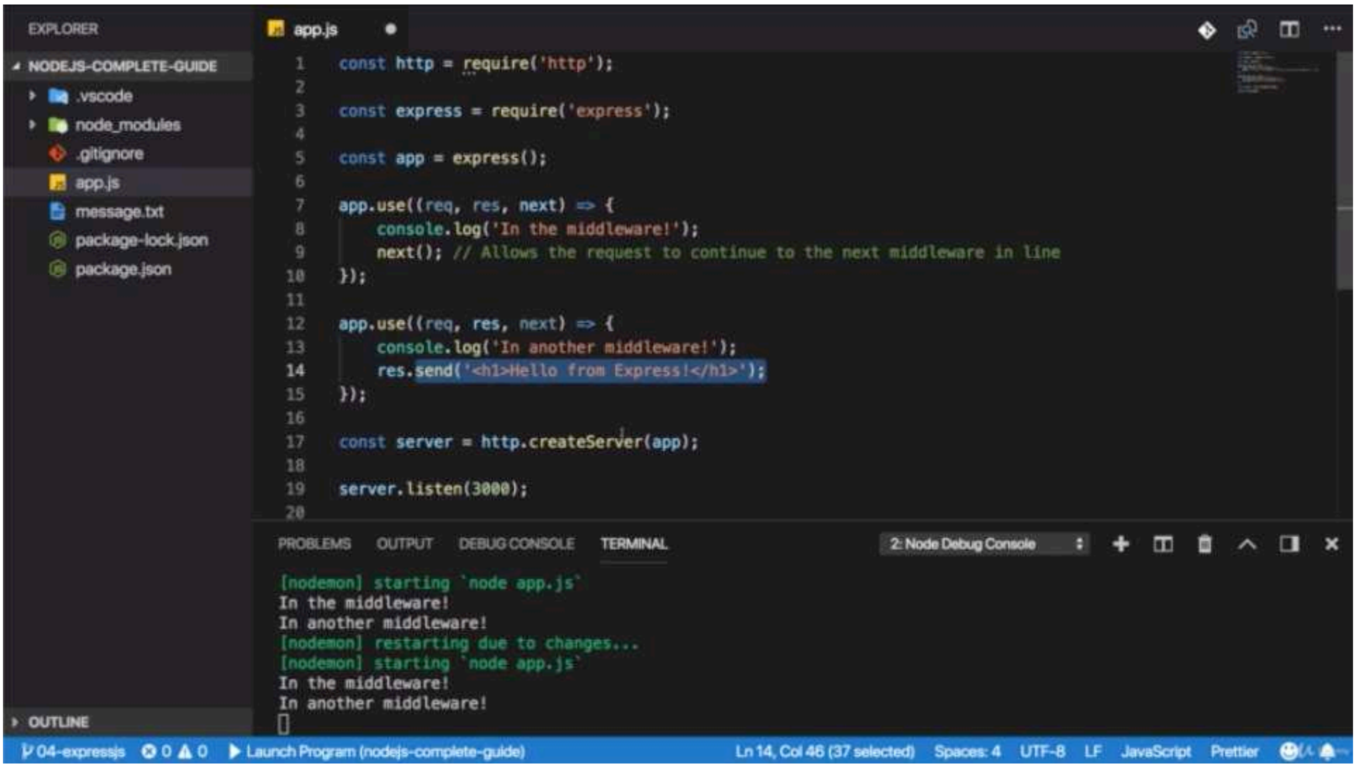
![](images/61-express-js-looking-behind-the-scenes-1.png)



```
 *   });
 *
 * @param {Object} links
 * @return {ServerResponse}
 * @public
 */

res.links = function(links){
  var link = this.get('Link') || '';
  if (link) link += ', ';
  return this.set('Link', link + Object.keys(links).map(function(rel){
    return '<' + links[rel] + '>; rel="' + rel + '"';
  }).join(', '));
};

/**
 * Send a response.
 *
 * Examples:
 *
 *     res.send(Buffer.from('wahoo'));
 *     res.send({ some: 'json' });
 *     res.send('<p>some html</p>');
 *
 * @param {string|number|boolean|object|Buffer} body
 * @public
 */

res.send = function send(body) {
  var chunk = body;
  var encoding;
  var req = this.req;
  var type;

  // settings
  var app = this.app;

  // allow status / body
  if (arguments.length === 2) {
    // res.send(body, status) backwards compat
    if (typeof arguments[0] !== 'number' && typeof arguments[1] === 'number') {
```

- you will see how the send function.

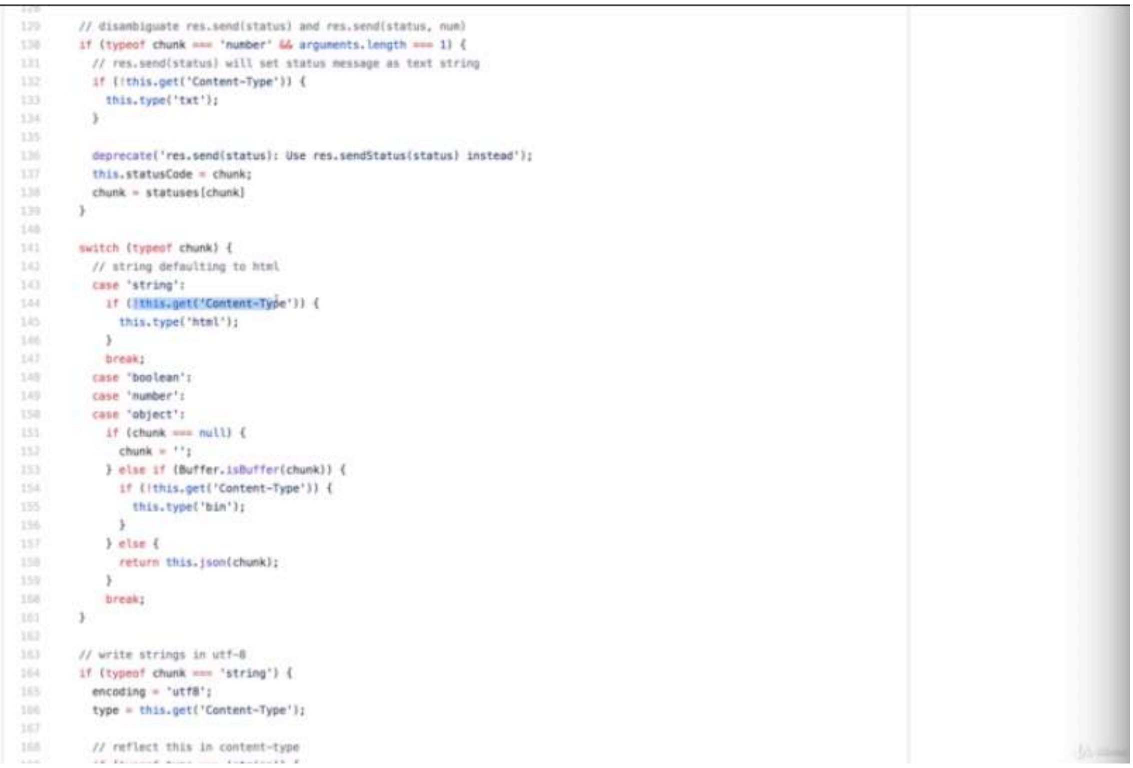
![](images/61-express-js-looking-behind-the-scenes-2.png)



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

const express = require('express');

const app = express();

app.use((req, res, next) => {
    console.log('In the middleware!');
    next(); // Allows the request to continue to the next middleware in line
});

app.use((req, res, next) => {
    console.log('In another middleware!');
    res.send('<h1>Hello from Express!</h1>');
});

const server = http.createServer(app);

server.listen(3000);
```

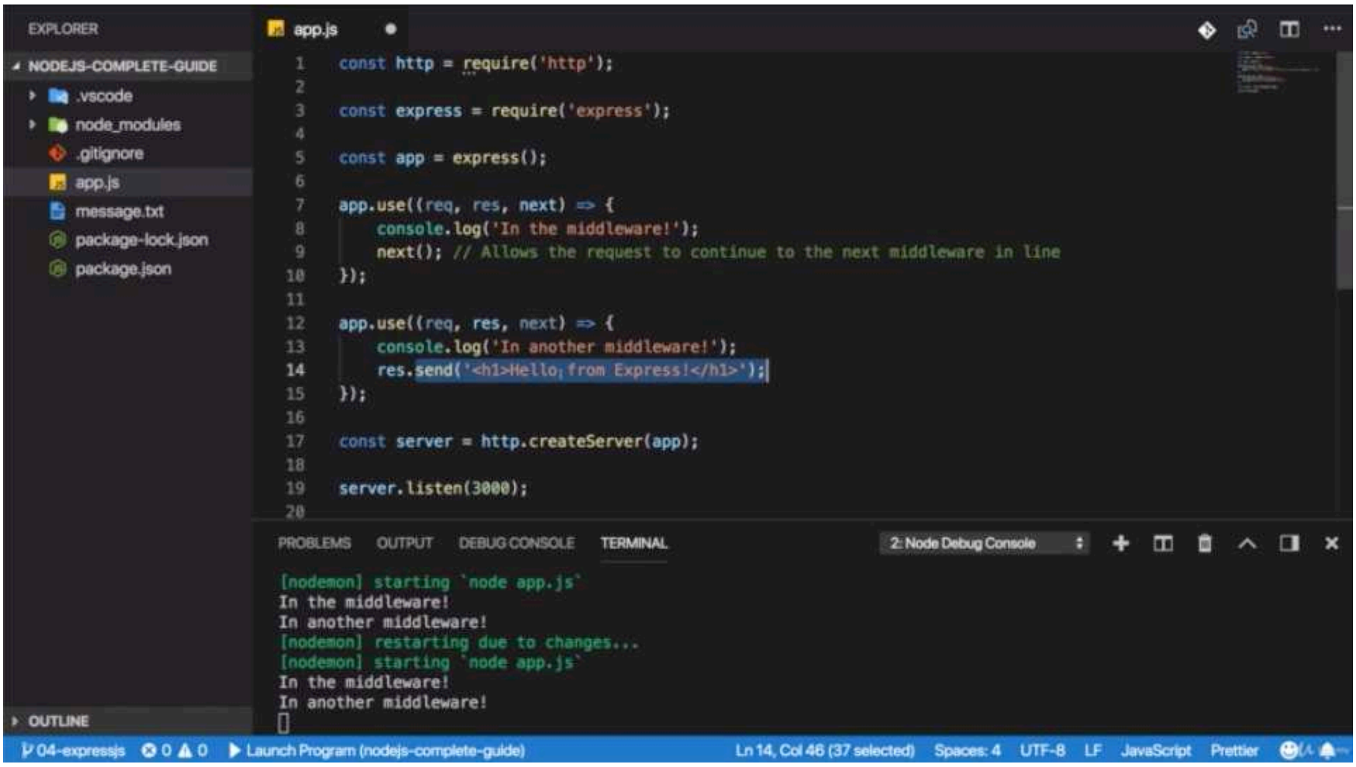
```
[nodemon] starting `node app.js`
In the middleware!
In another middleware!
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
In the middleware!
In another middleware!
```

- so basically a function we are calling here, how this is defined internally and this helps us understand
- by the way, this is always a great technique if you wanna see what something does behind the scene and if you need to do something yourself, set someHeader or if that is done for you and we had that default header of text/html. so let's see what send does internally.
![](images/61-express-js-looking-behind-the-scenes-3.png)



```
  // disambiguate res.send(status) and res.send(status, num)
  if (typeof chunk === 'number' && arguments.length === 1) {
    // res.send(status) will set status message as text string
    if (!this.get('Content-Type')) {
      this.type('txt');
    }

    deprecate('res.send(status): Use res.sendStatus(status) instead');
    this.statusCode = chunk;
    chunk = statuses[chunk]
  }

  switch (typeof chunk) {
    // string defaulting to html
    case 'string':
      if (!this.get('Content-Type')) {
        this.type('html');
      }
      break;
    case 'boolean':
    case 'number':
    case 'object':
      if (chunk === null) {
        chunk = '';
      } else if (Buffer.isBuffer(chunk)) {
        if (!this.get('Content-Type')) {
          this.type('bin');
        }
      } else {
        return this.json(chunk);
      }
      break;
  }

  // write strings in utf-8
  if (typeof chunk === 'string') {
    encoding = 'utf8';
    type = this.get('Content-Type');

    // reflect this in content-type
```

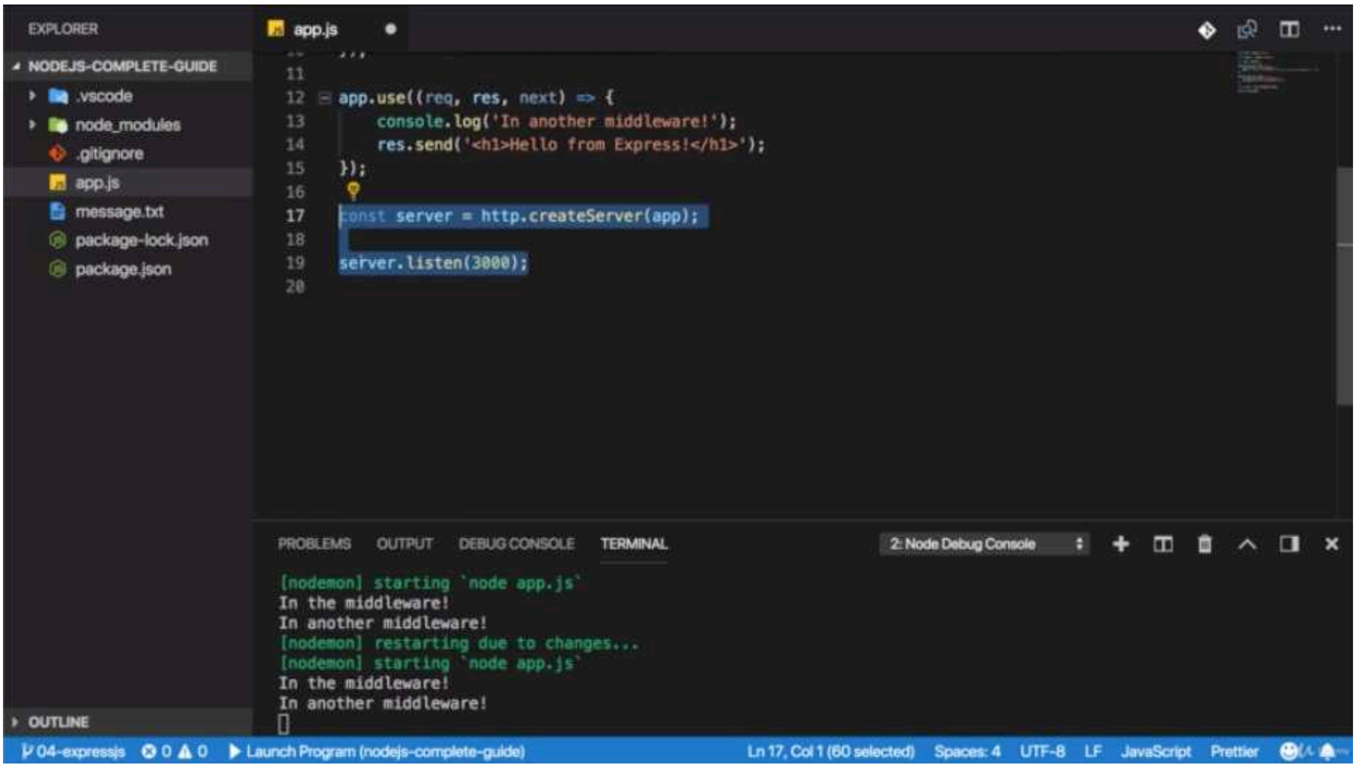
- it basically analyzes what kind of data you sending.
![](images/61-express-js-looking-behind-the-scenes-4.png)



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

const express = require('express');

const app = express();

app.use((req, res, next) => {
    console.log('In the middleware!');
    next(); // Allows the request to continue to the next middleware in line
});

app.use((req, res, next) => {
    console.log('In another middleware!');
    res.send('<h1>Hello from Express!</h1>');
});

const server = http.createServer(app);

server.listen(3000);
```

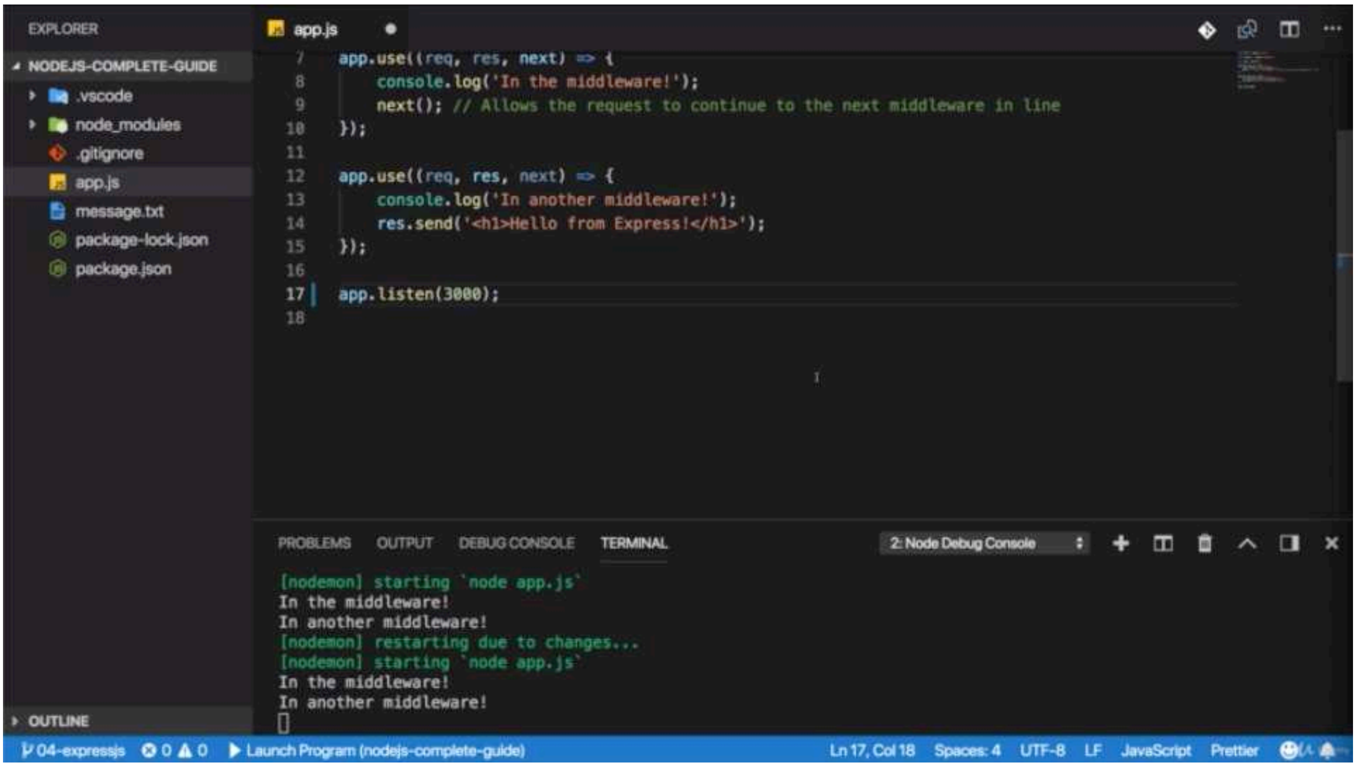
```
[nodemon] starting `node app.js`
In the middleware!
In another middleware!
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
In the middleware!
In another middleware!
```

- if it’s a string data, so some text as we are doing it here, in this case, it sets content-type to html. but only if we haven’t set it yet. so it checks if the content type header is not present yet in which case it sets it, otherwise it would leave our default.
- if we have other values like a number, a boolean and so on, it would set it to binary or json data.
![](images/61-express-js-looking-behind-the-scenes-5.png)
![](images/61-express-js-looking-behind-the-scenes-6.png)



```
app.use((req, res, next) => {
    console.log('In another middleware!');
    res.send('<h1>Hello from Express!</h1>');
});

const server = http.createServer(app);

server.listen(3000);
```

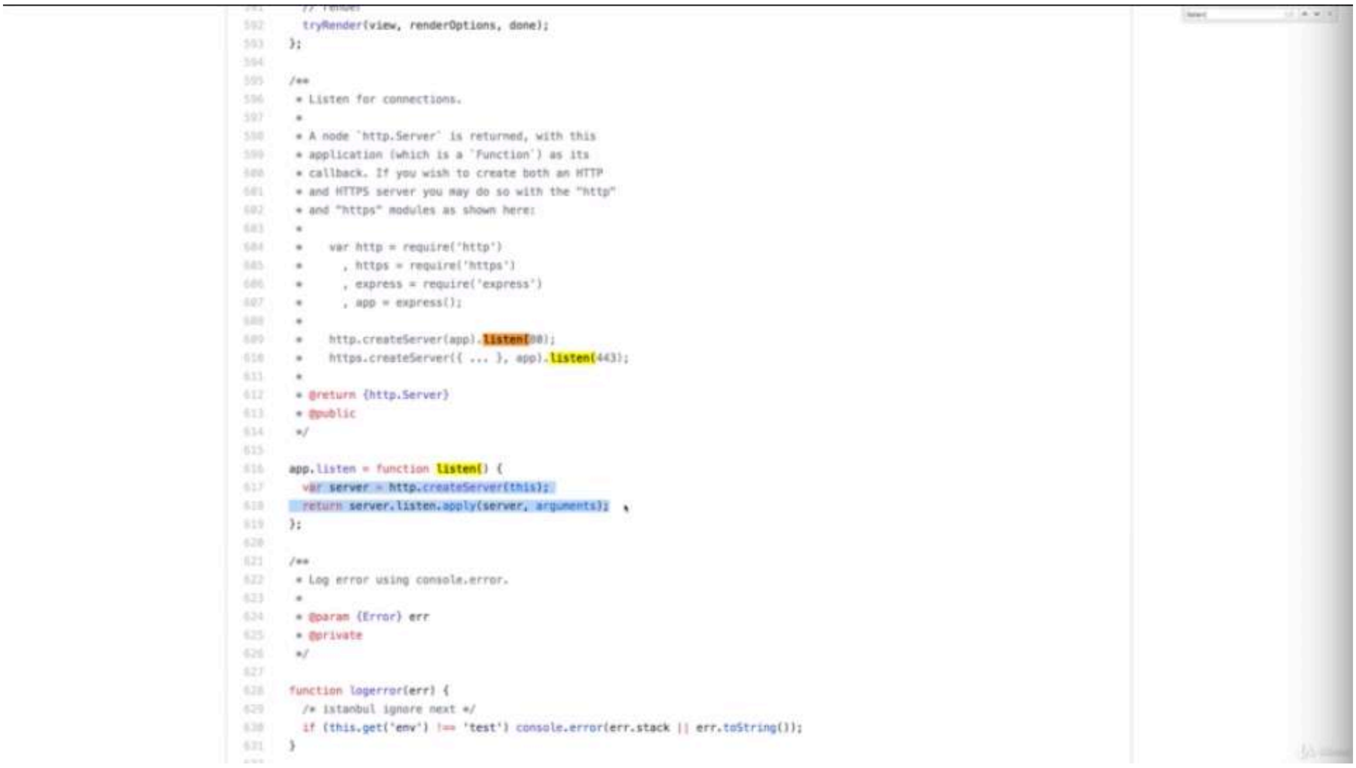
```
[nodemon] starting `node app.js`
In the middleware!
In another middleware!
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
In the middleware!
In another middleware!
```



```
app.use((req, res, next) => {
    console.log('In the middleware!');
    next(); // Allows the request to continue to the next middleware in line
});

app.use((req, res, next) => {
    console.log('In another middleware!');
    res.send('<h1>Hello from Express!</h1>');
});

app.listen(3000);
```

```
[nodemon] starting `node app.js`
In the middleware!
In another middleware!
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
In the middleware!
In another middleware!
```

- now one other interesting thing to see is that we can actually also shorten this code here where we set up the server.
- app.listen() do both these things for us, something we can see in the official code.
![](images/61-express-js-looking-behind-the-scenes-7.png)



```
  // render
  tryRender(view, renderOptions, done);
};

/**
 * Listen for connections.
 *
 * A node `http.Server` is returned, with this
 * application (which is a `Function`) as its
 * callback. If you wish to create both an HTTP
 * and HTTPS server you may do so with the "http"
 * and "https" modules as shown here:
 *
 *    var http = require('http')
 *      , https = require('https')
 *      , express = require('express')
 *      , app = express();
 *
 *    http.createServer(app).listen(80);
 *    https.createServer({ ... }, app).listen(443);
 *
 * @return {http.Server}
 * @public
 */

app.listen = function listen() {
  var server = http.createServer(this);
  return server.listen.apply(server, arguments);
};

/**
 * Log error using console.error.
 *
 * @param {Error} err
 * @private
 */

function logerror(err) {
  /* istanbul ignore next */
  if (this.get('env') !== 'test') console.error(err.stack || err.toString());
}
```

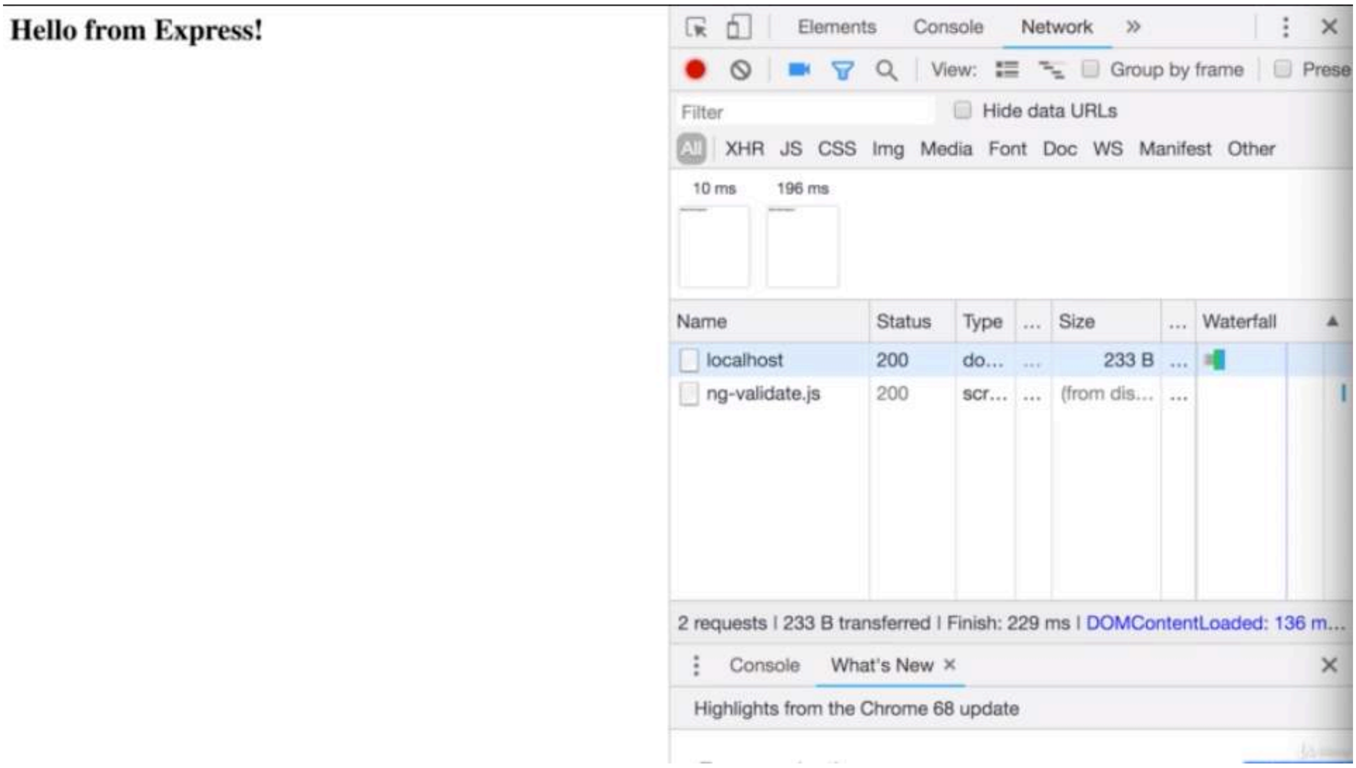
- listen function in the end does the 2 things we did before.

```
//app.js

const http = require('http');

const express = require('express');

const app = express()

app.use((req, res, next) => {
    console.log('In the middleware')
    next();
})

app.use((req, res, next) => {
    console.log('In another middleware')
    res.send('<h1>Hello from Express!</h1>')
})

app.listen(3000);
```

# * Chapter 62: Handling Different Routes

1. update
- app.js

![](images/62-handling-different-routes-1.png)
![](images/62-handling-different-routes-2.png)
![](images/62-handling-different-routes-3.png)

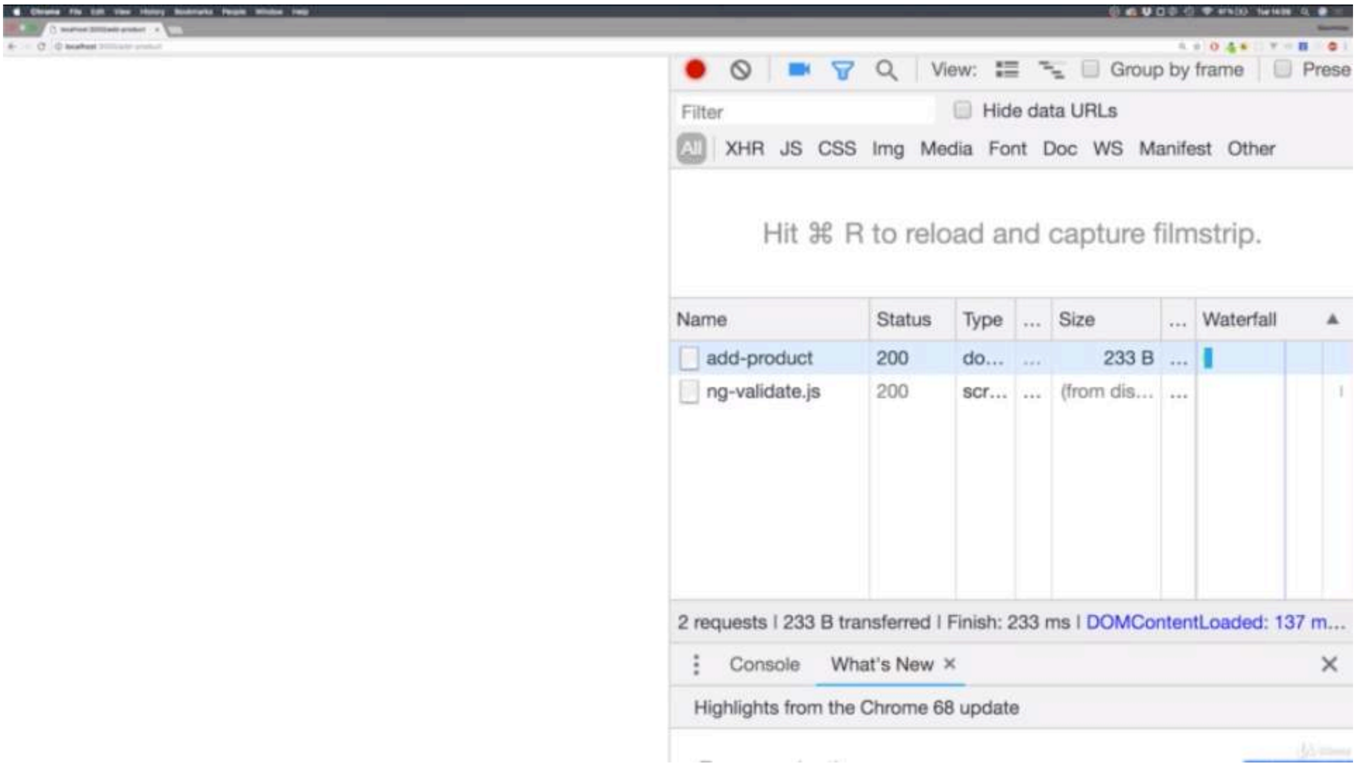


Hello from Express!

| Name | Status | Type | ... | Size | ... | Waterfall |
|---|---|---|---|---|---|---|
| localhost | 200 | do... | ... | 233 B | ... | |
| ng-validate.js | 200 | scr... | ... | (from dis... | ... | |

2 requests | 233 B transferred | Finish: 229 ms | DOMContentLoaded: 136 m...

Highlights from the Chrome 68 update

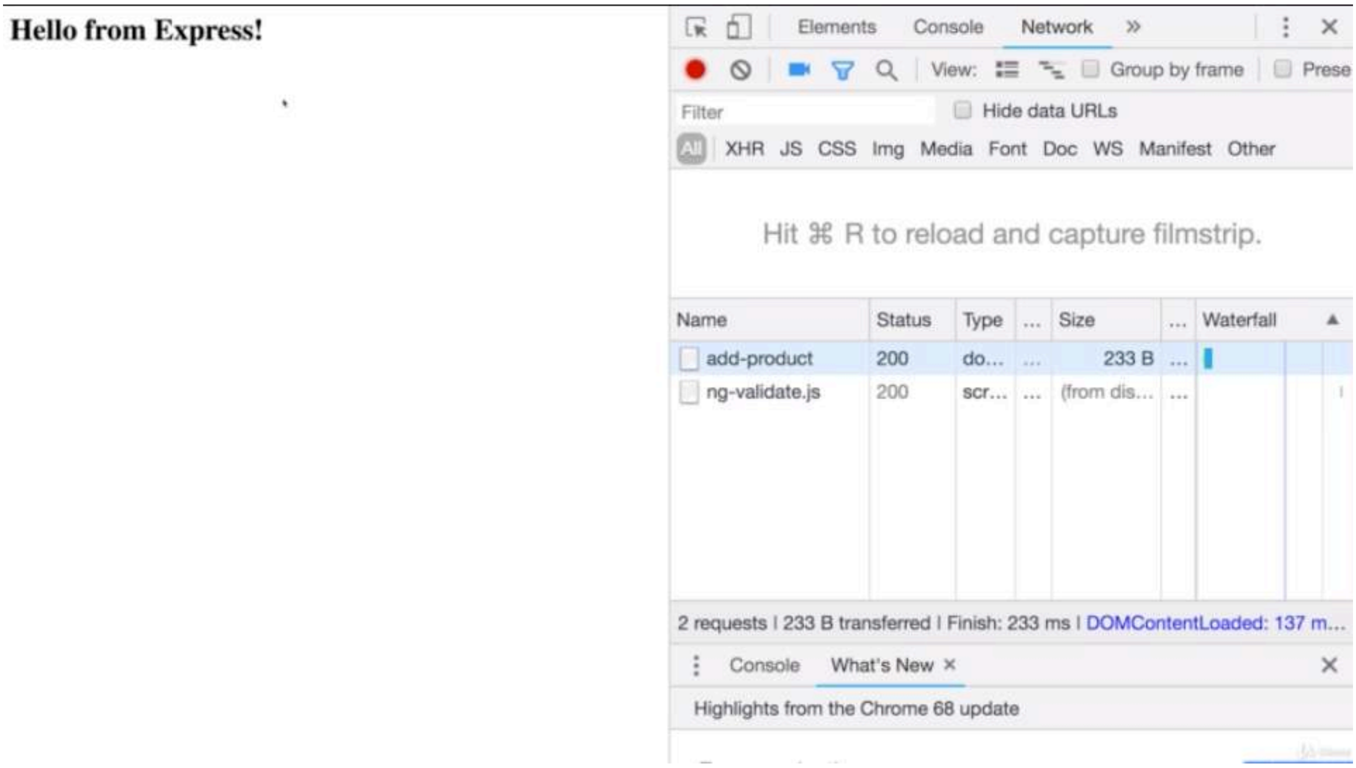


Hit ⌘ R to reload and capture filmstrip.

| Name | Status | Type | ... | Size | ... | Waterfall |
|---|---|---|---|---|---|---|
| add-product | 200 | do... | ... | 233 B | ... | |
| ng-validate.js | 200 | scr... | ... | (from dis... | ... | |

2 requests | 233 B transferred | Finish: 233 ms | DOMContentLoaded: 137 m...

Highlights from the Chrome 68 update


Hello from Express!

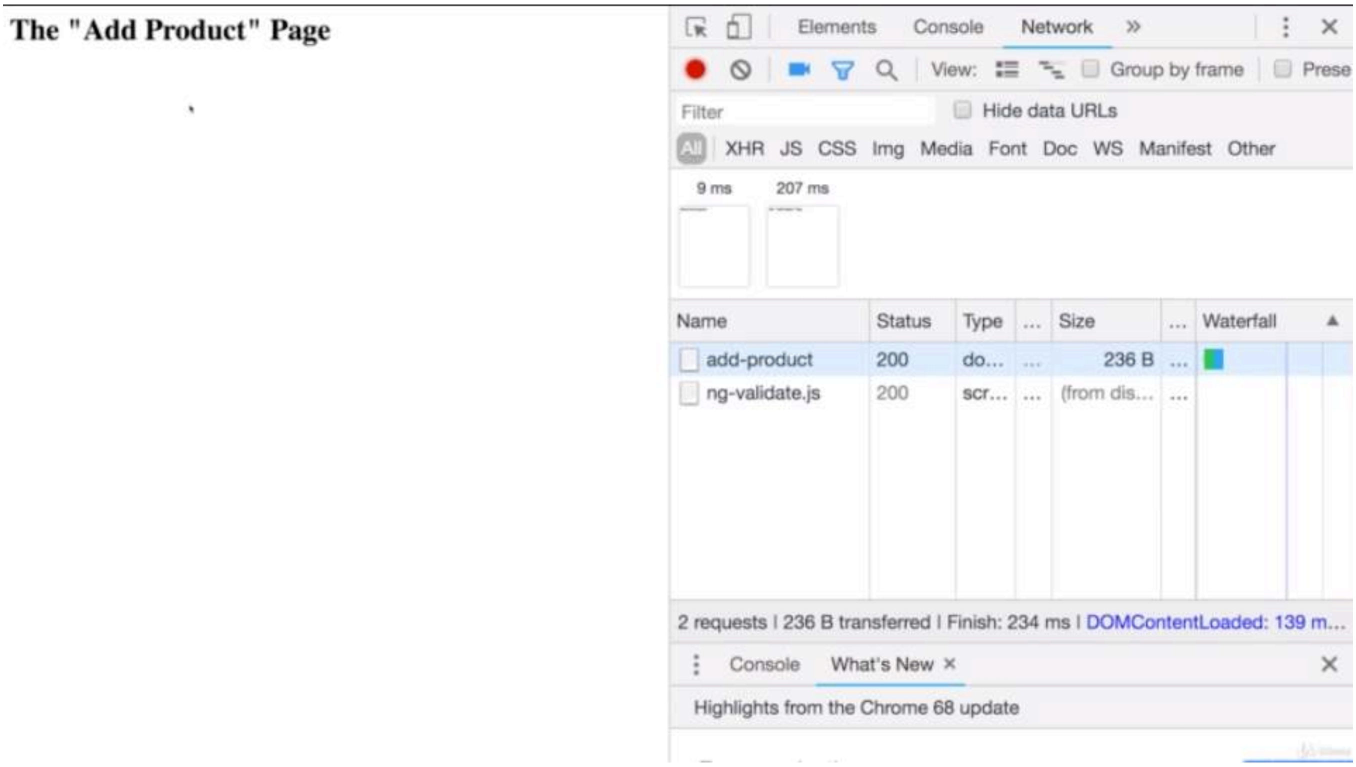
Hit ⌘ R to reload and capture filmstrip.

| Name | Status | Type | ... | Size | ... | Waterfall |
|---|---|---|---|---|---|---|
| add-product | 200 | do... | ... | 233 B | ... | |
| ng-validate.js | 200 | scr... | ... | (from dis... | ... | |

2 requests | 233 B transferred | Finish: 233 ms | DOMContentLoaded: 137 m...

Highlights from the Chrome 68 update

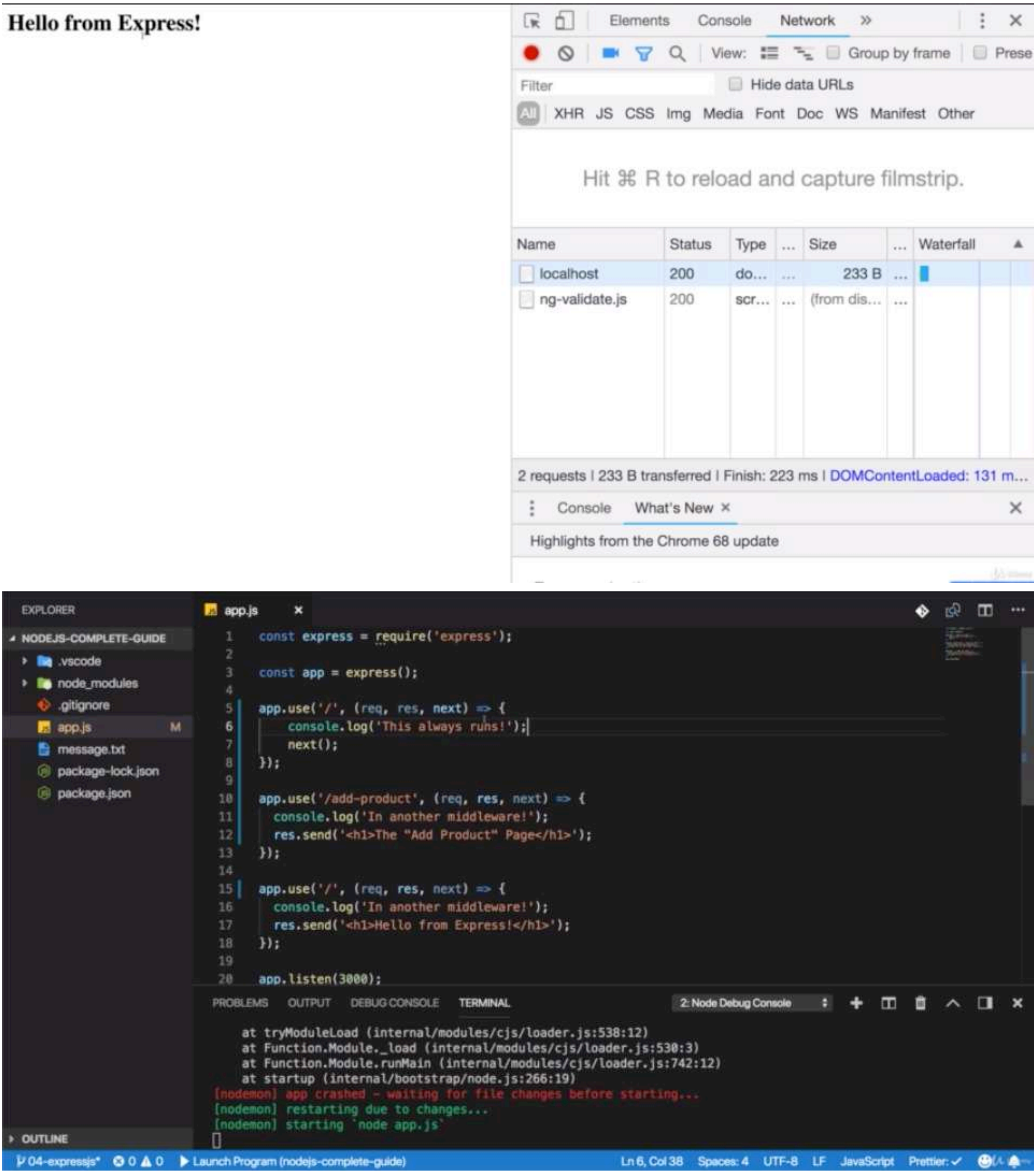
- if we type not ‘/‘ but ‘/add-product’, we still see ‘Hello from Express’ and we still see i’m in another middleware, so this middleware gets executed for both slash and add-product because this ‘/' doesn’t mean that the full path. so the part after the domain has to be a slash but that it has to start with that.
![](images/62-handling-different-routes-4.png)
![](images/62-handling-different-routes-5.png)
![](images/62-handling-different-routes-6.png)
![](images/62-handling-different-routes-7.png)
![](images/62-handling-different-routes-8.png)
![](images/62-handling-different-routes-9.png)


The "Add Product" Page

| Name | Status | Type | ... | Size | ... | Waterfall |
|---|---|---|---|---|---|---|
| add-product | 200 | do... | ... | 236 B | ... | |
| ng-validate.js | 200 | scr... | ... | (from dis... | ... | |

2 requests | 236 B transferred | Finish: 234 ms | DOMContentLoaded: 139 m...

Highlights from the Chrome 68 update



# Hello from Express!

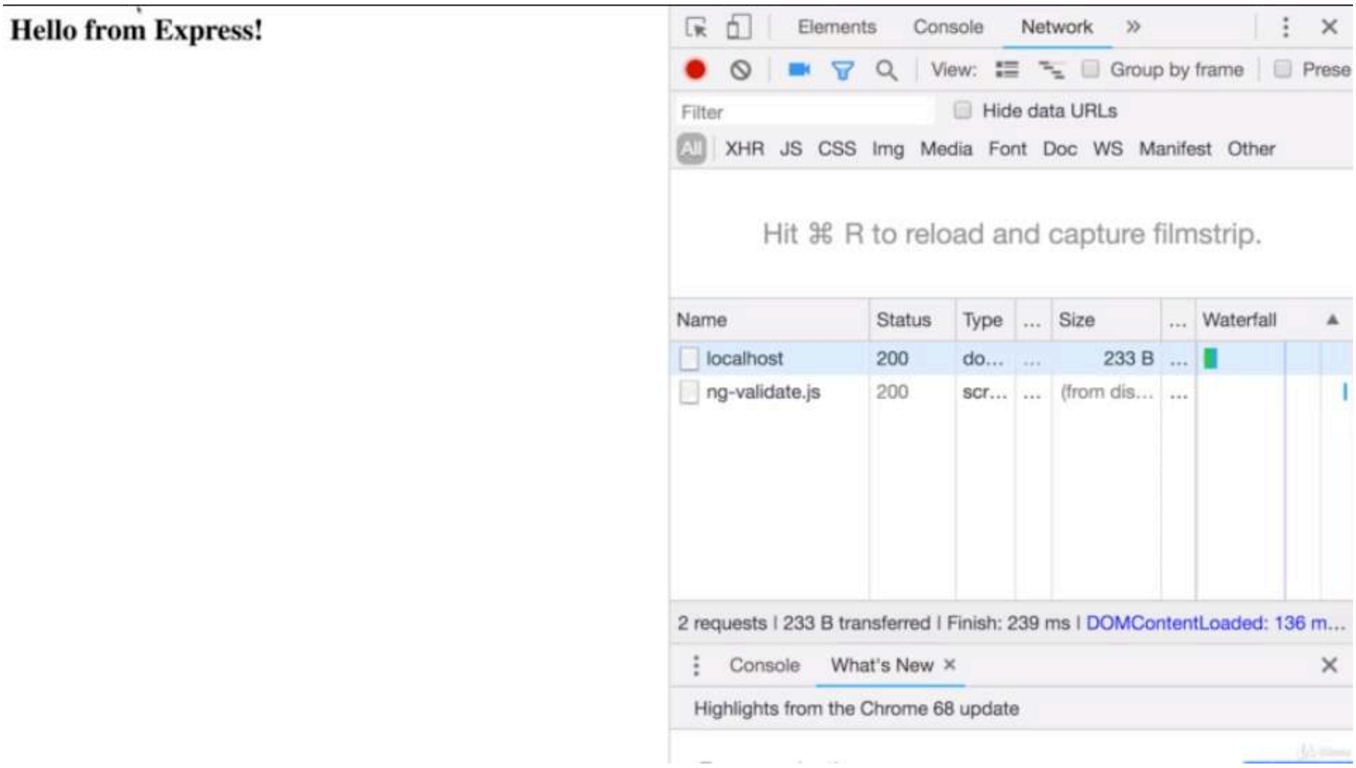
| Name | Status | Type | ... | Size | ... | Waterfall |
|---|---|---|---|---|---|---|
| localhost | 200 | do... | ... | 233 B | ... | |
| ng-validate.js | 200 | scr... | ... | (from dis... | ... | |

2 requests | 233 B transferred | Finish: 223 ms | DOMContentLoaded: 131 m...

app.js

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

const app = express();

app.use('/', (req, res, next) => {
    console.log('This always runs!');
    next();
});

app.use('/add-product', (req, res, next) => {
  console.log('In another middleware!');
  res.send('<h1>The "Add Product" Page</h1>');
});

app.use('/', (req, res, next) => {
  console.log('In another middleware!');
  res.send('<h1>Hello from Express!</h1>');
});

app.listen(3000);
```

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



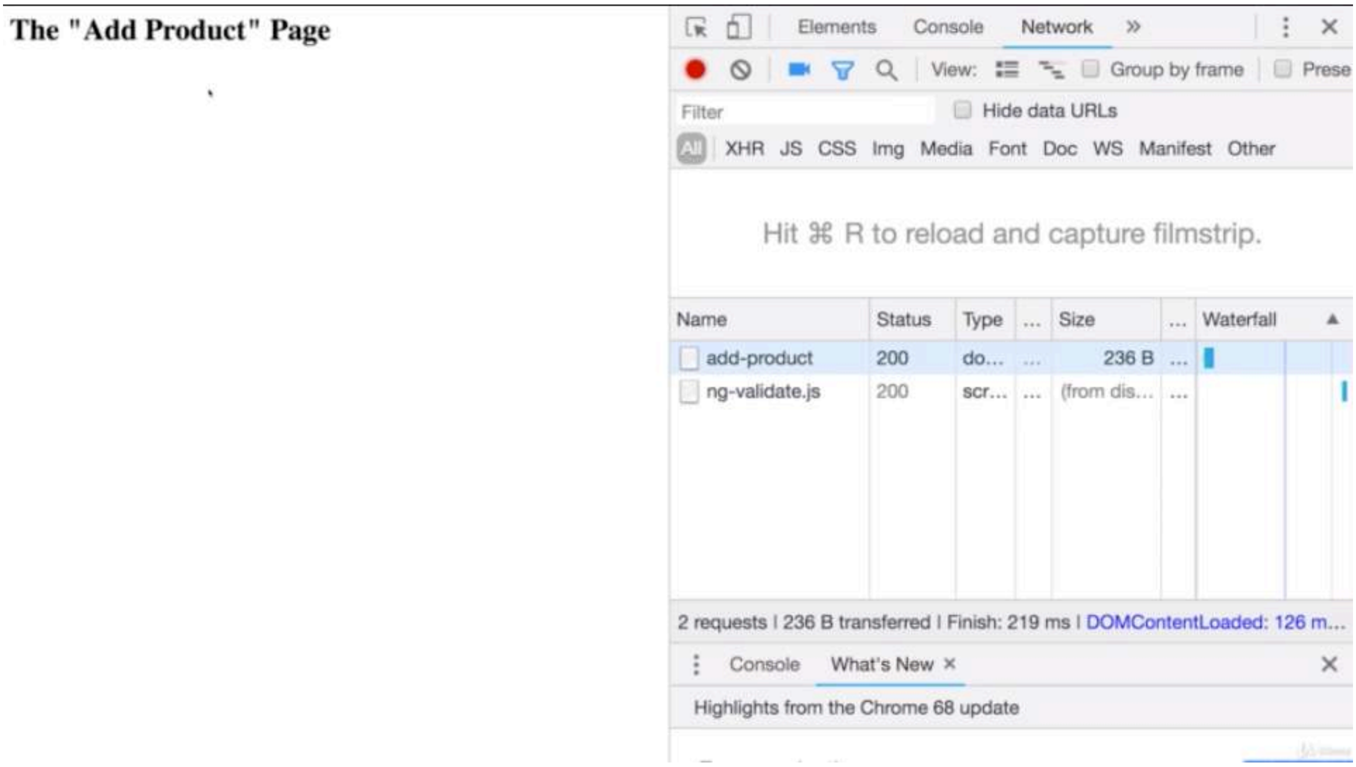
Hello from Express!

Hit ⌘ R to reload and capture filmstrip.

| Name | Status | Type | ... | Size | ... | Waterfall |
|---|---|---|---|---|---|---|
| localhost | 200 | do... | ... | 233 B | ... | |
| ng-validate.js | 200 | scr... | ... | (from dis... | ... | |

2 requests | 233 B transferred | Finish: 239 ms | DOMContentLoaded: 136 m...

Highlights from the Chrome 68 update



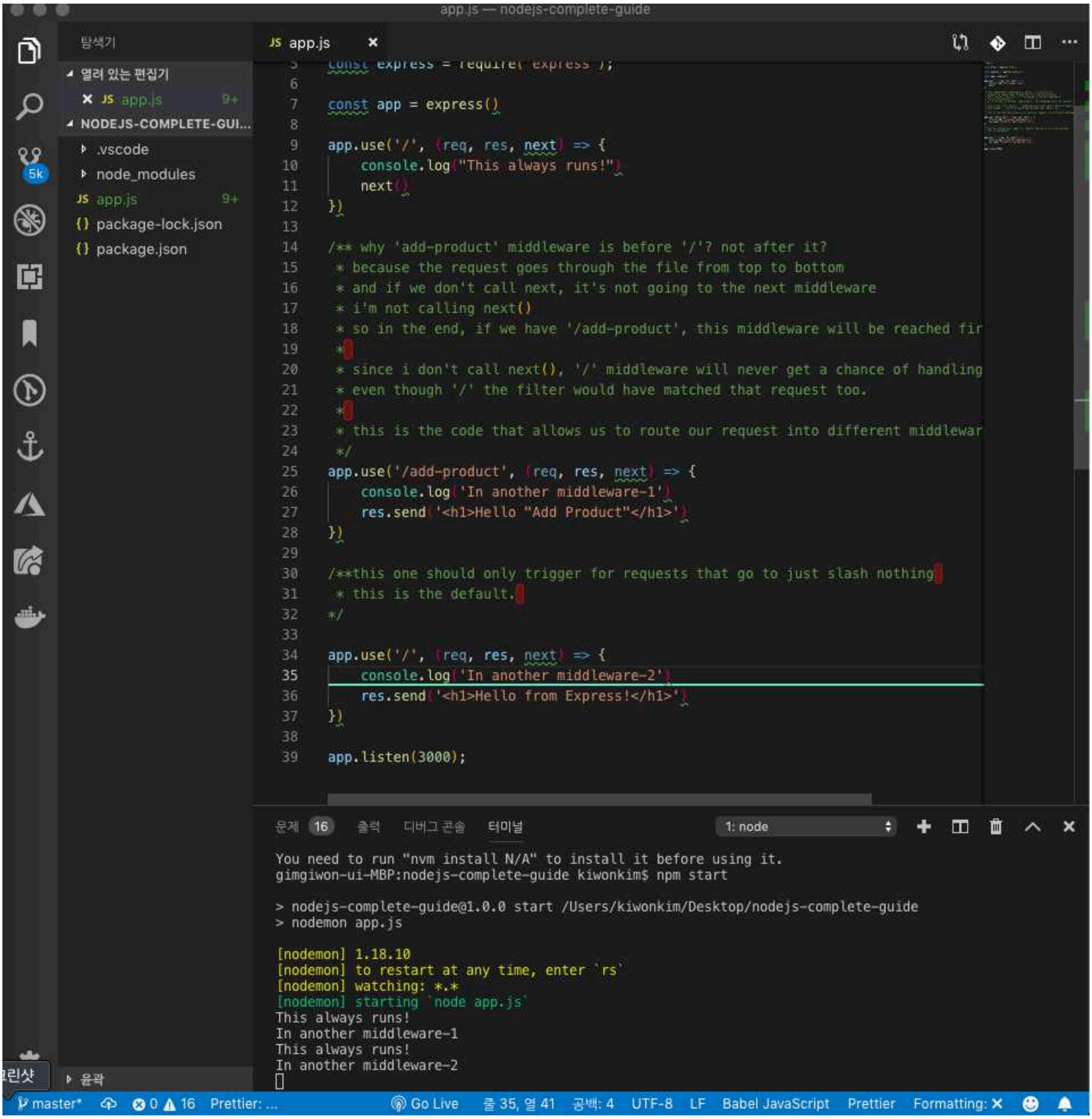
The "Add Product" Page

Hit ⌘ R to reload and capture filmstrip.

| Name | Status | Type | ... | Size | ... | Waterfall |
|---|---|---|---|---|---|---|
| add-product | 200 | do... | ... | 236 B | ... | |
| ng-validate.js | 200 | scr... | ... | (from dis... | ... | |

2 requests | 236 B transferred | Finish: 219 ms | DOMContentLoaded: 126 m...

Highlights from the Chrome 68 update



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

const app = express()

app.use('/', (req, res, next) => {
    console.log("This always runs!")
    next()
})

/** why 'add-product' middleware is before '/'? not after it?
 * because the request goes through the file from top to bottom
 * and if we don't call next, it's not going to the next middleware
 * i'm not calling next()
 * so in the end, if we have '/add-product', this middleware will be reached fir
 *
 * since i don't call next(), '/' middleware will never get a chance of handling
 * even though '/' the filter would have matched that request too.
 *
 * this is the code that allows us to route our request into different middlewar
 */
app.use('/add-product', (req, res, next) => {
    console.log('In another middleware-1')
    res.send('<h1>Hello "Add Product"</h1>')
})

/**this one should only trigger for requests that go to just slash nothing
 * this is the default.
*/

app.use('/', (req, res, next) => {
    console.log('In another middleware-2')
    res.send('<h1>Hello from Express!</h1>')
})

app.listen(3000);
```

```
You need to run "nvm install N/A" to install it before using it.
gimgiwon-ui-MBP:nodejs-complete-guide kiwonkim$ npm start

> nodejs-complete-guide@1.0.0 start /Users/kiwonkim/Desktop/nodejs-complete-guide
> nodemon app.js

[nodemon] 1.18.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node app.js`
This always runs!
In another middleware-1
This always runs!
In another middleware-2
```

- we have this runs twice because this first app.use() always runs.

```
//app.js

const express = require('express');

const app = express()

/** why 'add-product' middleware is before '/'? not after it?
 * because the request goes through the file from top to bottom
 * and if we don't call next, it's not going to the next middleware
 * i'm not calling next()
 * so in the end, if we have '/add-product', this middleware will be reached first.
 *
 * since i don't call next(), '/' middleware will never get a chance of handling that
request
 * even though '/' the filter would have matched that request too.
 *
 * this is the code that allows us to route our request into different middleware.
 */
app.use('/add-product', (req, res, next) => {
    console.log('In another middleware-1')
    res.send('<h1>Hello "Add Product"</h1>')
})

/**this one should only trigger for requests that go to just slash nothing
 * this is the default.
*/

app.use('/', (req, res, next) => {
    console.log('In another middleware-2')
    res.send('<h1>Hello from Express!</h1>')
})

app.listen(3000);
```

# * Assignment: Time To Practice - Express.js

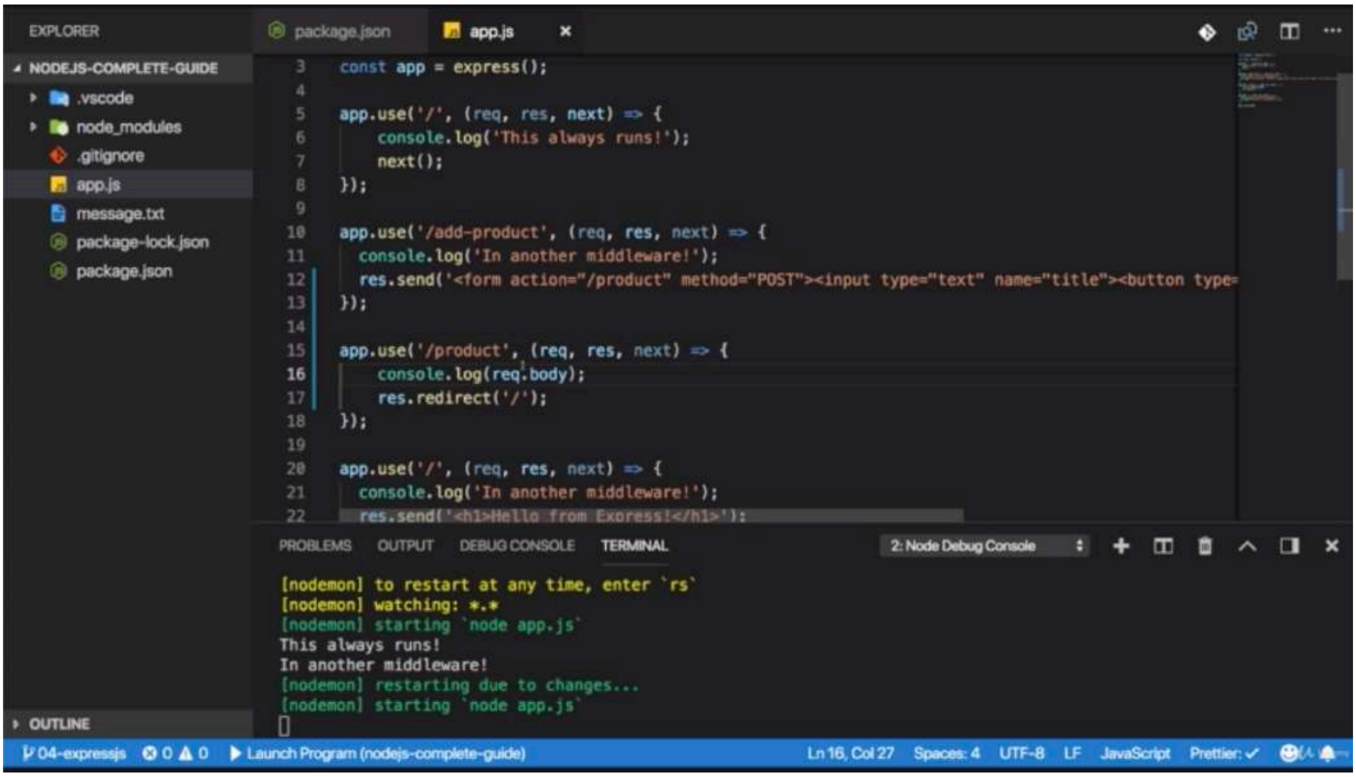
```
//app.js

const express = require('express');

const app = express();

// app.use((req, res, next) => {
//   console.log('First Middleware');
//   next();
// });

// app.use((req, res, next) => {
//   console.log('Second Middleware');
//   res.send('<p>Assignment solved (almost!)</p>');
// });

app.use('/users', (req, res, next) => {
    console.log('/users middleware');
    res.send('<p>The Middleware that handles just /users</p>');
});

app.use('/', (req, res, next) => {
    console.log('/ middleware');
    res.send('<p>The Middleware that handles just /</p>');
});


app.listen(3000);

```

```
//package.json

{
  "name": "nodejs-complete-guide",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "nodemon app.js"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.16.3"
  },
  "devDependencies": {
    "nodemon": "^1.18.3"
  }
}

```

# * Chapter 63: Parsing Incoming Requests

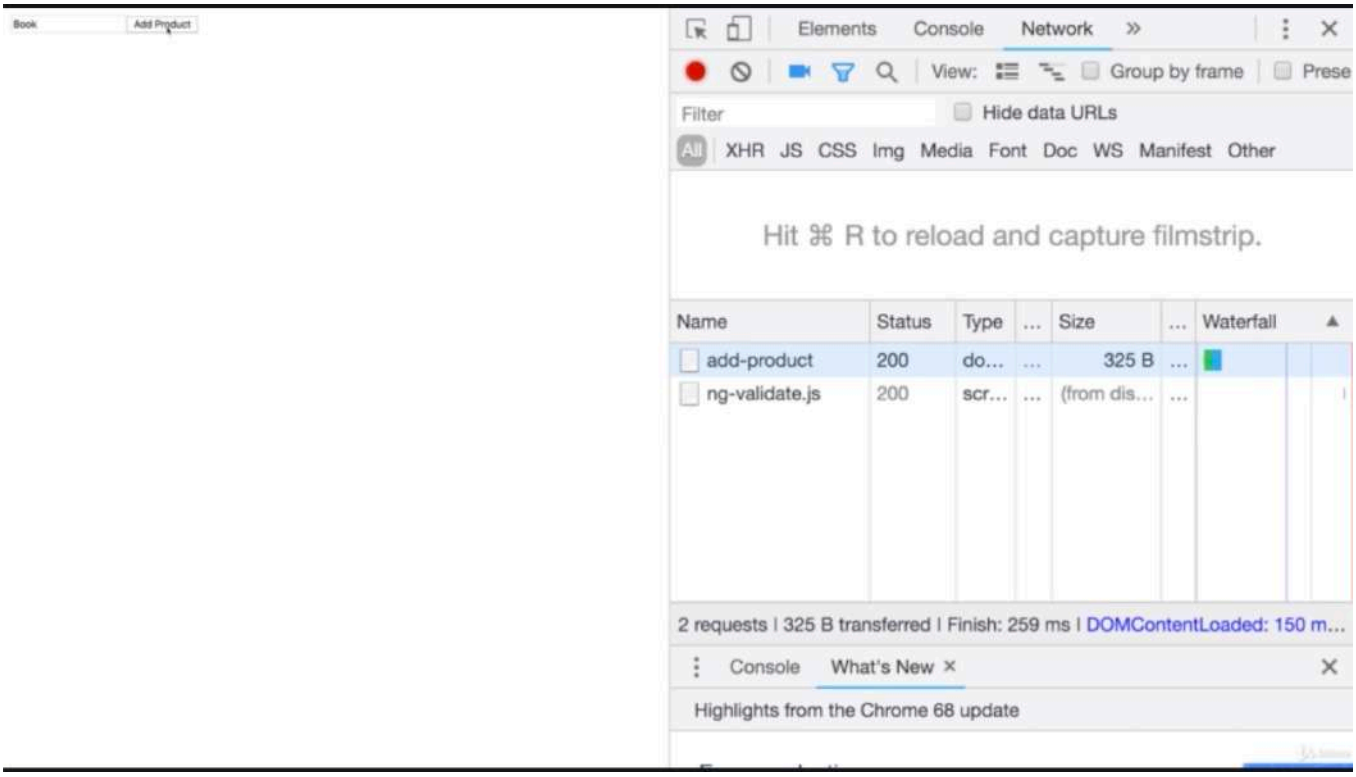
1. update
- app.js

![](images/63-parsing-incoming-requests-1.png)
![](images/63-parsing-incoming-requests-2.png)
![](images/63-parsing-incoming-requests-3.png)



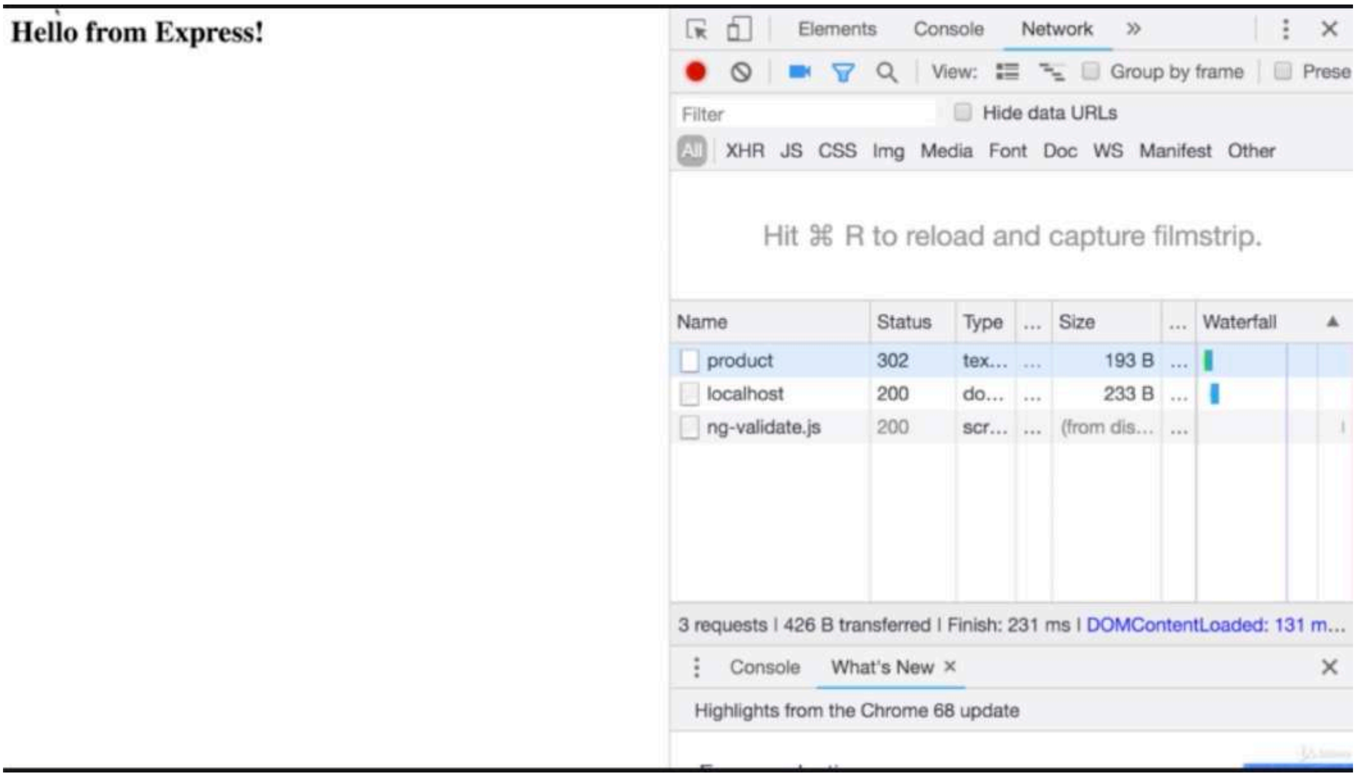
```
const app = express();

app.use('/', (req, res, next) => {
    console.log('This always runs!');
    next();
});

app.use('/add-product', (req, res, next) => {
  console.log('In another middleware!');
  res.send('<form action="/product" method="POST"><input type="text" name="title"><button type=
});

app.use('/product', (req, res, next) => {
    console.log(req.body);
    res.redirect('/');
});

app.use('/', (req, res, next) => {
  console.log('In another middleware!');
  res.send('<h1>Hello from Express!</h1>');
```

```
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node app.js`
This always runs!
In another middleware!
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
```



| Name | Status | Type | ... | Size | ... | Waterfall |
|---|---|---|---|---|---|---|
| add-product | 200 | do... | ... | 325 B | ... | |
| ng-validate.js | 200 | scr... | ... | (from dis... | ... | |

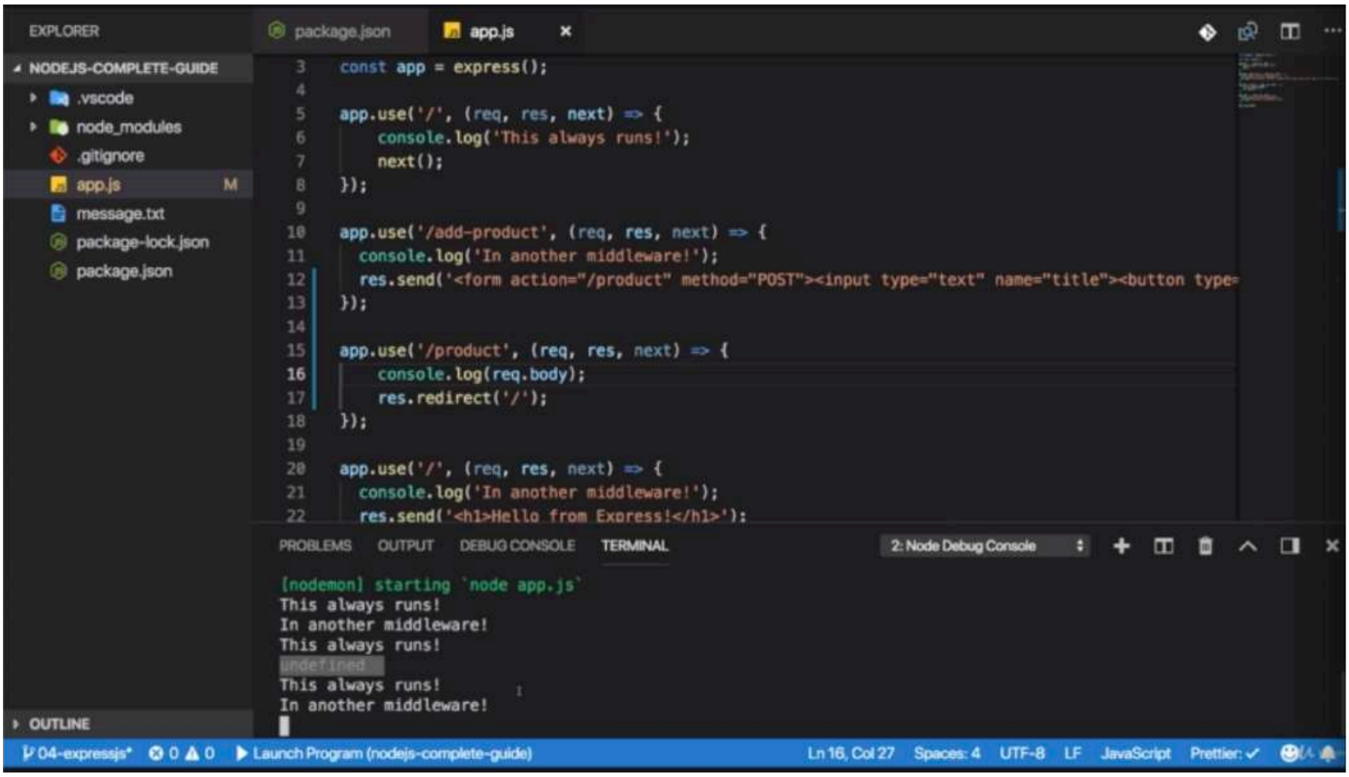
2 requests | 325 B transferred | Finish: 259 ms | DOMContentLoaded: 150 m...


Hello from Express!

| Name | Status | Type | ... | Size | ... | Waterfall |
|---|---|---|---|---|---|---|
| product | 302 | tex... | ... | 193 B | ... | |
| localhost | 200 | do... | ... | 233 B | ... | |
| ng-validate.js | 200 | scr... | ... | (from dis... | ... | |

3 requests | 426 B transferred | Finish: 231 ms | DOMContentLoaded: 131 m...

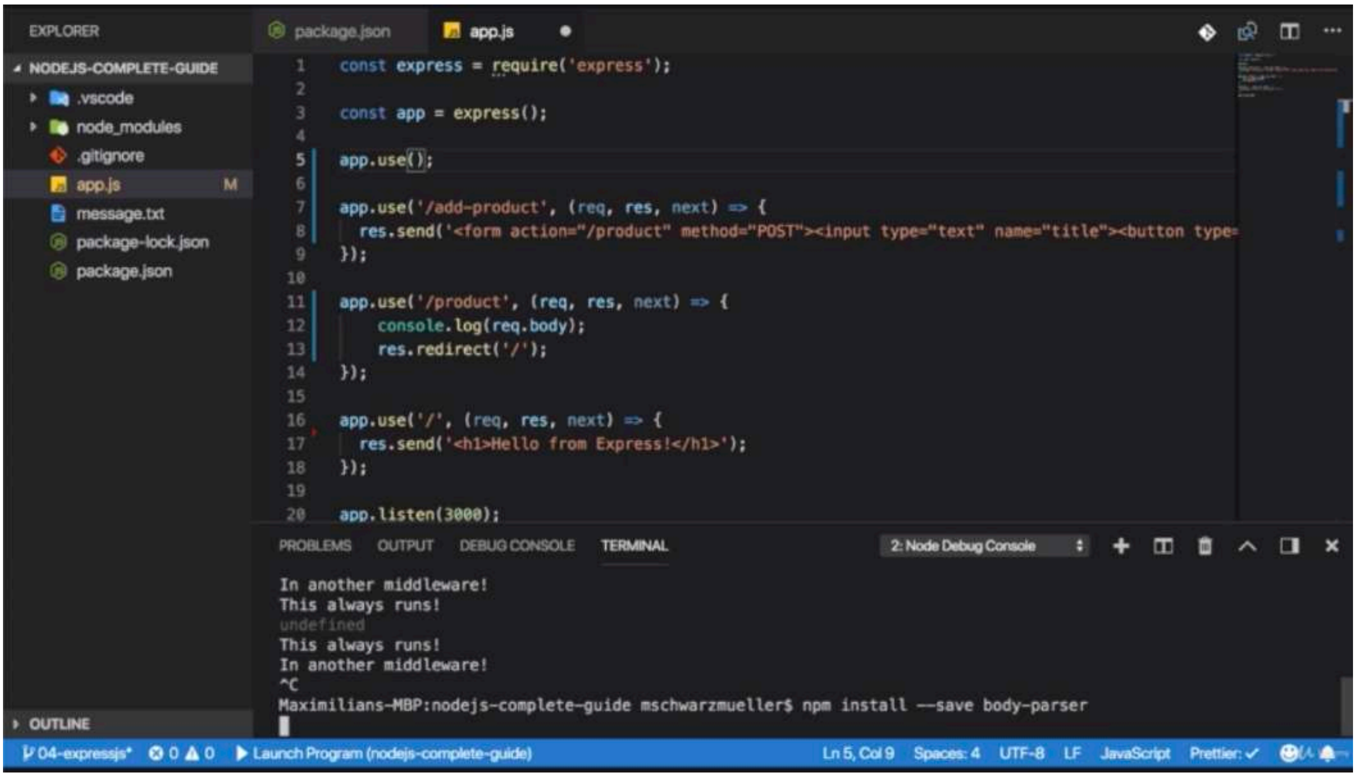
- if we go back to '/add-product' and fill out input field, we redirect to '/'
![](images/63-parsing-incoming-requests-4.png)



```javascript
const app = express();

app.use('/', (req, res, next) => {
    console.log('This always runs!');
    next();
});

app.use('/add-product', (req, res, next) => {
  console.log('In another middleware!');
  res.send('<form action="/product" method="POST"><input type="text" name="title"><button type=
});

app.use('/product', (req, res, next) => {
    console.log(req.body);
    res.redirect('/');
});

app.use('/', (req, res, next) => {
  console.log('In another middleware!');
  res.send('<h1>Hello from Express!</h1>');
```

```
[nodemon] starting `node app.js`
This always runs!
In another middleware!
This always runs!
undefined
This always runs!
In another middleware!
```

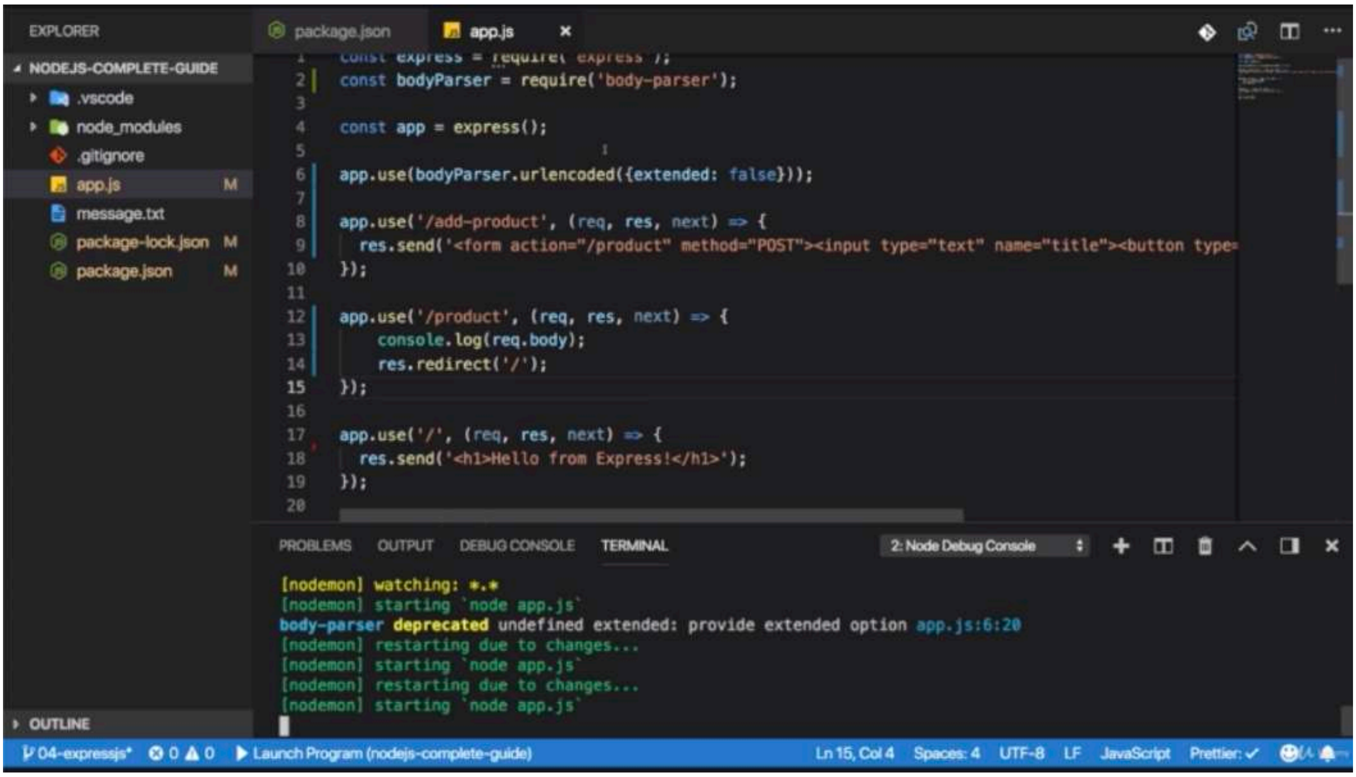
- and we see ‘undefined’ in the console.
- request gives us .body convenience property. but by default, request doesn’t try to parse the incoming request body. so we need to register a parser and we do that by adding another middleware.
- and you typically do that before your route handling middlewares because the parsing of the body should be done no matter where your request ends up. and there i wanna parse the incoming req.body.
- so we can install a third party package.
![](images/63-parsing-incoming-requests-5.png)



```javascript
const express = require('express');

const app = express();

app.use();

app.use('/add-product', (req, res, next) => {
  res.send('<form action="/product" method="POST"><input type="text" name="title"><button type=
});

app.use('/product', (req, res, next) => {
    console.log(req.body);
    res.redirect('/');
});

app.use('/', (req, res, next) => {
  res.send('<h1>Hello from Express!</h1>');
});

app.listen(3000);
```

```
In another middleware!
This always runs!
undefined
This always runs!
In another middleware!
^C
Maximilians-MBP:nodejs-complete-guide mschwarzmueller$ npm install --save body-parser
```

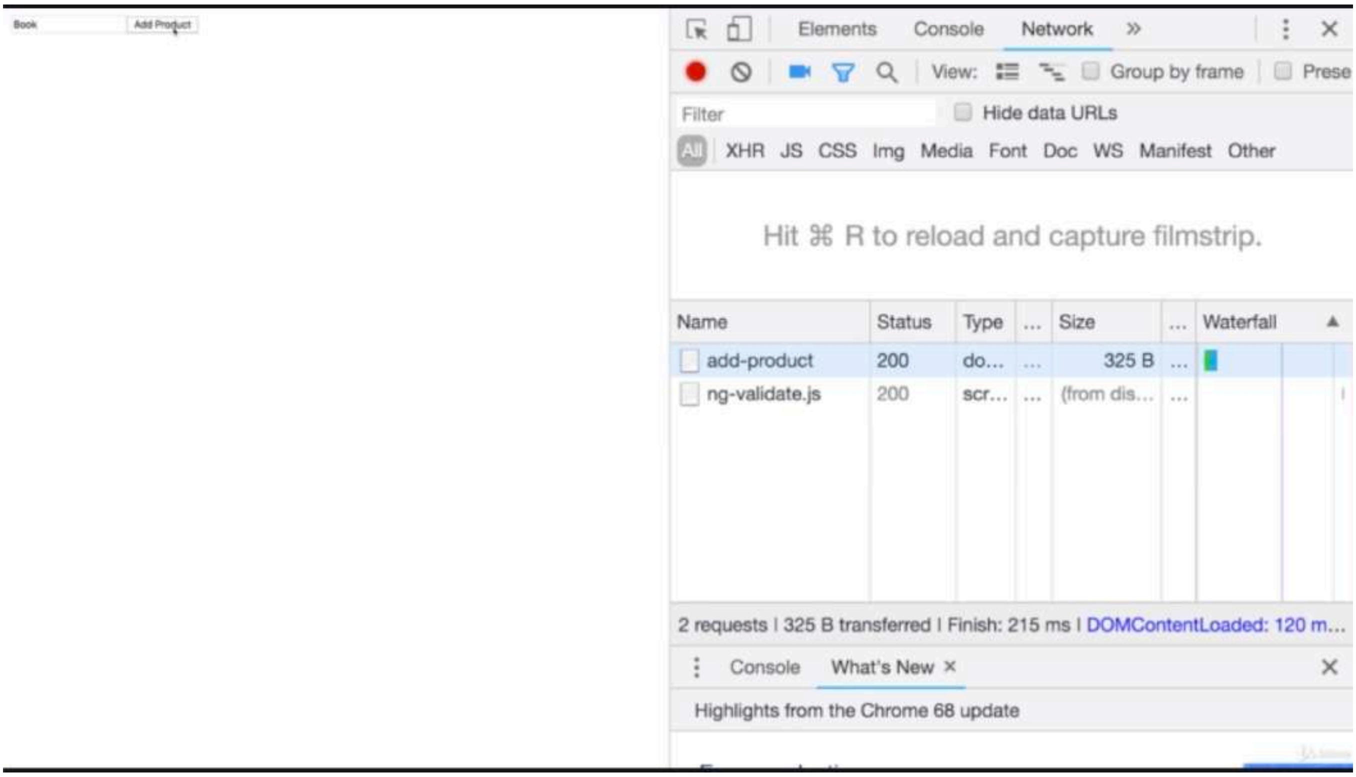
- —save because this will also be a package that is used in our code that does matter for production. and the package name is ‘body-parser
![](images/63-parsing-incoming-requests-6.png)’



```
const express = require('express');
const bodyParser = require('body-parser');

const app = express();

app.use(bodyParser.urlencoded({extended: false}));

app.use('/add-product', (req, res, next) => {
  res.send('<form action="/product" method="POST"><input type="text" name="title"><button type=
});

app.use('/product', (req, res, next) => {
    console.log(req.body);
    res.redirect('/');
});

app.use('/', (req, res, next) => {
  res.send('<h1>Hello from Express!</h1>');
});
```

```
[nodemon] watching: *.*
[nodemon] starting `node app.js`
body-parser deprecated undefined extended: provide extended option app.js:6:20
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
```

- we get that body parser enabled
![](images/63-parsing-incoming-requests-7.png)



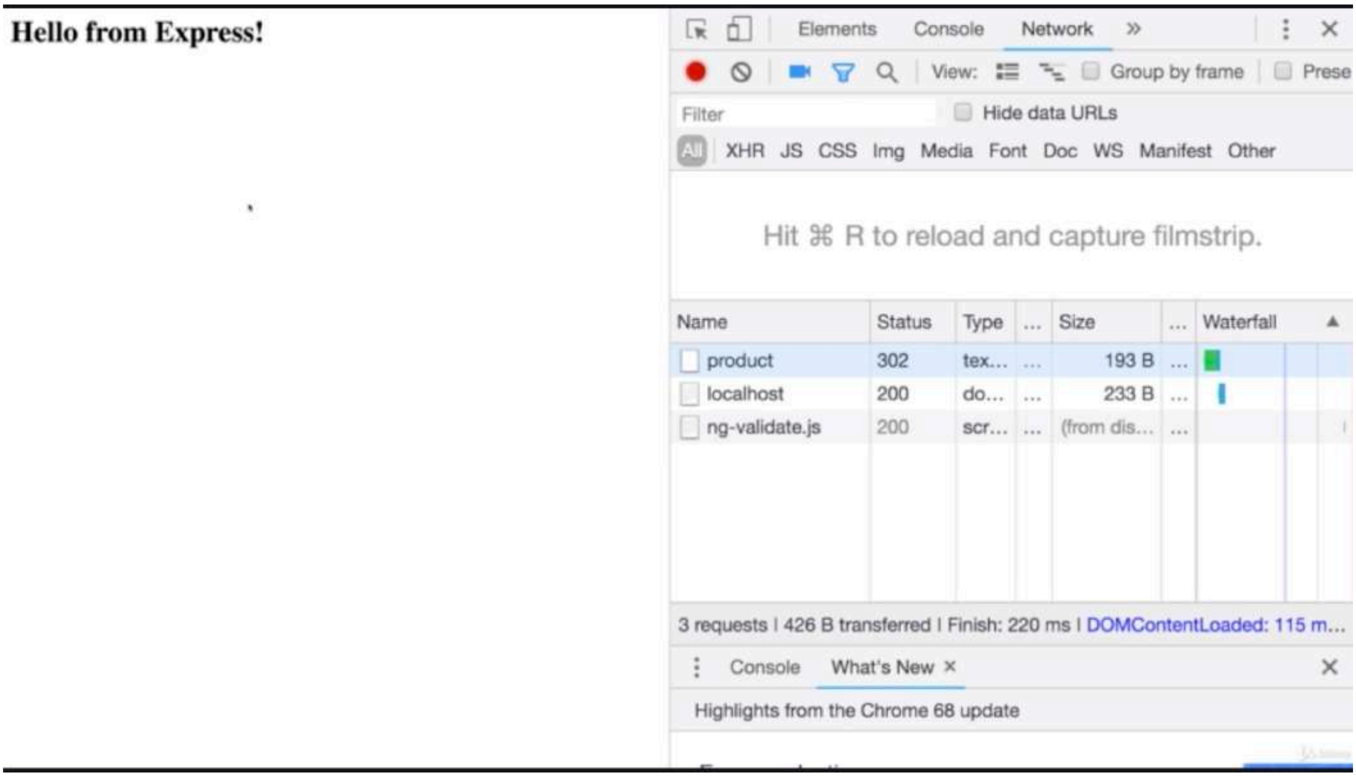
Hit ⌘ R to reload and capture filmstrip.

| Name | Status | Type | ... | Size | ... | Waterfall |
|---|---|---|---|---|---|---|
| add-product | 200 | do... | ... | 325 B | ... | |
| ng-validate.js | 200 | scr... | ... | (from dis... | ... | |

2 requests | 325 B transferred | Finish: 215 ms | DOMContentLoaded: 120 m...

Highlights from the Chrome 68 update

- and go back '/add-product' again, fill out input field.
![](images/63-parsing-incoming-requests-8.png)



Hello from Express!

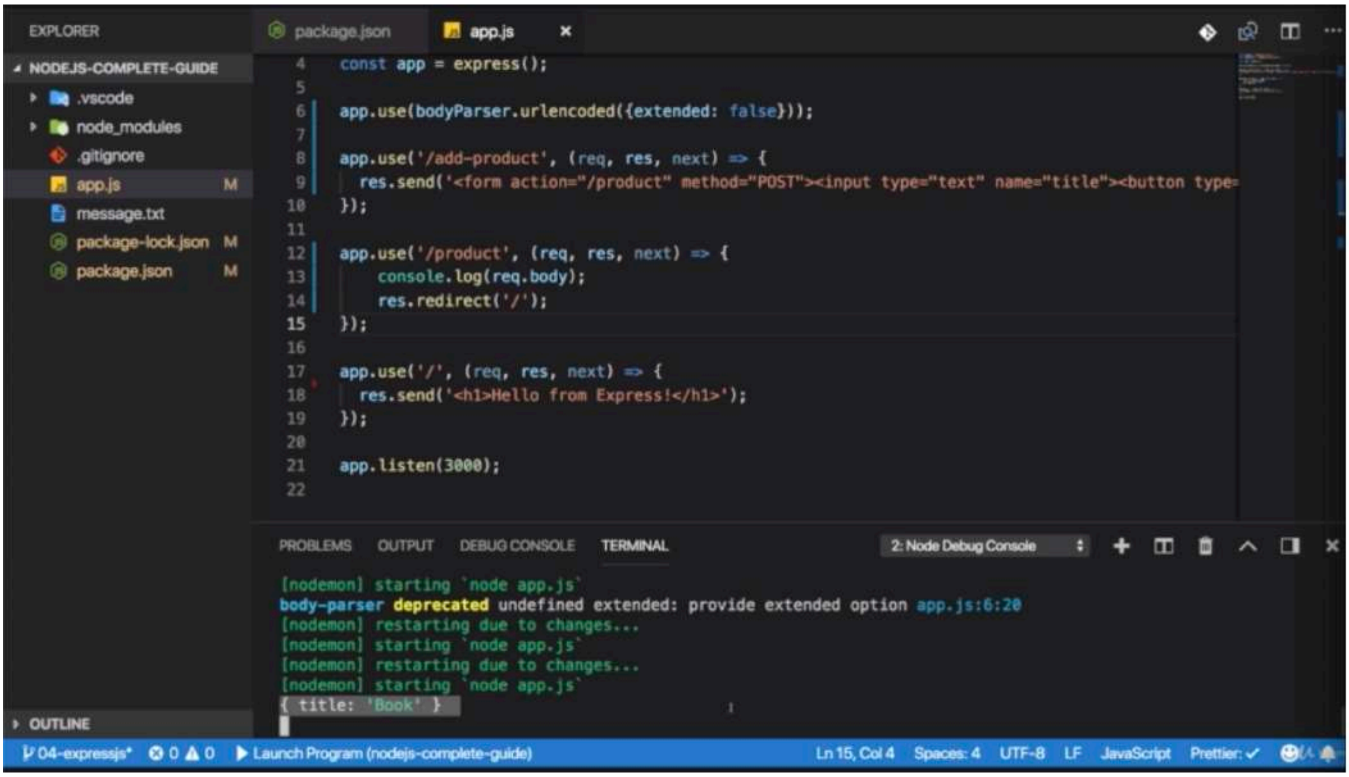
Hit ⌘ R to reload and capture filmstrip.

| Name | Status | Type | ... | Size | ... | Waterfall |
|---|---|---|---|---|---|---|
| product | 302 | tex... | ... | 193 B | ... | |
| localhost | 200 | do... | ... | 233 B | ... | |
| ng-validate.js | 200 | scr... | ... | (from dis... | ... | |

3 requests | 426 B transferred | Finish: 220 ms | DOMContentLoaded: 115 m...

Highlights from the Chrome 68 update

- and it redirect.
![](images/63-parsing-incoming-requests-9.png)



```
const app = express();

app.use(bodyParser.urlencoded({extended: false}));

app.use('/add-product', (req, res, next) => {
  res.send('<form action="/product" method="POST"><input type="text" name="title"><button type=
});

app.use('/product', (req, res, next) => {
    console.log(req.body);
    res.redirect('/');
});

app.use('/', (req, res, next) => {
  res.send('<h1>Hello from Express!</h1>');
});

app.listen(3000);
```

```
[nodemon] starting `node app.js`
body-parser deprecated undefined extended: provide extended option app.js:6:20
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
{ title: 'Book' }
```

- and we see this is what we get, a javascript object with a key-value pair which also makes extracting the value easier than we had to do before with the split function where we manually had to create that array and so on.

```
//app.js

const express = require('express');
const bodyParser = require('body-parser');

const app = express()

/**'urlencoded()' is a function you have to execute
```

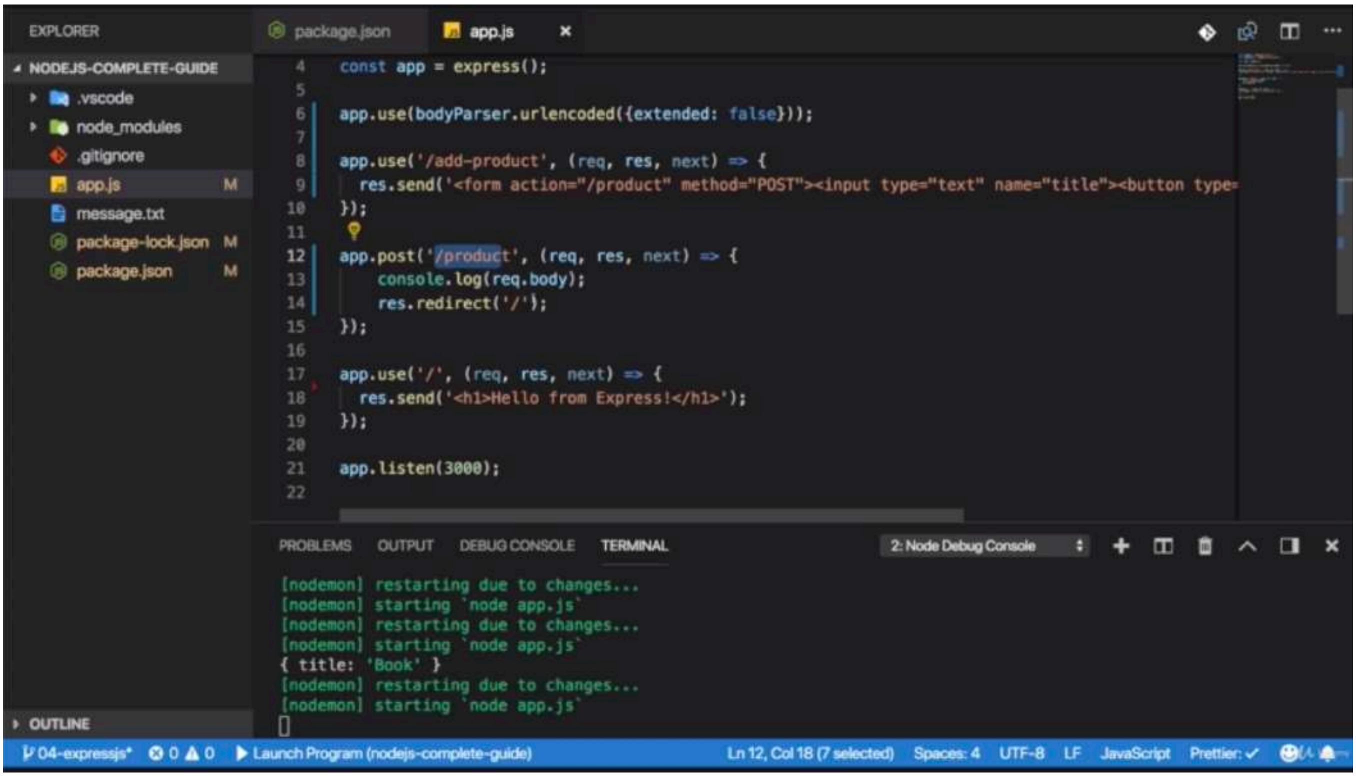
```
 * and you can pass options to configure it
 * but you don't have to here
 *
 * and 'bodyParser' is the middleware
 * so 'urlencoded()' function in the end just yields us such a middleware function like (req, res, next) => {}
 * so 'bodyParser.urlencoded()' parse such a function like (req, res, next) => {} in the end even though we can't see it
 * and in the end, this middleware function call 'next' in the end,
 * so that the request also reaches our middleware
 *
 * but before it does that, it will do that whole request body parsing we had to do manually in the previous core section.
 *
 * now this will not parse all kinds of possible bodies, files, json and so on
 * but this will parse bodies like the one which is sent through a form.
 */

 /** in 'urlencoded()', you should pass the config options
  * '{extended: false}' is added to comply with what we should use here
  */
app.use(bodyParser.urlencoded({extended: false}))

app.use('/add-product', (req, res, next) => {
    /**the path, the URL to which the request should be sent */
    res.send('<form action="/product" method="POST"><input type="text" name="title"><button type="submit">Add Product</</button></form>')
})

/**'/product' has to be after '/add-product' because of preventing '/product' from overlapping to '/add-product'
 * '/' has to be after '/product' because of preventing '/' from overlapping to '/product'
 */
app.use('/product', (req, res, next) => {
    /**'req.body' is a new field added by express */
    console.log(req.body)
    /**i can use res.redirect which certainly is easier than manually setting the status code and setting the location header.
     *
     */
    res.redirect('/');
})

app.use('/', (req, res, next) => {
    res.send('<h1>Hello from Express!</h1>')
})

app.listen(3000);
```

# * Chapter 64: Limiting Middleware Execution To POST Requests

1. update

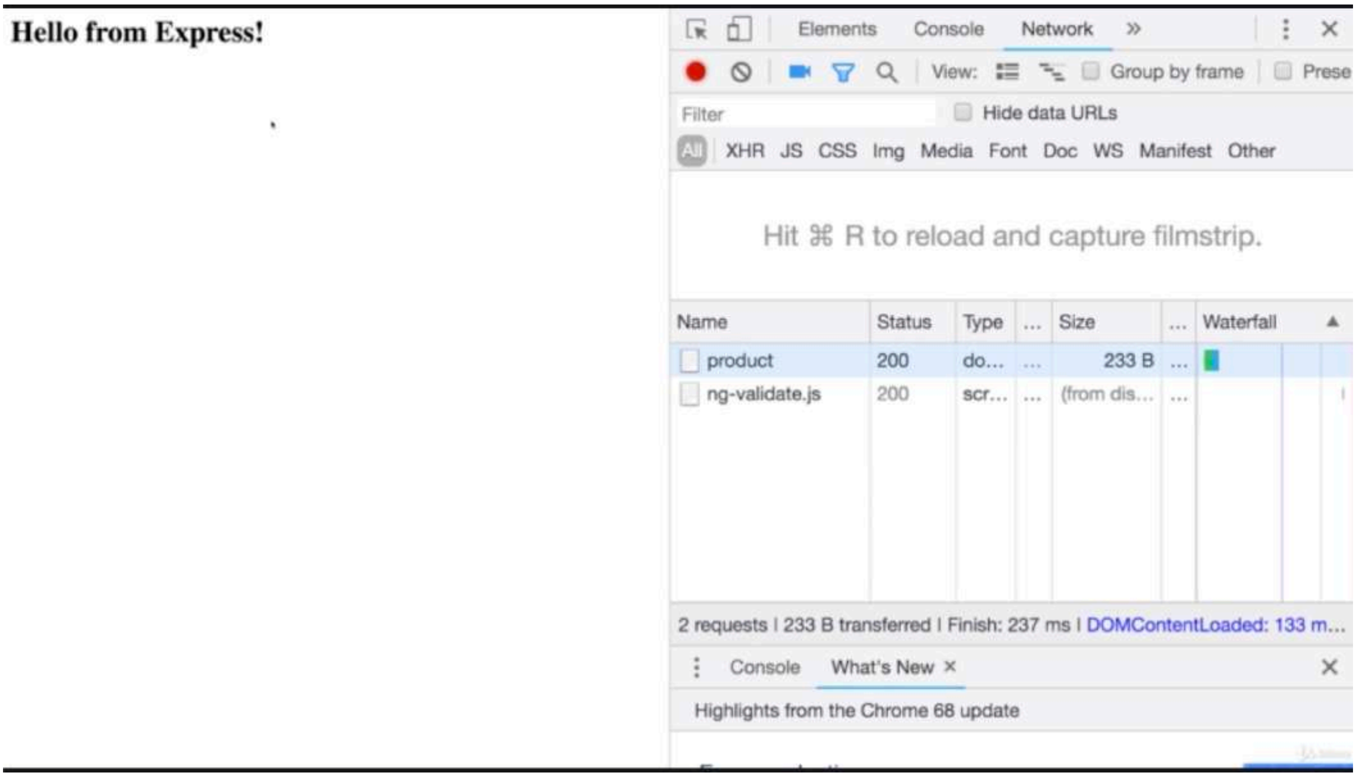
- app.js

![](images/64-limiting-middleware-execution-to-post-requests-1.png)
![](images/64-limiting-middleware-execution-to-post-requests-2.png)
![](images/64-limiting-middleware-execution-to-post-requests-3.png)



```
const app = express();

app.use(bodyParser.urlencoded({extended: false}));

app.use('/add-product', (req, res, next) => {
  res.send('<form action="/product" method="POST"><input type="text" name="title"><button type=
});

app.post('/product', (req, res, next) => {
    console.log(req.body);
    res.redirect('/');
});

app.use('/', (req, res, next) => {
  res.send('<h1>Hello from Express!</h1>');
});

app.listen(3000);
```

```
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
{ title: 'Book' }
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
```

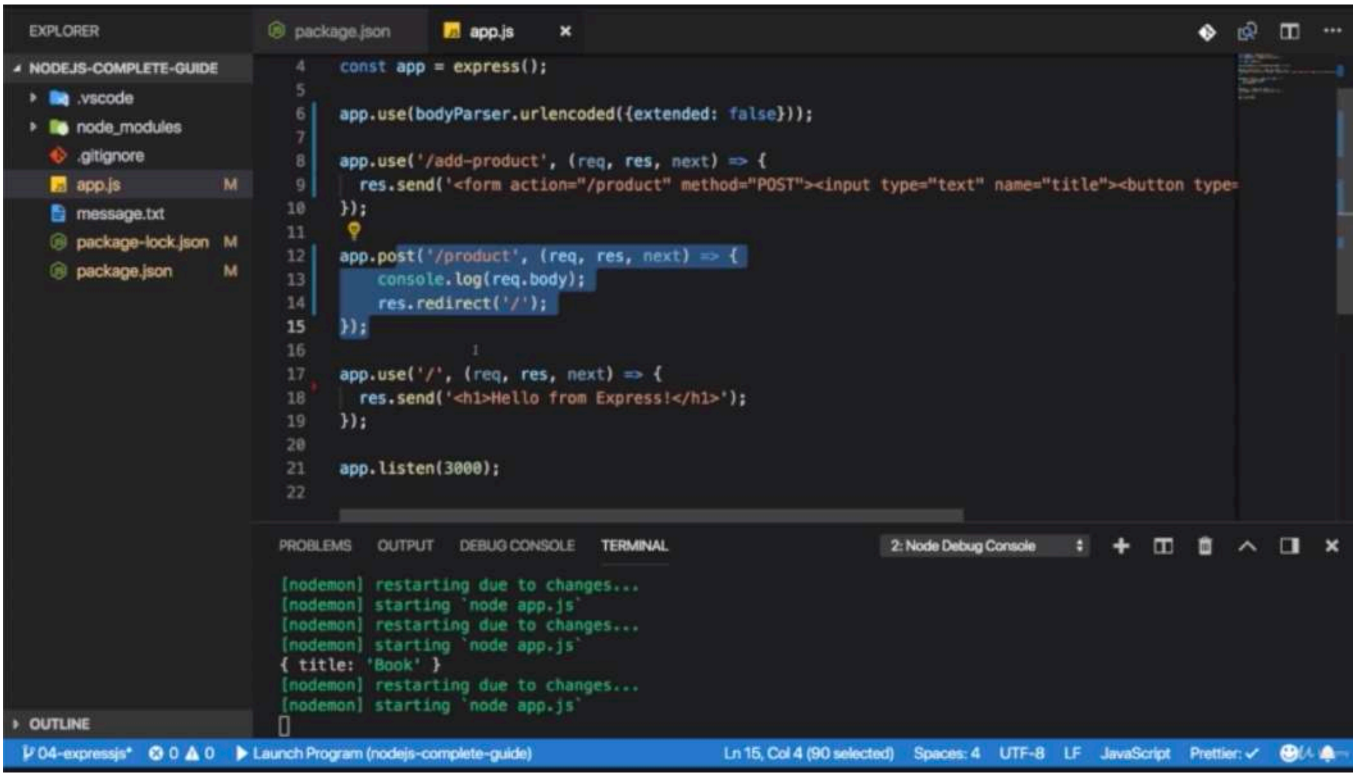


**Hello from Express!**

| Name | Status | Type | ... | Size | ... | Waterfall |
|---|---|---|---|---|---|---|
| product | 200 | do... | ... | 233 B | ... | |
| ng-validate.js | 200 | scr... | ... | (from dis... | ... | |

2 requests | 233 B transferred | Finish: 237 ms | DOMContentLoaded: 133 m...

Highlights from the Chrome 68 update



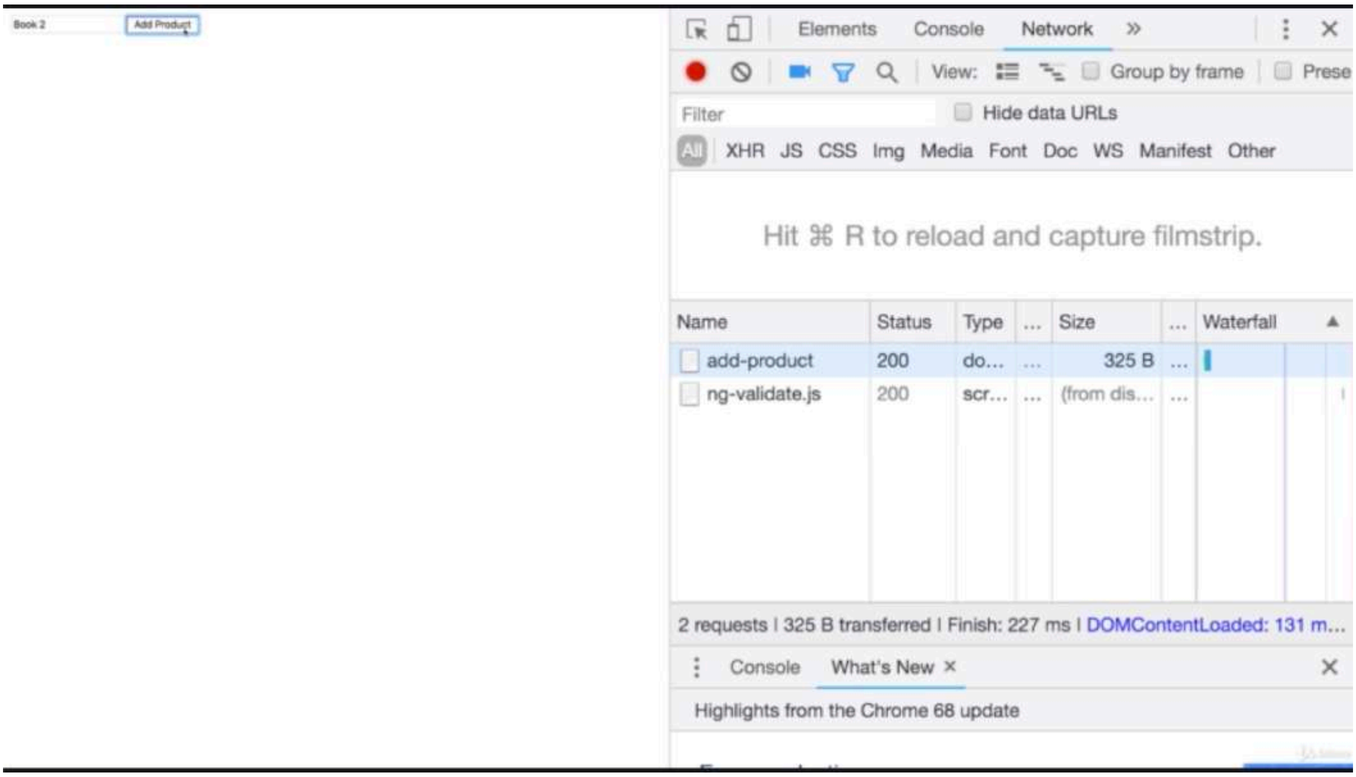
```
const app = express();

app.use(bodyParser.urlencoded({extended: false}));

app.use('/add-product', (req, res, next) => {
  res.send('<form action="/product" method="POST"><input type="text" name="title"><button type=
});

app.post('/product', (req, res, next) => {
    console.log(req.body);
    res.redirect('/');
});

app.use('/', (req, res, next) => {
  res.send('<h1>Hello from Express!</h1>');
});

app.listen(3000);
```

```
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
{ title: 'Book' }
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
```

- if i go to '/product', i see 'hello from express' so i don't end up here

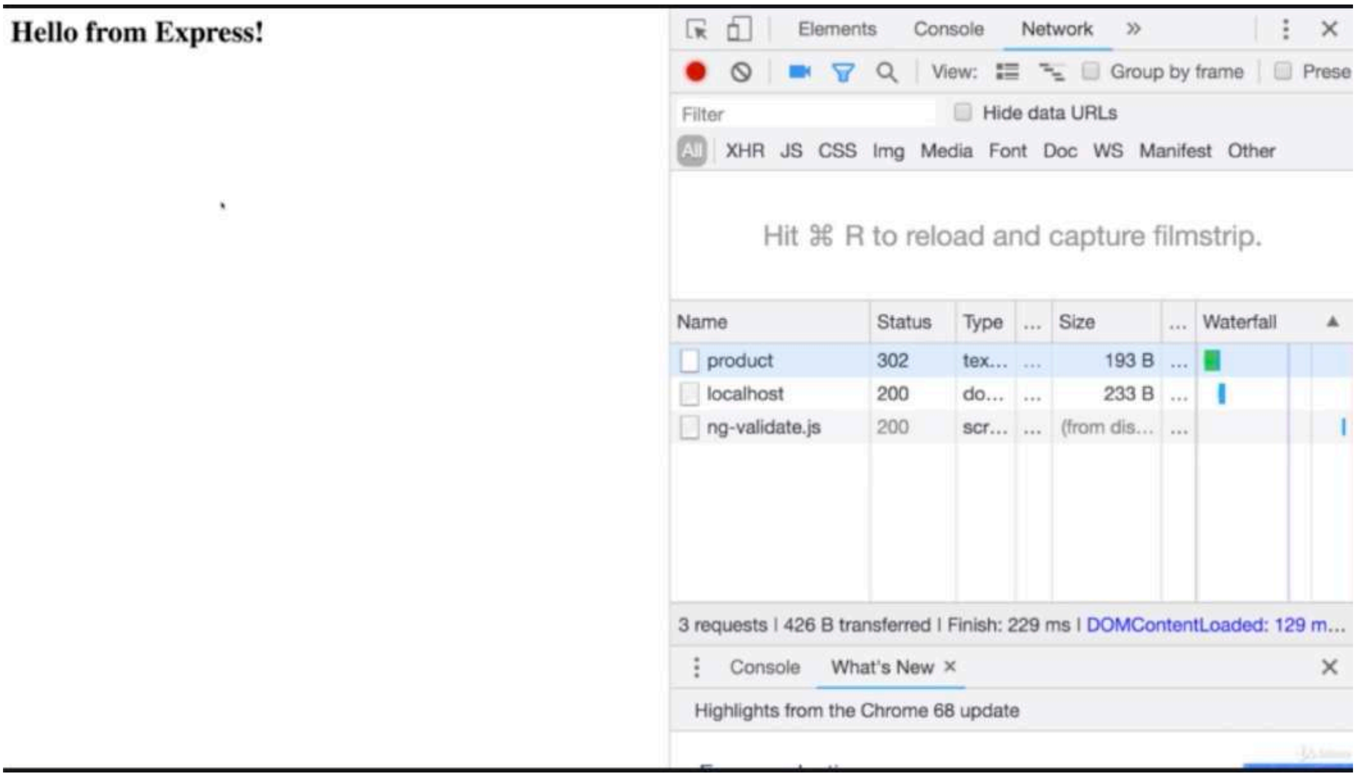
```
//app.js

app.post('/product', (req, res, next) => {
    console.log(req.body)
    res.redirect('/');
})
```

even though i entered '/product' but it was a GET request.
![](images/64-limiting-middleware-execution-to-post-requests-4.png)
![](images/64-limiting-middleware-execution-to-post-requests-5.png)
![](images/64-limiting-middleware-execution-to-post-requests-6.png)



| Name | Status | Type | ... | Size | ... | Waterfall |
|---|---|---|---|---|---|---|
| add-product | 200 | do... | ... | 325 B | ... | |
| ng-validate.js | 200 | scr... | ... | (from dis... | ... | |

2 requests | 325 B transferred | Finish: 227 ms | DOMContentLoaded: 131 m...



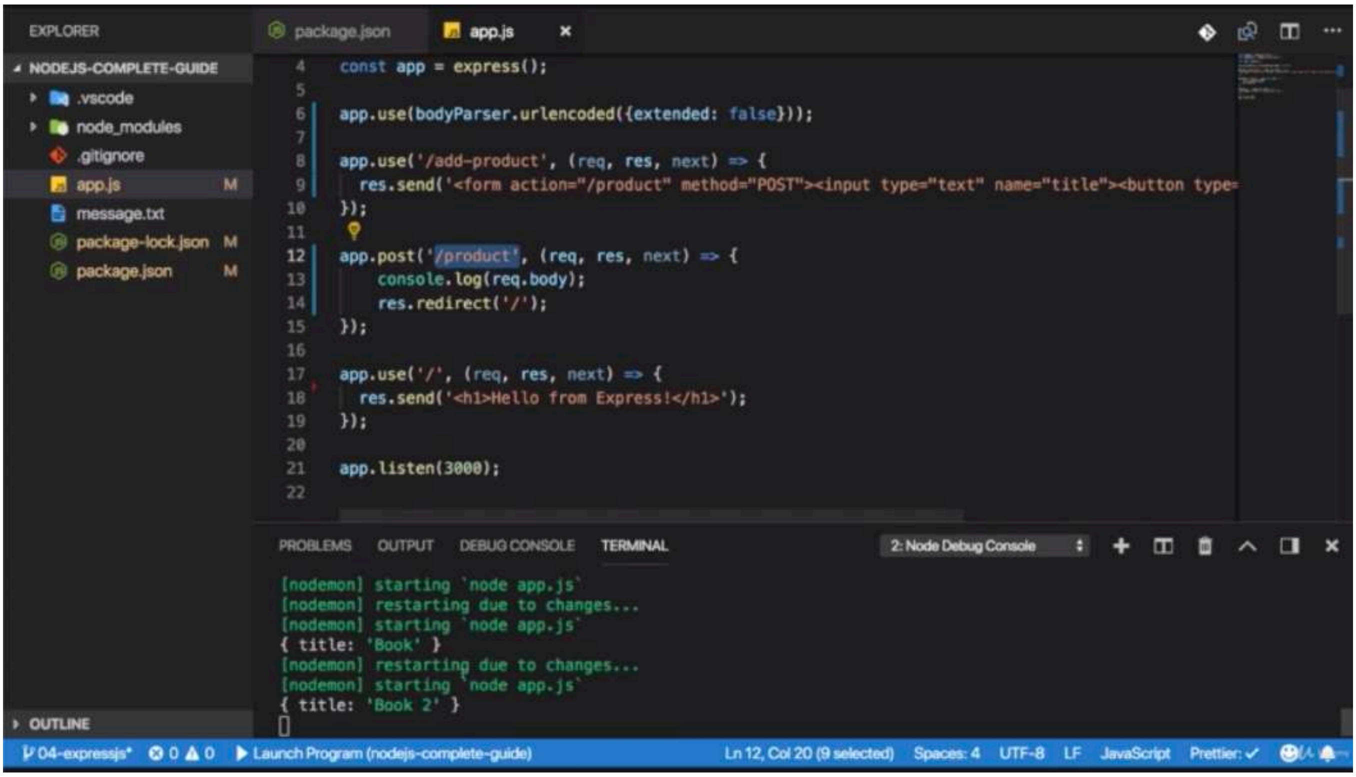
Hello from Express!

Hit ⌘ R to reload and capture filmstrip.

| Name | Status | Type | ... | Size | ... | Waterfall |
|---|---|---|---|---|---|---|
| product | 302 | tex... | ... | 193 B | ... | |
| localhost | 200 | do... | ... | 233 B | ... | |
| ng-validate.js | 200 | scr... | ... | (from dis... | ... | |

3 requests | 426 B transferred | Finish: 229 ms | DOMContentLoaded: 129 m...

Highlights from the Chrome 68 update



```
const app = express();

app.use(bodyParser.urlencoded({extended: false}));

app.use('/add-product', (req, res, next) => {
  res.send('<form action="/product" method="POST"><input type="text" name="title"><button type=
});

app.post('/product', (req, res, next) => {
    console.log(req.body);
    res.redirect('/');
});

app.use('/', (req, res, next) => {
  res.send('<h1>Hello from Express!</h1>');
});

app.listen(3000);
```

```
[nodemon] starting `node app.js`
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
{ title: 'Book' }
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
{ title: 'Book 2' }
```

- but if i send POST request through that form, '/add-product', you see we get this output 'book 2'. so we clearly made it into this below middleware due to our filtering.

```
//app.js

app.post('/product', (req, res, next) => {
    console.log(req.body)
    res.redirect('/')
})
```

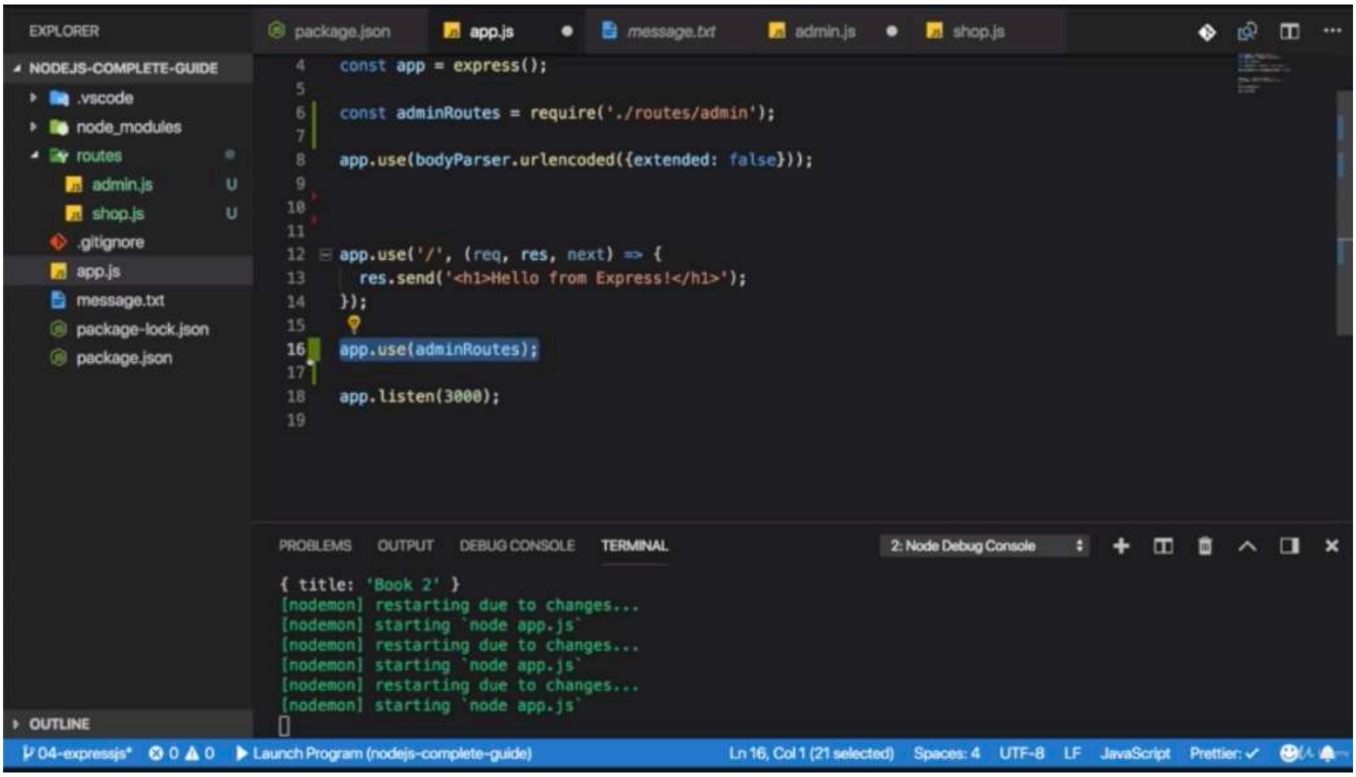
- so this is another way of using that middleware function, instead of 'use()' which will work with all http methods.
- we can also use get or post to filter for these.

```
//app.js

const express = require('express');
const bodyParser = require('body-parser');

const app = express()

app.use(bodyParser.urlencoded({extended: false}))

app.use('/add-product', (req, res, next) => {
    res.send('<form action="/product" method="POST"><input type="text" name="title"><button type="submit">Add Product</</button></form>')
})

/**this middleware always execute, not just for POST requests but also for GET requests.
 * what can we do regarding tha?
 * we can use app.get() which is basically app.use(), it has the same syntax as app.use
 * it only will fire for incoming GET request.
 * this is another form of filtering besides filtering for the path, app.get allows us to filter for GET request.
 *
  */

/**'app.post()' to filter for incoming POST requests
 * and this 'app.post()' will only trigger for incoming POST requests with this path
 * and not for GET request. */
app.post('/product', (req, res, next) => {
    console.log(req.body)
    res.redirect('/');
})

app.use('/', (req, res, next) => {
    res.send('<h1>Hello from Express!</h1>')
})

app.listen(3000);
```

# * Chapter 65: Using Express Router

1. update
- app.js
- ./routes/admin.js
- ./routes/shop.js

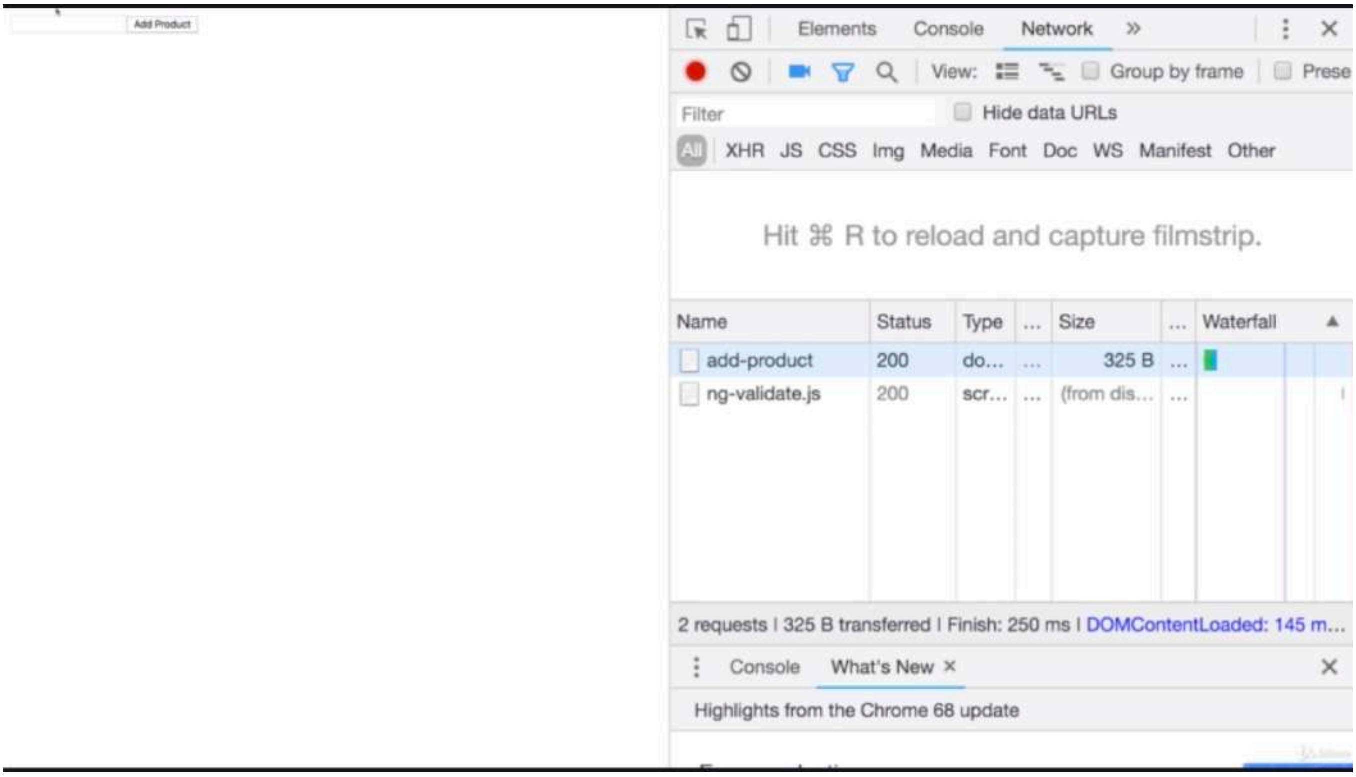
![](images/65-using-express-router-1.png)



```
const app = express();

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

app.use(bodyParser.urlencoded({extended: false}));


app.use('/', (req, res, next) => {
  res.send('<h1>Hello from Express!</h1>');
});

app.use(adminRoutes);

app.listen(3000);
```

```
{ title: 'Book 2' }
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
```

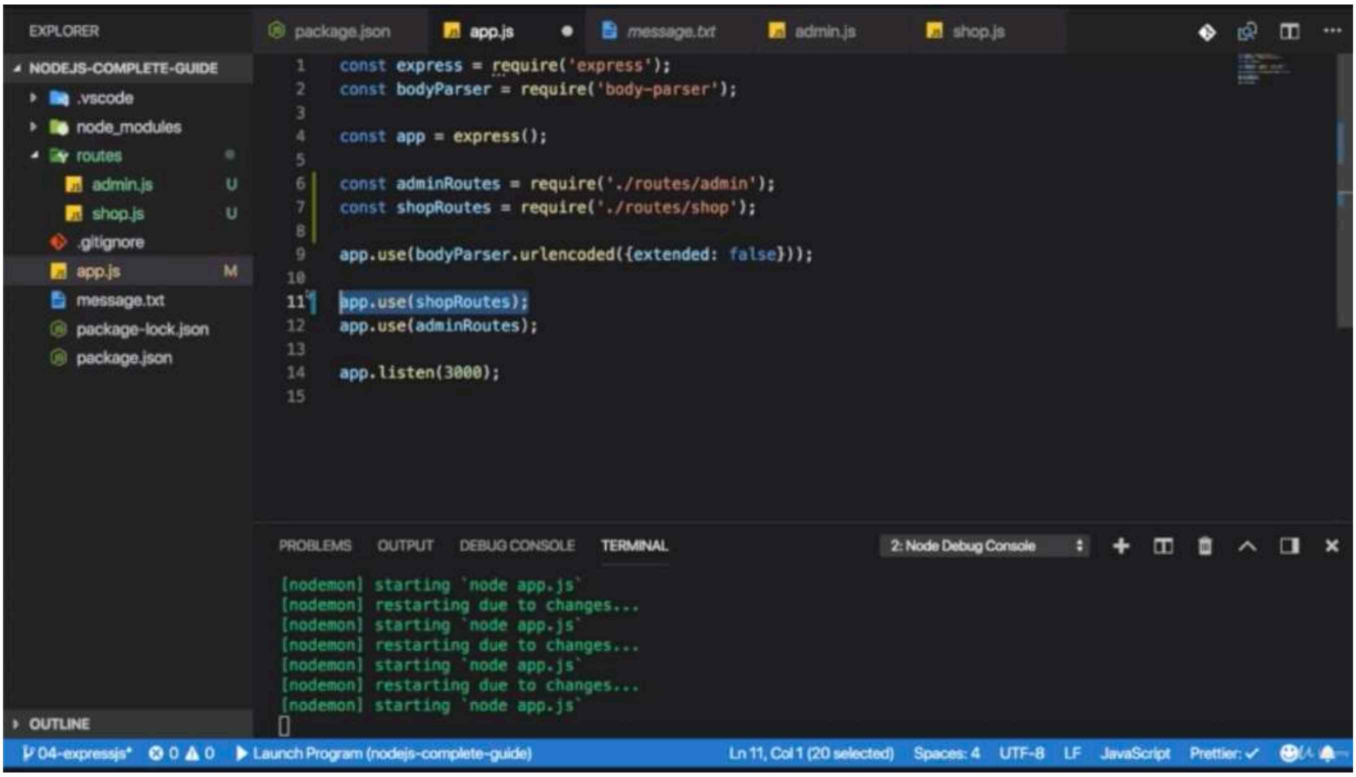
- the order matters. so if we put this after this middleware, we will never reach that.
![](images/65-using-express-router-2.png)



| Name | Status | Type | ... | Size | ... | Waterfall |
|---|---|---|---|---|---|---|
| add-product | 200 | do... | ... | 325 B | ... | |
| ng-validate.js | 200 | scr... | ... | (from dis... | ... | |

2 requests | 325 B transferred | Finish: 250 ms | DOMContentLoaded: 145 m...

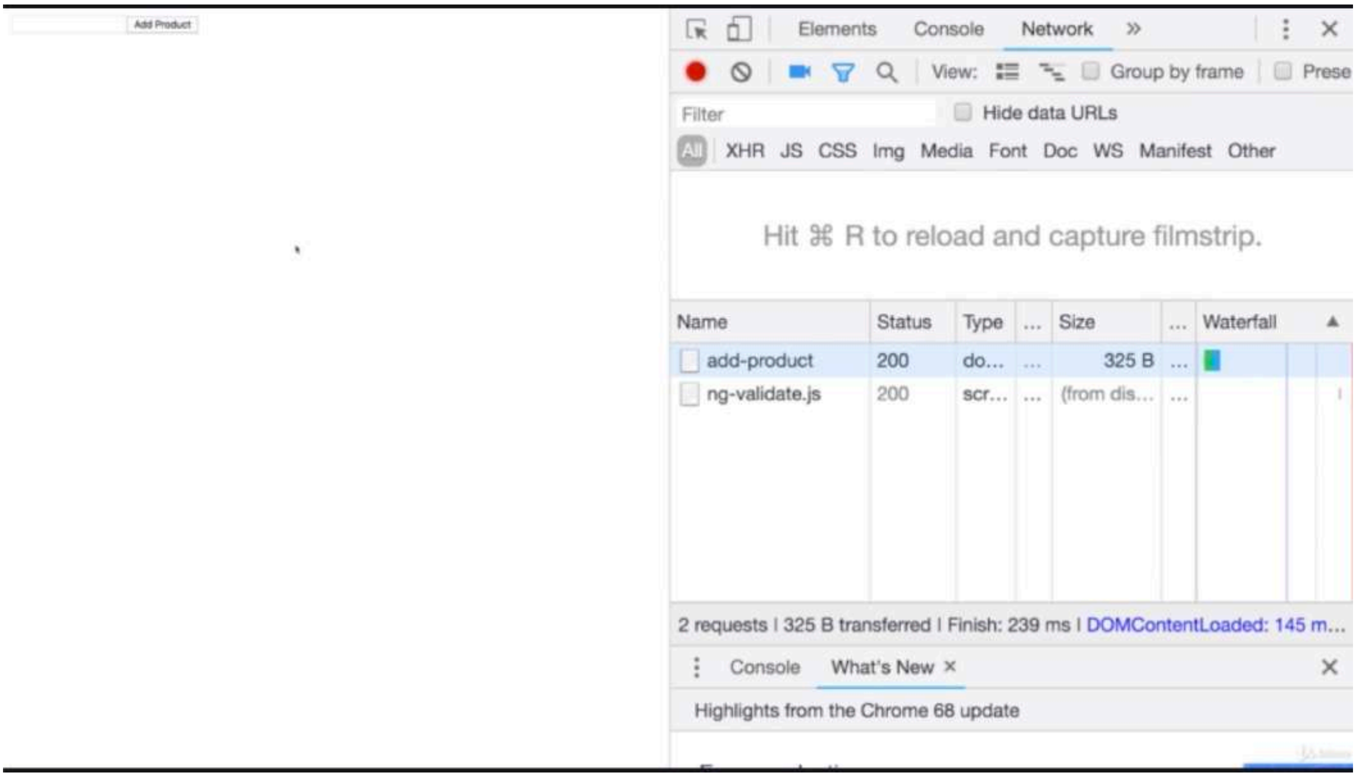
- if i save and reload, this works.
![](images/65-using-express-router-3.png)



```javascript
const express = require('express');
const bodyParser = require('body-parser');

const app = express();

const adminRoutes = require('./routes/admin');
const shopRoutes = require('./routes/shop');

app.use(bodyParser.urlencoded({extended: false}));

app.use(shopRoutes);
app.use(adminRoutes);

app.listen(3000);
```

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

- if i would switch the position of ‘app.use(shopRoutes)’ and ‘app.use(adminRoutes)’
![](images/65-using-express-router-4.png)
![](images/65-using-express-router-5.png)

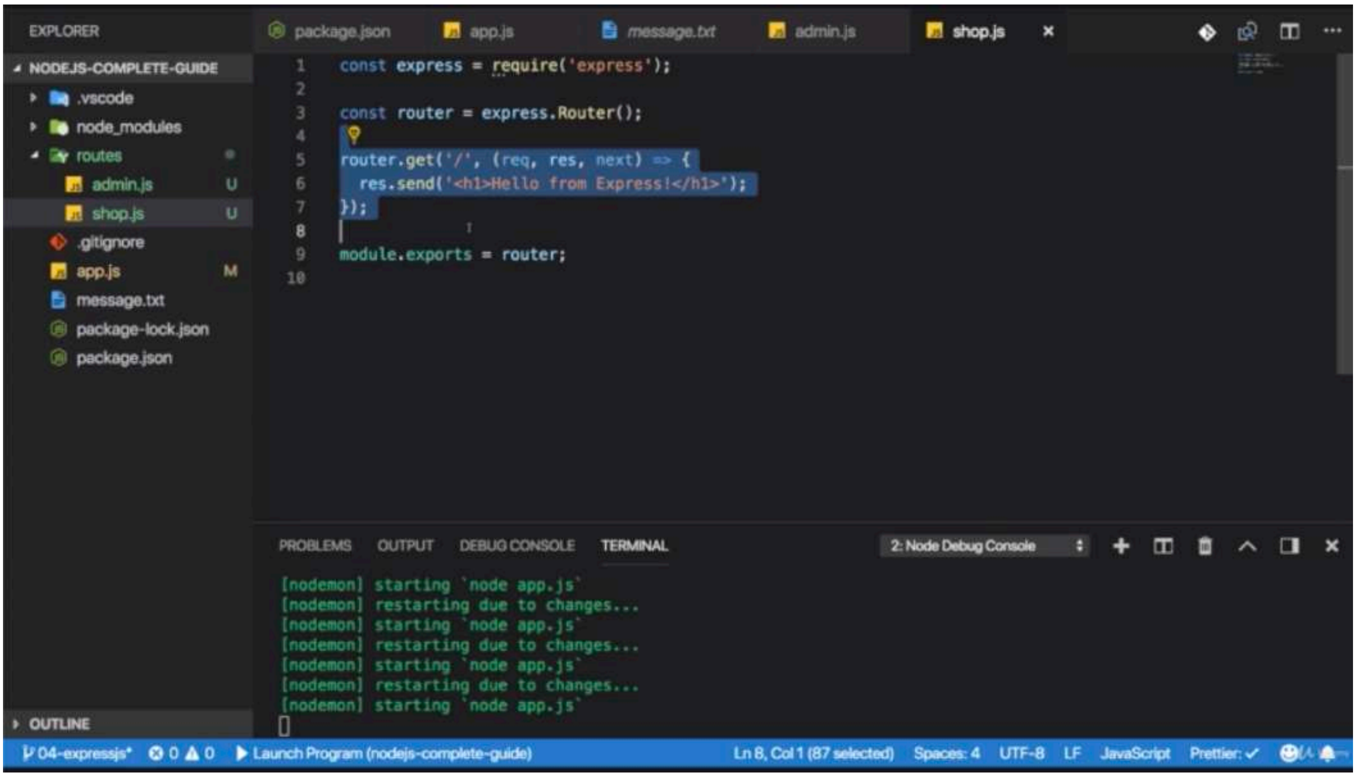


Hit ⌘ R to reload and capture filmstrip.

| Name | Status | Type | ... | Size | ... | Waterfall |
|---|---|---|---|---|---|---|
| add-product | 200 | do... | ... | 325 B | ... | |
| ng-validate.js | 200 | scr... | ... | (from dis... | ... | |

2 requests | 325 B transferred | Finish: 239 ms | DOMContentLoaded: 145 m...

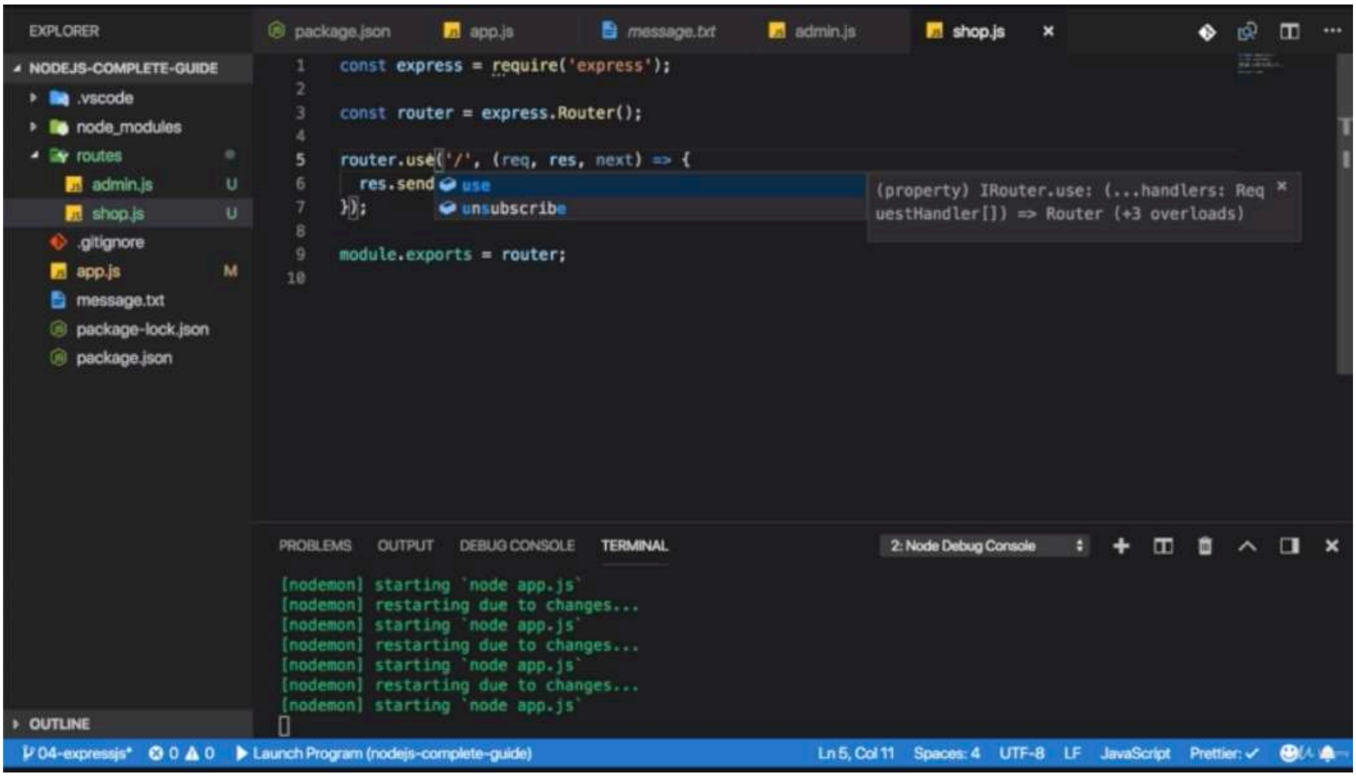
Highlights from the Chrome 68 update



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

const router = express.Router();

router.get('/', (req, res, next) => {
  res.send('<h1>Hello from Express!</h1>');
});

module.exports = router;
```

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

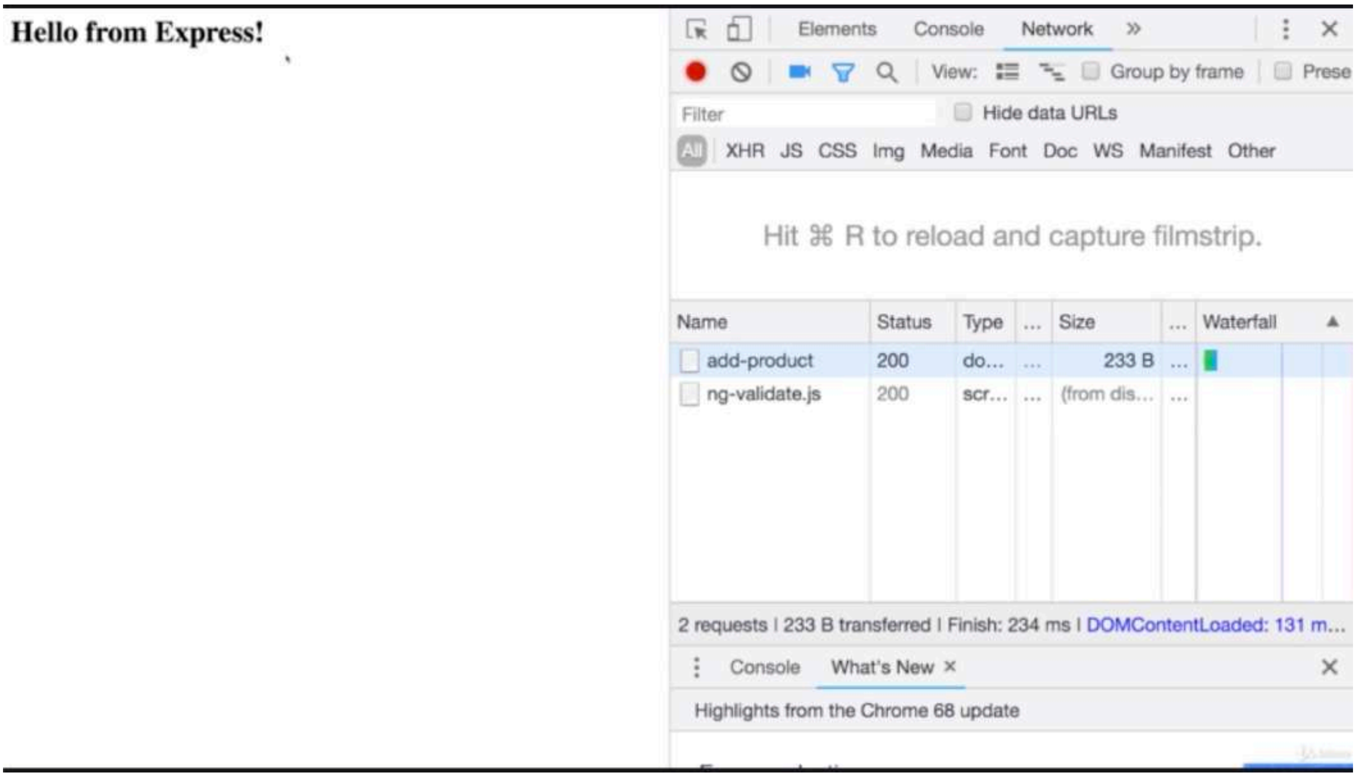
- and i reload, it would work and we would not end up in this route
- this only happens because i have .get() here. get, post and so on will actually do an exact match here.
![](images/65-using-express-router-6.png)



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

const router = express.Router();

router.use('/', (req, res, next) => {
  res.send
});

module.exports = router;
```

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

- if i would use 'use()' here as i did before to handle any incoming http method,
![](images/65-using-express-router-7.png)



Hello from Express!

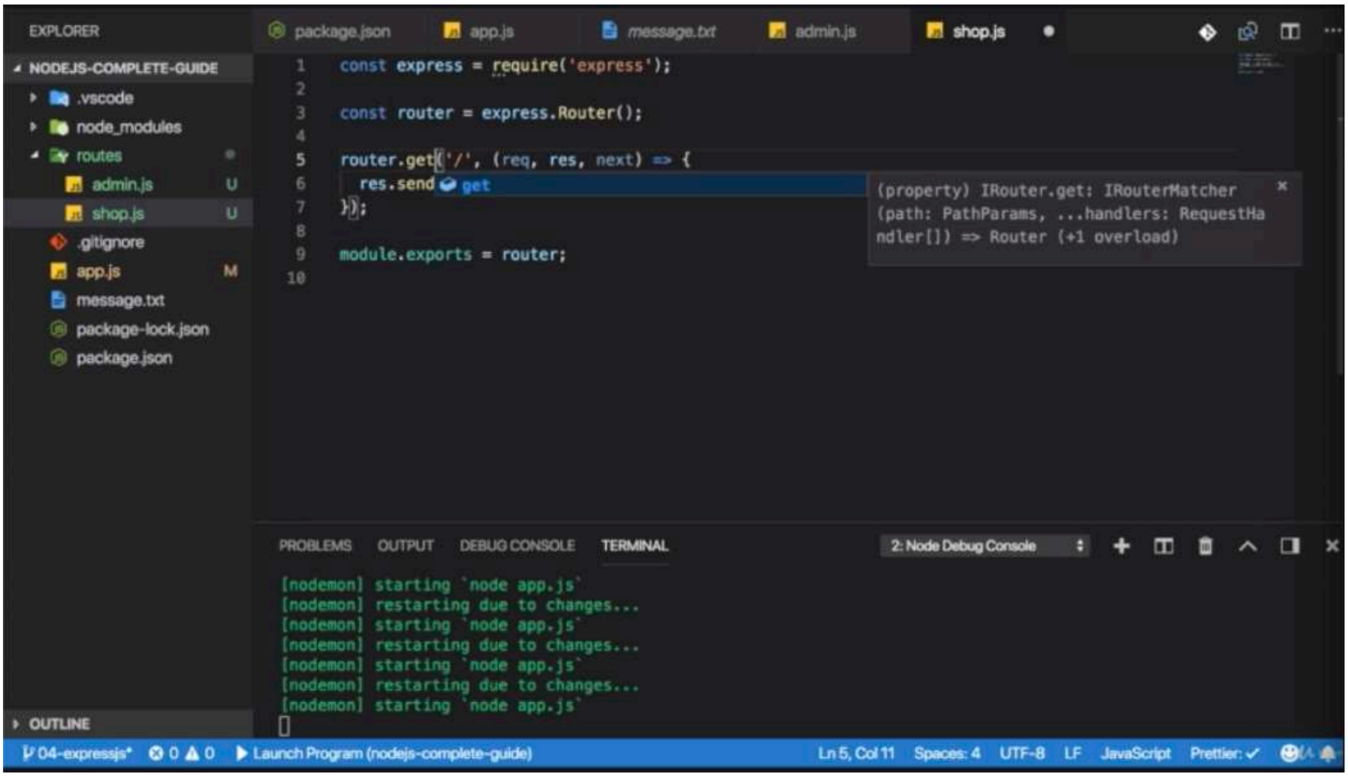
Elements
Console
Network
View:
Group by frame
Prese
Filter
Hide data URLs
All XHR JS CSS Img Media Font Doc WS Manifest Other
Hit ⌘ R to reload and capture filmstrip.

| Name | Status | Type | ... | Size | ... | Waterfall |
|---|---|---|---|---|---|---|
| add-product | 200 | do... | ... | 233 B | ... | |
| ng-validate.js | 200 | scr... | ... | (from dis... | ... | |

2 requests | 233 B transferred | Finish: 234 ms | DOMContentLoaded: 131 m...

Highlights from the Chrome 68 update

- then if i reload here, we see hello from express again.
![](images/65-using-express-router-8.png)

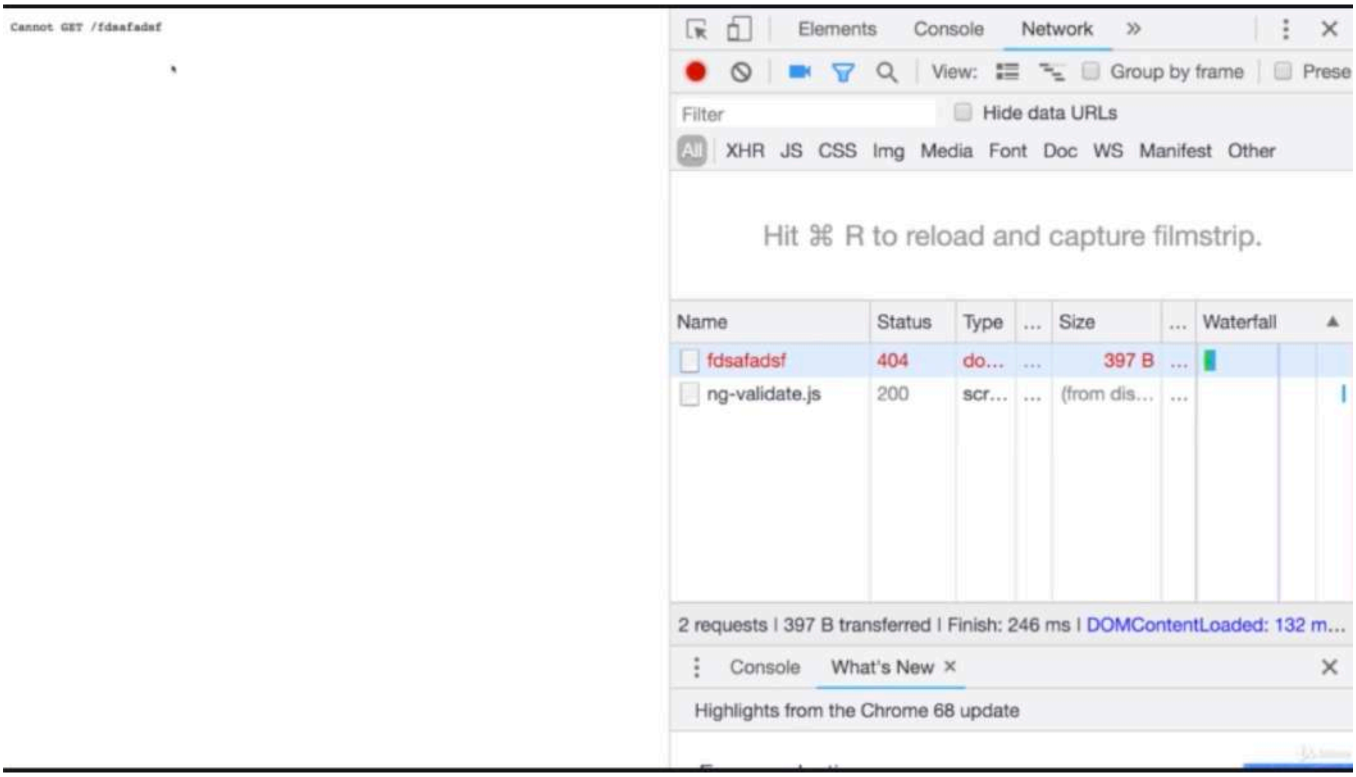


```
const express = require('express');

const router = express.Router();

router.get('/', (req, res, next) => {
  res.send get
});

module.exports = router;
```

```
(property) IRouter.get: IRouterMatcher
(path: PathParams, ...handlers: RequestHa
ndler[]) => Router (+1 overload)
```

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

- so this exact matching is not achieved by using the router but because we use get here, and that would have been the same if we stick to app way of doing this in the app.js file we had previously
- so 'get()' method make sure that it's not just a get method but this exact path
![](images/65-using-express-router-9.png)



```
Cannot GET /fdsafadsf
```

| Name | Status | Type | ... | Size | ... | Waterfall |
|---|---|---|---|---|---|---|
| fdsafadsf | 404 | do... | ... | 397 B | ... | |
| ng-validate.js | 200 | scr... | ... | (from dis... | ... | |

Hit ⌘ R to reload and capture filmstrip.

2 requests | 397 B transferred | Finish: 246 ms | DOMContentLoaded: 132 m...

Highlights from the Chrome 68 update

- and therefore now if i enter some random stuff, i get an error because now i got no single middleware that would handle that stuff.

```
//app.js

/**the order of import doesn't matter */
const express = require('express');
const bodyParser = require('body-parser');

const app = express()

const adminRoutes = require('./routes/admin');
const shopRoutes = require('./routes/shop');

app.use(bodyParser.urlencoded({extended: false}))

/**this order matters */
app.use(adminRoutes);
app.use(shopRoutes);

app.listen(3000);
```

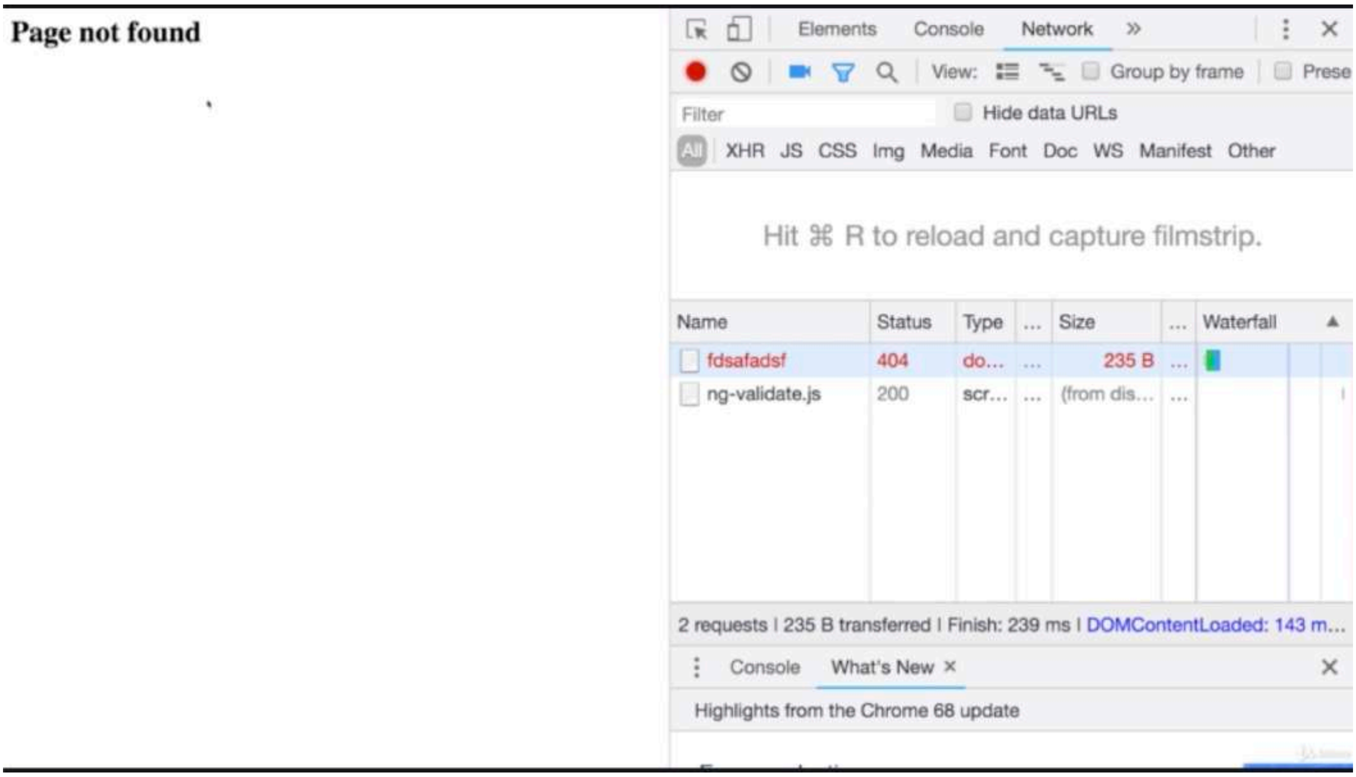
```
//./routes/admin.js

/**this should be the route that handles the creation of products which the admin of the shop can do.*/

const express = require('express')

/** 'Router()' is like a mini express app tied to the other express app or pluggable into the other express app */
const router = express.Router();

/**'router' can be used to again define a use() function for all requests,
 * get function for GET, post function for POST
 *
 * 'router' functions basically work in exactly the ame way as the app.use function does
 * or the app.get() and so on.
 *
 * i will rename this to 'get()' because i only wanna handle GET requests to 'add-product'
 * and return this form.
 * */
router.get('/add-product', (req, res, next) => {
    res.send('<form action="/product" method="POST"><input type="text" name="title"><button type="submit">Add Product</</button></form>')
})

router.post('/product', (req, res, next) => {
    console.log(req.body)
    res.redirect('/');
})

/**with that, we can import that into the app.js file
 *
 * 'router()' is a valid middleware function.
 * so we can take admin routes and just call app.use and put our admin routes in app.js file
*/
module.exports = router;
```

```
//./routes/shop.js

const express = require('express');

const router = express.Router();

router.get('/', (req, res, next) => {
    res.send('<h1>Hello from Express!</h1>')
})

module.exports = router
```

# * Chapter 66: Adding A 404 Error Page

1. update
- app.js

![](images/66-adding-a-404-error-page-1.png)



| Name | Status | Type | ... | Size | ... | Waterfall |
|---|---|---|---|---|---|---|
| fdsafadsf | 404 | do... | ... | 235 B | ... | |
| ng-validate.js | 200 | scr... | ... | (from dis... | ... | |

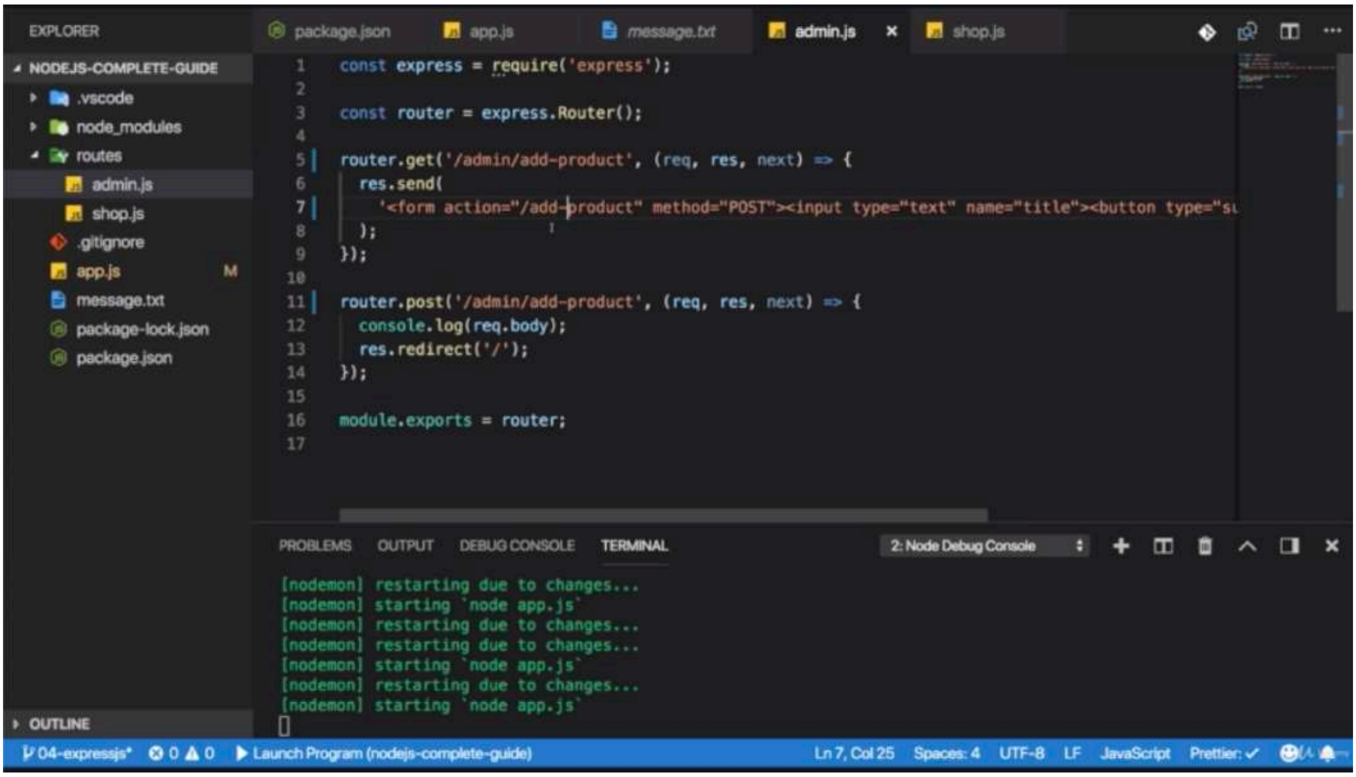
2 requests | 235 B transferred | Finish: 239 ms | DOMContentLoaded: 143 m...

```
//app.js

/**the order of import doesn't matter */
const express = require('express');
const bodyParser = require('body-parser');

const app = express()

const adminRoutes = require('./routes/admin');
const shopRoutes = require('./routes/shop');

app.use(bodyParser.urlencoded({extended: false}))

app.use(adminRoutes);
app.use(shopRoutes);

/**if we got no fitting middleware and we don't have one here,
 * then we make it all the way to the bottom
 * and eventually we don't handle that request.
 * so to send 404 error page, we simply have to add a catch all middleware at the bottom
 */

/**maybe we also wanna set the 404 status code
 * and you can do that by chaining another method prior to send
 * and that is the status() method.
 */
app.use((req, res, next) => {
    res.status(404).send('<h1>Page not found</h1>')
})

app.listen(3000);
```

# * Chapter 67: Filtering Paths

1. update
- app.js
- ./routes/admin.js
- ./routes/shop.js

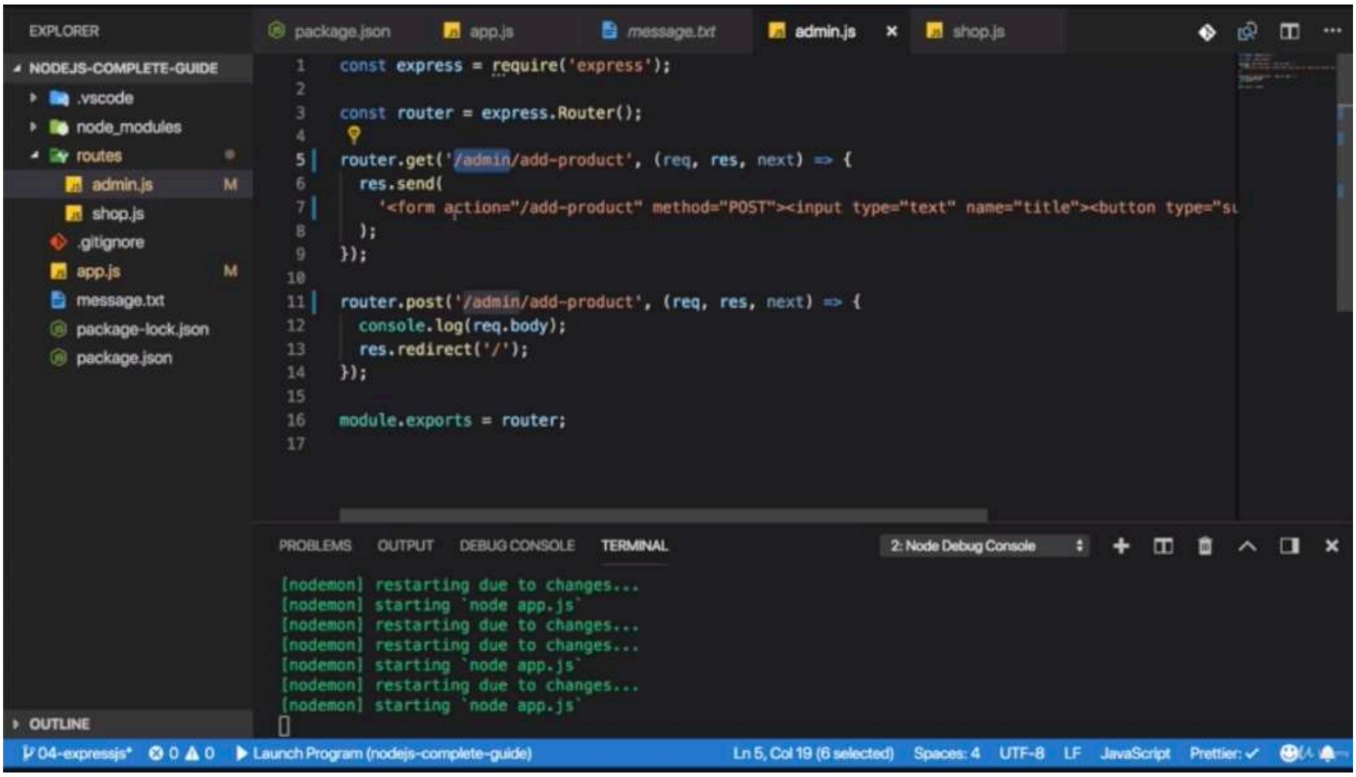
![](images/67-filtering-paths-1.png)



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

const router = express.Router();

router.get('/admin/add-product', (req, res, next) => {
  res.send(
    '<form action="/add-product" method="POST"><input type="text" name="title"><button type="su
  );
});

router.post('/admin/add-product', (req, res, next) => {
  console.log(req.body);
  res.redirect('/');
});

module.exports = router;
```

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

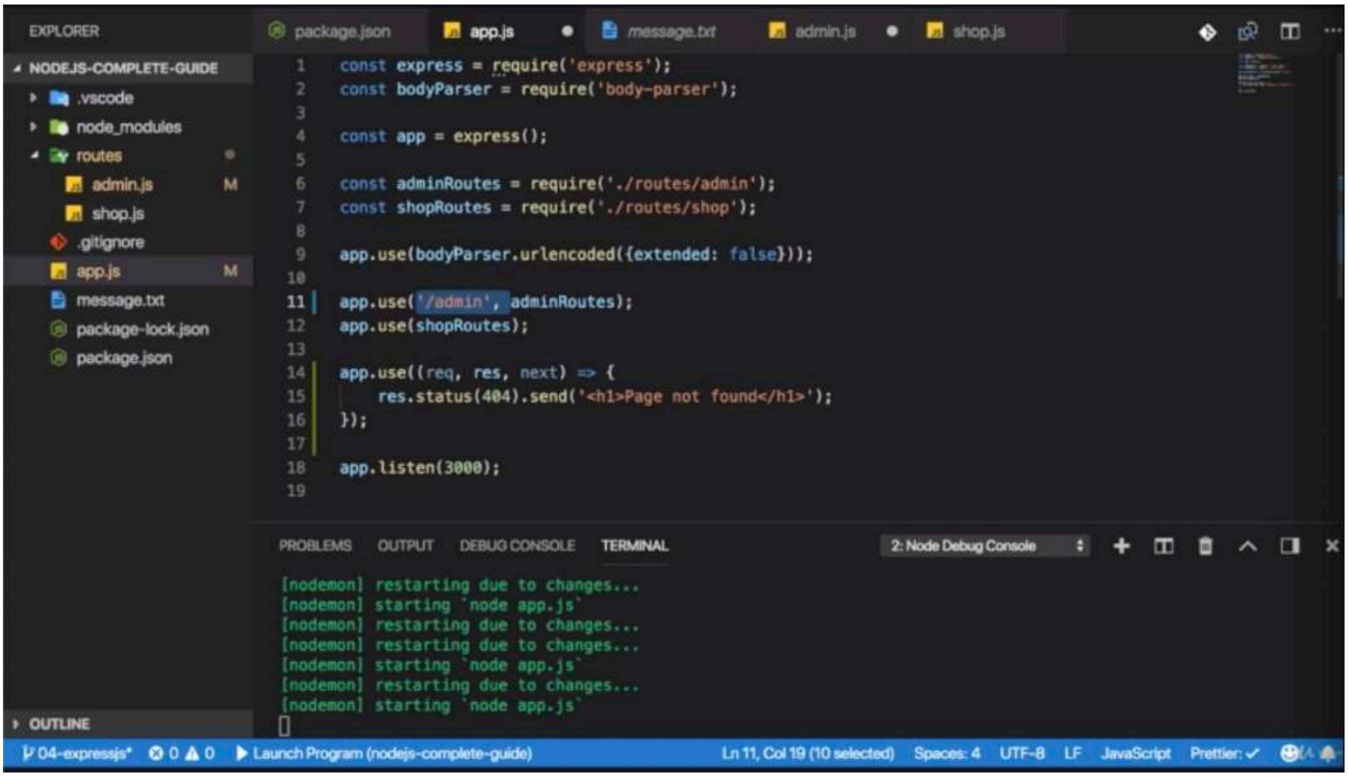
- the same path can be used if the methods differ.
![](images/67-filtering-paths-2.png)



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

const router = express.Router();

router.get('/admin/add-product', (req, res, next) => {
  res.send(
    '<form action="/add-product" method="POST"><input type="text" name="title"><button type="su
  );
});

router.post('/admin/add-product', (req, res, next) => {
  console.log(req.body);
  res.redirect('/');
});

module.exports = router;
```

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

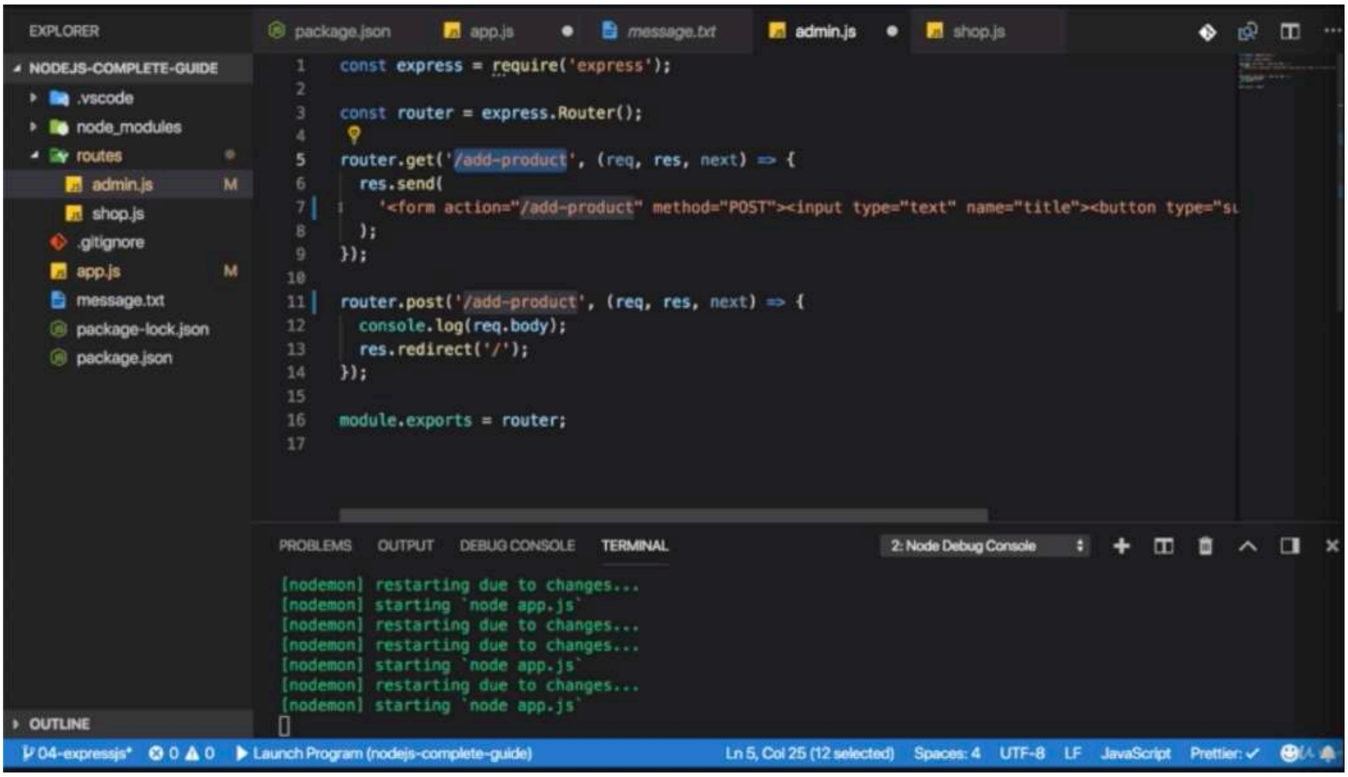
- if we have such a setup where our paths in such a router file start with the same part or with the same segment '/admin', we can take that segment out of this route
![](images/67-filtering-paths-3.png)



```
const express = require('express');
const bodyParser = require('body-parser');

const app = express();

const adminRoutes = require('./routes/admin');
const shopRoutes = require('./routes/shop');

app.use(bodyParser.urlencoded({extended: false}));

app.use('/admin', adminRoutes);
app.use(shopRoutes);

app.use((req, res, next) => {
    res.status(404).send('<h1>Page not found</h1>');
});

app.listen(3000);
```

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

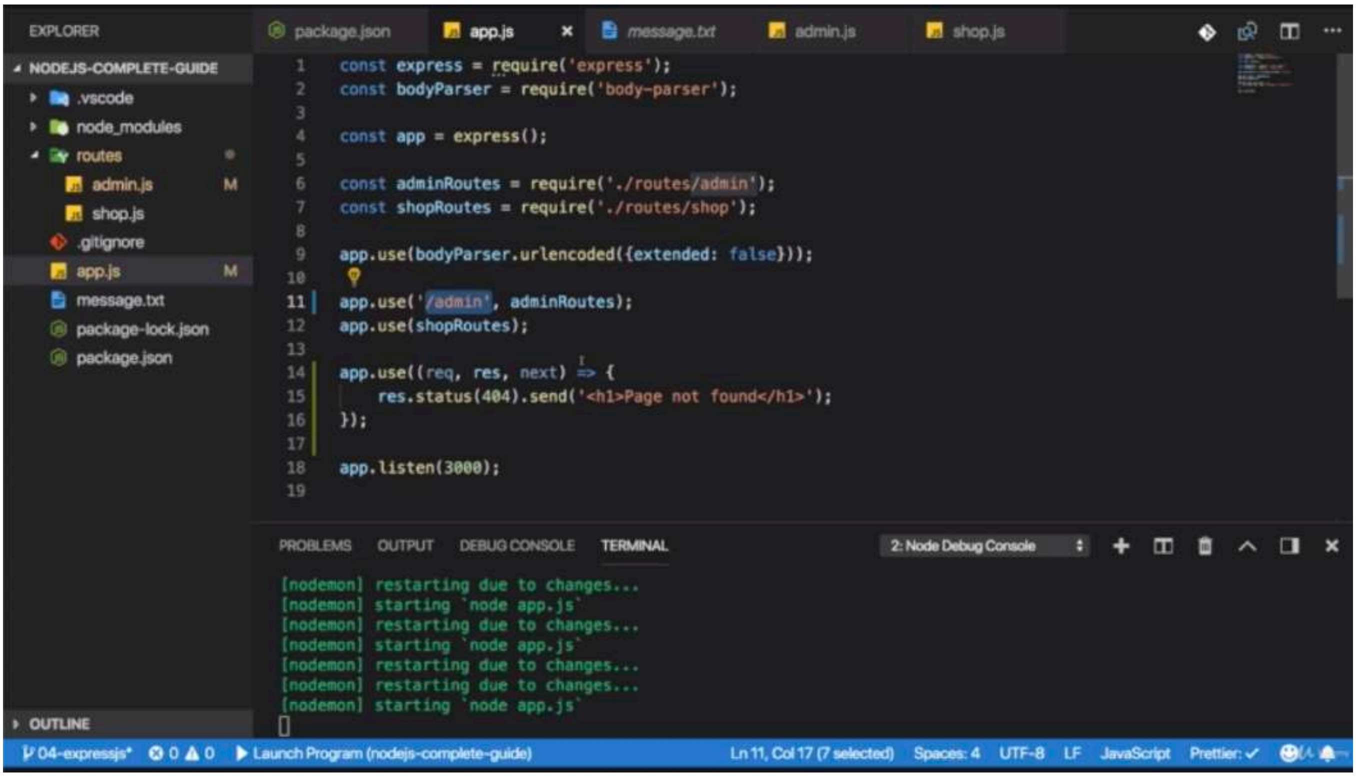
- and then go to the app.js file and add it here, so add that segment as a filter.
- only routes starting with '/admin' will go into the admin routes file.
![](images/67-filtering-paths-4.png)



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

const router = express.Router();

router.get('/add-product', (req, res, next) => {
  res.send(
    '<form action="/add-product" method="POST"><input type="text" name="title"><button type="su
  );
});

router.post('/add-product', (req, res, next) => {
  console.log(req.body);
  res.redirect('/');
});

module.exports = router;
```

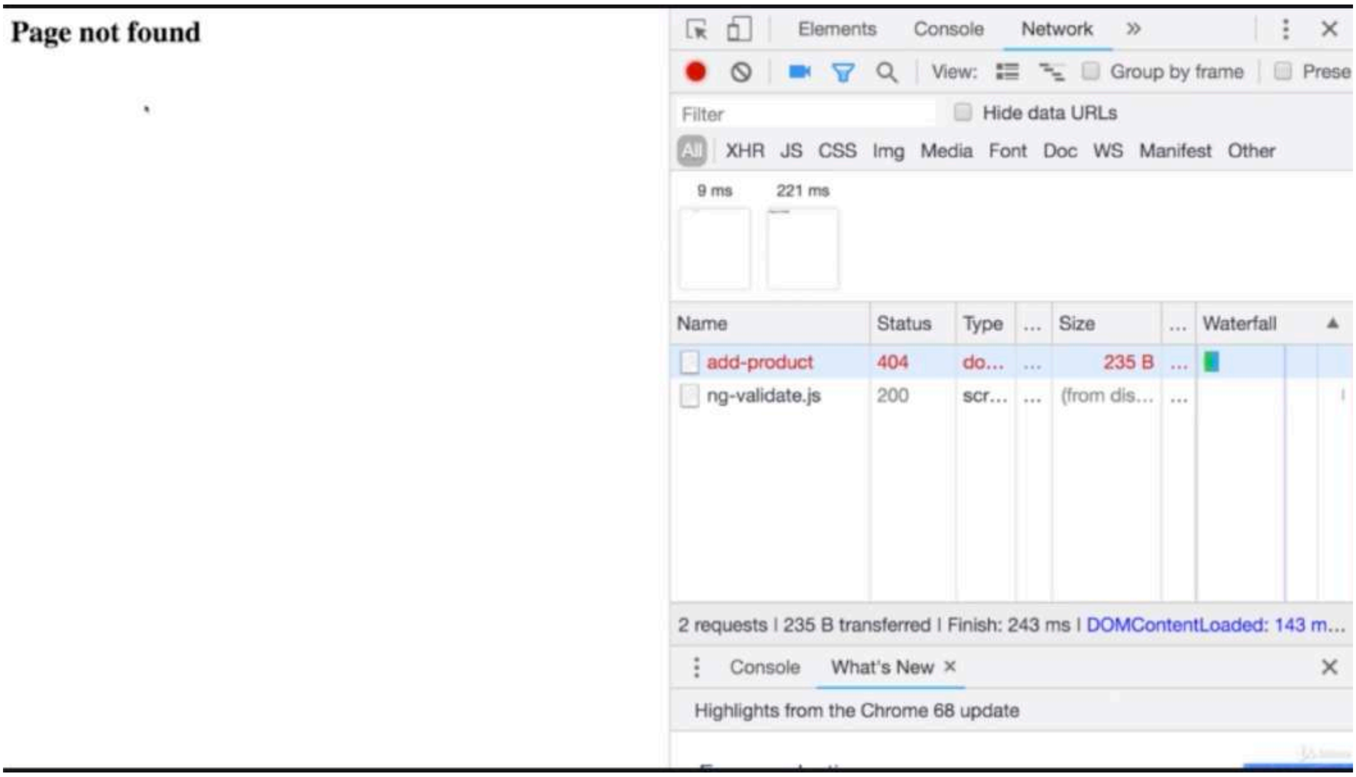
```
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
[nodemon] restarting due to changes...
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
```

- so now '/add-product' will match the '/admin/add-product' route because '/admin' was already stripped out.
![](images/67-filtering-paths-5.png)
![](images/67-filtering-paths-6.png)



```javascript
const express = require('express');
const bodyParser = require('body-parser');

const app = express();

const adminRoutes = require('./routes/admin');
const shopRoutes = require('./routes/shop');

app.use(bodyParser.urlencoded({extended: false}));

app.use('/admin', adminRoutes);
app.use(shopRoutes);

app.use((req, res, next) => {
    res.status(404).send('<h1>Page not found</h1>');
});

app.listen(3000);
```

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



**Page not found**

| Name | Status | Type | ... | Size | ... | Waterfall |
|---|---|---|---|---|---|---|
| add-product | 404 | do... | ... | 235 B | ... | |
| ng-validate.js | 200 | scr... | ... | (from dis... | ... | |

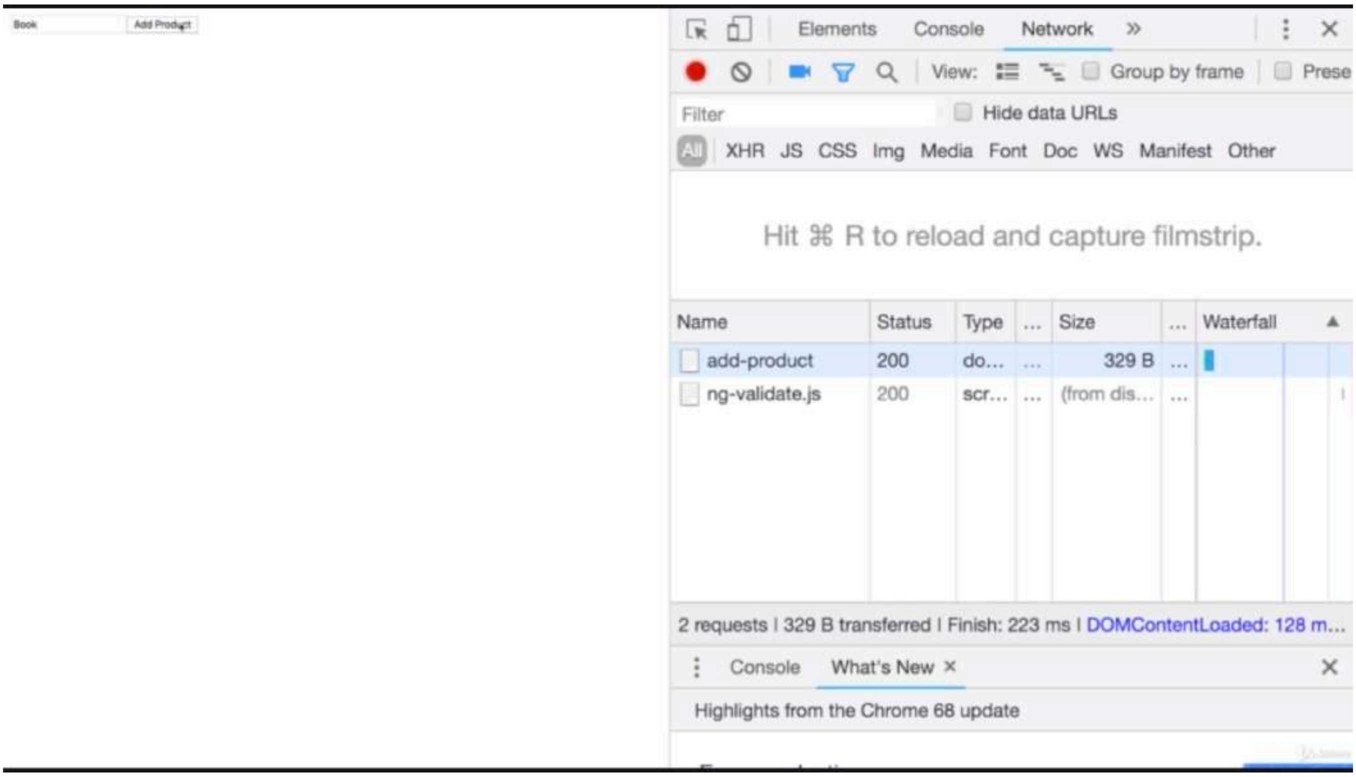
2 requests | 235 B transferred | Finish: 243 ms | DOMContentLoaded: 143 m...

Highlights from the Chrome 68 update

- if i go to just ‘/add-product’, we see ‘Page not found’ because it doesn’t exist anymore.

![](images/67-filtering-paths-7.png)

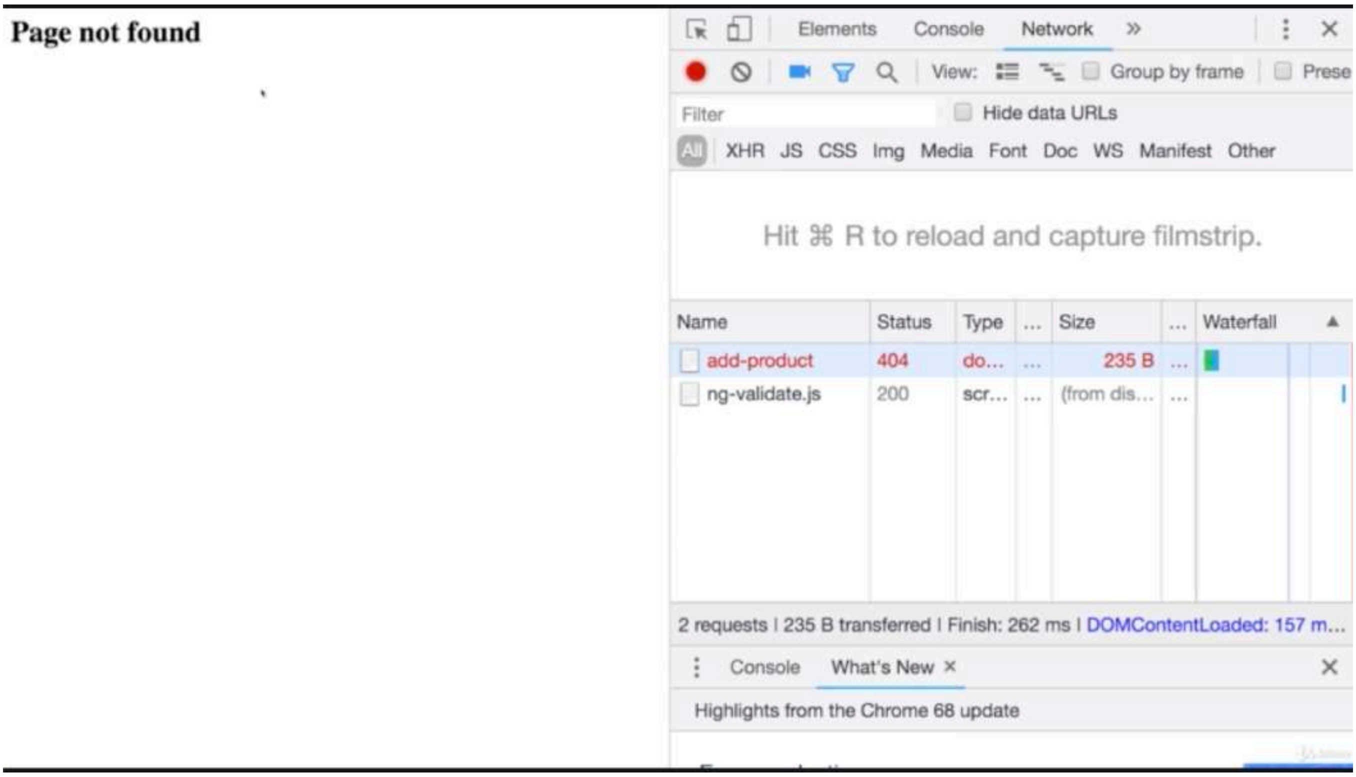
![](images/67-filtering-paths-8.png)



Book
Add Product
Elements Console Network »
View: Group by frame Prese
Filter Hide data URLs
All XHR JS CSS Img Media Font Doc WS Manifest Other
Hit ⌘ R to reload and capture filmstrip.
Console What's New ×
Highlights from the Chrome 68 update

| Name | Status | Type | ... | Size | ... | Waterfall |
|---|---|---|---|---|---|---|
| add-product | 200 | do... | ... | 329 B | ... | |
| ng-validate.js | 200 | scr... | ... | (from dis... | ... | |

2 requests | 329 B transferred | Finish: 223 ms | DOMContentLoaded: 128 m...



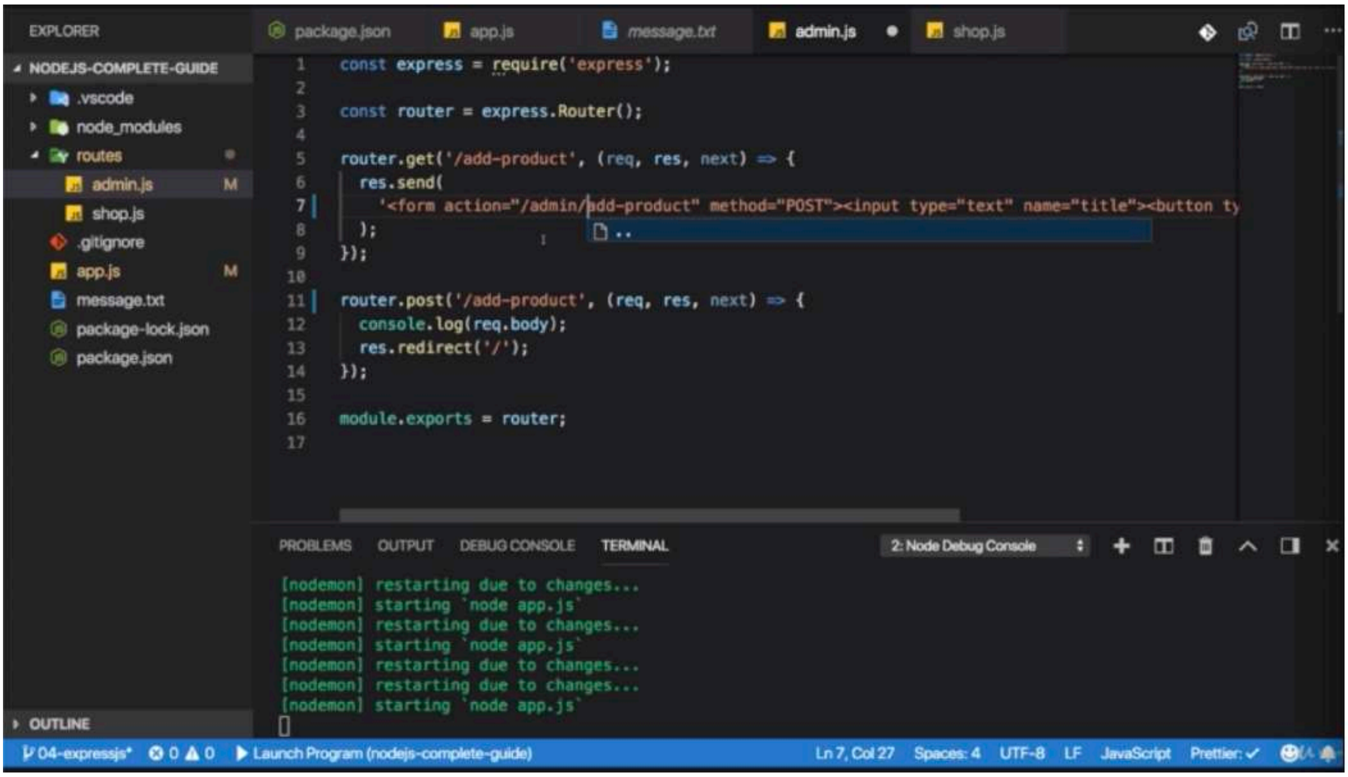
Page not found

Elements Console Network »
View: Group by frame Prese
Filter Hide data URLs
All XHR JS CSS Img Media Font Doc WS Manifest Other
Hit ⌘ R to reload and capture filmstrip.
Console What's New ×
Highlights from the Chrome 68 update

| Name | Status | Type | ... | Size | ... | Waterfall |
|---|---|---|---|---|---|---|
| add-product | 404 | do... | ... | 235 B | ... | |
| ng-validate.js | 200 | scr... | ... | (from dis... | ... | |

2 requests | 235 B transferred | Finish: 262 ms | DOMContentLoaded: 157 m...

- but if we go to ‘admin/add-product’, here is the form.
- if i fill out input, i get 'page not found’
![](images/67-filtering-paths-9.png)



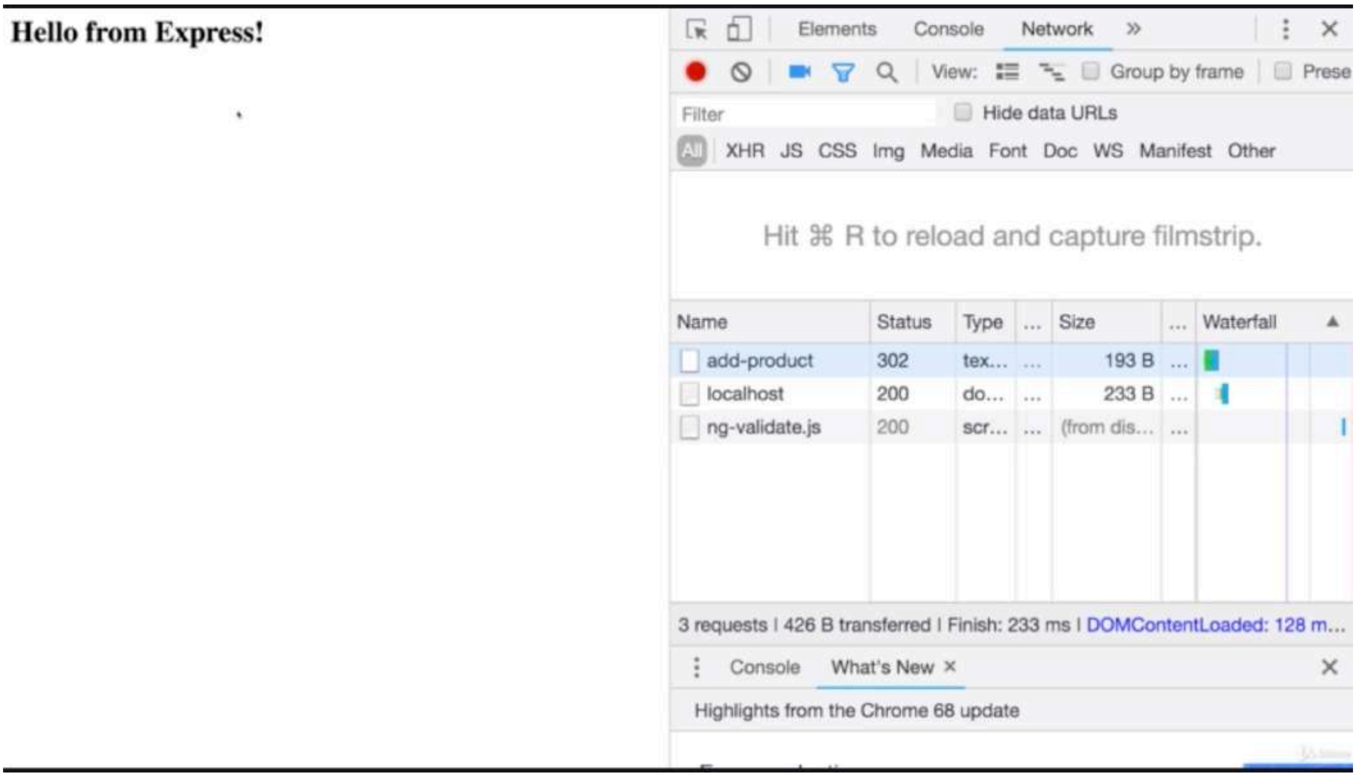
```
const express = require('express');

const router = express.Router();

router.get('/add-product', (req, res, next) => {
  res.send(
    '<form action="/admin/add-product" method="POST"><input type="text" name="title"><button ty
  );
});

router.post('/add-product', (req, res, next) => {
  console.log(req.body);
  res.redirect('/');
});

module.exports = router;
```

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

- so this should be done like below. need to be added '/admin' in 'send()' because we wanna reach that route which is the admin.js  file which is only reachable through requests that have /admin at the beginning.

```
//./routes/admin.js

router.get('/add-product', (req, res, next) => {
    res.send('<form action="/admin/add-product" method="POST"><input type="text" name="title"><button type="submit">Add Product</</button></form>')
})
```

![](images/67-filtering-paths-10.png)



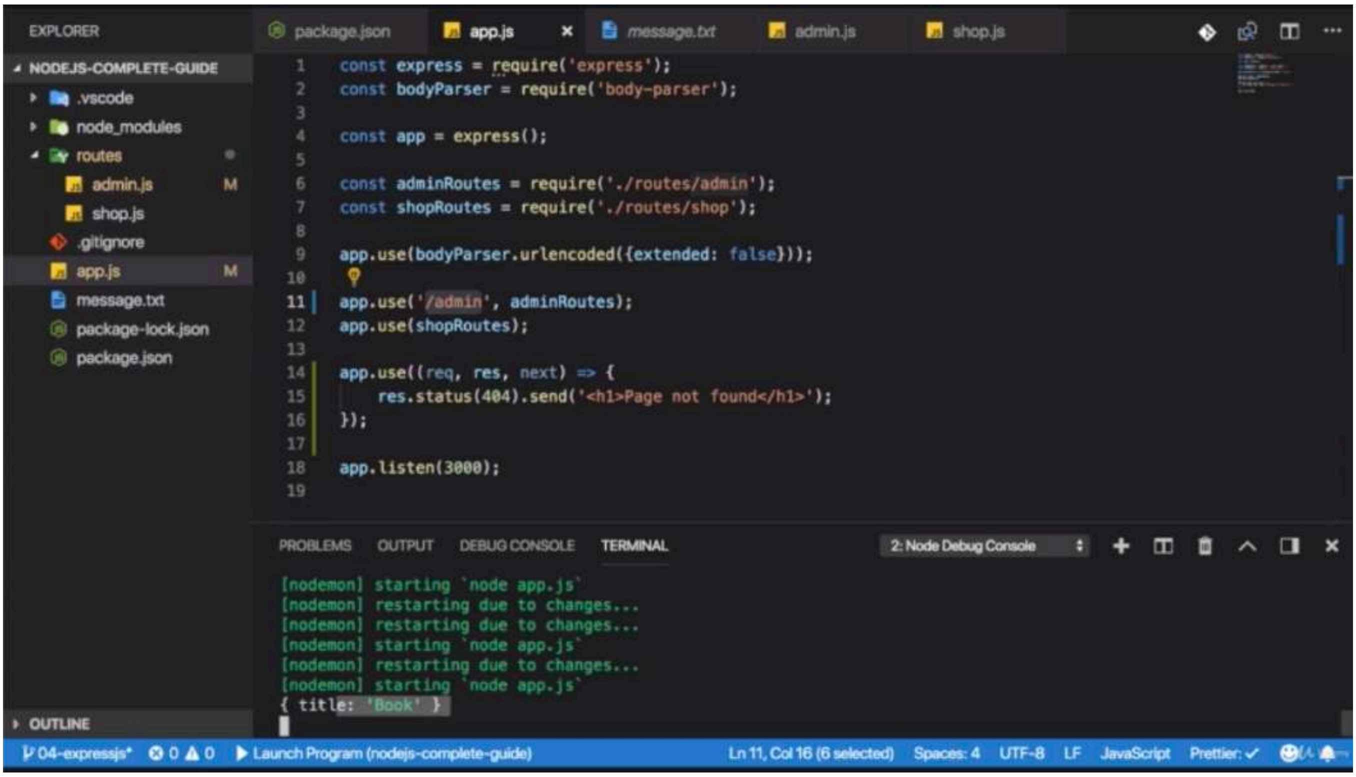
Hello from Express!

| Name | Status | Type | ... | Size | ... | Waterfall |
|---|---|---|---|---|---|---|
| add-product | 302 | tex... | ... | 193 B | ... | |
| localhost | 200 | do... | ... | 233 B | ... | |
| ng-validate.js | 200 | scr... | ... | (from dis... | ... | |

3 requests | 426 B transferred | Finish: 233 ms | DOMContentLoaded: 128 m...

Highlights from the Chrome 68 update

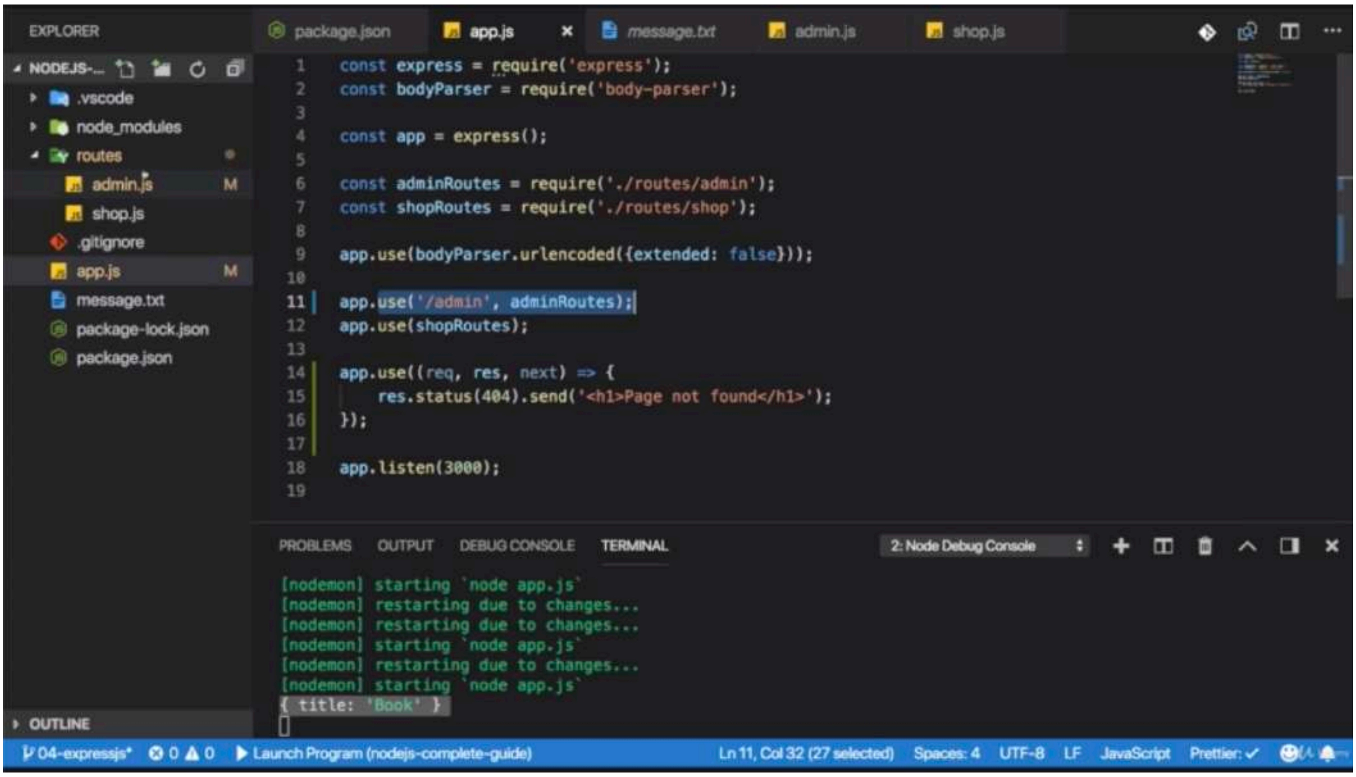
- so go to '/admin/add-product' and fill out input field, the it's redirected.
![](images/67-filtering-paths-11.png)



```javascript
const express = require('express');
const bodyParser = require('body-parser');

const app = express();

const adminRoutes = require('./routes/admin');
const shopRoutes = require('./routes/shop');

app.use(bodyParser.urlencoded({extended: false}));

app.use('/admin', adminRoutes);
app.use(shopRoutes);

app.use((req, res, next) => {
    res.status(404).send('<h1>Page not found</h1>');
});

app.listen(3000);
```

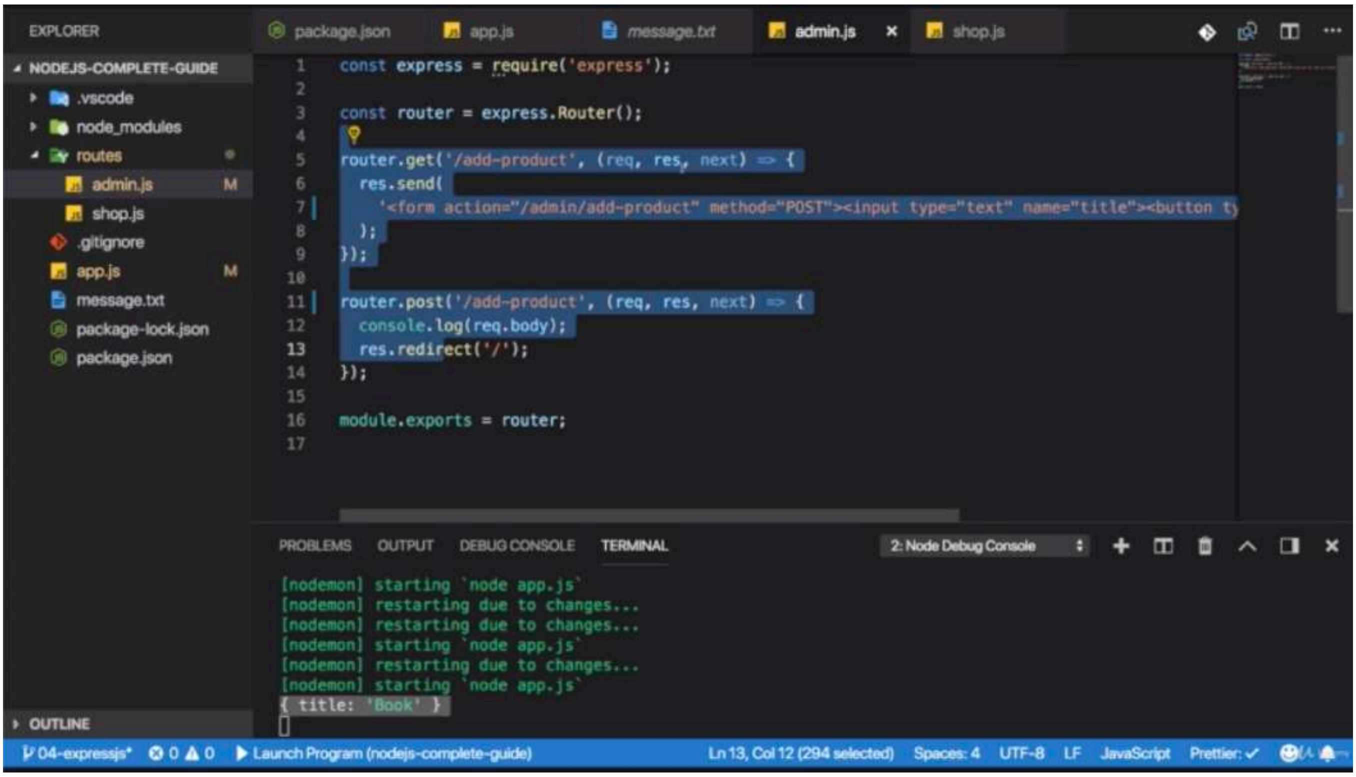
```
[nodemon] starting `node app.js`
[nodemon] restarting due to changes...
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
{ title: 'Book' }
```

- and also see that we are logging this here.
![](images/67-filtering-paths-12.png)
![](images/67-filtering-paths-13.png)



```javascript
const express = require('express');
const bodyParser = require('body-parser');

const app = express();

const adminRoutes = require('./routes/admin');
const shopRoutes = require('./routes/shop');

app.use(bodyParser.urlencoded({extended: false}));

app.use('/admin', adminRoutes);
app.use(shopRoutes);

app.use((req, res, next) => {
    res.status(404).send('<h1>Page not found</h1>');
});

app.listen(3000);
```

```
[nodemon] starting `node app.js`
[nodemon] restarting due to changes...
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
{ title: 'Book' }
```



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

const router = express.Router();

router.get('/add-product', (req, res, next) => {
  res.send(
    '<form action="/admin/add-product" method="POST"><input type="text" name="title"><button ty
  );
});

router.post('/add-product', (req, res, next) => {
  console.log(req.body);
  res.redirect('/');
});

module.exports = router;
```

```
[nodemon] starting `node app.js`
[nodemon] restarting due to changes...
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
{ title: 'Book' }
```

- so this filtering mechanism here in app.js allow us to put a common starting segment for our path which all routes in a given file use to outsource that into this app.js so that we don't have to repeat it for all the routes here.

```
//app.js

const express = require('express');
const bodyParser = require('body-parser');

const app = express()

const adminRoutes = require('./routes/admin');
const shopRoutes = require('./routes/shop');

app.use(bodyParser.urlencoded({extended: false}))

app.use('/admin', adminRoutes);
app.use(shopRoutes);

app.use((req, res, next) => {
    res.status(404).send('<h1>Page not found</h1>')
})

app.listen(3000);
```

```
//./routes/admin.js

const express = require('express')

const router = express.Router();


// /admin/add-product => GET
router.get('/add-product', (req, res, next) => {
    res.send('<form action="/admin/add-product" method="POST"><input type="text" name="title"><button type="submit">Add Product</</button></form>')
})


// /admin/add-product => POST
router.post('/add-product', (req, res, next) => {
    console.log(req.body)
    res.redirect('/');
})

module.exports = router;
```

```
//./routes/shop.js

const express = require('express');

const router = express.Router();

router.get('/', (req, res, next) => {
    res.send('<h1>Hello from Express!</h1>')
})

module.exports = router
```

# * Chapter 68: Creating HTML Pages

1. update
- shop.html
- add-product.html

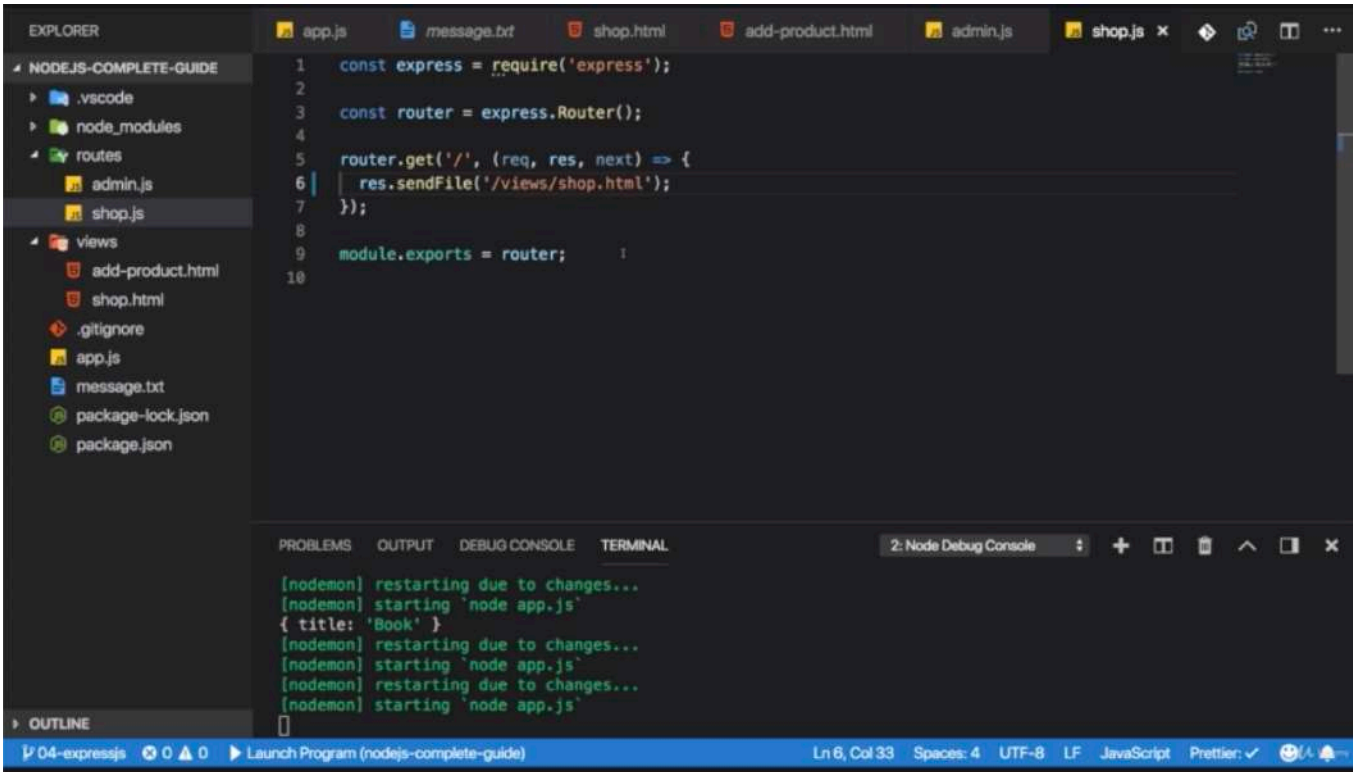
```
<!--./views/shop.html-->

<!--
    it's a file i wanna serve for users visiting just '/'
-->

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Add Product</title>
</head>

<body>
    <header>
        <nav>
            <ul>
                <li><a href="/">Shop</a></li>
                <li><a href="/admin/add-product">Add Product</a></li>
            </ul>
        </nav>
    </header>

    <main>
        <h1>My Products</h1>
        <p>List of all the products...</p>
    </main>
</body>

</html>
```

```
<!--./views/add-product.html-->

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Add Product</title>
</head>

<body>
    <header>
        <nav>
            <ul>
                <li><a href="/">Shop</a></li>
                <li><a href="/admin/add-product">Add Product</a></li>
            </ul>
        </nav>
    </header>

    <main>
        <form action="/add-product" method="POST">
            <input type="text" name="title">
            <button type="submit">Add Product</button>
        </form>
    </main>
</body>

</html>
```

# * Chapter 69: Serving HTML Pages

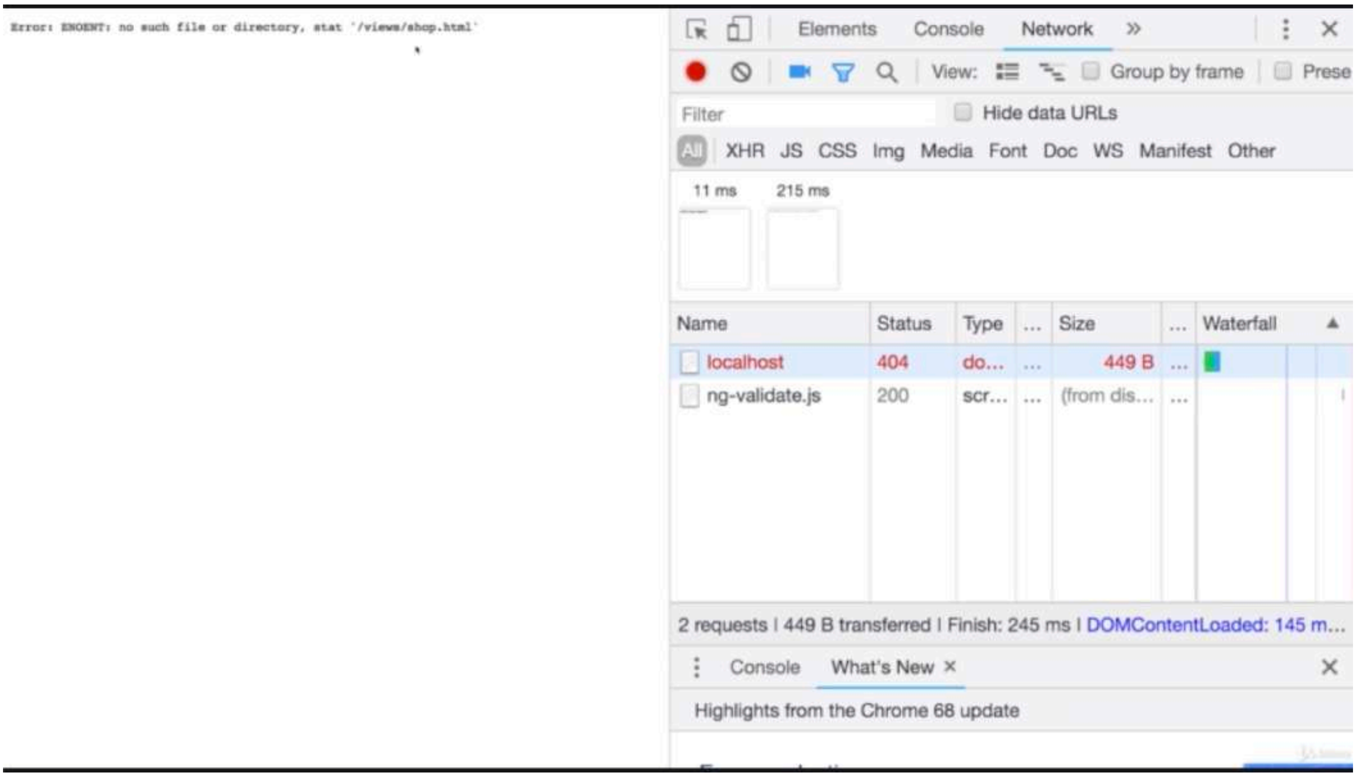
1. update
- admin.js
- shop.js

![](images/69-serving-html-pages-1.png)
![](images/69-serving-html-pages-2.png)
![](images/69-serving-html-pages-3.png)
![](images/69-serving-html-pages-4.png)



```javascript
const express = require('express');

const router = express.Router();

router.get('/', (req, res, next) => {
  res.sendFile('/views/shop.html');
});

module.exports = router;
```

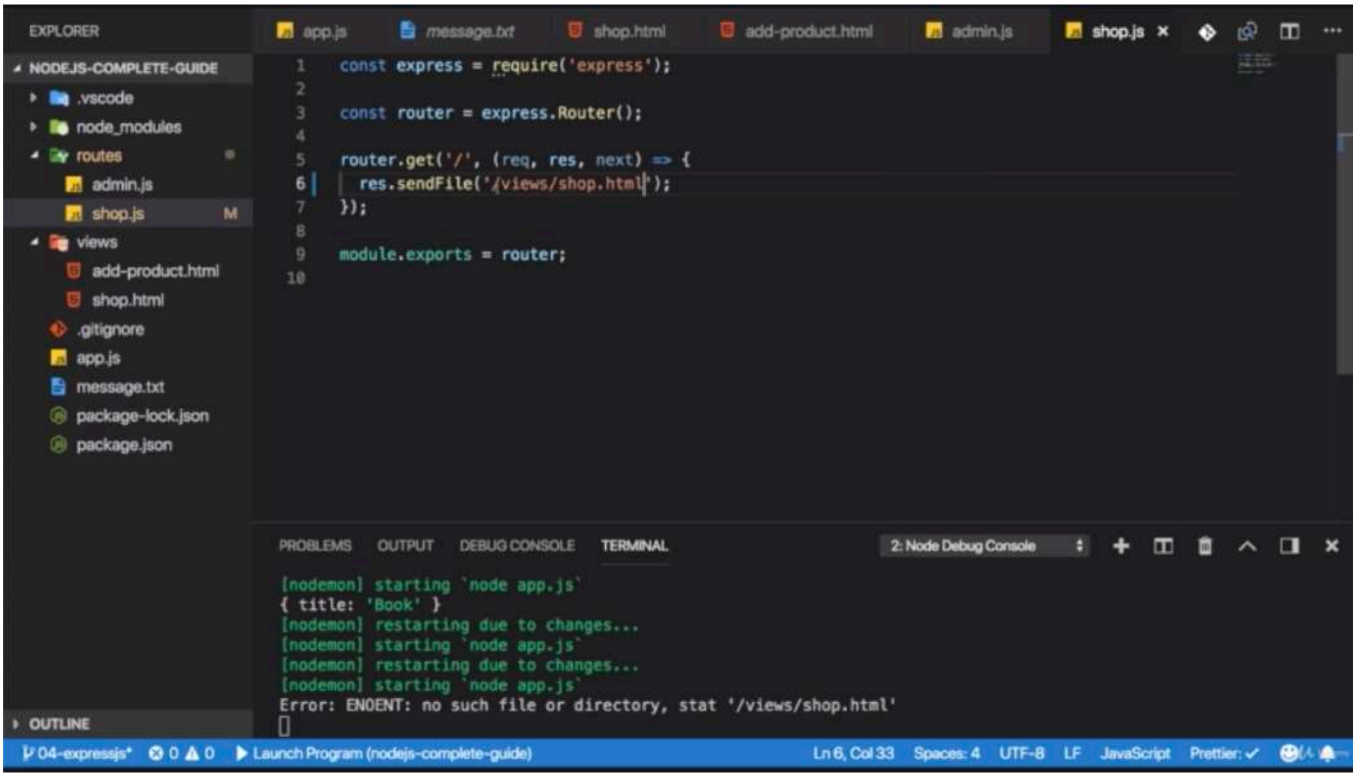
```
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
{ title: 'Book' }
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
```



```
Error: ENOENT: no such file or directory, stat '/views/shop.html'
```

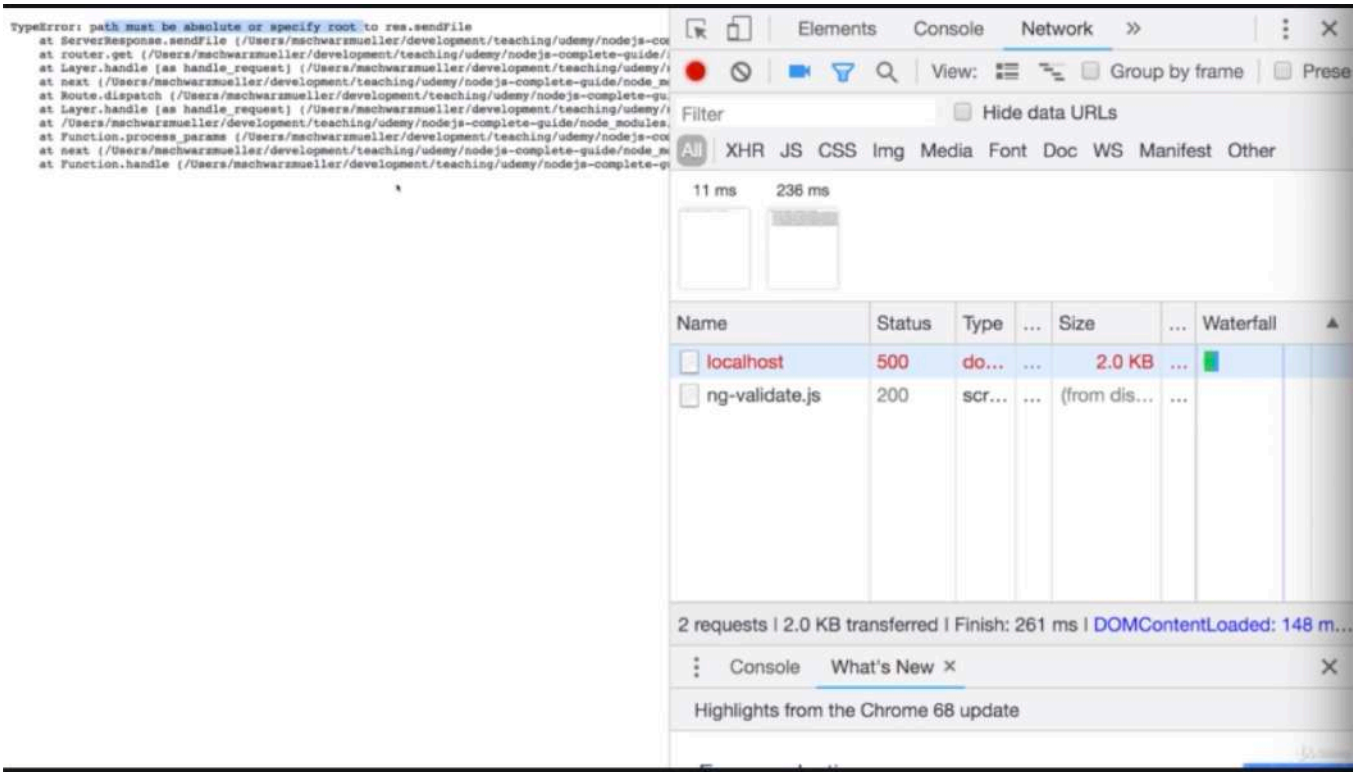
| Name | Status | Type | ... | Size | ... | Waterfall |
|---|---|---|---|---|---|---|
| localhost | 404 | do... | ... | 449 B | ... | 🟩 |
| ng-validate.js | 200 | scr... | ... | (from dis... | ... | |

2 requests | 449 B transferred | Finish: 245 ms | DOMContentLoaded: 145 m...



```javascript
const express = require('express');

const router = express.Router();

router.get('/', (req, res, next) => {
  res.sendFile('/views/shop.html');
});

module.exports = router;
```

```
[nodemon] starting `node app.js`
{ title: 'Book' }
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
Error: ENOENT: no such file or directory, stat '/views/shop.html'
```

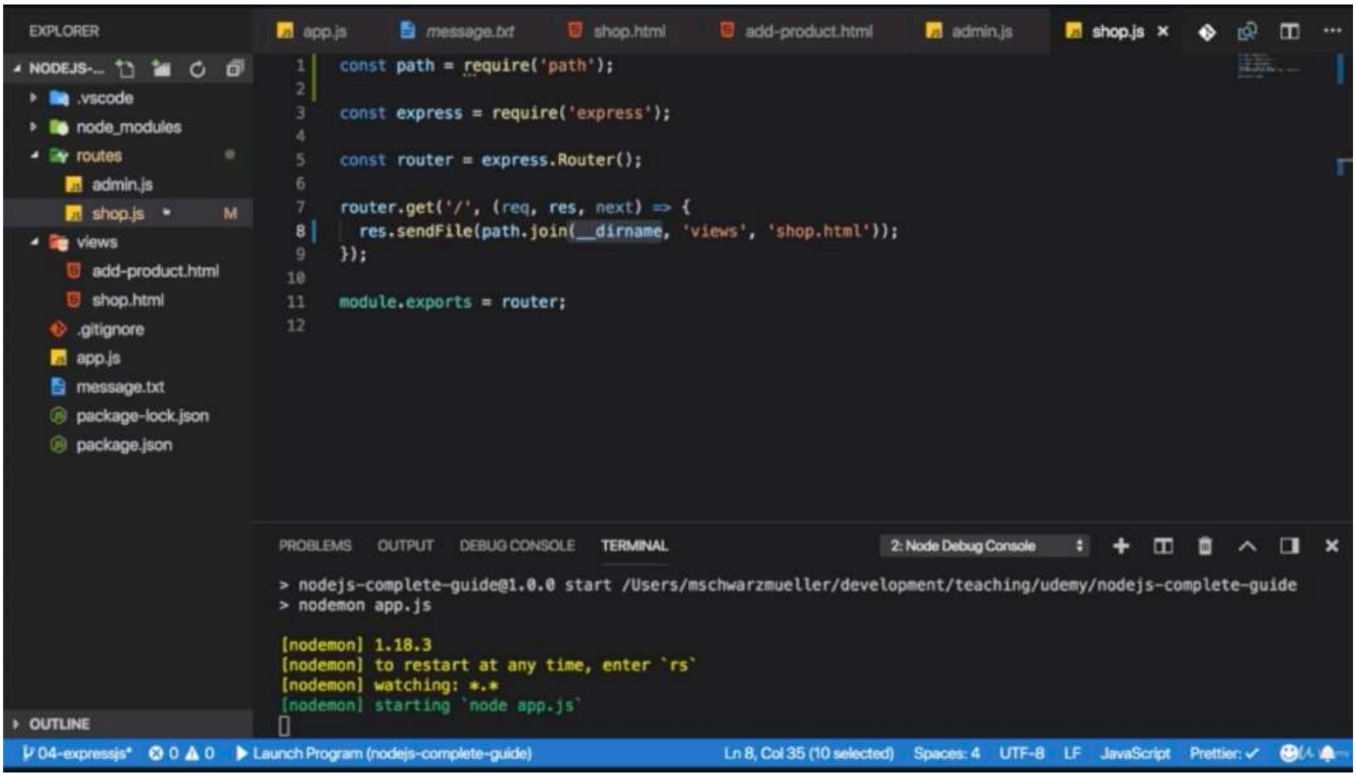


```
TypeError: path must be absolute or specify root to res.sendFile
    at ServerResponse.sendFile (/Users/mschwarzmueller/development/teaching/udemy/nodejs-co
    at router.get (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/
    at Layer.handle [as handle_request] (/Users/mschwarzmueller/development/teaching/udemy/
    at next (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/node_m
    at Route.dispatch (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-gu
    at Layer.handle [as handle_request] (/Users/mschwarzmueller/development/teaching/udemy/
    at /Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/node_modules
    at Function.process_params (/Users/mschwarzmueller/development/teaching/udemy/nodejs-co
    at next (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/node_m
    at Function.handle (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-g
```

| Name | Status | Type | ... | Size | ... | Waterfall |
|---|---|---|---|---|---|---|
| localhost | 500 | do... | ... | 2.0 KB | ... | |
| ng-validate.js | 200 | scr... | ... | (from dis... | ... | |

2 requests | 2.0 KB transferred | Finish: 261 ms | DOMContentLoaded: 148 m...

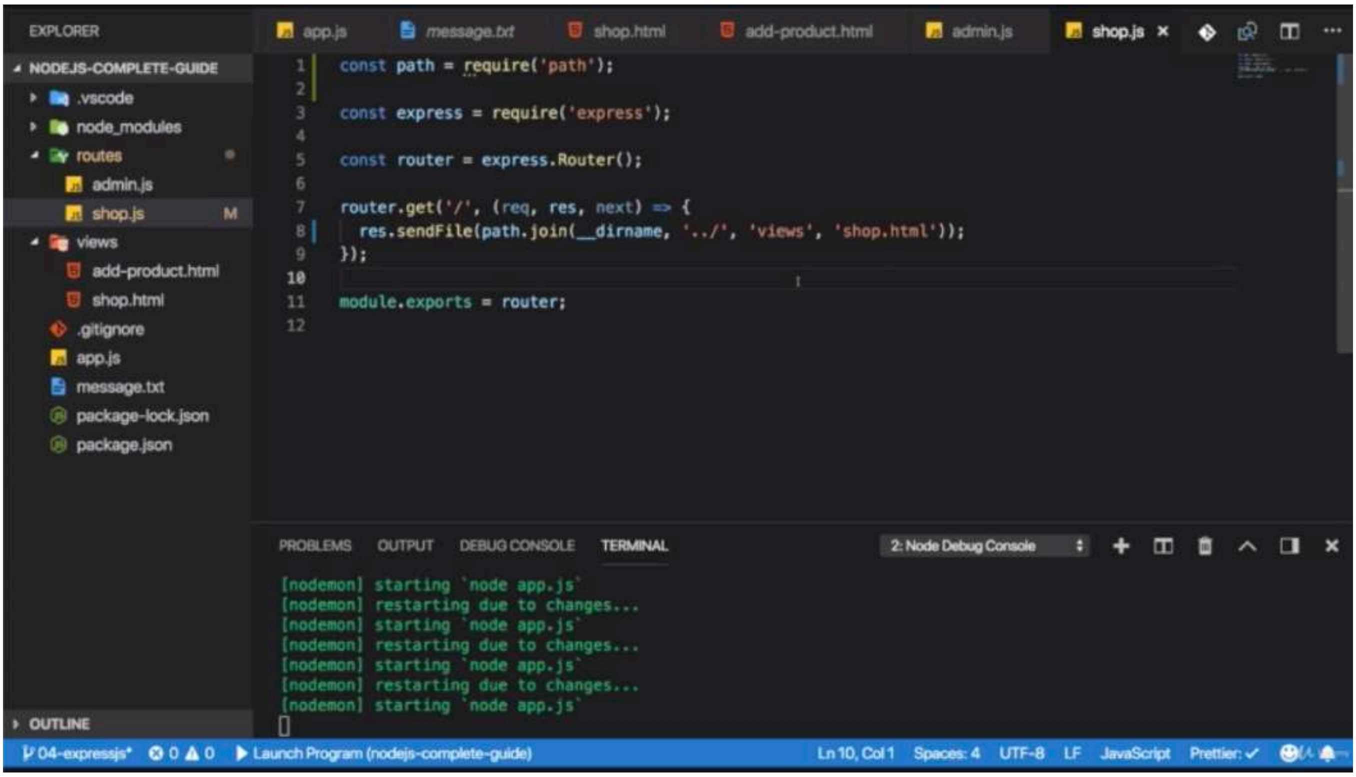
- i don’t see anything because this path ‘/views/shop.html’ is incorrect. if we now reload, path must be absolute is the error we get.
- absolute path would be correct but slash like this refers to our root folder on our operating system. not the project folder.
![](images/69-serving-html-pages-5.png)



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

const express = require('express');

const router = express.Router();

router.get('/', (req, res, next) => {
  res.sendFile(path.join(__dirname, 'views', 'shop.html'));
});

module.exports = router;
```

```
> 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`
```

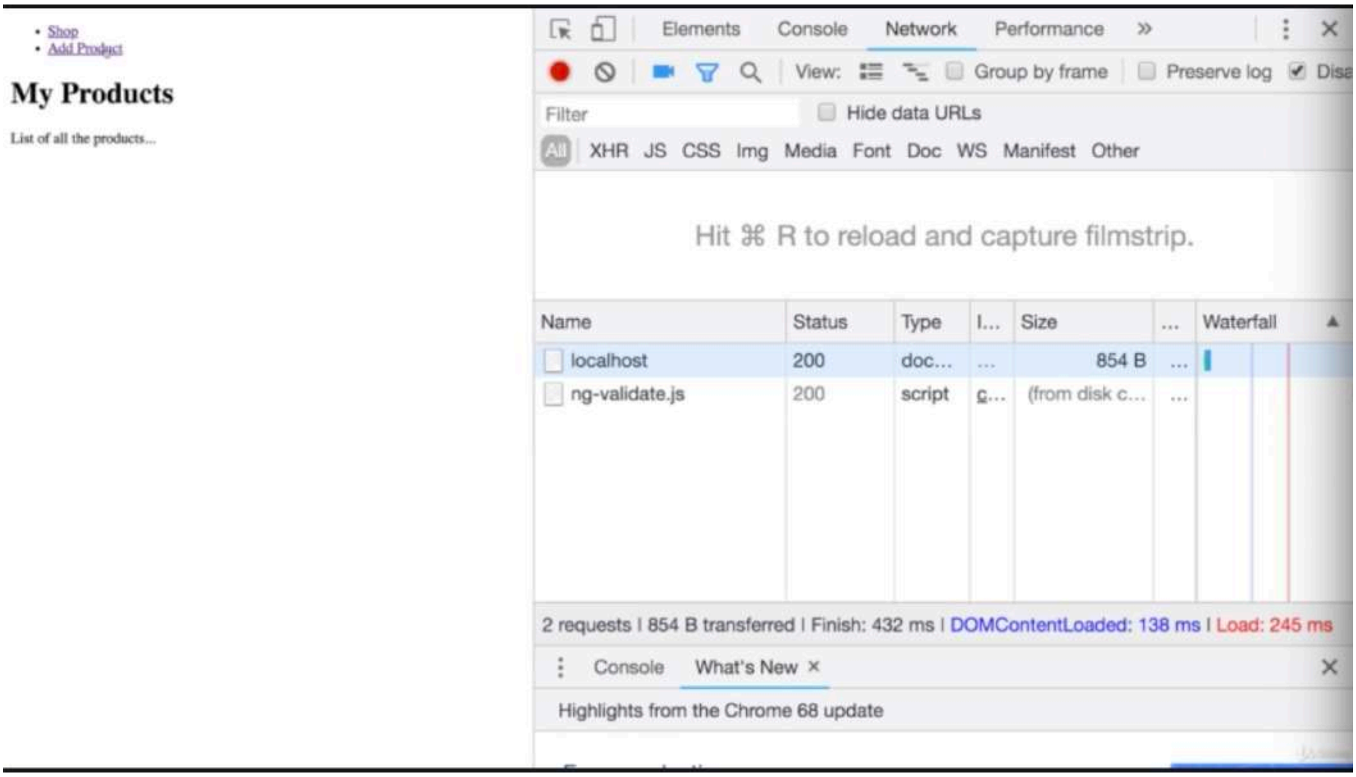
- so in order to construct the path to this directory and this file ultimately, we can use a feature provided by node.js called 'path' the core module.
- 'join()' yields us a path at the end, it returns a path but it construct this path by concatenating the different segment.
- the first segment we should pass here is then a global variable made available by node.js and that is '__dirname' which is a global variable to holds the absolute path on our operating system. to this project folder and now we can add a , 'views' because the first segment is the path to this whole project folder __dirname. and the next segment is that we wanna go into the views folder and then the 3rd segment will be our file. so here shop.html and don't add / because we use path.join() not because of the absolute path, we could build this with __dirname and then concatenating this manually too.
- but we are using path.join() because this will automatically build the path in a way that works on both linux and windows systems. on linux, you have paths like '/user/products' but in windows backslash is used like '\user\products' adding path.
- therefore if you manually construct this with slashes, it would not run on windows and the other way around.
- path.join() detects operating system you are running on and then automatically build a correct path.
![](images/69-serving-html-pages-6.png)



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

const express = require('express');

const router = express.Router();

router.get('/', (req, res, next) => {
  res.sendFile(path.join(__dirname, '../', 'views', 'shop.html'));
});

module.exports = router;
```

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

- but actually __dirname will point routes folder because '__dirname' gives us the path to a file in which we use it and we are using it in the shop.js file in the routes folder. but views folder is located in a sibling folder to routes folder.
- so we use '../' and this simply means go up 1 level. so this will now build a path where it first goes into the folder of these files, so into routes folder then it goes up 1 level then into views folder.
![](images/69-serving-html-pages-7.png)



- Shop
- Add Product

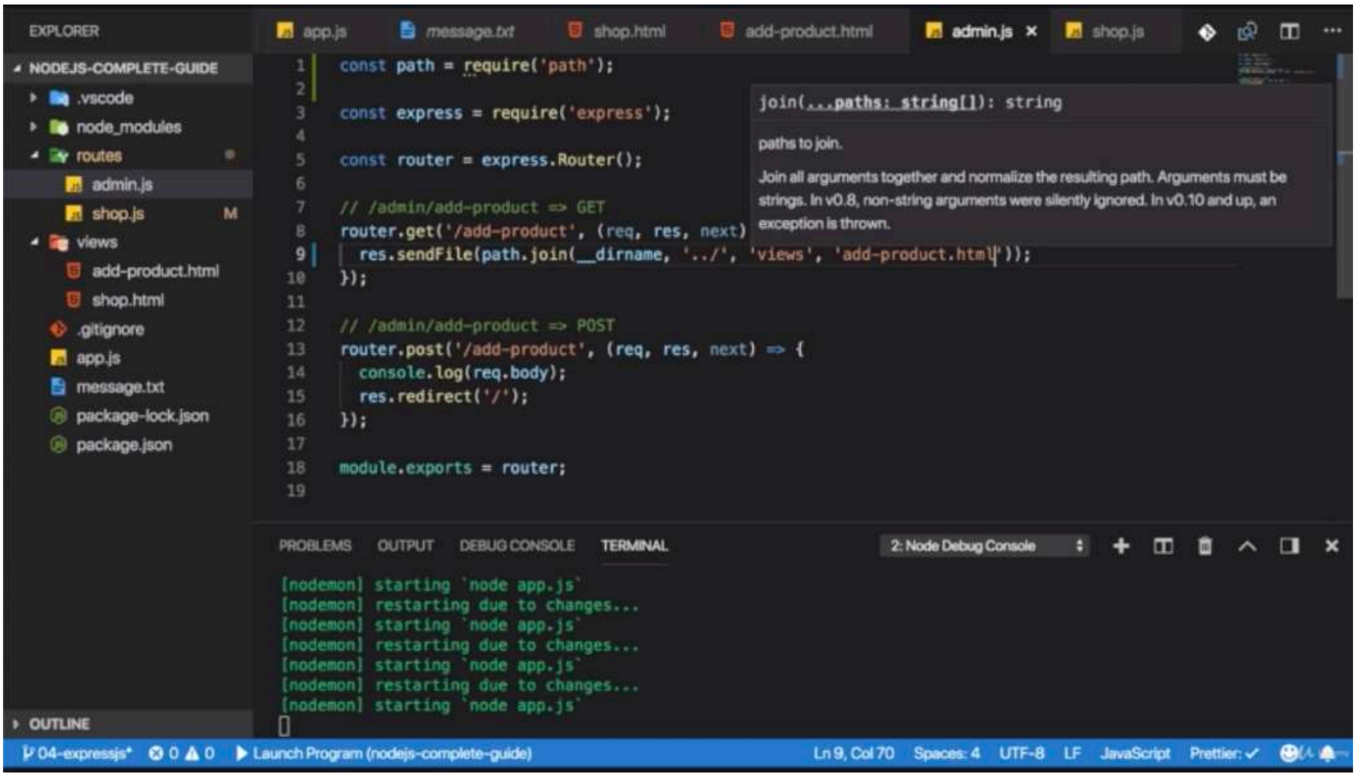
My Products

List of all the products...

| Name | Status | Type | I... | Size | ... | Waterfall |
|---|---|---|---|---|---|---|
| localhost | 200 | doc... | ... | 854 B | ... | |
| ng-validate.js | 200 | script | c... | (from disk c... | ... | |

2 requests | 854 B transferred | Finish: 432 ms | DOMContentLoaded: 138 ms | Load: 245 ms

- so if we go to 'localhost:3000/' again, we see that html file being served.
![](images/69-serving-html-pages-8.png)
![](images/69-serving-html-pages-9.png)
![](images/69-serving-html-pages-10.png)



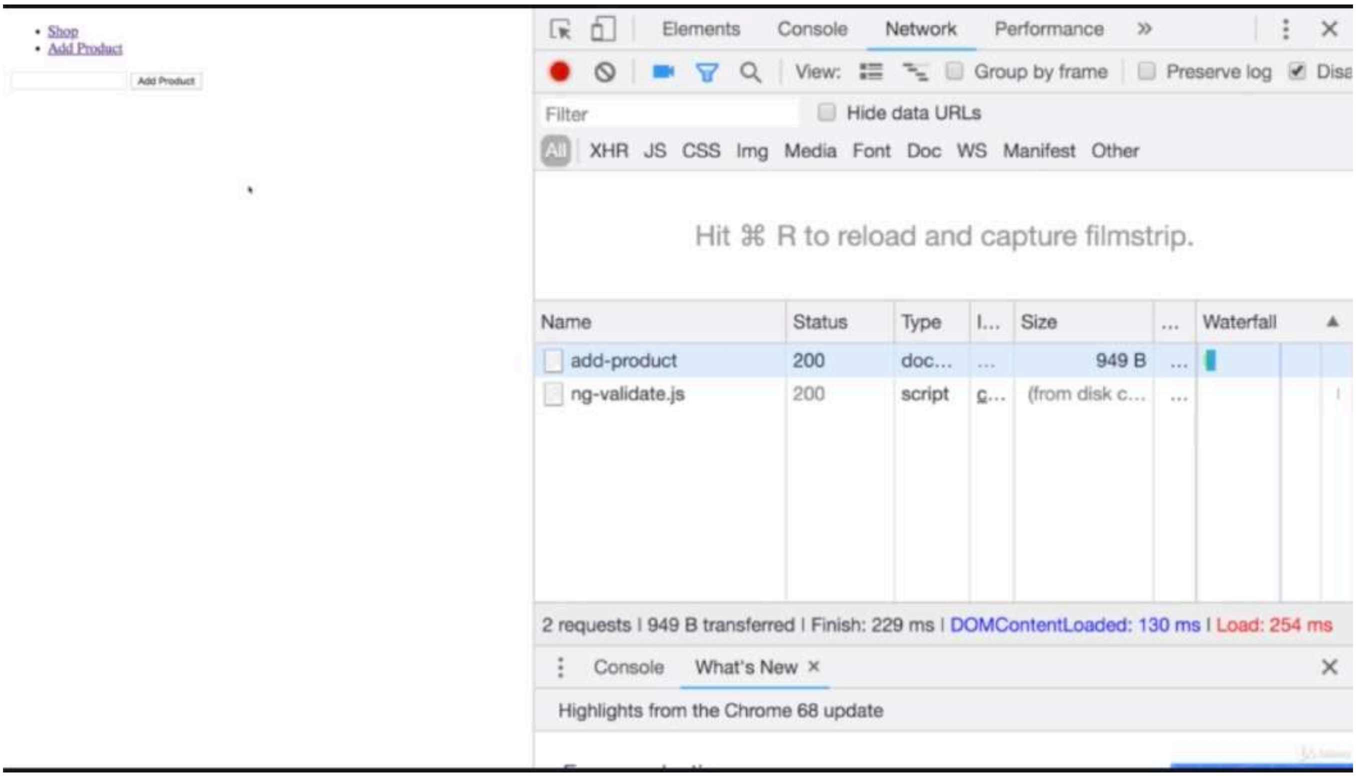
```
const path = require('path');

const express = require('express');

const router = express.Router();

// /admin/add-product => GET
router.get('/add-product', (req, res, next)
  res.sendFile(path.join(__dirname, '../', 'views', 'add-product.html'));
});

// /admin/add-product => POST
router.post('/add-product', (req, res, next) => {
  console.log(req.body);
  res.redirect('/');
});

module.exports = router;
```

```
join(...paths: string[]): string
```

paths to join.

Join all arguments together and normalize the resulting path. Arguments must be strings. In v0.8, non-string arguments were silently ignored. In v0.10 and up, an exception is thrown.

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



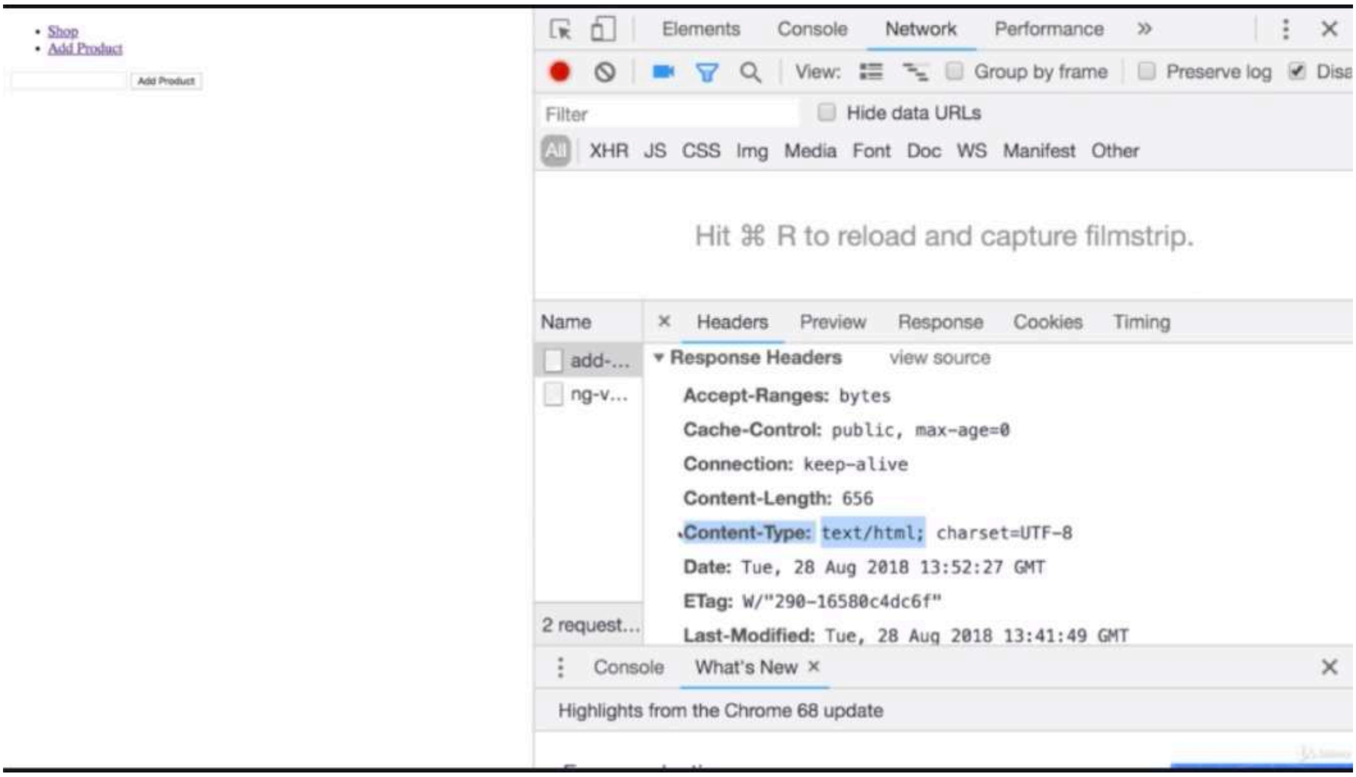
- Shop
- Add Product

Hit ⌘ R to reload and capture filmstrip.

| Name | Status | Type | I... | Size | ... | Waterfall |
|---|---|---|---|---|---|---|
| add-product | 200 | doc... | ... | 949 B | ... | |
| ng-validate.js | 200 | script | c... | (from disk c... | ... | |

2 requests | 949 B transferred | Finish: 229 ms | DOMContentLoaded: 130 ms | Load: 254 ms

Highlights from the Chrome 68 update


• Shop
• Add Product
Add Product
Elements
Console
Network
Performance
View:
Group by frame
Preserve log
Disa
Filter
Hide data URLs
All XHR JS CSS Img Media Font Doc WS Manifest Other
Hit ⌘ R to reload and capture filmstrip.
Name
Headers Preview Response Cookies Timing
add-...
ng-v...
2 request...
Console
What's New
Highlights from the Chrome 68 update

▾ Response Headers view source

Accept-Ranges: bytes
Cache-Control: public, max-age=0
Connection: keep-alive
Content-Length: 656
Content-Type: text/html; charset=UTF-8
Date: Tue, 28 Aug 2018 13:52:27 GMT
ETag: W/"290-16580c4dc6f"
Last-Modified: Tue, 28 Aug 2018 13:41:49 GMT

- if we go to '/admin/add-product', we see this page too.

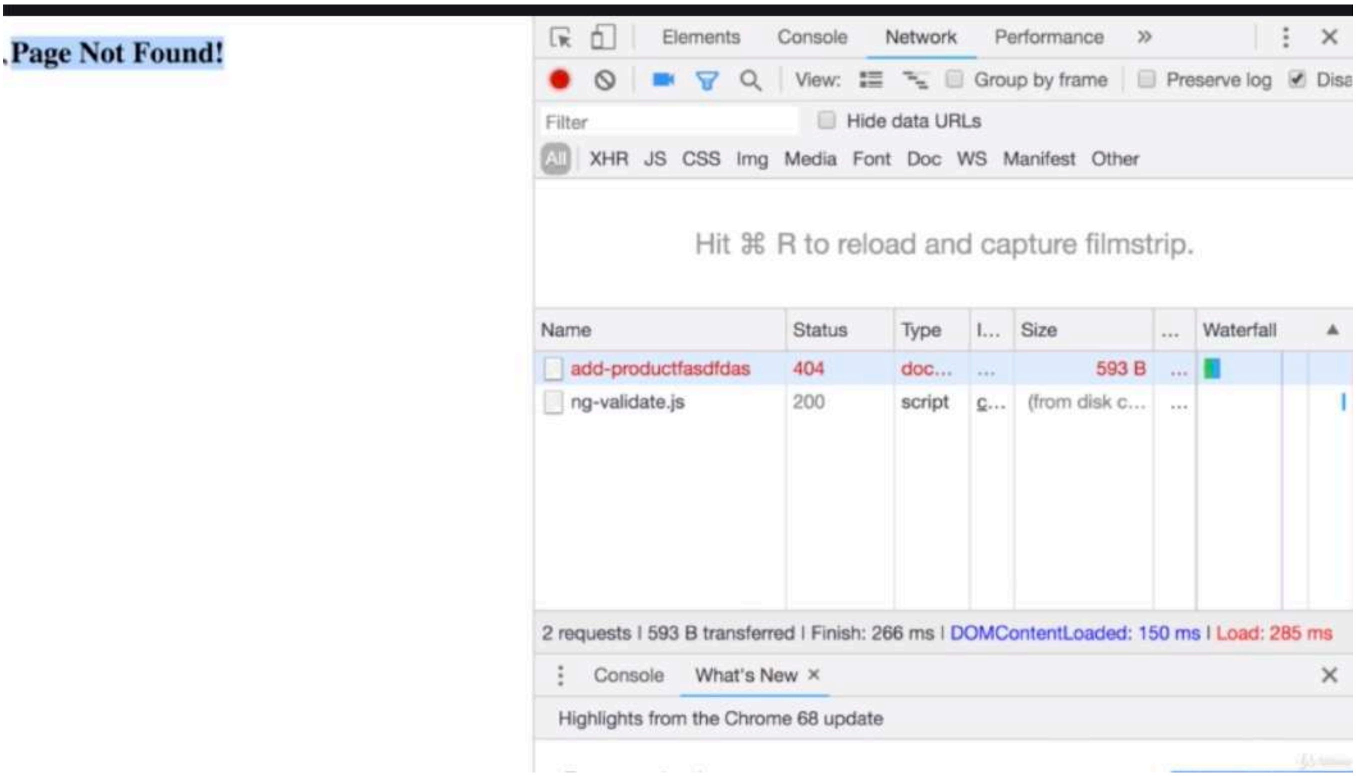
```
//./routes/admin.js
const path = require('path')

const express = require('express')

const router = express.Router();


// /admin/add-product => GET
router.get('/add-product', (req, res, next) => {
    res.sendFile(path.join(__dirname, '../', 'views', 'add-product.html'))
})


// /admin/add-product => POST
router.post('/add-product', (req, res, next) => {
    console.log(req.body)
    res.redirect('/');
})

module.exports = router;
```

```
//./routes/shop.js
const path = require('path');

const express = require('express');

const router = express.Router();

router.get('/', (req, res, next) => {
    res.sendFile(path.join(__dirname, '../', 'views', 'shop.html'))
})

```

```
module.exports = router
```

# * Chapter 70: Returning A 404 Page

1. update
- 404.html
- app.js
![](images/70-returning-a-404-page-1.png)



**Page Not Found!**

Hit ⌘ R to reload and capture filmstrip.

| Name | Status | Type | I... | Size | ... | Waterfall |
|---|---|---|---|---|---|---|
| add-productfasdfdas | 404 | doc... | ... | 593 B | ... | |
| ng-validate.js | 200 | script | c... | (from disk c... | ... | |

2 requests | 593 B transferred | Finish: 266 ms | DOMContentLoaded: 150 ms | Load: 285 ms

Highlights from the Chrome 68 update

```
//app.js

const path = require('path')

const express = require('express');
const bodyParser = require('body-parser');

const app = express()

const adminRoutes = require('./routes/admin');
const shopRoutes = require('./routes/shop');

app.use(bodyParser.urlencoded({extended: false}))

app.use('/admin', adminRoutes);
app.use(shopRoutes);

app.use((req, res, next) => {
    res.status(404).sendFile(path.join(__dirname, 'views', '404.html'))
})

app.listen(3000);
```

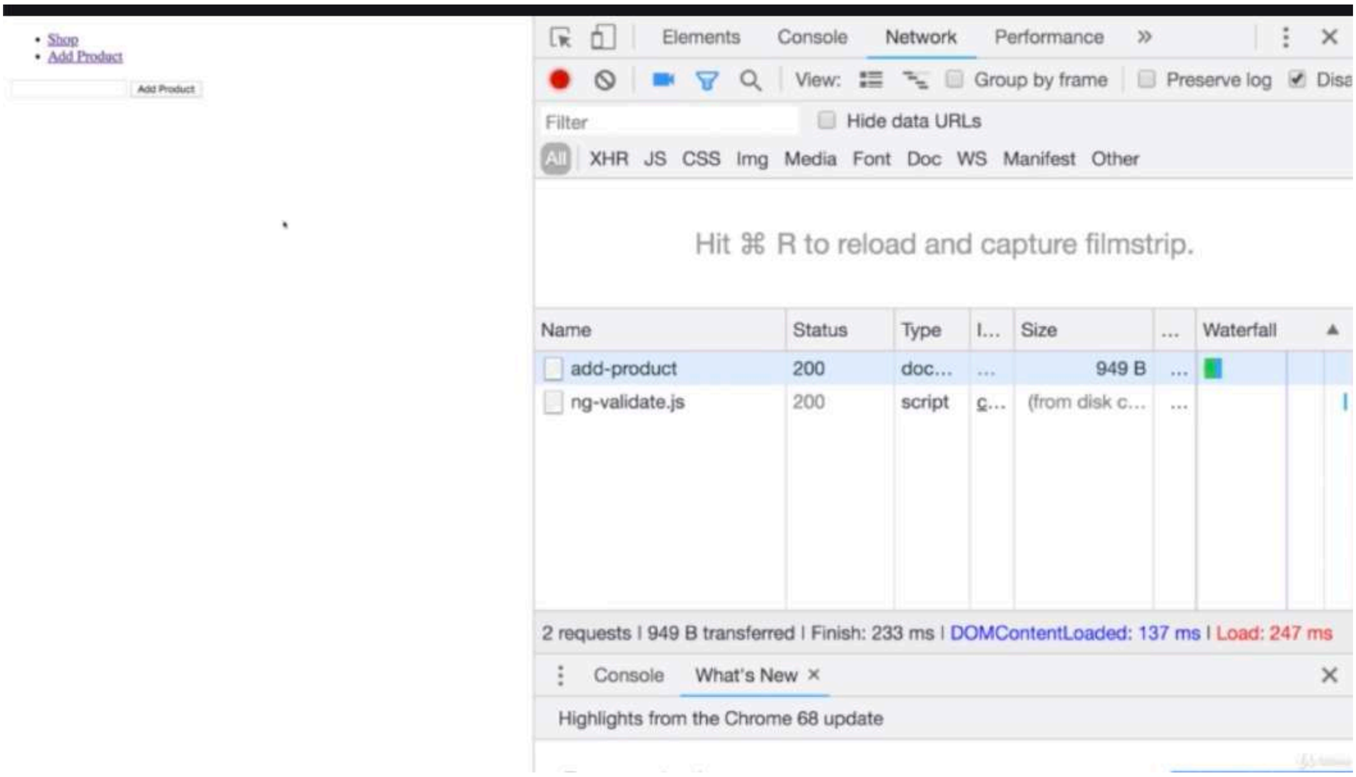
```
<!--./views/404.html-->

```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Page Not Found</title>
</head>
<body>
    <h1>Page Not Found</h1>
</body>
</html>
```

# * Chapter 71: Using A Helper Function For Navigation

1. update
- ./util/path.js
- shop.js
- admin.js

![](images/71-using-a-helper-function-for-navigation-1.png)



Hit ⌘ R to reload and capture filmstrip.

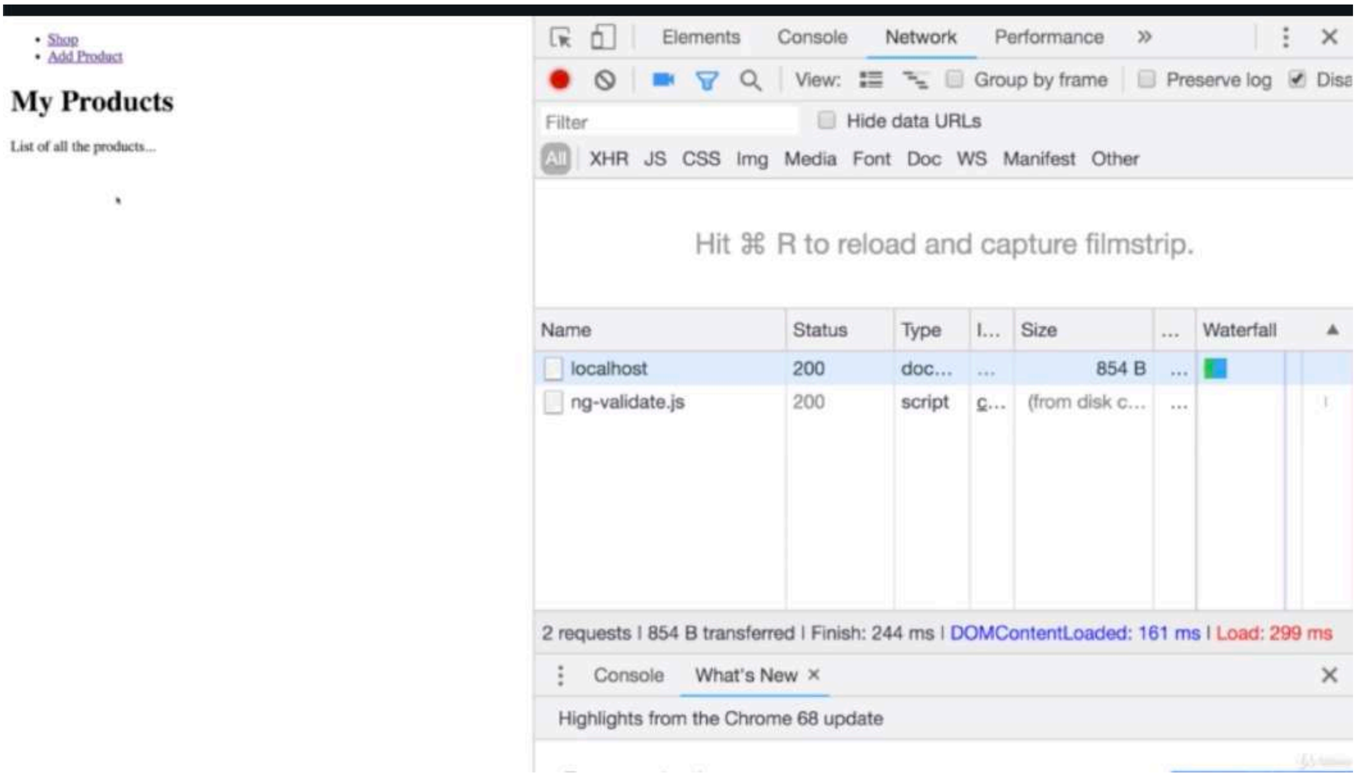
| Name | Status | Type | I... | Size | ... | Waterfall |
|---|---|---|---|---|---|---|
| add-product | 200 | doc... | ... | 949 B | ... | 🟩 |
| ng-validate.js | 200 | script | c... | (from disk c... | ... | |

2 requests | 949 B transferred | Finish: 233 ms | DOMContentLoaded: 137 ms | Load: 247 ms

Highlights from the Chrome 68 update

- after reload, it still works. because now we are in the end having a pretty neat way of constructing a path to our root directory.
- i will do the same in shop.js

![](images/71-using-a-helper-function-for-navigation-2.png)



- Shop
- Add Product

My Products

List of all the products...

| Name | Status | Type | I... | Size | ... | Waterfall |
|---|---|---|---|---|---|---|
| localhost | 200 | doc... | ... | 854 B | ... | 🟩 |
| ng-validate.js | 200 | script | c... | (from disk c... | ... | |

2 requests | 854 B transferred | Finish: 244 ms | DOMContentLoaded: 161 ms | Load: 299 ms

Highlights from the Chrome 68 update

- you could have sticked to the old approach but this one is a even cleaner one and one that should work on all operating systems and it always gives you the path to the root file.

```
//./routes/shop.js
const path = require('path');

const express = require('express');

const rootDir = require('../util/path')

const router = express.Router();

router.get('/', (req, res, next) => {
    res.sendFile(path.join(rootDir, 'views', 'shop.html'))
})

module.exports = router
```

```
//./routes/admin.js
const path = require('path')

const express = require('express')

const rootDir = require('../util/path')

const router = express.Router();


// /admin/add-product => GET
router.get('/add-product', (req, res, next) => {
    res.sendFile(path.join(rootDir, 'views', 'add-product.html'))
})


// /admin/add-product => POST
```

```
router.post('/add-product', (req, res, next) => {
    console.log(req.body)
    res.redirect('/');
})

module.exports = router;
```

```
//./util/path.js

const path = require('path')

/**if we use that
 * we just have to find out
 * which directory or for which file we wanna get the directory name
 *
 * there we can use the global 'process' variable that is also a variable that is available in all files.
 * you don't need to import it
 * and there you will have a mainModule property.
 * this will refer to the main module that started your application
 * so to the module we created in app.js
 * and now we can call file name to find out in which file this module was spun up.
 *
 * so 'process.mainModule.filename' gives us the path to the file
 * that is responsible for the fact that our application is running
 * and filename is what we put into dirname to get a path to that directory.
 * */
module.exports = path.dirname(process.mainModule.filename)
```

# * Chapter 72: Styling Our Pages

1. update
- shop.html
- add-product.html
- 404.html

```
<!--shop.html-->

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Add Product</title>
    <style>
        body {
            padding: 0;
            margin: 0;
            font-family: sans-serif;
        }

        main {
            padding: 1rem;
        }

        .main-header {
            width: 100%;
            height: 3.5rem;
            background-color: #dbc441;
            padding: 0 1.5rem;
        }

        .main-header__nav {
            height: 100%;
            display: flex;
            align-items: center;
        }

        .main-header__item-list {
            list-style: none;
            margin: 0;
            padding: 0;
            display: flex;
        }

        .main-header__item {
            margin: 0 1rem;
            padding: 0;
        }

        .main-header__item a {
            text-decoration: none;
            color: black;
        }

        .main-header__item a:hover,
        .main-header__item a:active,
        .main-header__item a.active {
            color: #3e00a1;
        }
    </style>
</head>

<body>
    <header class="main-header">
        <nav class="main-header__nav">
            <ul class="main-header__item-list">
                <li class="main-header__item"><a class="active" href="/">Shop</a></li>
                <li class="main-header__item"><a href="/admin/add-product">Add Product</a></li>
            </ul>
        </nav>
    </header>

    <main>
        <h1>My Products</h1>
        <p>List of all the products...</p>
    </main>
</body>
```

```html

</html>
```

```html
<!--add-product.html-->

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Add Product</title>
    <style>
        body {
            padding: 0;
            margin: 0;
            font-family: sans-serif;
        }

        main {
            padding: 1rem;
        }

        .main-header {
            width: 100%;
            height: 3.5rem;
            background-color: #dbc441;
            padding: 0 1.5rem;
        }

        .main-header__nav {
            height: 100%;
            display: flex;
            align-items: center;
        }

        .main-header__item-list {
            list-style: none;
            margin: 0;
            padding: 0;
            display: flex;
        }

        .main-header__item {
            margin: 0 1rem;
            padding: 0;
        }

        .main-header__item a {
            text-decoration: none;
            color: black;
        }

        .main-header__item a:hover,
        .main-header__item a:active,
        .main-header__item a.active {
            color: #3e00a1;
        }

        .product-form {
            width: 20rem;
            max-width: 90%;
            margin: auto;
        }

        .form-control {
            margin: 1rem 0;
        }

        .form-control label,
        .form-control input {
            display: block;
            width: 100%;
        }

        .form-control input {
            border: 1px solid #dbc441;
            font: inherit;
            border-radius: 2px;
        }

        button {
            font: inherit;
            border: 1px solid #3e00a1;
            color: #3e00a1;
            background: white;
            border-radius: 3px;
            cursor: pointer;
        }

        button:hover,
        button:active {
            background-color: #3e00a1;
            color: white;
        }
    </style>
</head>

<body>
    <header class="main-header">
        <nav class="main-header__nav">
            <ul class="main-header__item-list">
                <li class="main-header__item"><a href="/">Shop</a></li>
                <li class="main-header__item"><a class="active" href="/admin/add-product">Add Product</a></li>
            </ul>
        </nav>
    </header>

    <main>
        <form class="product-form" action="/admin/add-product" method="POST">
            <div class="form-control">
```

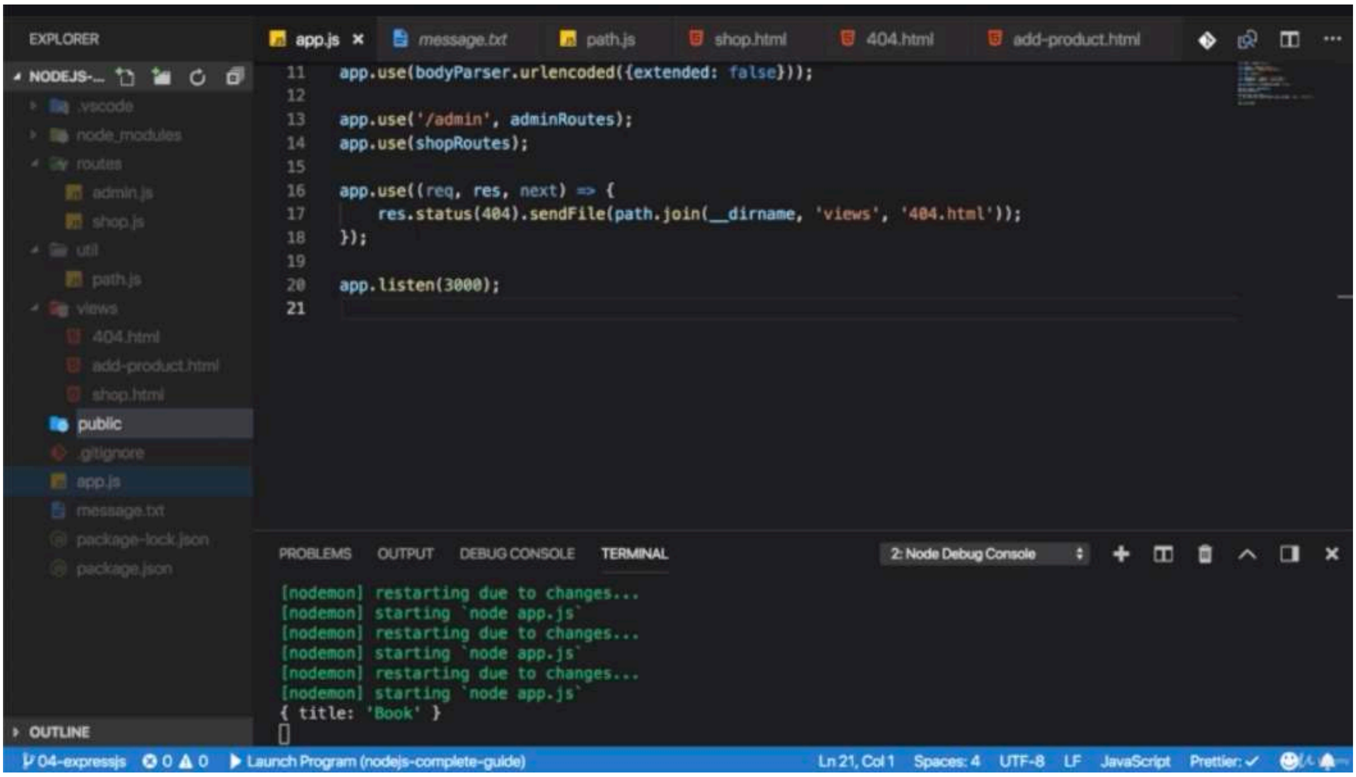
```
                <label for="title">Title</label>
                <input type="text" name="title" id="title">
            </div>

            <button type="submit">Add Product</button>
        </form>
    </main>
</body>

</html>
```

```
<!--./views/404.html-->

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Page Not Found</title>
    <style>
        body {
            padding: 0;
            margin: 0;
            font-family: sans-serif;
        }

        main {
            padding: 1rem;
        }

        .main-header {
            width: 100%;
            height: 3.5rem;
            background-color: #dbc441;
            padding: 0 1.5rem;
        }

        .main-header__nav {
            height: 100%;
            display: flex;
            align-items: center;
        }

        .main-header__item-list {
            list-style: none;
            margin: 0;
            padding: 0;
            display: flex;
        }

        .main-header__item {
            margin: 0 1rem;
            padding: 0;
        }

        .main-header__item a {
            text-decoration: none;
            color: black;
        }

        .main-header__item a:hover,
        .main-header__item a:active,
        .main-header__item a.active {
            color: #3e00a1;
        }
    </style>
</head>

<body>
    <header class="main-header">
        <nav class="main-header__nav">
            <ul class="main-header__item-list">
                <li class="main-header__item"><a class="active" href="/">Shop</a></li>
                <li class="main-header__item"><a href="/admin/add-product">Add Product</a></li>
            </ul>
        </nav>
    </header>
    <h1>Page Not Found!</h1>
</body>

</html>
```

# * Chapter 73: Serving Files Statically

1. update
- app.js
- ./views/shop.html
- ./views/add-product.html
- ./views/404.html
- ./public/css/main.css
- ./public/css/product.css

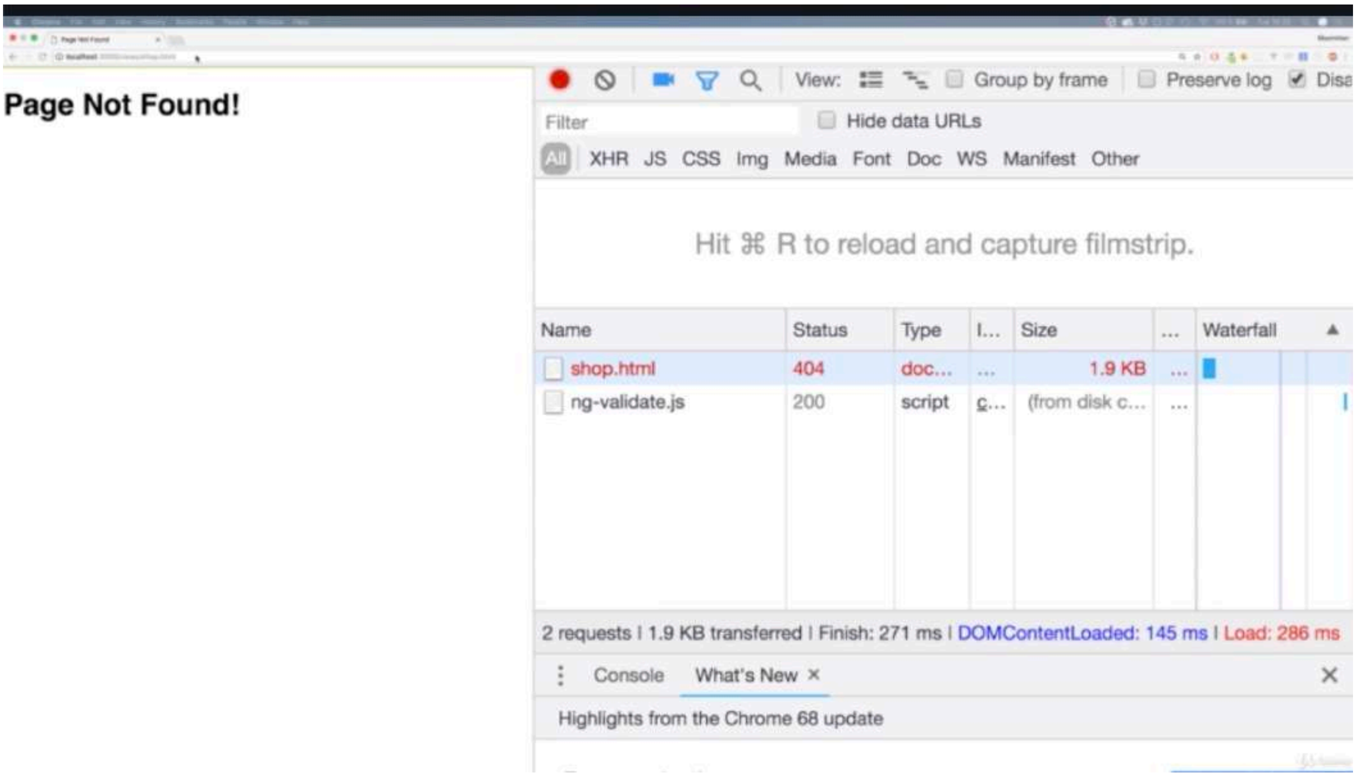
![](images/73-serving-files-statically-1.png)



```javascript
app.use(bodyParser.urlencoded({extended: false}));

app.use('/admin', adminRoutes);
app.use(shopRoutes);

app.use((req, res, next) => {
    res.status(404).sendFile(path.join(__dirname, 'views', '404.html'));
});

app.listen(3000);
```

```
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
{ title: 'Book' }
```

- but the convention is to call it 'public' because you wanna indicate that this is a folder that holds content which are always exposed to the public crowd or which is always exposed to the public. so where you don't need any permissions to access and that's important.
![](images/73-serving-files-statically-2.png)



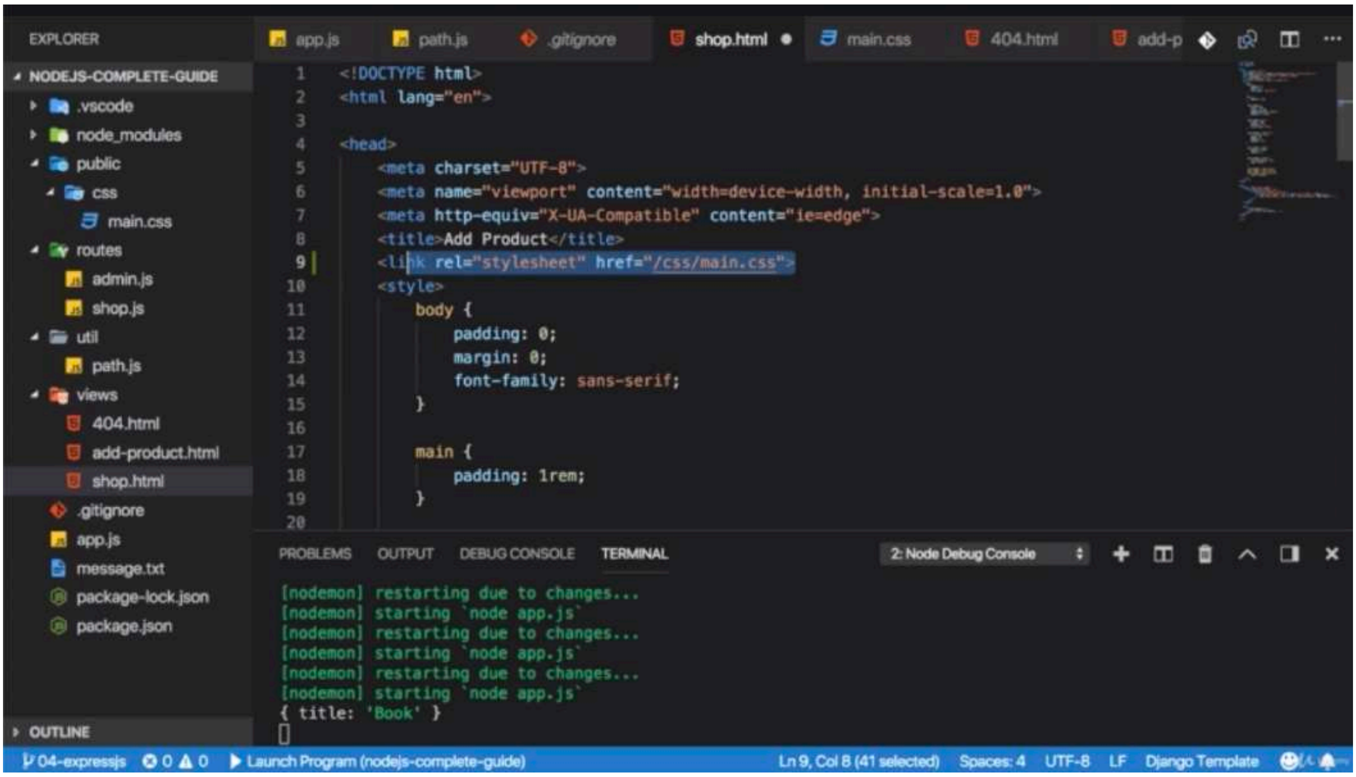
**Page Not Found!**

| Name | Status | Type | I... | Size | ... | Waterfall |
|---|---|---|---|---|---|---|
| shop.html | 404 | doc... | ... | 1.9 KB | ... | |
| ng-validate.js | 200 | script | c... | (from disk c... | ... | |

2 requests | 1.9 KB transferred | Finish: 271 ms | DOMContentLoaded: 145 ms | Load: 286 ms

Highlights from the Chrome 68 update

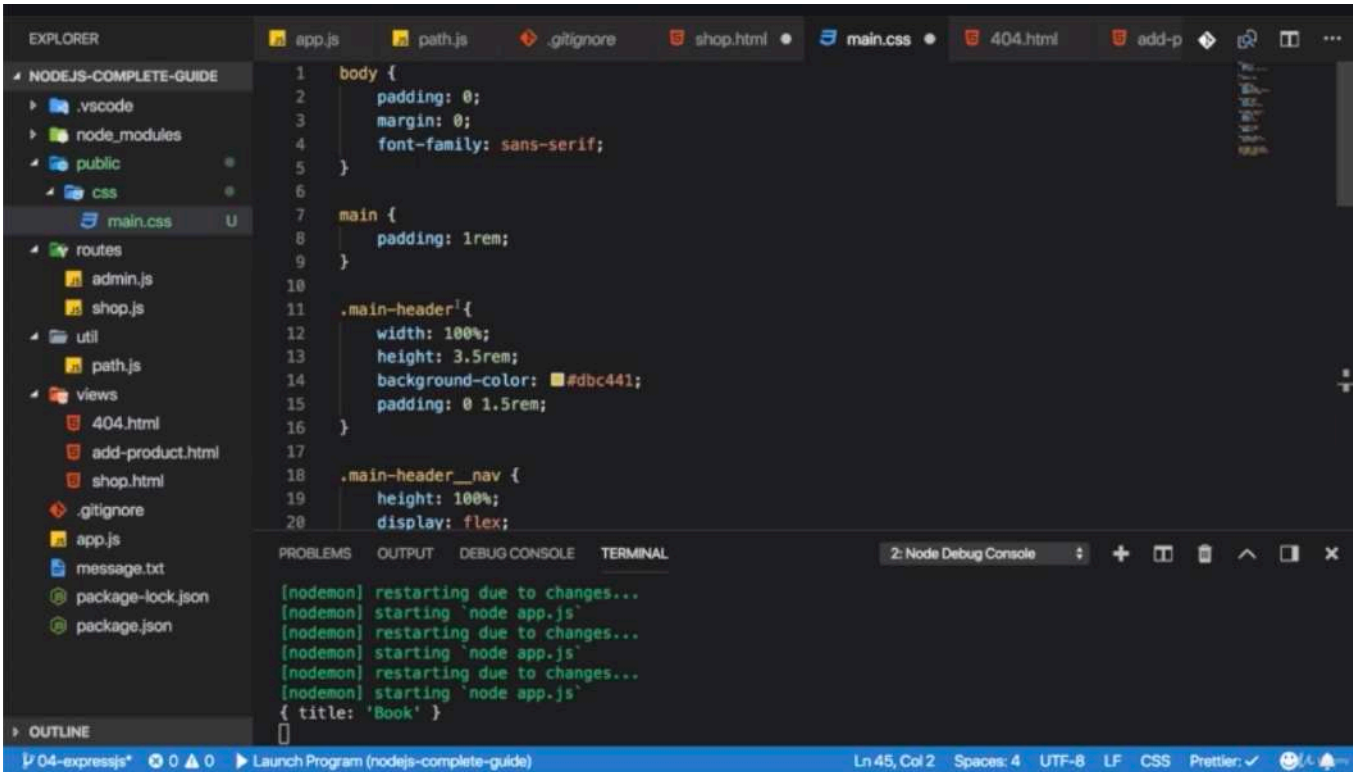
- all your files are not accessible by your users. if you ever tried to enter localhost and then something like views, shop.html, that will not work because this is simply accepted by express and it tries to find a route that matches this. it tries to find it here in app.js. and also in shop routes and so on. it doesn't find that route and therefore it doesn't give you access, you can't access the file system here through URL
![](images/73-serving-files-statically-3.png)



```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Add Product</title>
    <link rel="stylesheet" href="/css/main.css">
    <style>
        body {
            padding: 0;
            margin: 0;
            font-family: sans-serif;
        }

        main {
            padding: 1rem;
        }

```

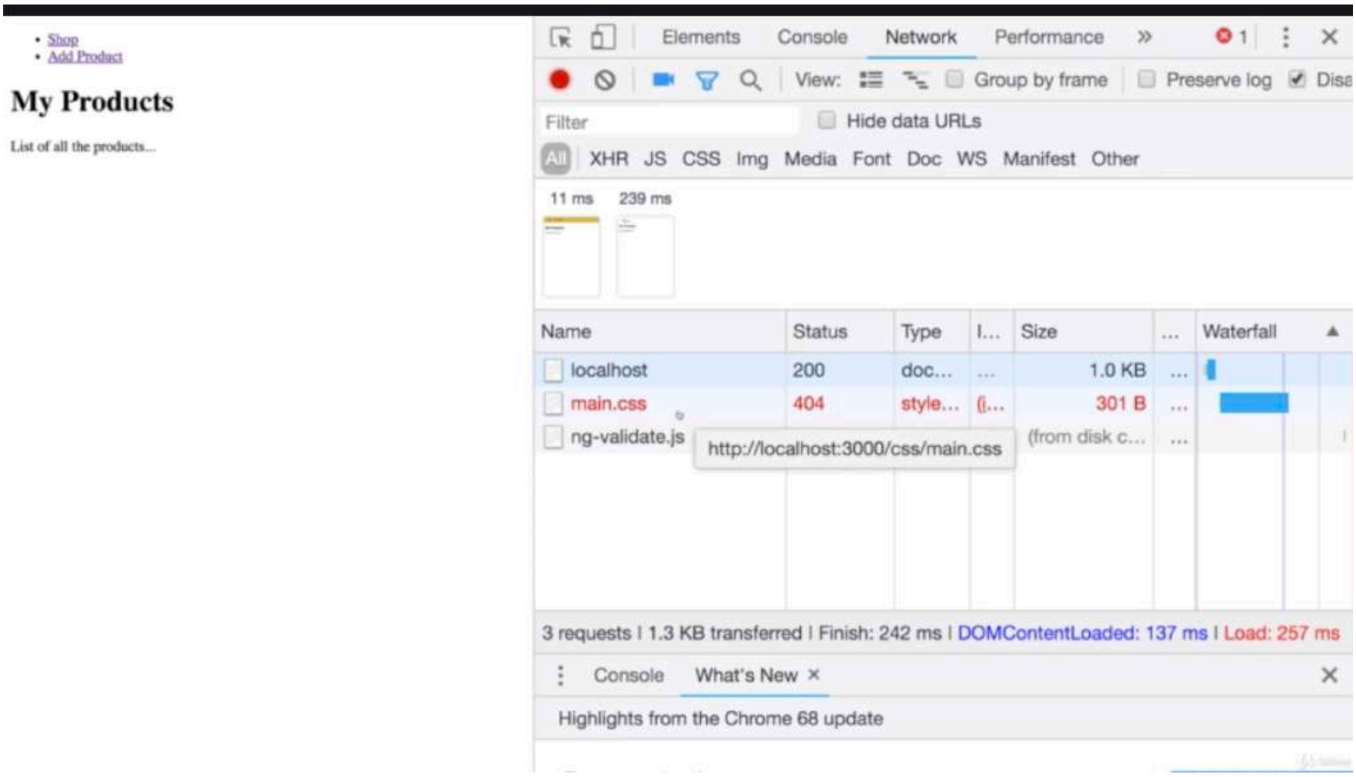
```
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
{ title: 'Book' }
```

- but now i wanna make an exception. i want the some requests can access the file system because let's say in shop.html, i wanna have something like a link here where i point at something like main.css anything like that.
- and my imagination would be that in public folder, i have a css folder with main css file. that's the file i wanna serve with this link.
![](images/73-serving-files-statically-4.png)
![](images/73-serving-files-statically-5.png)



```css
body {
    padding: 0;
    margin: 0;
    font-family: sans-serif;
}

main {
    padding: 1rem;
}

.main-header {
    width: 100%;
    height: 3.5rem;
    background-color: #dbc441;
    padding: 0 1.5rem;
}

.main-header__nav {
    height: 100%;
    display: flex;
```

```
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
{ title: 'Book' }
```



- Shop
- Add Product

My Products

List of all the products...

| Name | Status | Type | I... | Size | ... | Waterfall |
|---|---|---|---|---|---|---|
| localhost | 200 | doc... | ... | 1.0 KB | ... | |
| main.css | 404 | style... | (i... | 301 B | ... | |
| ng-validate.js | | | | (from disk c... | ... | |

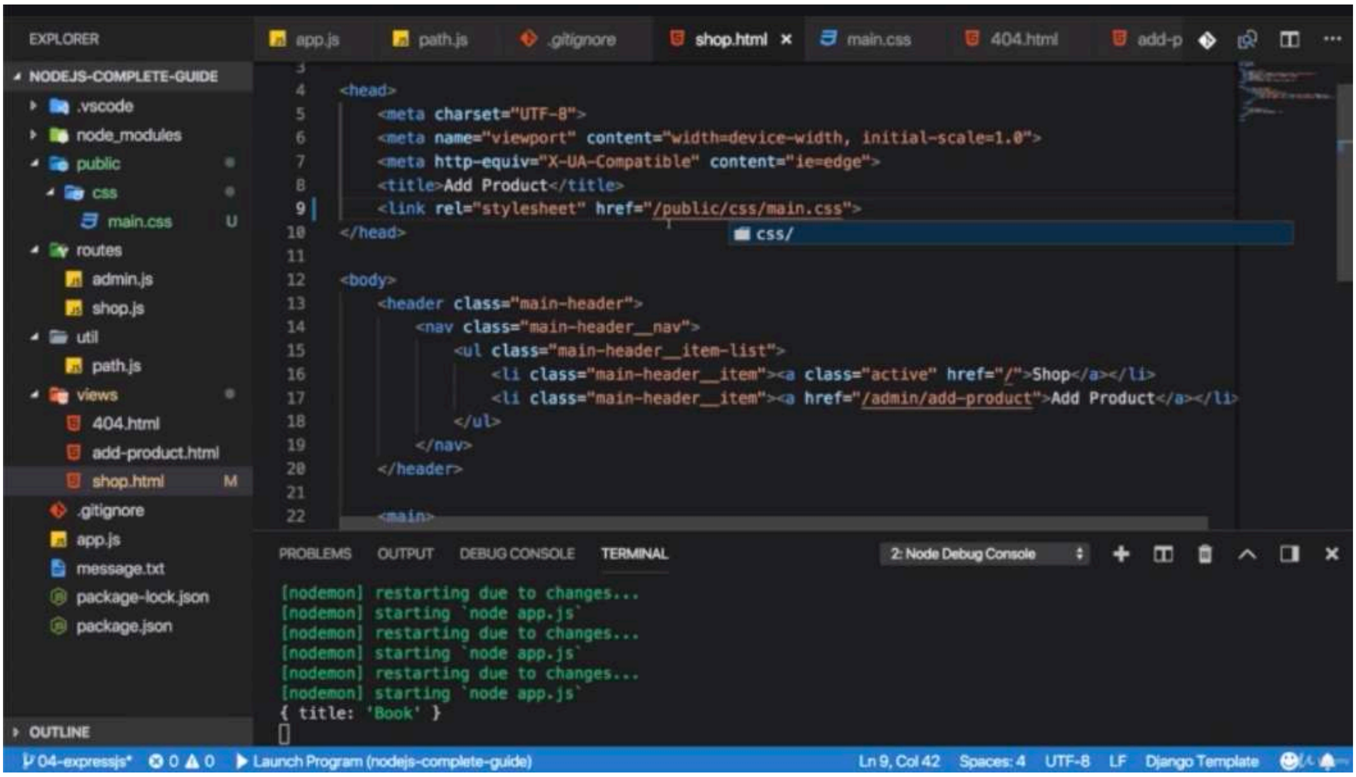
http://localhost:3000/css/main.css

3 requests | 1.3 KB transferred | Finish: 242 ms | DOMContentLoaded: 137 ms | Load: 257 ms

Highlights from the Chrome 68 update

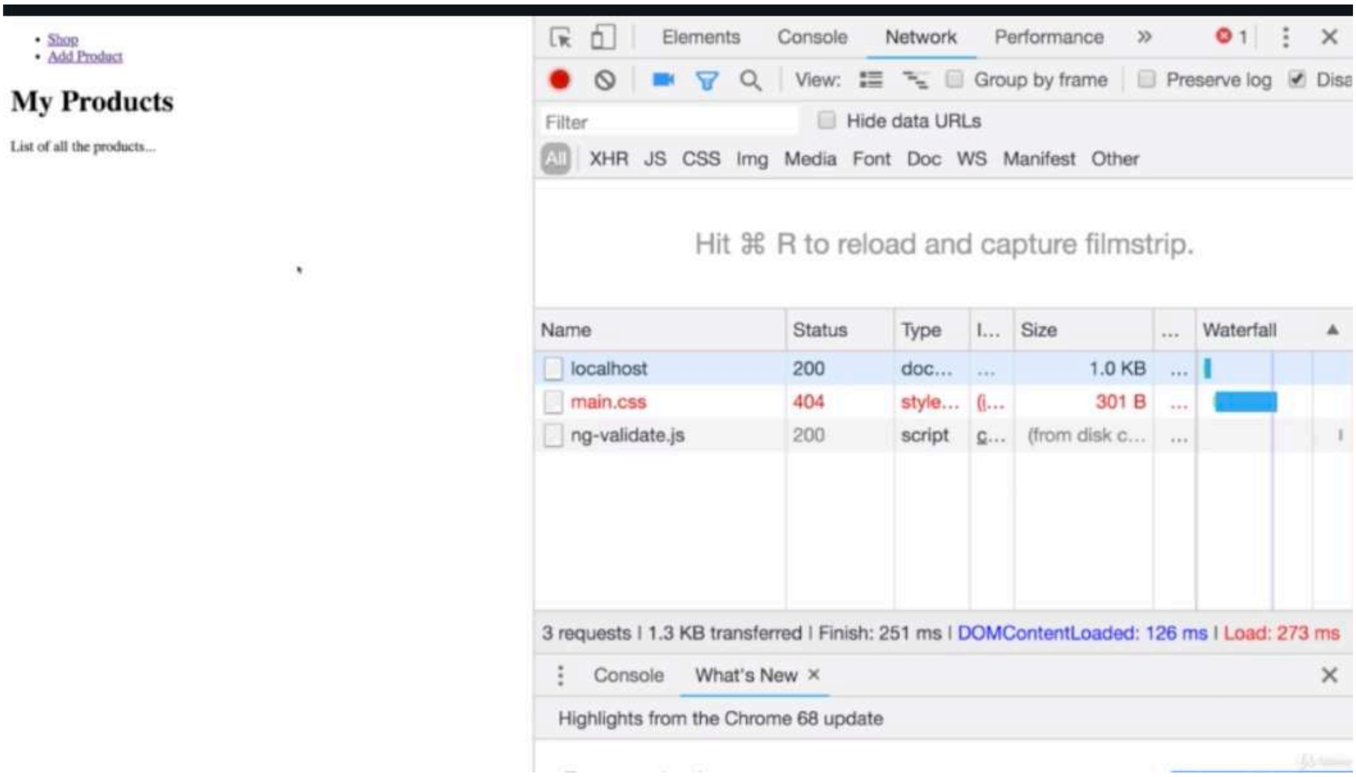
- if i save and reload my main page, all the styling is gone because it can’t find the main css file as far as you can see here in the developer tools because we can’t access the file system.
- path is incorrect. it’s public/css.

![](images/73-serving-files-statically-6.png)
![](images/73-serving-files-statically-7.png)
![](images/73-serving-files-statically-8.png)



```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Add Product</title>
    <link rel="stylesheet" href="/public/css/main.css">
</head>

<body>
    <header class="main-header">
        <nav class="main-header__nav">
            <ul class="main-header__item-list">
                <li class="main-header__item"><a class="active" href="/">Shop</a></li>
                <li class="main-header__item"><a href="/admin/add-product">Add Product</a></li>
            </ul>
        </nav>
    </header>

    <main>
```

```
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
{ title: 'Book' }
```



- Shop
- Add Product

My Products

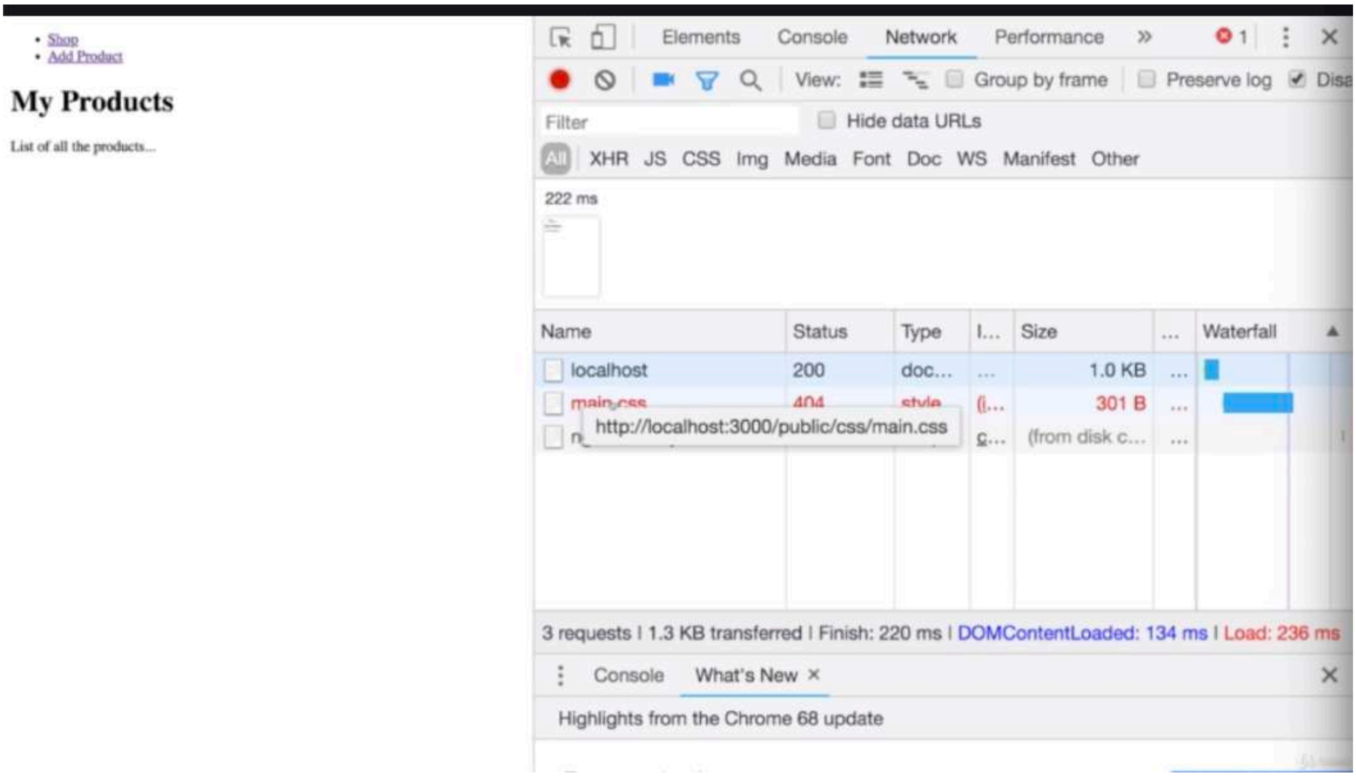
List of all the products...

Hit ⌘ R to reload and capture filmstrip.

| Name | Status | Type | I... | Size | ... |
|---|---|---|---|---|---|
| localhost | 200 | doc... | ... | 1.0 KB | ... |
| main.css | 404 | style... | (i... | 301 B | ... |
| ng-validate.js | 200 | script | c... | (from disk c... | ... |

3 requests | 1.3 KB transferred | Finish: 251 ms | DOMContentLoaded: 126 ms | Load: 273 ms

Highlights from the Chrome 68 update



- Shop
- Add Product

My Products

List of all the products...

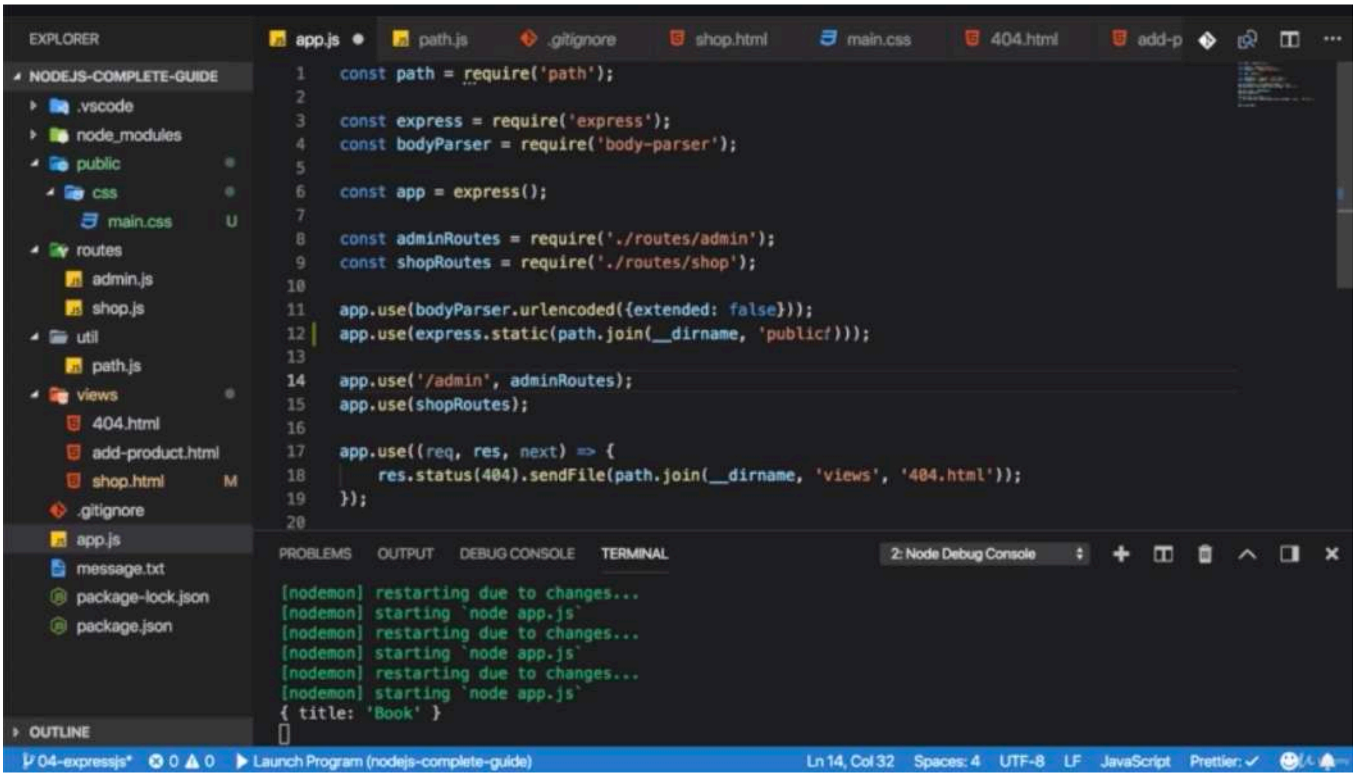
| Name | Status | Type | I... | Size | ... |
|---|---|---|---|---|---|
| localhost | 200 | doc... | ... | 1.0 KB | ... |
| main.css | 404 | style | (i... | 301 B | ... |
| n | | | c... | (from disk c... | ... |

http://localhost:3000/public/css/main.css

3 requests | 1.3 KB transferred | Finish: 220 ms | DOMContentLoaded: 134 ms | Load: 236 ms

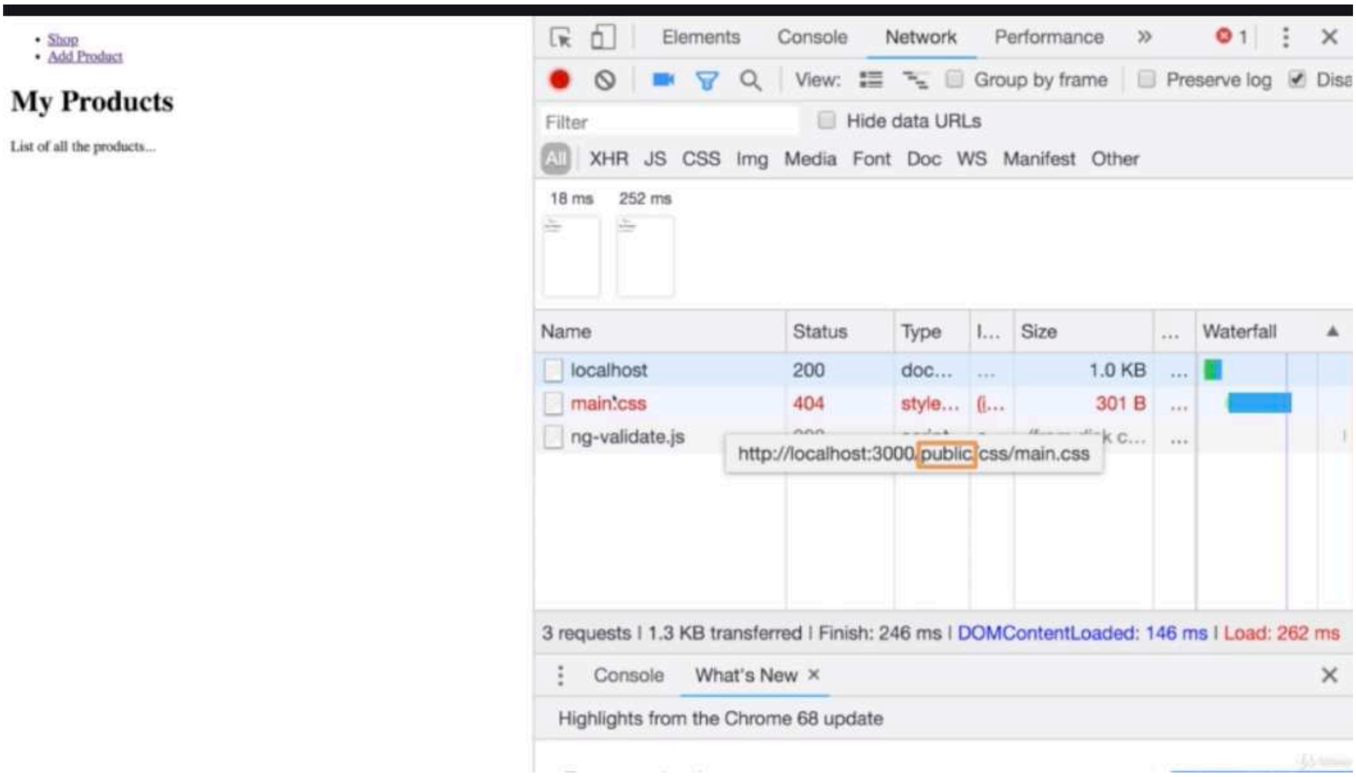
Highlights from the Chrome 68 update

- if i change like that and reload and you will see it never work. and now it does look in the public folder.
- for this, we need a feature express.js offers us. we need to be able to serve files statically and statically simply means not handled by the express router or other middleware. but instead directly forwarded to the file system
![](images/73-serving-files-statically-9.png)
![](images/73-serving-files-statically-10.png)



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

const express = require('express');
const bodyParser = require('body-parser');

const app = express();

const adminRoutes = require('./routes/admin');
const shopRoutes = require('./routes/shop');

app.use(bodyParser.urlencoded({extended: false}));
app.use(express.static(path.join(__dirname, 'public')));

app.use('/admin', adminRoutes);
app.use(shopRoutes);

app.use((req, res, next) => {
    res.status(404).sendFile(path.join(__dirname, 'views', '404.html'));
});
```

```
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
{ title: 'Book' }
```



- Shop
- Add Product

My Products

List of all the products...

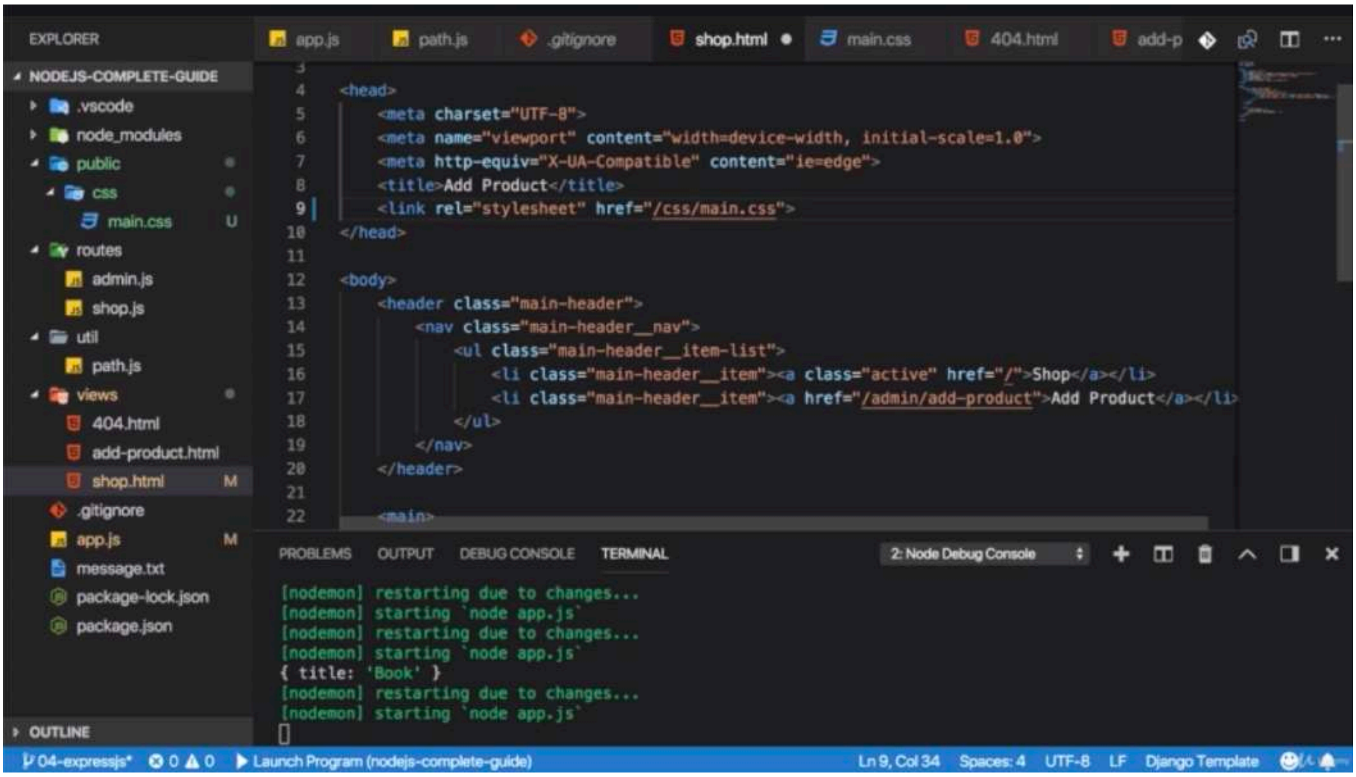
| Name | Status | Type | I... | Size | ... | Waterfall |
|---|---|---|---|---|---|---|
| localhost | 200 | doc... | ... | 1.0 KB | ... | |
| main.css | 404 | style... | (i... | 301 B | ... | |
| ng-validate.js | | | | k c... | ... | |

http://localhost:3000/public/css/main.css

3 requests | 1.3 KB transferred | Finish: 246 ms | DOMContentLoaded: 146 ms | Load: 262 ms

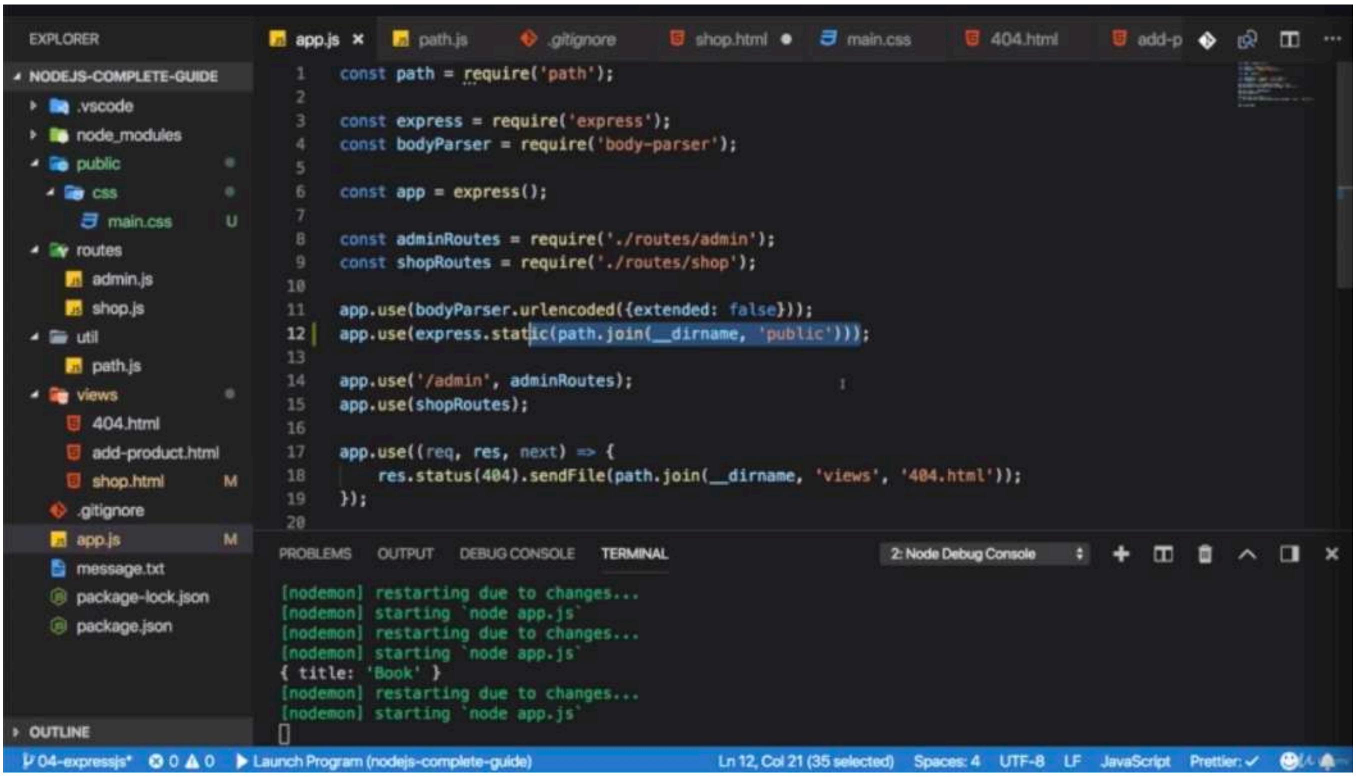
Highlights from the Chrome 68 update

- still doesn’t work because the path with public at the beginning is wrong.
![](images/73-serving-files-statically-11.png)
![](images/73-serving-files-statically-12.png)



```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Add Product</title>
    <link rel="stylesheet" href="/css/main.css">
</head>

<body>
    <header class="main-header">
        <nav class="main-header__nav">
            <ul class="main-header__item-list">
                <li class="main-header__item"><a class="active" href="/">Shop</a></li>
                <li class="main-header__item"><a href="/admin/add-product">Add Product</a></li>
            </ul>
        </nav>
    </header>

    <main>
```

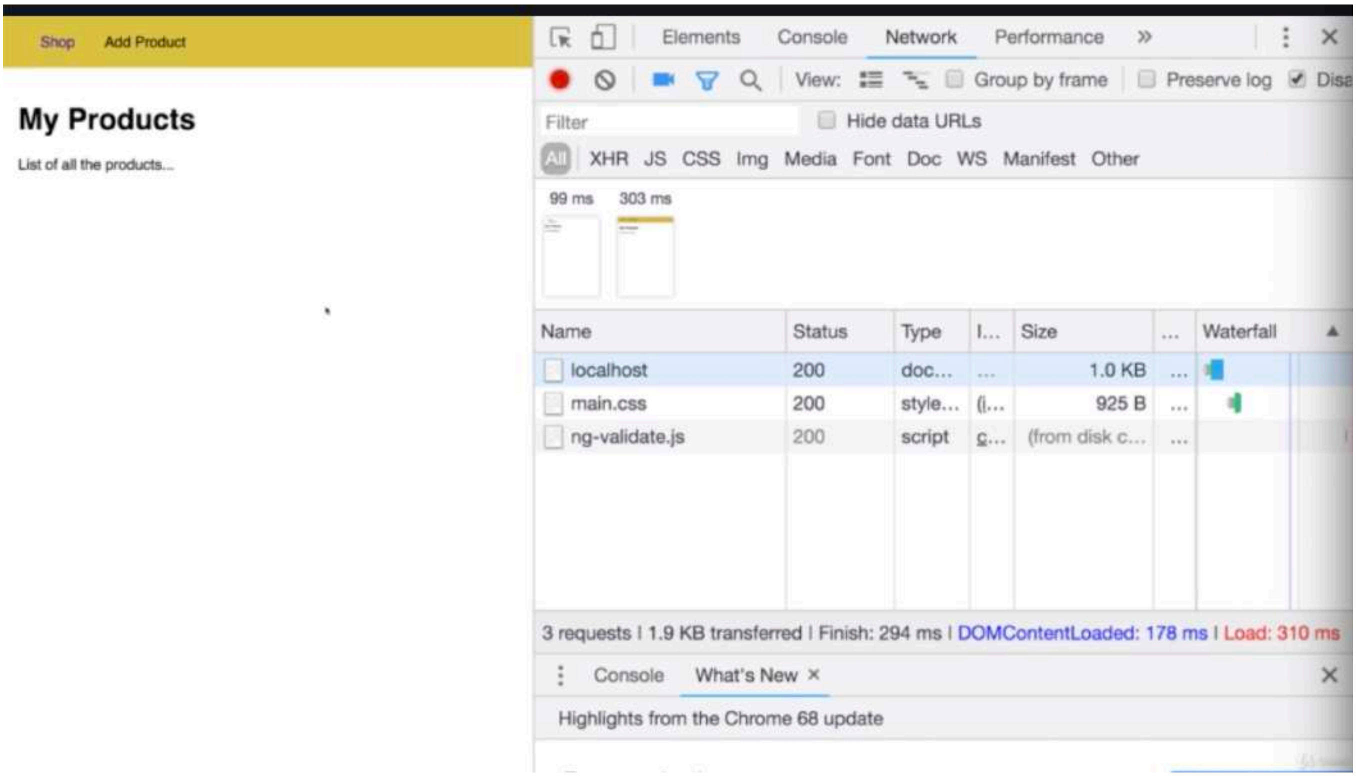
```
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
{ title: 'Book' }
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
```



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

const express = require('express');
const bodyParser = require('body-parser');

const app = express();

const adminRoutes = require('./routes/admin');
const shopRoutes = require('./routes/shop');

app.use(bodyParser.urlencoded({extended: false}));
app.use(express.static(path.join(__dirname, 'public')));

app.use('/admin', adminRoutes);
app.use(shopRoutes);

app.use((req, res, next) => {
    res.status(404).sendFile(path.join(__dirname, 'views', '404.html'));
});
```

```
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
{ title: 'Book' }
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
```

- we should omit this and directly act as if we are in the public folder already. because this is basically what express will do here.
- it will take any request that tries to find some file. so anything that tries to find a .css or .js files, if we have such a request, it automatically forwards it to the public folder
- and therefore then the remaining path has to be everything but that public.
![](images/73-serving-files-statically-13.png)
![](images/73-serving-files-statically-14.png)
![](images/73-serving-files-statically-15.png)
![](images/73-serving-files-statically-16.png)



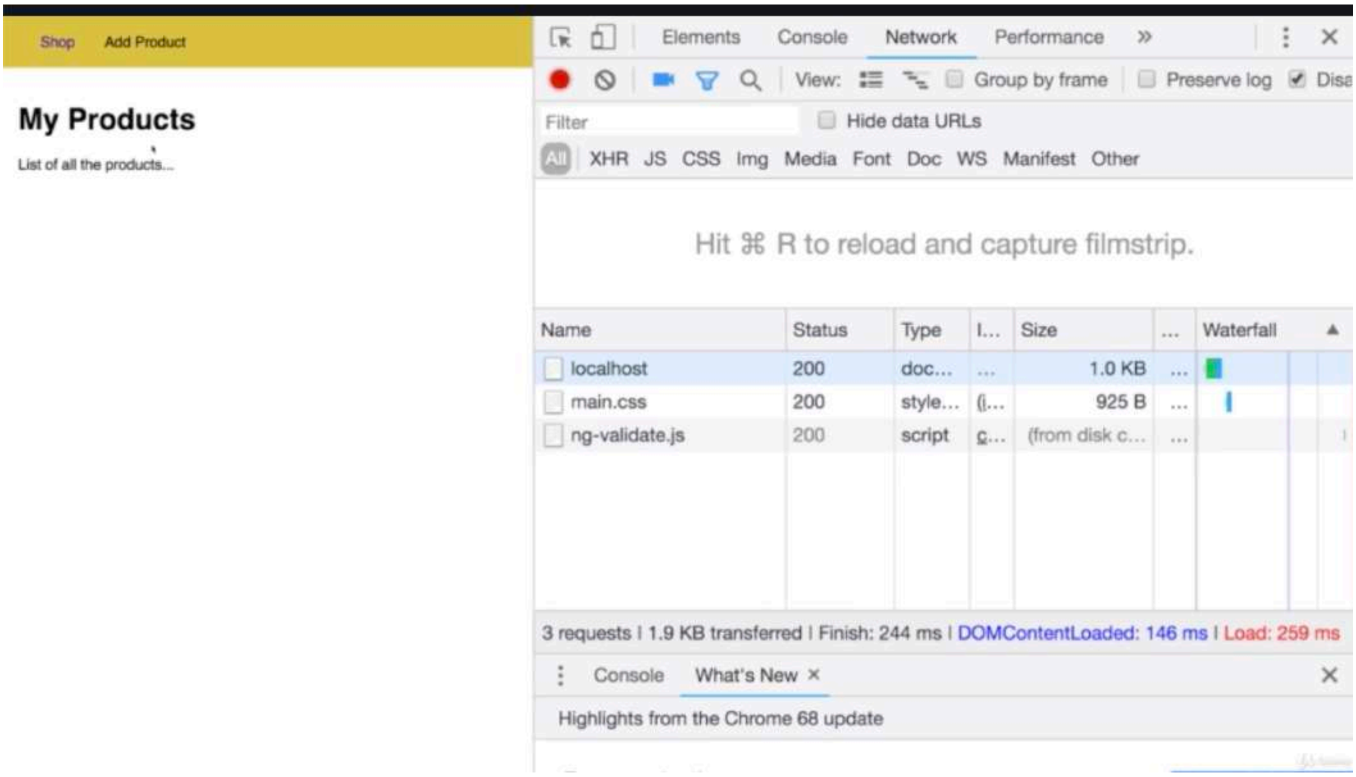
# My Products

List of all the products...

| Name | Status | Type | I... | Size | ... | Waterfall |
|---|---|---|---|---|---|---|
| localhost | 200 | doc... | ... | 1.0 KB | ... | |
| main.css | 200 | style... | (i... | 925 B | ... | |
| ng-validate.js | 200 | script | c... | (from disk c... | ... | |

3 requests | 1.9 KB transferred | Finish: 294 ms | DOMContentLoaded: 178 ms | Load: 310 ms

Highlights from the Chrome 68 update



# My Products

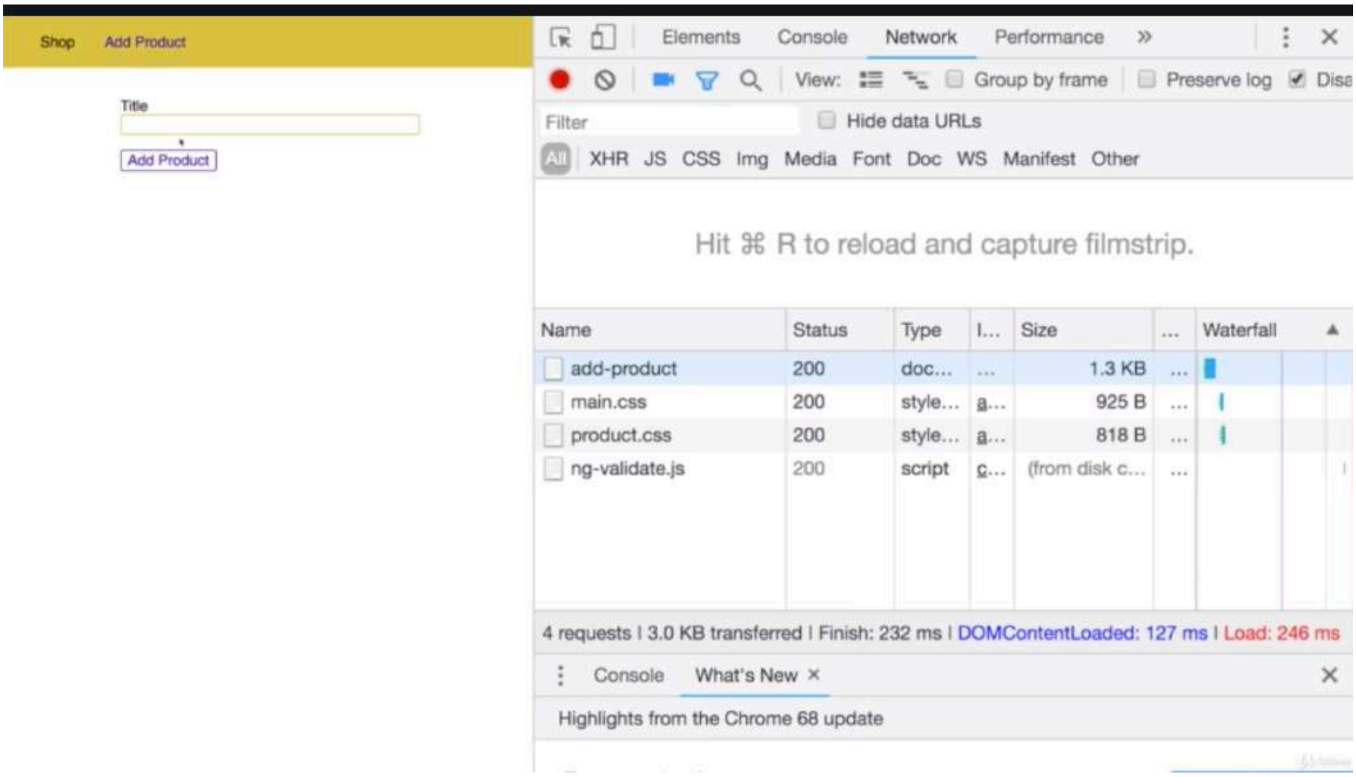
List of all the products...

Hit ⌘ R to reload and capture filmstrip.

| Name | Status | Type | I... | Size | ... | Waterfall |
|---|---|---|---|---|---|---|
| localhost | 200 | doc... | ... | 1.0 KB | ... | |
| main.css | 200 | style... | (i... | 925 B | ... | |
| ng-validate.js | 200 | script | c... | (from disk c... | ... | |

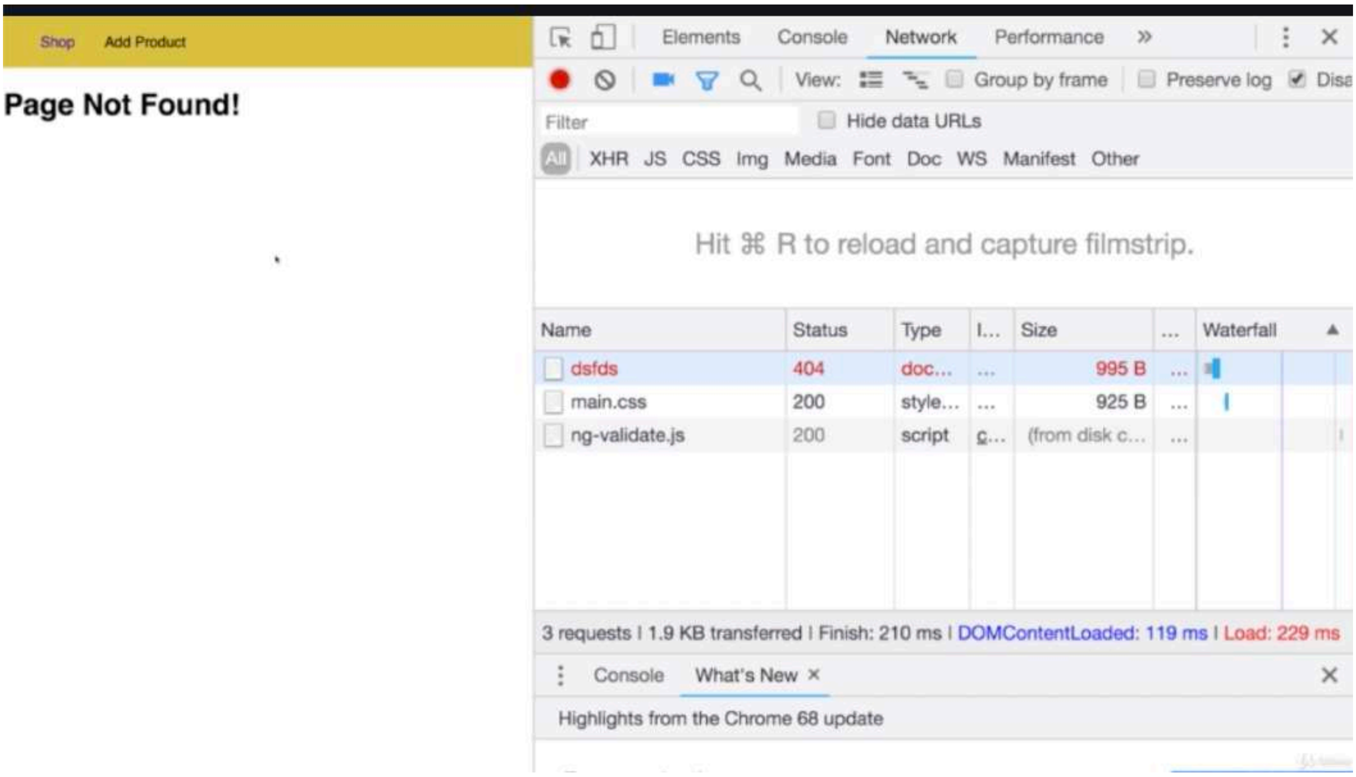
3 requests | 1.9 KB transferred | Finish: 244 ms | DOMContentLoaded: 146 ms | Load: 259 ms

Highlights from the Chrome 68 update



| Name | Status | Type | I... | Size | ... | Waterfall |
|---|---|---|---|---|---|---|
| add-product | 200 | doc... | ... | 1.3 KB | ... | |
| main.css | 200 | style... | a... | 925 B | ... | |
| product.css | 200 | style... | a... | 818 B | ... | |
| ng-validate.js | 200 | script | c... | (from disk c... | ... | |

4 requests | 3.0 KB transferred | Finish: 232 ms | DOMContentLoaded: 127 ms | Load: 246 ms



Page Not Found!

| Name | Status | Type | I... | Size | ... | Waterfall |
|---|---|---|---|---|---|---|
| dsfds | 404 | doc... | ... | 995 B | ... | |
| main.css | 200 | style... | ... | 925 B | ... | |
| ng-validate.js | 200 | script | c... | (from disk c... | ... | |

3 requests | 1.9 KB transferred | Finish: 210 ms | DOMContentLoaded: 119 ms | Load: 229 ms

```
//app.js

const path = require('path')

const express = require('express');
const bodyParser = require('body-parser');

const app = express()

const adminRoutes = require('./routes/admin');
const shopRoutes = require('./routes/shop');

app.use(bodyParser.urlencoded({extended: false}))
```

```
/**'static' is built-in method and this is a built-in middleware
 * it serve static files. so we can execute this function.
 *
 * we have to pass in a path to the folder which we wanna serve statically
 * so a folder which we wanna grant read access to.
 *
 * with this, user should be able to access the public path
 */

 /**this could register multiple static folders
  * and it will funnel the request through all of them until it has a first hit for the file it's looking for */
app.use(express.static(path.join(__dirname, 'public')))

app.use('/admin', adminRoutes);
app.use(shopRoutes);

app.use((req, res, next) => {
    res.status(404).sendFile(path.join(__dirname, 'views', '404.html'))
})

app.listen(3000);
```

```
<!--./views/shop.html-->

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Add Product</title>
    <link rel="stylesheet" href="/css/main.css">
</head>

<body>
    <header class="main-header">
        <nav class="main-header__nav">
            <ul class="main-header__item-list">
                <li class="main-header__item"><a class="active" href="/">Shop</a></li>
                <li class="main-header__item"><a href="/admin/add-product">Add Product</a></li>
            </ul>
        </nav>
    </header>

    <main>
        <h1>My Products</h1>
        <p>List of all the products...</p>
    </main>
</body>

</html>
```

```
<!--./views/add-product.html-->

```

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Add Product</title>
    <link rel="stylesheet" href="/css/main.css">
    <link rel="stylesheet" href="/css/product.css">
</head>

<body>
    <header class="main-header">
        <nav class="main-header__nav">
            <ul class="main-header__item-list">
                <li class="main-header__item"><a href="/">Shop</a></li>
                <li class="main-header__item"><a class="active" href="/admin/add-product">Add Product</a></li>
            </ul>
        </nav>
    </header>

    <main>
        <form class="product-form" action="/admin/add-product" method="POST">
            <div class="form-control">
                <label for="title">Title</label>
                <input type="text" name="title" id="title">
            </div>

            <button type="submit">Add Product</button>
        </form>
    </main>
</body>

</html>
```

```html
<!--./views/404.html-->

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Page Not Found</title>
    <link rel="stylesheet" href="/css/main.css">
</head>

<body>
    <header class="main-header">
        <nav class="main-header__nav">
            <ul class="main-header__item-list">
                <li class="main-header__item"><a class="active" href="/">Shop</a></li>
                <li class="main-header__item"><a href="/admin/add-product">Add Product</a></li>
            </ul>
        </nav>
    </header>
    <h1>Page Not Found!</h1>
</body>

</html>
```

```css
/*./public/css/main.css*/

body {
    padding: 0;
    margin: 0;
    font-family: sans-serif;
}

main {
    padding: 1rem;
}

.main-header {
    width: 100%;
    height: 3.5rem;
    background-color: #dbc441;
    padding: 0 1.5rem;
}

.main-header__nav {
    height: 100%;
    display: flex;
    align-items: center;
}

.main-header__item-list {
    list-style: none;
    margin: 0;
    padding: 0;
    display: flex;
}

.main-header__item {
    margin: 0 1rem;
    padding: 0;
}

.main-header__item a {
    text-decoration: none;
    color: black;
}

.main-header__item a:hover,
.main-header__item a:active,
.main-header__item a.active {
    color: #3e00a1;
}
```

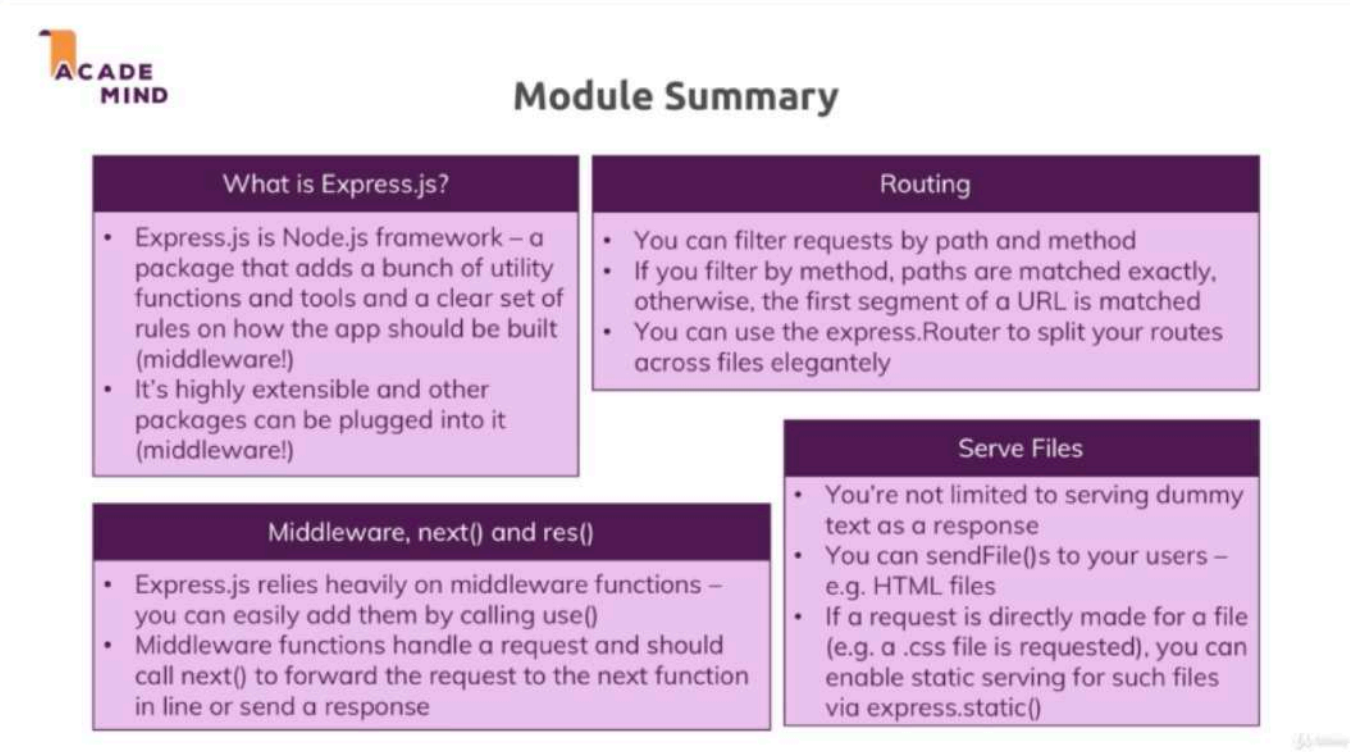
```css
/*./public/css/product.css*/
```

```css

body {
    padding: 0;
    margin: 0;
    font-family: sans-serif;
}

main {
    padding: 1rem;
}

.main-header {
    width: 100%;
    height: 3.5rem;
    background-color: #dbc441;
    padding: 0 1.5rem;
}

.main-header__nav {
    height: 100%;
    display: flex;
    align-items: center;
}

.main-header__item-list {
    list-style: none;
    margin: 0;
    padding: 0;
    display: flex;
}

.main-header__item {
    margin: 0 1rem;
    padding: 0;
}

.main-header__item a {
    text-decoration: none;
    color: black;
}

.main-header__item a:hover,
.main-header__item a:active,
.main-header__item a.active {
    color: #3e00a1;
}

.product-form {
    width: 20rem;
    max-width: 90%;
    margin: auto;
}

.form-control {
    margin: 1rem 0;
}
```

```

.form-control label,
.form-control input {
    display: block;
    width: 100%;
}

.form-control input {
    border: 1px solid #dbc441;
    font: inherit;
    border-radius: 2px;
}

button {
    font: inherit;
    border: 1px solid #3e00a1;
    color: #3e00a1;
    background: white;
    border-radius: 3px;
    cursor: pointer;
}

button:hover,
button:active {
    background-color: #3e00a1;
    color: white;
}
```

# * Chapter 74: Wrap Up

![](images/74-wrap-up-1.png)



ACADEMIND

Module Summary

**What is Express.js?**

- Express.js is Node.js framework – a package that adds a bunch of utility functions and tools and a clear set of rules on how the app should be built (middleware!)
- It's highly extensible and other packages can be plugged into it (middleware!)

**Routing**

- You can filter requests by path and method
- If you filter by method, paths are matched exactly, otherwise, the first segment of a URL is matched
- You can use the express.Router to split your routes across files elegantly

**Middleware, next() and res()**

- Express.js relies heavily on middleware functions – you can easily add them by calling use()
- Middleware functions handle a request and should call next() to forward the request to the next function in line or send a response

**Serve Files**

- You're not limited to serving dummy text as a response
- You can sendFile()s to your users – e.g. HTML files
- If a request is directly made for a file (e.g. a .css file is requested), you can enable static serving for such files via express.static()