

# 19. Error Handling

## \* Chapter 301: Module Introduction





### What's In This Module?

Different Types of Errors

Handling Errors

301

## \* Chapter 302: Types Of Errors & Error Handling





## How Bad Are Errors?

Errors are not necessarily the end of your app!



You just need to handle them correctly!

John Doe

## Different Types of Errors

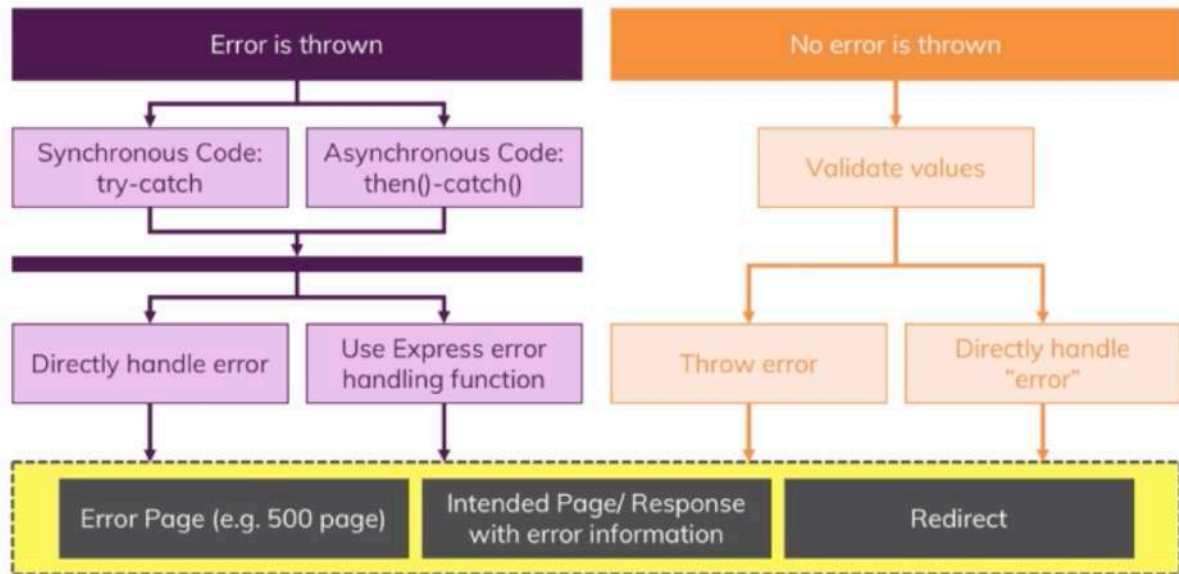
Technical/ Network Errors	"Expected" Errors	Bugs/ Logical Errors
e.g. MongoDB server is down	e.g. File can't be read, database operation fails	e.g. User object used when it doesn't exist
Show error page to user	Inform user, possibly retry	Fix during development!

John Doe

- in technical/network errors, there's not that much we can do. the best thing might be to show some error page to the user to let the user know that something is wrong on our end, that we are sorry and that we are working on fixing the issue.
- expected errors is like we are interacting with a file or with a database that can fail. not very often. and it's not really expected for this to fail but that can happen. maybe because there too many simultaneous requests to a certain file, anything like that.
- in bugs/logical errors, we should test our code and we should fix such issues. these are not errors, we should handle at runtime.



## Working with Errors



- in all cases, we have got different ways of communicating with our users.

## \* Chapter 303: Analyzing The Error Handling In The Current Project





```
app.js
40  });
41  app.use(csrfProtection);
42  app.use(flash());
43
44  app.use((req, res, next) => {
45    if (!req.session.user) {
46      return next();
47    }
48    User.findById(req.session.user_id)
49      .then(user => {
50        req.user = user;
51        next();
52      })
53      .catch(err => console.log(err));
54  });
55
56  app.use((req, res, next) => {
57    res.locals.isAuthenticated = req.session.isLoggedIn;
58    res.locals.csrfToken = req.csrfToken();
59    next();
60  });
```

```
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node app.js`
(node:27527) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version
. To use the new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
```

The screenshot shows the VS Code editor with the `auth.js` file open. The file contains the following code:

```
72     validationErrors: errors.array()  
73   });  
74 }  
75  
76 User.findOne({ email: email })  
77 .then(user => {  
78   if (!user) {  
79     return res.status(422).render('auth/login', {  
80       path: '/login',  
81       pageTitle: 'Login',  
82       errorMessage: 'Invalid email or password.',  
83       oldInput: {  
84         email: email,  
85         password: password  
86       },  
87       validationErrors: []  
88     });  
89   }  
90   bcrypt  
91     .compare(password, user.password)  
92     .then(doMatch => {
```

The terminal window at the bottom shows the following output:

```
[nodemon] to restart at any time, enter `rs`  
[nodemon] watching: *.*  
[nodemon] starting `node app.js`  
(node:27527) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version  
To use the new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
```

- i check in login whether this email address exist and if not, i already return the same page with an error code where i pass that information that the input was invalid.





The screenshot shows the VS Code editor with the `auth.js` file open. The file contains the following code:

```
1  const express = require('express');  
2  const { check, body } = require('express-validator/check');  
3  
4  const authController = require('../controllers/auth');  
5  const User = require('../models/user');  
6  
7  const router = express.Router();  
8  
9  router.get('/login', authController.getLogin);  
10  
11 router.get('/signup', authController.getSignup);  
12  
13 router.post(  
14   '/login',  
15   [  
16     body('email')  
17       .isEmail()  
18       .withMessage('Please enter a valid email address.')  
19       .normalizeEmail(),  
20     body('password', 'Password has to be valid.')  
21       .isLength({ min: 5 })
```

The terminal window at the bottom shows the following output:

```
[nodemon] to restart at any time, enter `rs`  
[nodemon] watching: *.*  
[nodemon] starting `node app.js`  
(node:27527) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version  
To use the new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
```

```
9 router.get('/login', authController.getLogin);
10
11 router.get('/signup', authController.getSignup);
12
13 router.post(
14   '/login',
15   [
16     body('email')
17       .isEmail()
18       .withMessage('Please enter a valid email address.')
19       .normalizeEmail(),
20     body('password', 'Password has to be valid.')
21       .isLength({ min: 5 })
22       .isAlphanumeric()
23       .trim()
24   ],
25   authController.postLogin
26 );
27
28 router.post(
29   '/signup',
30   [
31     body('email')
32       .isEmail()
33       .withMessage('Please enter a valid email address.')
34       .normalizeEmail(),
35     body('password', 'Password has to be valid.')
36       .isLength({ min: 5 })
37       .isAlphanumeric()
38       .trim()
39   ],
40   authController.postSignup
41 );
42
43 exports.getLogin = authController.getLogin;
44 exports.getSignup = authController.getSignup;
45 exports.postLogin = authController.postLogin;
46 exports.postSignup = authController.postSignup;
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1: node

[nodemon] to restart at any time, enter `rs`  
[nodemon] watching: \*.\*  
[nodemon] starting `node app.js`  
(node:27527) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version  
. To use the new parser, pass option { useNewUrlParser: true } to MongoClient.connect.

- we do the same with the validation in ./routes/auth.js where we use the express-validator package to add built-in validation function. there behind the scene, this package also throws and handles errors and allows us to collect all these errors which are not these technical error object which is data managed by that package.





```
57
58 exports.postLogin = (req, res, next) => {
59   const email = req.body.email;
60   const password = req.body.password;
61
62   const errors = validationResult(req);
63   if (!errors.isEmpty()) {
64     return res.status(422).render('auth/login', {
65       path: '/login',
66       pageTitle: 'Login',
67       errorMessage: errors.array()[0].msg,
68       oldInput: {
69         email: email,
70         password: password
71       },
72       validationErrors: errors.array()
73     });
74   }
75
76   User.findOne({ email: email })
77     .then(user => {
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1: node

[nodemon] to restart at any time, enter `rs`  
[nodemon] watching: \*.\*  
[nodemon] starting `node app.js`  
(node:27527) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version  
. To use the new parser, pass option { useNewUrlParser: true } to MongoClient.connect.



```

57
58 exports.postLogin = (req, res, next) => {
59   const email = req.body.email;
60   const password = req.body.password;
61
62   const errors = validationResult(req);
63   if (!errors.isEmpty()) {
64     return res.status(422).render('auth/login', {
65       path: '/login',
66       pageTitle: 'Login',
67       errorMessage: errors.array()[0].msg,
68       oldInput: {
69         email: email,
70         password: password
71       },
72       validationErrors: errors.array()
73     });
74   }
75
76   User.findOne({ email: email })
77     .then(user => {

```

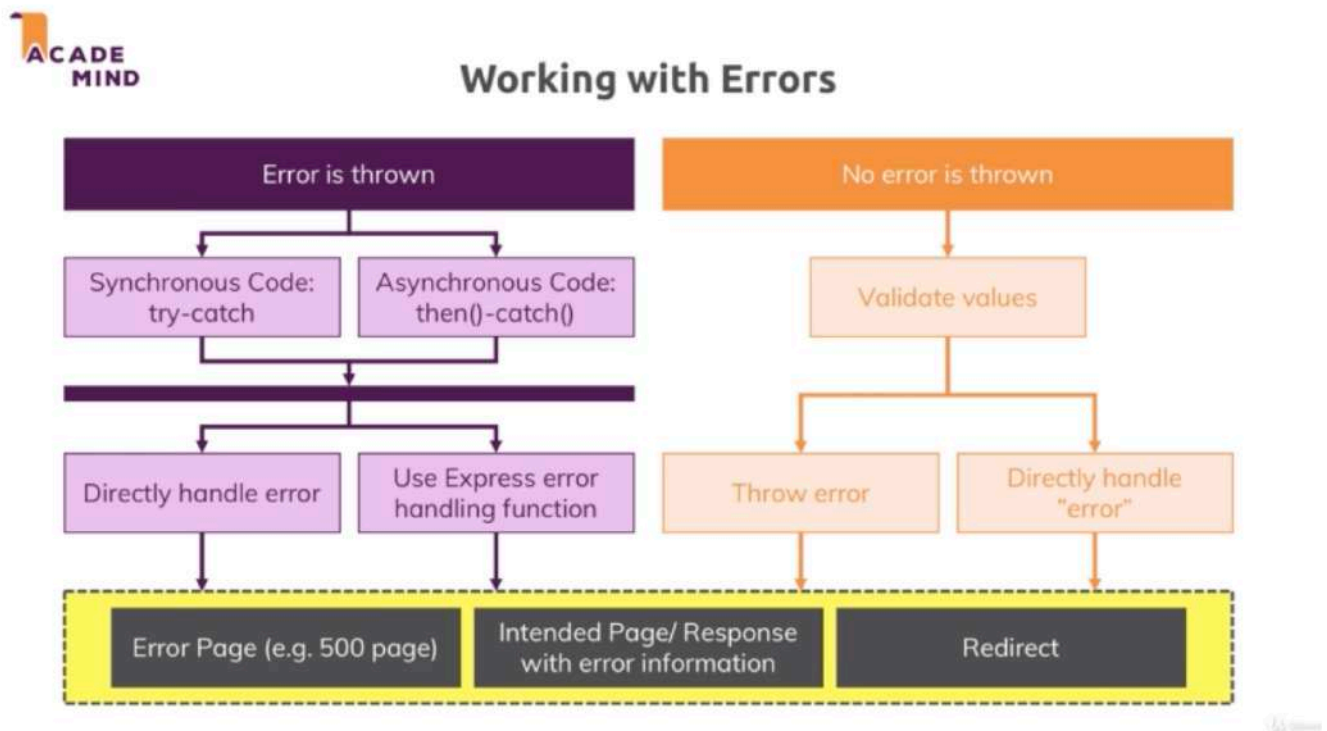
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```

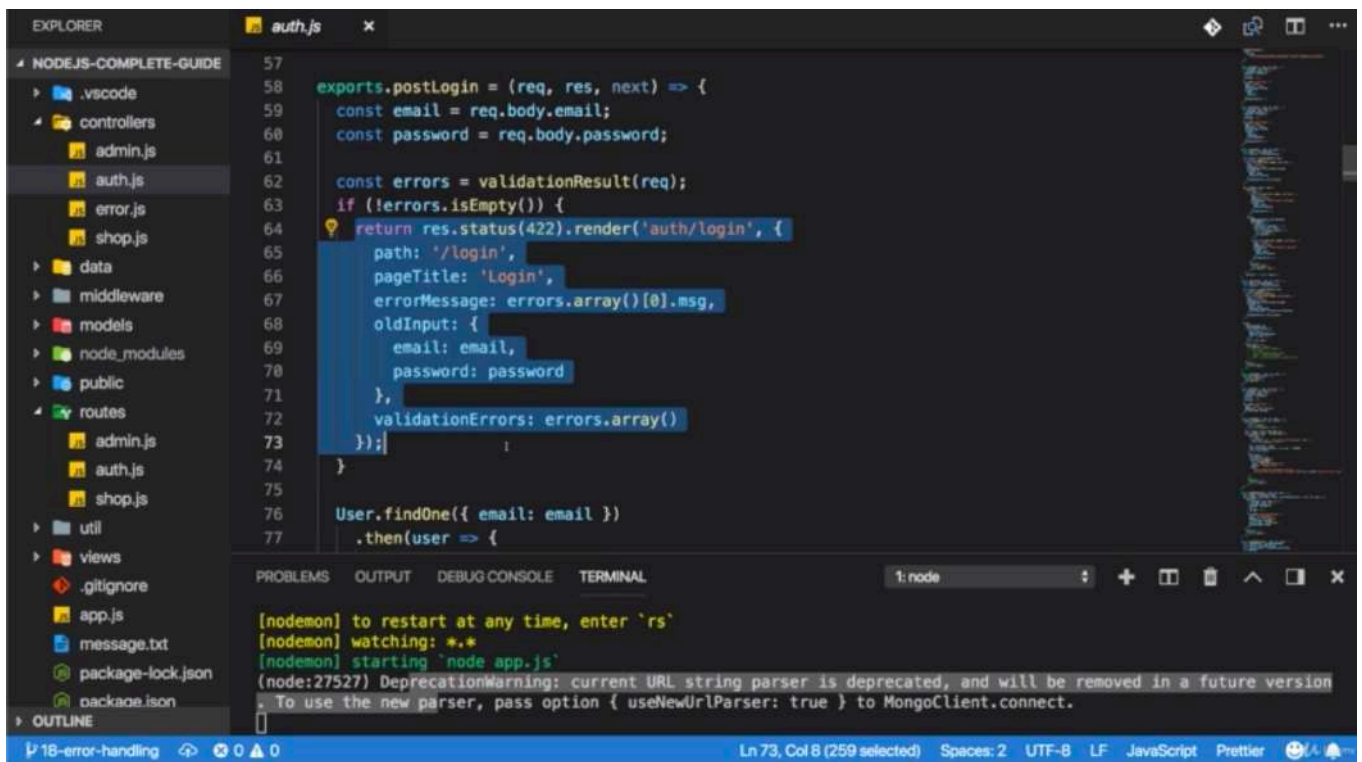
1: node
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node app.js`
(node:27527) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version
. To use the new parser, pass option { useNewUrlParser: true } to MongoClient.connect.

```

- we collect these errors and then we handle them manually  
 



- that would be the right side of this slide  
 



The screenshot shows the VS Code editor with the file explorer on the left displaying a project structure. The main editor window shows the file `auth.js` with the following code:

```
57
58 exports.postLogin = (req, res, next) => {
59   const email = req.body.email;
60   const password = req.body.password;
61
62   const errors = validationResult(req);
63   if (!errors.isEmpty()) {
64     return res.status(422).render('auth/login', {
65       path: '/login',
66       pageTitle: 'Login',
67       errorMessage: errors.array()[0].msg,
68       oldInput: {
69         email: email,
70         password: password
71       },
72       validationErrors: errors.array()
73     });
74   }
75
76   User.findOne({ email: email })
77     .then(user => {
```

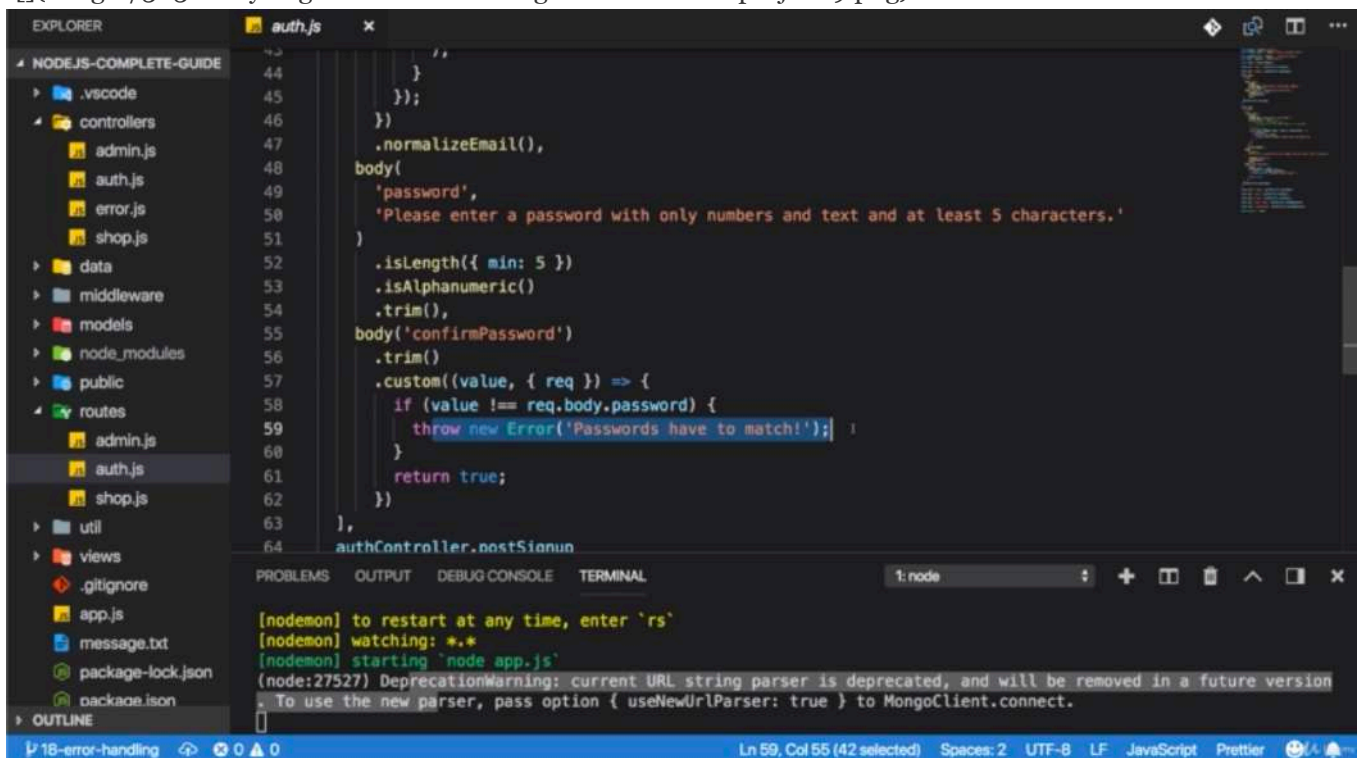
The bottom panel shows the terminal with the following output:

```
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node app.js`
(node:27527) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version
To use the new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
```

- and proceed our own

- technical errors can always be seen if you have an error message down the console log.





The screenshot shows the VS Code editor with the file explorer on the left displaying a project structure. The main editor window shows the file `auth.js` with the following code:

```
44   }
45   });
46 }
47 .normalizeEmail(),
48 body(
49   'password',
50   'Please enter a password with only numbers and text and at least 5 characters.'
51 )
52 .isLength({ min: 5 })
53 .isAlphanumeric()
54 .trim(),
55 body('confirmPassword')
56 .trim()
57 .custom((value, { req }) => {
58   if (value !== req.body.password) {
59     throw new Error('Passwords have to match!');
60   }
61   return true;
62 })
63 ],
64 authController.postSignIn
```

The bottom panel shows the terminal with the following output:

```
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node app.js`
(node:27527) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version
To use the new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
```

- in custom validator, i throw a technical error when passwords don't match.

- this error would bubble up and would be handled by express but this express-validator package happens to handle it.

## \* Chapter 304: Errors - Some Theory



The screenshot shows a VS Code editor with a file named `error-playground.js`. The code defines a function `sum(a, b)` that returns `a + b`. It is called with `sum(1)`, which results in `NaN` (Not a Number) being logged to the console. The terminal output shows the command `node error-playground.js` being executed twice, both resulting in `NaN`.

```
1 const sum = (a, b) => {
2   |
3   |   return a + b;
4   |
5   |
6   |   console.log(sum(1));
```

```
Maximilians-MBP:nodejs-complete-guide mschwarzmueller$ node error-playground.js
NaN
Maximilians-MBP:nodejs-complete-guide mschwarzmueller$ node error-playground.js
NaN
Maximilians-MBP:nodejs-complete-guide mschwarzmueller$
```

- if i fill in only one argument, then i get NaN(Not a Number) but not a technical error object.



The screenshot shows the same VS Code editor with `error-playground.js`. The code is updated to include a check for the number of arguments. If only one argument is provided, it throws a new `Error('Invalid arguments')`. The terminal output shows the command `node error-playground.js` being executed, resulting in a detailed error message: `Error: Invalid arguments` at `sum (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/error-playground.js:5:9)`.

```
1 const sum = (a, b) => {
2   |
3   |   return a + b;
4   |
5   |   throw new Error('Invalid arguments');
6   |
7   |
8   |   console.log(sum(1));
9   |
```

```
/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/error-playground.js:5
  throw new Error('Invalid arguments');
        ^
Error: Invalid arguments
    at sum (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/error-playground.js:5:9)
    at Object.<anonymous> (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/error-playground.js:8:13)
    at Module._compile (internal/modules/cjs/loader.js:689:30)
    at Object.Module._extensions..js (internal/modules/cjs/loader.js:700:10)
    at Module.load (internal/modules/cjs/loader.js:599:32)
    at tryModuleLoad (internal/modules/cjs/loader.js:538:12)
```

- here we have our own error message





```
1 const sum = (a, b) => {
2   if (a && b) {
3     return a + b;
4   }
5   throw new Error('Invalid arguments');
6 };
7
8 console.log(sum(1));
9
```

```
throw new Error('Invalid arguments');
^
Error: Invalid arguments
    at sum (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/error-playground.js:5:9)
    at Object.<anonymous> (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/error-playground.js:8:13)
    at Module._compile (internal/modules/cjs/loader.js:689:30)
    at Object.Module._extensions..js (internal/modules/cjs/loader.js:700:10)
    at Module.load (internal/modules/cjs/loader.js:599:32)
    at tryModuleLoad (internal/modules/cjs/loader.js:538:12)
    at Function.Module._load (internal/modules/cjs/loader.js:530:3)
```

- and then we got a callstack which allows us to find out at which function and which line number this error was thrown and what was called before that error.

- if we don't handle error, then our application just crashes.



```
1 const sum = (a, b) => {
2   if (a && b) {
3     return a + b;
4   }
5   throw new Error('Invalid arguments');
6 };
7
8 try {
9   console.log(sum(1));
10 } catch (error) {
11   console.log('Error occurred!');
12   console.log(error);
13 }
14
```

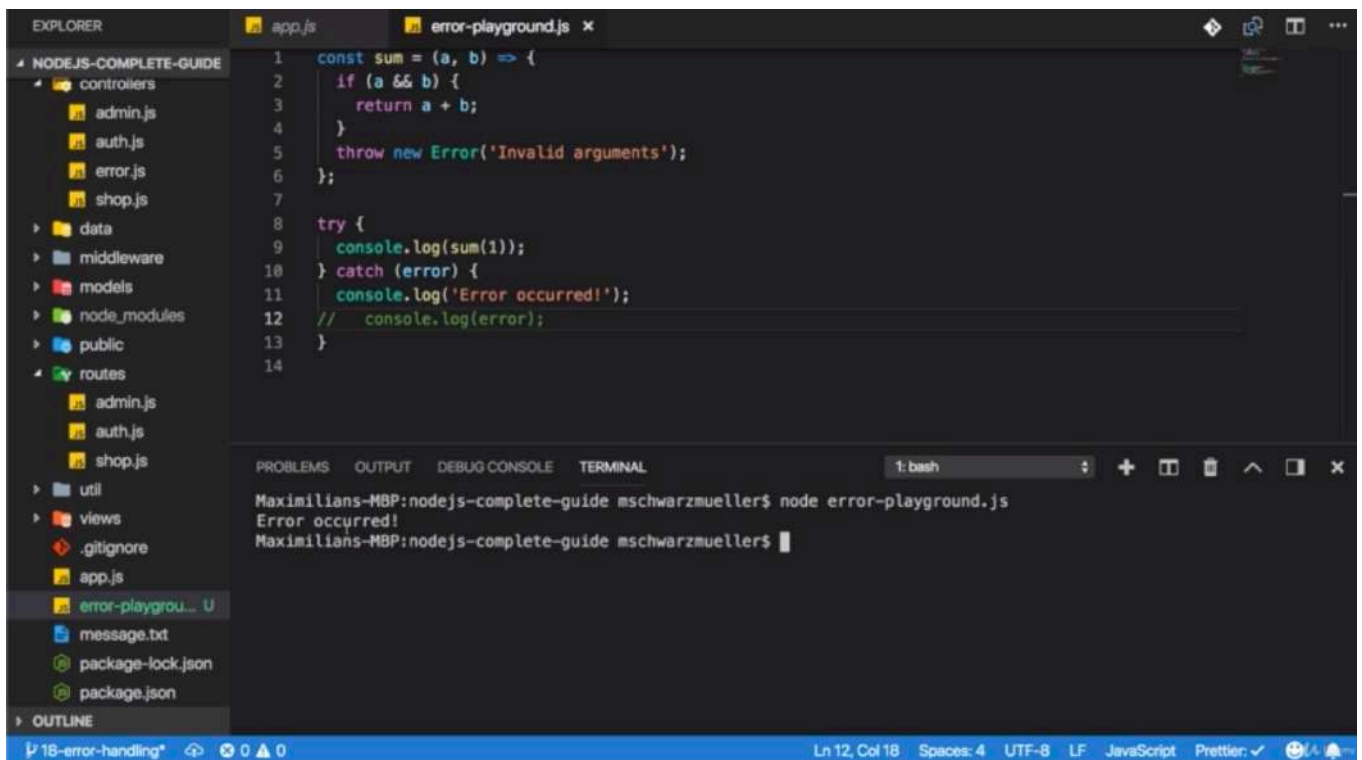
```
Maximilians-MBP:nodejs-complete-guide mschwarzmueller$ node error-playground.js
Error occurred!
Error: Invalid arguments
    at sum (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/error-playground.js:5:9)
    at Object.<anonymous> (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/error-playground.js:9:15)
    at Module._compile (internal/modules/cjs/loader.js:689:30)
    at Object.Module._extensions..js (internal/modules/cjs/loader.js:700:10)
    at Module.load (internal/modules/cjs/loader.js:599:32)
    at tryModuleLoad (internal/modules/cjs/loader.js:538:12)
    at Function.Module._load (internal/modules/cjs/loader.js:530:3)
    at Function.Module.runMain (internal/modules/cjs/loader.js:742:12)
```

- in synchronous code, use 'try and catch'

- i get this additional 'error occurred' message





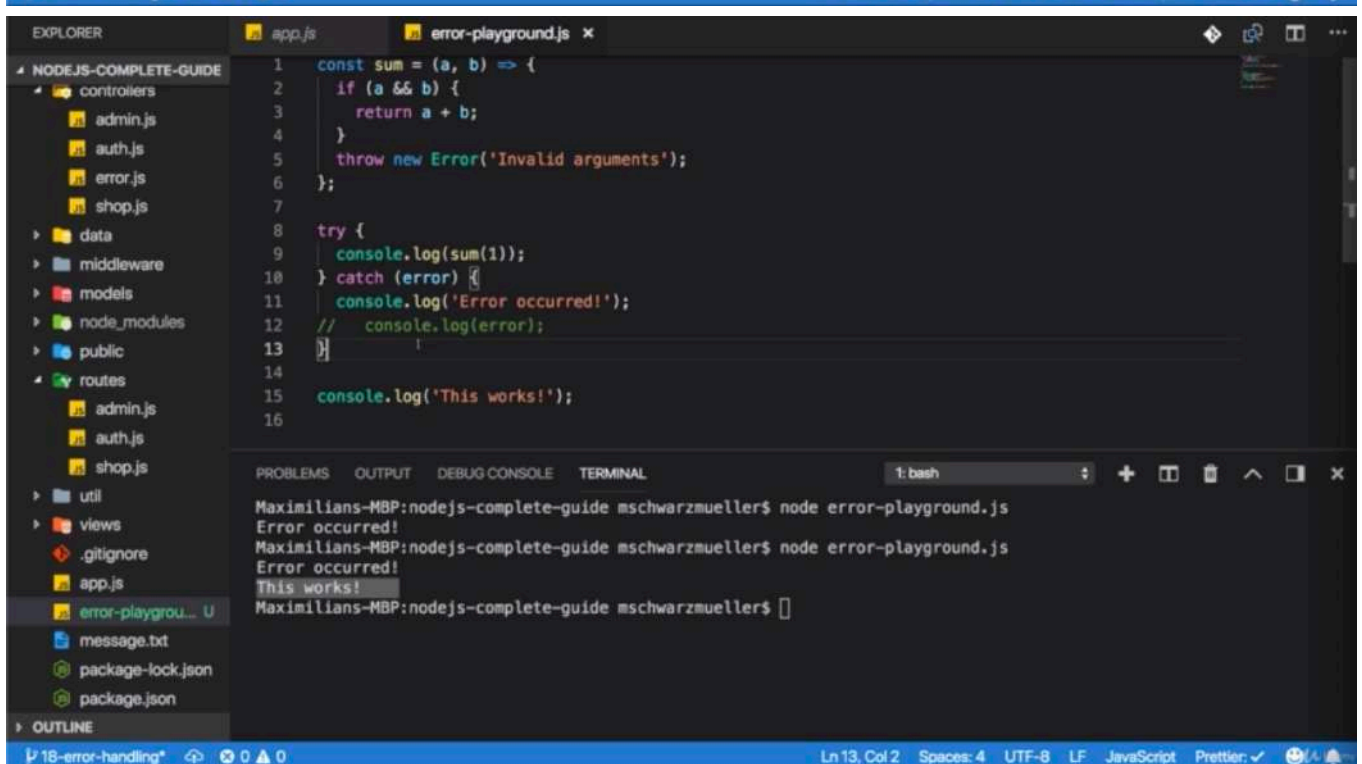


```
1 const sum = (a, b) => {
2   if (a && b) {
3     return a + b;
4   }
5   throw new Error('Invalid arguments');
6 };
7
8 try {
9   console.log(sum(1));
10 } catch (error) {
11   console.log('Error occurred!');
12   // console.log(error);
13 }
14
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1: bash

Maximilians-MBP:nodejs-complete-guide mschwarzmueller\$ node error-playground.js  
Error occurred!  
Maximilians-MBP:nodejs-complete-guide mschwarzmueller\$



```
1 const sum = (a, b) => {
2   if (a && b) {
3     return a + b;
4   }
5   throw new Error('Invalid arguments');
6 };
7
8 try {
9   console.log(sum(1));
10 } catch (error) {
11   console.log('Error occurred!');
12   // console.log(error);
13 }
14
15 console.log('This works!');
16
```

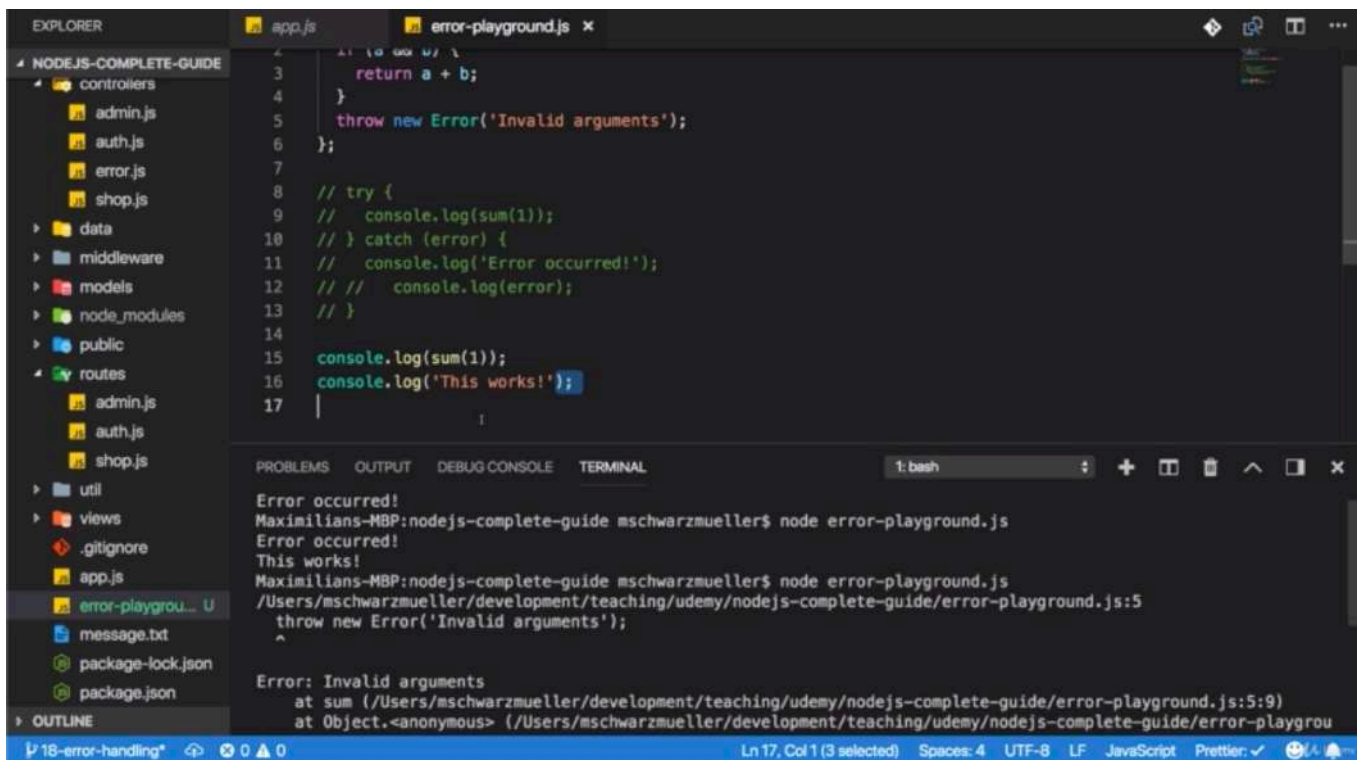
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1: bash

Maximilians-MBP:nodejs-complete-guide mschwarzmueller\$ node error-playground.js  
Error occurred!  
Maximilians-MBP:nodejs-complete-guide mschwarzmueller\$ node error-playground.js  
Error occurred!  
This works!  
Maximilians-MBP:nodejs-complete-guide mschwarzmueller\$

- and if i not log my error object, then i get just this. so then it doesn't crash and log it automatically. but we could do anything we want. we could continue with other code after this.





The screenshot shows a VS Code editor with a file explorer on the left. The file explorer shows a project structure for 'NODEJS-COMPLETE-GUIDE' with folders like 'controllers', 'data', 'middleware', 'models', 'node\_modules', 'public', 'routes', 'util', 'views', and 'gitignore'. The 'routes' folder is expanded, showing 'admin.js', 'auth.js', and 'shop.js'. The 'error-playground.js' file is open in the editor. The code in 'error-playground.js' is as follows:

```
1 // try {
2 //   console.log(sum(1));
3 // } catch (error) {
4 //   console.log('Error occurred!');
5 //   console.log(error);
6 // }
7
8 console.log(sum(1));
9 console.log('This works!');
```

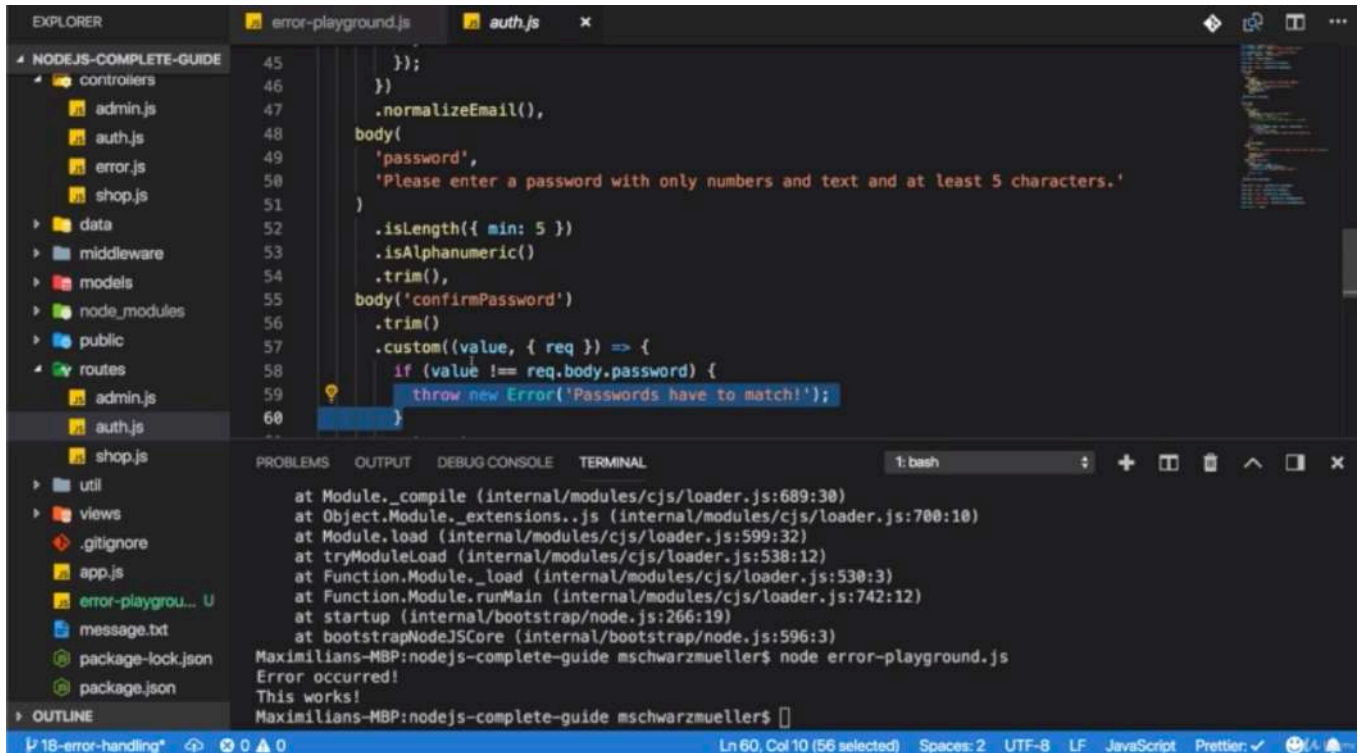
The terminal output shows the command 'node error-playground.js' being executed, resulting in the following output:

```
Maximilians-MBP:nodejs-complete-guide mschwarzmueller$ node error-playground.js
Error occurred!
This works!
Maximilians-MBP:nodejs-complete-guide mschwarzmueller$ node error-playground.js
/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/error-playground.js:5
  throw new Error('Invalid arguments');
        ^
Error: Invalid arguments
    at sum (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/error-playground.js:5:9)
    at Object.<anonymous> (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/error-playground.js:5:9)
```

- just to demonstrate this, if i comment out try and catch, and i just try to console log sum(1), so i call this with an error being thrown, then we don't see 'this works' anywhere because it crashes with our error that is being thrown. and this doesn't continue with other code.

- this is why handling code like try and catch is a good thing to do because this ensures that we can continue with code, that we can handle this gracefully, in our node express application. we could send an error response which renders a valid page without crashing everything but which informs the user that something bad happened.





The screenshot shows a VS Code editor with a file explorer on the left. The file explorer shows a project structure for 'NODEJS-COMPLETE-GUIDE' with folders like 'controllers', 'data', 'middleware', 'models', 'node\_modules', 'public', 'routes', 'util', 'views', and 'gitignore'. The 'routes' folder is expanded, showing 'admin.js', 'auth.js', and 'shop.js'. The 'auth.js' file is open in the editor. The code in 'auth.js' is as follows:

```
45 // ...
46 // ...
47 // ...
48 // ...
49 // ...
50 // ...
51 // ...
52 // ...
53 // ...
54 // ...
55 // ...
56 // ...
57 // ...
58 // ...
59 // ...
60 // ...
```

The terminal output shows the command 'node error-playground.js' being executed, resulting in the following output:

```
Maximilians-MBP:nodejs-complete-guide mschwarzmueller$ node error-playground.js
Error occurred!
This works!
Maximilians-MBP:nodejs-complete-guide mschwarzmueller$
```

- and this is what the express validator package does for us with our thrown error. in ./routes/auth.js, in the end express-validator catches this and then just adds it to its own error array and allows us to read that list of errors it caught.







```
72     validationErrors: errors.array()  
73   });  
74 }  
75  
76 User.findOne({ email: email })  
77 .then(user => {  
78   if (!user) {  
79     return res.status(422).render('auth/login', {  
80       path: '/login',  
81       pageTitle: 'Login',  
82       errorMessage: 'Invalid email or password.',  
83       oldInput: {  
84         email: email,  
85         password: password  
86       },  
87       validationErrors: []  
88     });  
89   }  
90 }  
91 )
```

```
105     oldInput: {  
106       email: email,  
107       password: password  
108     },  
109     validationErrors: []  
110   });  
111 }  
112 .catch(err => {  
113   console.log(err);  
114   res.redirect('/login');  
115 });  
116 }  
117 .catch(err => console.log(err));  
118 };  
119  
120 exports.postSignup = (req, res, next) => {  
121   const email = req.body.email;
```

- we also have asynchronous operations that can fail and such operations when using promises are handled with 'then and catch'
- so if the database operation fails because we don't have read access because the database server is down temporarily, anything like that. then we make it into this catch block.
- 'catch()' collects all errors that are thrown by any prior then block. so if we had more than 'then()' block in our chain here, catch would fire on any error thrown in any 'then()' block or any operation executed in a 'then()' block.

## \* Chapter 305: Throwing Errors In Code

1. update
- app.js



```

1 //app.js
2
3 const path = require('path');
4
5 const express = require('express');
6 const bodyParser = require('body-parser');
7 const mongoose = require('mongoose');
8 const session = require('express-session');
9 const MongoDBStore = require('connect-mongodb-session')(session);
10 const csrf = require('csrf');
11 const flash = require('connect-flash');
12
13 const errorController = require('./controllers/error');
14 const User = require('./models/user');
15
16 const MONGODB_URI =
17 'mongodb+srv://maximilian:rldnjs12@cluster0-z3v1k.mongodb.net/shop'
18
19 const app = express();
20 const store = new MongoDBStore({
21   uri: MONGODB_URI,
22   collection: 'sessions'
23 });
24 const csrfProtection = csrf();
25
26 app.set('view engine', 'ejs');
27 app.set('views', 'views');
28
29 const adminRoutes = require('./routes/admin');
30 const shopRoutes = require('./routes/shop');
31 const authRoutes = require('./routes/auth');
32
33 app.use(bodyParser.urlencoded({ extended: false }));
34 app.use(express.static(path.join(__dirname, 'public')));
35 app.use(
36   session({
37     secret: 'my secret',
38     resave: false,
39     saveUninitialized: false,
40     store: store
41   })
42 );
43 app.use(csrfProtection);
44 app.use(flash());
45
46 app.use((req, res, next) => {
47   if (!req.session.user) {
48     /**if '!req.session.user is true'
49     * then return 'next()'
50     * and execute next 'app.use()' right below
51     *
52     * if i wouldn't not add this check,
53     * it could try to find a user without the session object existing
54     * and that would then crash our app.
55     */
56     return next();

```

```

57 }
58 User.findById(req.session.user._id)
59   .then(user => {
60     if(!user) {
61       return next()
62     }
63     req.user = user;
64     next();
65   })
66   /**any potential errors that might be happening
67    * because this 'catch()' block will not fire
68    * if i don't find the user with this 'req.session.user._id'
69    * it will only fire if there are any technical issues,
70    * if the database is down
71    * or if the user of this app doesn't have sufficient permissions to execute this action
72    */
73   .catch(err => {
74     /**here in the 'catch' block,
75      * logging it is not useful.
76      * it will make more sense to throw a new Error here
77      * where we wrap the error object here
78      *
79      * throwing this error has a significant advantage
80      *
81      *   throw new Error(err)
82      *
83      * if we have some technical issue,
84      * we throw a real error
85      * as it turns out express.js gives us a way of taking care of such errors.
86      * that is why i'm doing like this.
87      */
88     throw new Error(err)
89     /**alternatively, we could simply call 'next()' to continue without req.user being set
90     */
91   });
92 });
93
94 app.use((req, res, next) => {
95   res.locals.isAuthenticated = req.session.isLoggedIn;
96   res.locals.csrfToken = req.csrfToken();
97   next();
98 });
99
100 app.use('/admin', adminRoutes);
101 app.use(shopRoutes);
102 app.use(authRoutes);
103
104 app.use(errorController.get404);
105
106 mongoose
107   .connect(MONGODB_URI)
108   .then(result => {
109     app.listen(3000);
110   })
111   .catch(err => {
112     console.log(err);

```

```
113   });  
114
```

## \* Chapter 306: Returning Error Pages

1. update
  - ./controllers/admin.js
  - ./views/admin/edit-product.ejs
  - ./views/500.ejs
  - ./controllers/error.js
  - app.js

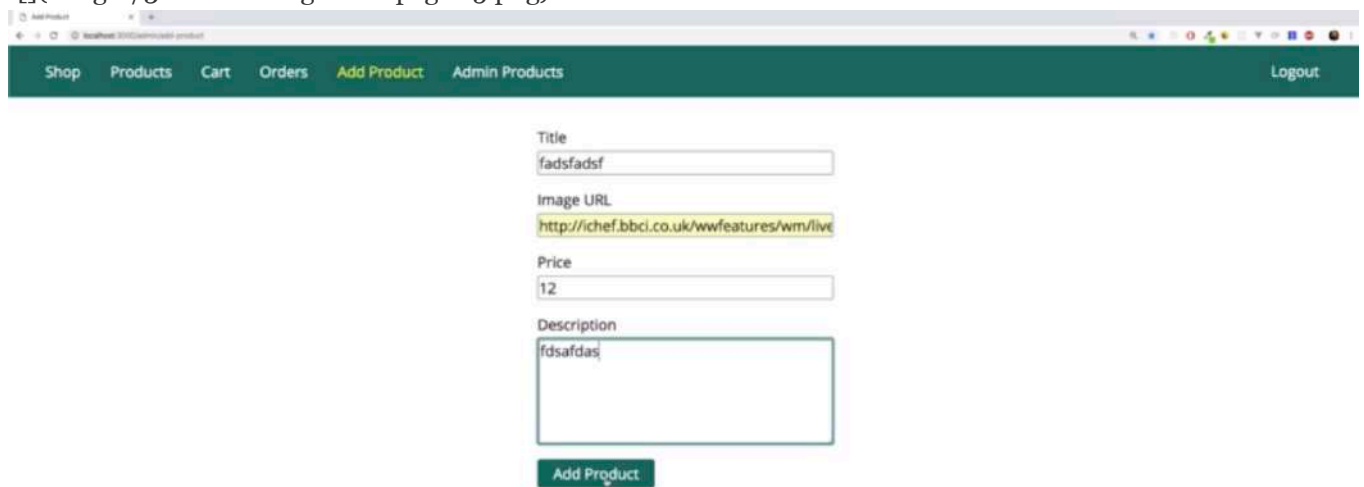












The screenshot shows a web browser window displaying a product management interface. The browser's address bar shows the URL "localhost:3000/admin/edit-product". The page has a dark green header with navigation links: "Shop", "Products", "Cart", "Orders", "Add Product" (highlighted in yellow), and "Admin Products". A "Logout" link is visible in the top right corner. The main content area contains a form with the following fields:

- Title:** A text input field containing the value "fadsfadsf".
- Image URL:** A text input field containing the value "http://chef.bbc.co.uk/ww/features/wm/live".
- Price:** A text input field containing the value "12".
- Description:** A text area containing the value "fdsafdas".

Below the form is a green button labeled "Add Product".

EXPLORER

app.js admin.js

NODEJS-COMPLETE-GUIDE

- .vscode
- controllers
  - admin.js
  - auth.js
  - error.js
  - shop.js
- data
- middleware
- models
- node\_modules
- public
- routes
  - admin.js
  - auth.js
  - shop.js
- util
- views
- .gitignore
- app.js
- message.txt
- package-lock.json
- package.json

49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65

```
    userId: req.user
  });
  product
    .save()
    .then(result => {
      // console.log(result);
      console.log('Created Product');
      res.redirect('/admin/products');
    })
    .catch(err => {
      console.log('An error occurred!');
      console.log(err);
    });
});

exports.getEditProduct = (req, res, next) => {
  const editMode = req.query.edit;
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1: node

```
[Symbol(mongoErrorContextSymbol)]: {} }
[nodemon] restarting due to changes...
[nodemon] starting 'node app.js'
(node:28096) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version.
To use the new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
An error occurred!
{ MongoError: E11000 duplicate key error collection: shop.products index: _id_ dup key: { : ObjectId('5badf72403fd8b5be0366e81') } }
    at Function.create (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/node_modules/mongodb-core/lib/error.js:43:12)
    at toError (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/node_modules/mongodb/lib/
```

18-error-handling\* 0 0 Ln 59, Col 41 Spaces: 2 UTF-8 LF JavaScript Prettier: ✓

Shop Products Cart Orders Add Product Admin Products Logout

Title

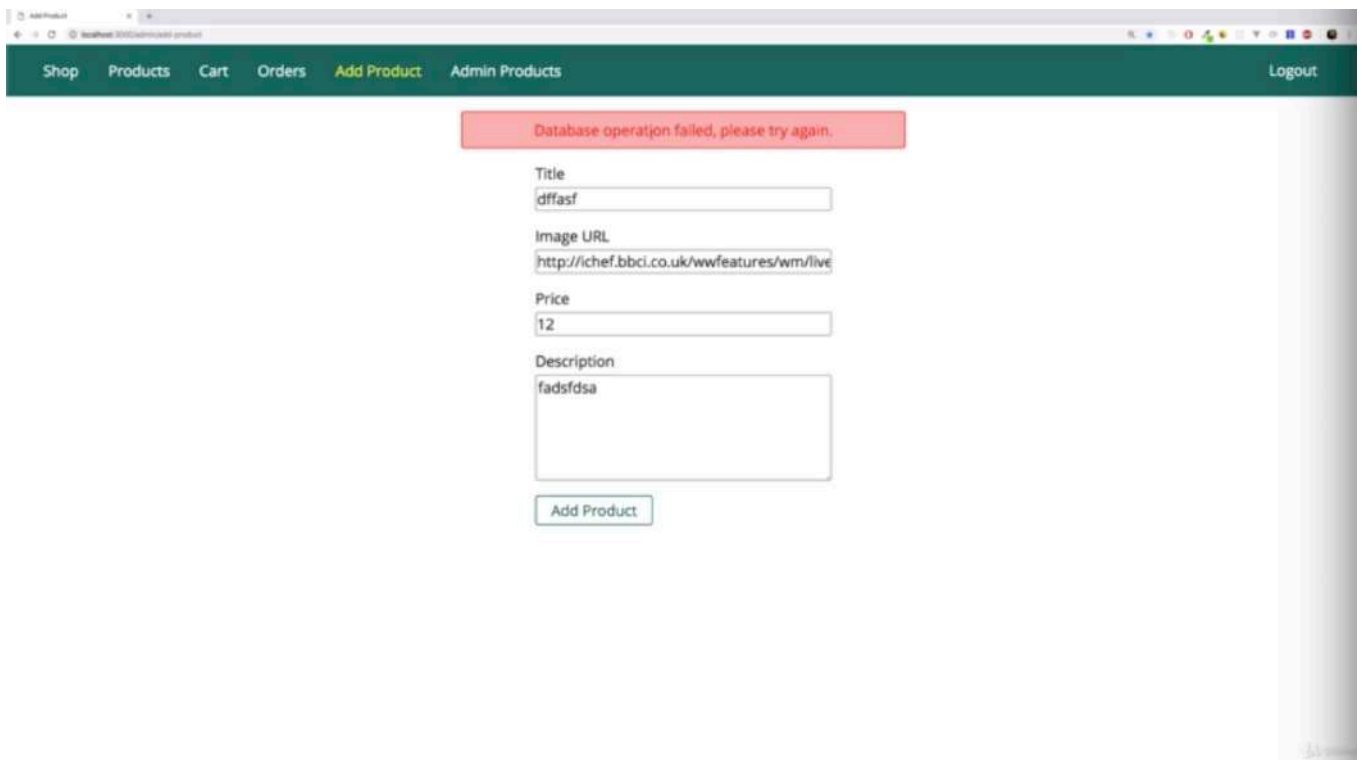
Image URL

Price

Description

Add Product





```
1 // ./controllers/admin.js
2
3 const mongoose = require('mongoose');
4
5 const { validationResult } = require('express-validator/check');
6
7 const Product = require('../models/product');
8
9 exports.getAddProduct = (req, res, next) => {
10   res.render('admin/edit-product', {
11     pageTitle: 'Add Product',
12     path: '/admin/add-product',
13     editing: false,
14     hasError: false,
```

```

15     errorMessage: null,
16     validationErrors: []
17   });
18 };
19
20 exports.postAddProduct = (req, res, next) => {
21   const title = req.body.title;
22   const imageUrl = req.body.imageUrl;
23   const price = req.body.price;
24   const description = req.body.description;
25   const errors = validationResult(req);
26
27   if (!errors.isEmpty()) {
28     console.log(errors.array());
29     return res.status(422).render('admin/edit-product', {
30       pageTitle: 'Add Product',
31       path: '/admin/add-product',
32       editing: false,
33       hasError: true,
34       product: {
35         title: title,
36         imageUrl: imageUrl,
37         price: price,
38         description: description
39       },
40       errorMessage: errors.array()[0].msg,
41       validationErrors: errors.array()
42     });
43   }
44
45   const product = new Product({
46     _id: new mongoose.Types.ObjectId('5badf72403fd8b5be0366e81'),
47     title: title,
48     price: price,
49     description: description,
50     imageUrl: imageUrl,
51     userId: req.user
52   });
53   product
54     .save()
55     .then(result => {
56       // console.log(result);
57       console.log('Created Product');
58       res.redirect('/admin/products');
59     })
60     .catch(err => {
61       // return res.status(500).render('admin/edit-product', {
62       //   pageTitle: 'Add Product',
63       //   path: '/admin/add-product',
64       //   editing: false,
65       //   hasError: true,
66       //   product: {
67       //     title: title,
68       //     imageUrl: imageUrl,
69       //     price: price,
70       //     description: description

```

```

71     // },
72     // errorMessage: 'Database operation failed, please try again.',
73     // validationErrors: []
74     // });
75     res.redirect('/500');
76 });
77 };
78
79 exports.getEditProduct = (req, res, next) => {
80     const editMode = req.query.edit;
81     if (!editMode) {
82         return res.redirect('/');
83     }
84     const prodId = req.params.productId;
85     Product.findById(prodId)
86         .then(product => {
87             if (!product) {
88                 return res.redirect('/');
89             }
90             res.render('admin/edit-product', {
91                 pageTitle: 'Edit Product',
92                 path: '/admin/edit-product',
93                 editing: editMode,
94                 product: product,
95                 hasError: false,
96                 errorMessage: null,
97                 validationErrors: []
98             });
99         })
100         .catch(err => console.log(err));
101 };
102
103 exports.postEditProduct = (req, res, next) => {
104     const prodId = req.body.productId;
105     const updatedTitle = req.body.title;
106     const updatedPrice = req.body.price;
107     const updatedImageUrl = req.body.imageUrl;
108     const updatedDesc = req.body.description;
109
110     const errors = validationResult(req);
111
112     if (!errors.isEmpty()) {
113         return res.status(422).render('admin/edit-product', {
114             pageTitle: 'Edit Product',
115             path: '/admin/edit-product',
116             editing: true,
117             hasError: true,
118             product: {
119                 title: updatedTitle,
120                 imageUrl: updatedImageUrl,
121                 price: updatedPrice,
122                 description: updatedDesc,
123                 _id: prodId
124             },
125             errorMessage: errors.array()[0].msg,
126             validationErrors: errors.array()

```

```

127     });
128   }
129
130   Product.findById(prodId)
131     .then(product => {
132       if (product.userId.toString() !== req.user._id.toString()) {
133         return res.redirect('/');
134       }
135       product.title = updatedTitle;
136       product.price = updatedPrice;
137       product.description = updatedDesc;
138       product.imageUrl = updatedImageUrl;
139       return product.save().then(result => {
140         console.log('UPDATED PRODUCT!');
141         res.redirect('/admin/products');
142       });
143     })
144     .catch(err => console.log(err));
145   });
146
147   exports.getProducts = (req, res, next) => {
148     Product.find({ userId: req.user._id })
149       // .select('title price _id')
150       // .populate('userId', 'name')
151       .then(products => {
152         console.log(products);
153         res.render('admin/products', {
154           prods: products,
155           pageTitle: 'Admin Products',
156           path: '/admin/products'
157         });
158       })
159       .catch(err => console.log(err));
160   });
161
162   exports.postDeleteProduct = (req, res, next) => {
163     const prodId = req.body.productId;
164     Product.deleteOne({ _id: prodId, userId: req.user._id })
165       .then(() => {
166         console.log('DESTROYED PRODUCT');
167         res.redirect('/admin/products');
168       })
169       .catch(err => console.log(err));
170   });
171

```

```

1 <!--./views/admin/edit-product.ejs-->
2
3 <%- include('../includes/head.ejs') %>
4   <link rel="stylesheet" href="/css/forms.css">
5   <link rel="stylesheet" href="/css/product.css">
6 </head>
7
8 <body>
9   <%- include('../includes/navigation.ejs') %>
10
11   <main>

```



```

12     <% if (errorMessage) { %>
13         <div class="user-message user-message--error"><%= errorMessage %></div>
14     <% } %>
15     <form class="product-form" action="/admin/<% if (editing) { %>edit-product<% } else
{ %>add-product<% } %>" method="POST">
16         <div class="form-control">
17             <label for="title">Title</label>
18             <input
19                 class="<%= validationErrors.find(e => e.param === 'title') ? 'invalid' :
'' %>"
20                 type="text"
21                 name="title"
22                 id="title"
23                 value="<% if (editing || hasError) { %><%= product.title %><% } %>">
24         </div>
25         <div class="form-control">
26             <label for="imageUrl">Image URL</label>
27             <input
28                 class="<%= validationErrors.find(e => e.param === 'imageUrl') ?
'invalid' : '' %>"
29                 type="text"
30                 name="imageUrl"
31                 id="imageUrl"
32                 value="<% if (editing || hasError) { %><%= product.imageUrl %><% } %>">
33         </div>
34         <div class="form-control">
35             <label for="price">Price</label>
36             <input
37                 class="<%= validationErrors.find(e => e.param === 'price') ? 'invalid' :
'' %>"
38                 type="number"
39                 name="price"
40                 id="price"
41                 step="0.01"
42                 value="<% if (editing || hasError) { %><%= product.price %><% } %>">
43         </div>
44         <div class="form-control">
45             <label for="description">Description</label>
46             <textarea
47                 class="<%= validationErrors.find(e => e.param === 'description') ?
'invalid' : '' %>"
48                 name="description"
49                 id="description"
50                 rows="5"><% if (editing || hasError) { %><%= product.description %><% }
%></textarea>
51         </div>
52         <% if (editing) { %>
53             <input type="hidden" value="<%= product._id %>" name="productId">
54             <% } %>
55
56         <input type="hidden" name="_csrf" value="<%= csrfToken %>">
57         <button class="btn" type="submit"><% if (editing) { %>Update Product<% } else {
%>Add Product<% } %></button>
58     </form>
59 </main>
60 <%- include('../includes/end.ejs') %>

```

```

1 <!--./views/500.ejs-->
2
3 <%- include('includes/head.ejs') %>
4 </head>
5
6 <body>
7   <%- include('includes/navigation.ejs') %>
8   <h1>Some error occurred!</h1>
9   <p>We're working on fixing this, sorry for the inconvenience!</p>
10
11 <%- include('includes/end.ejs') %>

```

```

1 //./controllers/error.js
2
3 exports.get404 = (req, res, next) => {
4   res.status(404).render('404', {
5     pageTitle: 'Page Not Found',
6     path: '/404',
7     isAuthenticated: req.session.isLoggedIn
8   });
9 };
10
11 exports.get500 = (req, res, next) => {
12   res.status(500).render('500', {
13     pageTitle: 'Error!',
14     path: '/500',
15     isAuthenticated: req.session.isLoggedIn
16   });
17 };
18

```

```

1 //app.js
2
3 const path = require('path');
4
5 const express = require('express');
6 const bodyParser = require('body-parser');
7 const mongoose = require('mongoose');
8 const session = require('express-session');
9 const MongoDBStore = require('connect-mongodb-session')(session);
10 const csrf = require('csrf');
11 const flash = require('connect-flash');
12
13 const errorController = require('./controllers/error');
14 const User = require('./models/user');
15
16 const MONGODB_URI =
17   'mongodb+srv://maximilian:rldnjs12@cluster0-z3v1k.mongodb.net/shop'
18
19 const app = express();
20 const store = new MongoDBStore({
21   uri: MONGODB_URI,
22   collection: 'sessions'
23 });
24 const csrfProtection = csrf();
25
26 app.set('view engine', 'ejs');

```

```

27 app.set('views', 'views');
28
29 const adminRoutes = require('./routes/admin');
30 const shopRoutes = require('./routes/shop');
31 const authRoutes = require('./routes/auth');
32
33 app.use(bodyParser.urlencoded({ extended: false }));
34 app.use(express.static(path.join(__dirname, 'public')));
35 app.use(
36   session({
37     secret: 'my secret',
38     resave: false,
39     saveUninitialized: false,
40     store: store
41   })
42 );
43 app.use(csrfProtection);
44 app.use(flash());
45
46 app.use((req, res, next) => {
47   if (!req.session.user) {
48     return next();
49   }
50   User.findById(req.session.user._id)
51     .then(user => {
52       if (!user) {
53         return next();
54       }
55       req.user = user;
56       next();
57     })
58     .catch(err => {
59       throw new Error(err);
60     });
61 });
62
63 app.use((req, res, next) => {
64   res.locals.isAuthenticated = req.session.isLoggedIn;
65   res.locals.csrfToken = req.csrfToken();
66   next();
67 });
68
69 app.use('/admin', adminRoutes);
70 app.use(shopRoutes);
71 app.use(authRoutes);
72
73 app.get('/500', errorController.get500);
74
75 app.use(errorController.get404);
76
77 mongoose
78   .connect(MONGODB_URI)
79   .then(result => {
80     app.listen(3000);
81   })
82   .catch(err => {

```

```
83     console.log(err);
84   });
85 }
```

## \* Chapter 307: Returning Error Pages

1.update

- ./controllers/admin.js
- app.js



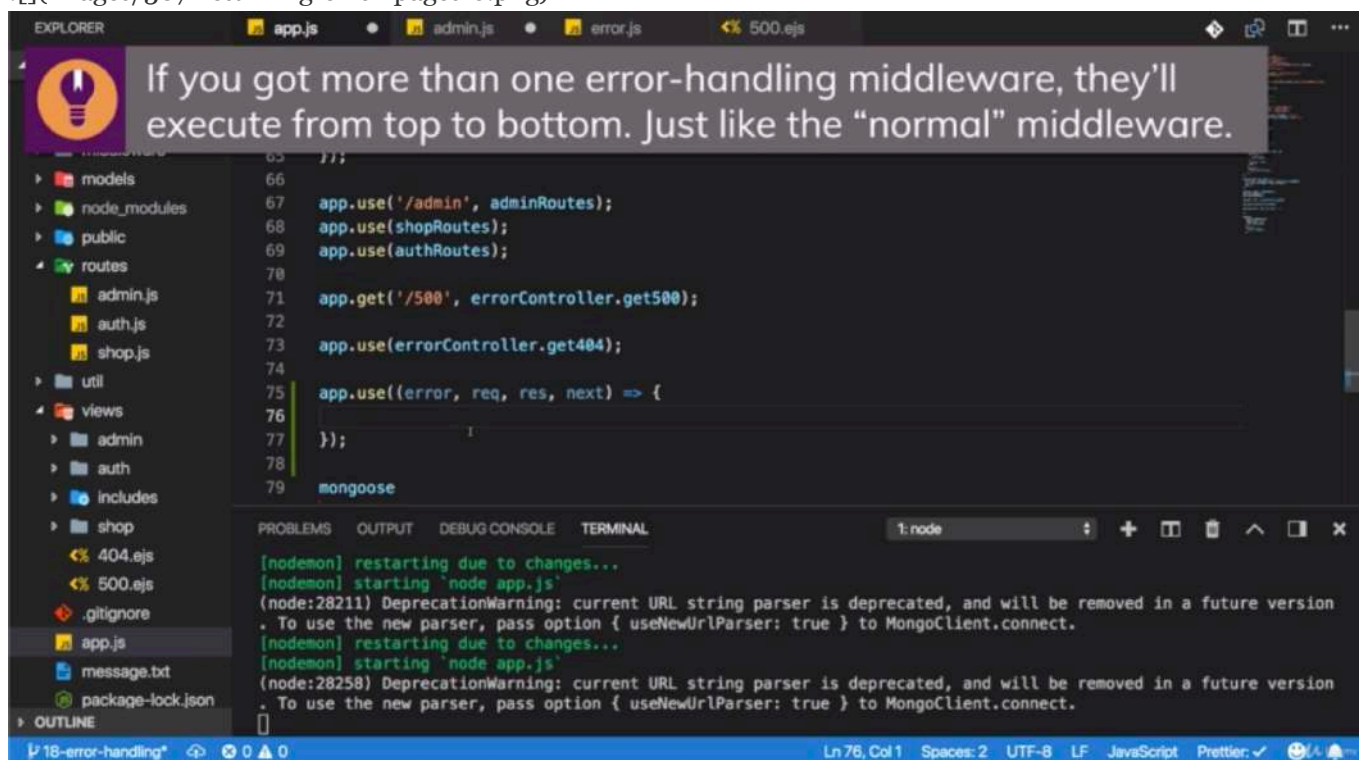














Add Product

ShopProductsCartOrdersAdd ProductAdmin ProductsLogout

Title

dasfdas

Image URL

http://ichef.bbci.co.uk/ww/features/wm/live

Price

12

Description

fdafdasf

Add Prgduct

Some error occurred!

We're working on fixing this, sorry for the inconvenience!

EXPLORER

- NODEJS-COMPLETE-GUIDE
  - .vscode
  - controllers
    - admin.js
    - auth.js
    - error.js
    - shop.js
  - data
  - middleware
  - models
  - node\_modules
  - public
  - routes
    - admin.js
    - auth.js
    - shop.js
  - util
  - views
    - admin
    - auth
    - includes
    - shop
- 404.ejs

admin.js

```
76     return next(error);
77   });
78 };
79
80 exports.getEditProduct = (req, res, next) => {
81   const editMode = req.query.edit;
82   if (!editMode) {
83     return res.redirect('/');
84   }
85   const prodId = req.params.productId;
86   Product.findById(prodId)
87     .then(product => {
88       throw new Error('Dummy');
89     })
90     .if (!product) {
91       return res.redirect('/');
92     }
93     res.render('admin/edit-product', {
94       pageTitle: 'Edit Product',
95       path: '/admin/edit-product',
96     });
97   }
98 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1: node

```
{ _id: 5bb20d1bb45d20386d850465,
  title: 'Third Book!',
  price: 22.99,
  description: 'fasfasdfdsfs',
  imageUrl:
    'http://ichef.bbci.co.uk/ww/features/wm/live/1280_640/images/live/p0/2v/dp/p02vdpfn.jpg',
  userId: 5bb209393fa36b3687f778cd,
  __v: 0 } }
```

Ln 88, Col 1 (31 selected) Spaces: 2 UTF-8 LF JavaScript Prettier: ✓

Shop Products Cart Orders Add Product Admin Products Logout

First Book



\$ 12.99

Does this work?

Edit Delete

Third Book!



\$ 22.99

fasfasdfdsfs

Edit Delete

Waiting for screenshot...

3/3 screens

## Some error occurred!

We're working on fixing this, sorry for the inconvenience!

- this error handler will not execute for 404 errors. there we still handle this manually because technically, the 404 error is simply just a valid URL which we catch with our catch all handler where we then just happen to render 404 page. it's not a technical error object that gets created at any point here.

```
1 // ./controllers/admin.js
2
3 const mongoose = require('mongoose');
4
5 const { validationResult } = require('express-validator/check');
6
7 const Product = require('../models/product');
8
9 exports.getAddProduct = (req, res, next) => {
10   res.render('admin/edit-product', {
11     pageTitle: 'Add Product',
12     path: '/admin/add-product',
13     editing: false,
14     hasError: false,
15     errorMessage: null,
16     validationErrors: []
17   });
18 };
19
20 exports.postAddProduct = (req, res, next) => {
21   const title = req.body.title;
22   const imageUrl = req.body.imageUrl;
23   const price = req.body.price;
24   const description = req.body.description;
25   const errors = validationResult(req);
26
27   if (!errors.isEmpty()) {
28     console.log(errors.array());
29     return res.status(422).render('admin/edit-product', {
30       pageTitle: 'Add Product',
31       path: '/admin/add-product',
```

```

32     editing: false,
33     hasError: true,
34     product: {
35         title: title,
36         imageUrl: imageUrl,
37         price: price,
38         description: description
39     },
40     errorMessage: errors.array()[0].msg,
41     validationErrors: errors.array()
42 });
43 }
44
45 const product = new Product({
46     _id: new mongoose.Types.ObjectId('5badf72403fd8b5be0366e81'),
47     title: title,
48     price: price,
49     description: description,
50     imageUrl: imageUrl,
51     userId: req.user
52 });
53 product
54     .save()
55     .then(result => {
56         // console.log(result);
57         console.log('Created Product');
58         res.redirect('/admin/products');
59     })
60     .catch(err => {
61         // return res.status(500).render('admin/edit-product', {
62         //     pageTitle: 'Add Product',
63         //     path: '/admin/add-product',
64         //     editing: false,
65         //     hasError: true,
66         //     product: {
67         //         title: title,
68         //         imageUrl: imageUrl,
69         //         price: price,
70         //         description: description
71         //     },
72         //     errorMessage: 'Database operation failed, please try again.',
73         //     validationErrors: []
74         // });
75         //res.redirect('/500');
76         const error = new Error(err)
77         error.httpStatusCode = 500
78         /**when we call 'next' as an error passed as an argument,
79          * then we let express know that an error occurred
80          * and it will skip all other middlewares
81          * and move right away to an error handling middleware
82          * so 'next(error)' is the trick with an error object being passed instead of throwing
83          it.
84          */
85         return next(error)
86     });

```

```

87 };
88
89 exports.getEditProduct = (req, res, next) => {
90   const editMode = req.query.edit;
91   if (!editMode) {
92     return res.redirect('/');
93   }
94   const prodId = req.params.productId;
95   Product.findById(prodId)
96     .then(product => {
97       if (!product) {
98         return res.redirect('/');
99       }
100       res.render('admin/edit-product', {
101         pageTitle: 'Edit Product',
102         path: '/admin/edit-product',
103         editing: editMode,
104         product: product,
105         hasError: false,
106         errorMessage: null,
107         validationErrors: []
108       });
109     })
110     .catch(err => {
111       const error = new Error(err)
112       error.httpStatusCode = 500
113       return next(error)
114     });
115 };
116
117 exports.postEditProduct = (req, res, next) => {
118   const prodId = req.body.productId;
119   const updatedTitle = req.body.title;
120   const updatedPrice = req.body.price;
121   const updatedImageUrl = req.body.imageUrl;
122   const updatedDesc = req.body.description;
123
124   const errors = validationResult(req);
125
126   if (!errors.isEmpty()) {
127     return res.status(422).render('admin/edit-product', {
128       pageTitle: 'Edit Product',
129       path: '/admin/edit-product',
130       editing: true,
131       hasError: true,
132       product: {
133         title: updatedTitle,
134         imageUrl: updatedImageUrl,
135         price: updatedPrice,
136         description: updatedDesc,
137         _id: prodId
138       },
139       errorMessage: errors.array()[0].msg,
140       validationErrors: errors.array()
141     });
142   }

```



```

143
144 Product.findById(prodId)
145   .then(product => {
146     if (product.userId.toString() !== req.user._id.toString()) {
147       return res.redirect('/');
148     }
149     product.title = updatedTitle;
150     product.price = updatedPrice;
151     product.description = updatedDesc;
152     product.imageUrl = updatedImageUrl;
153     return product.save().then(result => {
154       console.log('UPDATED PRODUCT!');
155       res.redirect('/admin/products');
156     });
157   })
158   .catch(err => console.log(err));
159 };
160
161 exports.getProducts = (req, res, next) => {
162   Product.find({ userId: req.user._id })
163     // .select('title price _id')
164     // .populate('userId', 'name')
165     .then(products => {
166       console.log(products);
167       res.render('admin/products', {
168         prods: products,
169         pageTitle: 'Admin Products',
170         path: '/admin/products'
171       });
172     })
173     .catch(err => console.log(err));
174 };
175
176 exports.postDeleteProduct = (req, res, next) => {
177   const prodId = req.body.productId;
178   Product.deleteOne({ _id: prodId, userId: req.user._id })
179     .then(() => {
180       console.log('DESTROYED PRODUCT');
181       res.redirect('/admin/products');
182     })
183     .catch(err => console.log(err));
184 };
185

```

```

1 //app.js
2
3 const path = require('path');
4
5 const express = require('express');
6 const bodyParser = require('body-parser');
7 const mongoose = require('mongoose');
8 const session = require('express-session');
9 const MongoDBStore = require('connect-mongodb-session')(session);
10 const csrf = require('csurf');
11 const flash = require('connect-flash');
12
13 const errorController = require('./controllers/error');

```

```
14 const User = require('./models/user');
15
16 const MONGODB_URI =
17 'mongodb+srv://maximilian:rldnjs12@cluster0-z3v1k.mongodb.net/shop'
18
19 const app = express();
20 const store = new MongoDBStore({
21   uri: MONGODB_URI,
22   collection: 'sessions'
23 });
24 const csrfProtection = csrf();
25
26 app.set('view engine', 'ejs');
27 app.set('views', 'views');
28
29 const adminRoutes = require('./routes/admin');
30 const shopRoutes = require('./routes/shop');
31 const authRoutes = require('./routes/auth');
32
33 app.use(bodyParser.urlencoded({ extended: false }));
34 app.use(express.static(path.join(__dirname, 'public')));
35 app.use(
36   session({
37     secret: 'my secret',
38     resave: false,
39     saveUninitialized: false,
40     store: store
41   })
42 );
43 app.use(csrfProtection);
44 app.use(flash());
45
46 app.use((req, res, next) => {
47   if (!req.session.user) {
48     return next();
49   }
50   User.findById(req.session.user._id)
51     .then(user => {
52       if (!user) {
53         return next();
54       }
55       req.user = user;
56       next();
57     })
58     .catch(err => {
59       throw new Error(err);
60     });
61 });
62
63 app.use((req, res, next) => {
64   res.locals.isAuthenticated = req.session.isLoggedIn;
65   res.locals.csrfToken = req.csrfToken();
66   next();
67 });
68
69 app.use('/admin', adminRoutes);
```

```

70 app.use(shopRoutes);
71 app.use(authRoutes);
72
73 app.get('/500', errorController.get500);
74
75 /**this could be never be reached
76  * because we have our catch all middleware down there
77  *
78  * but there's a special type of middleware which we haven't seen before
79  *
80  */
81 app.use(errorController.get404);
82
83 /**express is clever enough to detect that
84  * this is a special kind of middleware
85  * and it will move right away to these error handling middlewares
86  * when you call 'next(error)'
87  *
88  * if you got more than one error-handling middleware,
89  * they will execute from top to bottom just like the normal middleware
90  */
91 app.use((error, req, res, next) => {
92   /**i just wanna show you
93    * that you can pass extra information with the error object
94    * so that you can use it in this central error handler here
95    *
96    *   res.status(error.httpStatusCode).render(...)
97    *
98    */
99   res.redirect('/500')
100 })
101
102 mongoose
103   .connect(MONGODB_URI)
104   .then(result => {
105     app.listen(3000);
106   })
107   .catch(err => {
108     console.log(err);
109   });
110

```

## \* Chapter 308: Updating The App

1. update
  - ./controllers/admin.js
  - ./controllers/auth.js
  - ./controllers/shop.js












Page Not Found!

4

First Book




\$ 12.99

Does this work?

EditDelete

Third Book!



\$ 22.99

fasfasdfdsfs

EditDelete

Third Book!



\$ 22.99  
fasfasdfdsfs

EditDelete

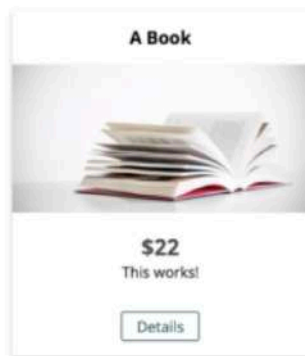
Third Book!



\$ 22.99  
fasfasdfdsfs

EditDelete





JS

```

1 // ./controllers/admin.js
2
3 const mongoose = require('mongoose');
4
5 const { validationResult } = require('express-validator/check');
6
7 const Product = require('../models/product');
8
9 exports.getAddProduct = (req, res, next) => {
10   res.render('admin/edit-product', {
11     pageTitle: 'Add Product',
12     path: '/admin/add-product',
13     editing: false,
14     hasError: false,
15     errorMessage: null,
16     validationErrors: []
17   });
18 };
19
20 exports.postAddProduct = (req, res, next) => {
21   const title = req.body.title;
22   const imageUrl = req.body.imageUrl;
23   const price = req.body.price;
24   const description = req.body.description;
25   const errors = validationResult(req);
26
27   if (!errors.isEmpty()) {
28     console.log(errors.array());
29     return res.status(422).render('admin/edit-product', {
30       pageTitle: 'Add Product',
31       path: '/admin/add-product',
32       editing: false,
33       hasError: true,
34       product: {
35         title: title,

```

```

36     imageUrl: imageUrl,
37     price: price,
38     description: description
39   },
40   errorMessage: errors.array()[0].msg,
41   validationErrors: errors.array()
42 });
43 }
44
45 const product = new Product({
46   //_id: new mongoose.Types.ObjectId('5badf72403fd8b5be0366e81'),
47   title: title,
48   price: price,
49   description: description,
50   imageUrl: imageUrl,
51   userId: req.user
52 });
53 product
54   .save()
55   .then(result => {
56     // console.log(result);
57     console.log('Created Product');
58     res.redirect('/admin/products');
59   })
60   .catch(err => {
61     // return res.status(500).render('admin/edit-product', {
62     //   pageTitle: 'Add Product',
63     //   path: '/admin/add-product',
64     //   editing: false,
65     //   hasError: true,
66     //   product: {
67     //     title: title,
68     //     imageUrl: imageUrl,
69     //     price: price,
70     //     description: description
71     //   },
72     //   errorMessage: 'Database operation failed, please try again.',
73     //   validationErrors: []
74     // });
75     //res.redirect('/500');
76     const error = new Error(err)
77     error.httpStatusCode = 500
78     return next(error)
79   });
80 };
81
82 exports.getEditProduct = (req, res, next) => {
83   const editMode = req.query.edit;
84   if (!editMode) {
85     return res.redirect('/');
86   }
87   const prodId = req.params.productId;
88   Product.findById(prodId)
89     .then(product => {
90       if (!product) {
91         return res.redirect('/');

```

```

92     }
93     res.render('admin/edit-product', {
94       pageTitle: 'Edit Product',
95       path: '/admin/edit-product',
96       editing: editMode,
97       product: product,
98       hasError: false,
99       errorMessage: null,
100      validationErrors: []
101    });
102  })
103  .catch(err => {
104    const error = new Error(err)
105    error.httpStatusCode = 500
106    return next(error)
107  });
108 };
109
110 exports.postEditProduct = (req, res, next) => {
111   const prodId = req.body.productId;
112   const updatedTitle = req.body.title;
113   const updatedPrice = req.body.price;
114   const updatedImageUrl = req.body.imageUrl;
115   const updatedDesc = req.body.description;
116
117   const errors = validationResult(req);
118
119   if (!errors.isEmpty()) {
120     return res.status(422).render('admin/edit-product', {
121       pageTitle: 'Edit Product',
122       path: '/admin/edit-product',
123       editing: true,
124       hasError: true,
125       product: {
126         title: updatedTitle,
127         imageUrl: updatedImageUrl,
128         price: updatedPrice,
129         description: updatedDesc,
130         _id: prodId
131       },
132       errorMessage: errors.array()[0].msg,
133       validationErrors: errors.array()
134     });
135   }
136
137   Product.findById(prodId)
138     .then(product => {
139       if (product.userId.toString() !== req.user._id.toString()) {
140         return res.redirect('/');
141       }
142       product.title = updatedTitle;
143       product.price = updatedPrice;
144       product.description = updatedDesc;
145       product.imageUrl = updatedImageUrl;
146       return product.save().then(result => {
147         console.log('UPDATED PRODUCT!');

```

```

148     res.redirect('/admin/products');
149   });
150 })
151 .catch(err => {
152   const error = new Error(err)
153   error.httpStatusCode = 500
154   return next(error)
155 });
156 };
157
158 exports.getProducts = (req, res, next) => {
159   Product.find({ userId: req.user._id })
160     // .select('title price -_id')
161     // .populate('userId', 'name')
162     .then(products => {
163       console.log(products);
164       res.render('admin/products', {
165         prods: products,
166         pageTitle: 'Admin Products',
167         path: '/admin/products'
168       });
169     })
170     .catch(err => {
171       const error = new Error(err)
172       error.httpStatusCode = 500
173       return next(error)
174     });
175 };
176
177 exports.postDeleteProduct = (req, res, next) => {
178   const prodId = req.body.productId;
179   Product.deleteOne({ _id: prodId, userId: req.user._id })
180     .then(() => {
181       console.log('DESTROYED PRODUCT');
182       res.redirect('/admin/products');
183     })
184     .catch(err => {
185       const error = new Error(err)
186       error.httpStatusCode = 500
187       return next(error)
188     });
189 };
190

```

```

1  //./controllers/auth.js
2
3  const crypto = require('crypto');
4
5  const bcrypt = require('bcryptjs');
6  const nodemailer = require('nodemailer');
7  const sendgridTransport = require('nodemailer-sendgrid-transport');
8  const { validationResult } = require('express-validator/check');
9
10 const User = require('../models/user');
11
12 const transporter = nodemailer.createTransport(
13   sendgridTransport({

```

```

14   auth: {
15     api_key:
16       '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'   }
17   })
18 );
19
20 exports.getLogin = (req, res, next) => {
21   let message = req.flash('error');
22   if (message.length > 0) {
23     message = message[0];
24   } else {
25     message = null;
26   }
27   res.render('auth/login', {
28     path: '/login',
29     pageTitle: 'Login',
30     errorMessage: message,
31     oldInput: {
32       email: '',
33       password: ''
34     },
35     validationErrors: []
36   });
37 };
38
39 exports.getSignup = (req, res, next) => {
40   let message = req.flash('error');
41   if (message.length > 0) {
42     message = message[0];
43   } else {
44     message = null;
45   }
46   res.render('auth/signup', {
47     path: '/signup',
48     pageTitle: 'Signup',
49     errorMessage: message,
50     oldInput: {
51       email: '',
52       password: '',
53       confirmPassword: ''
54     },
55     validationErrors: []
56   });
57 };
58
59 exports.postLogin = (req, res, next) => {
60   const email = req.body.email;
61   const password = req.body.password;
62
63   const errors = validationResult(req);
64   if (!errors.isEmpty()) {
65     return res.status(422).render('auth/login', {
66       path: '/login',
67       pageTitle: 'Login',
68       errorMessage: errors.array()[0].msg,
69       oldInput: {

```



```

70     email: email,
71     password: password
72   },
73   validationErrors: errors.array()
74 });
75 }
76
77 User.findOne({ email: email })
78   .then(user => {
79     if (!user) {
80       return res.status(422).render('auth/login', {
81         path: '/login',
82         pageTitle: 'Login',
83         errorMessage: 'Invalid email or password.',
84         oldInput: {
85           email: email,
86           password: password
87         },
88         validationErrors: []
89       });
90     }
91     bcrypt
92       .compare(password, user.password)
93       .then(doMatch => {
94         if (doMatch) {
95           req.session.isLoggedIn = true;
96           req.session.user = user;
97           return req.session.save(err => {
98             console.log(err);
99             res.redirect('/');
100           });
101         }
102         return res.status(422).render('auth/login', {
103           path: '/login',
104           pageTitle: 'Login',
105           errorMessage: 'Invalid email or password.',
106           oldInput: {
107             email: email,
108             password: password
109           },
110           validationErrors: []
111         });
112       })
113       .catch(err => {
114         console.log(err);
115         res.redirect('/login');
116       });
117   })
118   .catch(err => {
119     const error = new Error(err)
120     error.httpStatusCode = 500
121     return next(error)
122   });
123 };
124
125 exports.postSignup = (req, res, next) => {

```

```

126 const email = req.body.email;
127 const password = req.body.password;
128
129 const errors = validationResult(req);
130 if (!errors.isEmpty()) {
131     console.log(errors.array());
132     return res.status(422).render('auth/signup', {
133         path: '/signup',
134         pageTitle: 'Signup',
135         errorMessage: errors.array()[0].msg,
136         oldInput: {
137             email: email,
138             password: password,
139             confirmPassword: req.body.confirmPassword
140         },
141         validationErrors: errors.array()
142     });
143 }
144
145 bcrypt
146     .hash(password, 12)
147     .then(hashPassword => {
148         const user = new User({
149             email: email,
150             password: hashPassword,
151             cart: { items: [] }
152         });
153         return user.save();
154     })
155     .then(result => {
156         res.redirect('/login');
157         // return transporter.sendMail({
158         //     to: email,
159         //     from: 'shop@node-complete.com',
160         //     subject: 'Signup succeeded!',
161         //     html: '<h1>You successfully signed up!</h1>'
162         // });
163     })
164     .catch(err => {
165         const error = new Error(err)
166         error.httpStatusCode = 500
167         return next(error)
168     });
169 };
170
171 exports.postLogout = (req, res, next) => {
172     req.session.destroy(err => {
173         console.log(err);
174         res.redirect('/');
175     });
176 };
177
178 exports.getReset = (req, res, next) => {
179     let message = req.flash('error');
180     if (message.length > 0) {
181         message = message[0];

```

```

182 } else {
183     message = null;
184 }
185 res.render('auth/reset', {
186     path: '/reset',
187     pageTitle: 'Reset Password',
188     errorMessage: message
189 });
190 };
191
192 exports.postReset = (req, res, next) => {
193     crypto.randomBytes(32, (err, buffer) => {
194         if (err) {
195             console.log(err);
196             return res.redirect('/reset');
197         }
198         const token = buffer.toString('hex');
199         User.findOne({ email: req.body.email })
200             .then(user => {
201                 if (!user) {
202                     req.flash('error', 'No account with that email found. ');
203                     return res.redirect('/reset');
204                 }
205                 user.resetToken = token;
206                 user.resetTokenExpiration = Date.now() + 3600000;
207                 return user.save();
208             })
209             .then(result => {
210                 res.redirect('/');
211                 transporter.sendMail({
212                     to: req.body.email,
213                     from: 'shop@node-complete.com',
214                     subject: 'Password reset',
215                     html: `
216                         <p>You requested a password reset</p>
217                         <p>Click this <a href="http://localhost:3000/reset/${token}">link</a> to set a
218                         new password.</p>
219                     `);
220             })
221             .catch(err => {
222                 const error = new Error(err)
223                 error.httpStatusCode = 500
224                 return next(error)
225             });
226     });
227 };
228
229 exports.getNewPassword = (req, res, next) => {
230     const token = req.params.token;
231     User.findOne({ resetToken: token, resetTokenExpiration: { $gt: Date.now() } })
232         .then(user => {
233             let message = req.flash('error');
234             if (message.length > 0) {
235                 message = message[0];
236             } else {

```

```

237     message = null;
238   }
239   res.render('auth/new-password', {
240     path: '/new-password',
241     pageTitle: 'New Password',
242     errorMessage: message,
243     userId: user._id.toString(),
244     passwordToken: token
245   });
246 })
247 .catch(err => {
248   const error = new Error(err)
249   error.httpStatusCode = 500
250   return next(error)
251 });
252 };
253
254 exports.postNewPassword = (req, res, next) => {
255   const newPassword = req.body.password;
256   const userId = req.body.userId;
257   const passwordToken = req.body.passwordToken;
258   let resetUser;
259
260   User.findOne({
261     resetToken: passwordToken,
262     resetTokenExpiration: { $gt: Date.now() },
263     _id: userId
264   })
265     .then(user => {
266       resetUser = user;
267       return bcrypt.hash(newPassword, 12);
268     })
269     .then(hashPassword => {
270       resetUser.password = hashPassword;
271       resetUser.resetToken = undefined;
272       resetUser.resetTokenExpiration = undefined;
273       return resetUser.save();
274     })
275     .then(result => {
276       res.redirect('/login');
277     })
278     .catch(err => {
279       const error = new Error(err)
280       error.httpStatusCode = 500
281       return next(error)
282     });
283 };
284

```

```

1 //./controllers/shop.js
2
3 const Product = require('../models/product');
4 const Order = require('../models/order');
5
6 exports.getProducts = (req, res, next) => {
7   Product.find()
8     .then(products => {

```

```

9      console.log(products);
10     res.render('shop/product-list', {
11       prods: products,
12       pageTitle: 'All Products',
13       path: '/products'
14     });
15   })
16   .catch(err => {
17     const error = new Error(err)
18     error.httpStatusCode = 500
19     return next(error)
20   });
21 };
22
23 exports.getProduct = (req, res, next) => {
24   const prodId = req.params.productId;
25   Product.findById(prodId)
26     .then(product => {
27       res.render('shop/product-detail', {
28         product: product,
29         pageTitle: product.title,
30         path: '/products'
31       });
32     })
33     .catch(err => {
34       const error = new Error(err)
35       error.httpStatusCode = 500
36       return next(error)
37     });
38 };
39
40 exports.getIndex = (req, res, next) => {
41   Product.find()
42     .then(products => {
43       res.render('shop/index', {
44         prods: products,
45         pageTitle: 'Shop',
46         path: '/'
47       });
48     })
49     .catch(err => {
50       const error = new Error(err)
51       error.httpStatusCode = 500
52       return next(error)
53     });
54 };
55
56 exports.getCart = (req, res, next) => {
57   req.user
58     .populate('cart.items.productId')
59     .execPopulate()
60     .then(user => {
61       const products = user.cart.items;
62       res.render('shop/cart', {
63         path: '/cart',
64         pageTitle: 'Your Cart',

```

```

65     products: products
66   });
67 })
68 .catch(err => {
69   const error = new Error(err)
70   error.httpStatusCode = 500
71   return next(error)
72 });
73 };
74
75 exports.postCart = (req, res, next) => {
76   const prodId = req.body.productId;
77   Product.findById(prodId)
78     .then(product => {
79       return req.user.addToCart(product);
80     })
81     .then(result => {
82       console.log(result);
83       res.redirect('/cart');
84     })
85     .catch(err => {
86       const error = new Error(err)
87       error.httpStatusCode = 500
88       return next(error)
89     })
90 };
91
92 exports.postCartDeleteProduct = (req, res, next) => {
93   const prodId = req.body.productId;
94   req.user
95     .removeFromCart(prodId)
96     .then(result => {
97       res.redirect('/cart');
98     })
99     .catch(err => {
100       const error = new Error(err)
101       error.httpStatusCode = 500
102       return next(error)
103     });
104 };
105
106 exports.postOrder = (req, res, next) => {
107   req.user
108     .populate('cart.items.productId')
109     .execPopulate()
110     .then(user => {
111       const products = user.cart.items.map(i => {
112         return { quantity: i.quantity, product: { ...i.productId._doc } };
113       });
114       const order = new Order({
115         user: {
116           email: req.user.email,
117           userId: req.user
118         },
119         products: products
120       });

```



```

121     return order.save();
122   })
123   .then(result => {
124     return req.user.clearCart();
125   })
126   .then(() => {
127     res.redirect('/orders');
128   })
129   .catch(err => {
130     const error = new Error(err)
131     error.httpStatusCode = 500
132     return next(error)
133   });
134 };
135
136 exports.getOrders = (req, res, next) => {
137   Order.find({ 'user.userId': req.user._id })
138     .then(orders => {
139       res.render('shop/orders', {
140         path: '/orders',
141         pageTitle: 'Your Orders',
142         orders: orders
143       });
144     })
145     .catch(err => {
146       const error = new Error(err)
147       error.httpStatusCode = 500
148       return next(error)
149     });
150 };
151
152

```

## \* Chapter 309: Using The Error Handling Middleware Correctly

1.update  
- app.js







EXPLORER

- NODEJS-COMPLETE-GUIDE
  - controllers
    - admin.js
    - auth.js
    - error.js
    - shop.js
  - data
  - middleware
    - is-auth.js
  - models
  - node\_modules
  - public
  - routes
    - admin.js
    - auth.js
    - shop.js
  - util
  - views
  - .gitignore
  - app.js
  - message.txt
  - package-lock.json
  - package.json
- OUTLINE

```
40 ;
41 app.use(csrfProtection);
42 app.use(flash());
43
44 app.use((req, res, next) => {
45   if (!req.session.user) {
46     return next();
47   }
48   User.findById(req.session.user._id)
49     .then(user => {
50       throw new Error('Dummy');
51       if (!user) {
52         return next();
53       }
54       req.user = user;
55       next();
56     })
57     .catch(err => {
58       throw new Error(err);
59     });
60 });
61
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1: node

```
. To use the new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
[nodemon] restarting due to changes...
[nodemon] starting 'node app.js'
(node:33365) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version
. To use the new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
```

Ln 57, Col 20 (15 selected) Spaces: 2 UTF-8 LF JavaScript Prettier

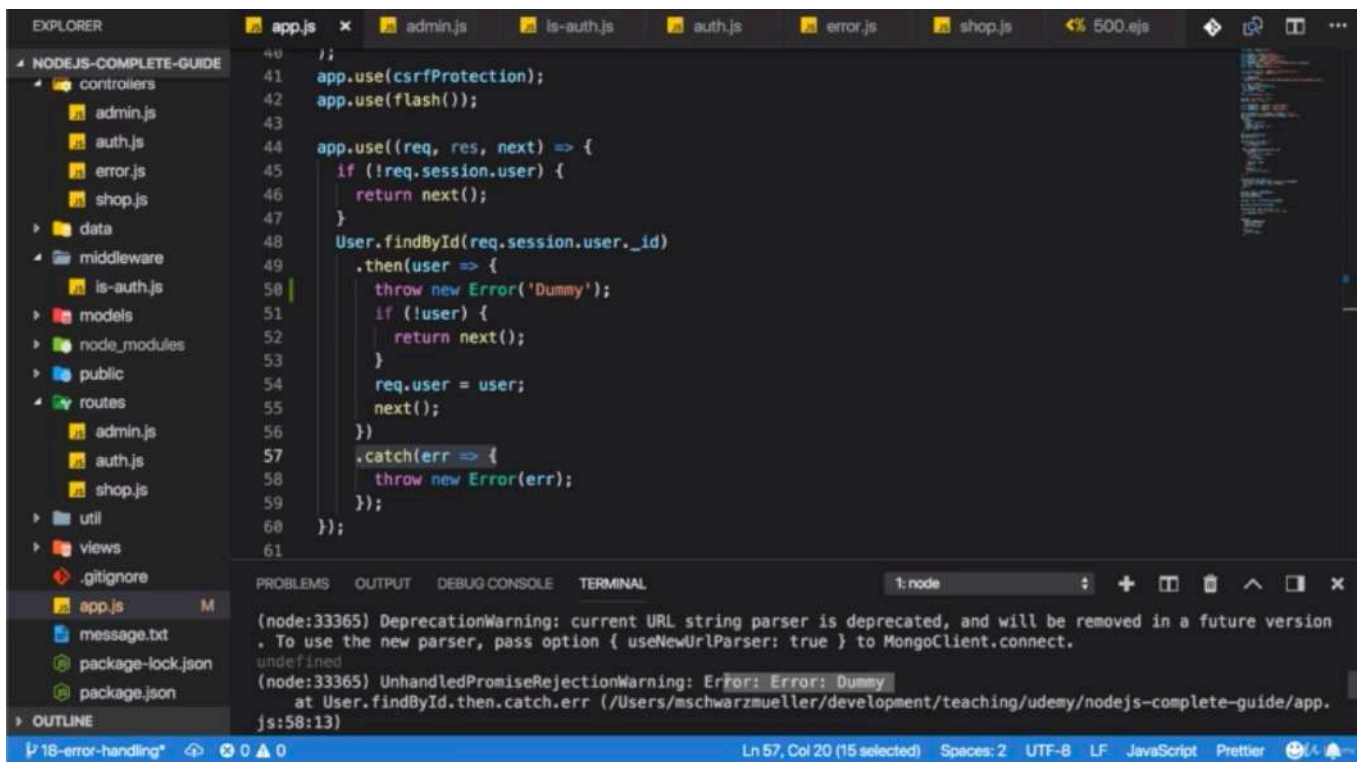
Shop Products Login Signup

E-Mail

Password

Login

[Reset Password](#)



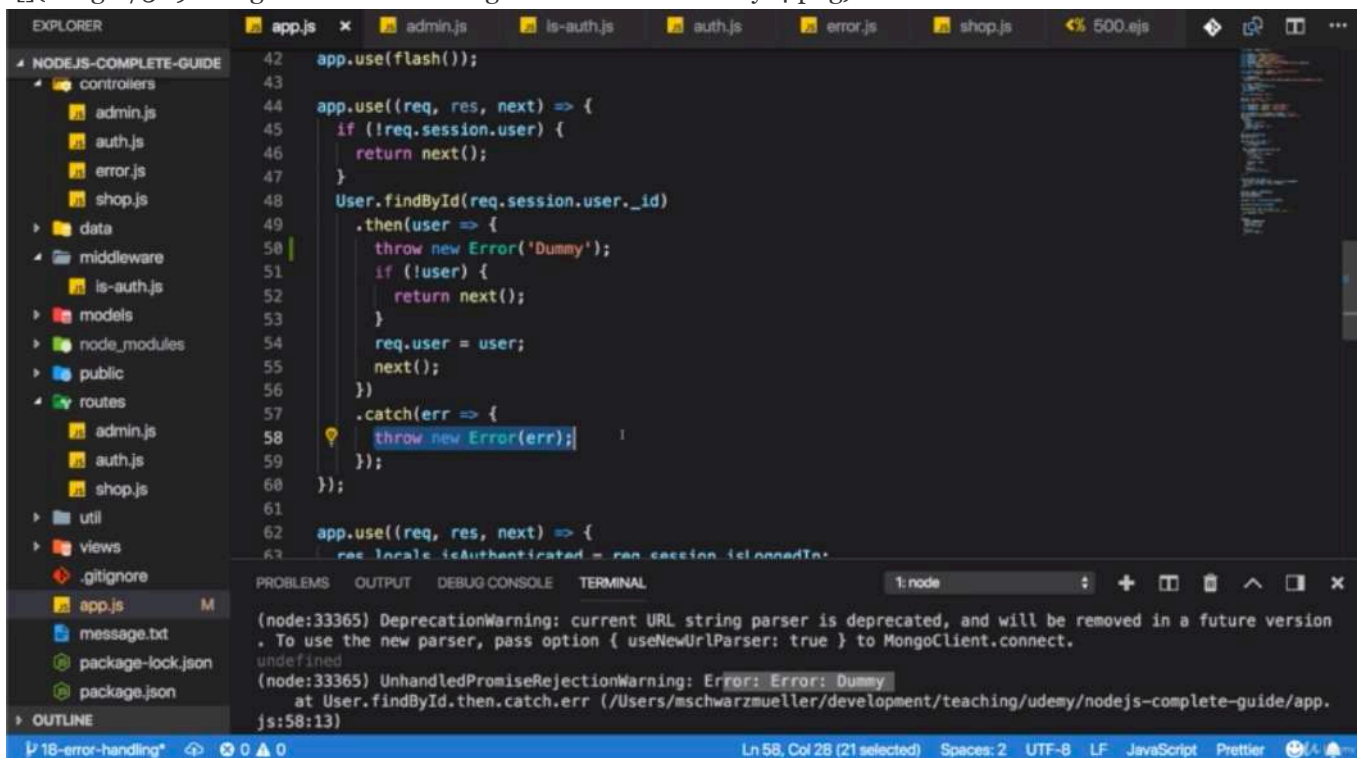
- the app is still crashing. so one important takeaway is throwing an error doesn't lead to our general error handling middleware being called like below

```

1 //app.js
2
3 app.use(error, req, res, next) => {
4   res.redirect('/500')
5 }

```



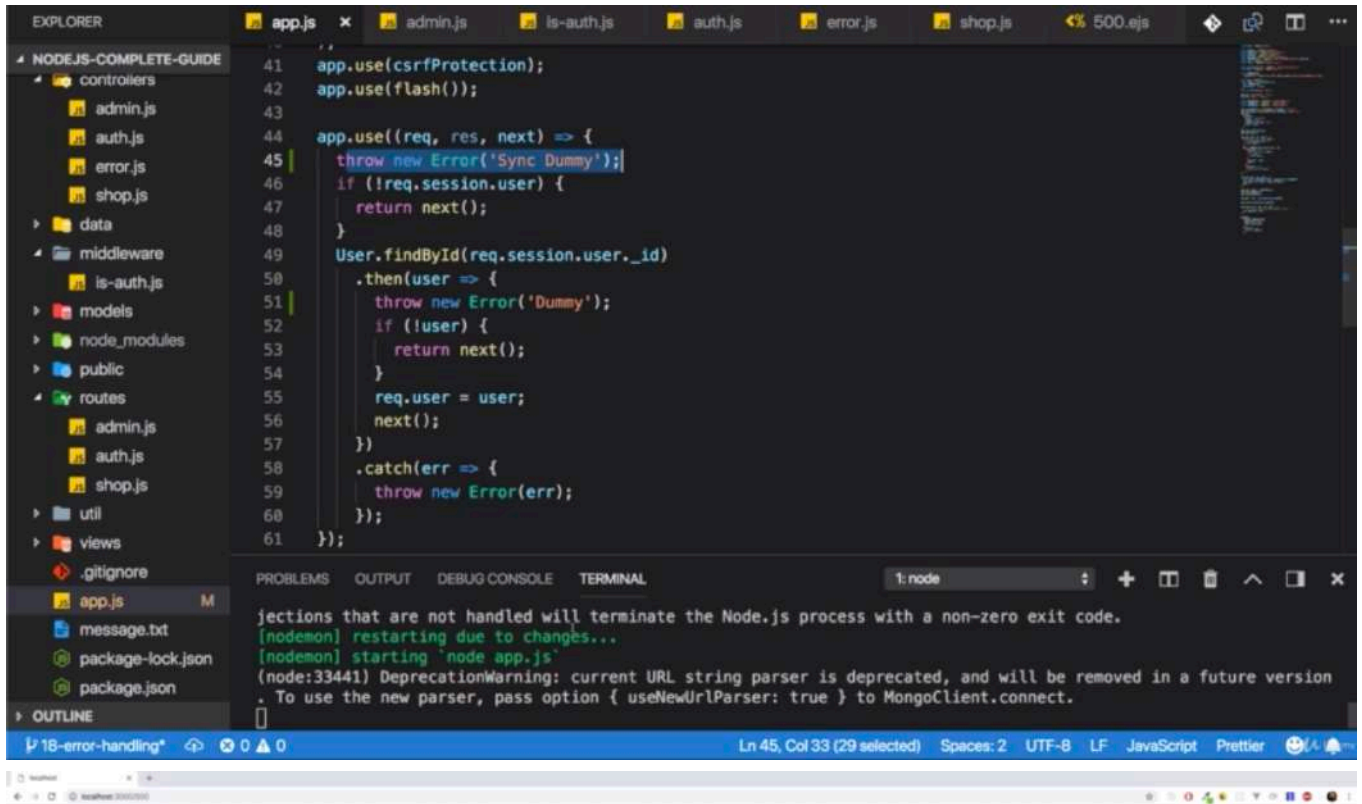


- this is true because we are inside some async code, we are inside a promise here. we are inside a 'then or catch' block.

- if you throw errors there, you will not reach that express error handling middleware







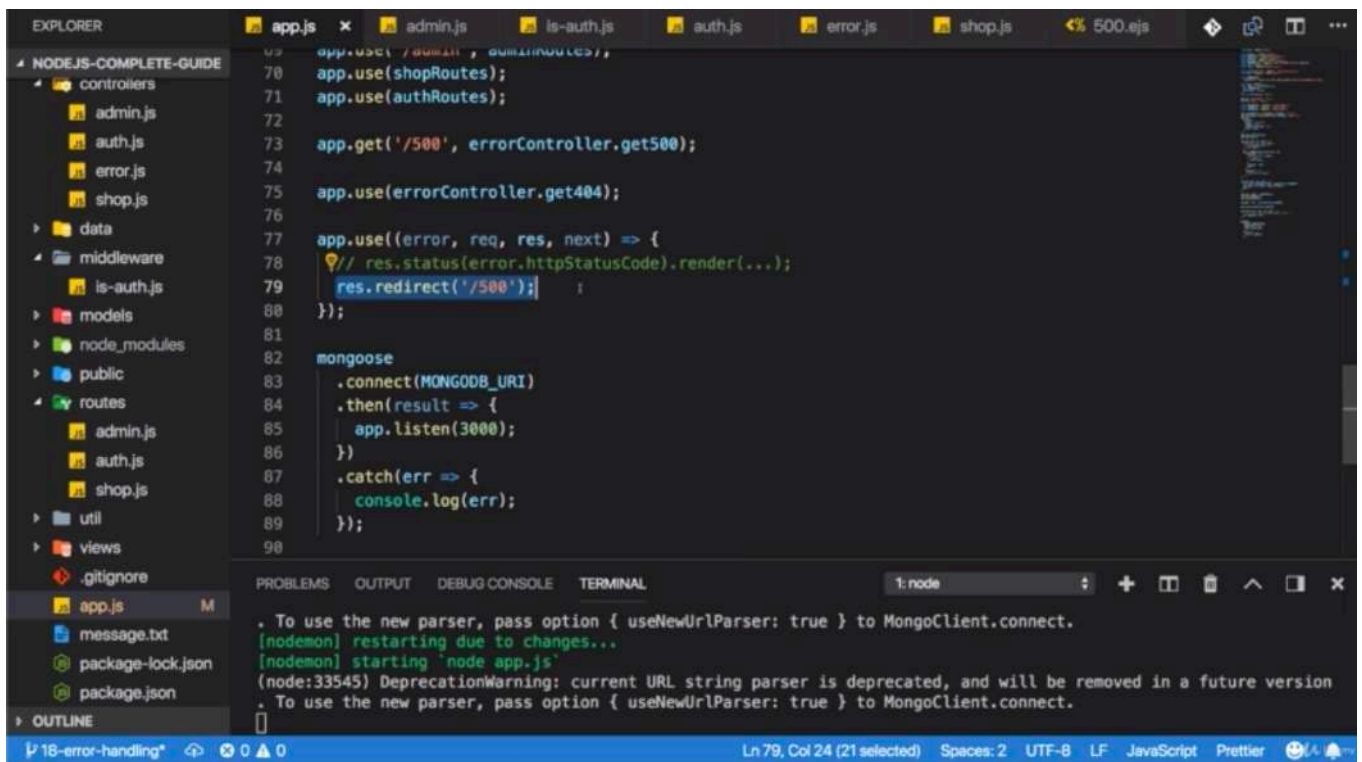
```
41 app.use(csrfProtection);
42 app.use(flash());
43
44 app.use((req, res, next) => {
45   throw new Error('Sync Dummy');
46   if (!req.session.user) {
47     return next();
48   }
49   User.findById(req.session.user._id)
50     .then(user => {
51       throw new Error('Dummy');
52       if (!user) {
53         return next();
54       }
55       req.user = user;
56       next();
57     })
58     .catch(err => {
59       throw new Error(err);
60     });
61 });
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1: node

jections that are not handled will terminate the Node.js process with a non-zero exit code.  
[nodemon] restarting due to changes...  
[nodemon] starting 'node app.js'  
(node:33441) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version.  
To use the new parser, pass option { useNewUrlParser: true } to MongoClient.connect.

- if you would throw an error outside of async code like outside of a promise 'then and catch' block or outside of a callback, this is now throwing in a normal function, not nested inside a promise or a callback.
  - if i now reload this page, you will see it tried to load the 500 page but it still failed. because we have our middleware in place where i retrieve my user and i throw the error, but this executes for every incoming request
- 



- when we redirect here, we send a new request, so we enter a infinite loop here. between 'a' and 'b' below

```

1 //app.js
2
3 //a
4 app.use((req, res, next) => {
5   throw new Error('Sync Dummy')
6   if (!req.session.user) {
7     return next();
8   }
9   User.findById(req.session.user._id)
10    .then(user => {
11      throw new Error('Dummy')
12      if (!user) {
13        return next();
14      }
15      req.user = user;
16      next();
17    })
18    .catch(err => {
19      throw new Error(err);
20    });
21 });
22
23 //b
24 app.use((error, req, res, next) => {
25   res.redirect('/500')
26 })

```

- we execute 'a' and it throws an error. we go to the error handling middleware, we trigger a new request.

- so one solution is not redirecting to 500 but immediately execute our rendering code which is 'get500' in ./controllers/error.js



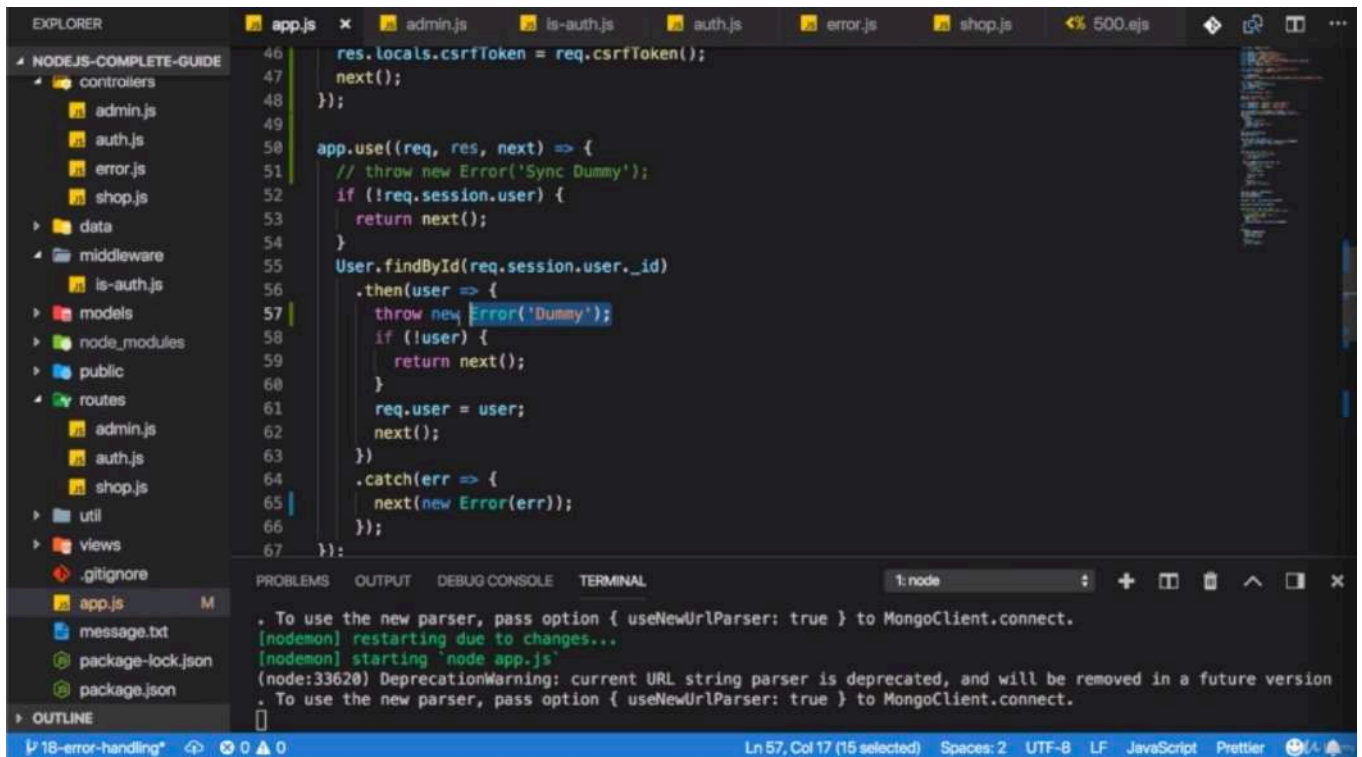






## Some error occurred!

We're working on fixing this, sorry for the inconvenience!



- because i still have this dummy error being thrown in my promise, hence the catch block executes, hence 'next(new Error(err))' executes.
- for one, you should avoid infinite loops triggered through your error handling middleware as we had it.
- for second, you can throw the error in synchronous code places. but inside of promise 'then or catch block or callback', you have to use 'next(new Error(err))'

```
1 //app.js
2
3 const path = require('path');
4
5 const express = require('express');
6 const bodyParser = require('body-parser');
7 const mongoose = require('mongoose');
8 const session = require('express-session');
```

```

9  const MongoDBStore = require('connect-mongodb-session')(session);
10 const csrf = require('csrf');
11 const flash = require('connect-flash');
12
13 const errorController = require('./controllers/error');
14 const User = require('./models/user');
15
16 const MONGODB_URI =
17 'mongodb+srv://maximilian:rldnjs12@cluster0-z3v1k.mongodb.net/shop'
18
19 const app = express();
20 const store = new MongoDBStore({
21   uri: MONGODB_URI,
22   collection: 'sessions'
23 });
24 const csrfProtection = csrf();
25
26 app.set('view engine', 'ejs');
27 app.set('views', 'views');
28
29 const adminRoutes = require('./routes/admin');
30 const shopRoutes = require('./routes/shop');
31 const authRoutes = require('./routes/auth');
32
33 app.use(bodyParser.urlencoded({ extended: false }));
34 app.use(express.static(path.join(__dirname, 'public')));
35 app.use(
36   session({
37     secret: 'my secret',
38     resave: false,
39     saveUninitialized: false,
40     store: store
41   })
42 );
43 app.use(csrfProtection);
44 app.use(flash());
45
46 app.use((req, res, next) => {
47   res.locals.isAuthenticated = req.session.isLoggedIn;
48   res.locals.csrfToken = req.csrfToken();
49   next();
50 });
51
52 app.use((req, res, next) => {
53   /**i'm throwing an error
54    * and we still reach this global error handling middleware below
55    *
56    * app.use((error, req, res, next) => {
57       res.status(500).render('500', {
58         pageTitle: 'Error!',
59         path: '/500',
60         isAuthenticated: req.session.isLoggedIn
61       });
62     })
63    *
64    * the reason is that in synchronous places,

```



```

65  * so outside of callbacks and promises,
66  * you throw an error and express will detect this
67  * and execute your next error handling middleware.
68  * inside of async code, 'then, catch, or callback',
69  * you have to use 'next(new Error(err))'
70  * so this is then detected by express again
71  * and this is what we used in the other files
72  * and inside of async code, you need to use 'next()' wrapping the error
73  * outside you can just 'throw new Error('Sync Dummy')'
74
75
76  */
77  //throw new Error('Sync Dummy')
78  if (!req.session.user) {
79      return next();
80  }
81  User.findById(req.session.user._id)
82      .then(user => {
83          if (!user) {
84              return next();
85          }
86          req.user = user;
87          next();
88      })
89      .catch(err => {
90          next(new Error(err))
91      });
92 });
93
94 app.use('/admin', adminRoutes);
95 app.use(shopRoutes);
96 app.use(authRoutes);
97
98 app.get('/500', errorController.get500);
99
100 app.use(errorController.get404);
101
102 app.use((error, req, res, next) => {
103     //res.redirect('/500')
104     res.status(500).render('500', {
105         pageTitle: 'Error!',
106         path: '/500',
107         isAuthenticated: req.session.isLoggedIn
108     });
109 })
110
111 mongoose
112     .connect(MONGODB_URI)
113     .then(result => {
114         app.listen(3000);
115     })
116     .catch(err => {
117         console.log(err);
118     });
119

```

## \* Chapter 310: Status Codes







### Errors & Http Response Codes

2xx (Success)	200	Operation succeeded
	201	Success, resource created
3xx (Redirect)	301	Moved permanently
4xx (Client-side error)	401	Not authenticated
	403	Not authorized
	404	Not found
	422	Invalid input
5xx (Server-side error)	500	Server-side error

The screenshot shows a VS Code editor with a project named 'NODE.JS-COMPLETE-GUIDE'. The file explorer on the left shows a directory structure with files like `admin.js`, `auth.js`, `error.js`, `shop.js`, `data`, `middleware`, `models`, `node_modules`, `public`, `routes`, `util`, and `views`. The main editor window shows the `admin.js` file with the following code:

```
135 product.findById(productId)
136 .then(product => {
137   if (product.userId.toString() !== req.user._id.toString()) {
138     return res.redirect('/');
139   }
140   product.title = updatedTitle;
141   product.price = updatedPrice;
142   product.description = updatedDesc;
143   product.imageUrl = updatedImageUrl;
144   return product.save().then(result => {
145     console.log('UPDATED PRODUCT!');
146     res.redirect('/admin/products');
147   });
148 })
149 .catch(err => {
150   const error = new Error(err);
151   error.httpStatusCode = 500;
152   return next(error);
153 });
```

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

```
title: 'Third Book!',
price: 22.99,
description: 'fasfasdfdsfs',
imageUrl:
'http://ichef.bbci.co.uk/wwfeatures/wm/live/1280_640/images/live/p0/2v/dp/p02vdpfn.jpg',
userId: 5bb209393fa36b3687f778cd,
_v: 0 } ]
undefined
```

- if we succeed or in all cases here, we redirect which again will use 300 automatically.



```
1 module.exports = (req, res, next) => {
2   if (!req.session.isLoggedIn) {
3     return res.status(401).redirect('/login');
4   }
5   next();
6 }
```

```
{
  title: 'Third Book!',
  price: 22.99,
  description: 'fasfasdfdsfs',
  imageUrl: 'http://ichef.bbci.co.uk/wwfeatures/wm/live/1280_640/images/live/p0/2v/dp/p02vdpfn.jpg',
  userId: 5bb209393fa36b3687f778cd,
  __v: 0 } }
```

- since i'm redirecting, i'm sending a 300 status code and we could also add status 401 here to make clear what the problem is.



```
1 module.exports = (req, res, next) => {
2   if (!req.session.isLoggedIn) {
3     return res.redirect('/login');
4   }
5   next();
6 }
```

```
{
  title: 'Third Book!',
  price: 22.99,
  description: 'fasfasdfdsfs',
  imageUrl: 'http://ichef.bbci.co.uk/wwfeatures/wm/live/1280_640/images/live/p0/2v/dp/p02vdpfn.jpg',
  userId: 5bb209393fa36b3687f778cd,
  __v: 0 } }
```

- but it will be overwritten with a 300 code due to redirect. so it doesn't make a lot of sense.









EXPLORER

- NODEJS-COMPLETE-GUIDE
  - .vscode
  - controllers
    - admin.js
    - auth.js
    - error.js
    - shop.js
  - data
  - middleware
    - is-auth.js
  - models
  - node\_modules
  - public
  - routes
    - admin.js
    - auth.js
    - shop.js
  - util
  - views
    - admin
    - auth
    - includes
    - shop

1 exports.get404 = (req, res, next) => {  
2 res.status(404).render('404', {  
3 pageTitle: 'Page Not Found',  
4 path: '/404',  
5 isAuthenticated: req.session.isLoggedIn  
6 });  
7 };  
8  
9 exports.get500 = (req, res, next) => {  
10 res.status(500).render('500', {  
11 pageTitle: 'Error!',  
12 path: '/500',  
13 isAuthenticated: req.session.isLoggedIn  
14 });  
15 };  
16

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1: node

```
'http://ichef.bbci.co.uk/wwfeatures/wm/live/1280_640/images/live/p0/2v/dp/p02vdpfn.jpg',  
userId: 5bb209393fa36b3687f778cd,  
__v: 0 } ]  
undefined  
[nodemon] restarting due to changes...  
[nodemon] starting 'node app.js'  
(node:28663) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version  
. To use the new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
```

Ln 10, Col 19 (12 selected) Spaces: 2 UTF-8 LF JavaScript Prettier: ✓

18-error-handling 0 0 0

localhost:3000/

Login Signup

A Book



\$22  
This works!

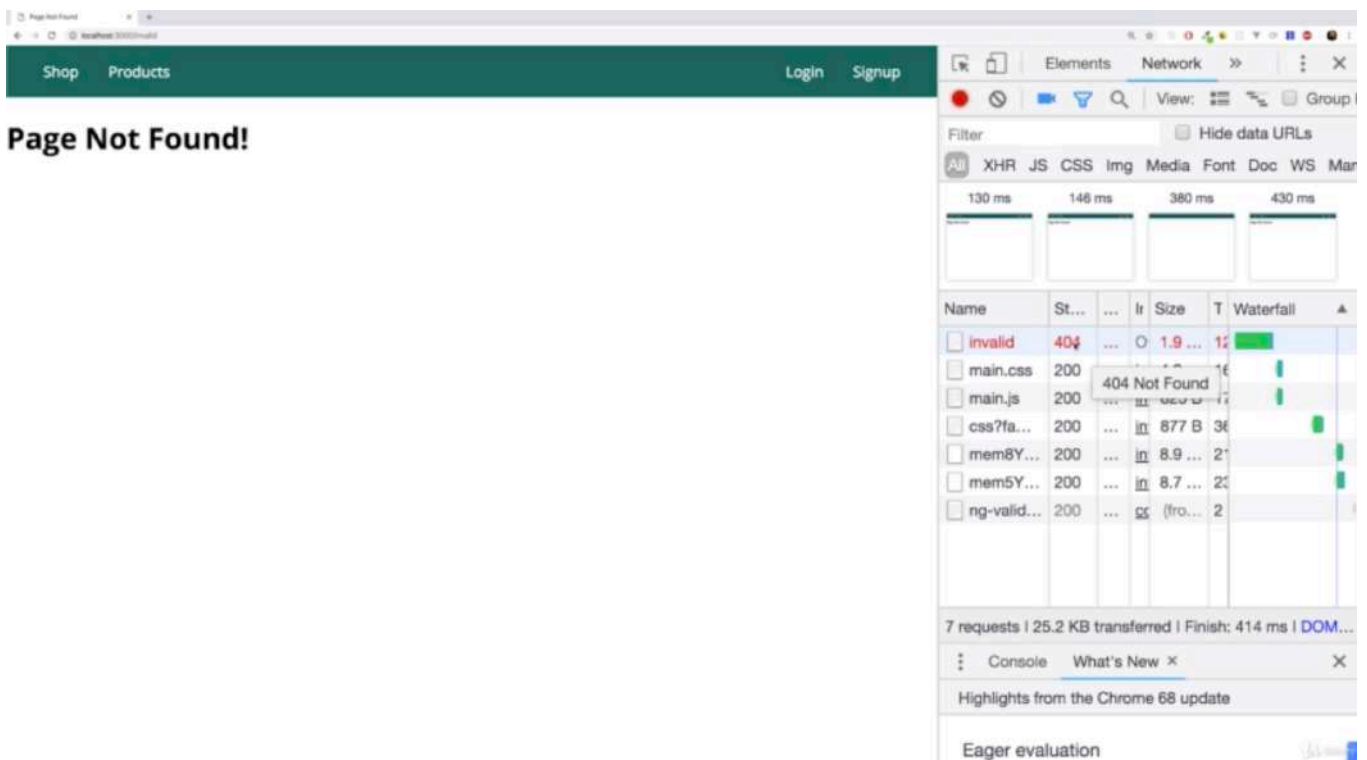
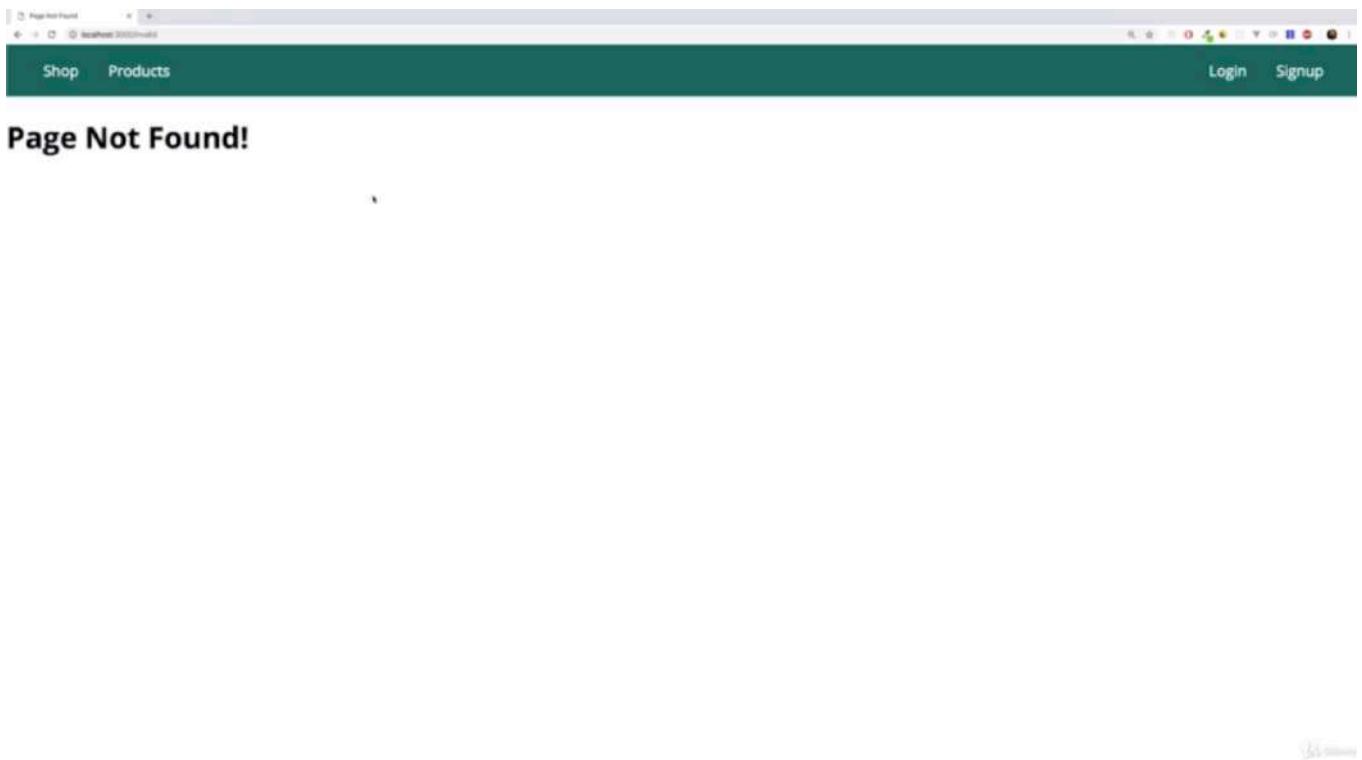
Details

Third Book!



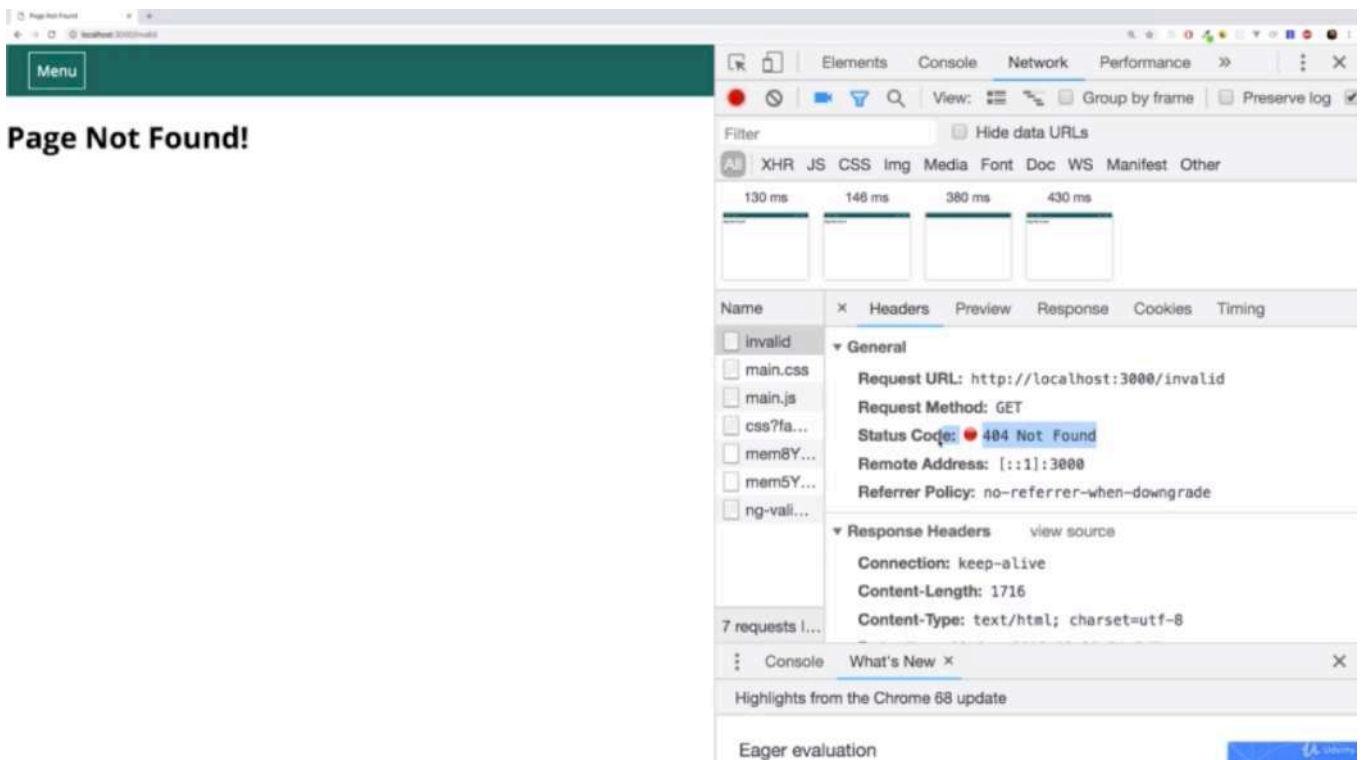
\$22.99  
fasfasdfdsfs

Details



- these status codes not means that our app crashes, that's important to understand. instead if i enter some invalid route here, i get 'page not found', and if i open chrome developer tools, and we ca see this 404 marked as red. but the error doesn't mean it crashes.





- we still render a valid page in the end, we pass that extra information of hey here's the page but you see that page because something goes wrong.

Which status codes are available?

MDN has a nice list: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>

Or, as a short overview:

### 1xx Informational

- 100 Continue
- 101 Switching Protocols
- 102 Processing

### 2xx Success

- 200 OK
- 201 Created
- 202 Accepted
- 203 Non-authoritative Information
- 204 No Content

- 205 Reset Content
- 206 Partial Content
- 207 Multi-Status
- 208 Already Reported
- 226 IM Used

### **3xx Redirection**

- 300 Multiple Choices
- 301 Moved Permanently
- 302 Found
- 303 See Other
- 304 Not Modified
- 305 Use Proxy
- 307 Temporary Redirect
- 308 Permanent Redirect

### **4xx Client Error**

- 400 Bad Request
- 401 Unauthorized
- 402 Payment Required
- 403 Forbidden
- 404 Not Found
- 405 Method Not Allowed
- 406 Not Acceptable



- 407 Proxy Authentication Required
- 408 Request Timeout
- 409 Conflict
- 410 Gone
- 411 Length Required
- 412 Precondition Failed
- 413 Payload Too Large
- 414 Request-URI Too Long
- 415 Unsupported Media Type
- 416 Requested Range Not Satisfiable
- 417 Expectation Failed
- 418 I'm a teapot
- 421 Misdirected Request
- 422 Unprocessable Entity
- 423 Locked
- 424 Failed Dependency
- 426 Upgrade Required
- 428 Precondition Required
- 429 Too Many Requests
- 431 Request Header Fields Too Large
- 444 Connection Closed Without Response
- 451 Unavailable For Legal Reasons

- 499 Client Closed Request

## **5xx Server Error**

- 500 Internal Server Error
- 501 Not Implemented
- 502 Bad Gateway
- 503 Service Unavailable
- 504 Gateway Timeout
- 505 HTTP Version Not Supported
- 506 Variant Also Negotiates
- 507 Insufficient Storage
- 508 Loop Detected
- 510 Not Extended
- 511 Network Authentication Required
- 599 Network Connect Timeout Error

Source: <https://httpstatuses.com/>

## **\* Chapter 312: Wrap Up**



## Module Summary

### Types of Errors & Handling Errors

- You can differentiate between different types of errors – technical errors (which are thrown) and "expected errors" (e.g. invalid user input)
- Error handling can be done with custom if-checks, try-catch, then()-catch() etc
- You can use the Express error handling middleware to handle all unhandled errors

### Errors & Status Code

- When returning responses, it can make sense to also set an appropriate Http status code – this lets the browser know what went wrong
- You got success (2xx), redirect (3xx), client-side errors (4xx) and server-side errors (5xx) codes to choose from
- Setting status codes does NOT mean that the response is incomplete or the app crashed!