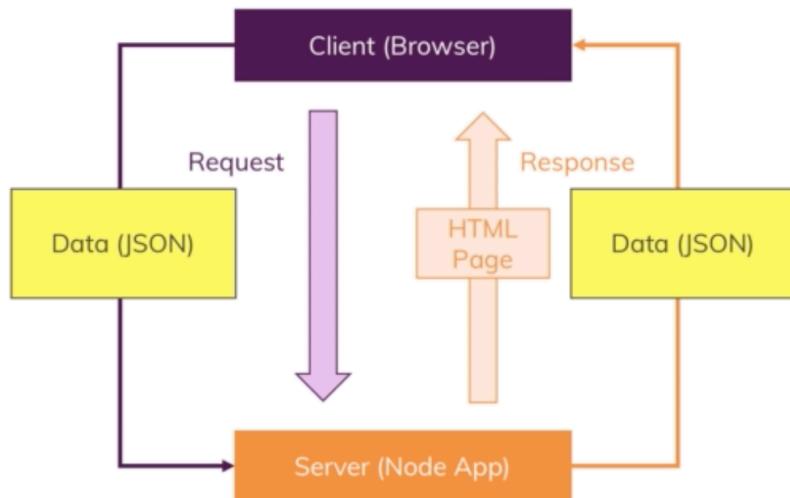


22. Understanding Async Requests

* Chapter 342: Module Introduction



What are Asynchronous Requests?

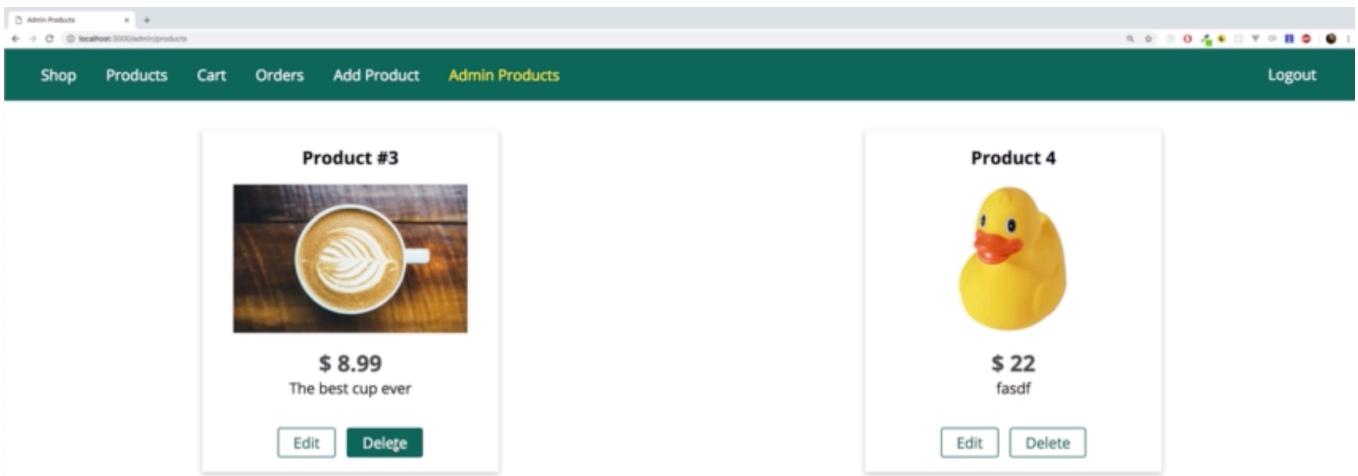


↓↓↓

- response always was a HTML page or a redirect to another route that then return an HTML page.
- response is not a new HTML page that needs to be rendered, but some data in that JSON format

* Chapter 343: Adding Client Side JS Code

1. update
- ./public/js/admin.js
- ./views/admin/products.ejs



The screenshot shows a browser window with the same product list as the first image. The developer tools are open, specifically the Elements tab, which displays the HTML structure of the page. A product card is selected, revealing its full DOM structure. The Styles tab is active, showing the CSS rules applied to the selected element. One rule, .product-item { width: 20rem; padding: 8px; }, is highlighted with a yellow background, indicating it is being examined.

- when i click delete, i would delete that, send that request to the server, get back a new version of the page where this product is then missing. and once we are done, the server will response with JSON Data. so some success message or something. once we get that message in our browser, we can delete this DOM element, so we can delete this article here. we could do all that with client side javascript.

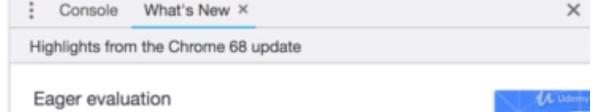
The screenshot shows a web application interface titled "Admin Products". The top navigation bar includes links for Shop, Products, Cart, Orders, Add Product, Admin Products, and Logout. Below the navigation is a list of products:

- Product #3**: An image of a coffee cup with latte art. Below it is the price **\$ 8.99** and the text "The best cup ever". At the bottom are "Edit" and "Delete" buttons.
- Product 4**: An image of a yellow rubber duck. Below it is the price **\$ 22** and the text "fasdf". At the bottom are "Edit" and "Delete" buttons.

The "Delete" button for Product 4 is highlighted with a red border. To the right of the browser window is the Chrome DevTools console, which displays the message "Clicked" twice, corresponding to the two delete button clicks.



This screenshot shows the same web application interface as the first one, but the "Delete" button for Product 4 is no longer highlighted with a red border. The rest of the page and the DevTools console output remain the same.



Screenshot of a web application showing two products: Product #3 (a cup of coffee) and Product 4 (a yellow rubber duck). The 'Delete' button for Product 4 is highlighted with a blue border.

Product #3

Product 4

\$ 8.99

The best cup ever

\$ 22

fasdf

Edit Delete

Edit Delete

Elements Console Network

```
<button class="btn" type="button" onclick="deleteProduct(this)">Delete</button>
```

Screenshot of the same web application after the delete operation. The 'Delete' button for Product 4 is now highlighted with a red border, indicating it has been clicked.

Product #3

Product 4

\$ 8.99

The best cup ever

\$ 22

fasdf

Edit Delete

Edit Delete

Elements Console Network

```
<button class="btn" type="button" onclick="deleteProduct(this)">Delete</button>
```

admin.js:2

```
<button class="btn" type="button" onclick="deleteProduct(this)">Delete</button>
```

Console What's New

Highlights from the Chrome 68 update

Eager evaluation

Screenshot of a web application showing two products in a list:

- Product #3**: A cup of coffee with latte art. Price: \$ 8.99. Description: The best cup ever. Buttons: Edit, Delete.
- Product 4**: A yellow rubber duck. Price: \$ 22. Description: fasdf. Buttons: Edit, Delete.

The browser's developer tools Console tab is open, showing the following output:

```
admin.js:2
<input type="hidden" value="5bb6133cee2611cdf67938bb"
      name="productId">
```

Screenshot of the same web application after a click, showing the product deletion process:

- Product #3**: A cup of coffee with latte art. Price: \$ 8.99. Description: The best cup ever. Buttons: Edit, Delete.
- Product 4**: A yellow rubber duck. Price: \$ 22. Description: fasdf. Buttons: Edit, Delete.

The browser's developer tools Console tab is open, showing the following output:

```
admin.js:2
<input type="hidden" value="5bb6133cee2611cdf67938bb"
      name="productId">
```

A message "Eager evaluation" is visible in the developer tools.

Screenshot of a web application showing two products in a list:

- Product #3**: A cup of coffee with latte art. Price: \$ 8.99. Description: The best cup ever. Buttons: Edit (light blue), Delete (dark green).
- Product 4**: A yellow rubber duck. Price: \$ 22. Description: fasdf. Buttons: Edit (light blue), Delete (dark green).

The browser's developer tools are open, showing the "Console" tab with the following message:

```
Highlight from the Chrome 68 update
```

Screenshot of the same web application after a click on the "Delete" button for Product #3.

- Product #3**: A cup of coffee with latte art. Price: \$ 8.99. Description: The best cup ever. Buttons: Edit (light blue), Delete (light blue).
- Product 4**: A yellow rubber duck. Price: \$ 22. Description: fasdf. Buttons: Edit (light blue), Delete (light blue).

The browser's developer tools are open, showing the "Console" tab with the following message:

```
Highlight from the Chrome 68 update
```

A new message appears in the "Console" tab:

```
5bb6133cee2611cdf67938bb admin.js:2
```

Screenshot of a web application showing two products in a grid layout. The left product is 'Product #3' featuring a latte with a heart-shaped foam design, priced at \$8.99 with the description 'The best cup ever'. The right product is 'Product 4' featuring a yellow rubber duck, priced at \$22 with the description 'fasdf'. Both products have 'Edit' and 'Delete' buttons below them. The browser's developer tools are open, showing the console tab with log entries: '5bb6133cee2611cdf67938bb' and '5bb61d5d2850d2d444d58e08'. The network tab shows a request to 'admin.js:2'. The status bar indicates 'Eager evaluation'.

```

1 <!--./views/admin/products.ejs-->
2
3 <%-- include('../includes/head.ejs') %>
4   <link rel="stylesheet" href="/css/product.css">
5 </head>
6
7 <body>
8   <%-- include('../includes/navigation.ejs') %>
9
10  <main>
11    <% if (prods.length > 0) { %>
12      <div class="grid">
13        <% for (let product of prods) { %>
14          <article class="card product-item">
15            <header class="card__header">
16              <h1 class="product__title">
17                <%= product.title %>
18              </h1>
19            </header>
20            <div class="card__image">
21              ">
22            </div>
23            <div class="card__content">
24              <h2 class="product__price">$
25                <%= product.price %>
26              </h2>
27              <p class="product__description">
28                <%= product.description %>
29              </p>
30            </div>
31            <div class="card__actions">
32              <a href="/admin/edit-product/<%= product._id %>
33                edit=true" class="btn">Edit</a>
34            <!--
35          </article>
36        <% } %>
37      </div>
38    <% } %>
39  </main>
40
41 <%-->
```

```

34           i will remove this entire '<form
35           action="/admin/delete-product" method="POST">'  

36           because this form was required for sending a  

37           request through the browser,  

38           sending a request with this x-www-url-form-
39           encoded-data  

40           so i will listen to a click to that button  

41           and then i will gather the productId and the  

42           CSRF token through the help of my client side javascript  

43           -->  

44           <input type="hidden" value="<%= product._id %>"  

45           name="productId">  

46           <input type="hidden" name="_csrf" value="<%=  

47           csrfToken %>">  

48           <!--  

49           i wanna react to a click on this 'Delete' button  

50           and therefore this button should not be of type  

51           of submit anymore  

52           but typeof 'button' instead  

53           'this' refer to the element on which we clicked.  

54           -->  

55           <button class="btn" type="button"  

56           onclick="deleteProduct(this)">Delete</button>  

57           </div>  

58           </article>  

59           <% } %>  

60           </div>  

61           <% } else { %>  

62           <h1>No Products Found!</h1>  

63           <% } %>  

64           </main>  

65           <%-- include('../includes/end.ejs') %>  

66           <!--  

67           i will import that javascript file from the js folder  

68           which is in a public folder  

69           which is served statically  

70           if we load it at the end of this file,  

71           we make sure that the entire DOM has been rendered  

72           and parsed by the time we execute our javascript code.  

73           -->  

74           <script src="/js/admin.js"></script>

```

```

1 //./public/js/admin.js
2
3 /**this is javascript code that will now not run on the server
4 * but that will run in the client.
5 */
6
7 const deleteProduct = (btn) => {
8   btn.parentNode.querySelector('[name=productId]').value
9   const csrf = btn.parentNode.querySelector('[name=_csrf]').value
10 }

```

* Chapter 346: Sending & Handling Background Requests

The screenshot shows a web application interface for managing products. At the top, there's a navigation bar with links for Shop, Products, Cart, Orders, Add Product, Admin Products, and Logout. Below the navigation, there are two product cards. The first card for 'Product #3' features an image of a latte, the price '\$ 8.99', the description 'The best cup ever', and two buttons: 'Edit' and 'Delete'. The second card for 'Product 4' features an image of a yellow rubber duck, the price '\$ 22', the description 'fasdf', and two buttons: 'Edit' and 'Delete'. To the right of the cards, a browser's developer tools Console tab is open, showing the output of a background request.

```
Console  What's New ×  
Highlights from the Chrome 68 update  
Eager evaluation
```

This screenshot shows the same web application interface after a background request has been processed. The 'Delete' button for 'Product 4' is now highlighted in blue, while the other buttons remain white. The browser's developer tools Console tab shows the response object from the background script.

```
admin.js:12  
Response {type: "basic", url: "http://localhost:3000/admin/products/5bb6133cee2611cdf67938bb", redirected: false, status: 200, ok: true, ...}
```

```
Console  What's New ×  
Highlights from the Chrome 68 update  
Eager evaluation
```

The screenshot shows a web application interface. At the top, there is a navigation bar with links: Shop, Products, Cart, Orders, Add Product, Admin Products, and Logout. Below the navigation bar, there are two product cards. The first card for 'Product #3' features an image of a coffee cup with latte art, a price of '\$ 8.99', and the text 'The best cup ever'. It has 'Edit' and 'Delete' buttons at the bottom. The second card for 'Product 4' features an image of a yellow rubber duck, a price of '\$ 22', and the text 'fasdf'. It also has 'Edit' and 'Delete' buttons. To the right of the cards, a browser's developer tools Console tab is open, displaying a response object with details like type: "basic", url: "http://localhost:3000/admin/product/5bb6133cee2611cdf67938bb", status: 200, ok: true, and headers.

This screenshot shows the same web application after some manipulation. Only the 'Product 4' card remains on the page. The browser's developer tools Console tab is still open, showing the same response object as the previous screenshot, indicating that the product list has been updated or filtered.

* Chapter 347: Manipulating The DOM

1. update
- ./routes/admin.js
- ./controllers/admin.js
- ./public/js/admin.js

Screenshot of a web browser showing the Admin Products interface. The page displays a product card for "Product 4" which is a green rubber duck. The card includes the price (\$22) and a placeholder text "fasdf". Below the card are two buttons: "Edit" and "Delete". The "Delete" button is highlighted with a blue border. The browser's developer tools are open, specifically the Elements tab, showing the HTML structure of the product card. The "card_actions" section of the card is selected. The "Styles" tab is active, showing the CSS rule for ".card_actions button" which has a border of 1px solid #ccc and padding of 4px. The "Console" tab shows the command "deleteProduct(this)" being executed.

```
<main>
  <div class="grid">
    <article class="card product-item">
      <header class="card__header">...</header>
      <div class="card__image">...</div>
      <div class="card__content">...</div>
      <div class="card__actions">
        <a href="/admin/edit-product/5bb61d5d2850d2d444d58e08?edit=true" class="btn">Edit</a>
        <input type="hidden" value="5bb61d5d2850d2d444d58e08" name="productId">
        <input type="hidden" name="_csrf" value="wsCVMskQ-6eqLcVkcV8EAjuyYAeo0XWv5ccs">
        ...
        <button class="btn" type="button" onclick="deleteProduct(this)">Delete</button> == $0
      </div>
    </article>
```

Styles Event Listeners DOM Breakpoints Properties Accessibility

Filter :hover .cls +

element.style { } .card__actions button, .card__actions a { main.css:205 }

: Console What's New X

Highlights from the Chrome 68 update

Eager evaluation

Screenshot of a web browser showing the Admin Products interface. The page displays a product card for "Product 4" which is now a yellow rubber duck. The card includes the price (\$22) and the same placeholder text "fasdf". Below the card are two buttons: "Edit" and "Delete". The "Delete" button is highlighted with a blue border. The browser's developer tools are open, specifically the Elements tab, showing the HTML structure of the product card. The "card_actions" section of the card is selected. The "Styles" tab is active, showing the CSS rule for ".card_actions button" which has a border of 1px solid #ccc and padding of 4px. The "Console" tab shows the command "deleteProduct(this)" being executed.

```
<main>
  <div class="grid">
    <article class="card product-item">
      <header class="card__header">...</header>
      <div class="card__image">...</div>
      <div class="card__content">...</div>
      <div class="card__actions">
        <a href="/admin/edit-product/5bb61d5d2850d2d444d58e08?edit=true" class="btn">Edit</a>
        <input type="hidden" value="5bb61d5d2850d2d444d58e08" name="productId">
        <input type="hidden" name="_csrf" value="wsCVMskQ-6eqLcVkcV8EAjuyYAeo0XWv5ccs">
        ...
        <button class="btn" type="button" onclick="deleteProduct(this)">Delete</button> == $0
      </div>
    </article>
```

Styles Event Listeners DOM Breakpoints Properties Accessibility

Filter :top

Console Default levels Grou X

: Console What's New X

Highlights from the Chrome 68 update

Eager evaluation

The screenshot shows a web application interface with a navigation bar at the top containing links for Shop, Products, Cart, Orders, Add Product, Admin Products, and Logout. To the right of the navigation bar is a developer tools window with tabs for Elements, Console, Network, and more. The Console tab is active, displaying the message '{message: "Success!"}' from 'admin.js:17'. Below the developer tools is a browser window showing a product list. The first product is a red kayak with the price \$11.99 and the description 'The first product'. The second product is a yellow rubber duck with the price \$22.99 and the description 'Pocket deuces'. Both products have a 'Details' button below them. A note in the browser window states: 'Please note that the page was NOT reloaded - instead the existing page was updated!'. The browser's status bar indicates it is on 'localhost:3000/admin/products'.

Login

localhost:3000/login

Shop Products Login Signup

E-Mail
test@test.com

Password

[Logia](#)

[Reset Password](#)

Console What's New

Highlights from the Chrome 68 update

Eager evaluation

Admin Products

localhost:3000/admin/products

Shop Products Cart Orders Add Product Admin Products Logout

Product #1



\$ 11.99

The first product

Edit Delete

Product #2



\$ 22.99

Pocket deuces

Edit Delete

Console What's New

Highlights from the Chrome 68 update

Eager evaluation

- let me log in with my other user

```

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', isAuthenticated, adminController.getAddProduct);
15
16 // /admin/products => GET
17 router.get('/products', isAuthenticated, adminController.getProducts);
18
19 // /admin/add-product => POST
20 router.post(
21   '/add-product',
22   [
23     body('title')
24       .isString()
25       .isLength({ min: 3 })
26       .trim(),
27     body('price').isFloat(),
28     body('description')
29       .isLength({ min: 5, max: 400 })
30       .trim()
31   ],
32   isAuthenticated,
33   adminController.postAddProduct
34 );
35
36 router.get('/edit-product/:productId', isAuthenticated, adminController.getEditProduct);
37
38 router.post(
39   '/edit-product',
40   [
41     body('title')
42       .isString()
43       .isLength({ min: 3 })
44       .trim(),
45     body('price').isFloat(),
46     body('description')
47       .isLength({ min: 5, max: 400 })
48       .trim()
49   ],
50   isAuthenticated,
51   adminController.postEditProduct
52 );
53
54 /**'delete' is HTTP method which makes sense for deleting
55 * now it's a only semantic thing
56 * and we can use 'post',
57 * you can in general use any HTTP verb to do anything
58 * because you define with your server side logic what happen.
59 */
60 router.delete('/product/:productId', isAuthenticated, adminController.deleteProduct);
61
62 module.exports = router;
63

```

```

1 // ./controllers/admin.js
2
3 const mongoose = require('mongoose');
4
5 const fileHelper = require('../util/file');
6

```

```
7 const { validationResult } = require('express-validator/check');
8
9 const Product = require('../models/product');
10
11 exports.getAddProduct = (req, res, next) => {
12   res.render('admin/edit-product', {
13     pageTitle: 'Add Product',
14     path: '/admin/add-product',
15     editing: false,
16     hasError: false,
17     errorMessage: null,
18     validationErrors: []
19   });
20 };
21
22 exports.postAddProduct = (req, res, next) => {
23   const title = req.body.title;
24   const image = req.file;
25   const price = req.body.price;
26   const description = req.body.description;
27   if (!image) {
28     return res.status(422).render('admin/edit-product', {
29       pageTitle: 'Add Product',
30       path: '/admin/add-product',
31       editing: false,
32       hasError: true,
33       product: {
34         title: title,
35         price: price,
36         description: description
37       },
38       errorMessage: 'Attached file is not an image.',
39       validationErrors: []
40     });
41   }
42   const errors = validationResult(req);
43
44   if (!errors.isEmpty()) {
45     console.log(errors.array());
46     return res.status(422).render('admin/edit-product', {
47       pageTitle: 'Add Product',
48       path: '/admin/add-product',
49       editing: false,
50       hasError: true,
51       product: {
52         title: title,
53         price: price,
54         description: description
55       },
56       errorMessage: errors.array()[0].msg,
57       validationErrors: errors.array()
58     });
59   }
60
61   const imageUrl = image.path;
62 }
```

```
63 const product = new Product({
64   // _id: new mongoose.Types.ObjectId('5badf72403fd8b5be0366e81'),
65   title: title,
66   price: price,
67   description: description,
68   imageUrl: imageUrl,
69   userId: req.user
70 });
71 product
72   .save()
73   .then(result => {
74     // console.log(result);
75     console.log('Created Product');
76     res.redirect('/admin/products');
77   })
78   .catch(err => {
79     // return res.status(500).render('admin/edit-product', {
80     //   pageTitle: 'Add Product',
81     //   path: '/admin/add-product',
82     //   editing: false,
83     //   hasError: true,
84     //   product: {
85     //     title: title,
86     //     imageUrl: imageUrl,
87     //     price: price,
88     //     description: description
89     //   },
90     //   errorMessage: 'Database operation failed, please try again.'
91     //   validationErrors: []
92     // });
93     // res.redirect('/500');
94     const error = new Error(err);
95     error.statusCode = 500;
96     return next(error);
97   });
98 };
99
100 exports.getEditProduct = (req, res, next) => {
101   const editMode = req.query.edit;
102   if (!editMode) {
103     return res.redirect('/');
104   }
105   const prodId = req.params.productId;
106   Product.findById(prodId)
107     .then(product => {
108       if (!product) {
109         return res.redirect('/');
110       }
111       res.render('admin/edit-product', {
112         pageTitle: 'Edit Product',
113         path: '/admin/edit-product',
114         editing: editMode,
115         product: product,
116         hasError: false,
117         errorMessage: null,
118         validationErrors: []
119       });
120     })
121     .catch(err => {
122       const error = new Error(err);
123       error.statusCode = 500;
124       return next(error);
125     });
126 };
127
128 exports.deleteProduct = (req, res, next) => {
129   const prodId = req.params.productId;
130   Product.findByIdAndDelete(prodId)
131     .then(() => {
132       res.redirect('/admin/products');
133     })
134     .catch(err => {
135       const error = new Error(err);
136       error.statusCode = 500;
137       return next(error);
138     });
139 };
140
141 module.exports = router;
```

```
119     });
120   })
121   .catch(err => {
122     const error = new Error(err);
123     error.httpStatusCode = 500;
124     return next(error);
125   });
126 });
127
128 exports.postEditProduct = (req, res, next) => {
129   const prodId = req.body.productId;
130   const updatedTitle = req.body.title;
131   const updatedPrice = req.body.price;
132   const image = req.file;
133   const updatedDesc = req.body.description;
134
135   const errors = validationResult(req);
136
137   if (!errors.isEmpty()) {
138     return res.status(422).render('admin/edit-product', {
139       pageTitle: 'Edit Product',
140       path: '/admin/edit-product',
141       editing: true,
142       hasError: true,
143       product: {
144         title: updatedTitle,
145         price: updatedPrice,
146         description: updatedDesc,
147         _id: prodId
148       },
149       errorMessage: errors.array()[0].msg,
150       validationErrors: errors.array()
151     });
152   }
153
154   Product.findById(prodId)
155     .then(product => {
156       if (product.userId.toString() !== req.user._id.toString()) {
157         return res.redirect('/');
158       }
159       product.title = updatedTitle;
160       product.price = updatedPrice;
161       product.description = updatedDesc;
162       if (image) {
163         fileHelper.deleteFile(product.imageUrl);
164         product.imageUrl = image.path;
165       }
166       return product.save().then(result => {
167         console.log('UPDATED PRODUCT!');
168         res.redirect('/admin/products');
169       });
170     })
171     .catch(err => {
172       const error = new Error(err);
173       error.httpStatusCode = 500;
174       return next(error);
175     });
176 }
```

```

175     });
176 };
177
178 exports.getProducts = (req, res, next) => {
179   Product.find({ userId: req.user._id })
180     // .select('title price _id')
181     // .populate('userId', 'name')
182     .then(products => {
183       console.log(products);
184       res.render('admin/products', {
185         prods: products,
186         pageTitle: 'Admin Products',
187         path: '/admin/products'
188       });
189     })
190     .catch(err => {
191       const error = new Error(err);
192       error.statusCode = 500;
193       return next(error);
194     });
195 };
196
197 exports.deleteProduct = (req, res, next) => {
198   /**'prodId' is not extracted from the req.body anymore
199    * because delete request also are not allowed to have a req.body
200    * but instead, we have now that URL parameter 'prodId' in ./routes/admin.js
201    * so i change body for params and that's it.
202    */
203   const prodId = req.params.productId;
204   Product.findById(prodId)
205     .then(product => {
206       if (!product) {
207         return next(new Error('Product not found.'));
208       }
209       fileHelper.deleteFile(product.imageUrl);
210       return Product.deleteOne({ _id: prodId, userId: req.user._id });
211     })
212     .then(() => {
213       console.log('DESTROYED PRODUCT');
214       /**i will not redirect anymore 'res.redirect('/admin/products)'
215        * because i will not load a new page.
216        * the request triggering this action will be sent behind the scenes for the existing
217        * page
218        * so i wanna keep that existing page
219        * therefoe my response will be a response
220        * where i send JSON data
221        * which is a special format
222        * and with express.js, i can use a JSON helper method to conveniently return JSON
223        * data
224        * and JSON is a data format that looks like a javascript object
225        * so with curly brace and then key-value pairs.
226        *
227        * since we don't redirect it and so on,
228        * where we get a status code set automatically,
229        * it would make sense to be very clear
230        * about the tatus code we have

```

```

229 *
230 * you pass a javascript object
231 * which will be transformed to JSON automatically
232 * so you can pass your normal javascript object here
233 * you don't need double quotation marks around your keys
234 *
235 * we return JSON responses
236 * because we don't wanna render a new page
237 */
238 res.status(200).json({ message: 'Success!' });
239 }
240 .catch(err => {
241   res.status(500).json({ message: 'Deleting product failed.' });
242 });
243 };
244

```

```

1 //./public/js/admin.js
2
3 const deleteProduct = btn => {
4   const prodId = btn.parentNode.querySelector('[name=productId]').value;
5   const csrf = btn.parentNode.querySelector('[name=_csrf]').value;
6
7   /**'closest()' method provided by javascript pass a selector to closest
8    * which gives you the closest element with that selector
9    * and the closest ancestor element
10   * and i will use <article>
11   * because i only have one article in my ancestor history for this button
12   * so if i select my closest article,
13   * that should be the element i wanna delete
14   */
15   const productElement = btn.closest('article');
16   /**'fetch()' method is a method supported by the browser for sending HTTP requests
17   * and it's not just for fetching data
18   * but for sending data.
19   *
20   * '/product/' will send that to the same server
21   * if you don't specify a different host with HTTP
22   * and i will add my prodId
23   * and then 2nd argument is an object where you can configure this fetch
24   *
25   * i'm not sending any JSON data with my request
26   * because it is delete request without a post body
27   * if it were and that is something we will see in the REST API section,
28   * then i would have to parse JSON data in my backend
29   * in app.js,
30   * we only have 2 parsers, one for urlencoded data which we don't have when we send
JSON data
31   *
32   *     'app.use(bodyParser.urlencoded({ extended: false }))'
33   *
34   * and one for multipart data which we don't have there
35   *
36   *     'app.use(multer({ storage: fileStorage, fileFilter }).single('image'))'
37   *
38   * we have to add a new body parser that is able to handle JSON data
39   * and extract that from incoming request.

```

```
40    * but i don't add it here because we don't need it here
41    */
42
43  fetch('/admin/product/' + prodId, {
44    method: 'DELETE',
45    /**
46     * **in 'headers', we could encode our CSRF token
47     * because we still need to attach this to our request.
48     * we can't send it in the req.body
49     * because delete request don't have a body
50     *
51     * the good thing is the csurf package we are using on the server
52     * doesn't just look into req.body,
53     * it also looks into the query parameters
54     * and therefore we could add it there
55     * it also looks into the headers.
56     * so there we can add a csrf token header.
57     */
58    headers: {
59      'csrf-token': csrf
60    }
61  })
62  .then(result => {
63    /**
64     * **we can return 'result.json()'
65     * which is a function that will throw or return a new promise
66     */
67    return result.json();
68  })
69  .then(data => {
70    console.log(data);
71    productElement.parentNode.removeChild(productElement);
72  })
73  .catch(err => {
74    console.log(err);
75  });
});
```