

18. Understanding Validation

*** Chapter 284: Module Introduction**



What's In This Module?

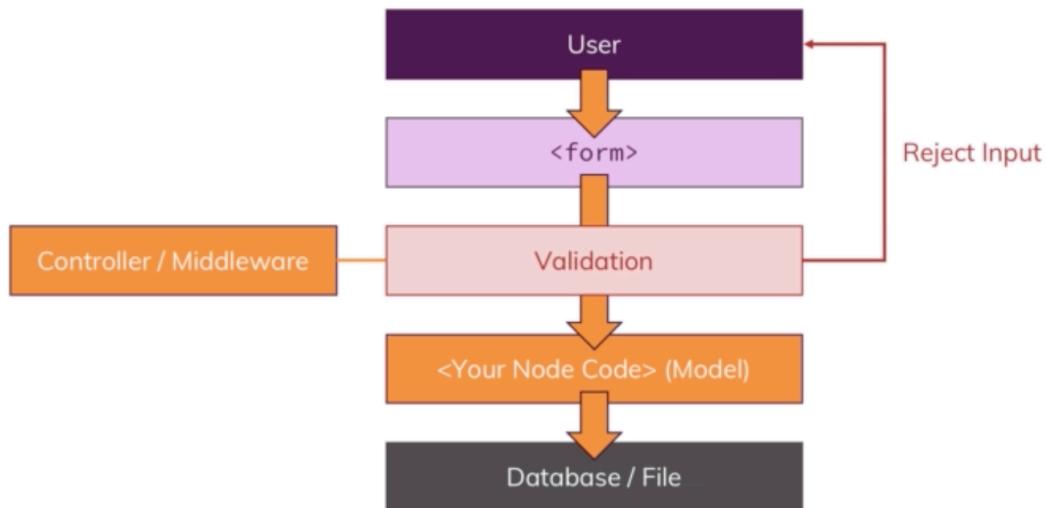
Why Validate?

How to Validate

by Academind

*** Chapter 285: Why Should We Use Validation?**

Why Validate?

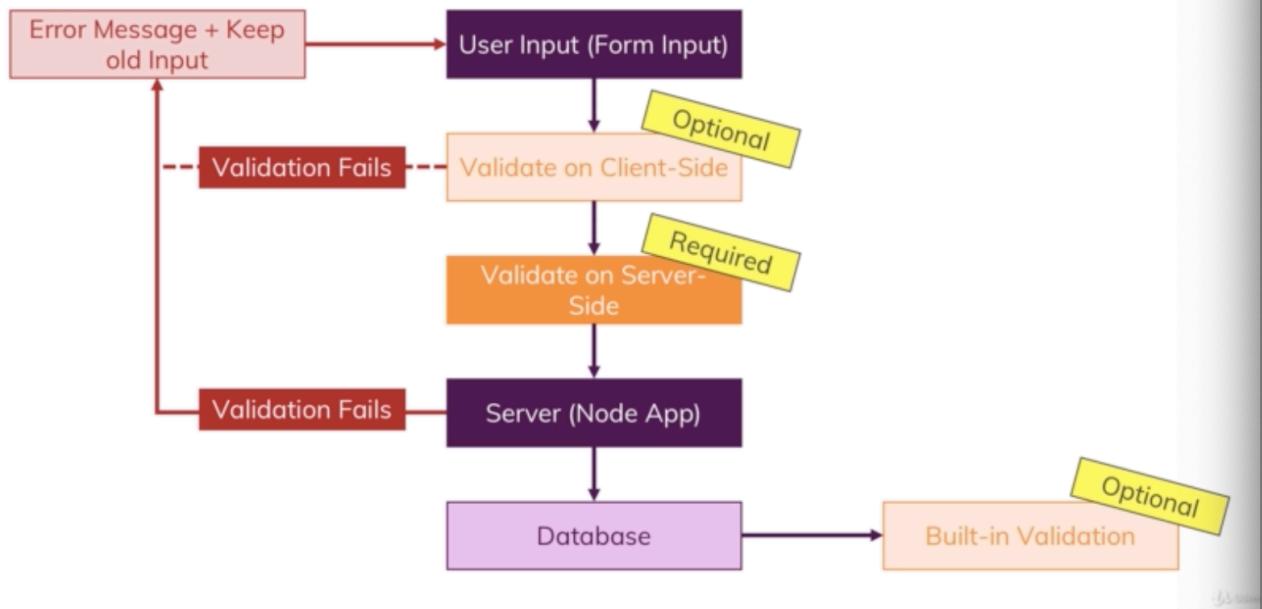


↓ Continue

- in our example project for example, we have a form for signing up, signing in and we got one for adding products.
- when this forms is submitted with a POST request as we controlled it in our form, then a Request is sent to our backend.
- if a user in our current application would try to login with something that is not a valid email address, we would allow that. we are not preventing the user from entering something incorrect.
- the same is true for adding a product, we don't care about what the user enters and this is what i wanna change in this module.
- this validation can then either succeed and allow the data to be written to the database or allow it to be handled by the rest of our node code or we reject the input and then return some information to the user prompting the user to correct the error.

* Chapter 286: How To Validate Input?

How to Validate



- since we use client side javascript, so javascript code that runs in the browser, the user can see that code, the user could change that code and the user can disable javascript. so this is not the protection that secures you against incorrect data being sent to your server. this is not secure solution.
- for some database engines and for most database engines like MongoDB, there is also built-in validation which you can turn on. it's optional because this can be a last resort but if you have a good server side validation in place as you should have. but this might not be required because there is not really a scenario where invalid data could reach your database. because you filter it out in that server side validation already.
- you should validate on the server side at all means but no matter what you choose, in the end, 'Server(Node App)' can also fail and then you should always return an error message-helpful error message and never reload the page but always keep the data the user already inserted because that is a horrible user experience which we know that you enter something incorrect and you get back and fill out from start again.

* Chapter 287: Setup & Basic Validation

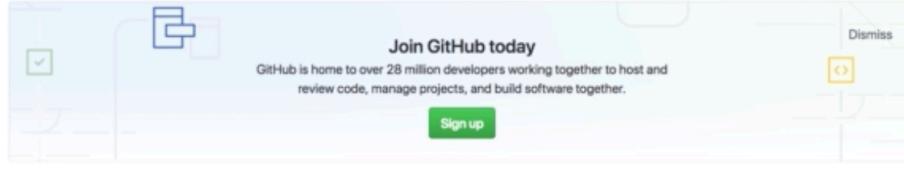
1. update
 - ./routes/auth.js
 - ./controllers/auth.js
 - ./views/auth/signup.ejs

Signed In GitHub - express-validator

Features Business Explore Marketplace Pricing Search Sign in or Sign up

express-validator / express-validator Watch 57 Star 3,314 Fork 373

Code Issues 43 Pull requests 8 Projects 0 Insights



An express.js middleware for validator.js. <https://express-validator.github.io>

nodejs javascript validation express

741 commits	1 branch	85 releases	75 contributors	MIT
Branch: master ▾ New pull request				Find file Clone or download
gustavohenke npm: update some deps				Latest commit 17e49a1 16 days ago
check	Upgrade validator 10.4 (#615)			2 months ago
docs	docs: update "Do not use a common word" example (#633)			a month ago
filter	utils: persist values right after selection			4 months ago
lib	legacy: move around utils that aren't used by other APIs			6 months ago
test	legacy: add base sanitizers to the TS interface			7 months ago

Sign In GitHub - express-validator

Documentation Changelog License

Upgrade notice

If you're arriving here as a express-validator v3 user after upgrading to v4+, please check the [upgrade guide](#) in order to find out what's different!

Also please note that, starting with v5.0.0, no new features will be accepted into the legacy API. Only bug fixes will be made.

Installation

```
npm install express-validator
```

Also make sure that you have Node.js 6 or newer in order to use it.

Documentation

Please refer to the documentation website on <https://express-validator.github.io>.

Changelog

Check the GitHub Releases page.

License

MIT License

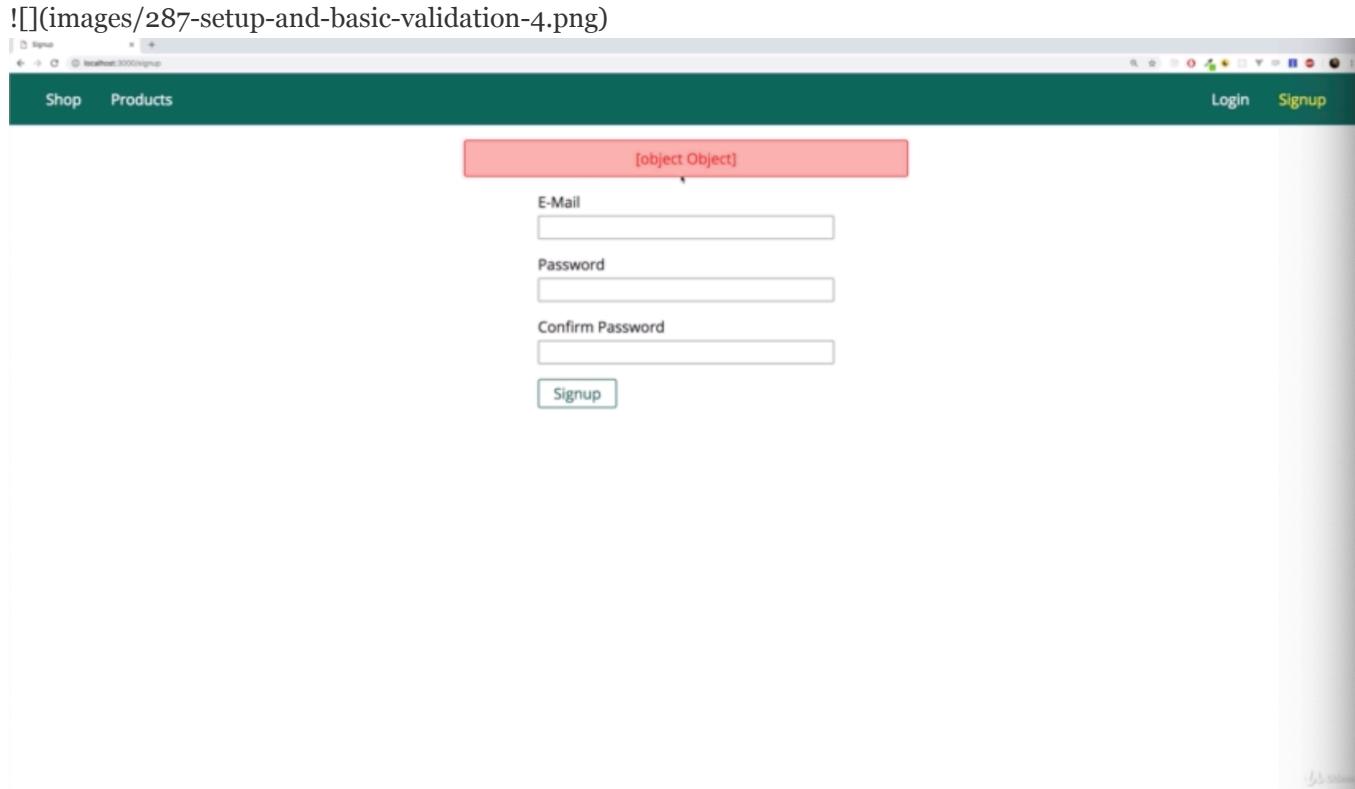


The screenshot shows the VS Code interface with the following details:

- EXPLORER** view: Shows the project structure under "NODEJS-COMPLETE-GUIDE".
 - folders: .vscode, controllers, data, middleware, models, node_modules, public, routes, util, views, admin.
 - files: admin.js, auth.js, error.js, shop.js under controllers; is-auth.js under middleware; order.js, product.js, user.js under models; admin.js, auth.js, shop.js under routes.
- PROBLEMS**, **OUTPUT**, **DEBUG CONSOLE**, **TERMINAL** tabs: The TERMINAL tab is active, showing the command line history:

```
Maximilians-MBP:nodejs-complete-guide mschwarzmueller$  
Maximilians-MBP:nodejs-complete-guide mschwarzmueller$  
Maximilians-MBP:nodejs-complete-guide mschwarzmueller$ npm install --save express-validator
```
- STATUS BAR**: Prettier icon, file count (0), and a small icon.

- the package we will be using is called ‘express-validator’



- '[object Object]' is because we don't get a message, we get an array of errors



```

    76  };
    77
    78 exports.postSignup = (req, res, next) => {
    79   const email = req.body.email;
    80   const password = req.body.password;
    81   const confirmPassword = req.body.confirmPassword;
    82   const errors = validationResult(req);
    83   if (!errors.isEmpty()) {
    84     console.log(errors.array());
    85     return res.status(422).render('auth/signup', {
    86       path: '/signup',
    87       pageTitle: 'Signup',
    88       errorMessage: errors.array()
    89     });
    90   }
    91   User.findOne({ email: email })
    92     .then(userDoc => {
    93       if (userDoc) {
    94         req.flash(
    95           'error',

```

(node:11099) 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.
[{ location: 'body',
 param: 'email',
 value: 'test',
 msg: 'Invalid value' }]

- but if we check our server side console, we see that this console log where i log an array of the errors that was gives me that array

```

1 // ./routes/auth.js
2
3 /**typically you wanna validate on your post or non-get routes
4 * because you wanna validate whenever the user sends data
5 * and that is not the case for our get routes for example.
6 */
7
8 /**you can use a feature called 'destructuring'
9 * where we use curly braces on the left side of the equal sign
10 * then you add some property names
11 * which you wanna pull out of the object that 'express-validator/check' give you
12 */
13 const express = require('express');
14 const { check } = require('express-validator/check')
15
16 const authController = require('../controllers/auth');
17
18 const router = express.Router();
19
20 router.get('/login', authController.getLogin);
21
22 router.get('/signup', authController.getSignup);
23
24 router.post('/login', authController.postLogin);
25
26 /**field name in 'check()'
27 * so this tell this middleware that the express-validator
28 * that i'm interested in confirming that e-mail value and so on.
29 *
30 * 'isEmail()' is a built-in methods in express-validator
31 * which looks for that field in the body, query parameters, headers, cookies
32 * and if finds that field and then checks if that is valid email address.

```

```
33 */
34 router.post('/signup', check('email').isEmail(), authController.postSignup);
35
36 router.post('/logout', authController.postLogout);
37
38 router.get('/reset', authController.getReset);
39
40 router.post('/reset', authController.postReset);
41
42 router.get('/reset/:token', authController.getNewPassword);
43
44 router.post('/new-password', authController.postNewPassword);
45
46 module.exports = router;
```

```
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 /**'validationResult' will be a function that allows us to gather all the errors
9 * prior validation middleware
10 */
11 const { validationResult } = require('express-validator/check')
12
13 const User = require('../models/user');
14
15 const transporter = nodemailer.createTransport(
16   sendgridTransport({
17     auth: {
18       api_key:
19         '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'
20     }
21   })
22 );
23
24 exports.getLogin = (req, res, next) => {
25   let message = req.flash('error');
26   if (message.length > 0) {
27     message = message[0];
28   } else {
29     message = null;
30   }
31   res.render('auth/login', {
32     path: '/login',
33     pageTitle: 'Login',
34     errorMessage: message
35   });
36 };
37
38 exports.getSignup = (req, res, next) => {
39   let message = req.flash('error');
40   if (message.length > 0) {
41     message = message[0];
42   } else {
```

```

43     message = null;
44 }
45 res.render('auth/signup', {
46   path: '/signup',
47   pageTitle: 'Signup',
48   errorMessage: message
49 });
50 };
51
52 exports.postLogin = (req, res, next) => {
53   const email = req.body.email;
54   const password = req.body.password;
55   User.findOne({ email: email })
56     .then(user => {
57       if (!user) {
58         req.flash('error', 'Invalid email or password.');
59         return res.redirect('/login');
60       }
61       bcrypt
62         .compare(password, user.password)
63         .then(doMatch => {
64           if (doMatch) {
65             req.session.isLoggedIn = true;
66             req.session.user = user;
67             return req.session.save(err => {
68               console.log(err);
69               res.redirect('/');
70             });
71           }
72           req.flash('error', 'Invalid email or password.');
73           res.redirect('/login');
74         })
75         .catch(err => {
76           console.log(err);
77           res.redirect('/login');
78         });
79       }
80     .catch(err => console.log(err));
81 };
82
83 exports.postSignup = (req, res, next) => {
84   const email = req.body.email;
85   const password = req.body.password;
86   const confirmPassword = req.body.confirmPassword;
87   /**in the request,
88    * this express-validator middleware
89    * which we added in ./routes/auth.js file will have added errors that can be retrieved
90    * in ./routes/auth.js, collecting errors
91    * and this 'validationResult(req)' will go through that errors obejct
92    * managed by that middleware on the request.
93    * and will then collect them all in this errors constant.
94   */
95   const errors = validationResult(req)
96   if(!errors.isEmpty()){
97     console.log(errors.array())
98     /**422 means a common status code for indicating that validation failed

```

```

99     * it will still send a response just with this different status code
100    */
101   return res.status(422).render('auth/signup', {
102     path: '/signup',
103     pageTitle: 'Signup',
104     errorMessage: errors.array()
105   });
106 }
107 User.findOne({ email: email })
108   .then(userDoc => {
109     if (userDoc) {
110       req.flash(
111         'error',
112         'E-Mail exists already, please pick a different one.'
113       );
114       return res.redirect('/signup');
115     }
116     return bcrypt
117       .hash(password, 12)
118       .then(hashedPassword => {
119         const user = new User({
120           email: email,
121           password: hashedPassword,
122           cart: { items: [] }
123         });
124         return user.save();
125       })
126       .then(result => {
127         res.redirect('/login');
128         return transporter.sendMail({
129           to: email,
130           from: 'shop@node-complete.com',
131           subject: 'Signup succeeded!',
132           html: '<h1>You successfully signed up!</h1>'
133         });
134       })
135       .catch(err => {
136         console.log(err);
137       });
138     })
139     .catch(err => {
140       console.log(err);
141     });
142 };
143
144 exports.postLogout = (req, res, next) => {
145   req.session.destroy(err => {
146     console.log(err);
147     res.redirect('/');
148   });
149 };
150
151 exports.getReset = (req, res, next) => {
152   let message = req.flash('error');
153   if (message.length > 0) {
154     message = message[0];

```

```

155 } else {
156   message = null;
157 }
158 res.render('auth/reset', {
159   path: '/reset',
160   pageTitle: 'Reset Password',
161   errorMessage: message
162 });
163 };
164
165 exports.postReset = (req, res, next) => {
166   crypto.randomBytes(32, (err, buffer) => {
167     if (err) {
168       console.log(err);
169       return res.redirect('/reset');
170     }
171     const token = buffer.toString('hex');
172     User.findOne({ email: req.body.email })
173       .then(user => {
174         if (!user) {
175           req.flash('error', 'No account with that email found.');
176           return res.redirect('/reset');
177         }
178         user.resetToken = token;
179         user.resetTokenExpiration = Date.now() + 3600000;
180         return user.save();
181       })
182       .then(result => {
183         res.redirect('/');
184         transporter.sendMail({
185           to: req.body.email,
186           from: 'shop@node-complete.com',
187           subject: 'Password reset',
188           html: `
189             <p>You requested a password reset</p>
190             <p>Click this <a href="http://localhost:3000/reset/${token}">link</a> to set a
new password.</p>
191             `
192         });
193       })
194       .catch(err => {
195         console.log(err);
196       });
197   });
198 };
199
200 exports.getNewPassword = (req, res, next) => {
201   const token = req.params.token;
202   User.findOne({
203     resetToken: token,
204     resetTokenExpiration: { $gt: Date.now() }
205   })
206     .then(user => {
207       let message = req.flash('error');
208       if (message.length > 0) {
209         message = message[0];

```

```

210     } else {
211         message = null;
212     }
213     res.render('auth/new-password', {
214         path: '/new-password',
215         pageTitle: 'New Password',
216         errorMessage: message,
217         userId: user._id.toString(),
218         passwordToken: token
219     });
220 }
221 .catch(err => {
222     console.log(err);
223 });
224 };
225
226 exports.postNewPassword = (req, res, next) => {
227     /**
228      * i still wanna have that token
229      * because otherwise people could start entering random tokens
230      * and still reach that page and then maybe change users on the back
231      * and by entering random userIds and that hidden input field as well.
232      * so i wanna have that token.
233     */
234     const newPassword = req.body.password
235     const userId = req.body.userId
236     const passwordToken = req.body.passwordToken
237     let resetUser
238
239     User.findOne({
240         resetToken: passwordToken,
241         resetTokenExpiration: {$gt: Date.now()},
242         _id: userId
243     })
244         .then(user => {
245             resetUser = user
246             return bcrypt.hash(newPassword, 12)
247         })
248         .then(hashedPassword => {
249             resetUser.resetToken = undefined
250             resetUser.resetTokenExpiration = undefined
251             return resetUser.save()
252         })
253         .then(result => {
254             res.redirect('/login')
255         })
256         .catch(err => {
257             console.log(err)
258         })
259 }
260

```

```

1 <!--./views/auth/signup.ejs-->
2
3 <%-- include('../includes/head.ejs') %>
4     <link rel="stylesheet" href="/css/forms.css">
5     <link rel="stylesheet" href="/css/auth.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    <!--you can add 'noValidate' to the overall form,
16    validate to disable this check.-->
17    <form class="login-form" action="/signup" method="POST" novalidate>
18      <div class="form-control">
19        <label for="email">E-Mail</label>
20        <input type="email" name="email" id="email">
21      </div>
22      <div class="form-control">
23        <label for="password">Password</label>
24        <input type="password" name="password" id="password">
25      </div>
26      <div class="form-control">
27        <label for="confirmPassword">Confirm Password</label>
28        <input type="password" name="confirmPassword" id="confirmPassword">
29      </div>
30      <input type="hidden" name="_csrf" value="<%= csrfToken %>">
31      <button class="btn" type="submit">Signup</button>
32    </form>
33  </main>
34 <%- include('../includes/end.ejs') %>

```

* Chapter 288: Using Validation Error Messages

1. update
 - ./controllers/auth.js
 - ./routes/auth.js

Signup

localhost:3000/signup

Shop Products Login Signup

[object Object]

E-Mail

Password

Confirm Password

[Signup](#)

Signup

localhost:3000/signup

Shop Products Login Signup

Invalid value

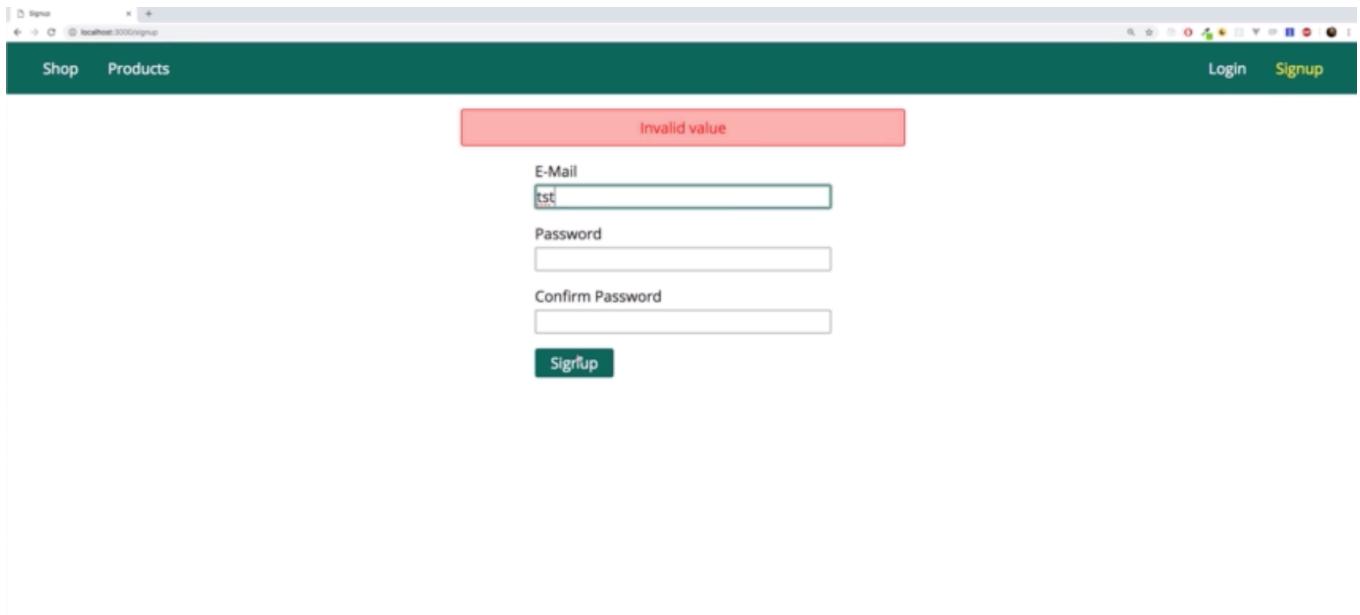
E-Mail

Password

Confirm Password

[Signup](#)

✓ J. Wiley



```
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  
21 exports.getLogin = (req, res, next) => {  
22   let message = req.flash('error');  
23   if (message.length > 0) {  
24     message = message[0];  
25   } else {  
26     message = null;  
27   }  
28   res.render('auth/login', {  
29     path: '/login',  
30     pageTitle: 'Login',  
31     errorMessage: message  
32   });  
33 };  
34  
35 exports.getSignup = (req, res, next) => {  
36   let message = req.flash('error');  
37   if (message.length > 0) {  
38     message = message[0];  
39   } else {  
40     message = null;  
41   }  
42   res.render('auth/signup', {  
43     path: '/signup',  
44     pageTitle: 'Signup',  
45     errorMessage: message  
46   });  
47 };  
48  
49 exports.postLogin = (req, res, next) => {  
50   const email = req.body.email;  
51   const password = req.body.password;  
52   User.findOne({ email: email })  
53     .then(user => {  
54       if (!user) {  
55         req.flash('error', 'Invalid email or password.');//  
56         return res.redirect('/login');//  
57       }  
58       bcrypt  
59         .compare(password, user.password)  
60         .then(doMatch => {  
61           if (doMatch) {  
62             req.session.isLoggedIn = true;  
63             req.session.user = user;  
64             return req.session.save(err => {  
65               if (err) {  
66                 console.log(err);  
67                 res.redirect('/');//  
68               }  
69             req.flash('error', 'Invalid email or password.');//  
70             res.redirect('/login');//  
71           }  
72         }  
73       }  
74     }  
75   }  
76 };
```

```
71      })
72      .catch(err => {
73        console.log(err);
74        res.redirect('/login');
75      });
76    })
77    .catch(err => console.log(err));
78  };
79
80 exports.postSignup = (req, res, next) => {
81   const email = req.body.email;
82   const password = req.body.password;
83   const confirmPassword = req.body.confirmPassword;
84   const errors = validationResult(req)
85   if(!errors.isEmpty()){
86     console.log(errors.array())
87     return res.status(422).render('auth/signup', {
88       path: '/signup',
89       pageTitle: 'Signup',
90       errorMessage: errors.array()[0].msg
91     });
92   }
93   User.findOne({ email: email })
94   .then(userDoc => {
95     if (userDoc) {
96       req.flash(
97         'error',
98         'E-Mail exists already, please pick a different one.'
99     );
100    return res.redirect('/signup');
101  }
102  return bcrypt
103    .hash(password, 12)
104    .then(hashedPassword => {
105      const user = new User({
106        email: email,
107        password: hashedPassword,
108        cart: { items: [] }
109      });
110      return user.save();
111    })
112    .then(result => {
113      res.redirect('/login');
114      return transporter.sendMail({
115        to: email,
116        from: 'shop@node-complete.com',
117        subject: 'Signup succeeded!',
118        html: '<h1>You successfully signed up!</h1>'
119      });
120    })
121    .catch(err => {
122      console.log(err);
123    });
124  })
125  .catch(err => {
126    console.log(err);
```

```

127     });
128 };
129
130 exports.postLogout = (req, res, next) => {
131   req.session.destroy(err => {
132     console.log(err);
133     res.redirect('/');
134   });
135 };
136
137 exports.getReset = (req, res, next) => {
138   let message = req.flash('error');
139   if (message.length > 0) {
140     message = message[0];
141   } else {
142     message = null;
143   }
144   res.render('auth/reset', {
145     path: '/reset',
146     pageTitle: 'Reset Password',
147     errorMessage: message
148   });
149 };
150
151 exports.postReset = (req, res, next) => {
152   crypto.randomBytes(32, (err, buffer) => {
153     if (err) {
154       console.log(err);
155       return res.redirect('/reset');
156     }
157     const token = buffer.toString('hex');
158     User.findOne({ email: req.body.email })
159       .then(user => {
160         if (!user) {
161           req.flash('error', 'No account with that email found.');
162           return res.redirect('/reset');
163         }
164         user.resetToken = token;
165         user.resetTokenExpiration = Date.now() + 3600000;
166         return user.save();
167       })
168       .then(result => {
169         res.redirect('/');
170         transporter.sendMail({
171           to: req.body.email,
172           from: 'shop@node-complete.com',
173           subject: 'Password reset',
174           html: `
175             <p>You requested a password reset</p>
176             <p>Click this <a href="http://localhost:3000/reset/${token}">link</a> to set a
new password.</p>
177             `
178         });
179       })
180       .catch(err => {
181         console.log(err);

```

```
182     });
183   });
184 };
185
186 exports.getNewPassword = (req, res, next) => {
187   const token = req.params.token;
188   User.findOne({
189     resetToken: token,
190     resetTokenExpiration: { $gt: Date.now() }
191   })
192     .then(user => {
193       let message = req.flash('error');
194       if (message.length > 0) {
195         message = message[0];
196       } else {
197         message = null;
198       }
199       res.render('auth/new-password', {
200         path: '/new-password',
201         pageTitle: 'New Password',
202         errorMessage: message,
203         userId: user._id.toString(),
204         passwordToken: token
205       });
206     })
207     .catch(err => {
208       console.log(err);
209     });
210 };
211
212 exports.postNewPassword = (req, res, next) => {
213   const newPassword = req.body.password
214   const userId = req.body.userId
215   const passwordToken = req.body.passwordToken
216   let resetUser
217
218   User.findOne({
219     resetToken: passwordToken,
220     resetTokenExpiration: {$gt: Date.now()},
221     _id: userId
222   })
223     .then(user => {
224       resetUser = user
225       return bcrypt.hash(newPassword, 12)
226     })
227     .then(hashedPassword => {
228       resetUser.resetToken = undefined
229       resetUser.resetTokenExpiration = undefined
230       return resetUser.save()
231     })
232     .then(result => {
233       res.redirect('/login')
234     })
235     .catch(err => {
236       console.log(err)
237     })

```

```

238
239 }
240
1 // ./routes/auth.js
2
3 const express = require('express');
4 const { check } = require('express-validator/check')
5
6 const authController = require('../controllers/auth');
7
8 const router = express.Router();
9
10 router.get('/login', authController.getLogin);
11
12 router.get('/signup', authController.getSignup);
13
14 router.post('/login', authController.postLogin);
15 /**you could add multiple ones
16 * but with that message,
17 * we will always refer to the validation method right in front of it.
18 */
19 router.post('/signup', check('email').isEmail().withMessage('Please enter a valid email'),
  authController.postSignup);
20
21 router.post('/logout', authController.postLogout);
22
23 router.get('/reset', authController.getReset);
24
25 router.post('/reset', authController.postReset);
26
27 router.get('/reset/:token', authController.getNewPassword);
28
29 router.post('/new-password', authController.postNewPassword);
30
31 module.exports = router;

```

* Chapter 289: Built-in & Custom Validators

1. update
 - ./routes/auth.js

Signed in

Getting Started - express-validator

[https://express-validator.github.io/docs/](#)

express-validator 5.3.0

Docs API GitHub

- Introduction
 - [Getting Started](#)
- Features
 - Sanitization
 - Custom validators/sanitizers
 - Custom Error Messages
 - Wildcards
 - Schema Validation
 - Whole Body Validation
- API
 - check API
 - filter API
 - Sanitization Chain API
 - Validation Chain API
 - Validation Result API
 - Legacy API

Getting Started

[EDIT](#)

express-validator is a set of [express.js](#) middlewares that wraps [validator.js](#) validator and sanitizer functions.

Installation

Install it using npm (make sure that you have Node.js 6 or newer):

```
npm install --save express-validator
```

Basic guide

It's recommended that you have basic knowledge of the express.js module before you go on with this guide.

Let's get started by writing a basic route to create a user in the database:

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

app.use(express.json());
app.post('/user', (req, res) => {
  User.create({
    username: req.body.username,
    password: req.body.password
  });
  res.send("User created!");
});
```

- ‘validator.js’ is another package that was implicitly installed with express-validator.

Signed in

GitHub - chriso/validator.js

Features Business Explore Marketplace Pricing Search Sign in or Sign up

chriso / validator.js

Code Issues 29 Pull requests 2 Projects 0 Wiki Insights

Join GitHub today

GitHub is home to over 28 million developers working together to host and review code, manage projects, and build software together.

Dismiss

Sign up

String validation

validation sanitization javascript node input validator validations validate sanitize

1,678 commits 1 branch 131 releases 191 contributors MIT

Branch: master New pull request Find file Clone or download

nachoalvarez and ProfNandas feat(isMobile): add es-UY locale support (#899) 3 days ago

lib feat(isMobile): add es-UY locale support (#899) 3 days ago

src feat(isMobile): add es-UY locale support (#899) 3 days ago

test feat(isMobile): add es-UY locale support (#899) 3 days ago

.eslintignore Lint tests 3 years ago

.eslintrc.json Remove src lint overrides 5 months ago

Waiting for cache.githubusercontent.com...

<code>isDecimal(str [, options])</code>	locate determine the decimal separator and is one of ['ar', 'ar-AE', 'ar-BH', 'ar-DZ', 'ar-EG', 'ar-IQ', 'ar-JO', 'ar-KW', 'ar-LB', 'ar-LY', 'ar-MA', 'ar-QA', 'ar-QM', 'ar-SA', 'ar-SD', 'ar-SY', 'ar-TN', 'ar-YE', 'bg-BG', 'cs-CZ', 'da-DK', 'de-DE', 'en-AU', 'en-GB', 'en-IN', 'en-HK', 'en-NZ', 'en-US', 'en-ZA', 'en-ZM', 'es-ES', 'fr-FR', 'hu-HU', 'it-IT', 'ku-IQ', 'nb-NO', 'nl-NL', 'nn-NO', 'pl-PL', 'pt-BR', 'pt-PT', 'ru-RU', 'sl-SI', 'sr-RS@latin', 'sv-SE', 'tr-TR', 'uk-UA'] . Note: decimal_digits is given as a range like '1,3', a specific value like '3' or min like '1'.
<code>isDivisibleBy(str, number)</code>	check if the string is a number that's divisible by another.
<code>isEmail(str [, options])</code>	check if the string is an email. options is an object which defaults to { allow_display_name: false, require_display_name: false, allow_utf8_local_part: true, require_tld: true, allow_ip_domain: false, domain_specific_validation: false }. If allow_display_name is set to true, the validator will also match Display Name <email-address>. If require_display_name is set to true, the validator will reject strings without the format Display Name <email-address>. If allow_utf8_local_part is set to false, the validator will not allow any non-English UTF8 character in email address' local part. If require_tld is set to false, e-mail addresses without having TLD in their domain will also be matched. If allow_ip_domain is set to true, the validator will allow IP addresses in the host part. If domain_specific_validation is true, some additional validation will be enabled, e.g. disallowing certain syntactically valid email addresses that are rejected by GMail.
<code>isEmpty(str [, options])</code>	check if the string has a length of zero. options is an object which defaults to { ignore_whitespace:false }.
<code>isFQDN(str [, options])</code>	check if the string is a fully qualified domain name (e.g. domain.com). options is an object which defaults to { require_tld: true, allow_underscores: false, allow_trailing_dot: false }.

<code>isEmail(str [, options])</code>	check if the string is an email. options is an object which defaults to { allow_display_name: false, require_display_name: false, allow_utf8_local_part: true, require_tld: true, allow_ip_domain: false, domain_specific_validation: false }. If allow_display_name is set to true, the validator will also match Display Name <email-address>. If require_display_name is set to true, the validator will reject strings without the format Display Name <email-address>. If allow_utf8_local_part is set to false, the validator will not allow any non-English UTF8 character in email address' local part. If require_tld is set to false, e-mail addresses without having TLD in their domain will also be matched. If allow_ip_domain is set to true, the validator will allow IP addresses in the host part. If domain_specific_validation is true, some additional validation will be enabled, e.g. disallowing certain syntactically valid email addresses that are rejected by GMail.
<code>isEmpty(str [, options])</code>	check if the string has a length of zero. options is an object which defaults to { ignore_whitespace:false }.
<code>isFQDN(str [, options])</code>	check if the string is a fully qualified domain name (e.g. domain.com). options is an object which defaults to { require_tld: true, allow_underscores: false, allow_trailing_dot: false }.

- there's many thing and you can create your own validator

Signup

localhost:3000/signup

Shop Products Login Signup

Please enter a valid email.

E-Mail
test@test.com

Password

Confirm Password

Signup

Signup

localhost:3000/signup

Shop Products Login Signup

This email address is forbidden.

E-Mail

Password

Confirm Password

Signup

John Doe

This email address is forbidden.

E-Mail
test2@test.com

Password

Confirm Password

Signup

E-Mail

Password

Confirm Password

Signup

```
1 // ./routes/auth.js
2
3 const express = require('express');
4 const { check } = require('express-validator/check')
5
6 const authController = require('../controllers/auth');
7
8 const router = express.Router();
9
10 router.get('/login', authController.getLogin);
11
12 router.get('/signup', authController.getSignup);
13
14 router.post('/login', authController.postLogin);
```

```

15
16 router.post(
17   '/signup',
18   check('email')
19     .isEmail()
20     .withMessage('Please enter a valid email')
21     .custom((value, {req}) => {
22       if (value === 'test@test.com'){
23         throw new Error('This Email Address is forbidden.')
24       }
25       return true
26     }),
27   authController.postSignup);
28
29 router.post('/logout', authController.postLogout);
30
31 router.get('/reset', authController.getReset);
32
33 router.post('/reset', authController.postReset);
34
35 router.get('/reset/:token', authController.getNewPassword);
36
37 router.post('/new-password', authController.postNewPassword);
38
39 module.exports = router;

```

* Chapter 290: More Validators

1. update
- ./routes/auth.js

Shop Products Login Signup

This email address is forbidden.

E-Mail

Password

Confirm Password

Signup

Shop Products Login Signup

Please enter a password with only numbers and text and at least 5 characters.

E-Mail

Password

Confirm Password

Signup

Join today

Signup

localhost:3000/signup

Shop Products Login Signup

Please enter a password with only numbers and text and at least 5 characters.

E-Mail
test33@test.com

Password

Confirm Password

[Signup](#)

Login

localhost:3000/login

Shop Products Login Signup

E-Mail exists already, please pick a different one.

E-Mail

Password

[Login](#) [Reset Password](#)

Join today

Signup

localhost:3000/signup

Shop Products Login Signup

E-Mail

Password

Confirm Password

[Signup](#)

Login

localhost:3000/login

Shop Products Login Signup

E-Mail

Password

[Login](#)

[Reset Password](#)

E-Mail
test22222@test.com

Password

Confirm Password

Signup

Please enter a password with only numbers and text and at least 5 characters.

E-Mail

Password

Confirm Password

Signup

```
1 // ./routes/auth.js
2
3 const express = require('express');
4 const { check, body } = require('express-validator/check')
5
6 const authController = require('../controllers/auth');
7
8 const router = express.Router();
9
10 router.get('/login', authController.getLogin);
11
12 router.get('/signup', authController.getSignup);
13
14 router.post('/login', authController.postLogin);
```

```

15
16 router.post(
17   '/signup',
18   [
19     check('email')
20       .isEmail()
21       .withMessage('Please enter a valid email')
22       .custom((value, {req}) => {
23         if (value === 'test@test.com'){
24           throw new Error('This Email Address is forbidden.')
25         }
26         return true
27       }),
28     /**this means please check password value in the body of the request
29      * so if there happens to be a password value in the headers,
30      * i don't care about that.
31    */
32   body(
33     'password',
34     /**if you want the same error message for all your validators
35      * but it should not be the default invalid value error message
36      * then grab error message in here the 2nd argument.
37    */
38     'Please enter a password with only numbers and text and at least 5 characters'
39   )
40     .isLength({min: 5})
41     .isAlphanumeric()
42   ],
43   authController.postSignup);
44
45 router.post('/logout', authController.postLogout);
46
47 router.get('/reset', authController.getReset);
48
49 router.post('/reset', authController.postReset);
50
51 router.get('/reset/:token', authController.getNewPassword);
52
53 router.post('/new-password', authController.postNewPassword);
54
55 module.exports = router;

```

* Chapter 291: Checking For Field Equality

1. update
- ./routes/auth.js

Screenshot of a Signup page showing a password validation error:

The page has a header with "Shop", "Products", "Login", and "Signup". The "Signup" button is highlighted.

The form fields are:

- E-Mail: test2@test.com
- Password: (containing 5 asterisks)
- Confirm Password: (containing 10 asterisks)

A red error message box at the top right contains the text: "Please enter a password with only numbers and text and at least 5 characters."

Signup button is visible below the form.

Screenshot of a Signup page showing a field equality error:

The page has a header with "Shop", "Products", "Login", and "Signup". The "Signup" button is highlighted.

The form fields are:

- E-Mail:
- Password:
- Confirm Password:

A red error message box at the top right contains the text: "Passwords have to match!"

Signup button is visible below the form.

- 'password' and 'confirm password' is not equal

A screenshot of a web browser showing a 'Signup' form. The URL is 'localhost:3000/signup'. The form has fields for 'E-Mail' (containing 'test2@test.com'), 'Password', and 'Confirm Password'. A red error message box at the top says 'Passwords have to match!'. A 'Signup' button is at the bottom.

A screenshot of a web browser showing a 'Signup' form. The URL is 'localhost:3000/signup'. The form has fields for 'E-Mail', 'Password', and 'Confirm Password'. A red error message box at the top says 'E-Mail exists already, please pick a different one.' A 'Signup' button is at the bottom.

- 'password' and 'confirm password' is equal.

```

1 // ./routes/auth.js
2
3 const express = require('express');
4 const { check, body } = require('express-validator/check')
5
6 const authController = require('../controllers/auth');
7
8 const router = express.Router();
9
10 router.get('/login', authController.getLogin);
11
12 router.get('/signup', authController.getSignup);
13

```

```

14 router.post('/login', authController.postLogin);
15
16 router.post(
17   '/signup',
18   [
19     check('email')
20       .isEmail()
21       .withMessage('Please enter a valid email')
22       .custom((value, {req}) => {
23         if (value === 'test@test.com'){
24           throw new Error('This Email Address is forbidden.')
25         }
26         return true
27       }),
28     body(
29       'password',
30       'Please enter a password with only numbers and text and at least 5 characters'
31     )
32       .isLength({min: 5})
33       .isAlphanumeric(),
34     body('confirmPassword').custom((value, {req}) => {
35       if (value !== req.body.password) {
36         throw new Error('Passwords have to match!')
37       }
38       return true
39     })
40   ],
41   authController.postSignup);
42
43 router.post('/logout', authController.postLogout);
44
45 router.get('/reset', authController.getReset);
46
47 router.post('/reset', authController.postReset);
48
49 router.get('/reset/:token', authController.getNewPassword);
50
51 router.post('/new-password', authController.postNewPassword);
52
53 module.exports = router;

```

* Chapter 292: Adding Async Validation

1. update
- ./controllers/auth.js
- ./routes/auth.js

Signup

localhost:3000/signup

Shop Products Login Signup

E-Mail exists already, please pick a different one.

E-Mail
test2@test.com

Password

Confirm Password

Signup

Signup

localhost:3000/signup

Shop Products Login Signup

E-Mail exists already, please pick a different one.

E-Mail
[empty]

Password
[empty]

Confirm Password
[empty]

Signup

Join today

The screenshot shows a web browser window with a dark green header bar. On the left of the header are 'Shop' and 'Products' links. On the right are 'Login' and 'Signup' links. Below the header is a form with a red border around the email input field. Inside the red border, the text 'E-Mail exists already, please pick a different one.' is displayed. The form has three input fields: 'E-Mail' containing 'test2121@test.com', 'Password' containing '*****', and 'Confirm Password' containing '*****'. At the bottom is a blue 'Signup' button.

The screenshot shows a web browser window with a dark green header bar. On the left of the header are 'Shop' and 'Products' links. On the right are 'Login' and 'Signup' links. Below the header is a form with two input fields: 'E-Mail' and 'Password', both currently empty. Below the password field are two buttons: a blue 'Login' button and a smaller grey 'Reset Password' button.

```
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  
21 exports.getLogin = (req, res, next) => {  
22   let message = req.flash('error');  
23   if (message.length > 0) {  
24     message = message[0];  
25   } else {  
26     message = null;  
27   }  
28   res.render('auth/login', {  
29     path: '/login',  
30     pageTitle: 'Login',  
31     errorMessage: message  
32   });  
33 };  
34  
35 exports.getSignup = (req, res, next) => {  
36   let message = req.flash('error');  
37   if (message.length > 0) {  
38     message = message[0];  
39   } else {  
40     message = null;  
41   }  
42   res.render('auth/signup', {  
43     path: '/signup',  
44     pageTitle: 'Signup',  
45     errorMessage: message  
46   });  
47 };  
48  
49 exports.postLogin = (req, res, next) => {  
50   const email = req.body.email;  
51   const password = req.body.password;  
52   User.findOne({ email: email })  
53     .then(user => {  
54       if (!user) {  
55         req.flash('error', 'Invalid email or password.');//  
56         return res.redirect('/login');//  
57       }  
58       bcrypt  
59         .compare(password, user.password)  
60         .then(doMatch => {  
61           if (doMatch) {  
62             req.session.isLoggedIn = true;  
63             req.session.user = user;  
64             return req.session.save(err => {  
65               if (err) {  
66                 console.log(err);  
67                 res.redirect('/');//  
68               }  
69             req.flash('error', 'Invalid email or password.');//  
70             res.redirect('/login');//  
71           }  
72         }  
73       }  
74     }  
75   }  
76 };
```

```
71     })
72     .catch(err => {
73       console.log(err);
74       res.redirect('/login');
75     });
76   })
77   .catch(err => console.log(err));
78 };
79
80 exports.postSignup = (req, res, next) => {
81   const email = req.body.email;
82   const password = req.body.password;
83   const errors = validationResult(req)
84   if(!errors.isEmpty()){
85     console.log(errors.array())
86     return res.status(422).render('auth/signup', {
87       path: '/signup',
88       pageTitle: 'Signup',
89       errorMessage: errors.array()[0].msg
90     });
91   }
92   bcrypt
93     .hash(password, 12)
94     .then(hashedPassword => {
95       const user = new User({
96         email,
97         password: hashedPassword,
98         cart: { items: [] }
99       });
100      return user.save();
101    })
102    .then(result => {
103      res.redirect('/login');
104      return transporter.sendMail({
105        to: email,
106        from: 'shop@node-complete.com',
107        subject: 'Signup succeeded!',
108        html: '<h1>You successfully signed up!</h1>'
109      });
110    })
111    .catch(err => {
112      console.log(err);
113    });
114 };
115
116 exports.postLogout = (req, res, next) => {
117   req.session.destroy(err => {
118     console.log(err);
119     res.redirect('/');
120   });
121 };
122
123 exports.getReset = (req, res, next) => {
124   let message = req.flash('error');
125   if (message.length > 0) {
126     message = message[0];
```

```
127 } else {
128     message = null;
129 }
130 res.render('auth/reset', {
131     path: '/reset',
132     pageTitle: 'Reset Password',
133     errorMessage: message
134 });
135 };
136
137 exports.postReset = (req, res, next) => {
138     crypto.randomBytes(32, (err, buffer) => {
139         if (err) {
140             console.log(err);
141             return res.redirect('/reset');
142         }
143         const token = buffer.toString('hex');
144         User.findOne({ email: req.body.email })
145             .then(user => {
146                 if (!user) {
147                     req.flash('error', 'No account with that email found.');
148                     return res.redirect('/reset');
149                 }
150                 user.resetToken = token;
151                 user.resetTokenExpiration = Date.now() + 3600000;
152                 return user.save();
153             })
154             .then(result => {
155                 res.redirect('/');
156                 transporter.sendMail({
157                     to: req.body.email,
158                     from: 'shop@node-complete.com',
159                     subject: 'Password reset',
160                     html: `
161                         <p>You requested a password reset</p>
162                         <p>Click this <a href="http://localhost:3000/reset/${token}">link</a> to set a
new password.</p>
163                         `
164                 });
165             })
166             .catch(err => {
167                 console.log(err);
168             });
169     });
170 };
171
172 exports.getNewPassword = (req, res, next) => {
173     const token = req.params.token;
174     User.findOne({
175         resetToken: token,
176         resetTokenExpiration: { $gt: Date.now() }
177     })
178         .then(user => {
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/new-password', {
186         path: '/new-password',
187         pageTitle: 'New Password',
188         errorMessage: message,
189         userId: user._id.toString(),
190         passwordToken: token
191     });
192 }
193 .catch(err => {
194     console.log(err);
195 });
196 };
197
198 exports.postNewPassword = (req, res, next) => {
199     const newPassword = req.body.password
200     const userId = req.body.userId
201     const passwordToken = req.body.passwordToken
202     let resetUser
203
204     User.findOne({
205         resetToken: passwordToken,
206         resetTokenExpiration: {$gt: Date.now()},
207         _id: userId
208     })
209     .then(user => {
210         resetUser = user
211         return bcrypt.hash(newPassword, 12)
212     })
213     .then(hashedPassword => {
214         resetUser.resetToken = undefined
215         resetUser.resetTokenExpiration = undefined
216         return resetUser.save()
217     })
218     .then(result => {
219         res.redirect('/login')
220     })
221     .catch(err => {
222         console.log(err)
223     })
224
225 }
226

```

```

1 // ./routes/auth.js
2
3 const express = require('express');
4 const { check, body } = require('express-validator/check')
5
6 const authController = require('../controllers/auth');
7 const User = require('../models/user')
8
9 const router = express.Router();
10
11 router.get('/login', authController.getLogin);

```

```
12
13 router.get('/signup', authController.getSignup);
14
15 router.post('/login', authController.postLogin);
16
17 router.post(
18   '/signup',
19   [
20     check('email')
21       .isEmail()
22       .withMessage('Please enter a valid email')
23       /**the express-validator package will check for a 'custom()' validator
24       * to return true or false,
25       * to return thrown error or to return a Promise.
26       * if 'then()' is a promise because every 'then()' block returns a new promise
27       * if we return a Promise,
28       * then express-validator will wait for this Promise to be fulfilled
29       * and if it fulfills with nothing no error,
30       * it treats this validation as successful
31       *
32       * if it resolves with some rejection in the end,
33       * which will happen if we make it into this if block,
34       * then express validator will detect this rejection
35       * and will store this as an error message.
36       *
37       * this is how we can add our own asynchronous validation
38       * because we have to reach out to the database
39       * which is not an instant task
40       * but express validator will wait for us here.
41     */
42     .custom((value, {req}) => {
43       //if (value === 'test@test.com'){
44       //  throw new Error('This Email Address is forbidden.')
45       //}
46       //return true
47       return User.findOne({ email: value })
48       .then(userDoc => {
49         if (userDoc) {
50           /**'Promise' is a built-in javascript object
51           * and with reject, i throw an error inside of the promise
52           * and i reject with this error message i used before
53           */
54           return Promise.reject(
55             'E-Mail exists already, please pick a different one.'
56           )
57         }
58       })
59     },
60     body(
61       'password',
62       'Please enter a password with only numbers and text and at least 5 characters'
63     )
64       .isLength({min: 5})
65       .isAlphanumeric(),
66     body('confirmPassword').custom((value, {req}) => {
67       if (value !== req.body.password) {
```

```

68     ....      throw new Error('Passwords have to match!')
69     ....    }
70     ....      return true
71   ....  })
72   .... ],
73   .... authController.postSignup);
74
75 router.post('/logout', authController.postLogout);
76
77 router.get('/reset', authController.getReset);
78
79 router.post('/reset', authController.postReset);
80
81 router.get('/reset/:token', authController.getNewPassword);
82
83 router.post('/new-password', authController.postNewPassword);
84
85 module.exports = router;

```

* Chapter 293: Keeping User Input

1. update
- ./controllers/auth.js
- ./views/auth/signup.ejs

- you can see that it kept the old e-mail because we return that with the response and we then output it in our view and we should do the same for password and confirm Password
-

- we kept all the data after clicking sign up button.

```

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
21 exports.getLogin = (req, res, next) => {

```

```
22 let message = req.flash('error');
23 if (message.length > 0) {
24   message = message[0];
25 } else {
26   message = null;
27 }
28 res.render('auth/login', {
29   path: '/login',
30   pageTitle: 'Login',
31   errorMessage: message
32 });
33 };
34
35 exports.getSignup = (req, res, next) => {
36   let message = req.flash('error');
37   if (message.length > 0) {
38     message = message[0];
39   } else {
40     message = null;
41   }
42   res.render('auth/signup', {
43     path: '/signup',
44     pageTitle: 'Signup',
45     errorMessage: message,
46     oldInput: {
47       email: '',
48       password: '',
49       confirmPassword: ''
50     }
51   });
52 };
53
54 exports.postLogin = (req, res, next) => {
55   const email = req.body.email;
56   const password = req.body.password;
57
58   const errors = validationResult(req);
59   if (!errors.isEmpty()) {
60     return res.status(422).render('auth/login', {
61       path: '/login',
62       pageTitle: 'Login',
63       errorMessage: errors.array()[0].msg
64     });
65   }
66
67 User.findOne({ email: email })
68   .then(user => {
69     if (!user) {
70       req.flash('error', 'Invalid email or password.');
71       return res.redirect('/login');
72     }
73     bcrypt
74       .compare(password, user.password)
75       .then(doMatch => {
76         if (doMatch) {
77           req.session.isLoggedIn = true;
```

```

78         req.session.user = user;
79         return req.session.save(err => {
80             console.log(err);
81             res.redirect('/');
82         });
83     }
84     req.flash('error', 'Invalid email or password.');
85     res.redirect('/login');
86 )
87 .catch(err => {
88     console.log(err);
89     res.redirect('/login');
90 });
91 }
92 .catch(err => console.log(err));
93 };
94
95 exports.postSignup = (req, res, next) => {
96     const email = req.body.email;
97     const password = req.body.password;
98
99     const errors = validationResult(req);
100    if (!errors.isEmpty()) {
101        console.log(errors.array());
102        return res.status(422).render('auth/signup', {
103            path: '/signup',
104            pageTitle: 'Signup',
105            errorMessage: errors.array()[0].msg,
106            oldInput: {
107                email: email,
108                password: password,
109                confirmPassword: req.body.confirmPassword
110            }
111        });
112    }
113
114 bcrypt
115     .hash(password, 12)
116     .then(hashedPassword => {
117         const user = new User({
118             email: email,
119             password: hashedPassword,
120             cart: { items: [] }
121         });
122         return user.save();
123     })
124     .then(result => {
125         res.redirect('/login');
126         // return transporter.sendMail({
127         //     to: email,
128         //     from: 'shop@node-complete.com',
129         //     subject: 'Signup succeeded!',
130         //     html: '<h1>You successfully signed up!</h1>'
131         // });
132     })
133     .catch(err => {

```

```

134     console.log(err);
135   });
136 };
137
138 exports.postLogout = (req, res, next) => {
139   req.session.destroy(err => {
140     console.log(err);
141     res.redirect('/');
142   });
143 };
144
145 exports.getReset = (req, res, next) => {
146   let message = req.flash('error');
147   if (message.length > 0) {
148     message = message[0];
149   } else {
150     message = null;
151   }
152   res.render('auth/reset', {
153     path: '/reset',
154     pageTitle: 'Reset Password',
155     errorMessage: message
156   });
157 };
158
159 exports.postReset = (req, res, next) => {
160   crypto.randomBytes(32, (err, buffer) => {
161     if (err) {
162       console.log(err);
163       return res.redirect('/reset');
164     }
165     const token = buffer.toString('hex');
166     User.findOne({ email: req.body.email })
167       .then(user => {
168         if (!user) {
169           req.flash('error', 'No account with that email found.');
170           return res.redirect('/reset');
171         }
172         user.resetToken = token;
173         user.resetTokenExpiration = Date.now() + 3600000;
174         return user.save();
175       })
176       .then(result => {
177         res.redirect('/');
178         transporter.sendMail({
179           to: req.body.email,
180           from: 'shop@node-complete.com',
181           subject: 'Password reset',
182           html: `
183             <p>You requested a password reset</p>
184             <p>Click this <a href="http://localhost:3000/reset/${token}">link</a> to set a
new password.</p>
185             `
186           });
187       })
188       .catch(err => {

```

```
189         console.log(err);
190     });
191   });
192 };
193
194 exports.getNewPassword = (req, res, next) => {
195   const token = req.params.token;
196   User.findOne({ resetToken: token, resetTokenExpiration: { $gt: Date.now() } })
197     .then(user => {
198       let message = req.flash('error');
199       if (message.length > 0) {
200         message = message[0];
201       } else {
202         message = null;
203       }
204       res.render('auth/new-password', {
205         path: '/new-password',
206         pageTitle: 'New Password',
207         errorMessage: message,
208         userId: user._id.toString(),
209         passwordToken: token
210       });
211     })
212     .catch(err => {
213       console.log(err);
214     });
215 };
216
217 exports.postNewPassword = (req, res, next) => {
218   const newPassword = req.body.password;
219   const userId = req.body.userId;
220   const passwordToken = req.body.passwordToken;
221   let resetUser;
222
223   User.findOne({
224     resetToken: passwordToken,
225     resetTokenExpiration: { $gt: Date.now() },
226     _id: userId
227   })
228     .then(user => {
229       resetUser = user;
230       return bcrypt.hash(newPassword, 12);
231     })
232     .then(hashedPassword => {
233       resetUser.password = hashedPassword;
234       resetUser.resetToken = undefined;
235       resetUser.resetTokenExpiration = undefined;
236       return resetUser.save();
237     })
238     .then(result => {
239       res.redirect('/login');
240     })
241     .catch(err => {
242       console.log(err);
243     });
244};
```

```

1 <!--./views/auth/signup.ejs-->
2
3 <%-- include('../includes/head.ejs') %>
4   <link rel="stylesheet" href="/css/forms.css">
5   <link rel="stylesheet" href="/css/auth.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="login-form" action="/signup" method="POST" novalidate>
16      <div class="form-control">
17        <label for="email">E-Mail</label>
18        <input type="email" name="email" id="email" value="<%= oldInput.email %>">
19      </div>
20      <div class="form-control">
21        <label for="password">Password</label>
22        <input type="password" name="password" id="password" value="<%= oldInput.password %>">
23      </div>
24      <div class="form-control">
25        <label for="confirmPassword">Confirm Password</label>
26        <input type="password" name="confirmPassword" id="confirmPassword" value="<%= oldInput.confirmPassword %>">
27      </div>
28      <input type="hidden" name="_csrf" value="<%= csrfToken %>">
29      <button class="btn" type="submit">Signup</button>
30    </form>
31  </main>
32 <%-- include('../includes/end.ejs') %>

```

* Chapter 294: Adding Conditional CSS Classes

1. update
- ./controllers/auth.js
- ./views/signup.ejs
- ./public/css/forms.css

Shop Products Login Signup

E-Mail exists already, please pick a different one.

E-Mail

Password

Confirm Password

The screenshot shows a web application's sign-up page. At the top, there are navigation links for 'Shop' and 'Products'. On the right, there are 'Login' and 'Signup' buttons. The main area contains a form with the following fields:

- Email: A text input field containing "test@test.com".
- Password: A text input field containing "*****".
- Confirm Password: A text input field containing "*****".
- Sign Up: A blue button.

A red error message box is displayed above the form, stating: "E-Mail exists already, please pick a different one." The browser's developer tools are open, showing the HTML structure and the CSS rule for the invalid email input field. The CSS rule is: ".form-control input.invalid { border-color: #f00; }".

```
<!DOCTYPE html>
<html lang="en">
  <head>...</head>
  <body cz-shortcut-listen="true">
    <div class="backdrop"></div>
    <header class="main-header">...</header>
    <nav class="mobile-nav">...</nav>
    <main>
      <div class="user-message user-message--error">E-Mail exists already, please pick a different one.</div>
      <form class="login-form" action="/signup" method="POST" novalidate>
        <div class="form-control">
          <label for="email">E-Mail</label>
          <input class="invalid" type="email" name="email" id="email" value="test@test.com" ... $0>
        ...
      </form>
    </main>
  </body>
</html>
```

Styles Computed Event Listeners >

Filter element.style { }

.form-control input.invalid { border-color: #f00; }

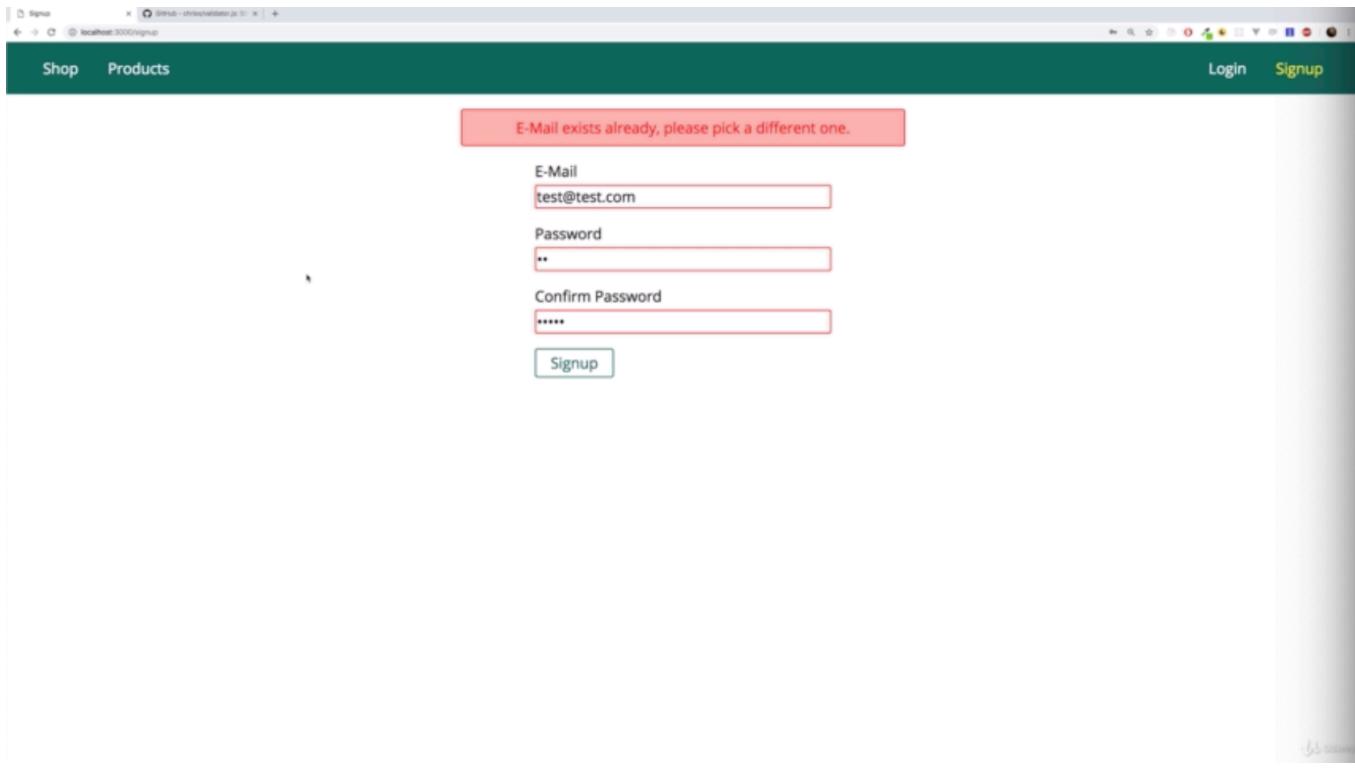
:hover .cls +

Console What's New X

Highlights from the Chrome 68 update

Eager evaluation

- that red box is because this email is invalid.



```
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     /**we don't get an error regarding this not being found
32      * when it's first rendered.
33      */
34     oldInput: {
35       email: '' ,
```

```
36     password: '',
37   },
38   validationErrors: []
39 });
40 };
41
42 exports.getSignup = (req, res, next) => {
43   let message = req.flash('error');
44   if (message.length > 0) {
45     message = message[0];
46   } else {
47     message = null;
48   }
49   res.render('auth/signup', {
50     path: '/signup',
51     pageTitle: 'Signup',
52     errorMessage: message,
53     oldInput: {
54       email: '',
55       password: '',
56       confirmPassword: ''
57     },
58     /**because we got no errors. */
59     validationErrors: []
60   });
61 };
62
63 exports.postLogin = (req, res, next) => {
64   const email = req.body.email;
65   const password = req.body.password;
66
67   const errors = validationResult(req);
68   if (!errors.isEmpty()) {
69     return res.status(422).render('auth/login', {
70       path: '/login',
71       pageTitle: 'Login',
72       errorMessage: errors.array()[0].msg,
73       oldInput: {
74         email: email,
75         password: password
76       },
77       validationErrors: errors.array()
78     });
79   }
80
81   User.findOne({ email: email })
82     .then(user => {
83       if (!user) {
84         return res.status(422).render('auth/login', {
85           path: '/login',
86           pageTitle: 'Login',
87           errorMessage: 'Invalid email or password.',
88           oldInput: {
89             email: email,
90             password: password
91           },
92         });
93       }
94     })
95     .catch(error => {
96       console.error(error);
97       res.status(500).send('There was a problem with your request');
98     });
99 }
```

```
92     validationErrors: []
93   });
94 }
95 bcrypt
96   .compare(password, user.password)
97   .then(doMatch => {
98     if (doMatch) {
99       req.session.isLoggedIn = true;
100      req.session.user = user;
101      return req.session.save(err => {
102        console.log(err);
103        res.redirect('/');
104      });
105    }
106    return res.status(422).render('auth/login', {
107      path: '/login',
108      pageTitle: 'Login',
109      errorMessage: 'Invalid email or password.',
110      oldInput: {
111        email: email,
112        password: password
113      },
114      validationErrors: []
115    });
116  })
117  .catch(err => {
118    console.log(err);
119    res.redirect('/login');
120  });
121})
122 .catch(err => console.log(err));
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       /**so the full array is now returned as well
142        * so with 'errorMessage', i'm returning not only just the first message which i
143        * picked to show
144        * but also the full array of errors with 'validationErrors'
145        */
146       validationErrors: errors.array()
147     });
148 }
```

```
147 }
148
149 bcrypt
150   .hash(password, 12)
151   .then(hashedPassword => {
152     const user = new User({
153       email: email,
154       password: hashedPassword,
155       cart: { items: [] }
156     });
157     return user.save();
158   })
159   .then(result => {
160     res.redirect('/login');
161     // return transporter.sendMail({
162     //   to: email,
163     //   from: 'shop@node-complete.com',
164     //   subject: 'Signup succeeded!',
165     //   html: '<h1>You successfully signed up!</h1>'
166     // });
167   })
168   .catch(err => {
169     console.log(err);
170   });
171 };
172
173 exports.postLogout = (req, res, next) => {
174   req.session.destroy(err => {
175     console.log(err);
176     res.redirect('/');
177   });
178 };
179
180 exports.getReset = (req, res, next) => {
181   let message = req.flash('error');
182   if (message.length > 0) {
183     message = message[0];
184   } else {
185     message = null;
186   }
187   res.render('auth/reset', {
188     path: '/reset',
189     pageTitle: 'Reset Password',
190     errorMessage: message
191   });
192 };
193
194 exports.postReset = (req, res, next) => {
195   crypto.randomBytes(32, (err, buffer) => {
196     if (err) {
197       console.log(err);
198       return res.redirect('/reset');
199     }
200     const token = buffer.toString('hex');
201     User.findOne({ email: req.body.email })
202       .then(user => {
```

```
203     if (!user) {
204         req.flash('error', 'No account with that email found.');
205         return res.redirect('/reset');
206     }
207     user.resetToken = token;
208     user.resetTokenExpiration = Date.now() + 3600000;
209     return user.save();
210 })
211 .then(result => {
212     res.redirect('/');
213     transporter.sendMail({
214         to: req.body.email,
215         from: 'shop@node-complete.com',
216         subject: 'Password reset',
217         html: `
218             <p>You requested a password reset</p>
219             <p>Click this <a href="http://localhost:3000/reset/${token}">link</a> to set a
220             new password.</p>
221             `
222     });
223 })
224 .catch(err => {
225     console.log(err);
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         console.log(err);
249     });
250 };
251
252 exports.postNewPassword = (req, res, next) => {
253     const newPassword = req.body.password;
254     const userId = req.body.userId;
255     const passwordToken = req.body.passwordToken;
256     let resetUser;
```

```

258 User.findOne({
259   resetToken: passwordToken,
260   resetTokenExpiration: { $gt: Date.now() },
261   _id: userId
262 })
263   .then(user => {
264     resetUser = user;
265     return bcrypt.hash(newPassword, 12);
266   })
267   .then(hashedPassword => {
268     resetUser.password = hashedPassword;
269     resetUser.resetToken = undefined;
270     resetUser.resetTokenExpiration = undefined;
271     return resetUser.save();
272   })
273   .then(result => {
274     res.redirect('/login');
275   })
276   .catch(err => {
277     console.log(err);
278   });
279 };
280

```

```

1 <!--./views/auth/signup.ejs-->
2
3 <%-- include('../includes/head.ejs') %>
4   <link rel="stylesheet" href="/css/forms.css">
5   <link rel="stylesheet" href="/css/auth.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="login-form" action="/signup" method="POST" novalidate>
16       <div class="form-control">
17         <label for="email">E-Mail</label>
18         <input
19           class="<% validationErrors.find(e => e.param === 'email') ? 'invalid' :
20             '' %>">
20           type="email"
21           name="email"
22           id="email"
23           value="<%= oldInput.email %>">
24         </div>
25         <div class="form-control">
26           <label for="password">Password</label>
27           <input
28             class="<% validationErrors.find(e => e.param === 'password') ?
29               'invalid' : '' %>">
29             type="password"
30             name="password"
31             id="password">

```

```

32           value="<% oldInput.password %>">
33     </div>
34     <div class="form-control">
35       <label for="confirmPassword">Confirm Password</label>
36       <input
37         class="<% validationErrors.find(e => e.param === 'confirmPassword') ?
38 'invalid' : '' %>">
39         type="password"
40         name="confirmPassword"
41         id="confirmPassword"
42         value="<% oldInput.confirmPassword %>">
43     </div>
44     <input type="hidden" name="_csrf" value="<% csrfToken %>">
45     <button class="btn" type="submit">Signup</button>
46   </form>
47 </main>
48 <%- include('../includes/end.ejs') %>
```

```

1 /*./public/css/forms.css*/
2
3 .form-control {
4   margin: 1rem 0;
5 }
6
7 .form-control label,
8 .form-control input,
9 .form-control textarea {
10   display: block;
11   width: 100%;
12   margin-bottom: 0.25rem;
13 }
14
15 .form-control input,
16 .form-control textarea {
17   border: 1px solid #a1a1a1;
18   font: inherit;
19   border-radius: 2px;
20 }
21
22 .form-control input:focus,
23 .form-control textarea:focus {
24   outline-color: #00695c;
25 }
26
27 .form-control input.invalid {
28   border-color: red;
29 }
```

* Chapter 295: Adding Validation To Login

1. update
- ./controllers/auth.js
- ./views/auth/login.ejs

The screenshot shows a web browser window with a green header bar containing 'Shop', 'Products', 'Login', and 'Signup' links. Below the header is a form with two fields: 'E-Mail' and 'Password'. The 'E-Mail' field contains 'test@test.com' and has a yellow background, indicating it is invalid. The 'Password' field is empty and has a white background. Below the form are two buttons: a dark blue 'Login' button and a light blue 'Reset Password' link.



Password has to be valid.

The screenshot shows the same login form. The 'E-Mail' field now has a red border and a red asterisk, and its background is white. The 'Password' field also has a red border and a red asterisk, and its background is white. The 'Login' and 'Reset Password' buttons remain the same.

Login

localhost:3000/login

Shop Products Login Signup

Password has to be valid.

E-Mail

Password

[Reset Password](#)

Login

localhost:3000/login

Shop Products Login Signup

Invalid email or password.

E-Mail

Password

[Reset Password](#)

John Doe

John Doe

Login

localhost:3000/login

Shop Products Login Signup

Invalid email or password.

E-Mail
test@test.com

Password

[Login](#) [Reset Password](#)

Login

localhost:3000/login

Shop Products Login Signup

Invalid email or password.

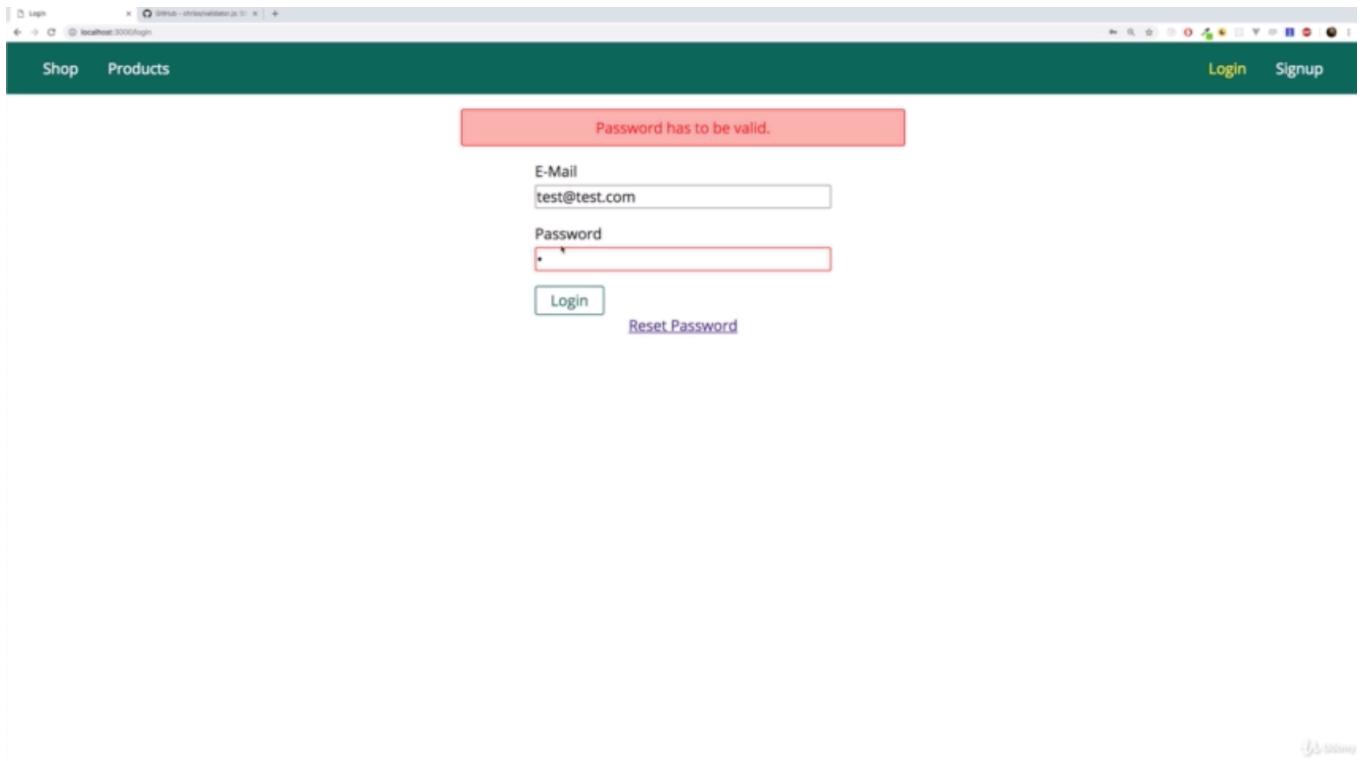
E-Mail
test@test.com

Password
•••••

[Login](#) [Reset Password](#)

John Doe

John Doe



```
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     /**we don't get an error regarding this not being found
32      * when it's first rendered.
33      */
34     oldInput: {
35       email: '',
```

```
36     password: '',
37   },
38   validationErrors: []
39 });
40 };
41
42 exports.getSignup = (req, res, next) => {
43   let message = req.flash('error');
44   if (message.length > 0) {
45     message = message[0];
46   } else {
47     message = null;
48   }
49   res.render('auth/signup', {
50     path: '/signup',
51     pageTitle: 'Signup',
52     errorMessage: message,
53     oldInput: {
54       email: '',
55       password: '',
56       confirmPassword: ''
57     },
58     validationErrors: []
59   });
60 };
61
62 exports.postLogin = (req, res, next) => {
63   const email = req.body.email;
64   const password = req.body.password;
65
66   const errors = validationResult(req);
67   if (!errors.isEmpty()) {
68     return res.status(422).render('auth/login', {
69       path: '/login',
70       pageTitle: 'Login',
71       errorMessage: errors.array()[0].msg,
72       oldInput: {
73         email: email,
74         password: password
75       },
76       validationErrors: errors.array()
77     });
78 }
79
80 User.findOne({ email: email })
81   .then(user => {
82     if (!user) {
83       return res.status(422).render('auth/login', {
84         path: '/login',
85         pageTitle: 'Login',
86         errorMessage: 'Invalid email or password.',
87         oldInput: {
88           email: email,
89           password: password
90         },
91         /**we have to add some object with param
```

```
92     * and that in this case it could be param email and param password
93     * if you wanna make sure you don't show what led to the error like below
94     *
95     *     validationErrors: [{param: 'email', param: 'password'}]
96     *
97     * but you could leave that out and only give out the message
98     */
99     validationErrors: []
100 );
101 }
102 bcrypt
103     .compare(password, user.password)
104     .then(doMatch => {
105         if (doMatch) {
106             req.session.isLoggedIn = true;
107             req.session.user = user;
108             return req.session.save(err => {
109                 console.log(err);
110                 res.redirect('/');
111             });
112         }
113         return res.status(422).render('auth/login', {
114             path: '/login',
115             pageTitle: 'Login',
116             errorMessage: 'Invalid email or password.',
117             oldInput: {
118                 email: email,
119                 password: password
120             },
121             validationErrors: []
122         });
123     })
124     .catch(err => {
125         console.log(err);
126         res.redirect('/login');
127     });
128 })
129 .catch(err => console.log(err));
130 };
131
132 exports.postSignup = (req, res, next) => {
133     const email = req.body.email;
134     const password = req.body.password;
135
136     const errors = validationResult(req);
137     if (!errors.isEmpty()) {
138         console.log(errors.array());
139         return res.status(422).render('auth/signup', {
140             path: '/signup',
141             pageTitle: 'Signup',
142             errorMessage: errors.array()[0].msg,
143             oldInput: {
144                 email: email,
145                 password: password,
146                 confirmPassword: req.body.confirmPassword
147             },

```

```
148     validationErrors: errors.array()
149   });
150 }
151
152 bcrypt
153   .hash(password, 12)
154   .then(hashedPassword => {
155     const user = new User({
156       email: email,
157       password: hashedPassword,
158       cart: { items: [] }
159     });
160     return user.save();
161   })
162   .then(result => {
163     res.redirect('/login');
164     // return transporter.sendMail({
165     //   to: email,
166     //   from: 'shop@node-complete.com',
167     //   subject: 'Signup succeeded!',
168     //   html: '<h1>You successfully signed up!</h1>'
169     // });
170   })
171   .catch(err => {
172     console.log(err);
173   });
174 };
175
176 exports.postLogout = (req, res, next) => {
177   req.session.destroy(err => {
178     console.log(err);
179     res.redirect('/');
180   });
181 };
182
183 exports.getReset = (req, res, next) => {
184   let message = req.flash('error');
185   if (message.length > 0) {
186     message = message[0];
187   } else {
188     message = null;
189   }
190   res.render('auth/reset', {
191     path: '/reset',
192     pageTitle: 'Reset Password',
193     errorMessage: message
194   });
195 };
196
197 exports.postReset = (req, res, next) => {
198   crypto.randomBytes(32, (err, buffer) => {
199     if (err) {
200       console.log(err);
201       return res.redirect('/reset');
202     }
203     const token = buffer.toString('hex');
```

```

204     User.findOne({ email: req.body.email })
205       .then(user => {
206         if (!user) {
207           req.flash('error', 'No account with that email found.');
208           return res.redirect('/reset');
209         }
210         user.resetToken = token;
211         user.resetTokenExpiration = Date.now() + 3600000;
212         return user.save();
213       })
214       .then(result => {
215         res.redirect('/');
216         transporter.sendMail({
217           to: req.body.email,
218           from: 'shop@node-complete.com',
219           subject: 'Password reset',
220           html: `
221             <p>You requested a password reset</p>
222             <p>Click this <a href="http://localhost:3000/reset/${token}">link</a> to set a
new password.</p>
223             `
224         });
225       })
226       .catch(err => {
227         console.log(err);
228       });
229   );
230 };
231
232 exports.getNewPassword = (req, res, next) => {
233   const token = req.params.token;
234   User.findOne({ resetToken: token, resetTokenExpiration: { $gt: Date.now() } })
235     .then(user => {
236       let message = req.flash('error');
237       if (message.length > 0) {
238         message = message[0];
239       } else {
240         message = null;
241       }
242       res.render('auth/new-password', {
243         path: '/new-password',
244         pageTitle: 'New Password',
245         errorMessage: message,
246         userId: user._id.toString(),
247         passwordToken: token
248       });
249     })
250     .catch(err => {
251       console.log(err);
252     });
253 };
254
255 exports.postNewPassword = (req, res, next) => {
256   const newPassword = req.body.password;
257   const userId = req.body.userId;
258   const passwordToken = req.body.passwordToken;

```

```

259 let resetUser;
260
261 User.findOne({
262   resetToken: passwordToken,
263   resetTokenExpiration: { $gt: Date.now() },
264   _id: userId
265 })
266   .then(user => {
267     resetUser = user;
268     return bcrypt.hash(newPassword, 12);
269   })
270   .then(hashedPassword => {
271     resetUser.password = hashedPassword;
272     resetUser.resetToken = undefined;
273     resetUser.resetTokenExpiration = undefined;
274     return resetUser.save();
275   })
276   .then(result => {
277     res.redirect('/login');
278   })
279   .catch(err => {
280     console.log(err);
281   });
282 };
283

```

```

1 <!--./views/auth/login.ejs-->
2
3 <%-- include('../includes/head.ejs') %>
4   <link rel="stylesheet" href="/css/forms.css">
5   <link rel="stylesheet" href="/css/auth.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="login-form" action="/login" method="POST">
16       <div class="form-control">
17         <label for="email">E-Mail</label>
18         <input
19           class="<%= validationErrors.find(e => e.param === 'email') ? 'invalid' : '' %>">
20           type="email"
21           name="email"
22           id="email"
23           value="<%= oldInput.email %>">
24       </div>
25       <div class="form-control">
26         <label for="password">Password</label>
27         <input
28           class="<%= validationErrors.find(e => e.param === 'password') ? 'invalid' : '' %>">
29           type="password"

```

```

30           name="password"
31           id="password"
32           value="<% oldInput.password %>">
33       </div>
34       <input type="hidden" name="_csrf" value="<% csrfToken %>">
35       <button class="btn" type="submit">Login</button>
36   </form>
37   <div class="centered">
38     <a href="/reset">Reset Password</a>
39   </div>
40 </main>
41 <%- include('../includes/end.ejs') %>

```

* Chapter 296: Sanitizing Data

1. update
- ./routes/auth.js

The screenshot shows a browser window displaying the GitHub repository for 'chris/validator.js'. The specific page shown is 'Sanitizers'. The page contains a table listing several sanitization functions, each with a brief description. Below the table, there is additional information about the 'ltrim' function, specifically regarding email address canonicalization.

Sanitizer	Description
<code>blacklist(input, chars)</code>	remove characters that appear in the blacklist. The characters are used in a RegExp and so you will need to escape some chars, e.g. <code>blacklist(input, '\x{1}\x{2}\x{3}')</code> .
<code>escape(input)</code>	replace <, >, &, ', " and / with HTML entities.
<code>unescape(input)</code>	replaces HTML encoded entities with <, >, &, ', " and / .
<code>ltrim([, chars])</code>	trim characters from the left-side of the input. canonicalizes an email address. (This doesn't validate that the input is an email, if you want to validate the email use <code>isEmail</code> beforehand)

options is an object with the following keys and default values:

- `all_lowercase: true` - Transforms the local part (before the @ symbol) of all email addresses to lowercase. Please note that this may violate RFC 5321, which gives providers the possibility to treat the local part of email addresses in a case sensitive way (although in practice most - yet not all - providers don't). The domain part of the email address is always lowercased, as it's case insensitive per RFC 1035.
- `gmail_lowercase: true` - GMail addresses are known to be case-insensitive, so this switch allows lowercasing them even when `all_lowercase` is set to false. Please note that when `all_lowercase` is true, GMail addresses are lowercased regardless of the value of this setting.
- `gmail_remove_dots: true`: Removes dots from the local part of the email address, as GMail ignores them (e.g. "john.doe" and "johndoe" are considered equal).
- `gmail_remove_subaddress: true`: Normalizes addresses by removing "sub-addresses", which is the part following a "+" sign (e.g. "foo+bar@gmail.com" becomes "foo@gmail.com").
- `gmail_convert_gmaildotcom: true`: Converts addresses with domain @googlemail.com to @gmail.com, as they're equivalent.

normalizeEmail(email
[, options])

- `all_lowercase: true` - Transforms the local part (before the @ symbol) of all email addresses to lowercase. Please note that this may violate RFC 5321, which gives providers the possibility to treat the local part of email addresses in a case sensitive way (although in practice most - yet not all - providers don't). The domain part of the email address is always lowercased, as it's case insensitive per RFC 1035.
- `gmail_lowercase: true` - GMail addresses are known to be case-insensitive, so this switch allows lowercasing them even when `all_lowercase` is set to false. Please note that when `all_lowercase` is true, GMail addresses are lowercased regardless of the value of this setting.
- `gmail_remove_dots: true`: Removes dots from the local part of the email address, as GMail ignores them (e.g. "john.doe" and "john.doe" are considered equal).
- `gmail_remove_subaddress: true`: Normalizes addresses by removing "sub-addresses", which is the part following a "+" sign (e.g. "foo+bar@gmail.com" becomes "foo@gmail.com").
- `gmail_convert_gmaildotcom: true`: Converts addresses with domain @googlemail.com to @gmail.com, as they're equivalent.
- `outlookdotcom_lowercase: true` - Outlook.com addresses (including Windows Live and Hotmail) are known to be case-insensitive, so this switch allows lowercasing them even when `all_lowercase` is set to false. Please note that when `all_lowercase` is true, Outlook.com addresses are lowercased regardless of the value of this setting.
- `outlookdotcom_remove_subaddress: true`: Normalizes addresses by removing "sub-addresses", which is the part following a "+" sign (e.g. "foo+bar@outlook.com" becomes "foo@outlook.com").
- `yahoo_lowercase: true` - Yahoo Mail addresses are known to be case-insensitive, so this switch allows lowercasing them even when `all_lowercase` is set to false. Please note that when `all_lowercase` is true, Yahoo Mail addresses are lowercased regardless of the value of this setting.
- `yahoo_remove_subaddress: true`: Normalizes addresses by removing "sub-addresses", which is the part following a "-" sign (e.g. "foo-bar@yahoo.com" becomes "foo@yahoo.com").
- `icloud_lowercase: true` - iCloud addresses (including MobileMe) are known to be case-insensitive, so this switch allows lowercasing them even when `all_lowercase` is set to false. Please note that when `all_lowercase` is true, iCloud addresses are lowercased regardless of the value of this setting.
- `icloud_remove_subaddress: true`: Normalizes addresses by removing "sub-addresses", which is

- you can ensure that there is no excess whitespace in a string passed by the user on the left or on the right. or you can normalize an email which means it's converted to lowercase and things like that.

- there are a couple of things you can do to make sure the data you get is not just valid but also is stored in a uniform way.

The screenshot shows the MongoDB Compass interface connected to a cluster named "Cluster0-shard-0". The sidebar on the left lists databases "admin", "local", and "shop" (which is expanded), along with "orders", "products", "sessions", and "users". The main area displays the "shop.users" collection, which has 11 documents. The "Documents" tab is selected, showing a list of documents with their _id, cart, email, password, and __v fields. One document is highlighted with a red border and a "Document Flagged For Deletion." message at the bottom. The bottom navigation bar includes buttons for "CANCEL", "DELETE", and "Edit".

My Cluster

Cluster0-shard-0 REPLICA SET 3 NODES MongoDB 3.6.8 Enterprise

C 3 DBS 11 COLLECTIONS

shop.users

Documents Aggregations Explain Plan Indexes

FILTER

INSERT DOCUMENT VIEW LIST TABLE

Displaying documents 1 - 4 of 4 < > C

Documents N/A TOTAL SIZE N/A AVG. SIZE N/A INDEXES N/A TOTAL SIZE N/A AVG. SIZE N/A

_id: ObjectId("5bade42501bea653ba3a74e5")
cart: Object
email: "test2@test.com"
password: "\$2a\$12\$p8X2kME/tXuPTKNQ25YjxuzuPay.FqIWePVP52dX32pHR4c2Cje3G"
__v: 1

_id: ObjectId("5bb1e26dc08732d672c33b4")
cart: Object
email: "test33@test.com"
password: "\$2a\$12\$SMJbe9a.ispdevprNmicw.Anp5/Nv/EVsrdFsUR7j80w3C2hRb0vW"
__v: 0

_id: ObjectId("5bb1e4a39e22c32e21f0c0a5")
cart: Object
email: "test121@test.com"
password: "\$2a\$12\$2tLVV/cZvBdQ2NrXe4frk.1jMeKvRXeAl00ZaRqTTW3kAatkHg2P6"
__v: 0

Document Deleted.

CANCEL DELETE

The screenshot shows the MongoDB Compass interface with the 'shop.users' collection selected. The interface includes a sidebar with databases and collections, and a main panel for document management. The main panel shows four documents with their details. The fourth document is highlighted with a red border. A message 'Document Deleted.' is displayed at the bottom of the main panel.

My Cluster

Cluster0-shard-0 REPLICA SET 3 NODES MongoDB 3.6.8 Enterprise

C 3 DBS 11 COLLECTIONS

shop.users

Documents Aggregations Explain Plan Indexes

FILTER

INSERT DOCUMENT VIEW LIST TABLE

Displaying documents 1 - 0 of 0 < > C

Documents N/A TOTAL SIZE N/A AVG. SIZE N/A INDEXES N/A TOTAL SIZE N/A AVG. SIZE N/A

The screenshot shows the MongoDB Compass interface with the 'shop.users' collection selected. The main panel displays a message 'Displaying documents 1 - 0 of 0' and shows the standard header for document management. There are no documents listed in the main panel.

Signup

localhost:3000/signup

Shop Products Login Signup

E-Mail

Confirm Password

Signup

localhost:3000/signup

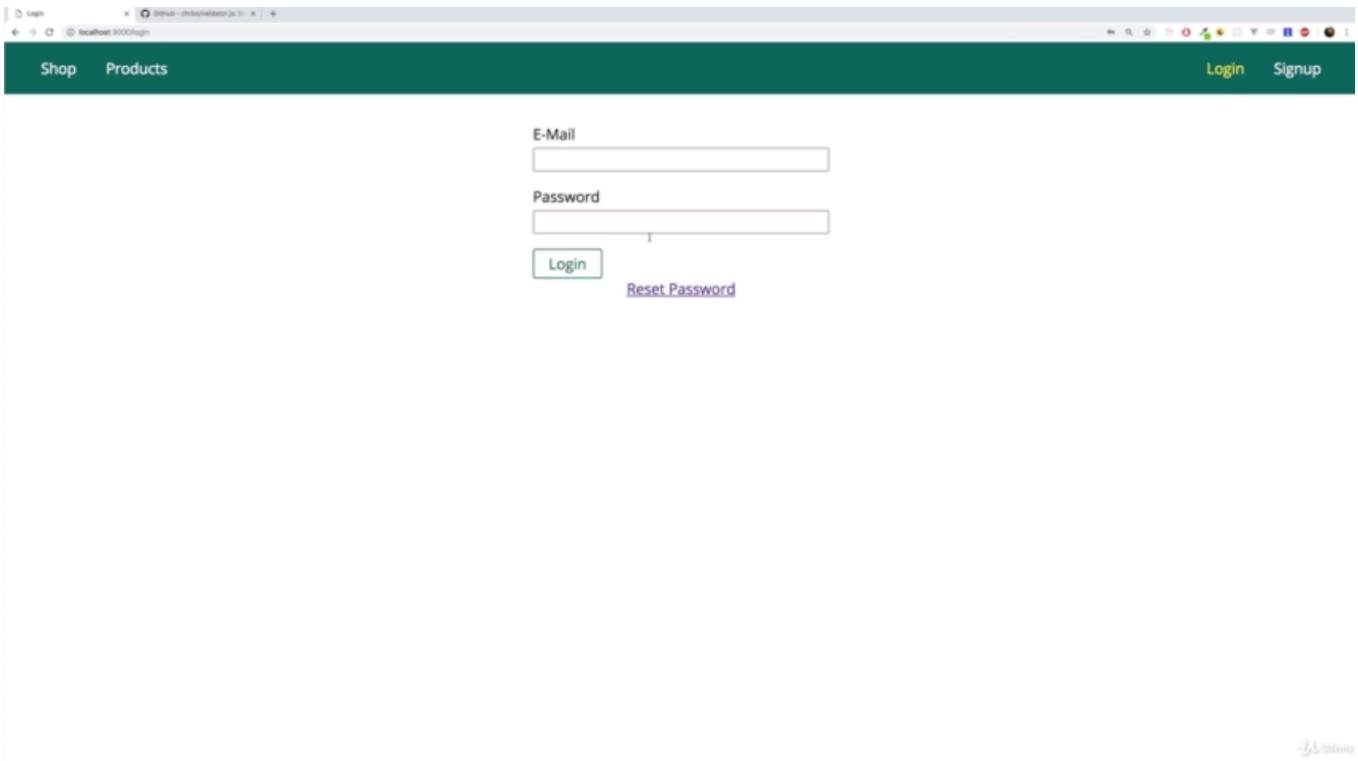
Shop Products Login Signup

E-Mail

Password

Confirm Password

Join today



- even though there's whitespace and uppercase, this can fix this automatically by '.trim()' and 'normalizeEmail()'

```

1 // ./routes/auth.js
2
3 const express = require('express');
4 const { check, body } = require('express-validator/check')
5
6 const authController = require('../controllers/auth');
7 const User = require('../models/user')
8
9 const router = express.Router();
10
11 router.get(
12   '/login',
13   [

```

```
14     body('email')
15         .isEmail()
16         .withMessage('Please enter a valid email address')
17         /**'normalizeEmail()' is built-in sanitizers */
18         .normalizeEmail(),
19     body('password', 'Password has to be valid.')
20         .isLength({ min: 5 })
21         .isAlphanumeric()
22         /**we could trim the password to remove excess whitespace */
23         .trim()
24     ],
25     authController.getLogin)
26
27 router.get('/signup', authController.getSignup);
28
29 router.post('/login', authController.postLogin);
30
31 router.post(
32     '/signup',
33     [
34         check('email')
35             .isEmail()
36             .withMessage('Please enter a valid email')
37             .custom((value, {req}) => {
38                 //if (value === 'test@test.com'){
39                 //    throw new Error('This Email Address is forbidden.')
40                 //}
41                 //return true
42                 return User.findOne({ email: value })
43             .then(userDoc => {
44                 if (userDoc) {
45                     return Promise.reject(
46                         'E-Mail exists already, please pick a different one.'
47                     )
48                 }
49             })
50         })
51         .normalizeEmail(),
52     body(
53         'password',
54         'Please enter a password with only numbers and text and at least 5 characters'
55     )
56         .isLength({min: 5})
57         .isAlphanumeric()
58         .trim(),
59     body('confirmPassword')
60         .trim()
61         .custom((value, {req}) => {
62             if (value !== req.body.password) {
63                 throw new Error('Passwords have to match!')
64             }
65             return true
66         })
67     ],
68     authController.postSignup);
69
```

```
70 router.post('/logout', authController.postLogout);
71
72 router.get('/reset', authController.getReset);
73
74 router.post('/reset', authController.postReset);
75
76 router.get('/reset/:token', authController.getNewPassword);
77
78 router.post('/new-password', authController.postNewPassword);
79
80 module.exports = router;
```

* Chapter 297: Validating Product Addition

1. update
- ./routes/admin.js
- ./controllers/admin.js
- ./views/edit-product.ejs

Add Product

localhost:3000/admin/add-product

Shop Products Cart Orders Add Product Admin Products Logout

Invalid value

Title

Image URL

Price

Description

Add Product

Add Product

localhost:3000/admin/add-product

Shop Products Cart Orders Add Product Admin Products Logout

Invalid value

Title

Image URL

Price

Description

Add Product

John Wiley

John Wiley

Add Product

localhost:3000/admin/add-product

Shop Products Cart Orders Add Product Admin Products Logout

Invalid value

Title
First Book

Image URL

Price

Description

Add Product

Add Product

localhost:3000/admin/edit-product

Shop Products Cart Orders Add Product Admin Products Logout

Invalid value

Title
First Book

Image URL

Price

Description
sd fsdgsgdgsdf

Add Product

John Doe

Add Product

localhost:3000/admin/add-product

Shop Products Cart Orders Add Product Admin Products Logout

Invalid value

Title
First Book

Image URL

Price

Description *

sd fsdgsdgsdfg

Add Product

Add Product

localhost:3000/admin/add-product

Shop Products Cart Orders Add Product Admin Products Logout

Title
First Book

Image URL
 http://ichef.bbci.co.uk/wwfeatures/wm/live

Price
12.99

Description
Does this work?

Add Product

John Wiley

Add Product

localhost:3000/admin/add-product

Shop Products Cart Orders Add Product Admin Products Logout

Invalid value

Title
First Book

Image URL
http://ichef.bbci.co.uk/wwfeatures/wm/live/

Price
12.99

Description
Does this work?

Add Product

My Cluster

Cluster0-shard-0 REPLICA SET 3 NODES MongoDB 3.6.8 Enterprise

shop.products

	DOCUMENTS N/A	TOTAL SIZE N/A	Avg. Size N/A	INDEXES N/A	Total Size N/A	Avg. Size N/A
Documents						
Aggregations						
Explain Plan						
Indexes						

FILTER OPTIONS FIND RESET ...

INSERT DOCUMENT VIEW LIST TABLE

Displaying documents 1 - 2 of 2 < > C

```
_id: ObjectId("5badf72403fd8b5be0366e81")
title: "A Book"
price: 22
description: "This works!"
imageUrl: "http://ichef.bbci.co.uk/wwfeatures/wm/live/1280_640/images/live/p0/2v/..."
userId: ObjectId("5bade42501bea653ba3a74e5")
__v: 0
```

```
_id: ObjectId("5bb20bd1a8630e3768aaf128")
title: "First Book"
price: 12.99
description: "Does this work?"
imageUrl: "http://ichef.bbci.co.uk/wwfeatures/wm/live/1280_640/images/live/p0/2v/..."
userId: ObjectId("5bb209393fa36b3687f778cd")
__v: 0
```

EXPLORER

login.ejs auth.js routes admin.js routes admin.js controllers edit-product.ejs

NODEJS-COMPLETE-GUIDE

- .vscode
- controllers
 - admin.js M
 - auth.js
 - error.js
 - shop.js
- data
- middleware
 - is-auth.js
- models
 - order.js
 - product.js
 - user.js
- node_modules
- public
- routes
 - admin.js M
 - auth.js
 - shop.js
- util
- views
 - admin

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Ln 23, Col 33 Spaces: 2 UTF-8 LF JavaScript Prettier: ✓

```
// ...  
14  
15 exports.postAddProduct = (req, res, next) => {  
16   const title = req.body.title;  
17   const imageUrl = req.body.imageUrl;  
18   const price = req.body.price;  
19   const description = req.body.description;  
20   const errors = validationResult(req);  
21  
22   if (!errors.isEmpty()) {  
23     console.log(errors.array());  
24     return res.status(422).render('admin/edit-product', {  
25       pageTitle: 'Add Product',  
26       path: '/admin/edit-product',  
27       editing: false,  
28       hasError: true,  
29       product: {  
30         title: title,  
31         imageUrl: imageUrl,  
32         price: price.  
(node:14417) 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.  
[ { location: 'body',  
  param: 'title',  
  value: 'First Book',  
  msg: 'Invalid value' } ]
```

17-validation* 0 ▲ 0

EXPLORER

login.ejs auth.js routes admin.js routes admin.js controllers edit-product.ejs

NODEJS-COMPLETE-GUIDE

- .vscode
- controllers
 - admin.js M
 - auth.js
 - error.js
 - shop.js
- data
- middleware
 - is-auth.js
- models
 - order.js
 - product.js
 - user.js
- node_modules
- public
- routes
 - admin.js M
 - auth.js
 - shop.js
- util
- views
 - admin

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Ln 22, Col 18 Spaces: 4 UTF-8 LF JavaScript Prettier: ✓

```
// ...  
10  
11 // /admin/add-product => GET  
12 router.get('/add-product', isAuthenticated, adminController.getAddProduct);  
13  
14 // /admin/products => GET  
15 router.get('/products', isAuthenticated, adminController.getProducts);  
16  
17 // /admin/add-product => POST  
18 router.post(  
19   '/add-product',  
20   [  
21     post(path: PathParams, ...handlers: RequestHandlerParams[]  
22     ): Router  
23     body('title')  
24     .isString()  
25     .isLength({ min: 3 })  
26     .trim(),  
27     body('imageUrl').isURL(),  
28     body('price').isFloat(),  
29     body('description')  
30     .isLength({ min: 5, max: 400 })  
31     .trim()  
32   ],  
33   [  
34     ...  
35     (node:14417) 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.  
[ { location: 'body',  
  param: 'title',  
  value: 'First Book',  
  msg: 'Invalid value' } ]
```

17-validation* 0 ▲ 0

Add Product

localhost:3000/admin/add-product

Shop Products Cart Orders Add Product Admin Products Logout

invalid value

Title
First Book

Image URL
<http://ichef.bbci.co.uk/wwfeatures/wm/live>

Price
12.99

Description
Does this work?

Add Product

Admin Products

localhost:3000/admin/products

Shop Products Cart Orders Add Product Admin Products Logout

First Book

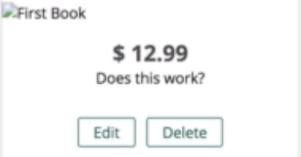


\$ 12.99

Does this work?

Edit Delete

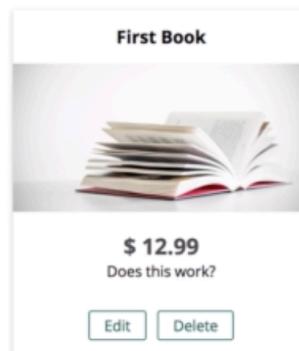
First Book



\$ 12.99

Does this work?

Edit Delete



✓ J. Wiley

Title

Image URL

Price

Description

Add Product

First Book



\$ 12.99
Does this work?

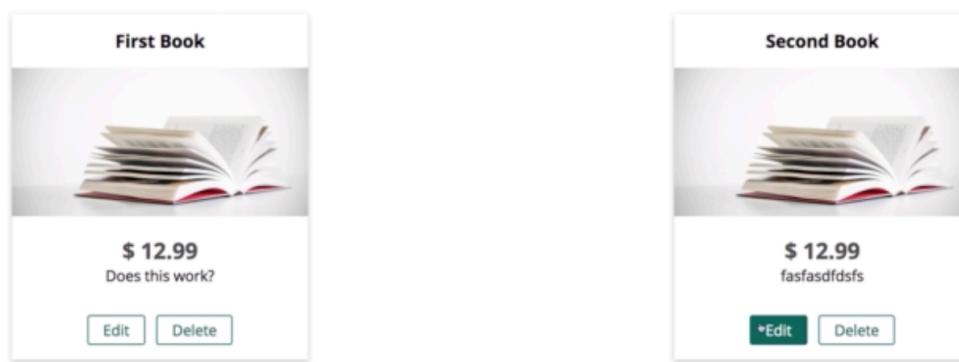
Edit **Delete**

Second Book



\$ 12.99
fasfasdfdsfs

Edit **Delete**



Title

Image URL

Price

Description



↓.simey

```

1 // ./routes/admin.js
2
3 const path = require('path');
4
5 const express = require('express');
6 const { body } = require('express-validator/check')
7
8 const adminController = require('../controllers/admin');
9 const isAuth = require('../middleware/is-auth');
10
11 const router = express.Router();
12
13 // /admin/add-product => GET
14 router.get('/add-product', isAuth, adminController.getAddProduct);
15
16 // /admin/products => GET
17 router.get('/products', isAuth, adminController.getProducts);
18
19 // /admin/add-product => POST
20 router.post('/add-product', [
21     body('title').isString().isLength({ min: 3 }).trim(),
22     body('imageUrl').isURL(),
23     body('price').isFloat(),
24     body('description').isLength({ min: 5, max: 400 }).trim()
25 ],
26     isAuth,
27     adminController.postAddProduct);
28
29 router.get('/edit-product/:productId', isAuth, adminController.getEditProduct);
30
31 router.post('/edit-product',[ 
32     body('title').isAlphanumeric().isLength({ min: 3 }).trim(),
33     body('imageUrl').isURL(),
34     body('price').isFloat(),
35     body('description').isLength({ min: 5, max: 400 }).trim()
36 ]
37 )

```

```
36 ],
37   isAuth,
38   adminController.postEditProduct);
39
40 router.post('/delete-product', isAuth, adminController.postDeleteProduct);
41
42 module.exports = router;
43
44 // ./controllers/admin.js
45
46 const { validationResult } = require('express-validator/check')
47
48 const Product = require('../models/product');
49
50 exports.getAddProduct = (req, res, next) => {
51   res.render('admin/edit-product', {
52     pageTitle: 'Add Product',
53     path: '/admin/add-product',
54     editing: false,
55     hasError: false,
56     errorMessage: null
57   });
58 }
59
60
61 exports.postAddProduct = (req, res, next) => {
62   const title = req.body.title;
63   const imageUrl = req.body.imageUrl;
64   const price = req.body.price;
65   const description = req.body.description;
66   const errors = validationResult(req)
67
68   if(!errors.isEmpty()) {
69     return res.status(422).render('admin/edit-product', {
70       pageTitle: 'Add Product',
71       path: '/admin/edit-product',
72       editing: false,
73       hasError: true,
74       product: {
75         title: title,
76         imageUrl: imageUrl,
77         price: price,
78         description: description
79       },
80       errorMessage: errors.array()[0].msg
81     })
82   }
83
84   const product = new Product({
85     title: title,
86     price: price,
87     description: description,
88     imageUrl: imageUrl,
89     userId: req.user
90   });
91   product
92     .save()
93     .then(result => {
```

```

50     // console.log(result);
51     console.log('Created Product');
52     res.redirect('/admin/products');
53   })
54   .catch(err => {
55     console.log(err);
56   });
57 };
58
59 exports.getEditProduct = (req, res, next) => {
60   const editMode = req.query.edit;
61   if (!editMode) {
62     return res.redirect('/');
63   }
64   const prodId = req.params.productId;
65   Product.findById(prodId)
66     .then(product => {
67       if (!product) {
68         return res.redirect('/');
69       }
70       res.render('admin/edit-product', {
71         pageTitle: 'Edit Product',
72         path: '/admin/edit-product',
73         editing: editMode,
74         product: product,
75         hasError: false,
76         errorMessage: null
77       });
78     })
79     .catch(err => console.log(err));
80 };
81
82 exports.postEditProduct = (req, res, next) => {
83   const prodId = req.body.productId;
84   const updatedTitle = req.body.title;
85   const updatedPrice = req.body.price;
86   const updatedImageUrl = req.body.imageUrl;
87   const updatedDesc = req.body.description;
88
89   Product.findById(prodId)
90     .then(product => {
91       /**
92        * i should convert both to a string
93        * because i'm also checking for type equality.
94       */
95       if(product.userId.toString() !== req.user._id.toString()){
96         return res.redirect('/');
97       }
98       product.title = updatedTitle;
99       product.price = updatedPrice;
100      product.description = updatedDesc;
101      product.imageUrl = updatedImageUrl;
102      return product.save()
103        .then(result => {
104          console.log('UPDATED PRODUCT!');
105          res.redirect('/admin/products');
106        })

```

```

106     })
107     .catch(err => console.log(err));
108 };
109
110 exports.getProducts = (req, res, next) => {
111   Product.find({userId: req.user._id})
112     // .select('title price _id')
113     // .populate('userId', 'name')
114     .then(products => {
115       console.log(products);
116       res.render('admin/products', {
117         prods: products,
118         pageTitle: 'Admin Products',
119         path: '/admin/products'
120       });
121     })
122     .catch(err => console.log(err));
123 };
124
125 exports.postDeleteProduct = (req, res, next) => {
126   const prodId = req.body.productId;
127   Product.deleteOne({_id: prodId, userId: req.user._id})
128     .then(() => {
129       console.log('DESTROYED PRODUCT');
130       res.redirect('/admin/products');
131     })
132     .catch(err => console.log(err));
133 };
134

```

```

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 type="text" name="title" id="title" value="<% if (editing ||
hasError) { %><%= product.title %><% } %>">
19       </div>
20       <div class="form-control">
21         <label for="imageUrl">Image URL</label>
22         <input type="text" name="imageUrl" id="imageUrl" value="<% if (editing ||
hasError) { %><%= product.imageUrl %><% } %>">
23       </div>
24       <div class="form-control">

```

```

25          <label for="price">Price</label>
26          <input type="number" name="price" id="price" step="0.01" value="<% if
27 (editing || hasError) { %><%= product.price %><% } %>">
28      </div>
29      <div class="form-control">
30          <label for="description">Description</label>
31          <textarea name="description" id="description" rows="5"><% if (editing ||
32 hasError) { %><%= product.description %><% } %></textarea>
33      </div>
34      <% if (editing) { %>
35          <input type="hidden" value="<%=_id %>" name="productId">
36      <% } %>
37
38          <input type="hidden" name="_csrf" value="<%=_csrfToken %>">
39          <button class="btn" type="submit"><% if (editing) { %>Update Product<% } else {
40 %>Add Product<% } %></button>
41      </form>
42  </main>
43
44  <%-- include('.../includes/end.ejs') %>

```

* Chapter 298: Validating Product Editing

1. update

- ./controllers/admin.js
- ./views/auth/edit-product.ejs
- ./public/css/forms.css
- ./routes/admin.js

The screenshot shows a web browser window with a green header bar containing navigation links: Shop, Products, Cart, Orders, Add Product, Admin Products, and Logout. The main content area displays a form for editing a product. The form fields are as follows:

- Title:** Second Book!
- Image URL:** http://ichef.bbci.co.uk/wwfeatures/wm/live
- Price:** 12.99
- Description:** fasfasdfdsfs

A red horizontal bar at the top of the form area displays the error message "Invalid value". At the bottom of the form is a green "Update Product" button.

The screenshot shows the VS Code interface with the following details:

- EXPLORER**: Shows the project structure under "NODEJS-COMPLETE-GUIDE" including files like auth.js, admin.js, auth.css, etc.
- CODE EDITOR**: Displays the content of `admin.js`:


```

32 |     adminController.postAddProduct
33 |   );
34 |
35 |   router.get('/edit-product/:productId', isAuthenticated, adminController.getEditProduct);
36 |
37 |   router.post(
38 |     '/edit-product',
39 |     [
40 |       body('title')
41 |         .isString()
42 |         .isLength({ min: 3 })
43 |         .trim(),
44 |       body('imageUrl').isURL(),
45 |       body('price').isFloat(),
46 |       body('description')
47 |         .isLength({ min: 5, max: 400 })
48 |         .trim()
49 |     ],
50 |     isAuthenticated,
51 |     adminController.postEditProduct
52 |

```
- PROBLEMS**: Shows an error message from `eem/index.js:333:33`:


```

eem/index.js:333:33)
at process._tickCallback (internal/process/next_tick.js:61:11)
message:
  'Cast to ObjectId failed for value "" at path "_id" for model "Product",
name: 'CastError',
stringValue: '',
kind: 'ObjectId',

```
- OUTPUT**: Shows the command `1: node`.
- DEBUG CONSOLE**: Shows the command `Ln 41, Col 18`.
- TERMINAL**: Shows the command `Ln 41, Col 18 Spaces: 4 UTF-8 LF JavaScript Prettier: ✓`.

- what's wrong? because the case to object id failed for that value.

The screenshot shows the VS Code interface with the following details:

- EXPLORER**: Shows the project structure under "NODEJS-COMPLETE-GUIDE" including files like auth.css, cart.css, forms.css, main.css, orders.css, product.css, admin.js, auth.js, shop.js, auth.ejs, edit-prod.ejs, products.ejs, login.ejs, new-passwor..., reset.ejs, signup.ejs, and includes/end.ejs.
- CODE EDITOR**: Displays the content of `products.ejs`:


```

42 <div class="form-control">
43   <label for="description">Description</label>
44   <textarea
45     class="<% validationErrors.find(e => e.param === 'description') ? 'invalid' : 'valid'" 
46     name="description"
47     id="description"
48     rows="5"><% if (editing || hasError) { %><%= product.description %><% } %></te
49   </div>
50   <% if (editing) { %>
51     <input type="hidden" value="<%= product._id %>" name="productId">
52   <% } %>
53 
54   <input type="hidden" name="_csrf" value="<%= csrfToken %>">
55   <button class="btn" type="submit"><% if (editing) { %>Update Product<% } else { %>Add I
56   </button>
57 </main>
58 <!-- include('../includes/end.ejs') -->

```
- PROBLEMS**: Shows an error message from `eem/index.js:333:33`:


```

eem/index.js:333:33)
at process._tickCallback (internal/process/next_tick.js:61:11)
message:
  'Cast to ObjectId failed for value "" at path "_id" for model "Product",
name: 'CastError',
stringValue: '',
kind: 'ObjectId',

```
- OUTPUT**: Shows the command `1: node`.
- DEBUG CONSOLE**: Shows the command `Ln 51, Col 17 (65 selected)`.
- TERMINAL**: Shows the command `Ln 51, Col 17 (65 selected) Spaces: 4 UTF-8 LF Django Template`.

- we need that hidden input where i have my product._id. and i load that product._id when i first render this page.

File: admin.js controllers

```

89 const updatedDesc = req.body.description;
90
91 const errors = validationResult(req);
92
93 if (!errors.isEmpty()) {
94   return res.status(422).render('admin/edit-product', {
95     pageTitle: 'Edit Product',
96     path: '/admin/edit-product',
97     editing: true,
98     hasError: true,
99     product: {
100       title: updatedTitle,
101       imageUrl: updatedImageUrl,
102       price: updatedPrice,
103       description: updatedDesc
104     },
105     errorMessage: errors.array()[0].msg,
106     validationErrors: errors.array()
107   });
108 }

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Ln 107, Col 8 (406 selected) Spaces: 2 UTF-8 LF JavaScript Prettier: ✓

- but if i re-rendered it due to a validation error, so due to this render function in my if check here, then i only set title, imageUrl, price, description.

File: edit-product.ejs

```

42 <div class="form-control">
43   <label for="description">Description</label>
44   <textarea
45     class="<%- validationErrors.find(e => e.param === 'description') ? 'invalid' : 'valid'" 
46     name="description"
47     id="description"
48     rows="5"><%- if (editing || hasError) { %><%= product.description %><%- } %</te
49   </div>
50   <% if (editing) { %>
51     <input type="hidden" value="<%- product._id %>" name="productId">
52   <% } %>
53
54   <input type="hidden" name="_csrf" value="<%- csrfToken %>">
55   <button class="btn" type="submit"><%- if (editing) { %>Update Product<% } else { %>Add
56   </form>
57 </main>
58 <!-- include('../includes/end.ejs') -->

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Ln 51, Col 60 (3 selected) Spaces: 4 UTF-8 LF Django Template

Screenshot of VS Code showing the code editor with a file named 'edit-product.ejs'. The code is a template for updating a product. It includes validation logic to check if errors are present and returns a page with the title, path, editing status, error status, product details, and updated title.

```

81     .catch(err => console.log(err));
82   };
83
84   exports.postEditProduct = (req, res, next) => {
85     const productId = req.body.productId;
86     const updatedTitle = req.body.title;
87     const updatedPrice = req.body.price;
88     const updatedImageUrl = req.body.imageUrl;
89     const updatedDesc = req.body.description;
90
91     const errors = validationResult(req);
92
93     if (!errors.isEmpty()) {
94       return res.status(422).render('admin/edit-product', {
95         pageTitle: 'Edit Product',
96         path: '/admin/edit-product',
97         editing: true,
98         hasError: true,
99         product: {
100           title: updatedTitle,

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

eem/index.js:333:33
at process._tickCallback (internal/process/next_tick.js:61:11)
message:
[Cast to ObjectId failed for value "" at path "_id" for model "Product"]
name: 'CastError',
stringValue: '',
kind: 'ObjectId',

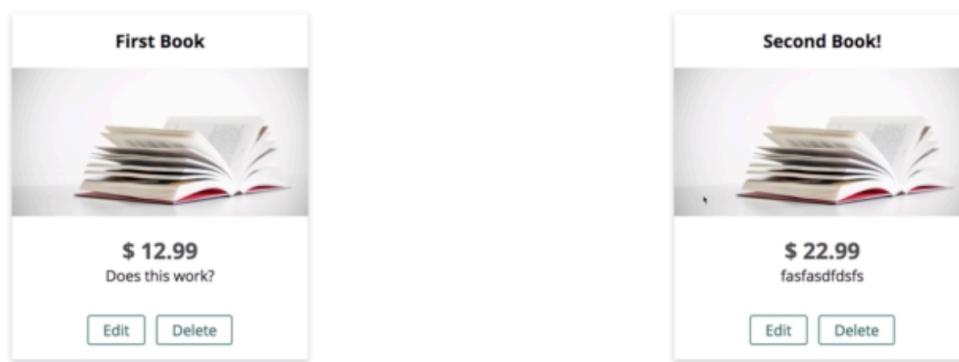
Ln 85, Col 12 (25 selected) Spaces: 2 UTF-8 LF JavaScript Prettier: ✓

- i also need to set _id. otherwise it render the page, but if i try to submit it, that _id will be missing and i need that ID because i extract it here

Screenshot of a web browser showing the 'Edit Product' form. The form fields are:

- Title: Second Book!
- Image URL: http://ichef.bbci.co.uk/wwfeatures/wm/live
- Price: 22.99
- Description: fasfasdfdsfs

The 'Update Product' button is visible at the bottom of the form.



Title

Image URL

Price

Description

Update Product

Shop Products Cart Orders Add Product Admin Products Logout

Invalid value

Title

Image URL
http://ichef.bbci.co.uk/wwfeatures/wm/live

Price
22.99

Description
fasfasdfdsfs

Shop Products Cart Orders Add Product Admin Products Logout

Invalid value

Title
Third Book!

Image URL
http://ichef.bbci.co.uk/wwfeatures/wm/live

Price
22.99

Description
fasfasdfdsfs

John Wiley

by [Liam](#)

```

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

```

```

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

```

```

92 if(!errors.isEmpty()) {
93     return res.status(422).render('admin/edit-product', {
94         pageTitle: 'Edit Product',
95         path: '/admin/edit-product',
96         editing: true,
97         hasError: true,
98         product: {
99             title: updatedTitle,
100            imageUrl: updatedImageUrl,
101            price: updatedPrice,
102            description: updatedDesc,
103            _id: prodId
104        },
105        errorMessage: errors.array()[0].msg,
106        validationErrors: errors.array()
107    })
108 }
109
110
111 Product.findById(prodId)
112     .then(product => {
113         /**i should convert both to a string
114          * because i'm also checking for type equality.
115          */
116         if(product.userId.toString() !== req.user._id.toString()){
117             return res.redirect('/')
118         }
119         product.title = updatedTitle;
120         product.price = updatedPrice;
121         product.description = updatedDesc;
122         product.imageUrl = updatedImageUrl;
123         return product.save()
124             .then(result => {
125                 console.log('UPDATED PRODUCT!');
126                 res.redirect('/admin/products');
127             })
128         })
129     .catch(err => console.log(err));
130 };
131
132 exports.getProducts = (req, res, next) => {
133     Product.find({userId: req.user._id})
134         // .select('title price _id')
135         // .populate('userId', 'name')
136         .then(products => {
137             console.log(products);
138             res.render('admin/products', {
139                 prods: products,
140                 pageTitle: 'Admin Products',
141                 path: '/admin/products'
142             });
143         })
144     .catch(err => console.log(err));
145 };
146
147 exports.postDeleteProduct = (req, res, next) => {

```

```

148 const prodId = req.body.productId;
149 Product.deleteOne({_id: prodId, userId: req.user._id})
150   .then(() => {
151     console.log('DESTROYED PRODUCT');
152     res.redirect('/admin/products');
153   })
154   .catch(err => console.log(err));
155 };
156

```

```

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') ?
48 'invalid' : '' %>">
49             name="description"
50             id="description"
51             rows="5"><% if (editing || hasError) { %><%= product.description %><% } %></textarea>
52     </div>
53     <% if (editing) { %>
54         <input type="hidden" value="<%= product._id %>" name="productId">
55     <% } %>
56
57         <input type="hidden" name="_csrf" value="<%= csrfToken %>">
58         <button class="btn" type="submit"><% if (editing) { %>Update Product<% } else {
59             Add Product<% } %></button>
60     </form>
61 </main>
62 <% include('../includes/end.ejs') %>
```

```

1 /*./public/css/forms.css*/
2
3 .form-control {
4     margin: 1rem 0;
5 }
6
7 .form-control label,
8 .form-control input,
9 .form-control textarea {
10    display: block;
11    width: 100%;
12    margin-bottom: 0.25rem;
13 }
14
15 .form-control input,
16 .form-control textarea {
17    border: 1px solid #a1a1a1;
18    font: inherit;
19    border-radius: 2px;
20 }
21
22 .form-control input:focus,
23 .form-control textarea:focus {
24    outline-color: #00695c;
25 }
26
27 .form-control input.invalid,
28 .form-control textarea.invalid {
29    border-color: red;
30 }
```

```

1 // ./routes/admin.js
2
3 const path = require('path');
4
5 const express = require('express');
```

```
6 const { body } = require('express-validator/check')
7
8 const adminController = require('../controllers/admin');
9 const isAuth = require('../middleware/is-auth');
10
11 const router = express.Router();
12
13 // /admin/add-product => GET
14 router.get('/add-product', isAuth, adminController.getAddProduct);
15
16 // /admin/products => GET
17 router.get('/products', isAuth, adminController.getProducts);
18
19 // /admin/add-product => POST
20 router.post('/add-product', [
21     body('title').isString().isLength({ min: 3 }).trim(),
22     body('imageUrl').isURL(),
23     body('price').isFloat(),
24     body('description').isLength({ min: 5, max: 400 }).trim()
25 ],
26     isAuth,
27     adminController.postAddProduct);
28
29 router.get('/edit-product/:productId', isAuth, adminController.getEditProduct);
30
31 router.post('/edit-product', [
32     body('title').isString().isLength({ min: 3 }).trim(),
33     body('imageUrl').isURL(),
34     body('price').isFloat(),
35     body('description').isLength({ min: 5, max: 400 }).trim()
36 ],
37     isAuth,
38     adminController.postEditProduct);
39
40 router.post('/delete-product', isAuth, adminController.postDeleteProduct);
41
42 module.exports = router;
```