

18. Understanding Validation

* Chapter 284: Module Introduction



What's In This Module?

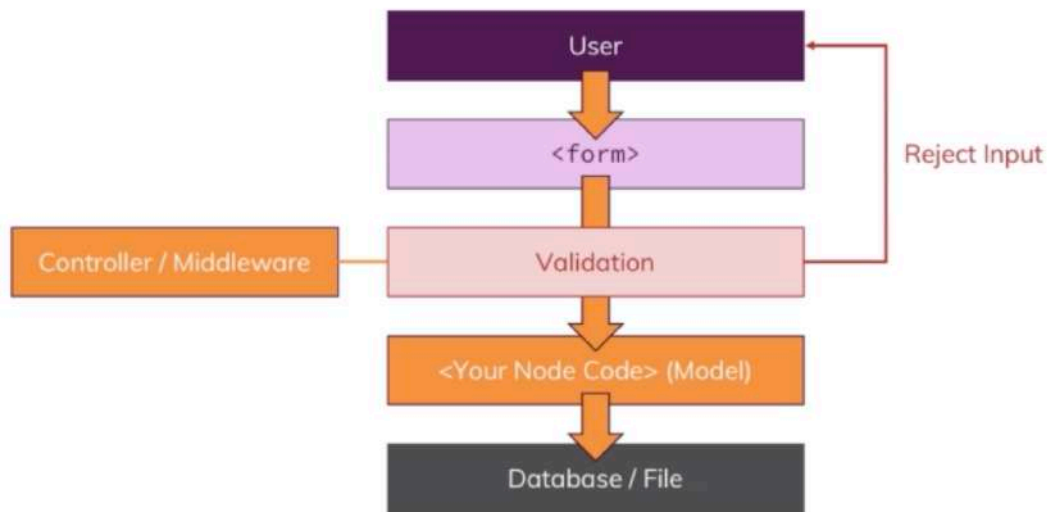
Why Validate?

How to Validate

3/11/2019

* Chapter 285: Why Should We Use Validation?

Why Validate?

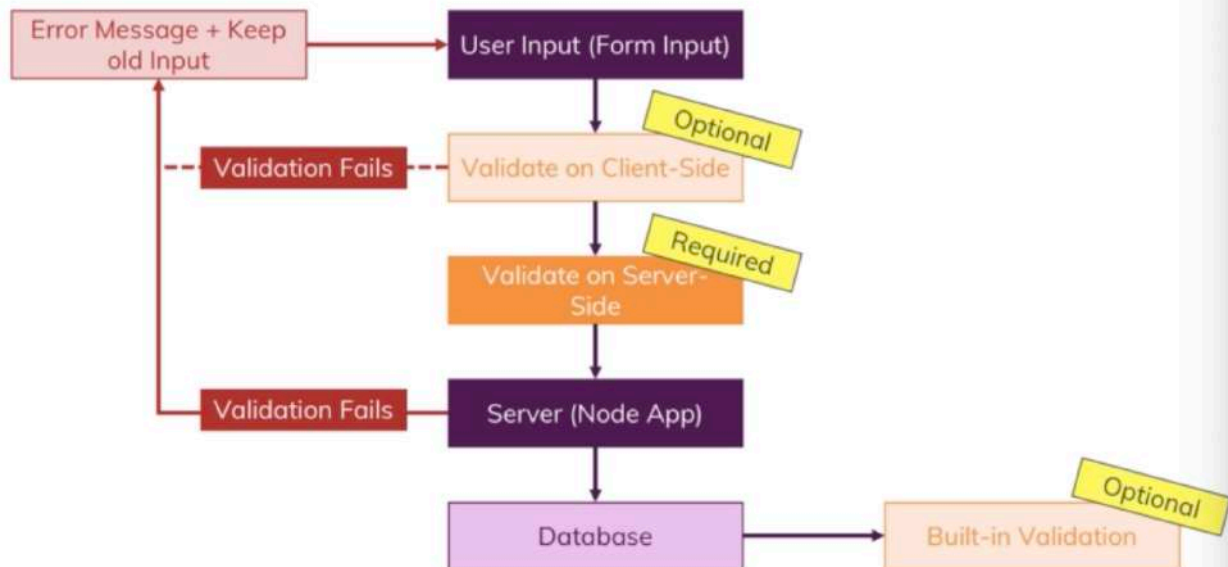


JS

- 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 chagne 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

GitHub repository page for `express-validator / express-validator`. The page shows the repository name, a search bar, and navigation links: Features, Business, Explore, Marketplace, Pricing. It also displays statistics: 57 Watch, 3,314 Star, and 373 Fork. The main navigation tabs are Code, Issues (43), Pull requests, Projects, and 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

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

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

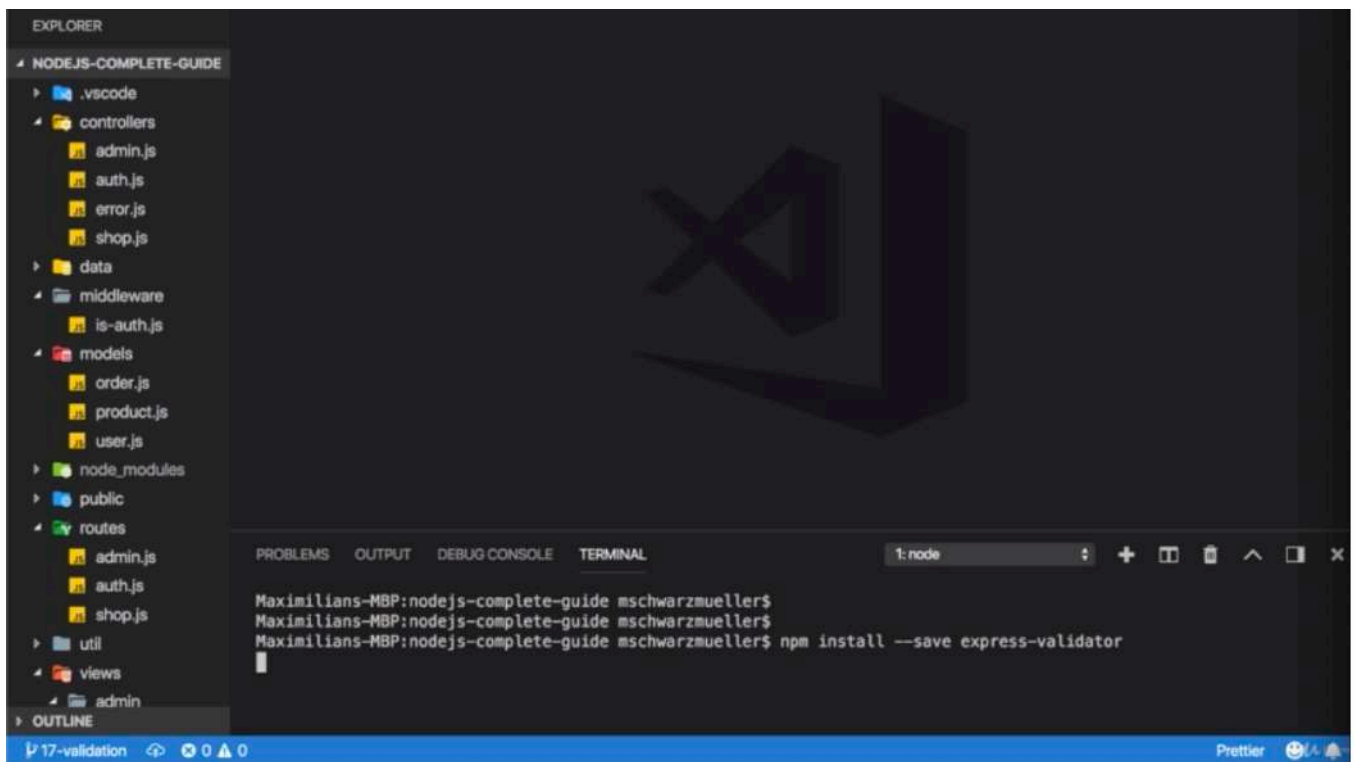
Please refer to the documentation website on <https://express-validator.github.io>.

Changelog

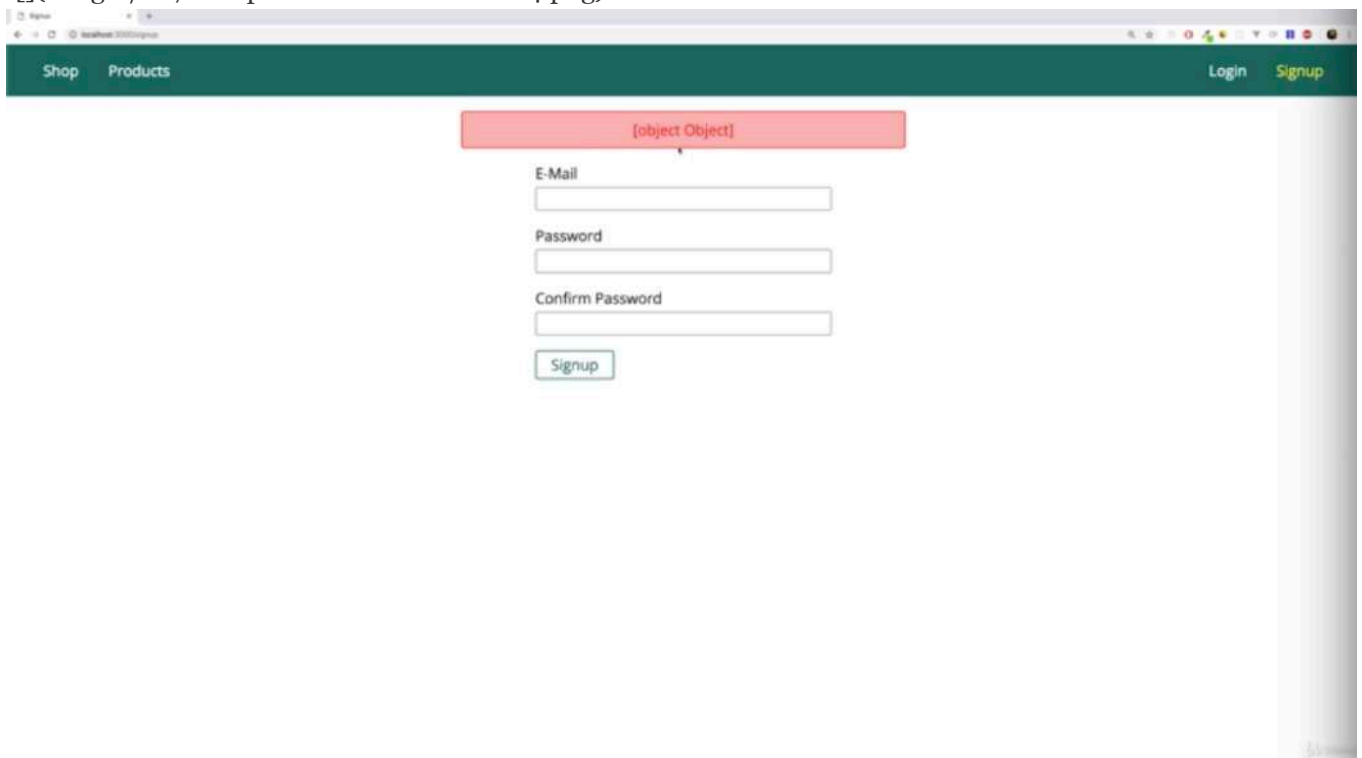
Check the [GitHub Releases page](#).

License

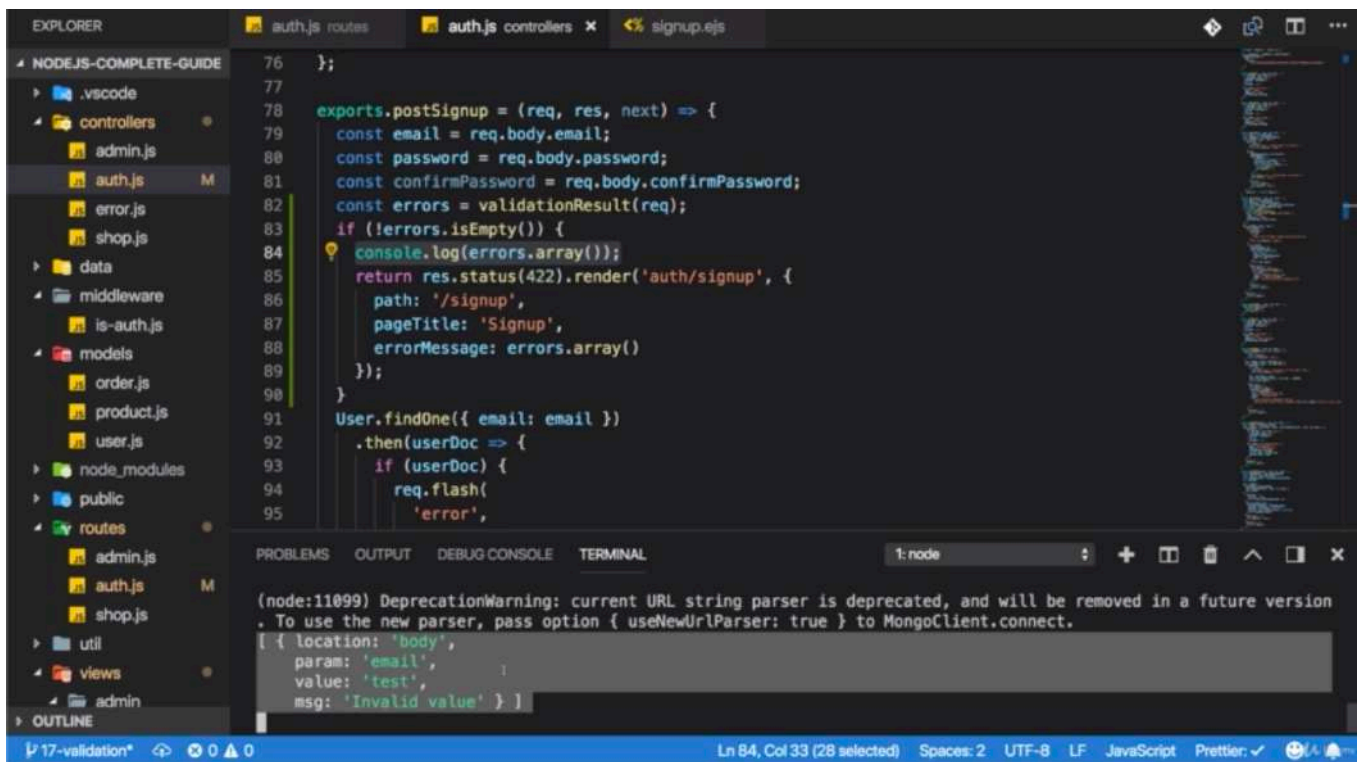
MIT License



- the 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



- 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 use 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 insterested 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 object
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(hashPassword => {
119          const user = new User({
120            email: email,
121            password: hashPassword,
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
191             new password.</p>
192           `
193         });
194       })
195       .catch(err => {
196         console.log(err);
197       });
198   });
199 };
200
201 exports.getNewPassword = (req, res, next) => {
202   const token = req.params.token;
203   User.findOne({
204     resetToken: token,
205     resetTokenExpiration: { $gt: Date.now() }
206   })
207     .then(user => {
208       let message = req.flash('error');
209       if (message.length > 0) {
210         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   /**i still wanna have that token
228    * because otherwise people could start entering random tokens
229    * and still reach that page and then maybe change users on the back
230    * and by entering random userIds and that hidden input field as well.
231    * so i wanna have that token.
232    */
233   const newPassword = req.body.password
234   const userId = req.body.userId
235   const passwordToken = req.body.passwordToken
236   let resetUser
237
238   User.findOne({
239     resetToken: passwordToken,
240     resetTokenExpiration: {$gt: Date.now()},
241     _id: userId
242   })
243     .then(user => {
244       resetUser = user
245       return bcrypt.hash(newPassword, 12)
246     })
247     .then(hashPassword => {
248       resetUser.resetToken = undefined
249       resetUser.resetTokenExpiration = undefined
250       return resetUser.save()
251     })
252     .then(result => {
253       res.redirect('/login')
254     })
255     .catch(err => {
256       console.log(err)
257     })
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

[object Object]

E-Mail

test

Password

Confirm Password

Signup

Invalid value

E-Mail

1

Password

Confirm Password

Signup

Shop Products Login Signup

Invalid value

E-Mail

Password

Confirm Password

Signup

Shop Products Login Signup

Please enter a valid email.

E-Mail

Password

Confirm Password

Signup

```
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               console.log(err);
66               res.redirect('/');
67             });
68           }
69           req.flash('error', 'Invalid email or password. ');
70           res.redirect('/login');

```

```

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(hashPassword => {
105           const user = new User({
106             email: email,
107             password: hashPassword,
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.');
```

new password.</p>

```

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
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(hashPassword => {
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     })

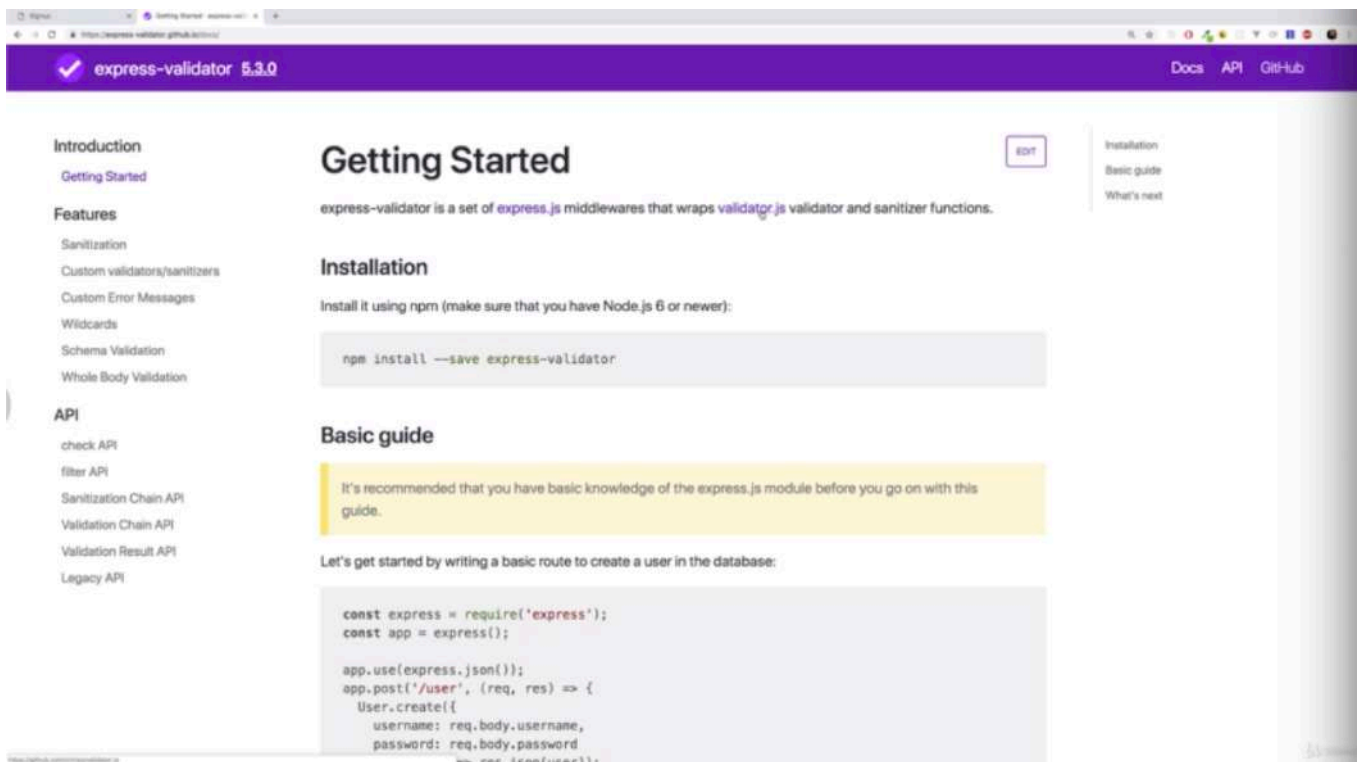
```

238
239 }
240

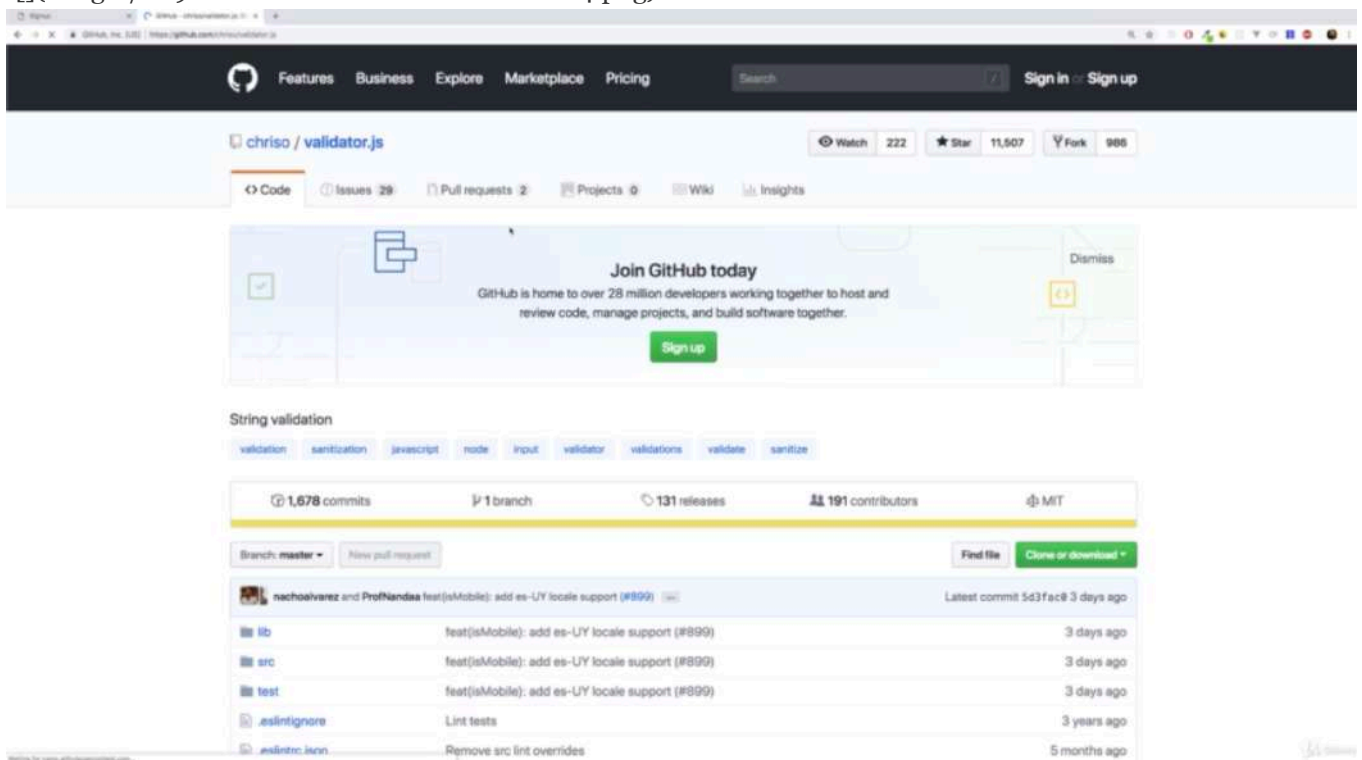
```
1 // ./routes/auth.js
2
3 const express = require('express');
4 const { check } = require('express-validator/check')
5
6 const authController = require('../controllers/auth');
7
8 const router = express.Router();
9
10 router.get('/login', authController.getLogin);
11
12 router.get('/signup', authController.getSignup);
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'),
20   authController.postSignup);
21
22 router.post('/logout', authController.postLogout);
23
24 router.get('/reset', authController.getReset);
25
26 router.post('/reset', authController.postReset);
27
28 router.get('/reset/:token', authController.getNewPassword);
29
30 router.post('/new-password', authController.postNewPassword);
31
32 module.exports = router;
```

* Chapter 289: Built-in & Custom Validators

1. update
- ./routes/auth.js



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



<code>isDecimal(str [, options])</code>	<p>locale determine the decimal separator and is one of ['ar', 'ar-AE', 'ar-BH', 'ar-DZ', 'ar-EG', 'ar-IQ', 'ar-JO', 'ar-KM', '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-HK', 'en-IN', 'en-NZ', 'en-US', 'en-ZA', 'en-ZH', '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', 'sr-RS@latin', 'sv-SE', 'tr-TR', 'uk-UA'].</p> <p>Note: decimal_digits is given as a range like '1,3', a specific value like '3' or min like '1'.</p>
<code>isDivisibleBy(str, number)</code>	check if the string is a number that's divisible by another.
<code>isEmail(str [, options])</code>	<p>check if the string is an email.</p> <p>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.</p>
<code>isEmpty(str [, options])</code>	<p>check if the string has a length of zero.</p> <p>options is an object which defaults to { ignore_whitespace: false }.</p>
<code>isFQDN(str [, options])</code>	<p>check if the string is a fully qualified domain name (e.g. domain.com).</p> <p>options is an object which defaults to { require_tld: true, allow_underscores: false, allow_trailing_dot: false }.</p>
<code>isFloat(str [, options])</code>	check if the string is a float.

<code>isEmail(str [, options])</code>	<p>check if the string is an email.</p> <p>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.</p>
<code>isEmpty(str [, options])</code>	<p>check if the string has a length of zero.</p> <p>options is an object which defaults to { ignore_whitespace: false }.</p>
<code>isFQDN(str [, options])</code>	<p>check if the string is a fully qualified domain name (e.g. domain.com).</p> <p>options is an object which defaults to { require_tld: true, allow_underscores: false, allow_trailing_dot: false }.</p>
<code>isFloat(str [, options])</code>	<p>check if the string is a float.</p> <p>options is an object which can contain the keys min, max, gt, and/or lt to validate the float is within boundaries (e.g. { min: 7.22, max: 9.55 }) it also has locale as an option.</p> <p>min and max are equivalent to 'greater or equal' and 'less or equal', respectively while gt and lt are their strict counterparts.</p> <p>locale determine the decimal separator and is one of ['ar', 'ar-AE', 'ar-BH', 'ar-DZ', 'ar-EG', 'ar-IQ', 'ar-JO', 'ar-KM', 'ar-LB', 'ar-LY', 'ar-MA', 'ar-QA', 'ar-QM',</p>

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

Please enter a valid email.

E-Mail
test@test.com

Password

Confirm Password

Signup

This email address is forbidden.

E-Mail

Password

Confirm Password

Signup

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

This email address is forbidden.

E-Mail

Password

Confirm Password

Signup

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

E-Mail

Password

Confirm Password

Signup

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

E-Mail
test33@test.com

Password

Confirm Password

Signup

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

E-Mail

Password

Login

[Reset Password](#)

E-Mail

tes444t@test.com

Password

Confirm Password

Signup

E-Mail

Password

Login

[Reset Password](#)

Shop Products Login Signup

E-Mail
test22222@test.com

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

```
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);
```

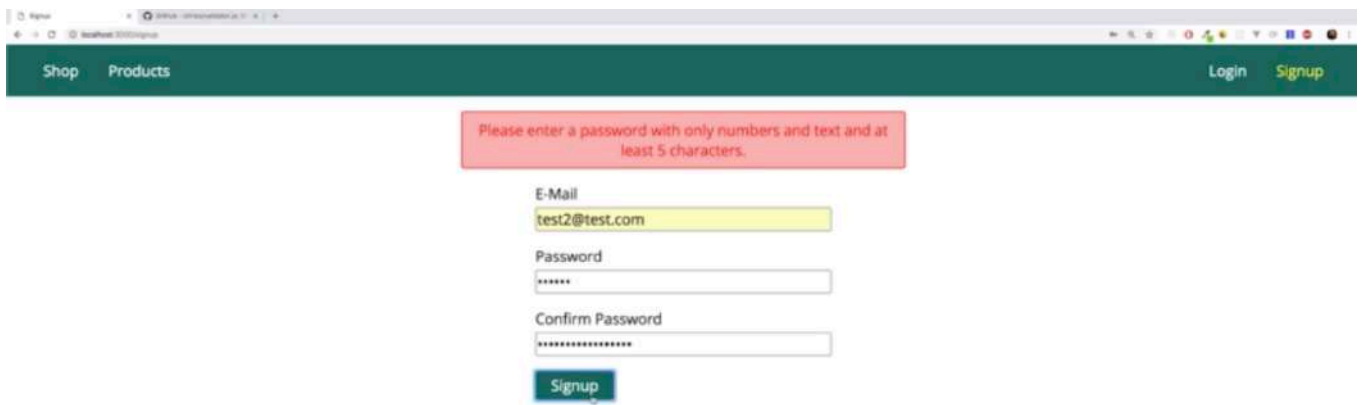
```

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;

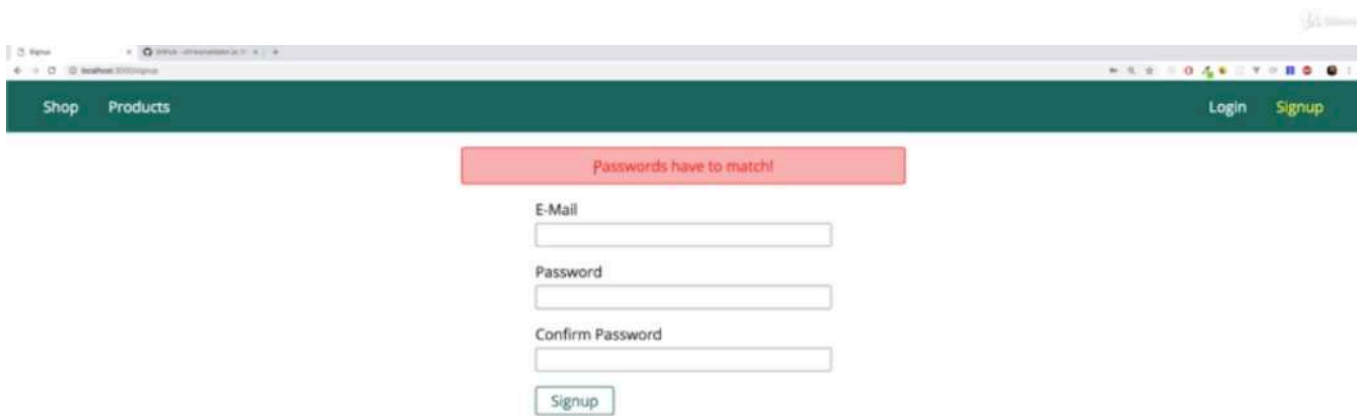
```

* Chapter 291: Checking For Field Equality

1. update
- ./routes/auth.js



The screenshot shows a web browser window with a dark green header containing 'Shop' and 'Products' links on the left, and 'Login' and 'Signup' links on the right. The main content area features a red error message box at the top that reads: 'Please enter a password with only numbers and text and at least 5 characters.' Below this, the signup form includes three input fields: 'E-Mail' with the value 'test2@test.com', 'Password' with masked characters '*****', and 'Confirm Password' with masked characters '*****'. A blue 'Signup' button is positioned at the bottom of the form.



The screenshot shows the same web browser window as above, but with a different error message. The red error message box now reads: 'Passwords have to match!'. The 'E-Mail' field is empty. The 'Password' and 'Confirm Password' fields are also empty, and the 'Signup' button is now a light blue button.

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

A screenshot of a web browser displaying a signup form. The browser's address bar shows 'localhost:3000/signup'. The page has a dark green header with 'Shop' and 'Products' links on the left, and 'Login' and 'Signup' links on the right. A red error message box at the top center reads 'Passwords have to match!'. Below this, the form contains three input fields: 'E-Mail' with the value 'test2@test.com', 'Password' with masked characters '*****', and 'Confirm Password' also with masked characters '*****'. A 'Signup' button is located at the bottom of the form.

A screenshot of a web browser displaying the same signup form. The browser's address bar shows 'localhost:3000/signup'. The page has a dark green header with 'Shop' and 'Products' links on the left, and 'Login' and 'Signup' links on the right. A red error message box at the top center reads 'E-Mail exists already, please pick a different one.'. Below this, the form contains three empty input fields: 'E-Mail', 'Password', and 'Confirm Password'. A 'Signup' button is located at the bottom of the form.

- '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

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

E-Mail
test2@test.com

Password

Confirm Password

Signup

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

E-Mail

Password

Confirm Password

Signup



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

E-Mail

Password

Confirm Password



E-Mail

Password

[Reset Password](#)

```
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               console.log(err);
66               res.redirect('/');
67             });
68           }
69           req.flash('error', 'Invalid email or password. ');
70           res.redirect('/login');

```

```

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(hashPassword => {
95         const user = new User({
96             email: email,
97             password: hashPassword,
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
163             new password.</p>
164           `
165         });
166       })
167       .catch(err => {
168         console.log(err);
169       });
170   });
171 };
172
173 exports.getNewPassword = (req, res, next) => {
174   const token = req.params.token;
175   User.findOne({
176     resetToken: token,
177     resetTokenExpiration: { $gt: Date.now() }
178   })
179     .then(user => {
180       let message = req.flash('error');
181       if (message.length > 0) {
182         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(hashPassword => {
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
185             new password.</p>
186           `
187         });
188       })
189       .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(hashPassword => {
233             resetUser.password = hashPassword;
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 };

```

```

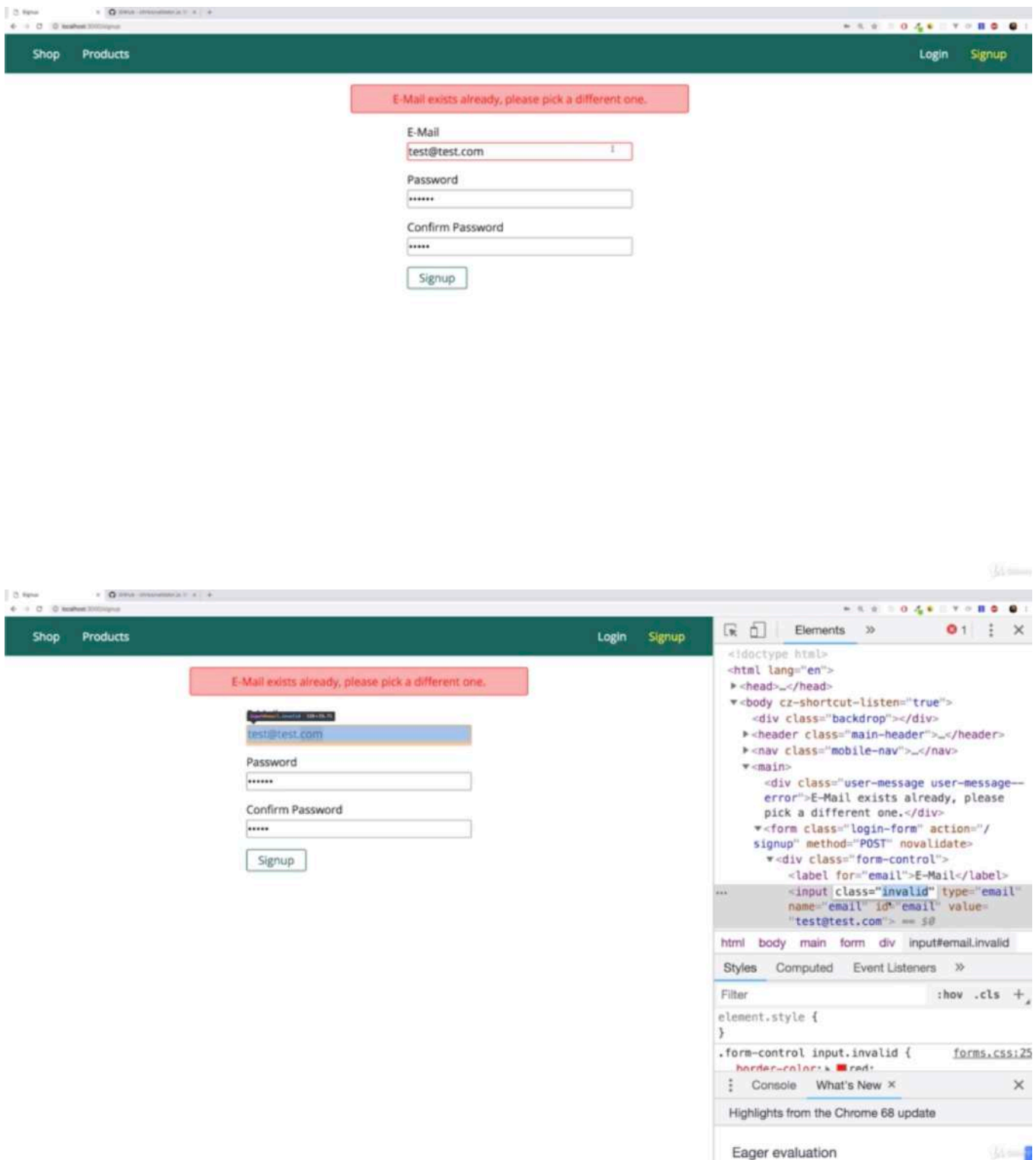
1 <!--./views/auth/signup.ejs-->
2
3 <%- include('../includes/head.ejs') %>
4   <link rel="stylesheet" href="/css/forms.css">
5   <link rel="stylesheet" href="/css/auth.css">
6 </head>
7
8 <body>
9   <%- include('../includes/navigation.ejs') %>
10
11   <main>
12     <% if (errorMessage) { %>
13       <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') %>

```

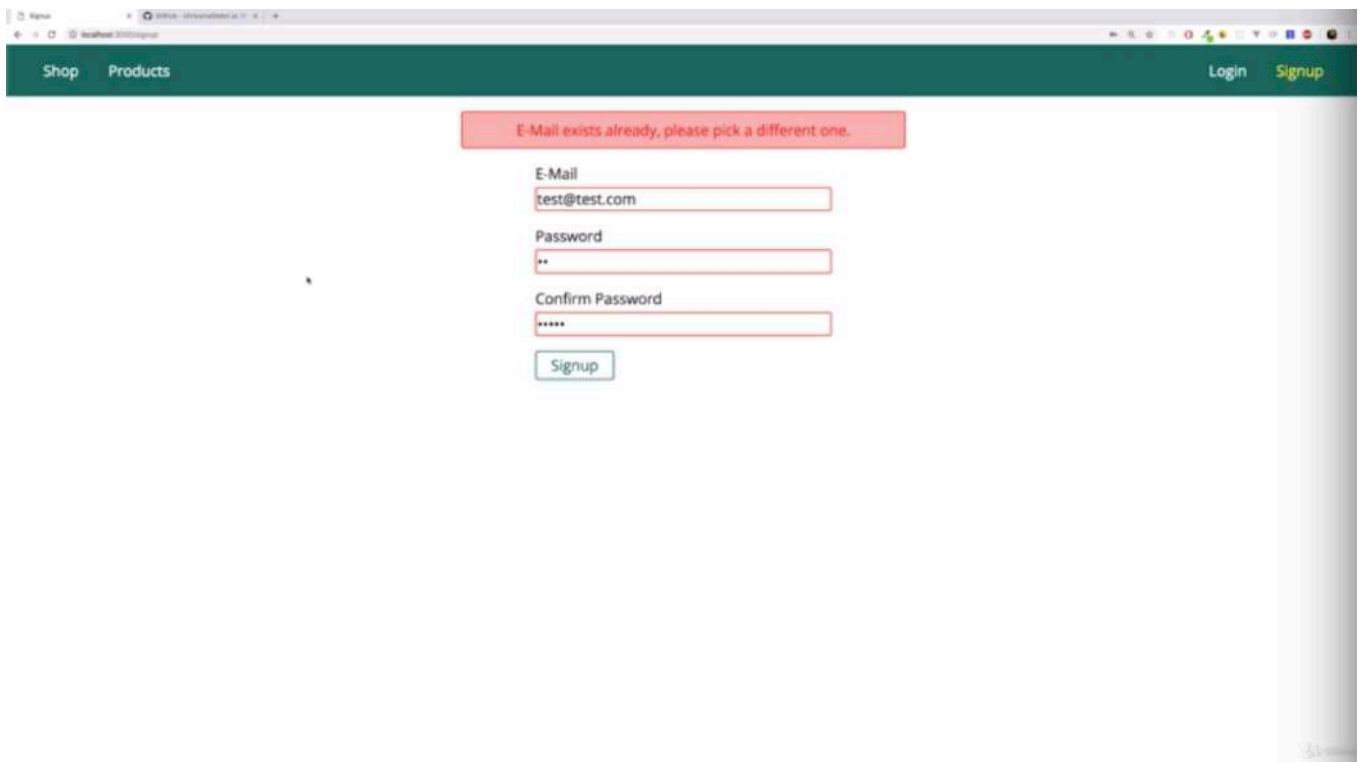
* Chapter 294: Adding Conditional CSS Classes

1. update

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



- 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     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        */
145       validationErrors: errors.array()
146     });

```

```

147 }
148
149 bcrypt
150   .hash(password, 12)
151   .then(hashPassword => {
152     const user = new User({
153       email: email,
154       password: hashPassword,
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
new password.</p>
220         `
221     });
222 }
223 .catch(err => {
224     console.log(err);
225 });
226 });
227 };
228
229 exports.getNewPassword = (req, res, next) => {
230     const token = req.params.token;
231     User.findOne({ resetToken: token, resetTokenExpiration: { $gt: Date.now() } })
232     .then(user => {
233         let message = req.flash('error');
```

234 if (message.length > 0) {

235 message = message[0];

236 } else {

237 message = null;

238 }

```

239         res.render('auth/new-password', {
240             path: '/new-password',
241             pageTitle: 'New Password',
242             errorMessage: message,
243             userId: user._id.toString(),
244             passwordToken: token
245         });
246     })
247     .catch(err => {
248         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;
257
```

```

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(hashPassword => {
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           '' %>"
21           type="email"
22           name="email"
23           id="email"
24           value="<%= oldInput.email %>">
25       </div>
26       <div class="form-control">
27         <label for="password">Password</label>
28         <input
29           class="<%= validationErrors.find(e => e.param === 'password') ?
30           'invalid' : '' %>"
31           type="password"
32           name="password"
33           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') ?
'invalid' : '' %>"
38             type="password"
39             name="confirmPassword"
40             id="confirmPassword"
41             value="<%= oldInput.confirmPassword %>">
42     </div>
43     <input type="hidden" name="_csrf" value="<%= csrfToken %>">
44     <button class="btn" type="submit">Signup</button>
45 </form>
46 </main>
47 <%- 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



Password has to be valid.

E-Mail
test@test.com

Password

Login

[Reset Password](#)

Invalid email or password.

E-Mail
test@test.com

Password

Login

[Reset Password](#)

Invalid email or password.

E-Mail
test@test.com

Password

Login

[Reset Password](#)

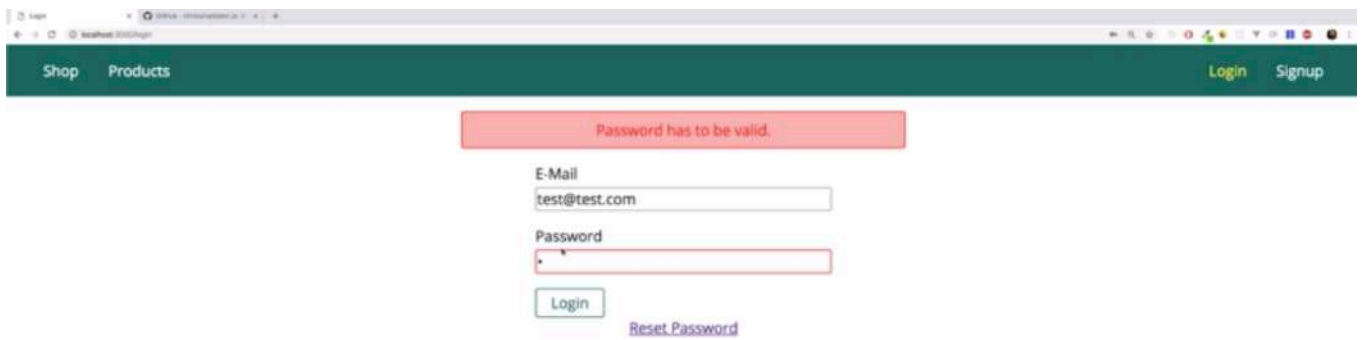
Invalid email or password.

E-Mail
test@test.com

Password
*

Login

[Reset Password](#)



Full Stack

```
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(hashPassword => {
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(hashPassword => {
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           " %>"
21           type="email"
22           name="email"
23           id="email"
24           value="<%= oldInput.email %>">
25       </div>
26       <div class="form-control">
27         <label for="password">Password</label>
28         <input
29           class="<%= validationErrors.find(e => e.param === 'password') ?
'invalid' : ' ' %>"
           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 the 'Sanitizers' section of the Express.js documentation. It lists several sanitizers and their descriptions:

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, '\\[\\]')</code> .
<code>escape(input)</code>	replace <code><</code> , <code>></code> , <code>&</code> , <code>'</code> , <code>"</code> and <code>/</code> with HTML entities.
<code>unescape(input)</code>	replaces HTML encoded entities with <code><</code> , <code>></code> , <code>&</code> , <code>'</code> , <code>"</code> and <code>/</code> .
<code>ltrim(input [, chars])</code>	trim characters from the left-side of the input.
	<p>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)</p> <p><code>options</code> is an object with the following keys and default values:</p> <ul style="list-style-type: none"> <code>all_lowercase: true</code> - Transforms the local part (before the <code>@</code> 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. <code>gmail_lowercase: true</code> - GMail addresses are known to be case-insensitive, so this switch allows lowercasing them even when <code>all_lowercase</code> is set to false. Please note that when <code>all_lowercase</code> is true, GMail addresses are lowercased regardless of the value of this setting. <code>gmail_remove_dots: true</code> - Removes dots from the local part of the email address, as GMail ignores them (e.g. "john.doe" and "johndoe" are considered equal). <code>gmail_remove_subaddress: true</code> - Normalizes addresses by removing "sub-addresses", which is the part following a "+" sign (e.g. "foo+bar@gmail.com" becomes "foo@gmail.com"). <code>gmail_convert_googlemaildotcom: true</code> - Converts addresses with domain <code>@googlemail.com</code> to <code>@gmail.com</code>, 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 "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_gmailmaildotcom**: true: Converts addresses with domain @gmail.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.

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

shop.users 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

INSERT DOCUMENT VIEW LIST TABLE

Displaying documents 1 - 6 of 6

```

> cart: Object
  email: "test33@test.com"
  password: "$2a$12$5v0jCwHvoXN3xzzw9Pls7F.85FTMzqfN.1QEHNuKRIWg18I1h0wTK"
  __v: 0

  _id: ObjectId("5bb1e279cdc0732d672c33b5")
> cart: Object
  email: "tes444t@test.com"
  password: "$2a$12$z/EK8XzdyAY8xHu4KKkJZ09ICY.v/2TvUb8zbqctAPRPv4AgnZRG"
  __v: 0

  _id: ObjectId("5bb1e4a39e22c32e21f0c0a5")
> cart: Object
  email: "test2121@test.com"
  password: "$2a$12$2t1VV/cZvB0QzNrkXe4frk.11MeKv0XeA100ZaRqTTN3kAatKHg2P6"
  __v: 0

  _id: ObjectId("5bb1e4c99e22c32e21f0c0a6")
> cart: Object
  email: "test2111@test.com"
  password: "$2a$12$5xrcqik.A0PdZ7yteA8sH.LCGecR8o7s9N84XSduH0SRef5aVv31C"
  __v: 0
  
```

Document Flagged For Deletion CANCEL DELETE

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

shop.users 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

INSERT DOCUMENT VIEW LIST TABLE

Displaying documents 1 - 4 of 4

```
> cart: Object
  email: "test2@test.com"
  password: "$2a$12$5p8X2ME/TXuPTXN0Z5YjxuzuPay.FqTwePVP52dX32pMR4c2Cje3G"
  __v: 1

  _id: ObjectId("5bade42501bea653ba3a74e5")
> cart: Object
  email: "test@test.com"
  password: "$2a$12$5M3be9a.1spdevprN6icw.Anp5/Nv/EVsrdFsUR7j80w3C2hRb0vW"
  __v: 0

  _id: ObjectId("5bb1e26dc0732d672c33b4")
> cart: Object
  email: "test33@test.com"
  password: "$2a$12$5v0jchRvoN3xz2v9PLs7F.B6FTMzqfh.IQEHNuKrI0Wg1B1Ih0wTK"
  __v: 0

  _id: ObjectId("5bb1e4a39e22c32e21f0c8a5")
> cart: Object
  email: "test2121@test.com"
  password: "$2a$12$2t1VV/cZvBdQdNrXe4frk.1lMeKvRxeA100ZaRqTTW3kAatKHg2P6"
  __v: 0
```

Document Deleted. CANCEL DELETE

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

shop.users 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

INSERT DOCUMENT VIEW LIST TABLE

Displaying documents 1 - 0 of 0

E-Mail

Test@test.com

test@test.com

test@test.com

Confirm Password

Signup

E-Mail

Test@test.com

Password

Confirm Password

Signup

Shop Products Login Signup

E-Mail

Password

Login Reset Password

My Cluster Cluster0-shard-0 REPLICASET 3 NODES MongoDB 3.6.8 Enterprise

shop.users 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

INSERT DOCUMENT VIEW LIST TABLE

Displaying documents 1 - 1 of 1

```
{
  "_id": ObjectId("5bb269393fa36b36877778cd"),
  "cart": Object,
  "email": "test@test.com",
  "password": "$2a$12$3o67HvuJG5y5CvZaF3ghUeqavjVe/XjRXGjdsxntuXBjcsyexU2r6",
  "__v": 0
}
```

- 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

Invalid value

Title

Image URL

Price

Description

Add Product

Invalid value

Title

Image URL

Price

Description

Add Product

Invalid value

Title

First Book

Image URL

Price

Description

Add Product

Invalid value

Title

First Book

Image URL

Price

Description

sd fsdgsdgsdf

Add Product

Invalid value

Title
First Book

Image URL

Price

Description
sd fsgdsdgsdfg

Add Product

Title
First Book

Image URL
<http://ichef.bbc.co.uk/ww/features/wm/live>

Price
12.99

Description
Does this work?

Add Product

ShopProductsCartOrdersAdd ProductAdmin ProductsLogout

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 REPLICASET 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

<div>></div>	<div><div><div><div>_id: ObjectId("5badf72403fd8b5be8366e81")</div><div>title: "A Book"</div><div>price: 22</div><div>description: "This works!"</div><div>imageUrl: "http://ichef.bbci.co.uk/wwfeatures/wm/live/1280_640/images/live/p0/2v/..."</div><div>userId: ObjectId("5bade42501bea653ba3a74e5")</div><div>__v: 0</div></div></div></div>	<div><div><div><div></div><div></div><div></div><div></div></div></div></div>
	<div><div><div><div>_id: ObjectId("5bb20bd1a0630e1768aaf128")</div><div>title: "First Book"</div><div>price: 12.99</div><div>description: "Does this work?"</div><div>imageUrl: "http://ichef.bbci.co.uk/wwfeatures/wm/live/1280_640/images/live/p0/2v/..."</div><div>userId: ObjectId("5bb209393fa36b3687f778cd")</div><div>__v: 0</div></div></div></div>	

```
12 111;
13 12};
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.
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1: node

(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' } ]
```

```
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     body('title')
22       .isString()
23       .isLength({ min: 3 })
24       .trim(),
25     body('imageUrl').isURL(),
26     body('price').isFloat(),
27     body('description')
28       .isLength({ min: 5, max: 400 })
29       .trim()
30   ],
31   adminController.postAddProduct
32 );
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1: node

(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' } ]
```

Invalid value

Title
First Book


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

Price
12,99

Description
Does this work?

Add Product


First Book



\$ 12.99
Does this work?

Edit Delete


First Book



\$ 12.99
Does this work?

Edit Delete

First Book



\$ 12.99

Does this work?

EditDelete

First Book




\$ 12.99

Does this work?

EditDelete

First Book



\$ 12.99

Does this work?

EditDelete

Title

Second Book

Image URL

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

Price

12.99

Description

fasfasdfdsfs

Add Product

First Book



\$ 12.99

Does this work?

EditDelete

Second Book




\$ 12.99

fasfasdfdsfs

EditDelete

First Book



\$ 12.99
Does this work?

EditDelete

Second Book



\$ 12.99
fasfasdfdsfs

EditDelete

Title
Second Book

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

Price
12.99

Description
fasfasdfdsfs

Update Product



John Doe

```

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     isAuth,
38     adminController.postEditProduct);
39
40 router.post('/delete-product', isAuth, adminController.postDeleteProduct);
41
42 module.exports = router;

```

```

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: title,
42         price: price,
43         description: description,
44         imageUrl: 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: 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             /**i should convert both to a string
92              * because i'm also checking for type equality.
93              */
94             if (product.userId.toString() !== req.user._id.toString()){
95                 return res.redirect('/')
96             }
97             product.title = updatedTitle;
98             product.price = updatedPrice;
99             product.description = updatedDesc;
100             product.imageUrl = updatedImageUrl;
101             return product.save()
102                 .then(result => {
103                     console.log('UPDATED PRODUCT!');
104                     res.redirect('/admin/products');
105                 })

```

```

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

```

```

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

```

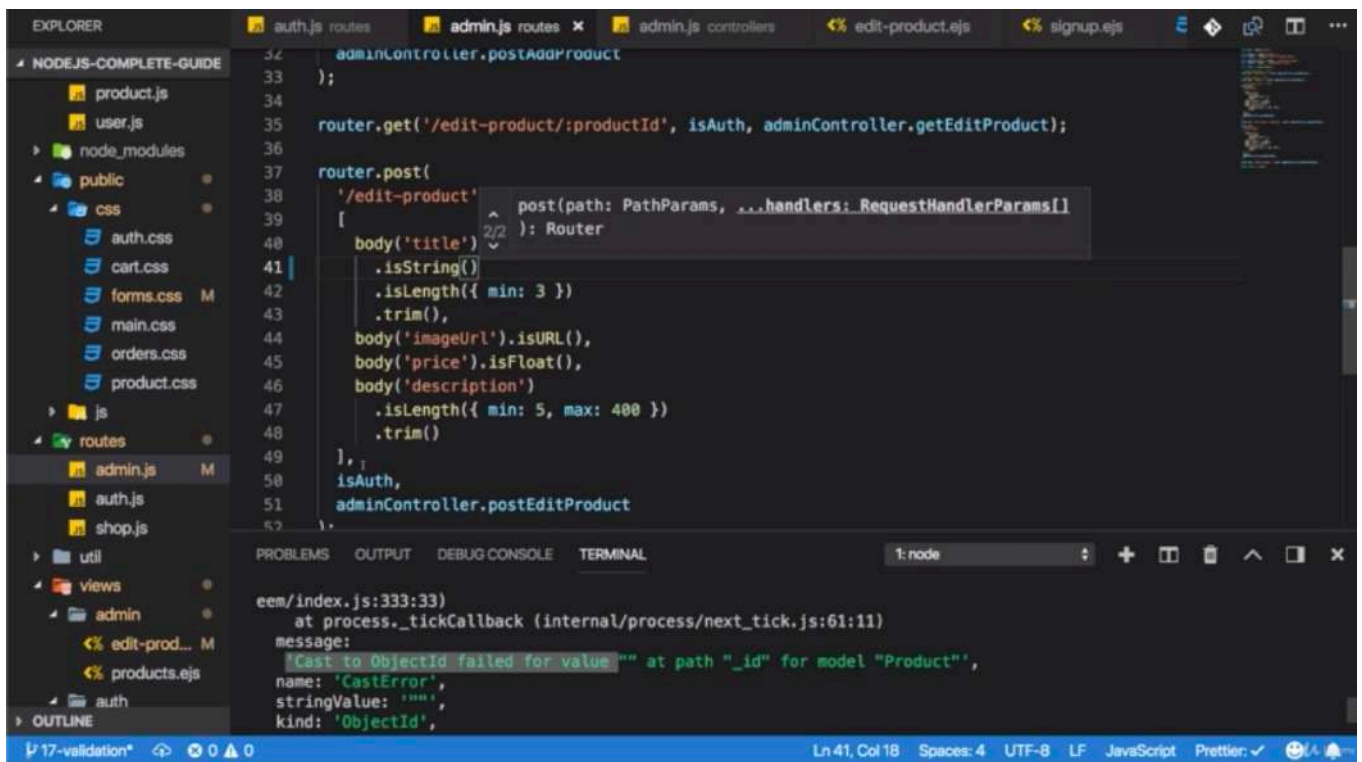
* Chapter 298: Validating Product Editing

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

The screenshot shows a web browser window displaying a product editing form. At the top, there is a navigation bar with links: Shop, Products, Cart, Orders, Add Product, Admin Products, and Logout. Below the navigation bar, a red error message "Invalid value" is displayed. The form contains the following fields:

- Title: Second Book!
- Image URL: http://ichef.bbci.co.uk/ww/features/wm/live
- Price: 12.99
- Description: fasfasdfsfs

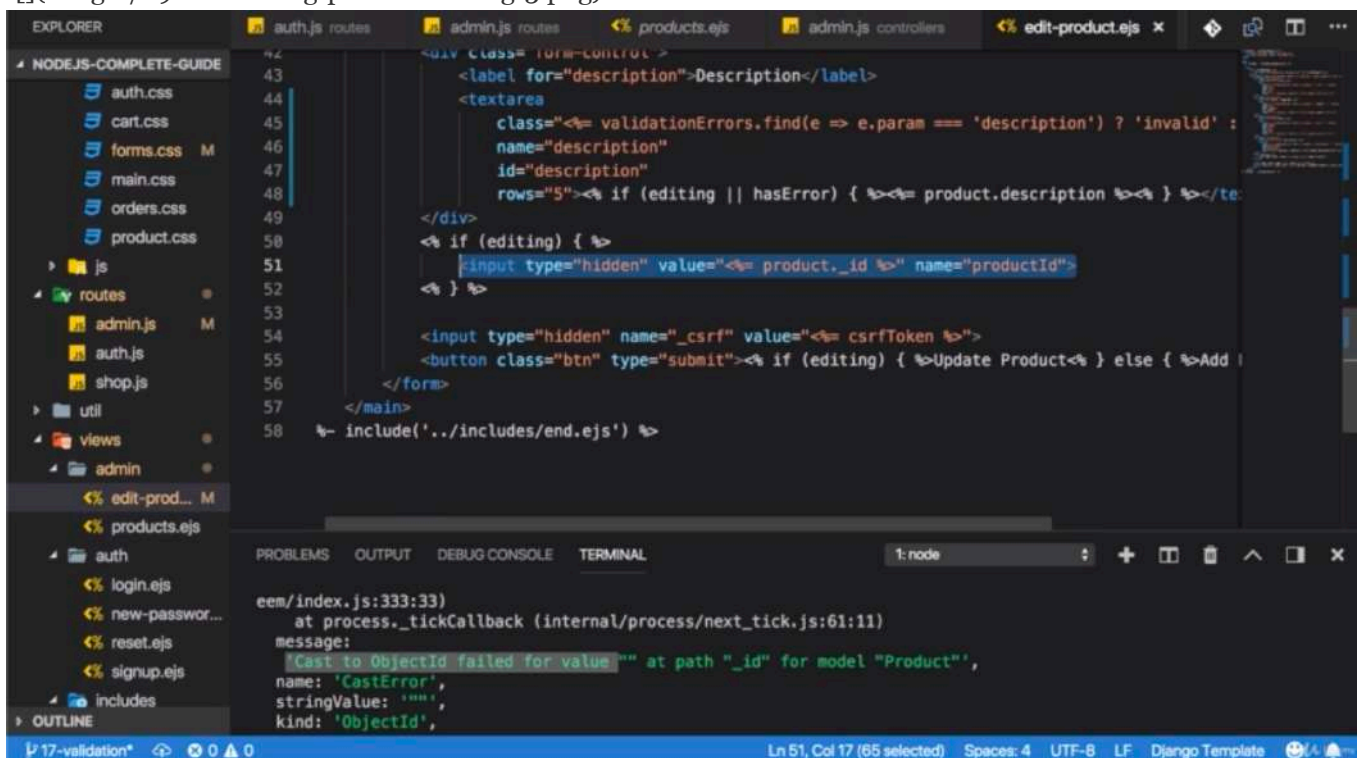
At the bottom of the form, there is a green "Update Product" button.



```
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 );
```

message:
Cast to ObjectId failed for value "" at path "_id" for model "Product",
name: 'CastError',
stringValue: '""',
kind: 'ObjectId',

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



```
43 <div class="form-control">
44   <label for="description">Description</label>
45   <textarea
46     class="form-control"
47     name="description"
48     id="description"
49     rows="5"><%= if (editing || hasError) { %><%= product.description %></te
50   </div>
51   <%= if (editing) { %>
52     <input type="hidden" value="<%= product._id %>" name="productId">
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') %>
```

message:
Cast to ObjectId failed for value "" at path "_id" for model "Product",
name: 'CastError',
stringValue: '""',
kind: 'ObjectId',

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


```
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

1: node

eeem/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",

- 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.


```
42 <div class="form-control">
43   <label for="description">Description</label>
44   <textarea
45     class="<%= validationErrors.find(e => e.param === 'description') ? 'invalid' :
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

1: node

eeem/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",


```
80 //
81 .catch(err => console.log(err));
82 };
83
84 exports.postEditProduct = (req, res, next) => {
85   const prodId = 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
1: node
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',
```

- 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



Title
Second Book!


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

Price
22.99

Description
fasfasdfdsfs

Update Product


First Book



\$ 12.99
Does this work?

EditDelete

Second Book!



\$ 22.99
fasfasdfdsfs

EditDelete

Title

Image URL

Price

Description

Update Product

Invalid value

Title

Image URL

Price

Description

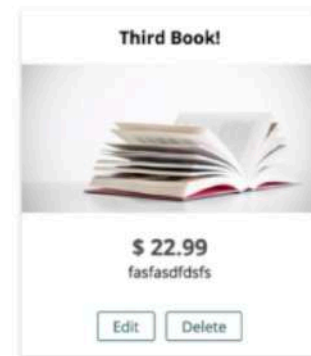
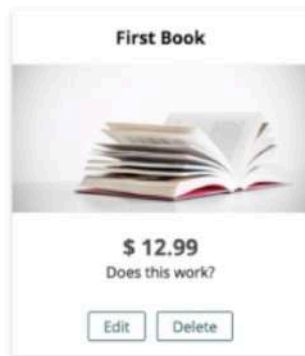
Invalid value

Title

Image URL

Price

Description



Full Stack

```

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: title,
42   price: price,
43   description: description,
44   imageUrl: 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: 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
16 { %>add-product<% } %>" method="POST">
17       <div class="form-control">
18         <label for="title">Title</label>
19         <input
20           class="<%= validationErrors.find(e => e.param === 'title') ? 'invalid' :
21           '' %>"
22           type="text"
23           name="title"
24           id="title"
25           value="<% if (editing || hasError) { %><%= product.title %><% } %>">
26       </div>
27       <div class="form-control">
28         <label for="imageUrl">Image URL</label>
29         <input
30           class="<%= validationErrors.find(e => e.param === 'imageUrl') ?
31           'invalid' : '' %>"
32           type="text"
33           name="imageUrl"
34           id="imageUrl"
35           value="<% if (editing || hasError) { %><%= product.imageUrl %><% } %>">
36       </div>
37       <div class="form-control">
38         <label for="price">Price</label>
39         <input
40           class="<%= validationErrors.find(e => e.param === 'price') ? 'invalid' :
41           '' %>"
42           type="number"
43           name="price"
44           id="price"
45           step="0.01"
46           value="<% if (editing || hasError) { %><%= product.price %><% } %>">
47       </div>

```

```

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

```

```

1 /*./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;
```