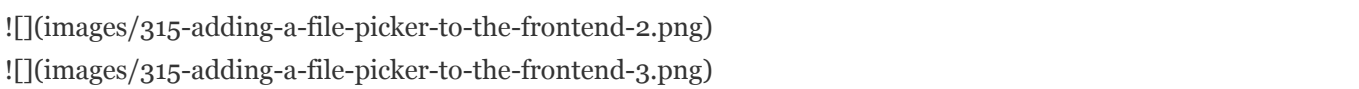
20. File Upload & Download

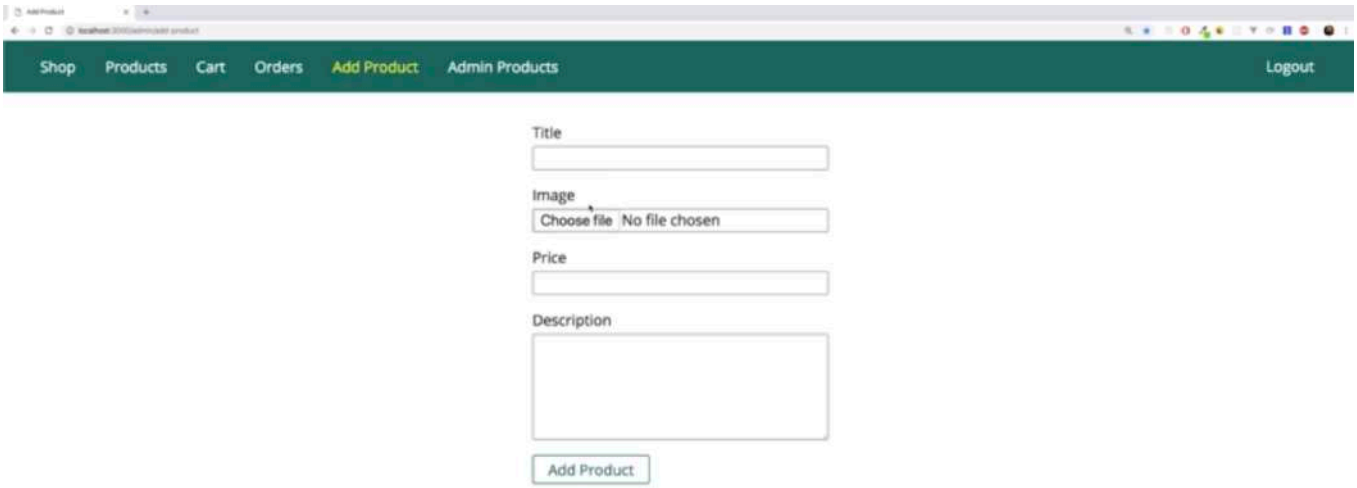
* Chapter 314: Module Introduction

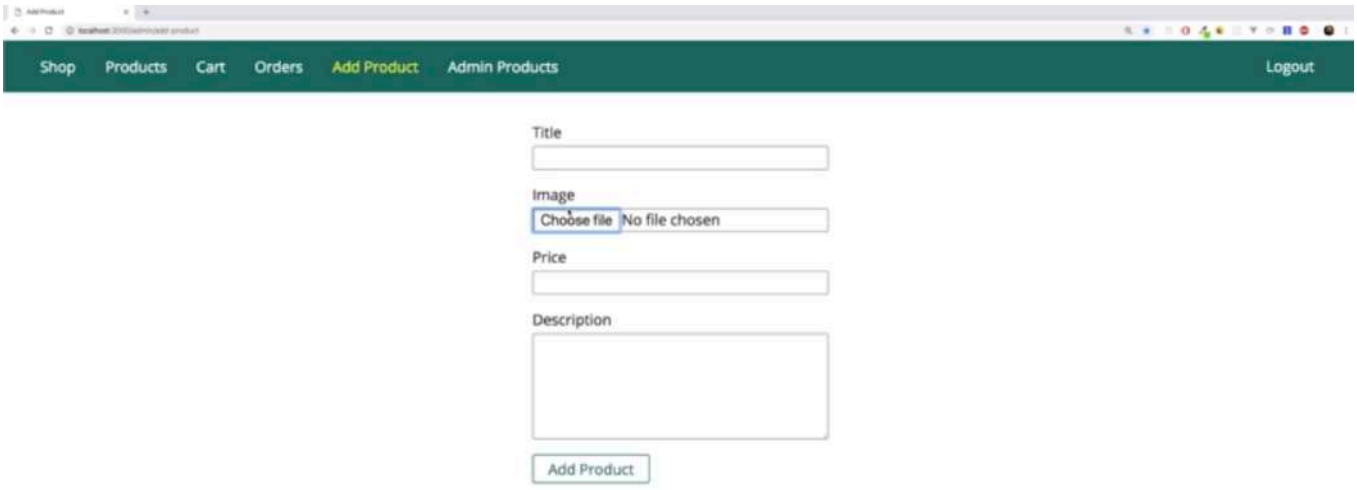
* Chapter 315: Adding A File Picker To The Frontend

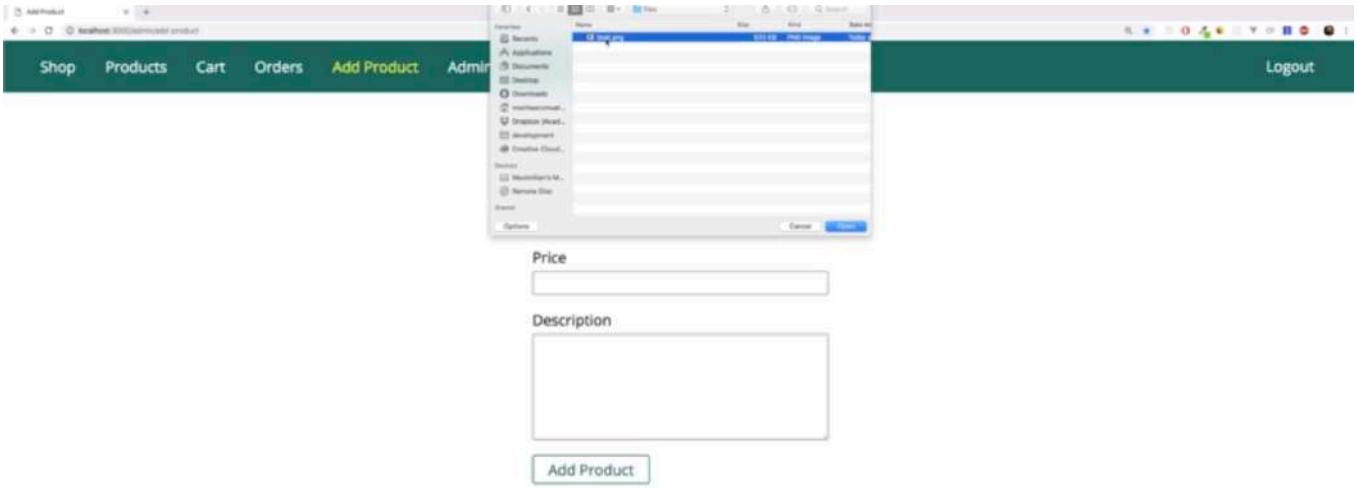
1. update
- ./views/admin/edit-product.ejs

- Adding file upload means that we have to do 2 things. first thing is that we need to adjust our form to show a file picker to our users. so a tool which they can use to select a file on their operating system on their computer.
- 2nd part is that we need to be able to accept that file in the place where we handle it, where we handle the incoming requests.




Shop
Products
Cart
Orders
Add Product
Admin Products
Logout
Title
Image
Choose file
No file chosen
Price
Description
Add Product


Shop
Products
Cart
Orders
Add Product
Admin Products
Logout
Title
Image
Choose file
No file chosen
Price
Description
Add Product



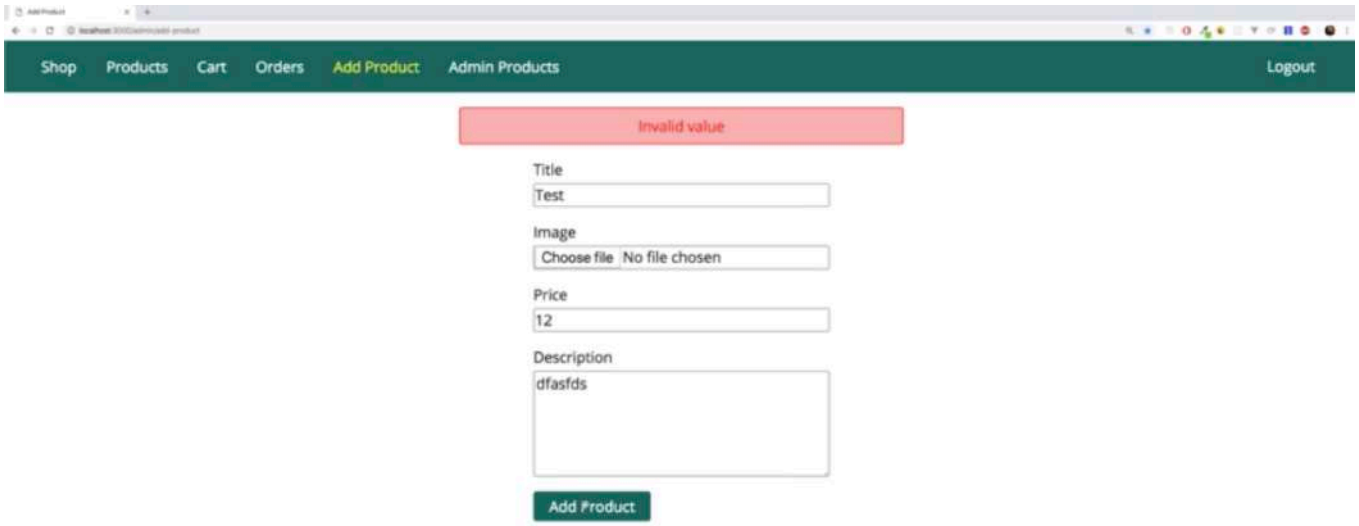
- now we have our file picker here which is a default HTML element

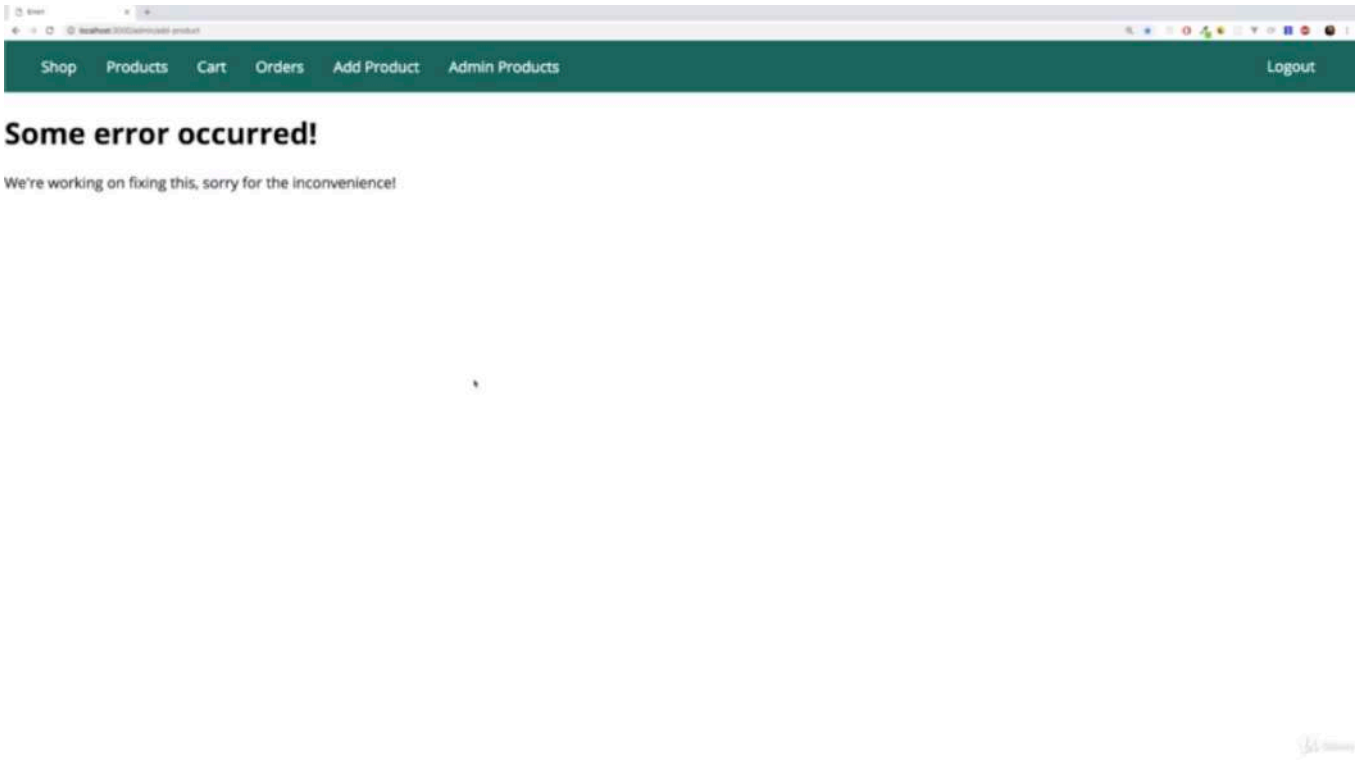
```
<!--./views/admin/edit-product.ejs-->

<%- include('../includes/head.ejs') %>
    <link rel="stylesheet" href="/css/forms.css">
    <link rel="stylesheet" href="/css/product.css">
</head>

<body>
   <%- include('../includes/navigation.ejs') %>

    <main>
        <% if (errorMessage) { %>
            <div class="user-message user-message--error"><%= errorMessage %></div>
        <% } %>
        <form class="product-form" action="/admin/<% if (editing) { %>edit-product<% } else { %>add-product<% } %>" method="POST">
            <div class="form-control">
                <label for="title">Title</label>
                <input
                    class="<%= validationErrors.find(e => e.param === 'title') ? 'invalid' : '' %>"
                    type="text"
                    name="title"
                    id="title"
                    value="<% if (editing || hasError) { %><%= product.title %><% } %>">
            </div>
            <!--
            <div class="form-control">
                <label for="imageUrl">Image URL</label>
                <input
                    class="<%= validationErrors.find(e => e.param === 'imageUrl') ? 'invalid' : '' %>"
                    type="text"
                    name="imageUrl"
```

```
                id="imageUrl"
                value="<% if (editing || hasError) { %><%= product.imageUrl %><% } %>">
            </div>
            -->
            <div class="form-control">
                <label for="image">Image</label>
                <input
                    type="file"
                    name="image"
                    id="image" >
            </div>
            <div class="form-control">
                <label for="price">Price</label>
                <input
                    class="<%= validationErrors.find(e => e.param === 'price') ? 'invalid' : '' %>"
                    type="number"
                    name="price"
                    id="price"
                    step="0.01"
                    value="<% if (editing || hasError) { %><%= product.price %><% } %>">
            </div>
            <div class="form-control">
                <label for="description">Description</label>
                <textarea
                    class="<%= validationErrors.find(e => e.param === 'description') ? 'invalid' : '' %>"
                    name="description"
                    id="description"
                    rows="5"><% if (editing || hasError) { %><%= product.description %><% } %></textarea>
            </div>
            <% if (editing) { %>
                <input type="hidden" value="<%= product._id %>" name="productId">
            <% } %>

            <input type="hidden" name="_csrf" value="<%= csrfToken %>">
            <button class="btn" type="submit"><% if (editing) { %>Update Product<% } else { %>Add Product<% } %></button>
        </form>
    </main>
<%- include('../includes/end.ejs') %>
```

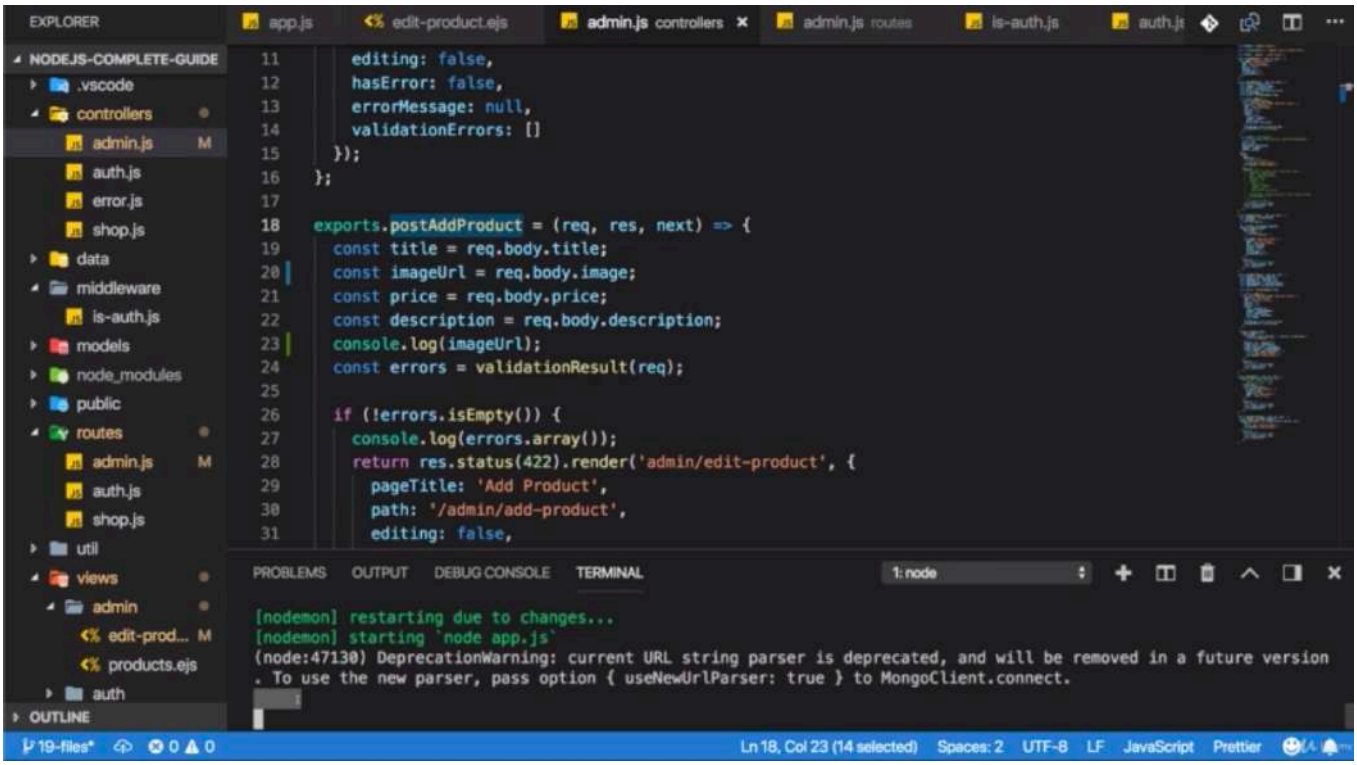
* Chapter 316: Handling Multipart Form Data

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

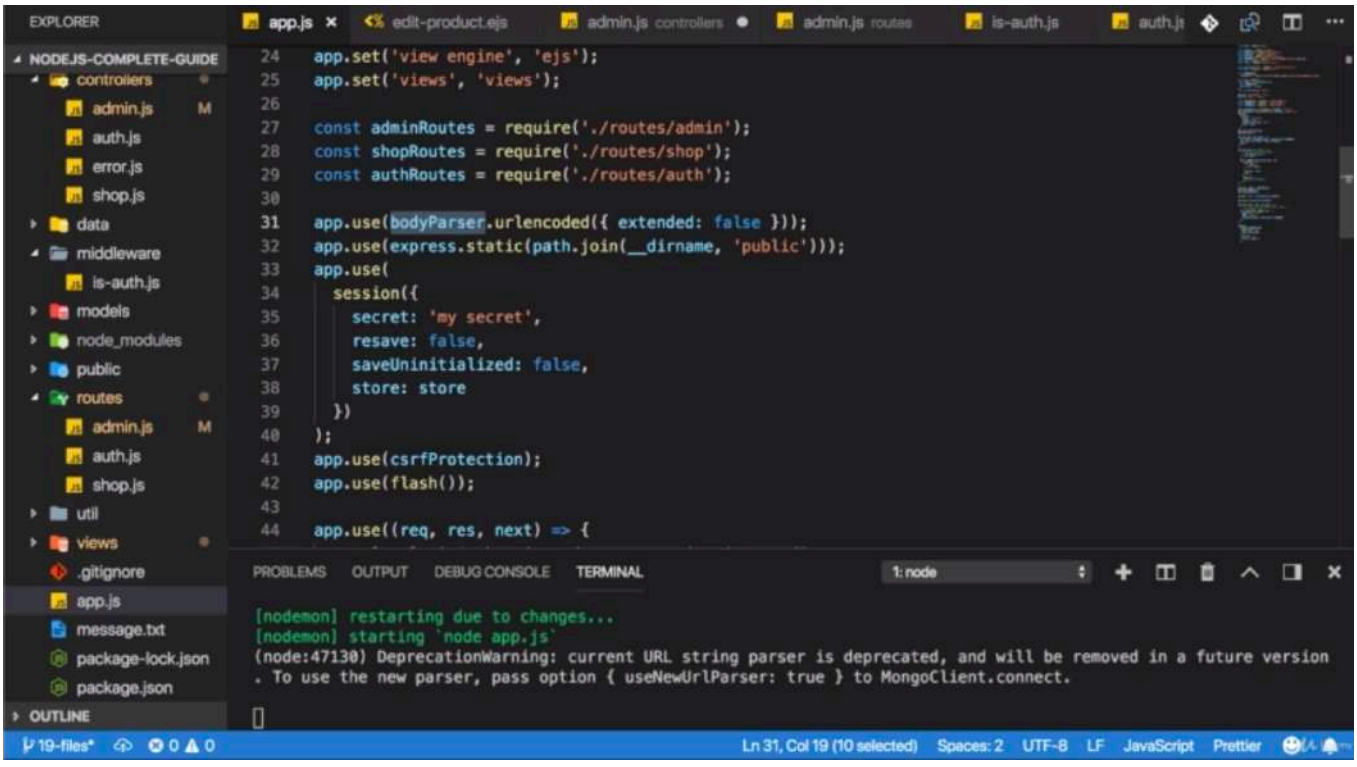




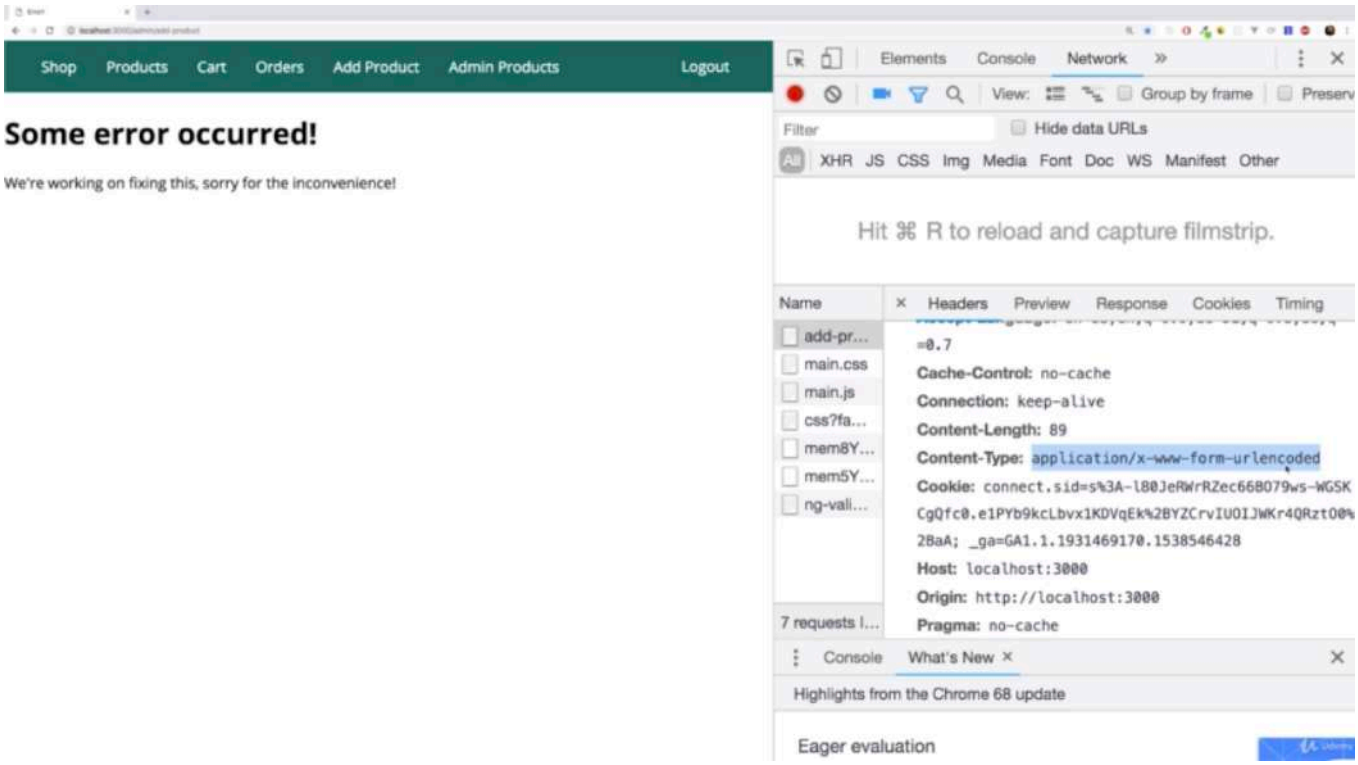
- the error we are getting is stemming from the fact that we are not able to extract our image correctly.



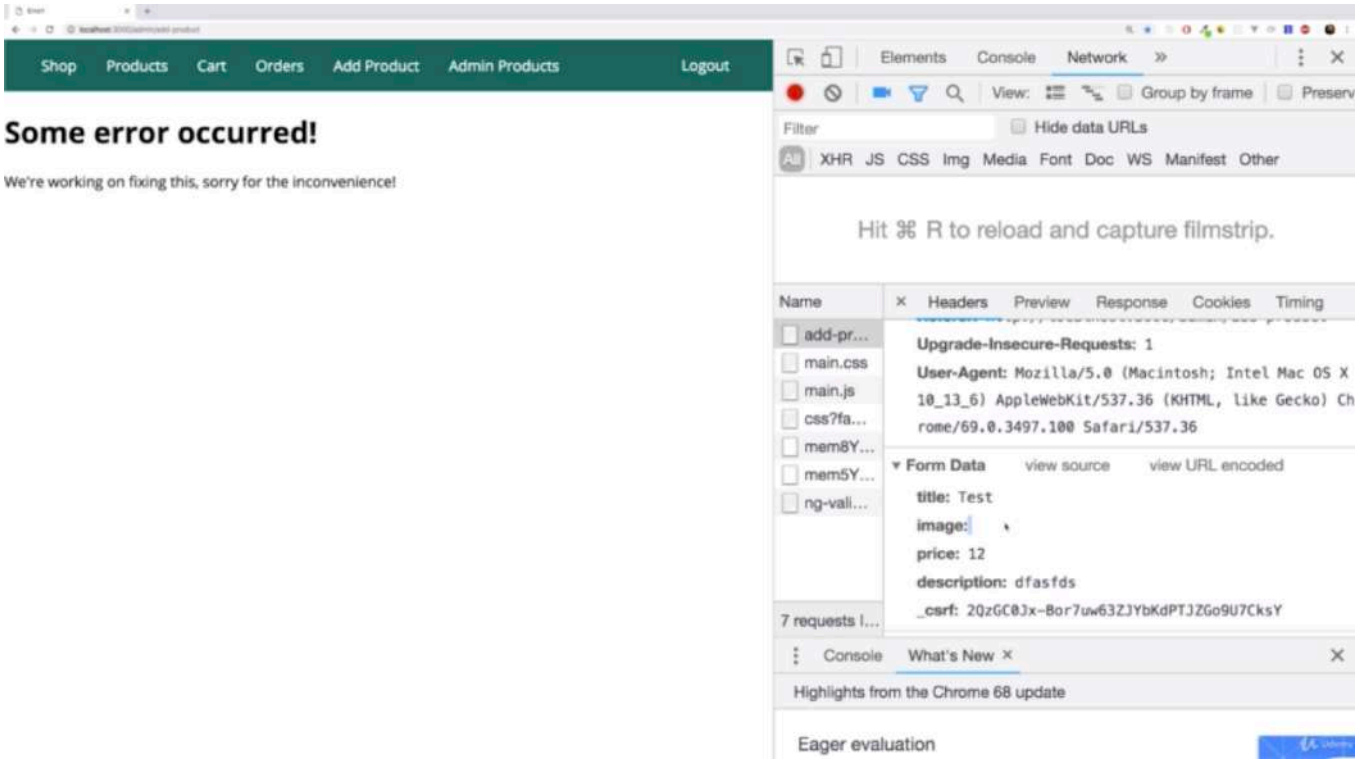
- if i console log, you don't see anything. because we failed to extract an image from the req.body.



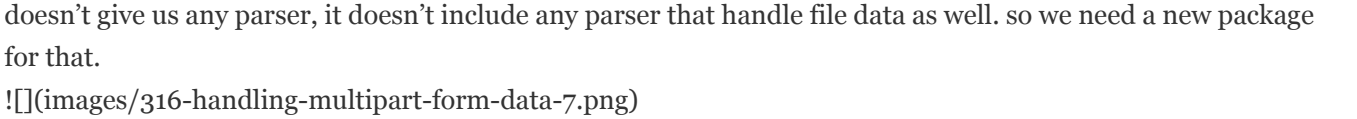
- because for extracting the content of our incoming requests, we set up a middleware in app.js, we are using a special middleware, 'body-parser' middleware and this middleware uses or exposes a couple of different parsers and we are using the 'urlencoded' parser.

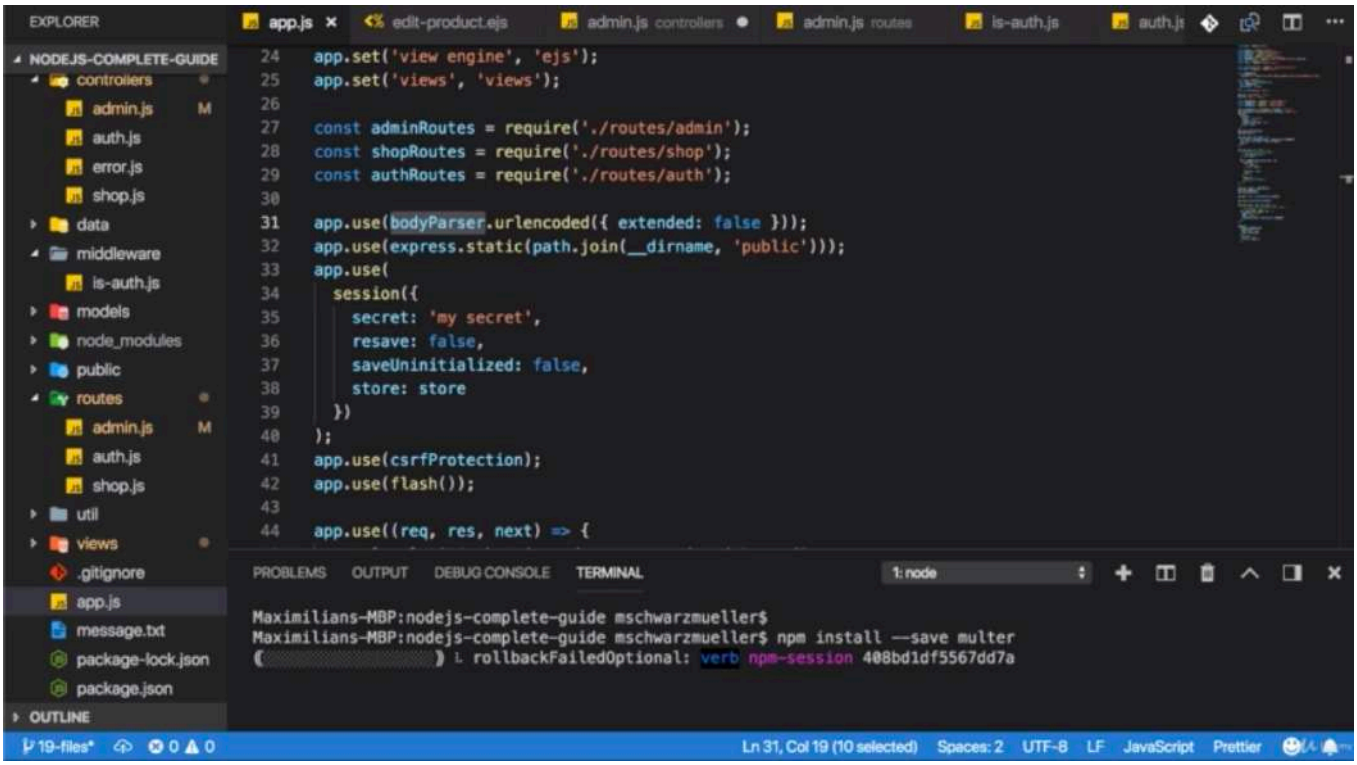


- urlencoded data is text data. so if a form is submitted without a file, just with text field, no matter if that text field then stores a number, a url or plain text but it's all encoded in text when it's submitted. this format is then called urlencoded.



- we can see that the content type is application and then x-www-form-urlencoded. this means it tries to put all the data as text into its form body.
- we can see that down there, just form data, title, price and so on and just image with nothing. this is invalid. this is an empty text because it can't extract our file as text because a file is binary data.
- because of that, failing extraction, we need to parse our data differently and the body parser that we are using doesn't give us any parser, it doesn't include any parser that handle file data as well. so we need a new package for that.





- 'npm install —save multer'
- multer is another third party package that parses incoming requests. but this package parses incoming requests for files. so it's able to handle file requests or requests with mixed data, with text and file data.
- we will still keep body-parser because we still have like for example, our sign up form where we only submit urlencoded data but now we will have to use a different encoding and that starts with our form.

```
// ./controllers/admin.js

const mongoose = require('mongoose');

const { validationResult } = require('express-validator/check');

const Product = require('../models/product');

exports.getAddProduct = (req, res, next) => {
  res.render('admin/edit-product', {
    pageTitle: 'Add Product',
    path: '/admin/add-product',
    editing: false,
    hasError: false,
    errorMessage: null,
    validationErrors: []
  });
};

exports.postAddProduct = (req, res, next) => {
  const title = req.body.title;
  const imageUrl = req.body.image;
  const price = req.body.price;
  const description = req.body.description;
  const errors = validationResult(req);

  if (!errors.isEmpty()) {
    console.log(errors.array());
    return res.status(422).render('admin/edit-product', {
```

```
      pageTitle: 'Add Product',
      path: '/admin/add-product',
      editing: false,
      hasError: true,
      product: {
        title: title,
        imageUrl: imageUrl,
        price: price,
        description: description
      },
      errorMessage: errors.array()[0].msg,
      validationErrors: errors.array()
    });
  }

  const product = new Product({
    //_id: new mongoose.Types.ObjectId('5badf72403fd8b5be0366e81'),
    title: title,
    price: price,
    description: description,
    imageUrl: imageUrl,
    userId: req.user
  });
  product
    .save()
    .then(result => {
      // console.log(result);
      console.log('Created Product');
      res.redirect('/admin/products');
    })
    .catch(err => {
      // return res.status(500).render('admin/edit-product', {
      //   pageTitle: 'Add Product',
      //   path: '/admin/add-product',
      //   editing: false,
      //   hasError: true,
      //   product: {
      //     title: title,
      //     imageUrl: imageUrl,
      //     price: price,
      //     description: description
      //   },
      //   errorMessage: 'Database operation failed, please try again.',
      //   validationErrors: []
      // });
      //res.redirect('/500');
      const error = new Error(err)
      error.httpStatusCode = 500
      return next(error)
    });
};

exports.getEditProduct = (req, res, next) => {
  const editMode = req.query.edit;
  if (!editMode) {
    return res.redirect('/');
```

```
  }
  const prodId = req.params.productId;
  Product.findById(prodId)
    .then(product => {
      if (!product) {
        return res.redirect('/');
      }
      res.render('admin/edit-product', {
        pageTitle: 'Edit Product',
        path: '/admin/edit-product',
        editing: editMode,
        product: product,
        hasError: false,
        errorMessage: null,
        validationErrors: []
      });
    })
    .catch(err => {
      const error = new Error(err)
      error.httpStatusCode = 500
      return next(error)
    });
};

exports.postEditProduct = (req, res, next) => {
  const prodId = req.body.productId;
  const updatedTitle = req.body.title;
  const updatedPrice = req.body.price;
  const updatedImageUrl = req.body.imageUrl;
  const updatedDesc = req.body.description;

  const errors = validationResult(req);

  if (!errors.isEmpty()) {
    return res.status(422).render('admin/edit-product', {
      pageTitle: 'Edit Product',
      path: '/admin/edit-product',
      editing: true,
      hasError: true,
      product: {
        title: updatedTitle,
        imageUrl: updatedImageUrl,
        price: updatedPrice,
        description: updatedDesc,
        _id: prodId
      },
      errorMessage: errors.array()[0].msg,
      validationErrors: errors.array()
    });
  }

  Product.findById(prodId)
    .then(product => {
      if (product.userId.toString() !== req.user._id.toString()) {
        return res.redirect('/');
      }
```

```
      product.title = updatedTitle;
      product.price = updatedPrice;
      product.description = updatedDesc;
      product.imageUrl = updatedImageUrl;
      return product.save().then(result => {
        console.log('UPDATED PRODUCT!');
        res.redirect('/admin/products');
      });
    })
    .catch(err => {
      const error = new Error(err)
      error.httpStatusCode = 500
      return next(error)
    });
};

exports.getProducts = (req, res, next) => {
  Product.find({ userId: req.user._id })
    // .select('title price -_id')
    // .populate('userId', 'name')
    .then(products => {
      console.log(products);
      res.render('admin/products', {
        prods: products,
        pageTitle: 'Admin Products',
        path: '/admin/products'
      });
    })
    .catch(err => {
      const error = new Error(err)
      error.httpStatusCode = 500
      return next(error)
    });
};

exports.postDeleteProduct = (req, res, next) => {
  const prodId = req.body.productId;
  Product.deleteOne({ _id: prodId, userId: req.user._id })
    .then(() => {
      console.log('DESTROYED PRODUCT');
      res.redirect('/admin/products');
    })
    .catch(err => {
      const error = new Error(err)
      error.httpStatusCode = 500
      return next(error)
    });
};

```

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

const path = require('path');

const express = require('express');
const { body } = require('express-validator/check')

```

```
const adminController = require('../controllers/admin');
const isAuth = require('../middleware/is-auth');

const router = express.Router();

// /admin/add-product => GET
router.get('/add-product', isAuth, adminController.getAddProduct);

// /admin/products => GET
router.get('/products', isAuth, adminController.getProducts);

// /admin/add-product => POST
router.post('/add-product', [
        body('title').isString().isLength({ min: 3 }).trim(),
        body('price').isFloat(),
        body('description').isLength({ min: 5, max: 400 }).trim()
    ],
    isAuth,
    adminController.postAddProduct);

router.get('/edit-product/:productId', isAuth, adminController.getEditProduct);

router.post('/edit-product',[
        body('title').isString().isLength({ min: 3 }).trim(),
        body('imageUrl').isURL(),
        body('price').isFloat(),
        body('description').isLength({ min: 5, max: 400 }).trim()
    ],
    isAuth,
    adminController.postEditProduct);

router.post('/delete-product', isAuth, adminController.postDeleteProduct);

module.exports = router;
```

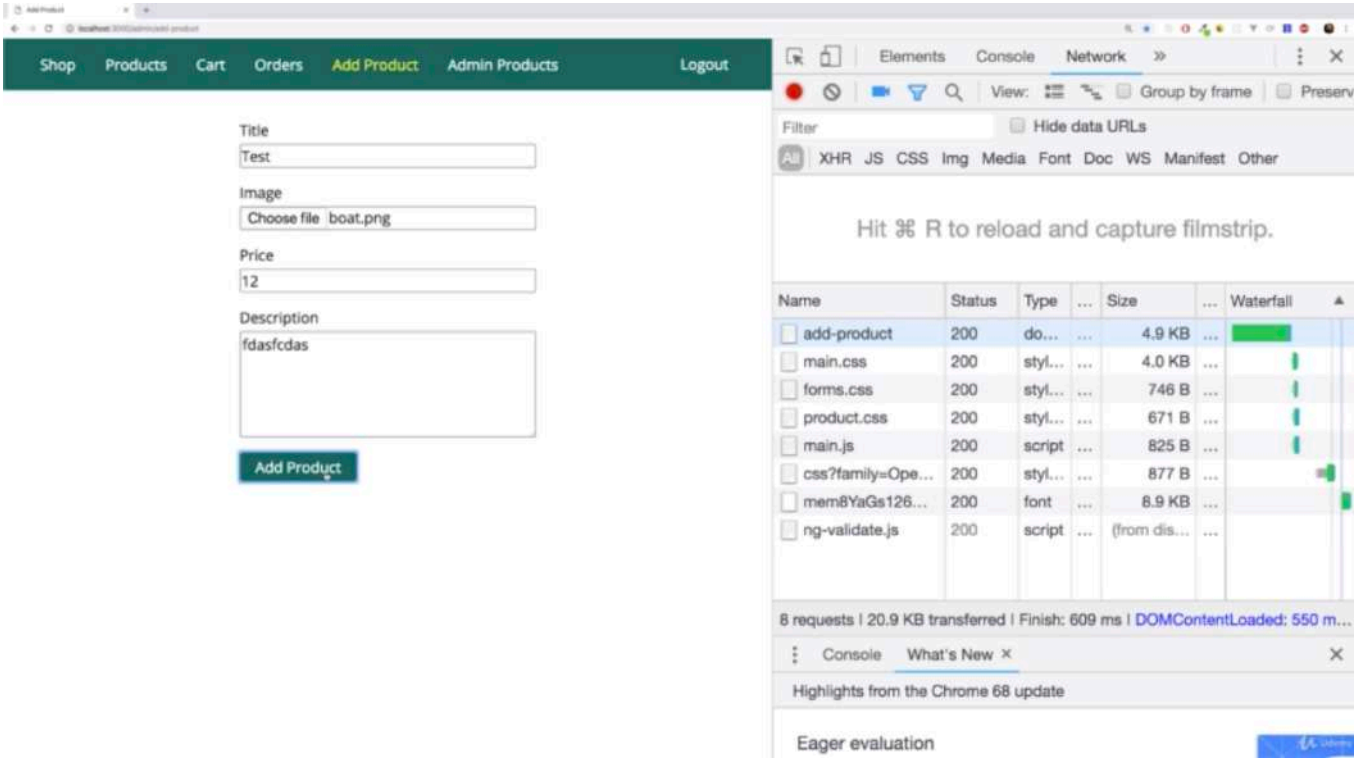
```
<!--./views/admin/edit-product.ejs-->

<%- include('../includes/head.ejs') %>
    <link rel="stylesheet" href="/css/forms.css">
    <link rel="stylesheet" href="/css/product.css">
</head>

<body>
   <%- include('../includes/navigation.ejs') %>

    <main>
        <% if (errorMessage) { %>
            <div class="user-message user-message--error"><%= errorMessage %></div>
        <% } %>
        <!--
            'enctype' field which i will set to multipart form data application
            x-www-form-url-encoded as the default
            but now we will switch to multipart form data which is the content-type
            telling the server that this submission and this request will not contain
plaintext
            but will contain mixed data, text and binary data
            and multer will be looking for incoming requests,
```

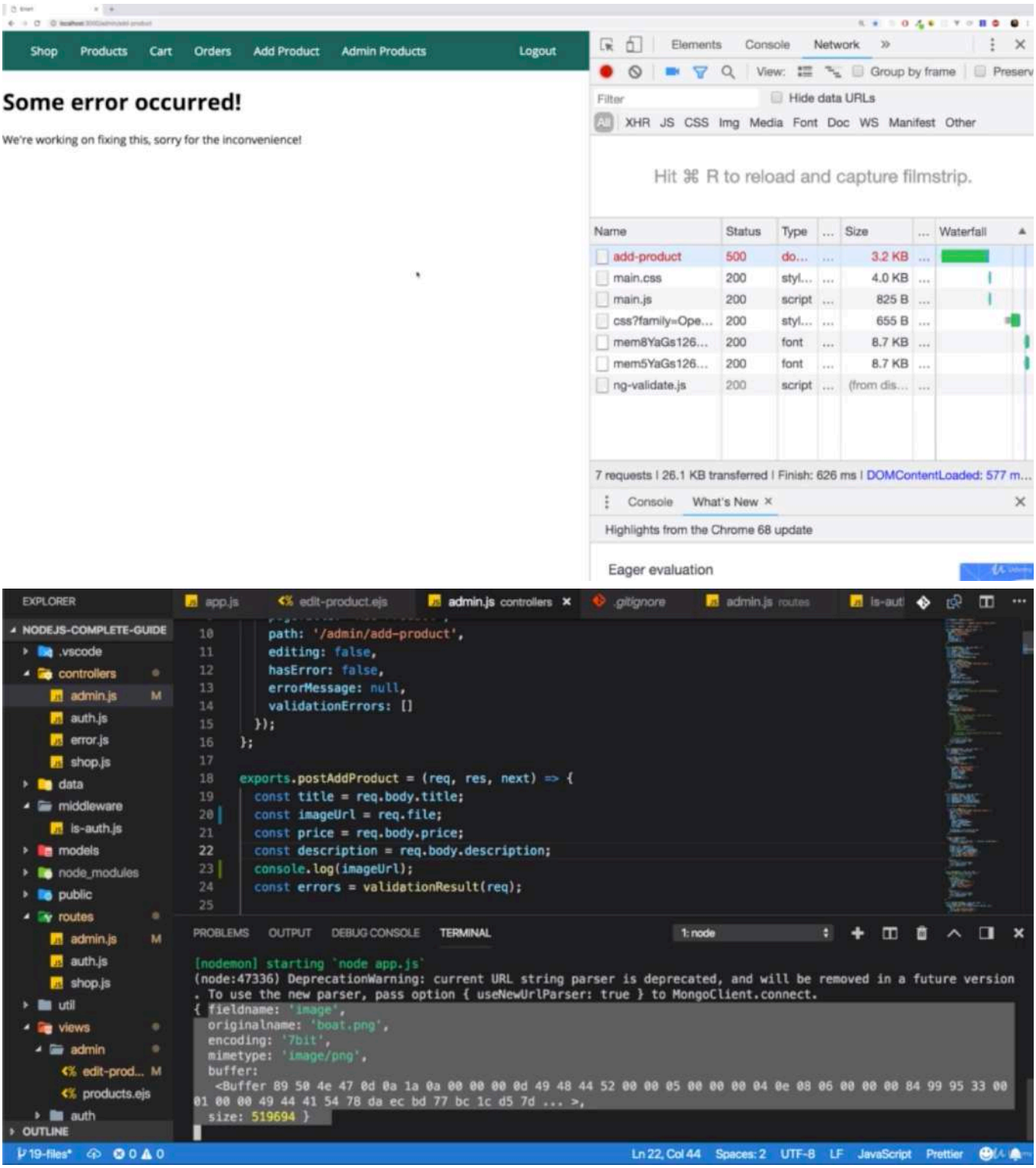
```
            with this type of data 'multipart/form-data'
            and will then be able to parse both the text and our file
        -->
        <form class="product-form" action="/admin/<% if (editing) { %>edit-product<% } else { %>add-product<% } %>" method="POST" enctype="multipart/form-data">
            <div class="form-control">
                <label for="title">Title</label>
                <input
                    class="<%= validationErrors.find(e => e.param === 'title') ? 'invalid' : '' %>"
                    type="text"
                    name="title"
                    id="title"
                    value="<% if (editing || hasError) { %><%= product.title %><% } %>">
            </div>
            <!--
            <div class="form-control">
                <label for="imageUrl">Image URL</label>
                <input
                    class="<%= validationErrors.find(e => e.param === 'imageUrl') ? 'invalid' : '' %>"
                    type="text"
                    name="imageUrl"
                    id="imageUrl"
                    value="<% if (editing || hasError) { %><%= product.imageUrl %><% } %>">
            </div>
            -->
            <div class="form-control">
                <label for="image">Image</label>
                <input
                    type="file"
                    name="image"
                    id="image" >
            </div>
            <div class="form-control">
                <label for="price">Price</label>
                <input
                    class="<%= validationErrors.find(e => e.param === 'price') ? 'invalid' : '' %>"
                    type="number"
                    name="price"
                    id="price"
                    step="0.01"
                    value="<% if (editing || hasError) { %><%= product.price %><% } %>">
            </div>
            <div class="form-control">
                <label for="description">Description</label>
                <textarea
                    class="<%= validationErrors.find(e => e.param === 'description') ? 'invalid' : '' %>"
                    name="description"
                    id="description"
                    rows="5"><% if (editing || hasError) { %><%= product.description %><% } %></textarea>
            </div>
            <% if (editing) { %>
```

```
72              <input type="hidden" value="<%= product._id %>" name="productId">
73              <% } %>
74
75              <input type="hidden" name="_csrf" value="<%= csrfToken %>">
76              <button class="btn" type="submit"><% if (editing) { %>Update Product<% } else {
    %>Add Product<% } %></button>
77          </form>
78      </main>
79  <%- include('../includes/end.ejs') %>
```

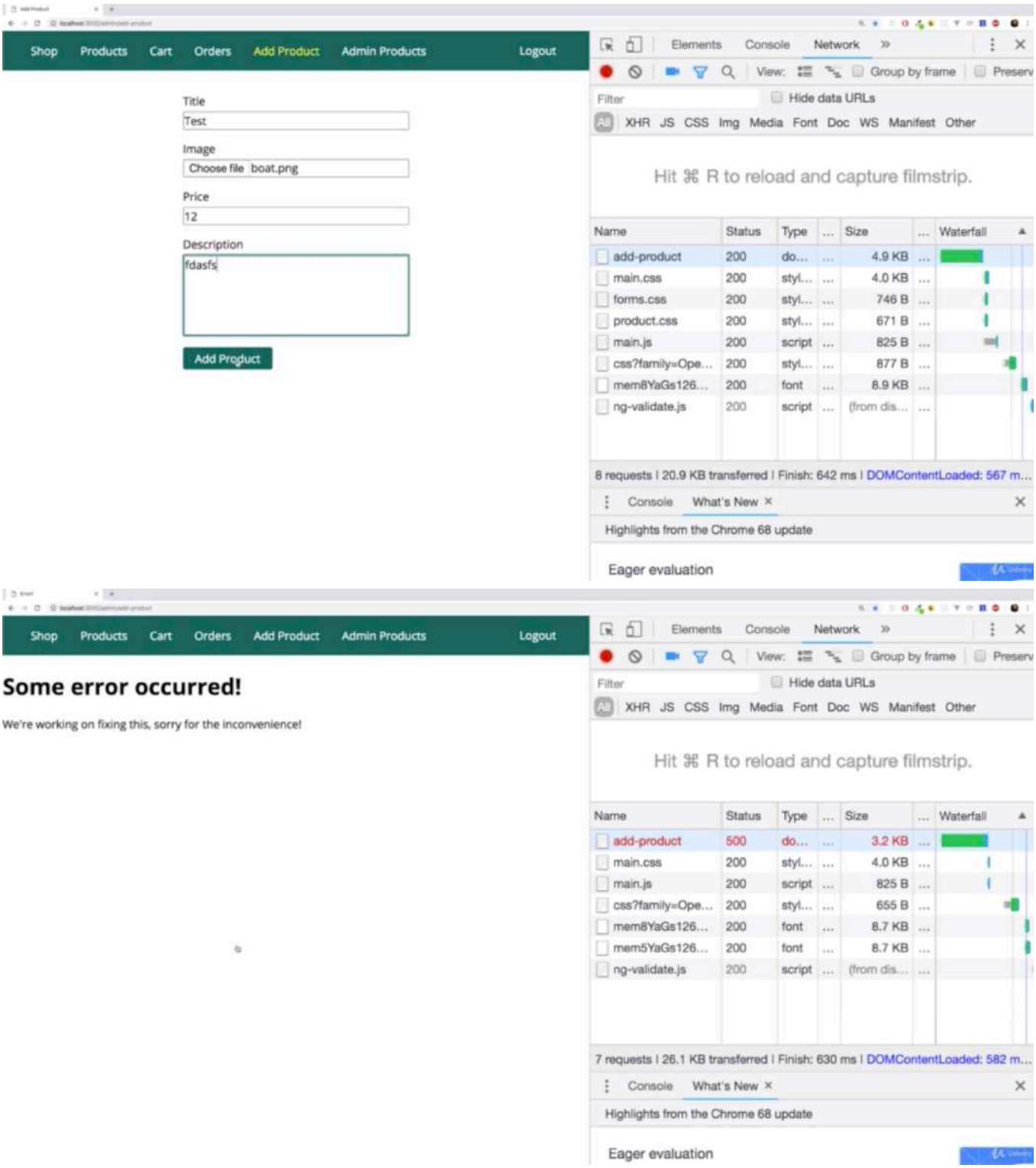
* Chapter 317: Handling File Uploads With Multer

1. update
- app.js
- ./controllers/admin.js

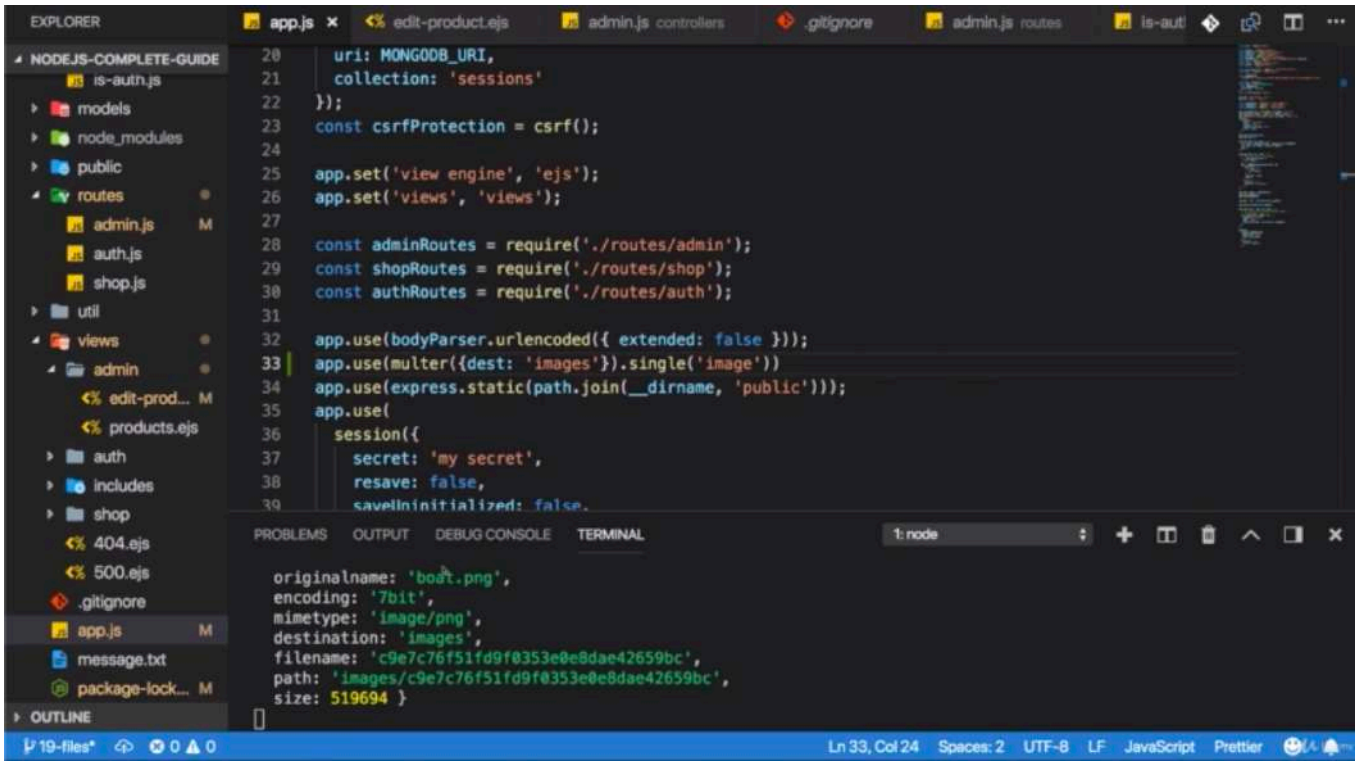




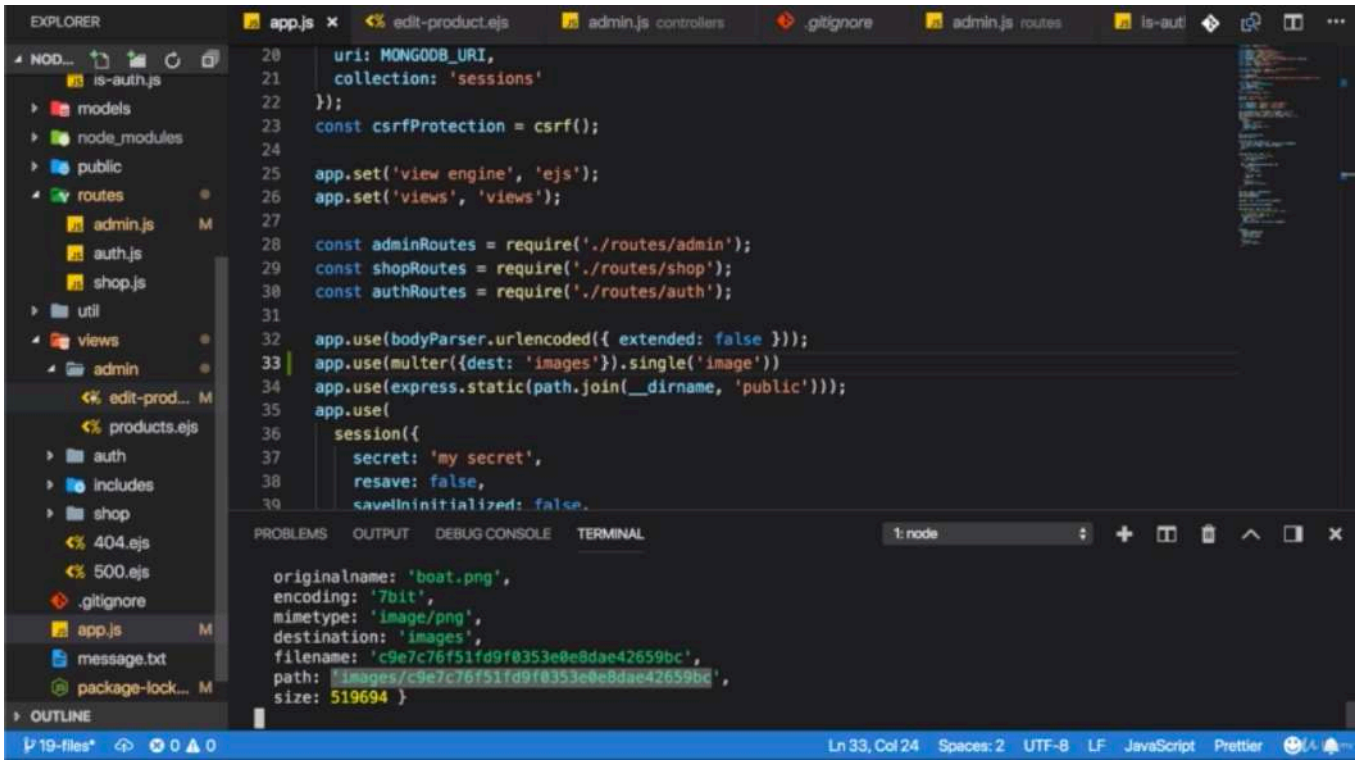
- what we see is that multer seems to have done something. it seems to have stored something in that file property on our request object.
- it stored the name of the field where it extracted that, it detected the file name, it detected the mimetype and the buffer which is how node handles the binary data. this is the result of the streamed data, the file was sent to our server as a stream or was handled as a stream to handle it if it was bigger and then this is the collected data in a buffer like a bus stop. it gives you a way of working with the stream data in this case it's the combined stream and data. and we work with the buffer to turn it into a file.



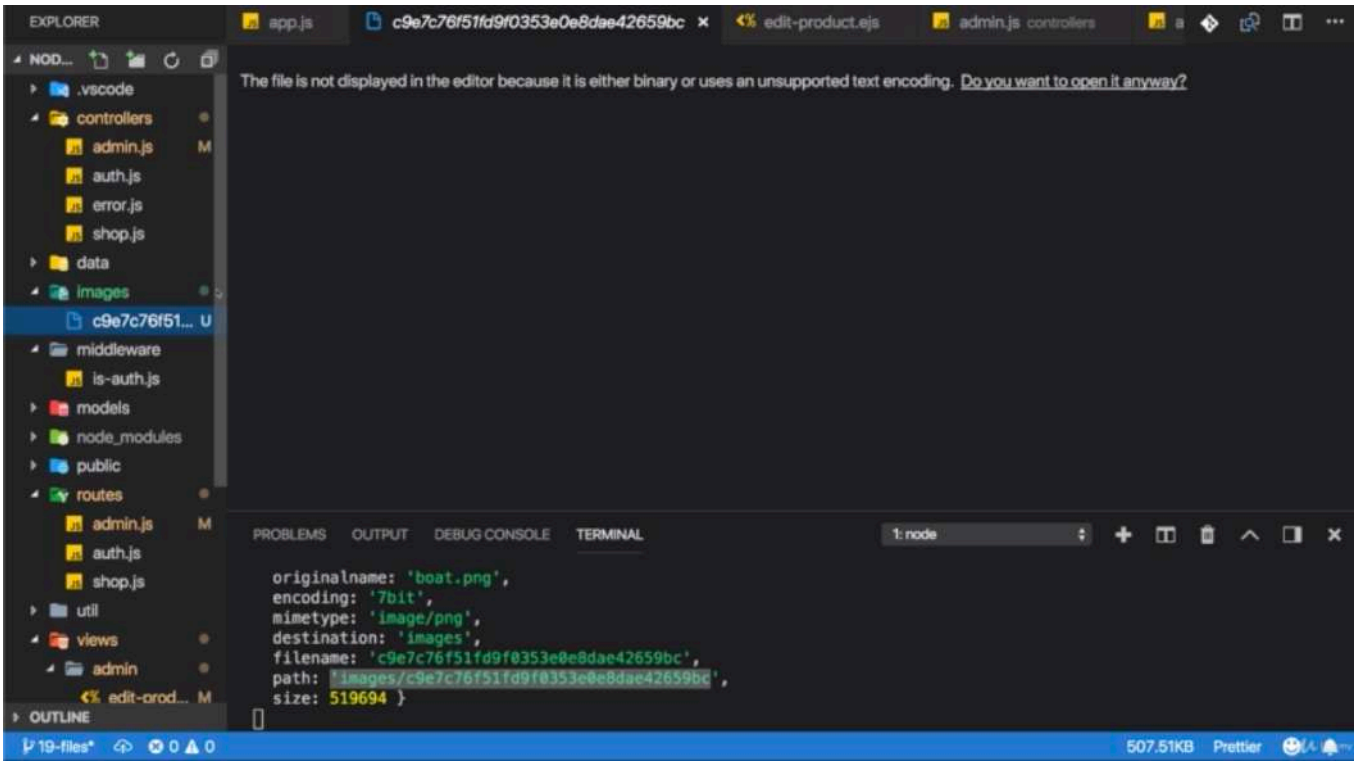
- we don't have the buffer because now multer is able to do something with the buffer, instead of just buffering it all in memory. it can turn that buffer back into binary data

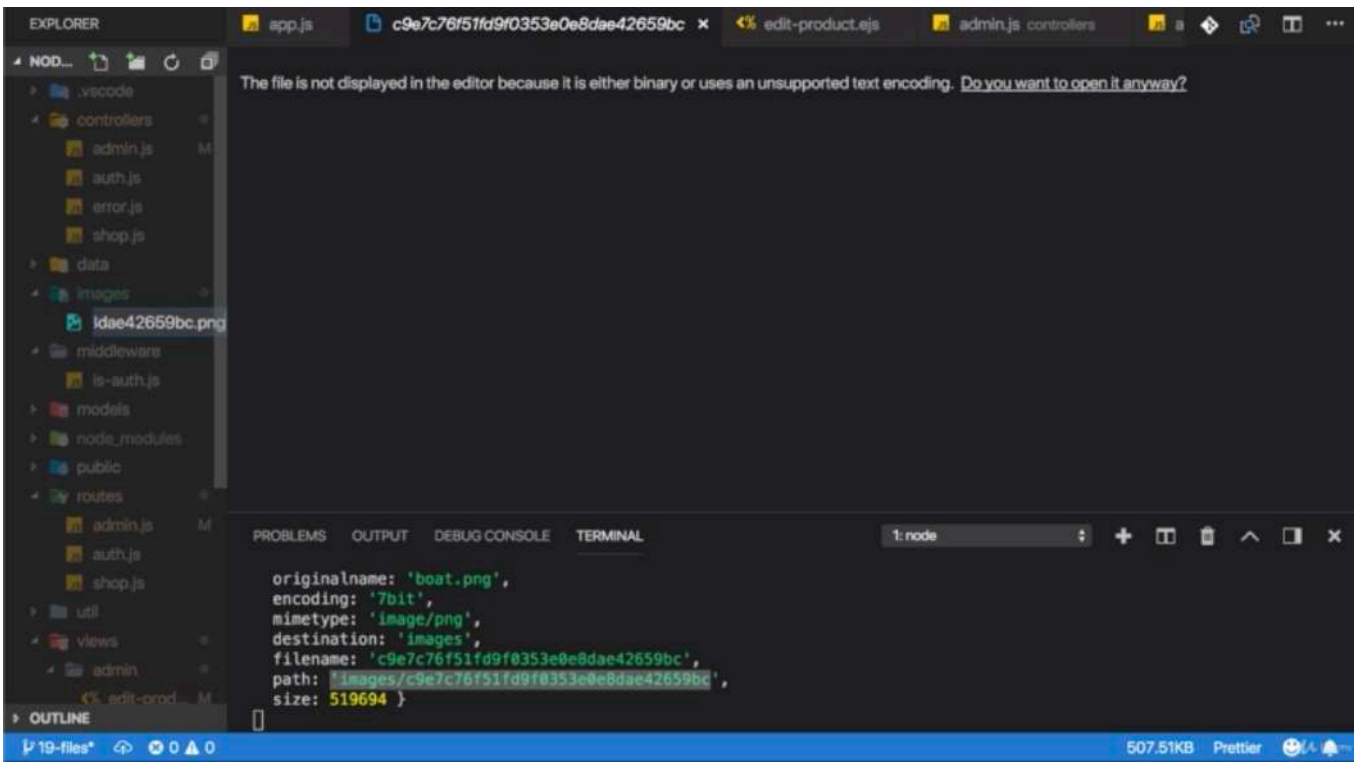


- and it stores it in this path here.



- and now if you have a look at your folder, you should not have an images folder with some file in there.
- now that file has some random hash name, doesn't have a file extension and isn't recognized as an image.


EXPLORER
app.js
c9e7c76f51fd9f0353e0e8dae42659bc
edit-product.ejs
admin.js controllers
The file is not displayed in the editor because it is either binary or uses an unsupported text encoding. Do you want to open it anyway?
.vscode
controllers
admin.js
auth.js
error.js
shop.js
data
images
c9e7c76f51...
middleware
is-auth.js
models
node_modules
public
routes
admin.js
auth.js
shop.js
util
views
admin
edit-prod...
OUTLINE
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
1: node
originalname: 'boat.png',
encoding: '7bit',
mimetype: 'image/png',
destination: 'images',
filename: 'c9e7c76f51fd9f0353e0e8dae42659bc',
path: 'images/c9e7c76f51fd9f0353e0e8dae42659bc',
size: 519694 }
19-files*
0 0
507.51KB
Prettier


EXPLORER
app.js
c9e7c76f51fd9f0353e0e8dae42659bc
edit-product.ejs
admin.js controllers
The file is not displayed in the editor because it is either binary or uses an unsupported text encoding. Do you want to open it anyway?
idae42659bc.png
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
1: node
originalname: 'boat.png',
encoding: '7bit',
mimetype: 'image/png',
destination: 'images',
filename: 'c9e7c76f51fd9f0353e0e8dae42659bc',
path: 'images/c9e7c76f51fd9f0353e0e8dae42659bc',
size: 519694 }
19-files*
0 0
507.51KB
Prettier

- but if i change that and i add '.png' in the end, this is the image i uploaded. so this works.

```
//app.js

const path = require('path');

const express = require('express');
const bodyParser = require('body-parser');
const mongoose = require('mongoose');
const session = require('express-session');
const MongoDBStore = require('connect-mongodb-session')(session);
const csrf = require('csurf');
const flash = require('connect-flash');
const multer = require('multer')

const errorController = require('./controllers/error');
const User = require('./models/user');

const MONGODB_URI =
'mongodb+srv://maximilian:rldnjs12@cluster0-z3vlk.mongodb.net/shop'

const app = express();
const store = new MongoDBStore({
  uri: MONGODB_URI,
  collection: 'sessions'
});
const csrfProtection = csrf();

app.set('view engine', 'ejs');
app.set('views', 'views');

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

/**'multer' is some middleware
```

```
 * which we execute on every incoming requests
 * and it then has a look at that request
 * and see if it's multipart form data
 * and tries to extract file if that is the case.
*/
app.use(bodyParser.urlencoded({ extended: false }));
/**i'm choosing 'image' here
 * because in ./views/admin/edit-product.ejs file,
 * this input name in file picker is named image
 *
 * 'dest: 'images' or 'dest: '/images' will add a product
 * and
 */
app.use(multer({dest: 'images'}).single('image'))
app.use(express.static(path.join(__dirname, 'public')));
app.use(
  session({
    secret: 'my secret',
    resave: false,
    saveUninitialized: false,
    store: store
  })
);
app.use(csrfProtection);
app.use(flash());

app.use((req, res, next) => {
  res.locals.isAuthenticated = req.session.isLoggedIn;
  res.locals.csrfToken = req.csrfToken();
  next();
});

app.use((req, res, next) => {
  //throw new Error('Sync Dummy')
  if (!req.session.user) {
    return next();
  }
  User.findById(req.session.user._id)
    .then(user => {
      if (!user) {
        return next();
      }
      req.user = user;
      next();
    })
    .catch(err => {
      next(new Error(err))
    });
});

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

app.get('/500', errorController.get500);

```

```
app.use(errorController.get404);

app.use((error, req, res, next) => {
  //res.redirect('/500')
  res.status(500).render('500', {
    pageTitle: 'Error!',
    path: '/500',
    isAuthenticated: req.session.isLoggedIn
  });
})

mongoose
  .connect(MONGODB_URI)
  .then(result => {
    app.listen(3000);
  })
  .catch(err => {
    console.log(err);
  });

```

```
// ./controllers/admin.js

const mongoose = require('mongoose');

const { validationResult } = require('express-validator/check');

const Product = require('../models/product');

exports.getAddProduct = (req, res, next) => {
  res.render('admin/edit-product', {
    pageTitle: 'Add Product',
    path: '/admin/add-product',
    editing: false,
    hasError: false,
    errorMessage: null,
    validationErrors: []
  });
};

exports.postAddProduct = (req, res, next) => {
  const title = req.body.title;
  const image = req.file;
  const price = req.body.price;
  const description = req.body.description;
  console.log(imageUrl)
  const errors = validationResult(req);

  if (!errors.isEmpty()) {
    console.log(errors.array());
    return res.status(422).render('admin/edit-product', {
      pageTitle: 'Add Product',
      path: '/admin/add-product',
      editing: false,
      hasError: true,
      product: {
        title: title,
```

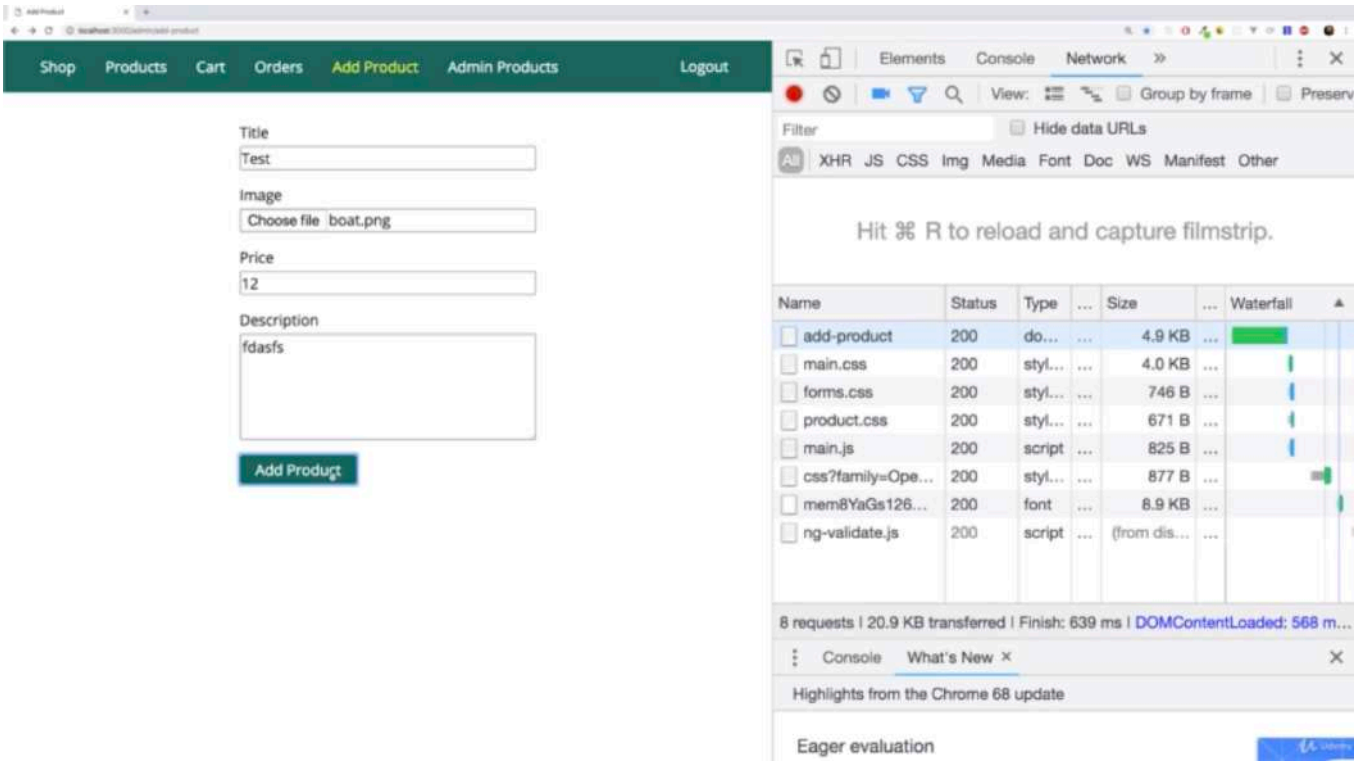
```
      imageUrl: imageUrl,
      price: price,
      description: description
    },
    errorMessage: errors.array()[0].msg,
    validationErrors: errors.array()
  });
}

const product = new Product({
  //_id: new mongoose.Types.ObjectId('5badf72403fd8b5be0366e81'),
  title: title,
  price: price,
  description: description,
  imageUrl: imageUrl,
  userId: req.user
});
product
  .save()
  .then(result => {
    // console.log(result);
    console.log('Created Product');
    res.redirect('/admin/products');
  })
  .catch(err => {
    // return res.status(500).render('admin/edit-product', {
    //   pageTitle: 'Add Product',
    //   path: '/admin/add-product',
    //   editing: false,
    //   hasError: true,
    //   product: {
    //     title: title,
    //     imageUrl: imageUrl,
    //     price: price,
    //     description: description
    //   },
    //   errorMessage: 'Database operation failed, please try again.',
    //   validationErrors: []
    // });
    //res.redirect('/500');
    const error = new Error(err)
    error.httpStatusCode = 500
    return next(error)
  });
};

exports.getEditProduct = (req, res, next) => {
  const editMode = req.query.edit;
  if (!editMode) {
    return res.redirect('/');
  }
  const prodId = req.params.productId;
  Product.findById(prodId)
    .then(product => {
      if (!product) {
        return res.redirect('/');
```

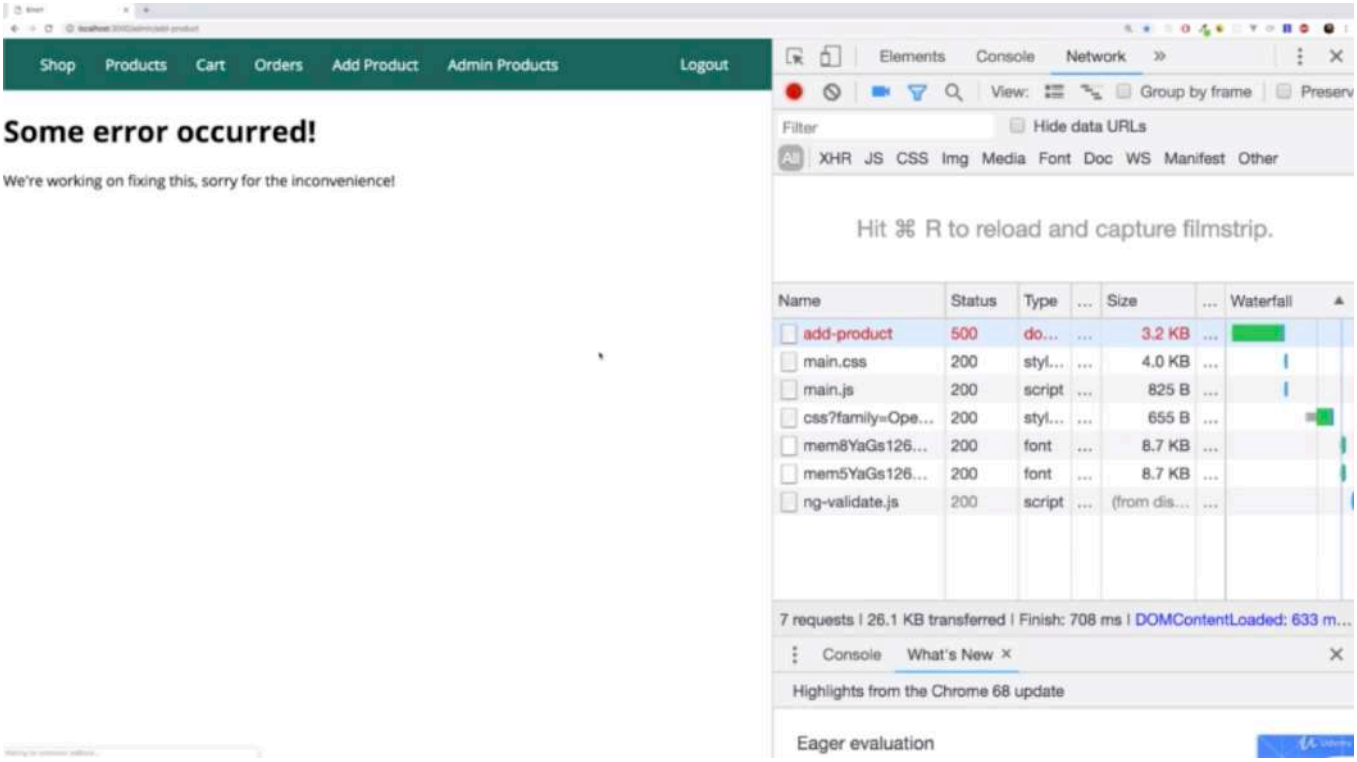
```
      }
      res.render('admin/edit-product', {
        pageTitle: 'Edit Product',
        path: '/admin/edit-product',
        editing: editMode,
        product: product,
        hasError: false,
        errorMessage: null,
        validationErrors: []
      });
    })
    .catch(err => {
      const error = new Error(err)
      error.httpStatusCode = 500
      return next(error)
    });
};

exports.postEditProduct = (req, res, next) => {
  const prodId = req.body.productId;
  const updatedTitle = req.body.title;
  const updatedPrice = req.body.price;
  const updatedImageUrl = req.body.imageUrl;
  const updatedDesc = req.body.description;

  const errors = validationResult(req);

  if (!errors.isEmpty()) {
    return res.status(422).render('admin/edit-product', {
      pageTitle: 'Edit Product',
      path: '/admin/edit-product',
      editing: true,
      hasError: true,
      product: {
        title: updatedTitle,
        imageUrl: updatedImageUrl,
        price: updatedPrice,
        description: updatedDesc,
        _id: prodId
      },
      errorMessage: errors.array()[0].msg,
      validationErrors: errors.array()
    });
  }

  Product.findById(prodId)
    .then(product => {
      if (product.userId.toString() !== req.user._id.toString()) {
        return res.redirect('/');
      }
      product.title = updatedTitle;
      product.price = updatedPrice;
      product.description = updatedDesc;
      product.imageUrl = updatedImageUrl;
      return product.save().then(result => {
        console.log('UPDATED PRODUCT!');
```

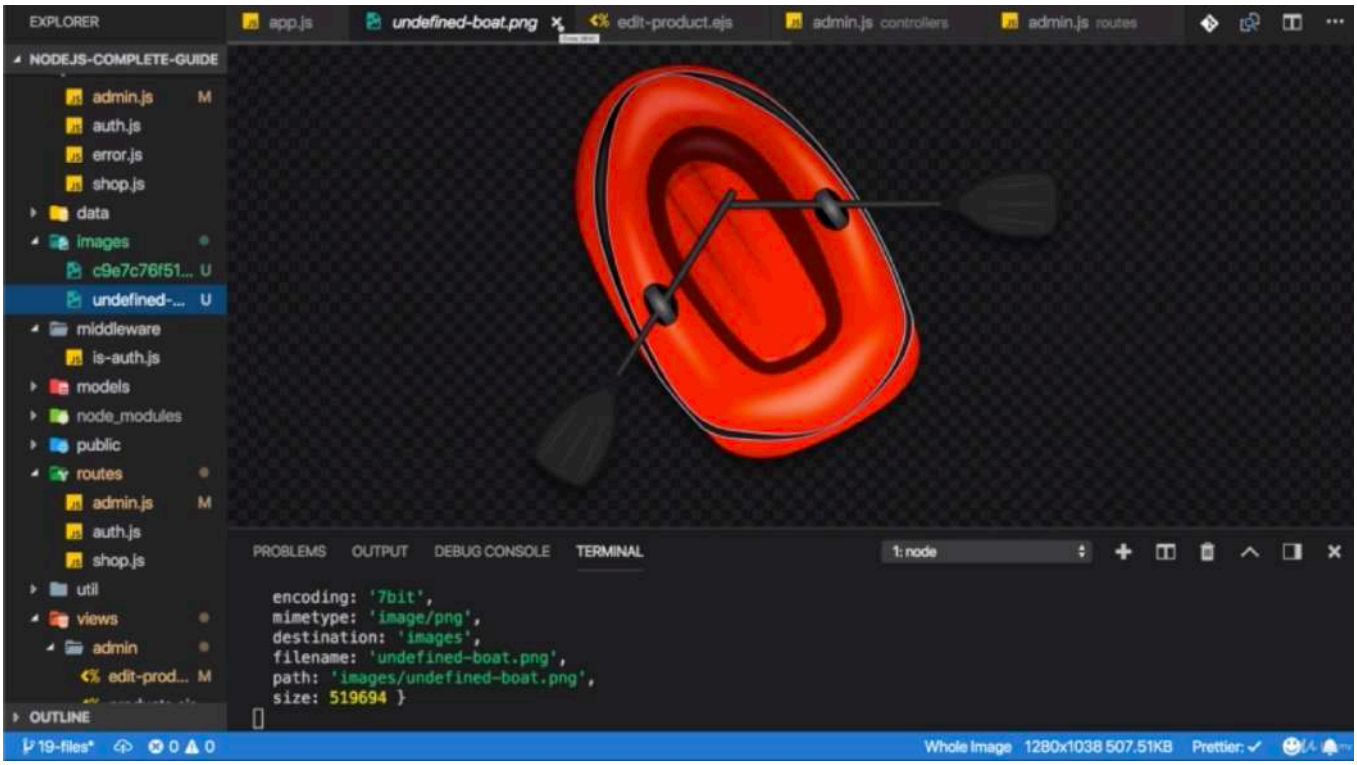
```
      res.redirect('/admin/products');
    });
  })
  .catch(err => {
    const error = new Error(err)
    error.httpStatusCode = 500
    return next(error)
  });
};

exports.getProducts = (req, res, next) => {
  Product.find({ userId: req.user._id })
    // .select('title price -_id')
    // .populate('userId', 'name')
    .then(products => {
      console.log(products);
      res.render('admin/products', {
        prods: products,
        pageTitle: 'Admin Products',
        path: '/admin/products'
      });
    })
    .catch(err => {
      const error = new Error(err)
      error.httpStatusCode = 500
      return next(error)
    });
};

exports.postDeleteProduct = (req, res, next) => {
  const prodId = req.body.productId;
  Product.deleteOne({ _id: prodId, userId: req.user._id })
    .then(() => {
      console.log('DESTROYED PRODUCT');
      res.redirect('/admin/products');
    })
    .catch(err => {
      const error = new Error(err)
      error.httpStatusCode = 500
      return next(error)
    });
};

```

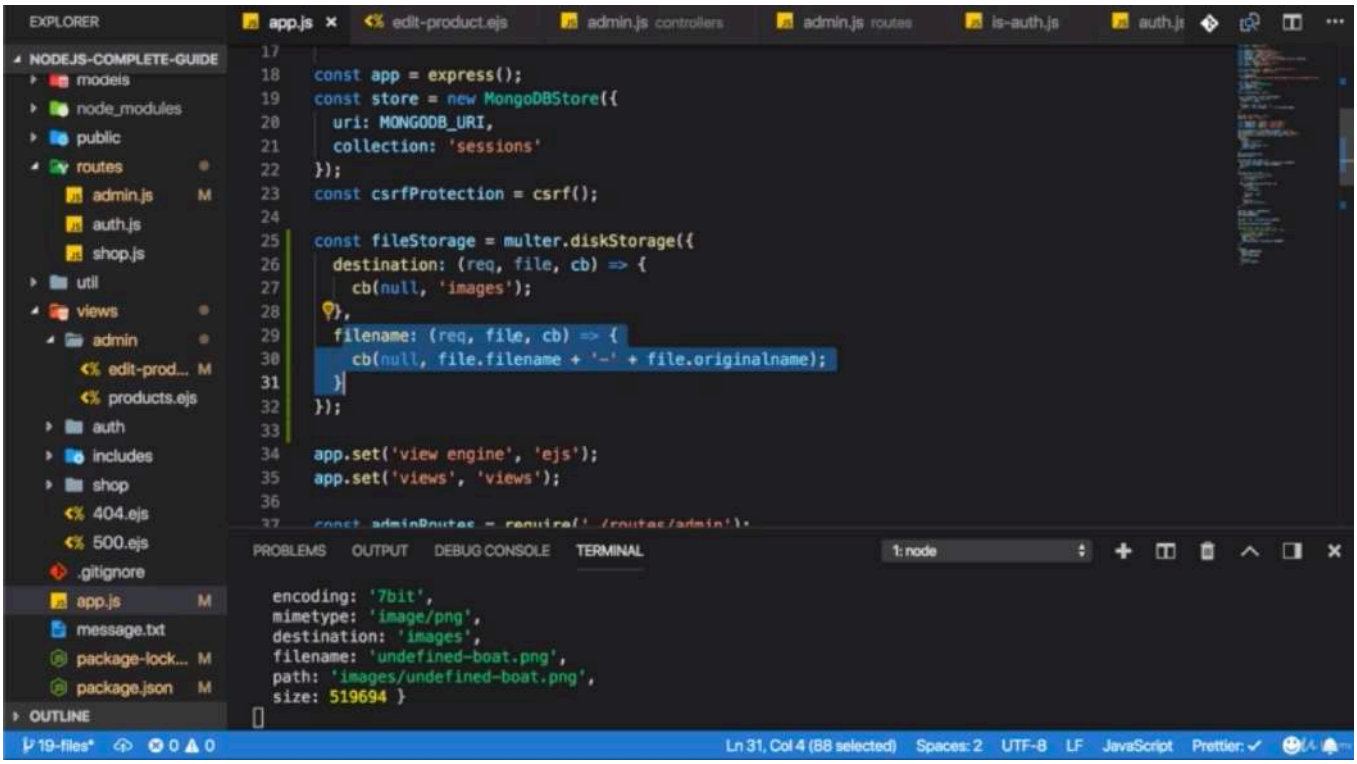
* Chapter 318: Configuring Multer To Adjust Filename & Filepath

1. update
- app.js

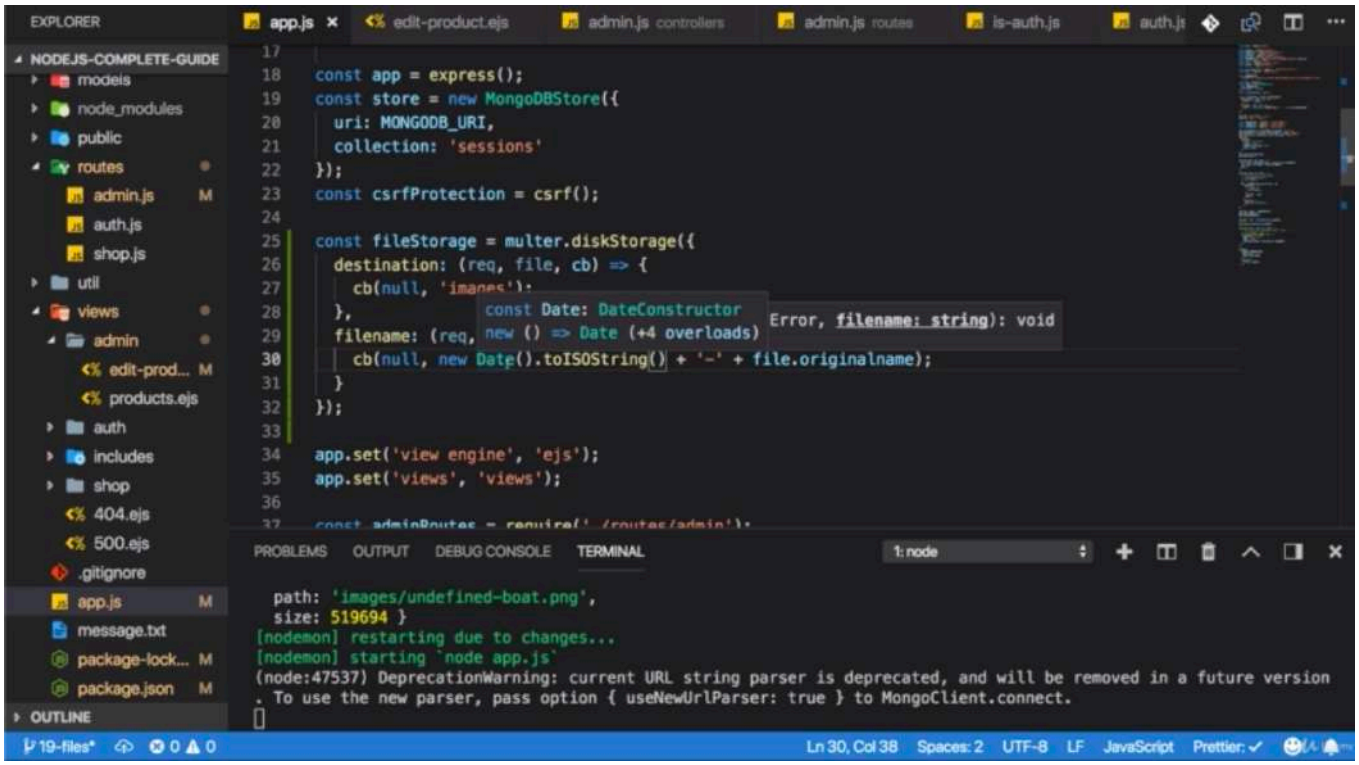

Shop
Products
Cart
Orders
Add Product
Admin Products
Logout
Title
Test
Image
Choose file boat.png
Price
12
Description
fdasfs
Add Product
Elements
Console
Network
Hit ⌘ R to reload and capture filmstrip.
Name
Status
Type
Size
Waterfall
add-product 200 do... 4.9 KB
main.css 200 styl... 4.0 KB
forms.css 200 styl... 746 B
product.css 200 styl... 671 B
main.js 200 script 825 B
css?family=Ope... 200 styl... 877 B
mem8YaGs126... 200 font 8.9 KB
ng-validate.js 200 script (from dis...
8 requests | 20.9 KB transferred | Finish: 639 ms | DOMContentLoaded: 568 m...
Console
What's New
Highlights from the Chrome 68 update
Eager evaluation


Shop
Products
Cart
Orders
Add Product
Admin Products
Logout
Some error occurred!
We're working on fixing this, sorry for the inconvenience!
Elements
Console
Network
Hit ⌘ R to reload and capture filmstrip.
Name
Status
Type
Size
Waterfall
add-product 500 do... 3.2 KB
main.css 200 styl... 4.0 KB
main.js 200 script 825 B
css?family=Ope... 200 styl... 655 B
mem8YaGs126... 200 font 8.7 KB
mem5YaGs126... 200 font 8.7 KB
ng-validate.js 200 script (from dis...
7 requests | 26.1 KB transferred | Finish: 708 ms | DOMContentLoaded: 633 m...
Console
What's New
Highlights from the Chrome 68 update
Eager evaluation





- you will see that in those images folder, you have undefined-boat.png instead of that random hash because since we did set our own filename with this function. multer doesn't genearte that hash.



- 'new Date().toISOString()' gives us a snapshop of the current date and that should also ensure uniqueness

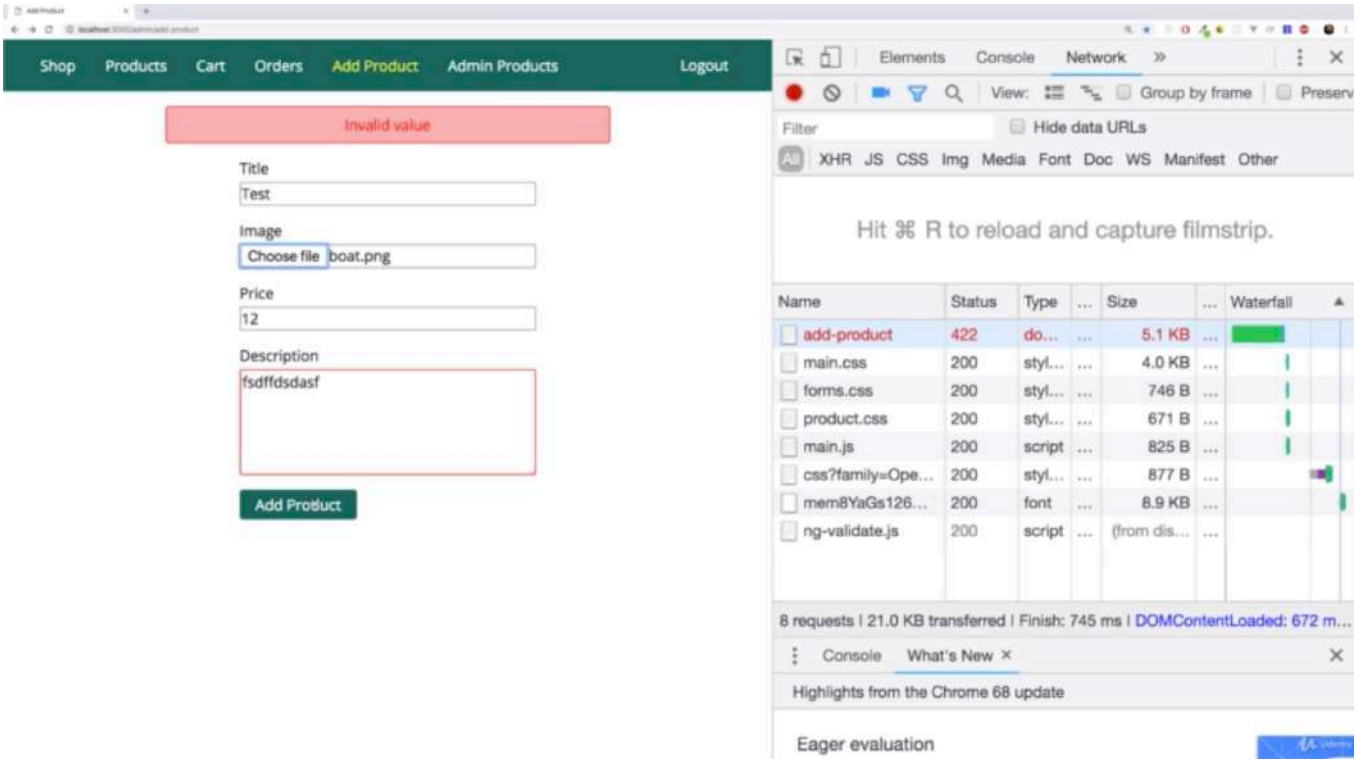
```
//app.js

const path = require('path');

const express = require('express');
const bodyParser = require('body-parser');
const mongoose = require('mongoose');
const session = require('express-session');
const MongoDBStore = require('connect-mongodb-session')(session);
const csrf = require('csurf');
const flash = require('connect-flash');
const multer = require('multer')

const errorController = require('./controllers/error');
const User = require('./models/user');

const MONGODB_URI =
'mongodb+srv://maximilian:rldnjs12@cluster0-z3vlk.mongodb.net/shop'

const app = express();
const store = new MongoDBStore({
  uri: MONGODB_URI,
  collection: 'sessions'
});
const csrfProtection = csrf();
/**'idskStorage()' is a storage engine
 * which you can use with multer
 * and pass the javascript object to configure that.
 *
 * these are 2 functions
 * which multer will call for an incoming file
 * these function then control how that file is handled
 * regarding the place where you store it and regarding the naming
 */
```

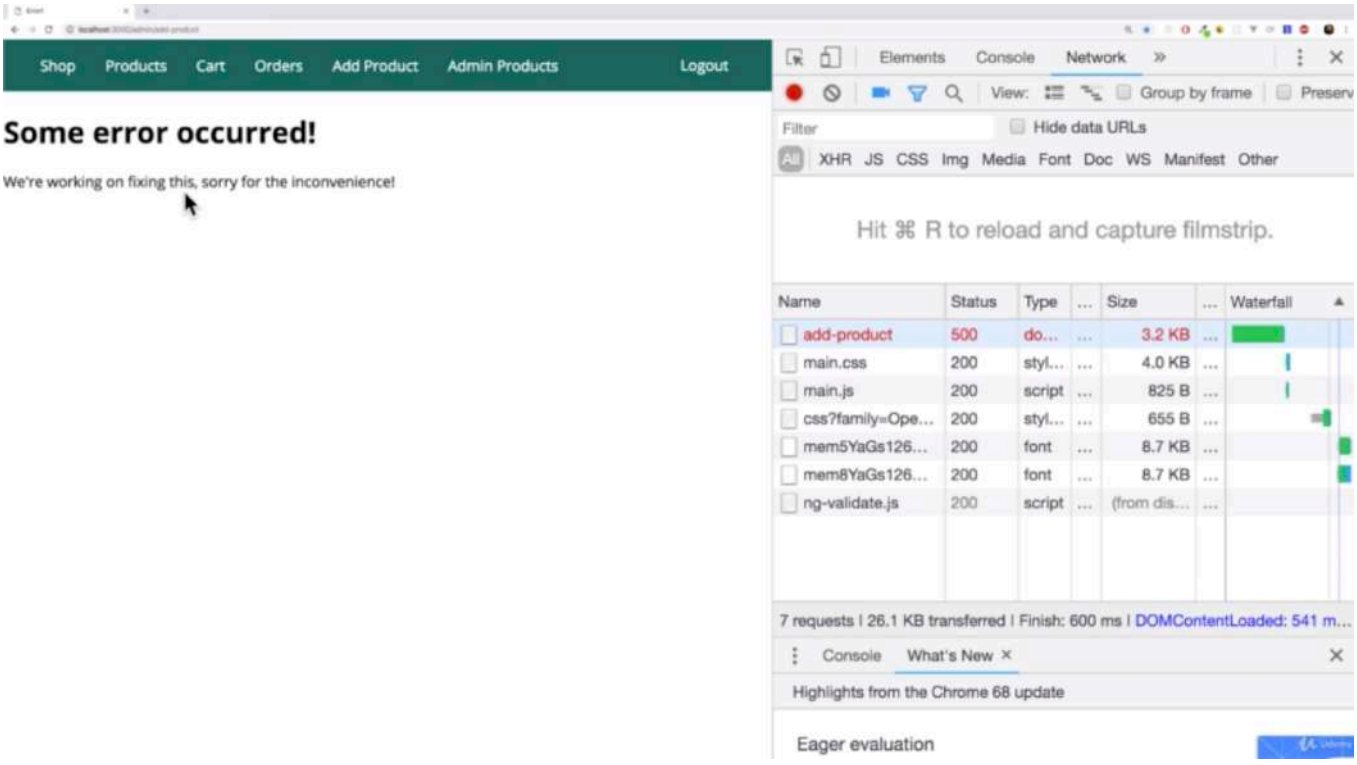
```
const fileStorage = multer.diskStorage({
  destination: (req, file, cb) => {
    /**the first argument is 'null'
     * which would be error message you throw to inform mulber
     * that something is wrong with that incoming file it should not store it
     * but if that is null, you tell multer that it's OK to store it
     *
     * the 2nd argument is the place
     * where you wanna store.
     */
    cb(null, 'images')
  },
  /**'filename' is a function
   *
   * and we can extract file
   * and then there is an 'originalname' object
   * which holds the original file name
   * there's also is a file name object
   * which holds this random hash.
   *
   * if we have 2 images with the same name,
   * they don't overwrite each other
   * so i will use that 'file.filename'
   * which is auto-generated by multer
   * and i will concatenate it
   * with a dash in-between with the originalname
   * and i also ensure that i end with the extension
   * and i set my own filename by calling that callback.
   */
  filename: (req, file, cb) => {
    cb(null, new Date().toISOString() + '-' + file.originalname)
  }
})

app.set('view engine', 'ejs');
app.set('views', 'views');

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

app.use(bodyParser.urlencoded({ extended: false }));
/**we can set the 'storage' key
 * which gives us way more configuration options thatn just the 'dest' option
 *
 * now we need to inform multer
 * that we wanna use this storage engine
 * and we we do this by setting 'fileStorage' as a value for the storage key in the multer
options
 */
app.use(multer({ storage: fileStorage }).single('image'))
app.use(express.static(path.join(__dirname, 'public')));
app.use(
  session({
    secret: 'my secret',
    resave: false,
```

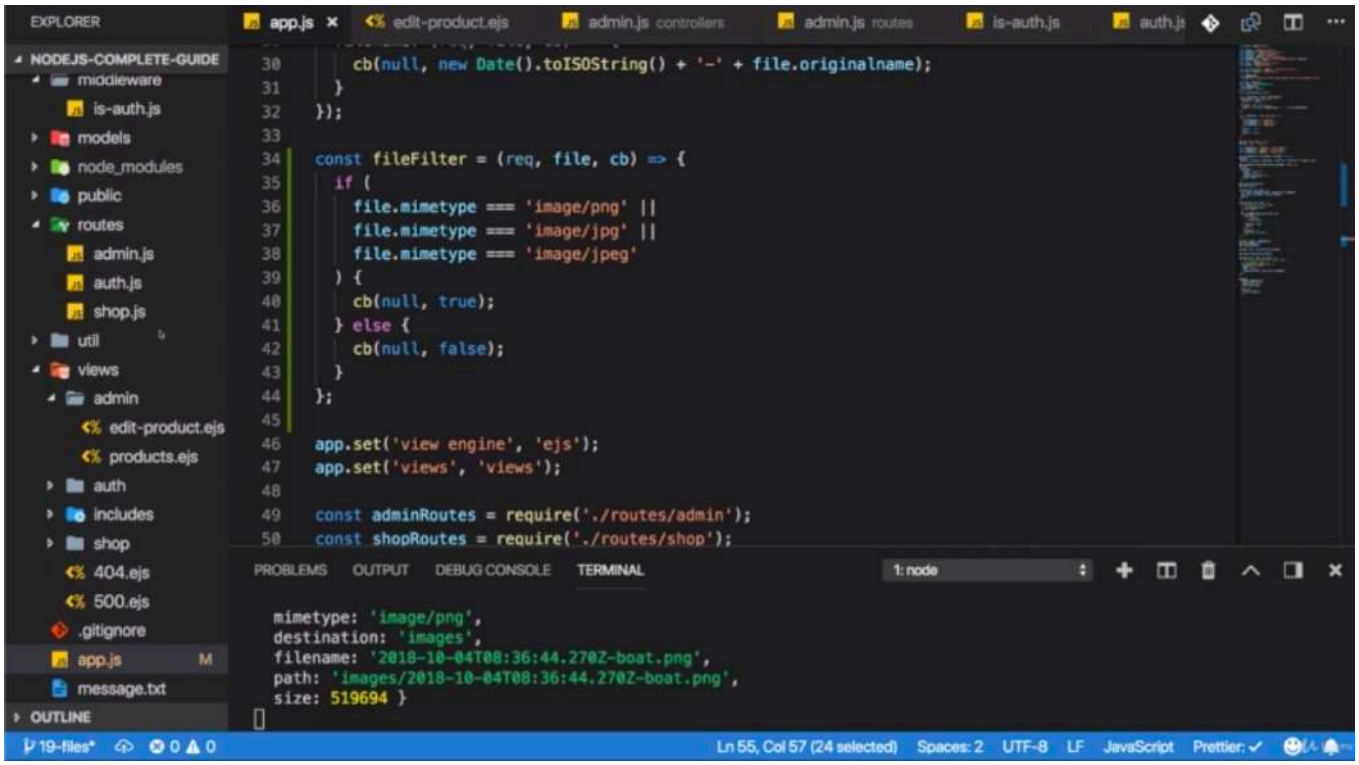
```
    saveUninitialized: false,
    store: store
  })
);
app.use(csrfProtection);
app.use(flash());

app.use((req, res, next) => {
  res.locals.isAuthenticated = req.session.isLoggedIn;
  res.locals.csrfToken = req.csrfToken();
  next();
});

app.use((req, res, next) => {
  //throw new Error('Sync Dummy')
  if (!req.session.user) {
    return next();
  }
  User.findById(req.session.user._id)
    .then(user => {
      if (!user) {
        return next();
      }
      req.user = user;
      next();
    })
    .catch(err => {
      next(new Error(err))
    });
});

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

app.get('/500', errorController.get500);

app.use(errorController.get404);

app.use((error, req, res, next) => {
  //res.redirect('/500')
  res.status(500).render('500', {
    pageTitle: 'Error!',
    path: '/500',
    isAuthenticated: req.session.isLoggedIn
  });
})

mongoose
  .connect(MONGODB_URI)
  .then(result => {
    app.listen(3000);
  })
  .catch(err => {
    console.log(err);
  });
```

* Chapter 319: Filtering Files By Mimetype

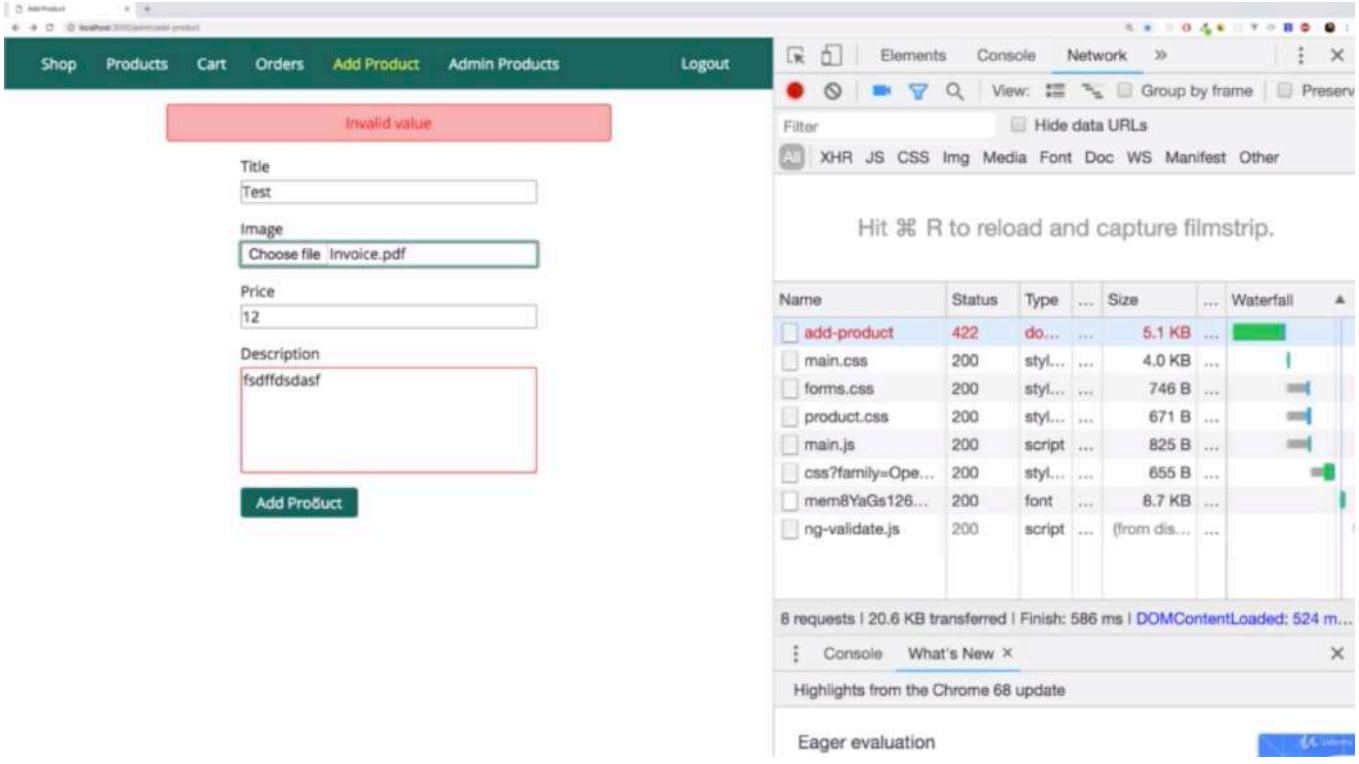
1. update
- app.js

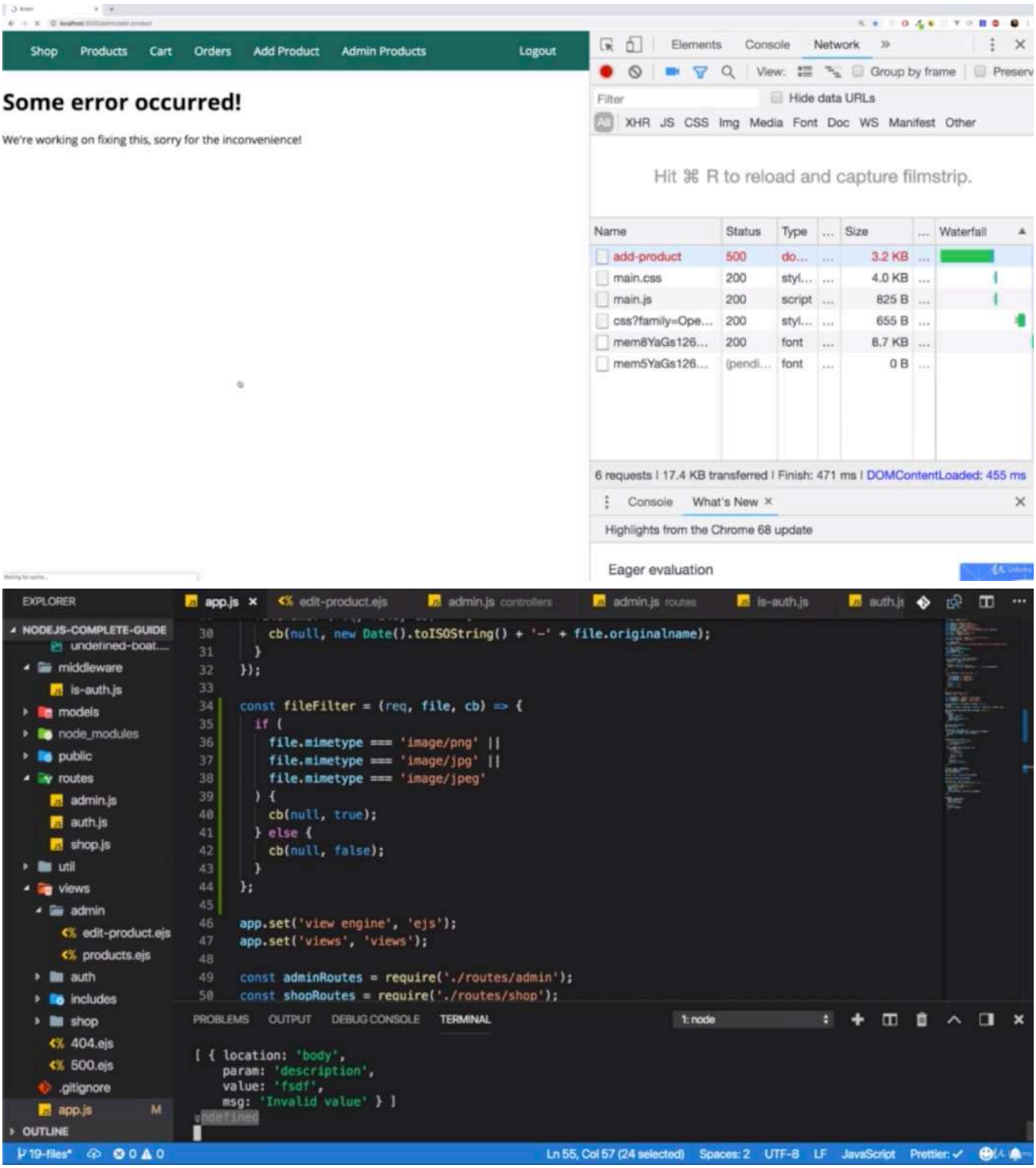




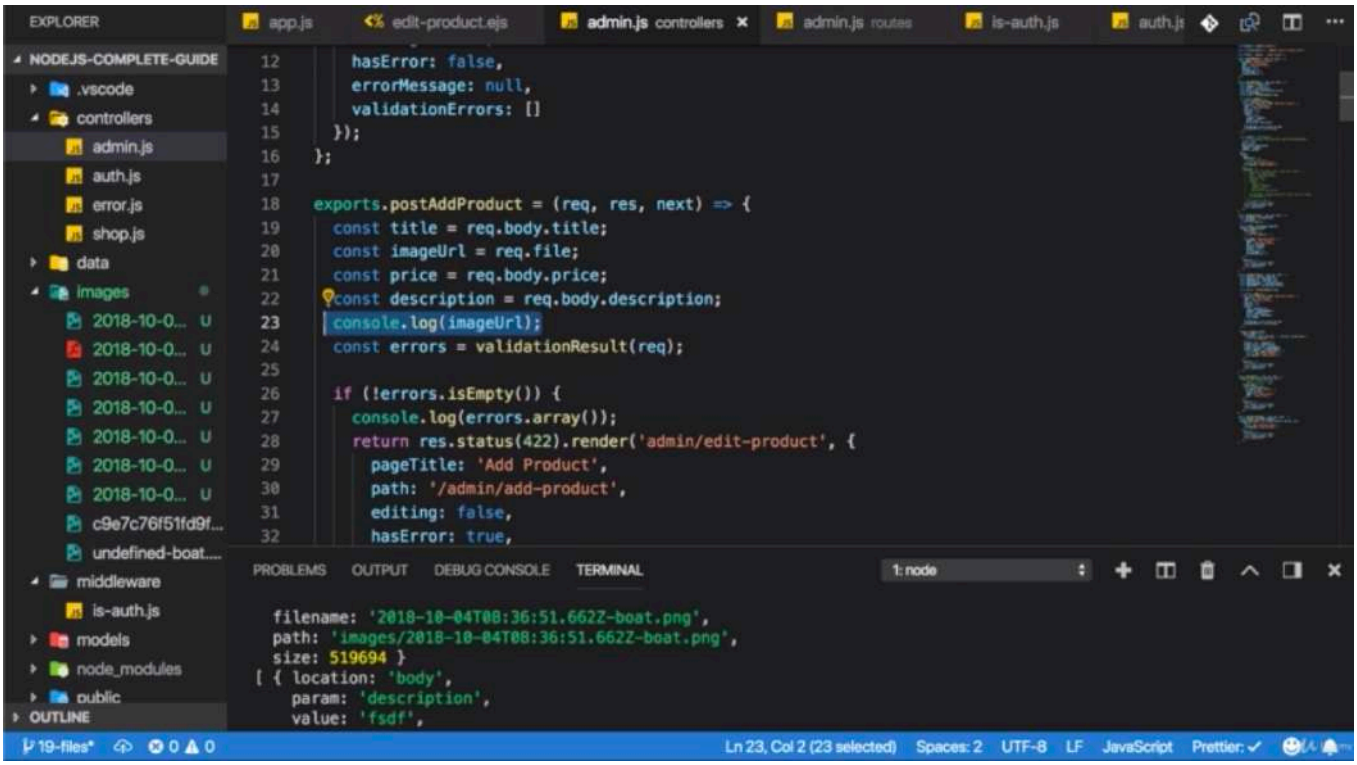


- if i submit a form with 'boat.png', i succeed, i get an error but i succeed. because i stores the file





- but if i try to submit a different value like pdf file, you can see i only get undefined



- and that undefined is stemming from my ./controllers/admin.js file

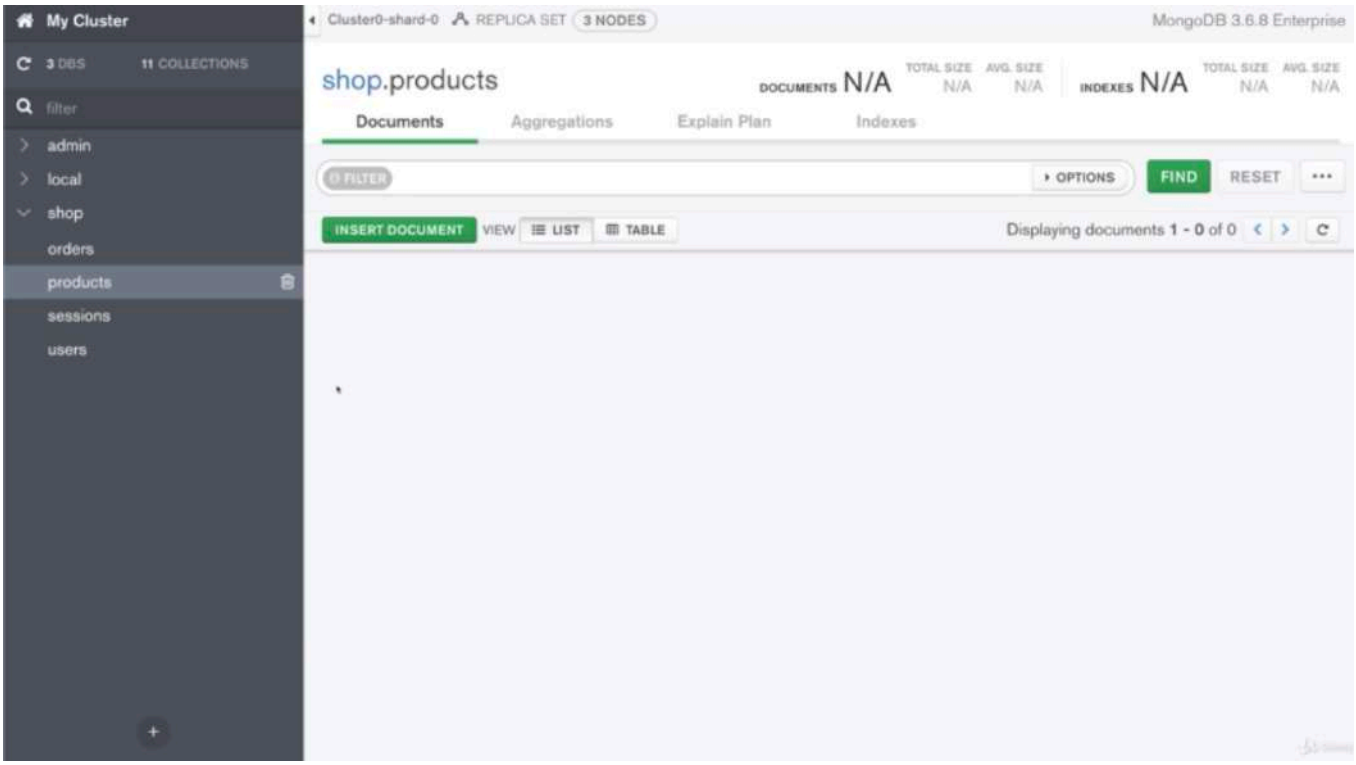
```
//app.js

const path = require('path');

const express = require('express');
const bodyParser = require('body-parser');
const mongoose = require('mongoose');
const session = require('express-session');
const MongoDBStore = require('connect-mongodb-session')(session);
const csrf = require('csurf');
const flash = require('connect-flash');
const multer = require('multer')

const errorController = require('./controllers/error');
const User = require('./models/user');

const MONGODB_URI =
'mongodb+srv://maximilian:rldnjs12@cluster0-z3vlk.mongodb.net/shop'

const app = express();
const store = new MongoDBStore({
  uri: MONGODB_URI,
  collection: 'sessions'
});
const csrfProtection = csrf();
const fileStorage = multer.diskStorage({
  destination: (req, file, cb) => {
    cb(null, 'images')
  },
  filename: (req, file, cb) => {
    cb(null, new Date().toISOString() + '-' + file.originalname)
  }
})

```

```
const fileFilter = (req, file, cb) => {
  if (
    file.mimetype === 'image/png' ||
    file.mimetype === 'image/jpg' ||
    file.mimetype === 'image/jpeg'
    ) {
    /**'null' as an error
     * and true if we wanna accept that file it should be stored
     * so false if we don't wanna accept and store that file.
     */
    cb(null, true)
  } else {
    cb(null, false)
  }
}

app.set('view engine', 'ejs');
app.set('views', 'views');

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

app.use(bodyParser.urlencoded({ extended: false }));
app.use(multer({ storage: fileStorage, fileFilter }).single('image'))
app.use(express.static(path.join(__dirname, 'public')));
app.use(
  session({
    secret: 'my secret',
    resave: false,
    saveUninitialized: false,
    store: store
  })
);
app.use(csrfProtection);
app.use(flash());

app.use((req, res, next) => {
  res.locals.isAuthenticated = req.session.isLoggedIn;
  res.locals.csrfToken = req.csrfToken();
  next();
});

app.use((req, res, next) => {
  //throw new Error('Sync Dummy')
  if (!req.session.user) {
    return next();
  }
  User.findById(req.session.user._id)
    .then(user => {
      if (!user) {
        return next();
      }
      req.user = user;
      next();
    })
```

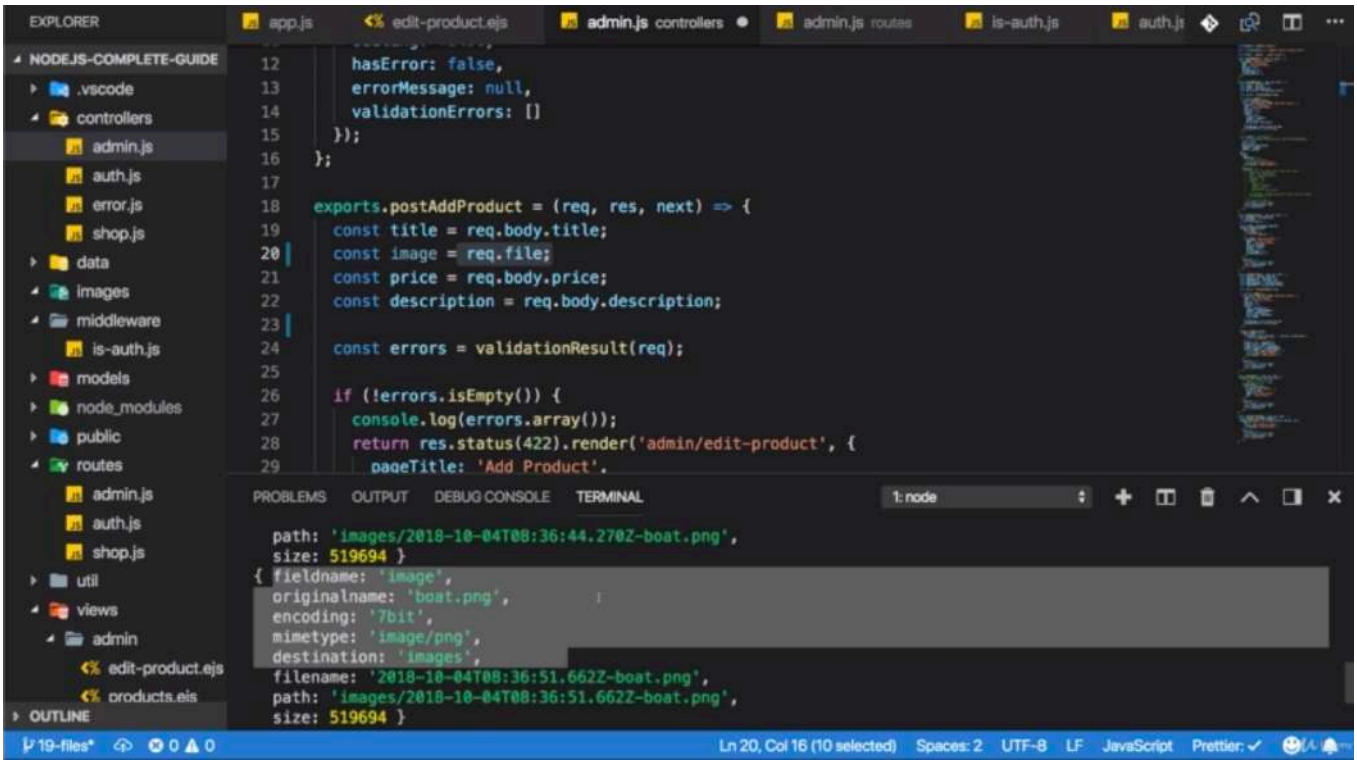
```
    .catch(err => {
      next(new Error(err))
    });
});

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

app.get('/500', errorController.get500);

app.use(errorController.get404);

app.use((error, req, res, next) => {
  //res.redirect('/500')
  res.status(500).render('500', {
    pageTitle: 'Error!',
    path: '/500',
    isAuthenticated: req.session.isLoggedIn
  });
})

mongoose
  .connect(MONGODB_URI)
  .then(result => {
    app.listen(3000);
  })
  .catch(err => {
    console.log(err);
  });

```

* Chapter 320: Storing File Data In The Database

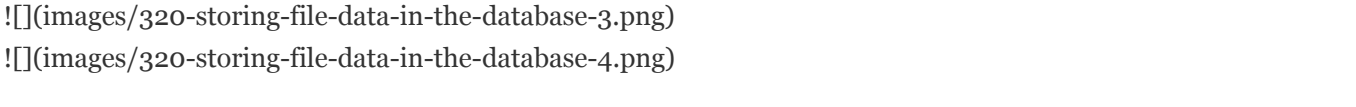
1. update
- ./controllers/admin.js
- ./routes/admin.js

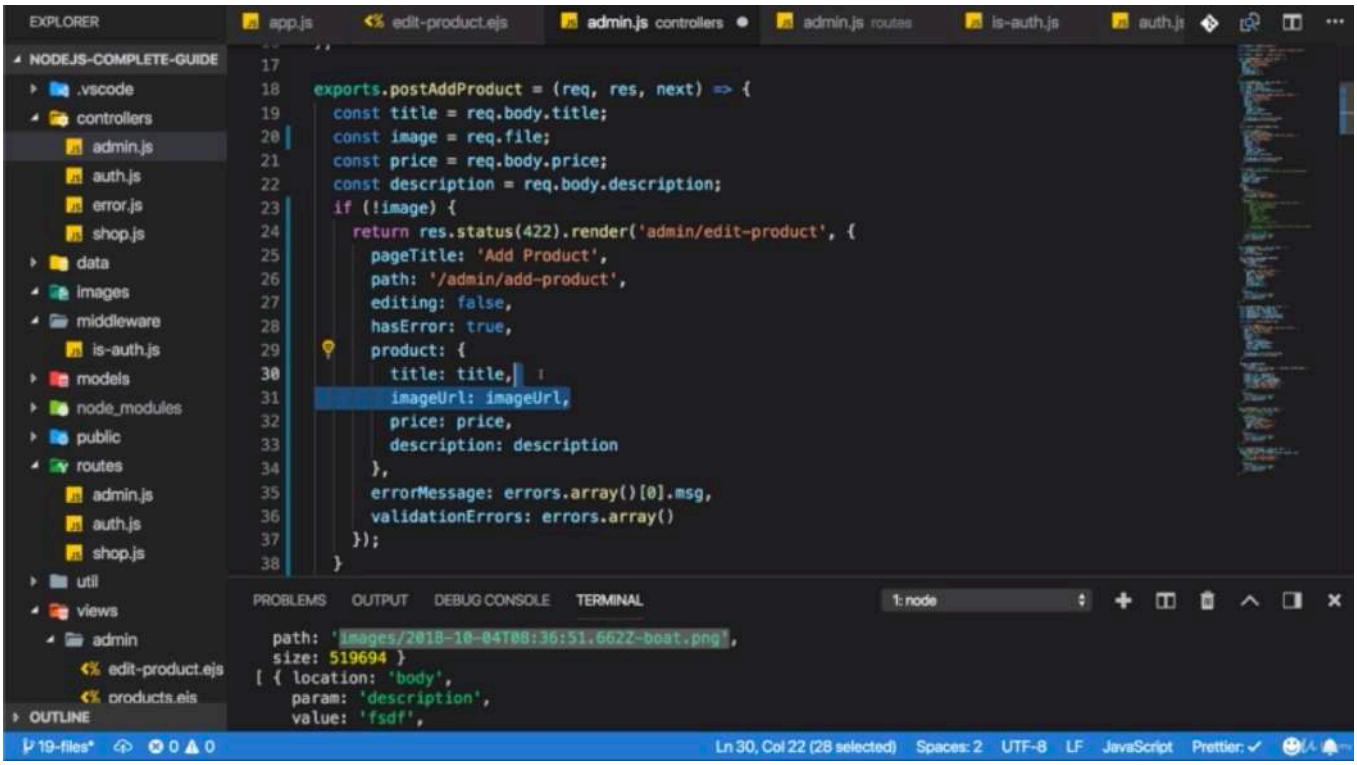


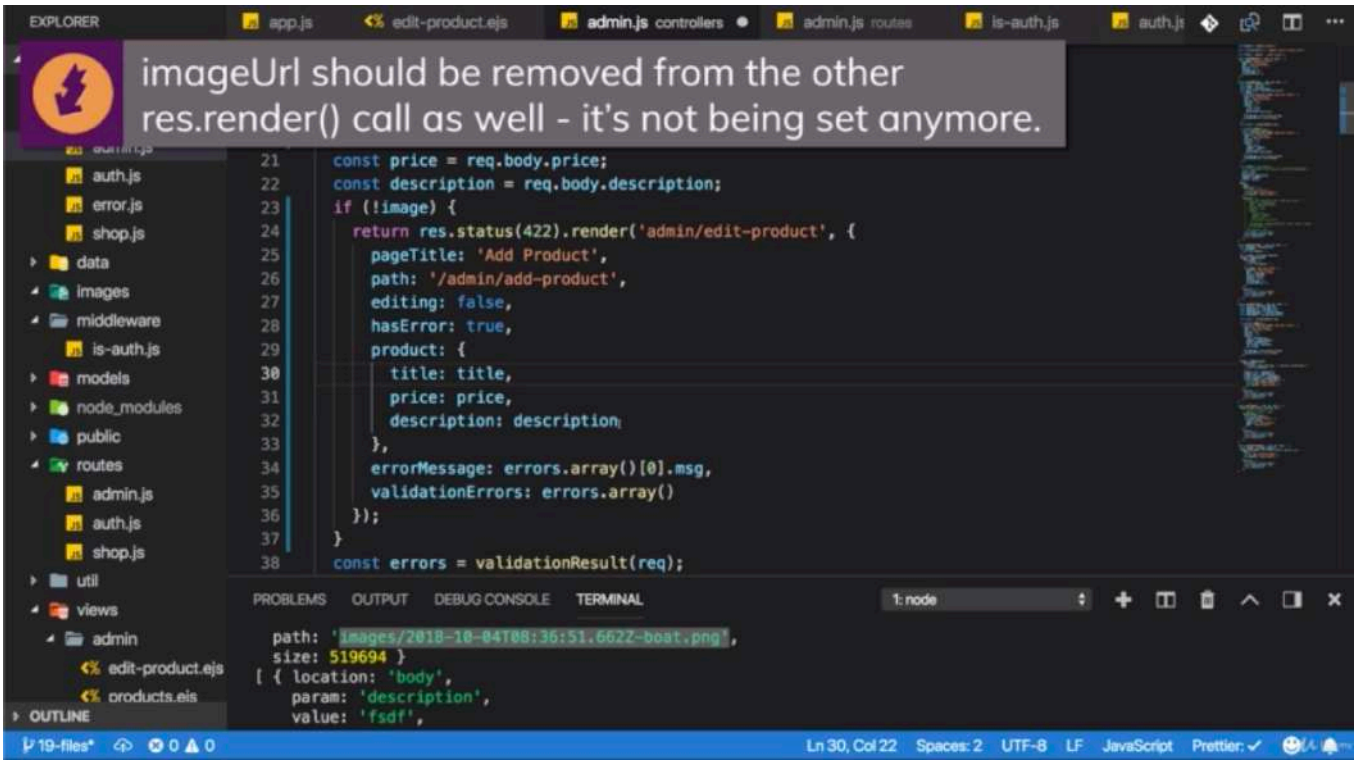
- i'm gonna remove all these products in database.



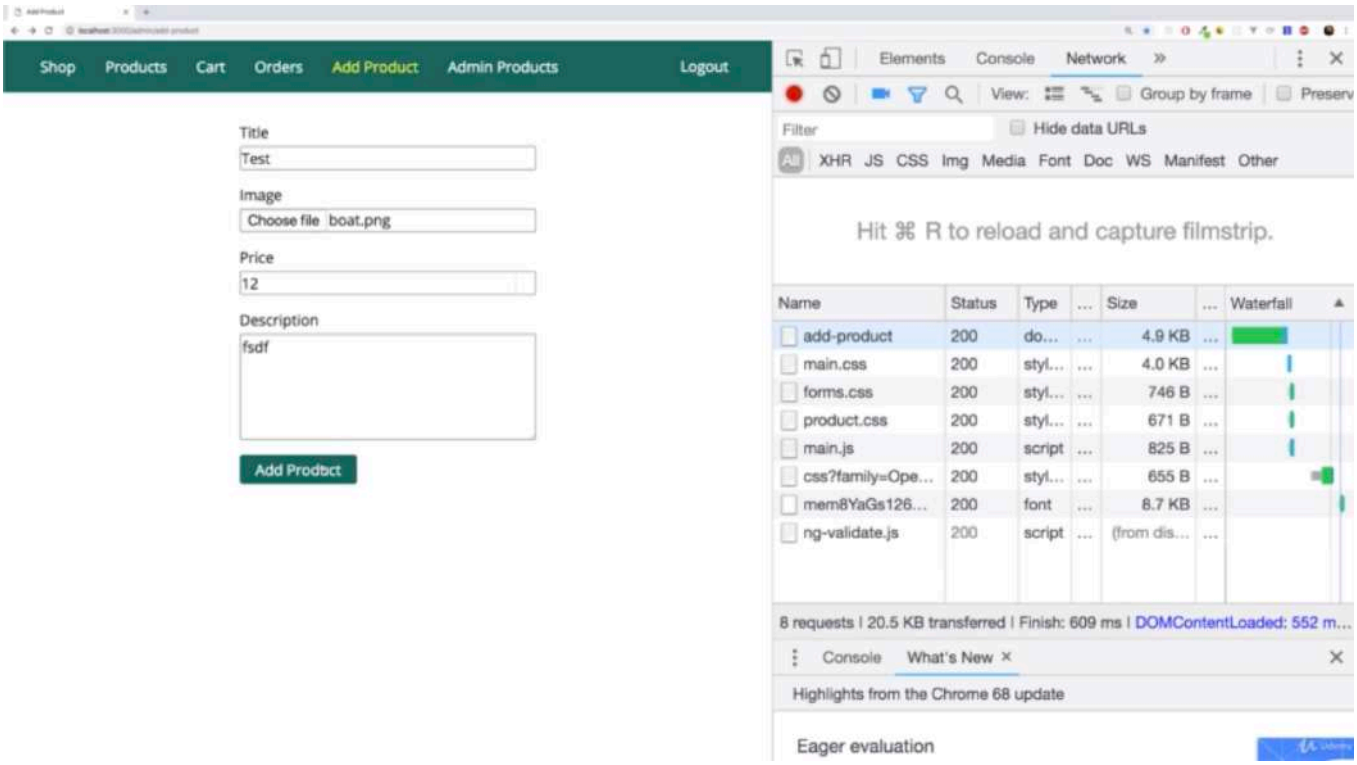
- first of all, image will be an object of this format with information about the file and where it was stored. so where the physical file can be found and i wanna save that.

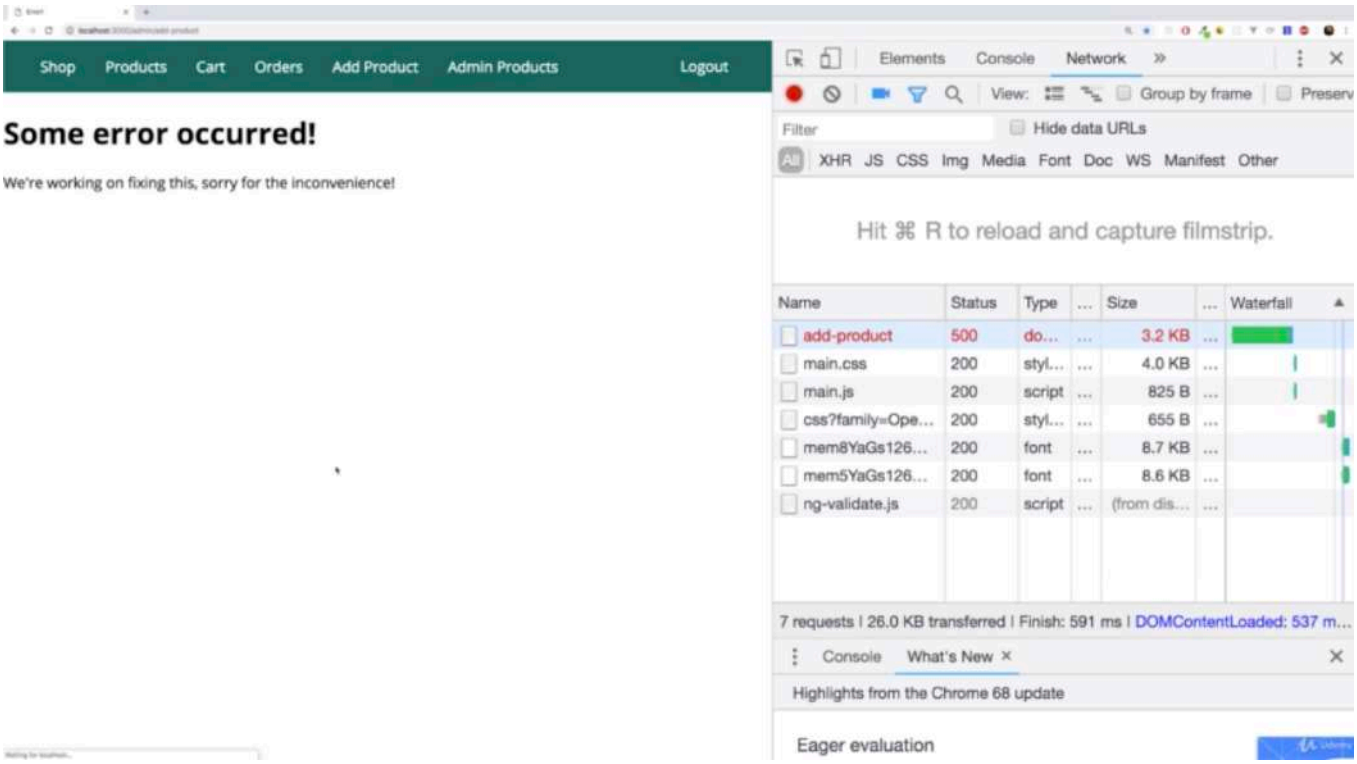



EXPLORER
app.js
edit-product.ejs
admin.js controllers
admin.js routes
is-auth.js
auth.js
NODEJS-COMPLETE-GUIDE
.vscode
controllers
admin.js
auth.js
error.js
shop.js
data
images
middleware
is-auth.js
models
node_modules
public
routes
admin.js
auth.js
shop.js
util
views
admin
edit-product.ejs
products.ejs
OUTLINE
exports.postAddProduct = (req, res, next) => {
const title = req.body.title;
const image = req.file;
const price = req.body.price;
const description = req.body.description;
if (!image) {
return res.status(422).render('admin/edit-product', {
pageTitle: 'Add Product',
path: '/admin/add-product',
editing: false,
hasError: true,
product: {
title: title,
imageUrl: imageUrl,
price: price,
description: description
},
errorMessage: errors.array()[0].msg,
validationErrors: errors.array()
});
}
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
1: node
path: 'images/2018-10-04T08:36:51.662Z-boat.png',
size: 519694 }
[{ location: 'body',
param: 'description',
value: 'fsdf',
19-files*
Ln 30, Col 22 (28 selected)
Spaces: 2
UTF-8
LF
JavaScript
Prettier:

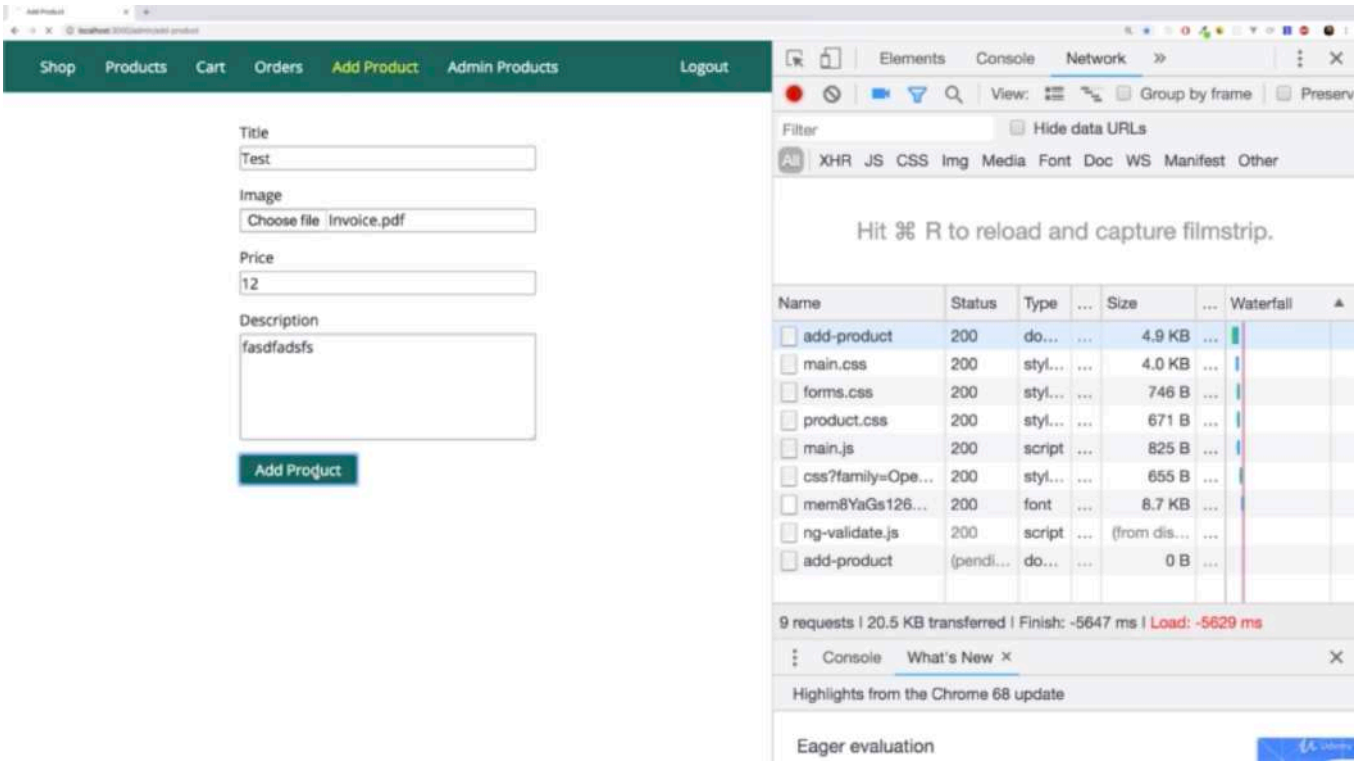

imageUrl should be removed from the other res.render() call as well - it's not being set anymore.
EXPLORER
app.js
edit-product.ejs
admin.js controllers
admin.js routes
is-auth.js
auth.js
const price = req.body.price;
const description = req.body.description;
if (!image) {
return res.status(422).render('admin/edit-product', {
pageTitle: 'Add Product',
path: '/admin/add-product',
editing: false,
hasError: true,
product: {
title: title,
price: price,
description: description
},
errorMessage: errors.array()[0].msg,
validationErrors: errors.array()
});
}
const errors = validationResult(req);
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
1: node
path: 'images/2018-10-04T08:36:51.662Z-boat.png',
size: 519694 }
[{ location: 'body',
param: 'description',
value: 'fsdf',
19-files*
Ln 30, Col 22
Spaces: 2
UTF-8
LF
JavaScript
Prettier:

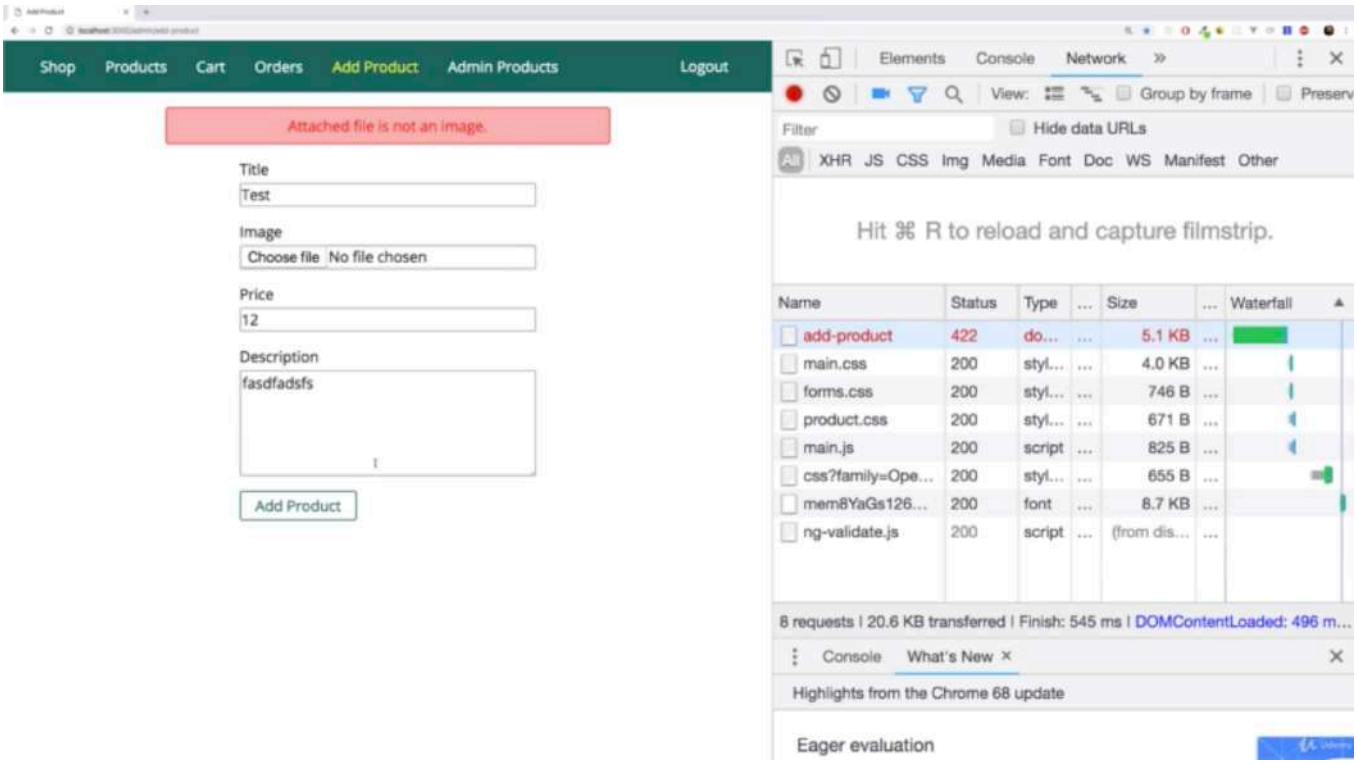
——————————————————————————



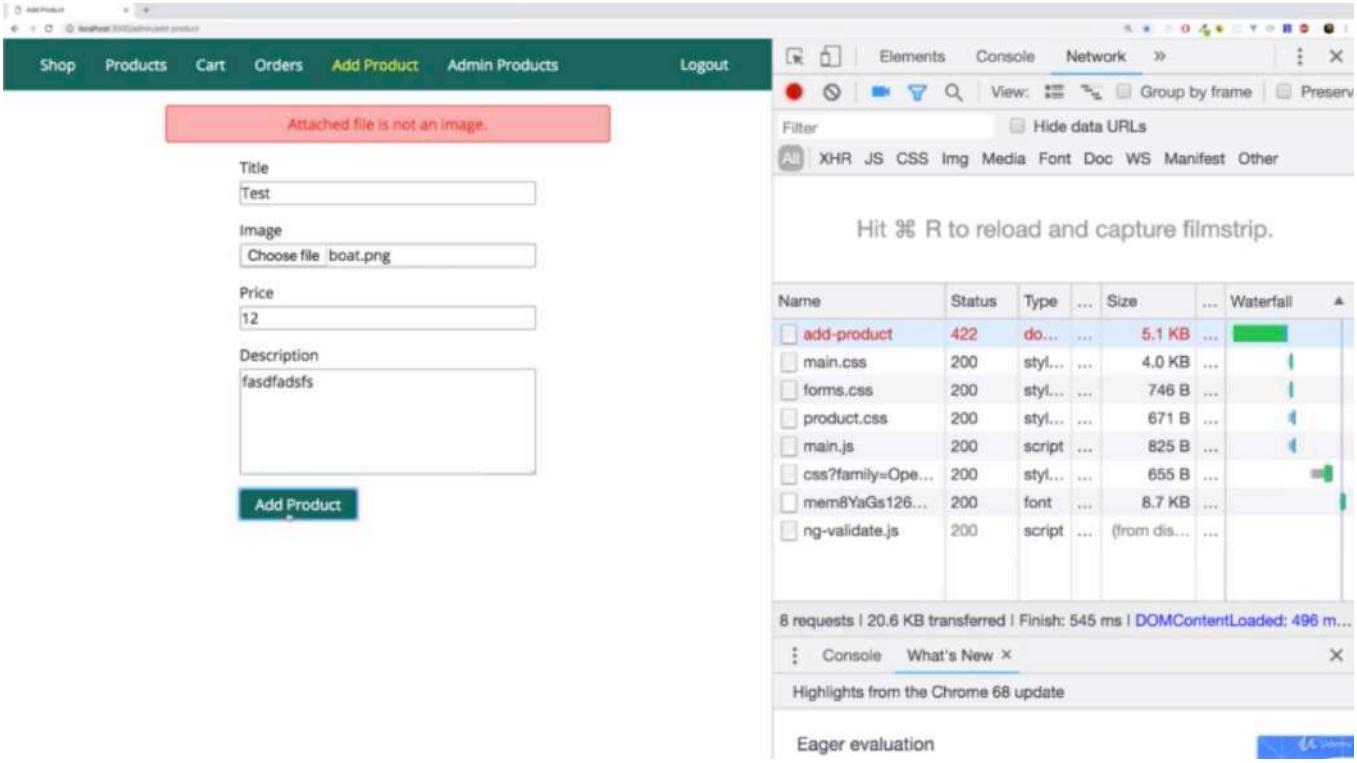


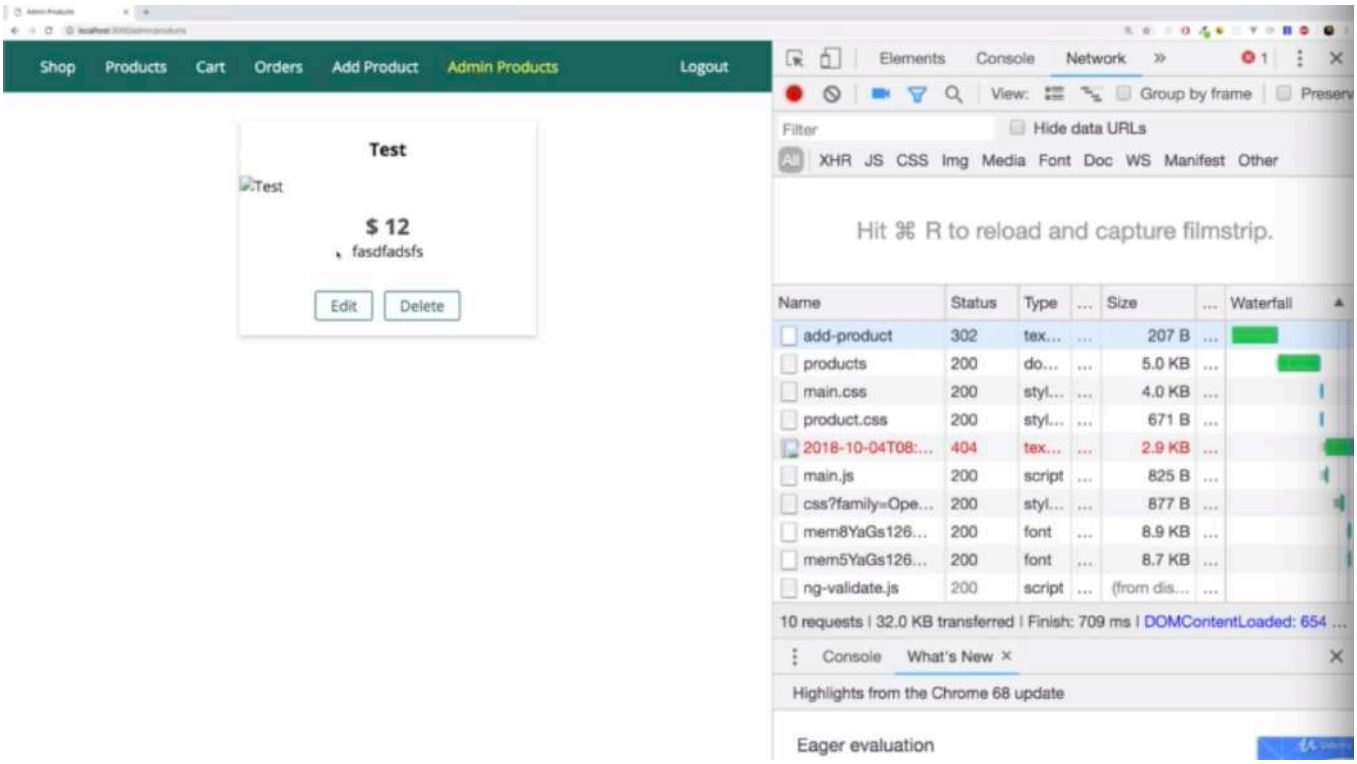
- if i upload an image, i succeed



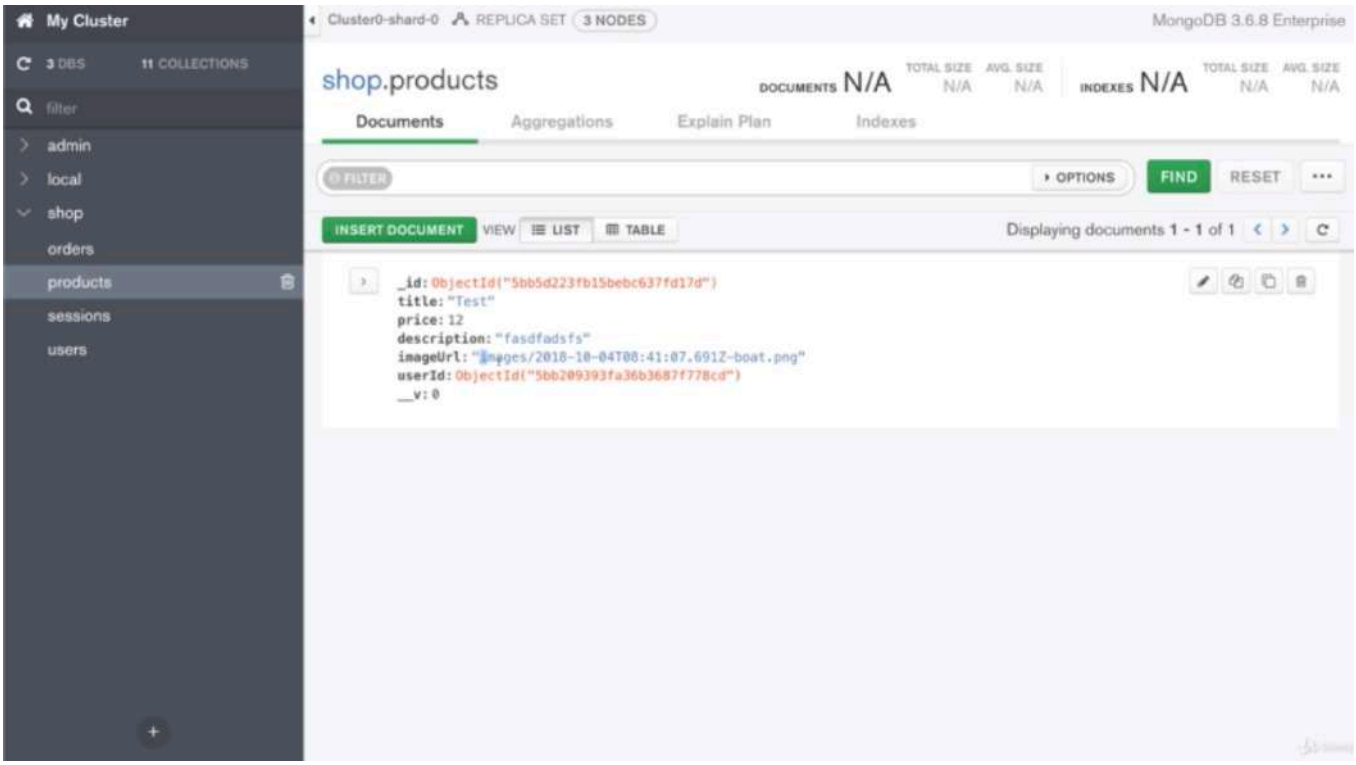


- but if i upload pdf file, then i get 'attached file is not a image'
- you should not store data like this in the database, file should not be stored in a database. they are too big.
- you need to store the path to the file. that is something you can construct with the information given to you in the file object and you can then pass that data to the database.

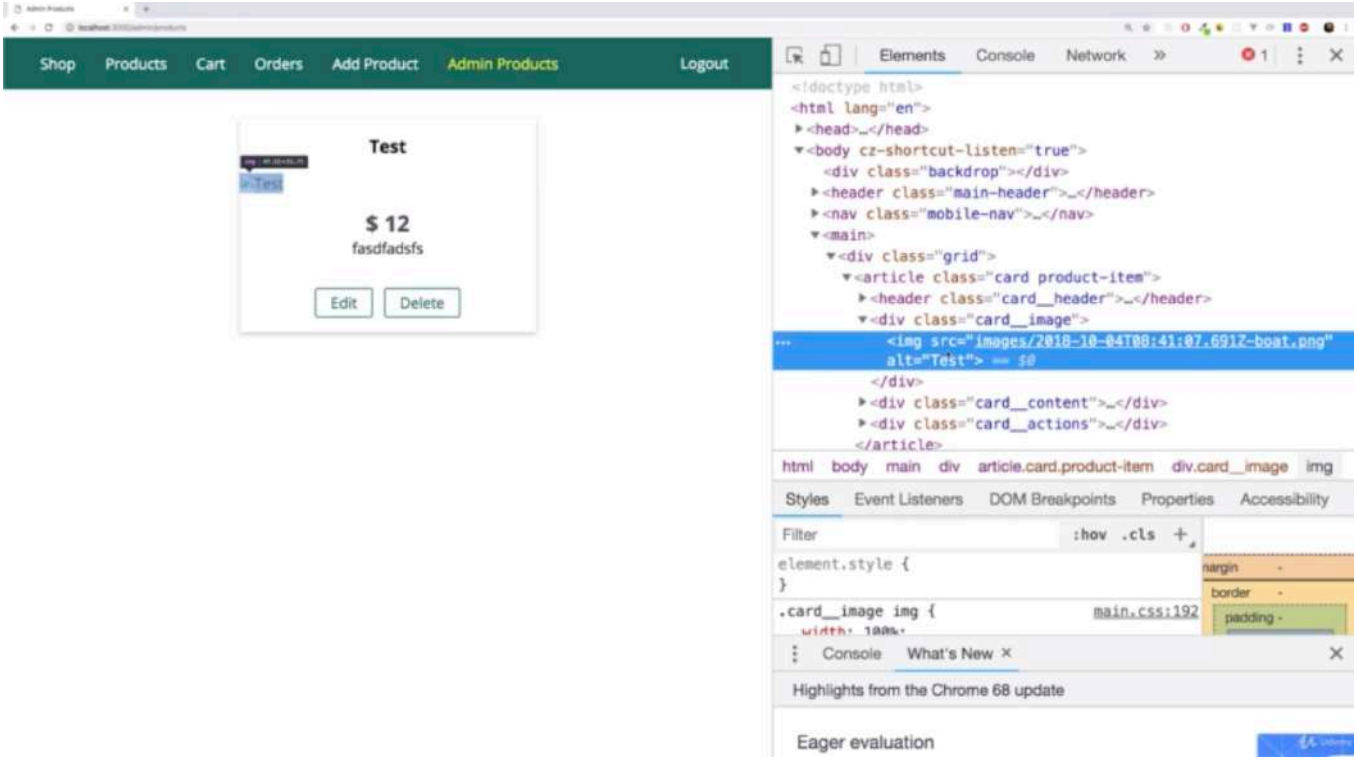




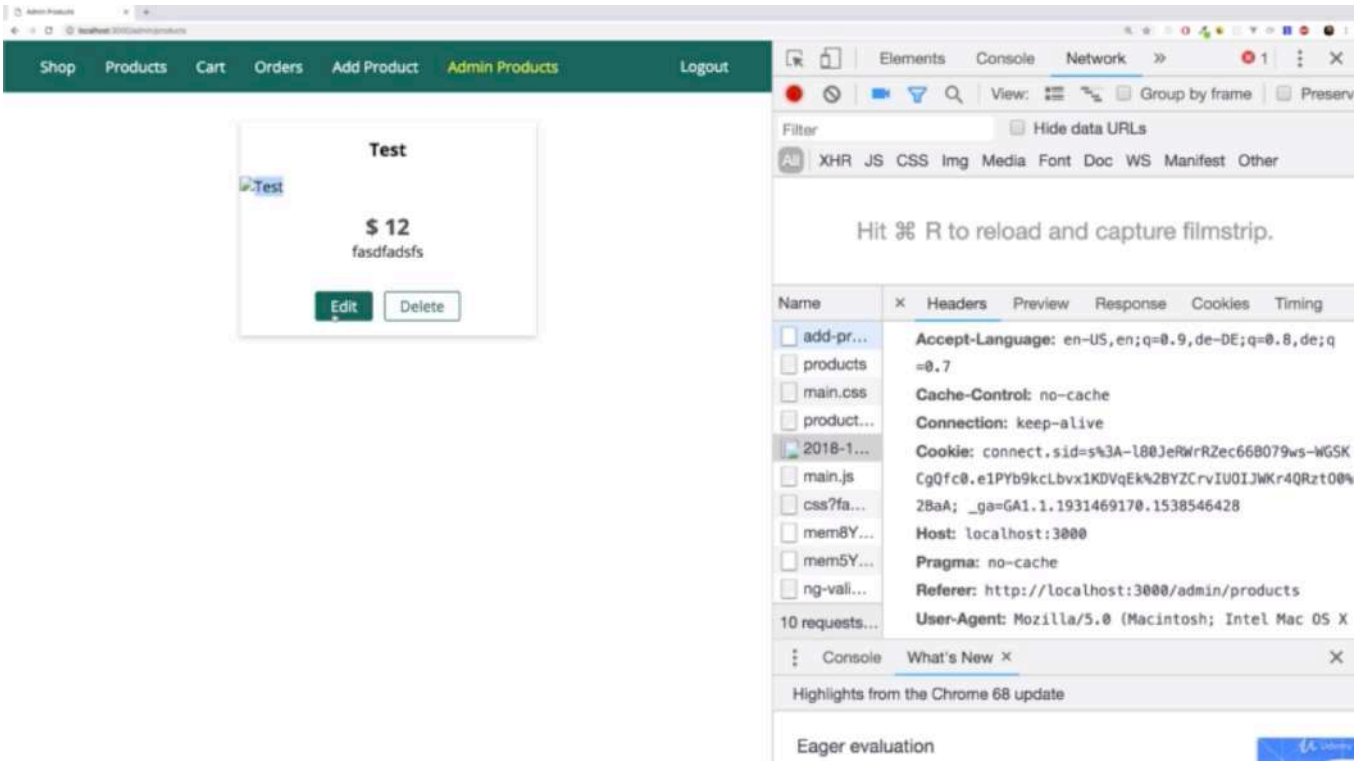
- if i save valid image and this works. saving does work.

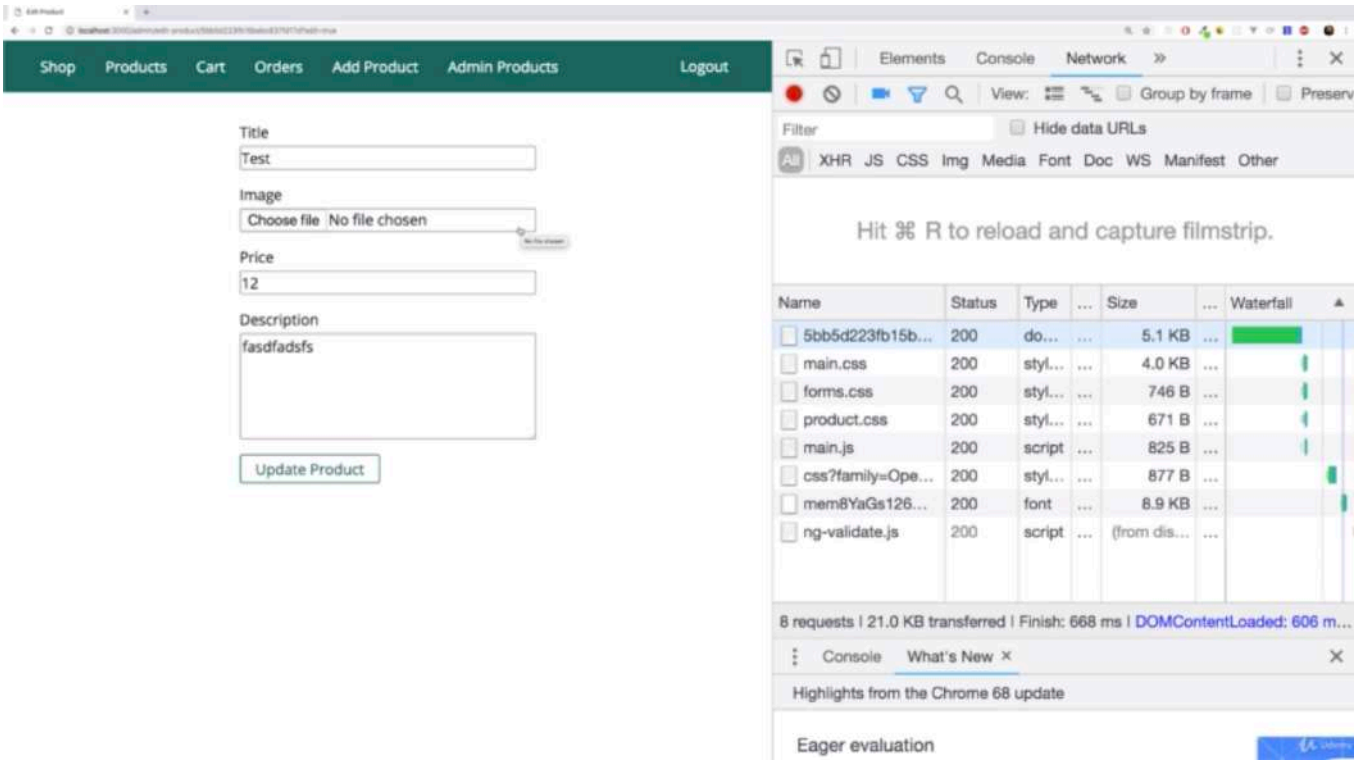


- if you look into my database for the products, you see this is the part that was stored.

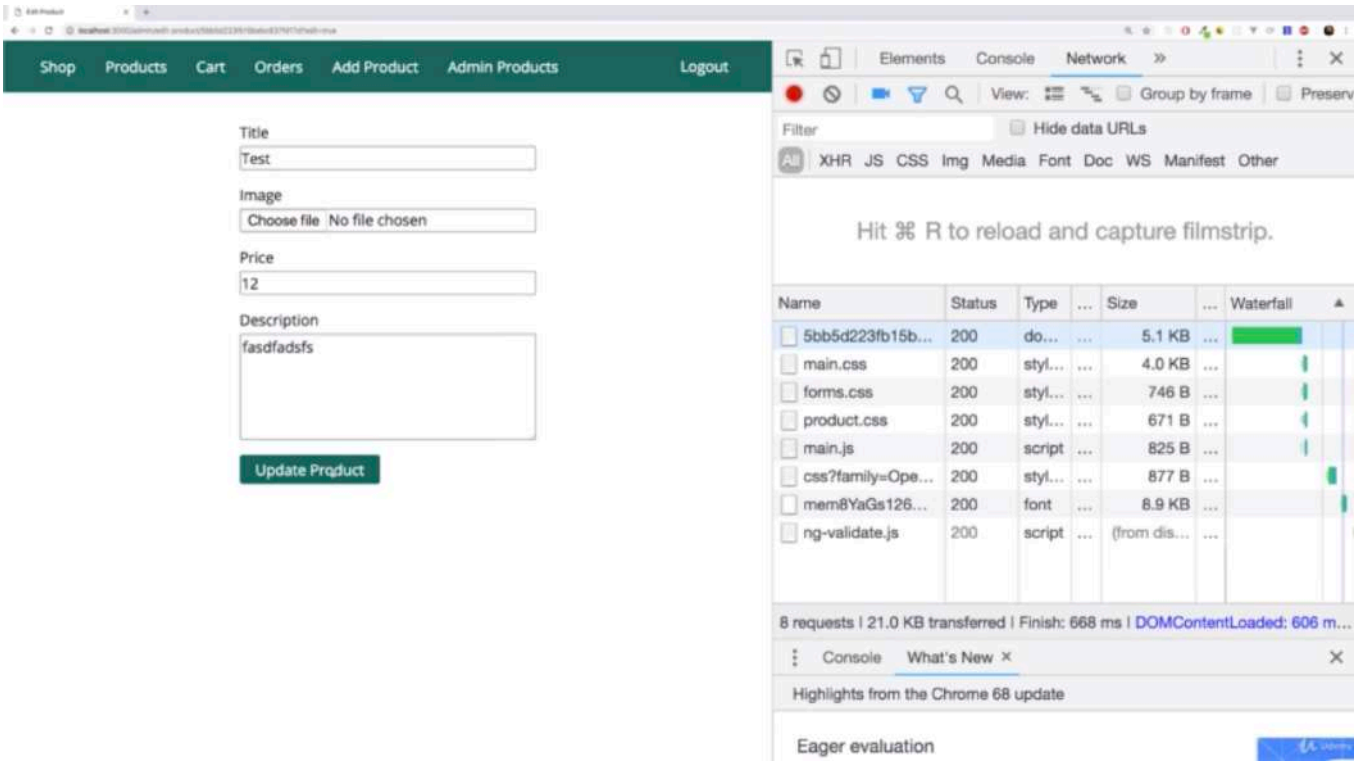


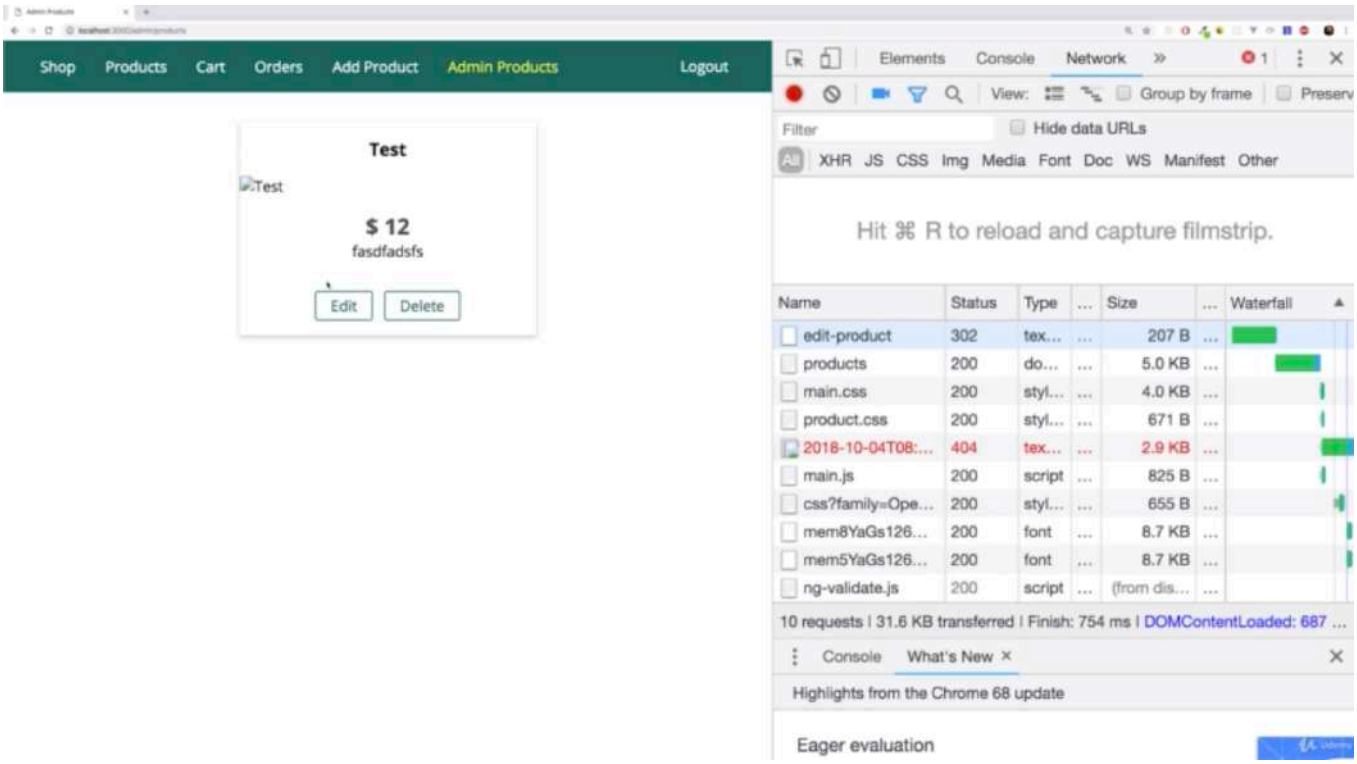
- but if we inspect this, we see the image is not rendered because it can't find this.

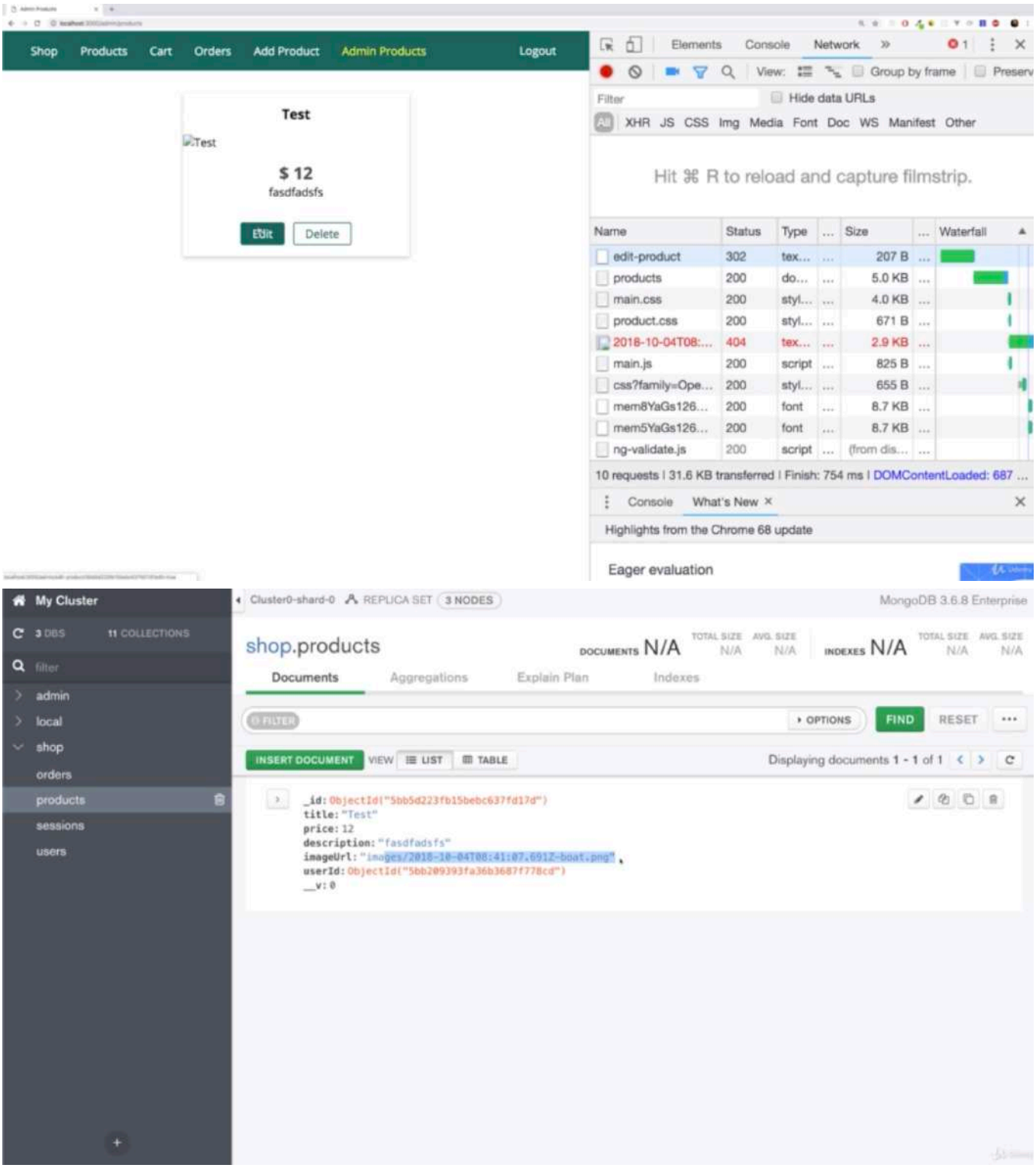




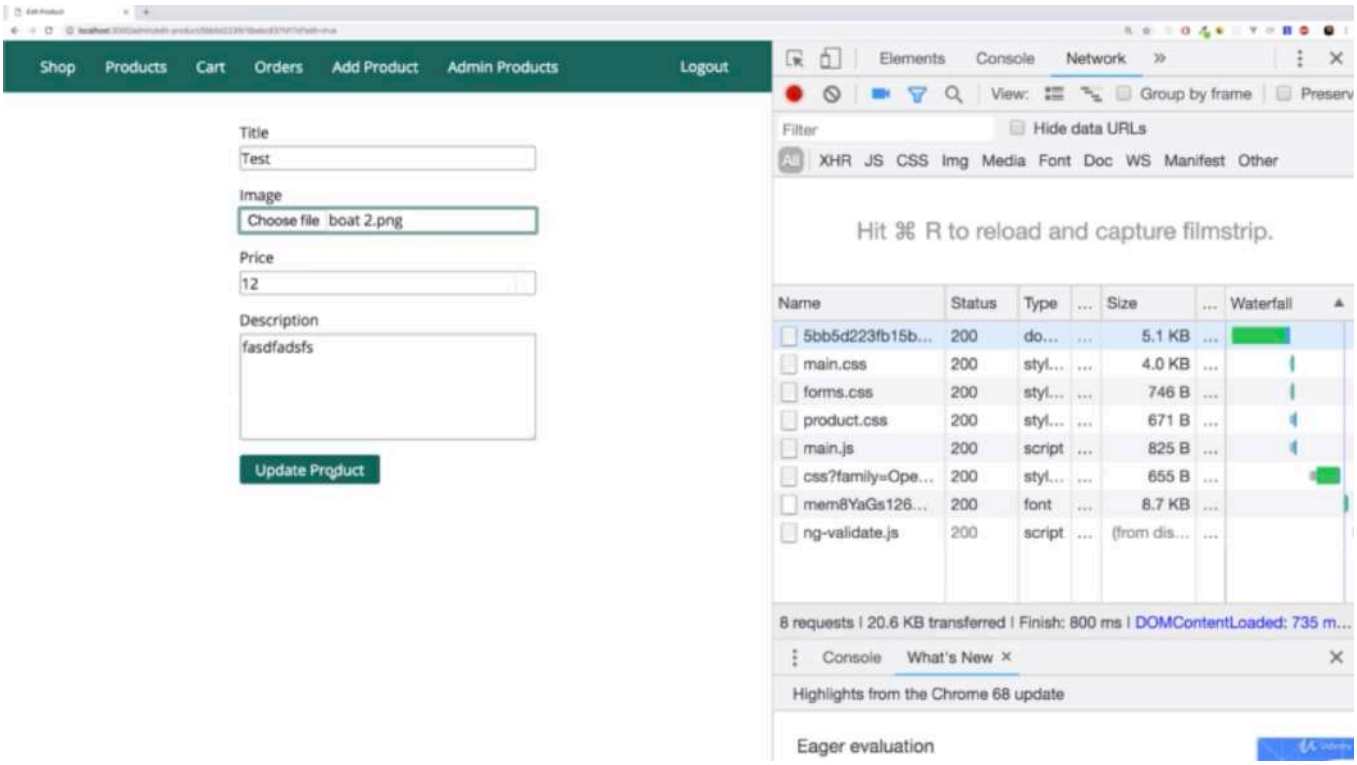
- i wanna have a behavior where if we choose 'no file' here, we keep the old one and we only overwrite it with a new file if i choose new one here.

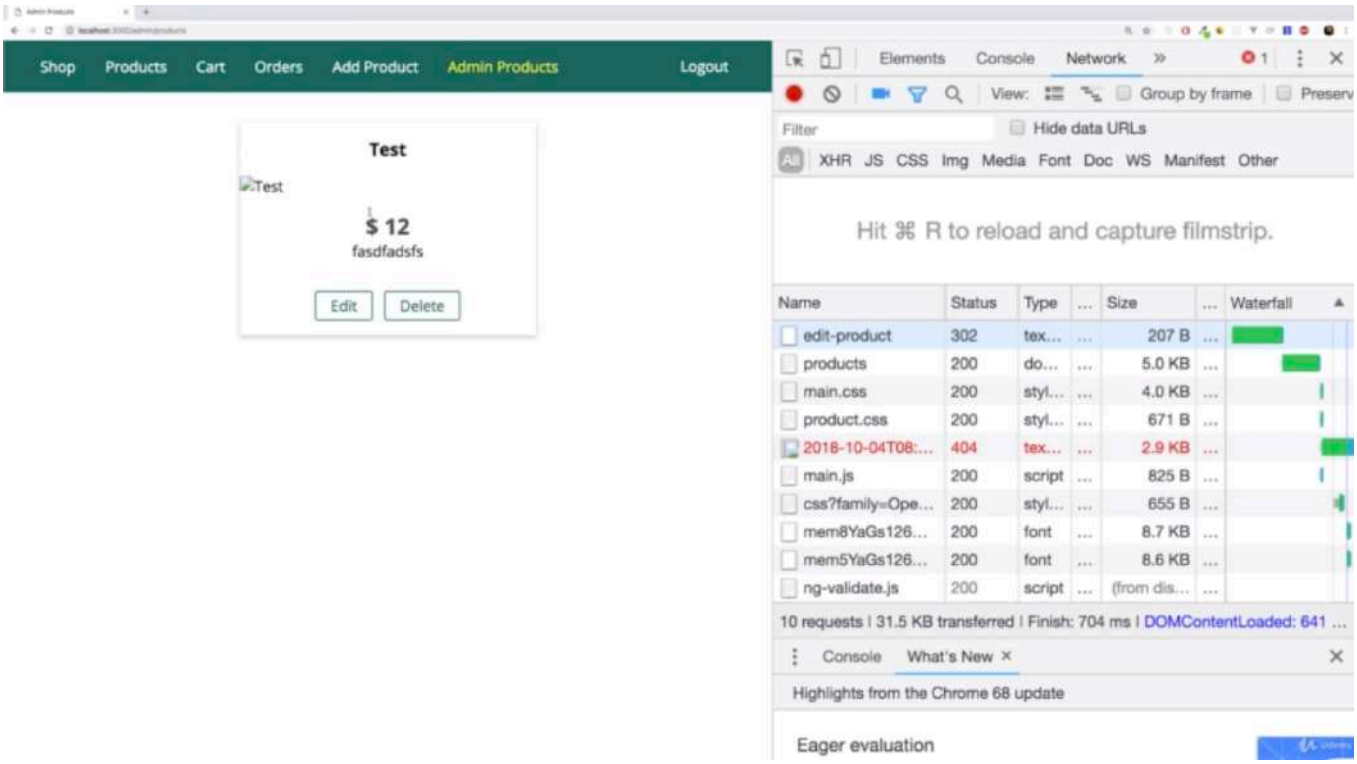

Shop
Products
Cart
Orders
Add Product
Admin Products
Logout
Title
Test
Image
Choose file
No file chosen
Price
12
Description
fasdfadsfs
Update Product
Elements
Console
Network
View:
Group by frame
Filter
Hide data URLs
XHR JS CSS Img Media Font Doc WS Manifest Other
Hit ⌘ R to reload and capture filmstrip.
Name
Status
Type
Size
Waterfall
5bb5d223fb15b...
main.css
forms.css
product.css
main.js
css?family=Ope...
mem8YaGs126...
ng-validate.js
8 requests | 21.0 KB transferred | Finish: 668 ms | DOMContentLoaded: 606 m...
Console
What's New
Highlights from the Chrome 68 update
Eager evaluation

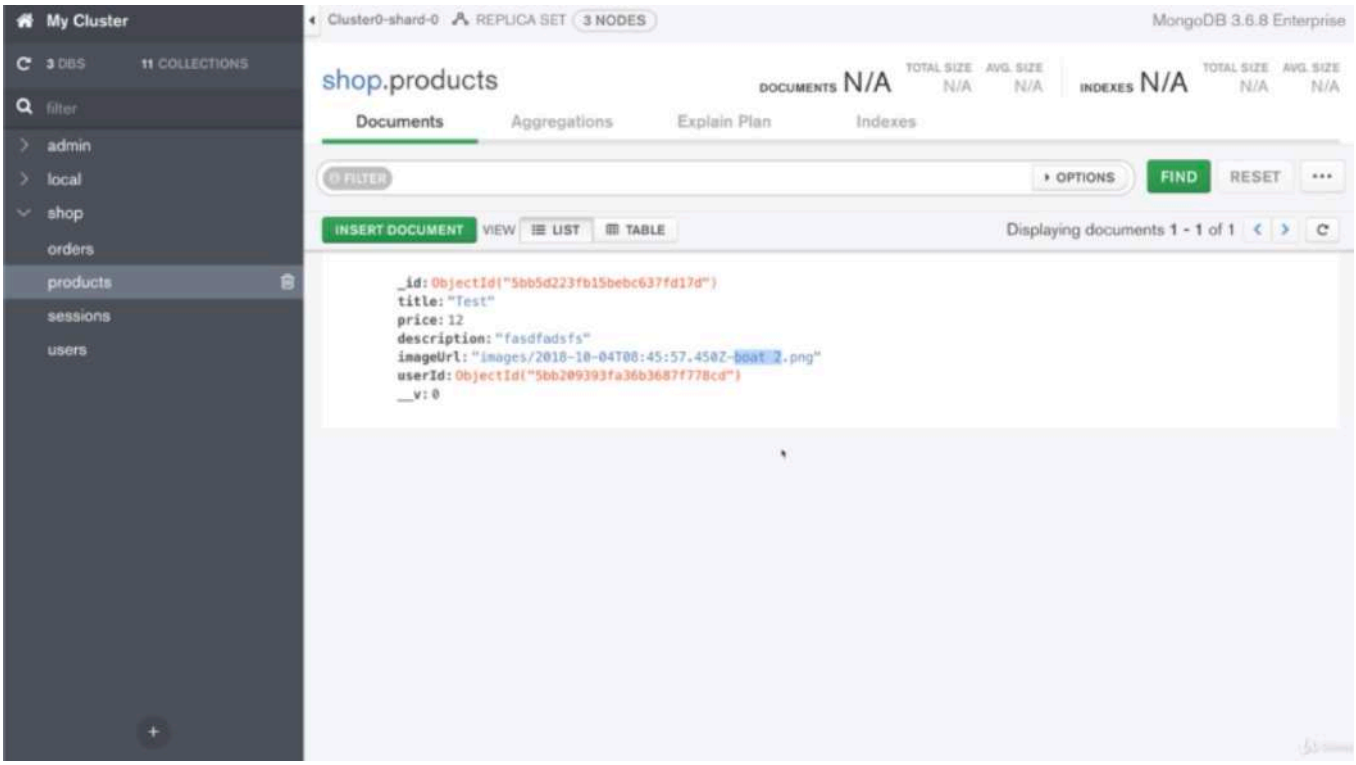

Shop
Products
Cart
Orders
Add Product
Admin Products
Logout
Test
Test
$ 12
fasdfadsfs
Edit
Delete
Elements
Console
Network
View:
Group by frame
Filter
Hide data URLs
XHR JS CSS Img Media Font Doc WS Manifest Other
Hit ⌘ R to reload and capture filmstrip.
Name
Status
Type
Size
Waterfall
edit-product
products
main.css
product.css
2018-10-04T08:...
main.js
css?family=Ope...
mem8YaGs126...
mem5YaGs126...
ng-validate.js
10 requests | 31.6 KB transferred | Finish: 754 ms | DOMContentLoaded: 687 ...
Console
What's New
Highlights from the Chrome 68 update
Eager evaluation



- if i go to my database, that imageUrl didn't change.


Shop
Products
Cart
Orders
Add Product
Admin Products
Logout
Title
Test
Image
Choose file boat 2.png
Price
12
Description
fasdfadsfs
Update Product
Elements
Console
Network
View:
Group by frame
Filter
Hide data URLs
XHR JS CSS Img Media Font Doc WS Manifest Other
Hit ⌘ R to reload and capture filmstrip.
Name Status Type Size Waterfall
5bb5d223fb15b... 200 do... 5.1 KB
main.css 200 styl... 4.0 KB
forms.css 200 styl... 746 B
product.css 200 styl... 671 B
main.js 200 script 825 B
css?family=Ope... 200 styl... 655 B
mem8YaGs126... 200 font 8.7 KB
ng-validate.js 200 script (from dis...
8 requests | 20.6 KB transferred | Finish: 800 ms | DOMContentLoaded: 735 m...
Console
What's New
Highlights from the Chrome 68 update
Eager evaluation


Shop
Products
Cart
Orders
Add Product
Admin Products
Logout
Test
Test
$ 12
fasdfadsfs
Edit
Delete
Elements
Console
Network
View:
Group by frame
Filter
Hide data URLs
XHR JS CSS Img Media Font Doc WS Manifest Other
Hit ⌘ R to reload and capture filmstrip.
Name Status Type Size Waterfall
edit-product 302 tex... 207 B
products 200 do... 5.0 KB
main.css 200 styl... 4.0 KB
product.css 200 styl... 671 B
2018-10-04T08:... 404 tex... 2.9 KB
main.js 200 script 825 B
css?family=Ope... 200 styl... 655 B
mem8YaGs126... 200 font 8.7 KB
mem5YaGs126... 200 font 8.6 KB
ng-validate.js 200 script (from dis...
10 requests | 31.5 KB transferred | Finish: 704 ms | DOMContentLoaded: 641 ...
Console
What's New
Highlights from the Chrome 68 update
Eager evaluation



- but if i select boat-2.png which is same image but a different file name, different file, i update this, and if we go to database, we see it set boat-2

```
// ./controllers/admin.js

const mongoose = require('mongoose');

const { validationResult } = require('express-validator/check');

const Product = require('../models/product');

exports.getAddProduct = (req, res, next) => {
  res.render('admin/edit-product', {
    pageTitle: 'Add Product',
    path: '/admin/add-product',
    editing: false,
    hasError: false,
    errorMessage: null,
    validationErrors: []
  });
};

exports.postAddProduct = (req, res, next) => {
  const title = req.body.title;
  const image = req.file;
  const price = req.body.price;
  const description = req.body.description;
  /**if image is undefined,
   * then that means that multer declined the incoming file
   *
   * if it is undefined,
   * then i wanna return a response with status 422
   * because i have an invalid input
   * so i wwanna return that response.
   */
```

```
  if (!image) {
    res.status(422).render('admin/edit-product', {
      pageTitle: 'Add Product',
      path: '/admin/add-product',
      editing: false,
      hasError: true,
      product: {
        title: title,
        price: price,
        description: description
      },
      errorMessage: 'Attached file is not an image',
      validationErrors: []
    });
  }
  const errors = validationResult(req);

  if (!errors.isEmpty()) {
    console.log(errors.array());
    return res.status(422).render('admin/edit-product', {
      pageTitle: 'Add Product',
      path: '/admin/add-product',
      editing: false,
      hasError: true,
      product: {
        title: title,
        imageUrl: imageUrl,
        price: price,
        description: description
      },
      errorMessage: errors.array()[0].msg,
      validationErrors: errors.array()
    });
  }
  /**if we made it after our entire validation,
   * i have a valid file and valid input data
   *
   * i will use my 'image' data which is that file object we get from multer
   * and there we have information like the file 'path'.
   * this is the 'path' that a file on my operating system
   * so this is the path i wanna use later on when fetching that image.
   * so this is the path i will store in the imageUrl
   * and i can store imageUrl in the database again.
   */
  const imageUrl = image.path

  const product = new Product({
    //_id: new mongoose.Types.ObjectId('5badf72403fd8b5be0366e81'),
    title: title,
    price: price,
    description: description,
    imageUrl: imageUrl,
    userId: req.user
  });
  product
    .save()
```

```
    .then(result => {
      // console.log(result);
      console.log('Created Product');
      res.redirect('/admin/products');
    })
    .catch(err => {
      // return res.status(500).render('admin/edit-product', {
      //   pageTitle: 'Add Product',
      //   path: '/admin/add-product',
      //   editing: false,
      //   hasError: true,
      //   product: {
      //     title: title,
      //     imageUrl: imageUrl,
      //     price: price,
      //     description: description
      //   },
      //   errorMessage: 'Database operation failed, please try again.',
      //   validationErrors: []
      // });
      //res.redirect('/500');
      const error = new Error(err)
      error.httpStatusCode = 500
      return next(error)
    });
};

exports.getEditProduct = (req, res, next) => {
  const editMode = req.query.edit;
  if (!editMode) {
    return res.redirect('/');
  }
  const prodId = req.params.productId;
  Product.findById(prodId)
    .then(product => {
      if (!product) {
        return res.redirect('/');
      }
      res.render('admin/edit-product', {
        pageTitle: 'Edit Product',
        path: '/admin/edit-product',
        editing: editMode,
        product: product,
        hasError: false,
        errorMessage: null,
        validationErrors: []
      });
    })
    .catch(err => {
      const error = new Error(err)
      error.httpStatusCode = 500
      return next(error)
    });
};

exports.postEditProduct = (req, res, next) => {
```

```
  const prodId = req.body.productId;
  const updatedTitle = req.body.title;
  const updatedPrice = req.body.price;
  /**if 'image' is undefined,
   * i know that no file was saved.
   * and this means i wanna keep the old file,
   * by the way same will be true
   * if i upload a pdf.
   * so we don't need to throw an error message or error
   *
   */
  const image = req.body.imageUrl;
  const updatedDesc = req.body.description;

  const errors = validationResult(req);

  if (!errors.isEmpty()) {
    return res.status(422).render('admin/edit-product', {
      pageTitle: 'Edit Product',
      path: '/admin/edit-product',
      editing: true,
      hasError: true,
      product: {
        /**we don't need 'imageUrl: updatedImageUrl'
         * because we don't render image in page anymore.
         */
        title: updatedTitle,
        price: updatedPrice,
        description: updatedDesc,
        _id: prodId
      },
      errorMessage: errors.array()[0].msg,
      validationErrors: errors.array()
    });
  }

  Product.findById(prodId)
    .then(product => {
      if (product.userId.toString() !== req.user._id.toString()) {
        return res.redirect('/');
      }
      product.title = updatedTitle;
      product.price = updatedPrice;
      product.description = updatedDesc;
      if (image) {
        /**this is the same logic as i have it in postAddProduct */
        product.imageUrl = updatedImageUrl;
      }
      return product.save().then(result => {
        console.log('UPDATED PRODUCT!');
        res.redirect('/admin/products');
      });
    })
    .catch(err => {
      const error = new Error(err)
      error.httpStatusCode = 500
```

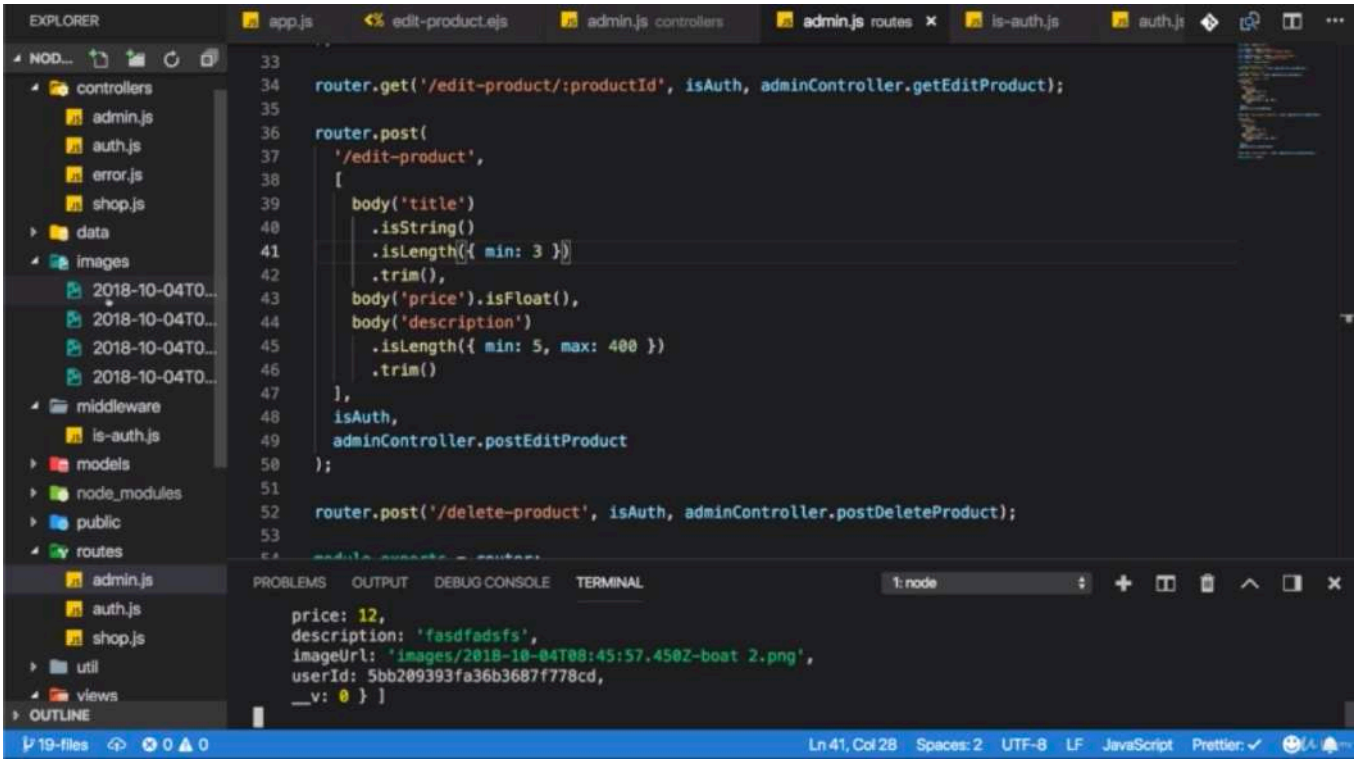
```
    return next(error)
  });
};

exports.getProducts = (req, res, next) => {
  Product.find({ userId: req.user._id })
    // .select('title price -_id')
    // .populate('userId', 'name')
    .then(products => {
      console.log(products);
      res.render('admin/products', {
        prods: products,
        pageTitle: 'Admin Products',
        path: '/admin/products'
      });
    })
    .catch(err => {
      const error = new Error(err)
      error.httpStatusCode = 500
      return next(error)
    });
};

exports.postDeleteProduct = (req, res, next) => {
  const prodId = req.body.productId;
  Product.deleteOne({ _id: prodId, userId: req.user._id })
    .then(() => {
      console.log('DESTROYED PRODUCT');
      res.redirect('/admin/products');
    })
    .catch(err => {
      const error = new Error(err)
      error.httpStatusCode = 500
      return next(error)
    });
};

```

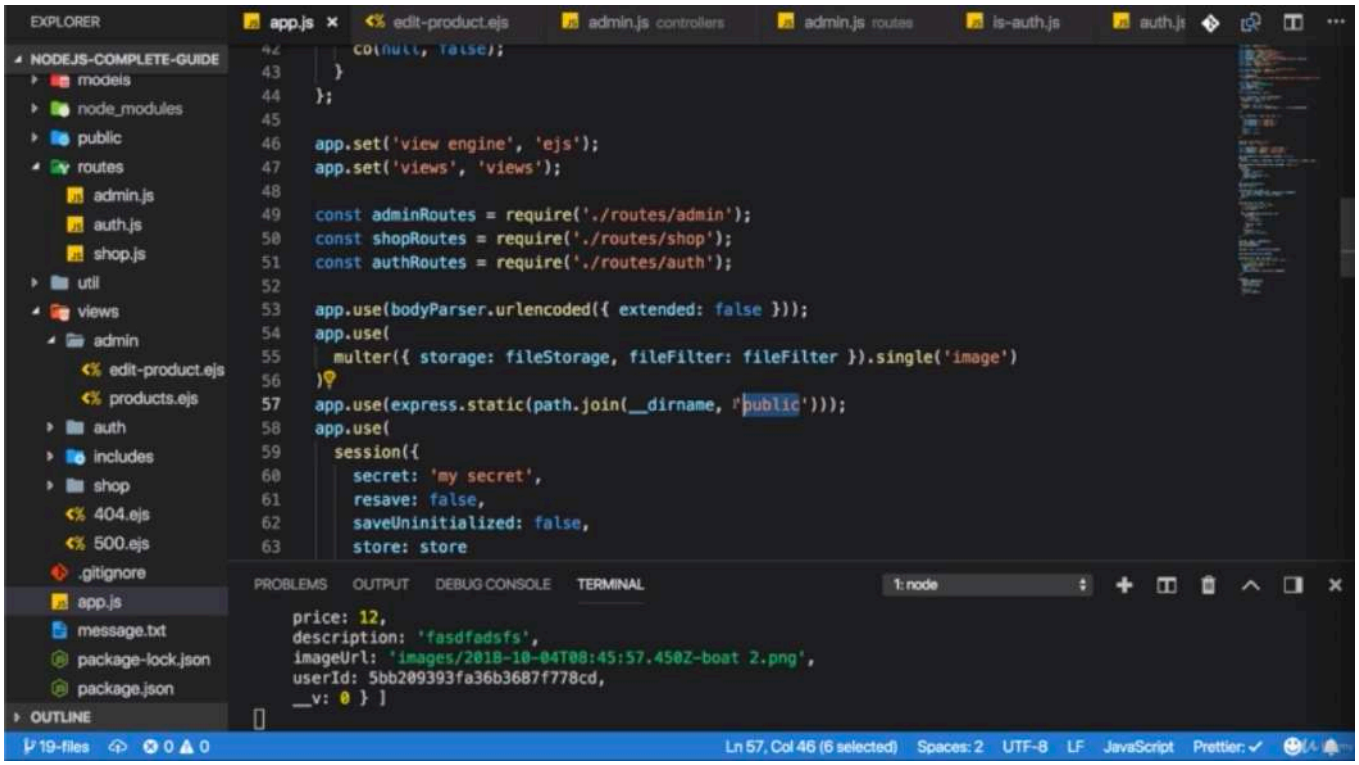
```
// ./routes/admin.js

const path = require('path');

const express = require('express');
const { body } = require('express-validator/check')

const adminController = require('../controllers/admin');
const isAuth = require('../middleware/is-auth');

const router = express.Router();

// /admin/add-product => GET
router.get('/add-product', isAuth, adminController.getAddProduct);

// /admin/products => GET
router.get('/products', isAuth, adminController.getProducts);

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

```
router.post('/add-product', [
        body('title').isString().isLength({ min: 3 }).trim(),
        body('price').isFloat(),
        body('description').isLength({ min: 5, max: 400 }).trim()
    ],
    isAuth,
    adminController.postAddProduct);

router.get('/edit-product/:productId', isAuth, adminController.getEditProduct);

router.post('/edit-product',[
        body('title').isString().isLength({ min: 3 }).trim(),
        body('price').isFloat(),
        body('description').isLength({ min: 5, max: 400 }).trim()
    ],
    isAuth,
    adminController.postEditProduct);

router.post('/delete-product', isAuth, adminController.postDeleteProduct);

module.exports = router;
```

* Chapter 322: Serving Images Statically

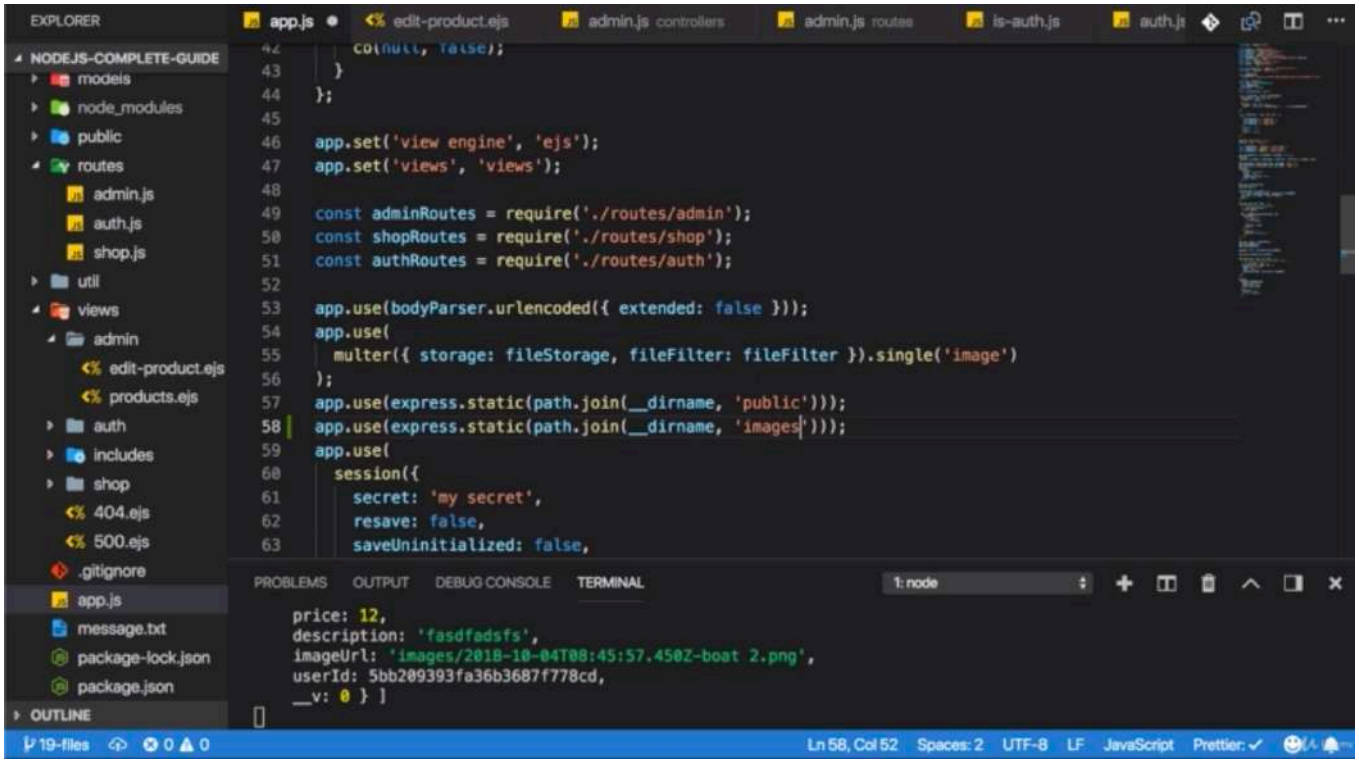
1. update
- app.js
- ./views/admin/products.ejs
- ./views/shop/index.ejs
- ./views/shop/product-list.ejs
- ./views/shop/product-detail.ejs





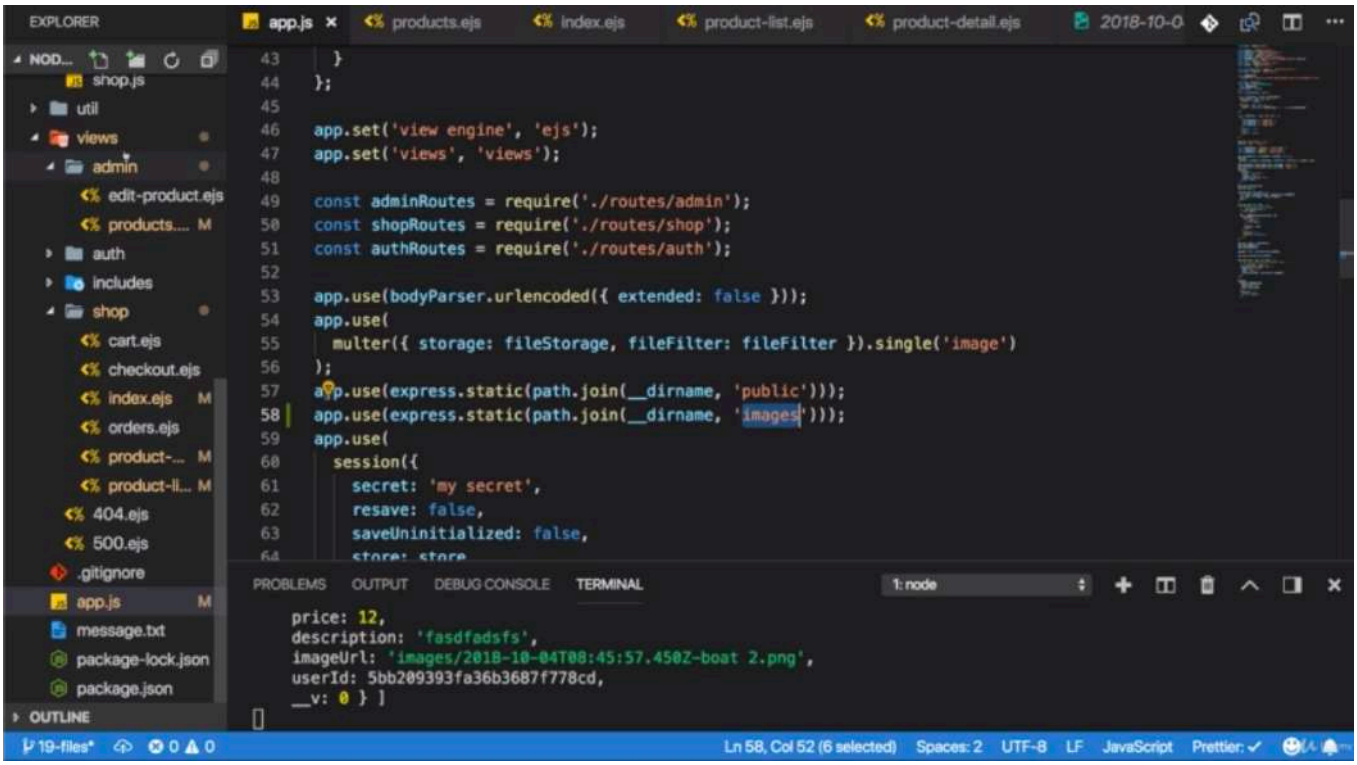
- we got multiple options for serving files. option 1 is we serve our images folder in a static way which means that we are serving a public folder with the express.static middleware. and we can serve more than one folder statically.

- 'statically serving a folder' means that requests to files in that folder will be handled automatically and the files will be returned so all the heavy lifting is done behind the scenes by express.

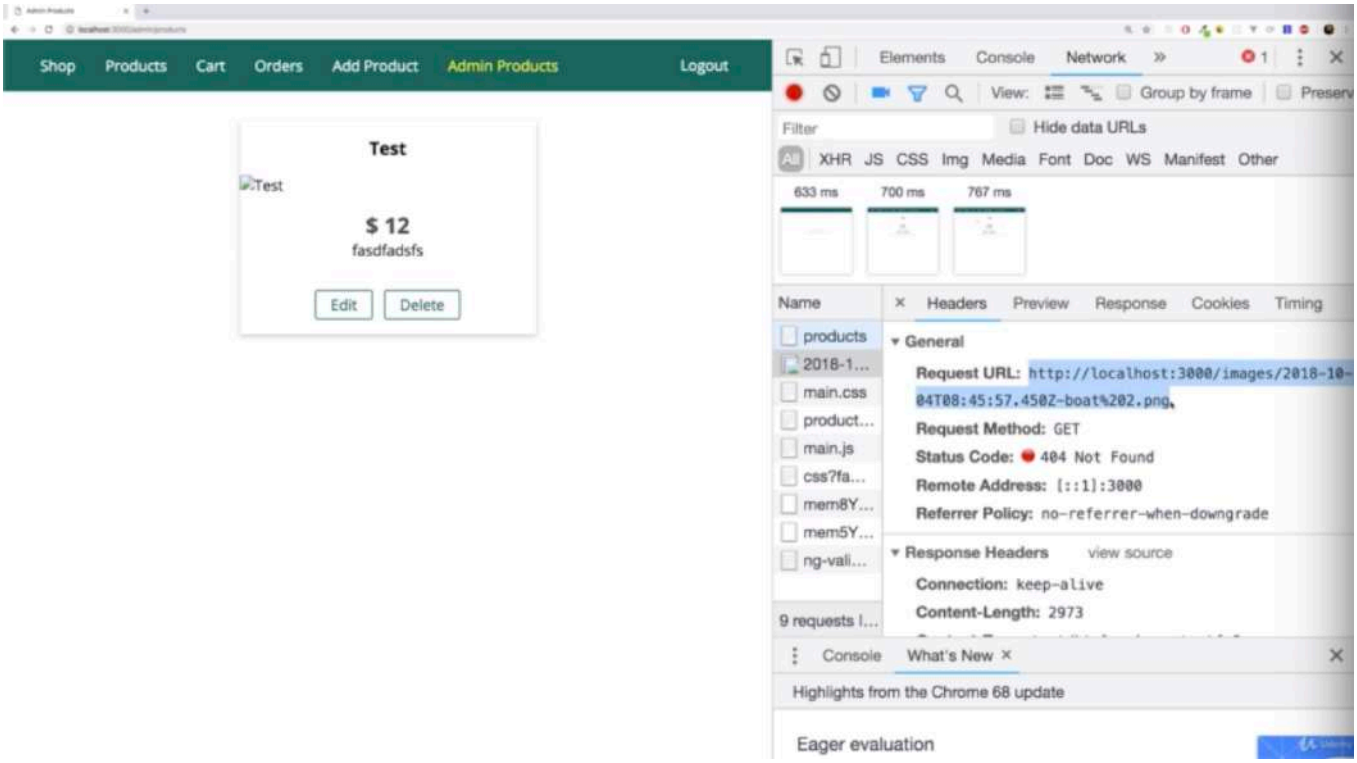


- and now also serve the images folder just like this.

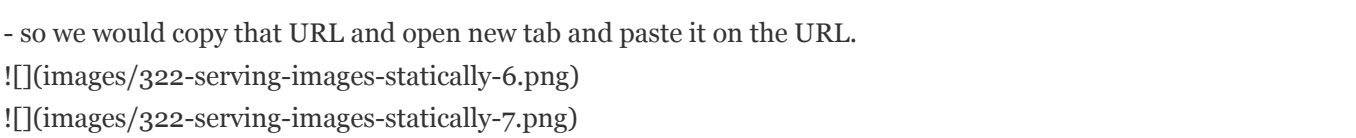
————————



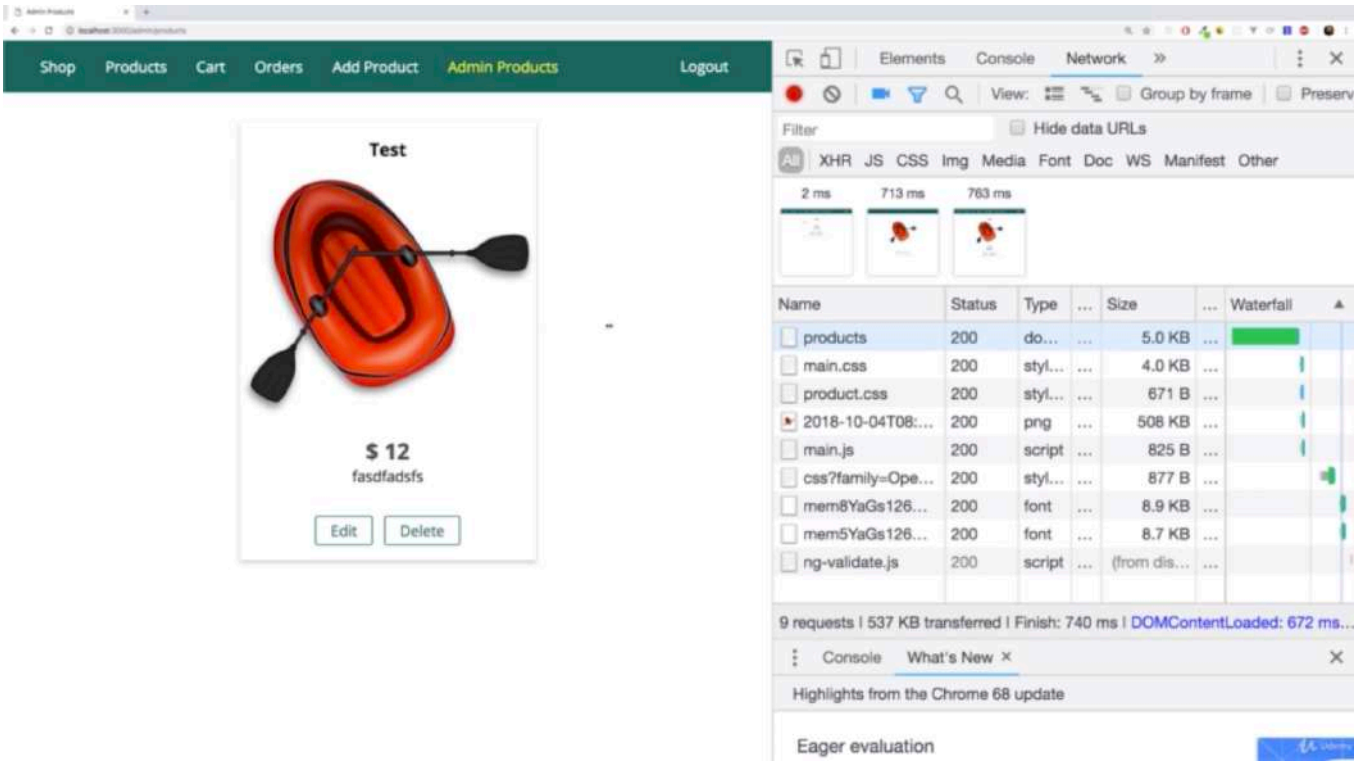
- we point to a folder like 'public' and ‘images' and we tell express serve the files from inside that folder as if they were on the root folder.



- so we would copy that URL and open new tab and paste it on the URL.

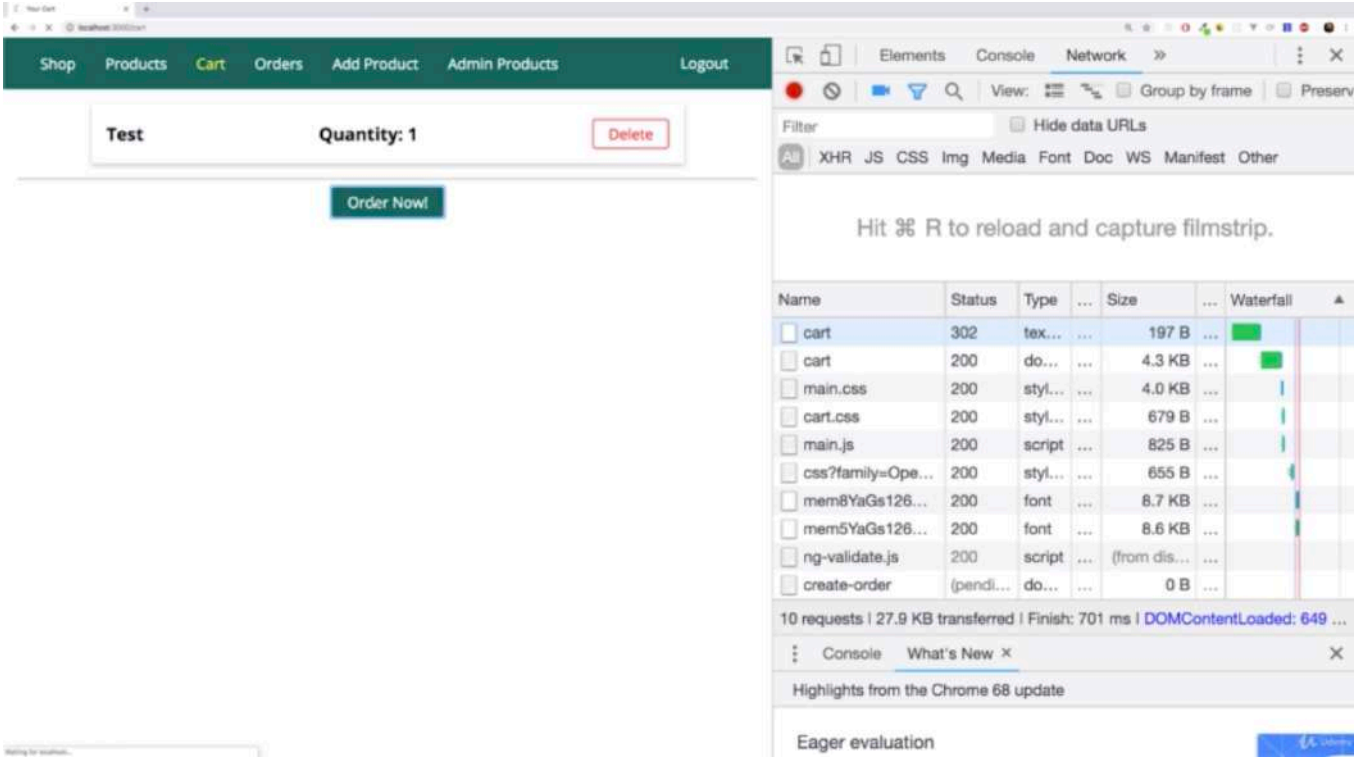


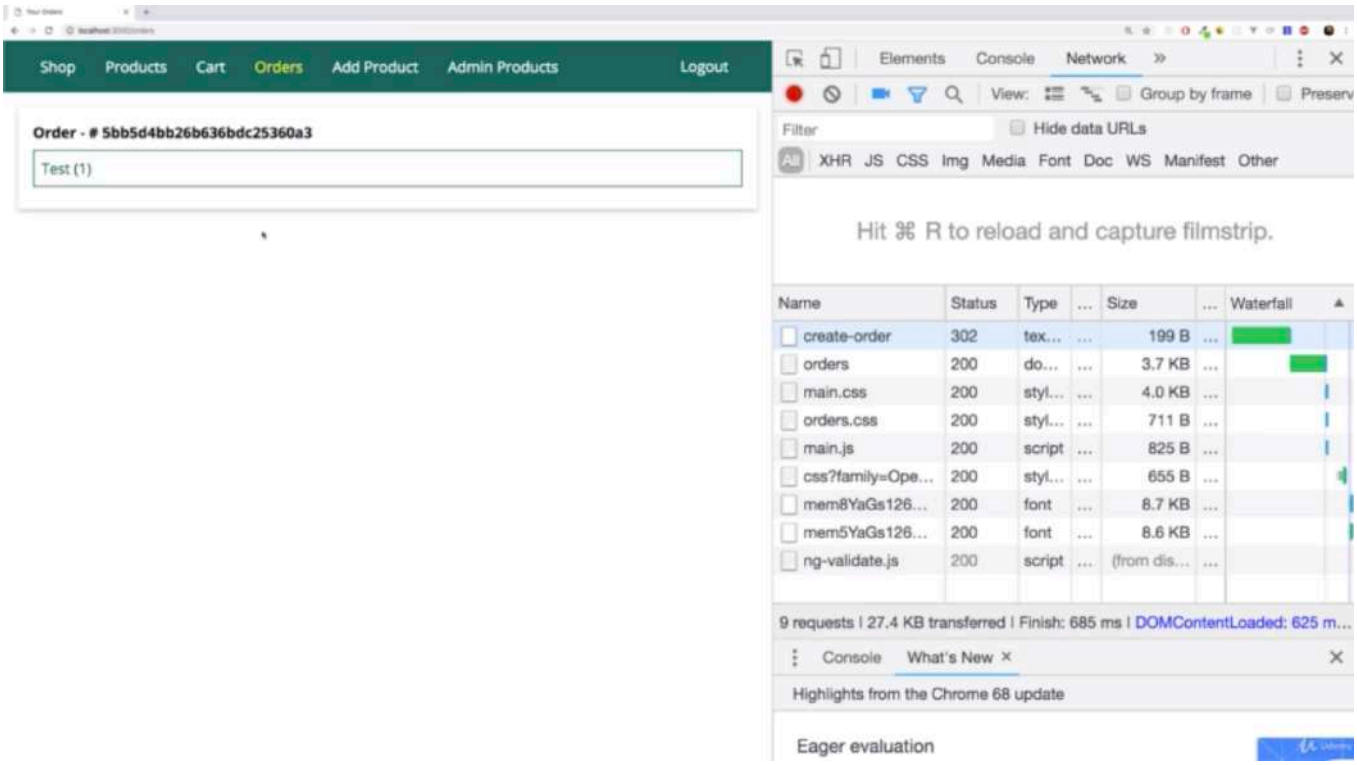
- and if we remove ‘images’ in URL, the reason for that is that express assumes that the files in the images folder are served as if they are in the root folder. so ‘/‘ slash nothing.



- so if we save and we reload, we see our image here.

————————————————





- now i wanna do something differently. i have an order and i wanna download an invoice for that order. and that will now work differently because invoices will not be public files that everyone should be able to access. i wanna be able to access my invoice. but no one else should be.

```
//app.js

const path = require('path');

const express = require('express');
const bodyParser = require('body-parser');
const mongoose = require('mongoose');
const session = require('express-session');
const MongoDBStore = require('connect-mongodb-session')(session);
const csrf = require('csurf');
const flash = require('connect-flash');
const multer = require('multer')

const errorController = require('./controllers/error');
const User = require('./models/user');

const MONGODB_URI =
'mongodb+srv://maximilian:rldnjs12@cluster0-z3vlk.mongodb.net/shop'

const app = express();
const store = new MongoDBStore({
  uri: MONGODB_URI,
  collection: 'sessions'
});
const csrfProtection = csrf();
const fileStorage = multer.diskStorage({
  destination: (req, file, cb) => {
    cb(null, 'images')
  },
  filename: (req, file, cb) => {
    cb(null, new Date().toISOString() + '-' + file.originalname)
```

```
  }
})

const fileFilter = (req, file, cb) => {
  if (
    file.mimetype === 'image/png' ||
    file.mimetype === 'image/jpg' ||
    file.mimetype === 'image/jpeg'
    ) {
    cb(null, true)
  } else {
    cb(null, false)
  }
}

app.set('view engine', 'ejs');
app.set('views', 'views');

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

app.use(bodyParser.urlencoded({ extended: false }));
app.use(multer({ storage: fileStorage, fileFilter }).single('image'))
/**we point to a folder like 'public' and 'images'
 * and we tell express serve the files from inside that folder
 * as if they were on the root folder.
 *
 * and if we remove ‘images’ in URL,
 * the reason for that is that express assumes that
 * the files in the images folder are served
 * as if they are in the root folder. so ‘/‘ slash nothing.
 *
 * so we can adjust our middleware
 * and if we have a request that goes to '/images'
 * that start with '/images'
 * then serve these files statically
 * and now '/images' is the folder we assume for this static serving
 */
app.use(express.static(path.join(__dirname, 'public')));
app.use('/images', express.static(path.join(__dirname, 'images')));
app.use(
  session({
    secret: 'my secret',
    resave: false,
    saveUninitialized: false,
    store: store
  })
);
app.use(csrfProtection);
app.use(flash());

app.use((req, res, next) => {
  res.locals.isAuthenticated = req.session.isLoggedIn;
  res.locals.csrfToken = req.csrfToken();
  next();
```

```
});

app.use((req, res, next) => {
  //throw new Error('Sync Dummy')
  if (!req.session.user) {
    return next();
  }
  User.findById(req.session.user._id)
    .then(user => {
      if (!user) {
        return next();
      }
      req.user = user;
      next();
    })
    .catch(err => {
      next(new Error(err))
    });
});

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

app.get('/500', errorController.get500);

app.use(errorController.get404);

app.use((error, req, res, next) => {
  //res.redirect('/500')
  res.status(500).render('500', {
    pageTitle: 'Error!',
    path: '/500',
    isAuthenticated: req.session.isLoggedIn
  });
})

mongoose
  .connect(MONGODB_URI)
  .then(result => {
    app.listen(3000);
  })
  .catch(err => {
    console.log(err);
  });

```

```
<!--./views/shop/index.ejs-->

<%- include('../includes/head.ejs') %>
    <link rel="stylesheet" href="/css/product.css">
</head>

<body>
    <%- include('../includes/navigation.ejs') %>

    <main>
```

```
        <% if (prods.length > 0) { %>
            <div class="grid">
                <% for (let product of prods) { %>
                    <article class="card product-item">
                        <header class="card__header">
                            <h1 class="product__title"><%= product.title %></h1>
                        </header>
                        <div class="card__image">
                            <img src="/<%= product.imageUrl %>"
                                alt="<%= product.title %>">
                        </div>
                        <div class="card__content">
                            <h2 class="product__price">$<%= product.price %></h2>
                            <p class="product__description"><%= product.description %></p>
                        </div>
                        <div class="card__actions">
                            <a href="/products/<%= product._id %>" class="btn">Details</a>
                            <% if (isAuthenticated) { %>
                                <%- include('../includes/add-to-cart.ejs', {product:
product}) %>
                            <% } %>
                        </div>
                    </article>
                <% } %>
            </div>
        <% } else { %>
            <h1>No Products Found!</h1>
        <% } %>
    </main>
<%- include('../includes/end.ejs') %>
```

```
<!--./views/shop/product-list.ejs-->

<%- include('../includes/head.ejs') %>
    <link rel="stylesheet" href="/css/product.css">
    </head>

    <body>
        <%- include('../includes/navigation.ejs') %>

            <main>
                <% if (prods.length > 0) { %>
                    <div class="grid">
                        <% for (let product of prods) { %>
                            <article class="card product-item">
                                <header class="card__header">
                                    <h1 class="product__title">
                                        <%= product.title %>
                                    </h1>
                                </header>
                                <div class="card__image">
                                    <img src="/<%= product.imageUrl %>" alt="<%=
product.title %>">
                                </div>
                                <div class="card__content">
                                    <h2 class="product__price">$
                                        <%= product.price %>
```

```
                    </h2>
                    <p class="product__description">
                        <%= product.description %>
                    </p>
                </div>
                <div class="card__actions">
                    <a href="/products/<%= product._id %>" class="btn">Details</a>
                    <% if (isAuthenticated) { %>
                        <%- include('../includes/add-to-cart.ejs', {product: product}) %>
                    <% } %>
                </div>
            </article>
            <% } %>
        </div>
        <% } else { %>
            <h1>No Products Found!</h1>
            <% } %>
    </main>
    <%- include('../includes/end.ejs') %>
```

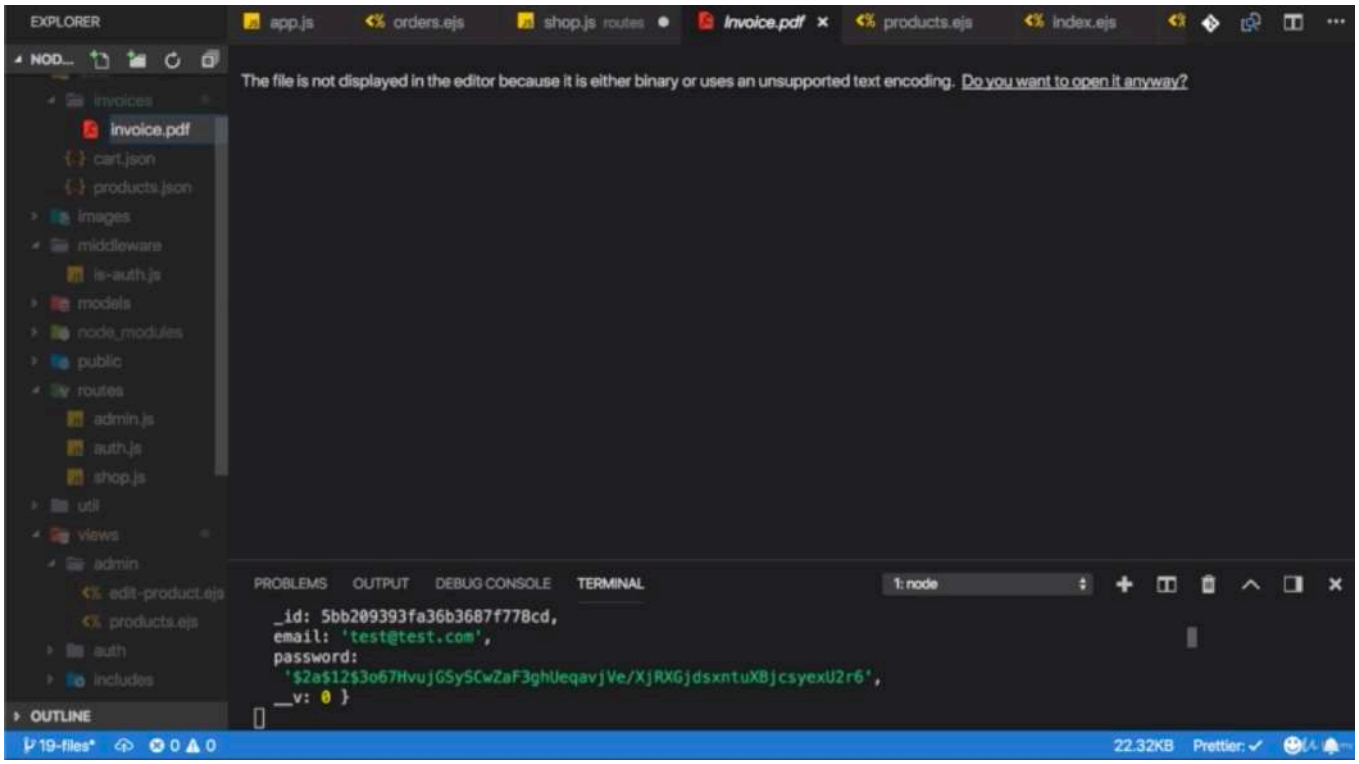
```
<!--./views/shop/product-detail.ejs-->

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

    <body>
        <%- include('../includes/navigation.ejs') %>
        <main class="centered">
            <h1><%= product.title %></h1>
            <hr>
            <div class="image">
                <img src="/<%= product.imageUrl %>" alt="<%= product.title %>">
            </div>
            <h2><%= product.price %></h2>
            <p><%= product.description %></p>
            <% if (isAuthenticated) { %>
                <%- include('../includes/add-to-cart.ejs', {product: product}) %>
            <% } %>
        </main>
        <%- include('../includes/end.ejs') %>
```

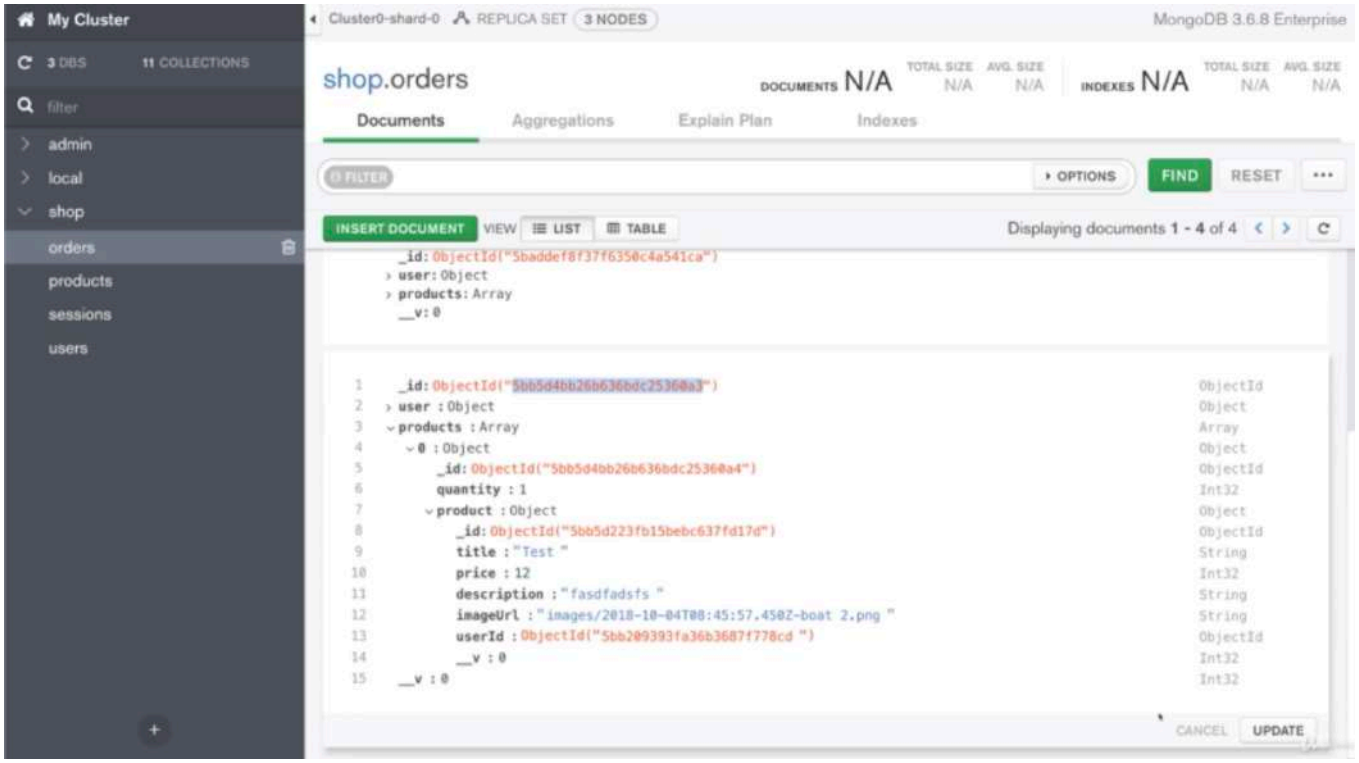
```
<!--./views/admin/products.ejs-->

<%- include('../includes/head.ejs') %>
    <link rel="stylesheet" href="/css/product.css">
    </head>

    <body>
        <%- include('../includes/navigation.ejs') %>

            <main>
                <% if (prods.length > 0) { %>
                    <div class="grid">
                        <% for (let product of prods) { %>
                            <article class="card product-item">
```

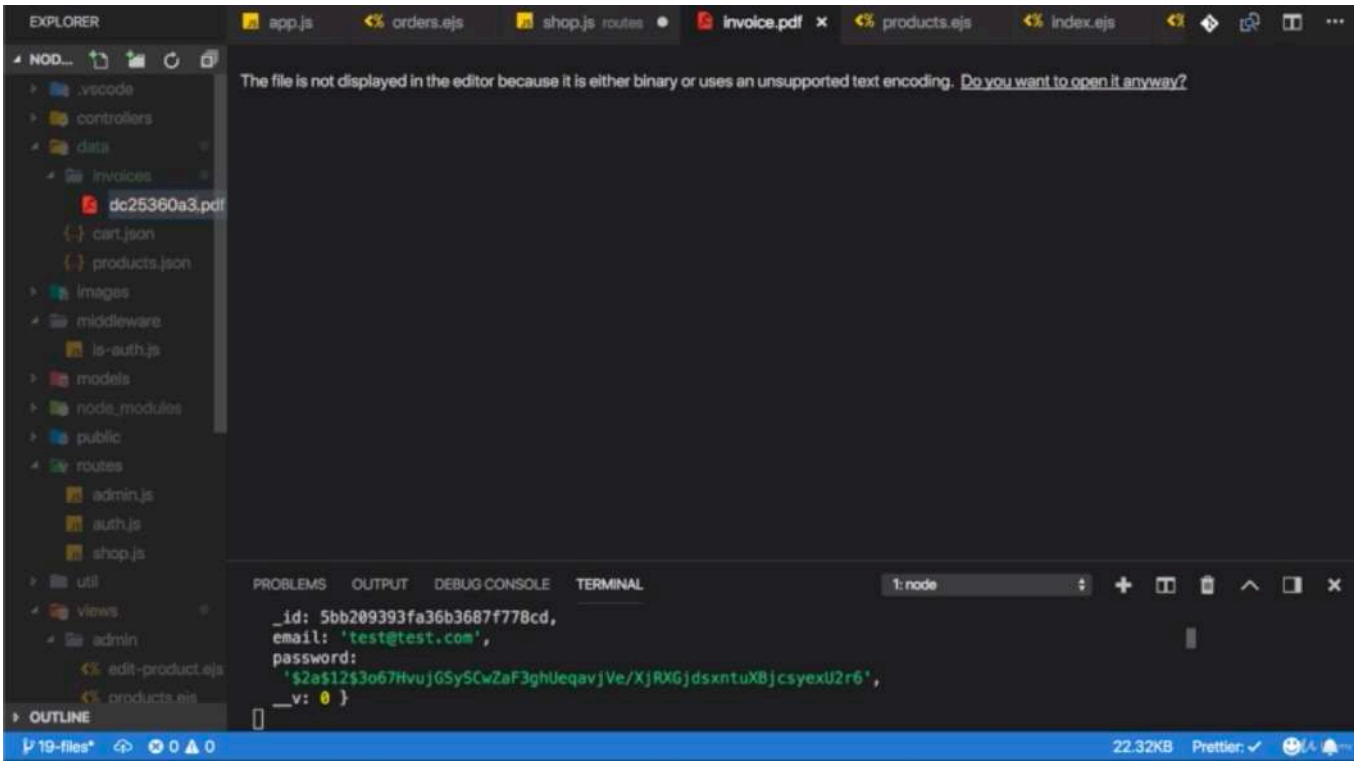
```
                    <header class="card__header">
                        <h1 class="product__title">
                            <%= product.title %>
                        </h1>
                    </header>
                    <div class="card__image">
                        <!--
                            we need to add a slash at the beginning
                            which will turn this into an absolute path
                            it will not append it to the current path
                            but rather create a new path with only our domain
                            and then the path which get rendered here
                            and i do this in products.
                        -->
                        <img src="/<%= product.imageUrl %>" alt="<%= product.title %>">
                    </div>
                    <div class="card__content">
                        <h2 class="product__price">$
                            <%= product.price %>
                        </h2>
                        <p class="product__description">
                            <%= product.description %>
                        </p>
                    </div>
                    <div class="card__actions">
                        <a href="/admin/edit-product/<%= product._id %>?edit=true" class="btn">Edit</a>
                        <form action="/admin/delete-product" method="POST">
                            <input type="hidden" value="<%= product._id %>" name="productId">
                            <input type="hidden" name="_csrf" value="<%= csrfToken %>">
                            <button class="btn" type="submit">Delete</button>
                        </form>

                    </div>
                </article>
                <% } %>
            </div>
            <% } else { %>
                <h1>No Products Found!</h1>
                <% } %>
        </main>
        <%- include('../includes/end.ejs') %>
```

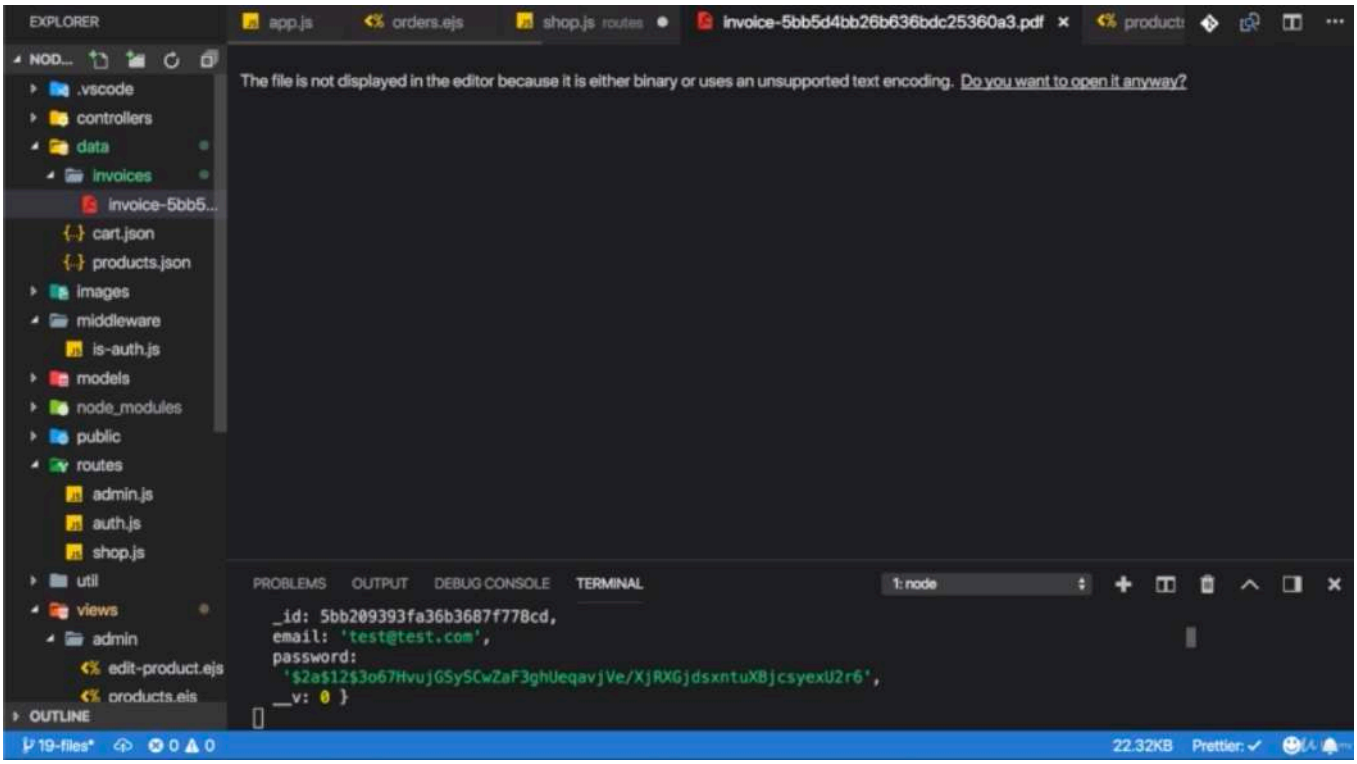
* Chapter 323: Downloading Files With Authentication

1. update
- ./views/shop/orders.ejs
- ./routes/shop.js
- ./controllers/shop.js

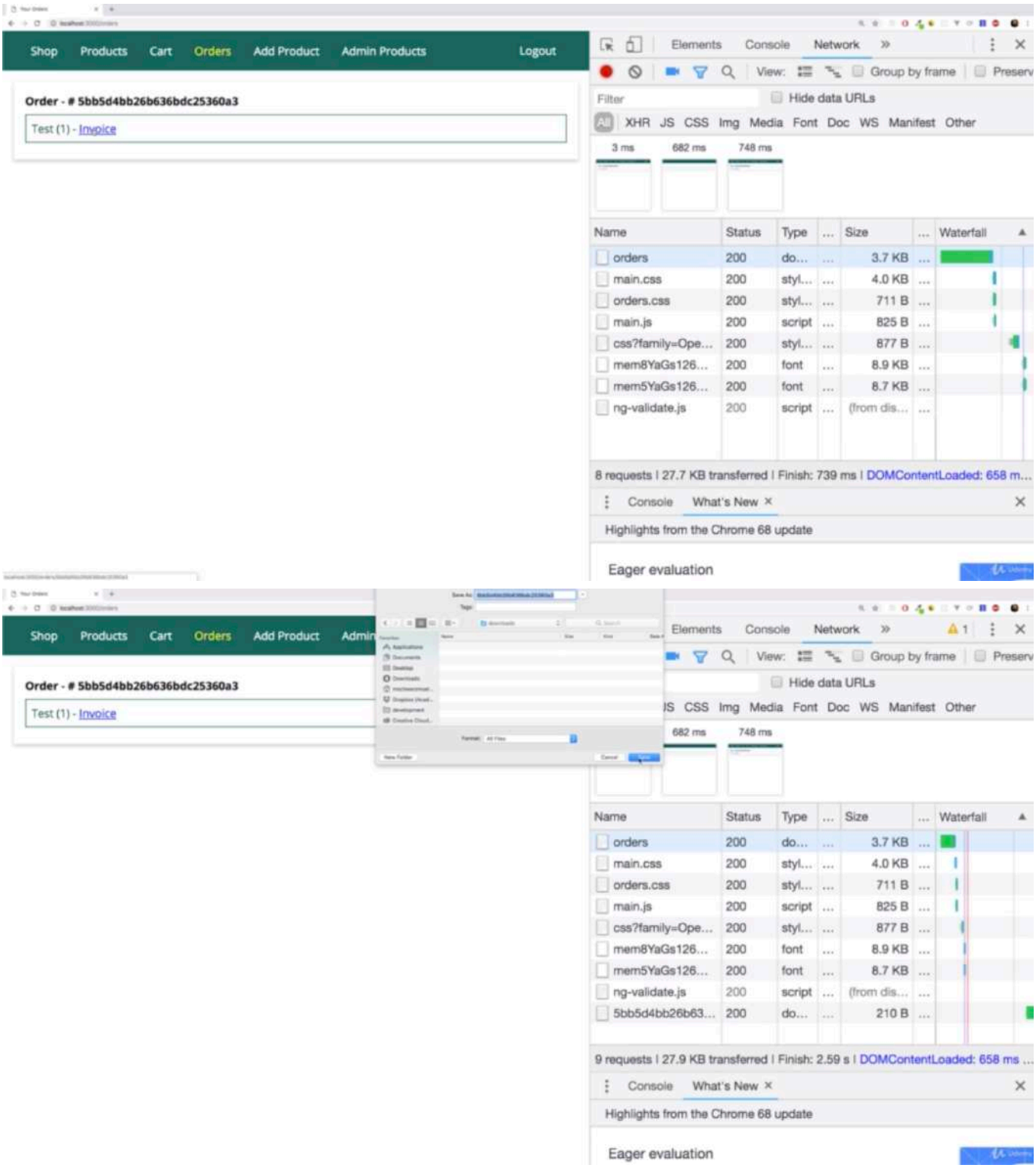

EXPLORER
app.js
orders.ejs
shop.js routes
invoice.pdf
products.ejs
index.ejs
The file is not displayed in the editor because it is either binary or uses an unsupported text encoding. Do you want to open it anyway?
invoice.pdf
cart.json
products.json
images
middleware
is-auth.js
models
node_modules
public
routes
admin.js
auth.js
shop.js
util
views
admin
edit-product.ejs
products.ejs
auth
includes
OUTLINE
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
1: node
_id: 5bb209393fa36b3687f778cd,
email: 'test@test.com',
password:
'$2a$12$3o67HvujGSySCwZaF3ghUeqavjVe/XjRXGjdsxntuXBjcsyexU2r6',
__v: 0 }
19-files*
0 0
22.32KB
Prettier:

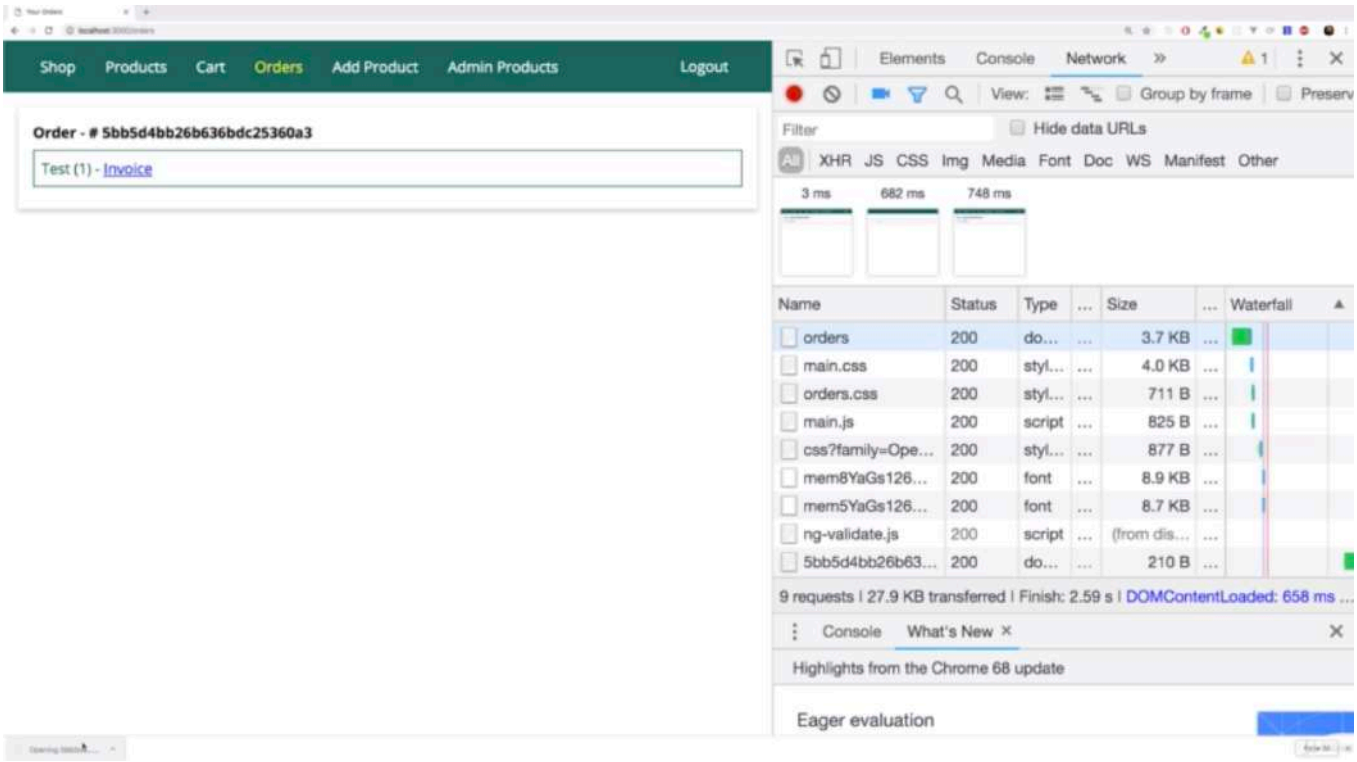

My Cluster
3 DBS
11 COLLECTIONS
filter
admin
local
shop
orders
products
sessions
users
Cluster0-shard-0
REPLICA SET
3 NODES
MongoDB 3.6.8 Enterprise
shop.orders
DOCUMENTS N/A
INDEXES N/A
Documents
Aggregations
Explain Plan
Indexes
FILTER
OPTIONS
FIND
RESET
INSERT DOCUMENT
VIEW
LIST
TABLE
Displaying documents 1 - 4 of 4
_id: ObjectId("5baddef8f37f6350c4a541ca")
user: Object
products: Array
__v: 0
_id: ObjectId("5bb5d4bb26b636bdc25360a3")
user : Object
products : Array
0 : Object
_id: ObjectId("5bb5d4bb26b636bdc25360a4")
quantity : 1
product : Object
_id: ObjectId("5bb5d223fb15bebc637fd17d")
title : "Test "
price : 12
description : "fasdfadsfs "
imageUrl : "images/2018-10-04T08:45:57.450Z-boat 2.png "
userId : ObjectId("5bb209393fa36b3687f778cd ")
__v : 0
__v : 0
CANCEL
UPDATE





- grab that latest object's _id in orders collection, and paste it to the invoice.pdf title name like 'invoice-5ccec2e256bcaa07fd3631f0.pdf'


Shop
Products
Cart
Orders
Add Product
Admin Products
Logout
Order - # 5bb5d4bb26b636bdc25360a3
Test (1) - Invoice
Elements
Console
Network
View:
Group by frame
Filter
Hide data URLs
XHR JS CSS Img Media Font Doc WS Manifest Other
3 ms
682 ms
748 ms
Name
Status
Type
Size
Waterfall
orders 200 do... 3.7 KB
main.css 200 styl... 4.0 KB
orders.css 200 styl... 711 B
main.js 200 script 825 B
css?family=Ope... 200 styl... 877 B
mem8YaGs126... 200 font 8.9 KB
mem5YaGs126... 200 font 8.7 KB
ng-validate.js 200 script (from dis...
8 requests | 27.7 KB transferred | Finish: 739 ms | DOMContentLoaded: 658 m...
Console
What's New
Highlights from the Chrome 68 update
Eager evaluation
Shop
Products
Cart
Orders
Add Product
Order - # 5bb5d4bb26b636bdc25360a3
Test (1) - Invoice
Elements
Console
Network
View:
Group by frame
Hide data URLs
682 ms
748 ms
Name
Status
Type
Size
Waterfall
orders 200 do... 3.7 KB
main.css 200 styl... 4.0 KB
orders.css 200 styl... 711 B
main.js 200 script 825 B
css?family=Ope... 200 styl... 877 B
mem8YaGs126... 200 font 8.9 KB
mem5YaGs126... 200 font 8.7 KB
ng-validate.js 200 script (from dis...
5bb5d4bb26b63... 200 do... 210 B
9 requests | 27.9 KB transferred | Finish: 2.59 s | DOMContentLoaded: 658 ms
Console
What's New
Highlights from the Chrome 68 update
Eager evaluation



- if i click 'invoice' anchor tag, and i get my download option here, let me save that and try to open that file. it should open as a PDF or you should be able to open it as a PDF but it was not the most convenient way of downloading this.

```
<!--./views/shop/orders.ejs-->

<%- include('../includes/head.ejs') %>
    <link rel="stylesheet" href="/css/orders.css">
    </head>

    <body>
        <%- include('../includes/navigation.ejs') %>
        <main>
            <% if (orders.length <= 0) { %>
                <h1>Nothing there!</h1>
            <% } else { %>
                <ul class="orders">
                    <% orders.forEach(order => { %>
                        <li class="orders__item">
                            <h1>Order - # <%= order._id %></h1>
                            <ul class="orders__products">
                                <% order.products.forEach(p => { %>
                                    <li class="orders__products-item"><%= p.product.title %> (<%= p.quantity %>) - <a href="/orders/<%= order._id %>">Invoice</a></li>
                                <% }); %>
                            </ul>
                        </li>
                    <% }); %>
                </ul>
            <% } %>
        </main>
        <%- include('../includes/end.ejs') %>
```

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

```

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

const express = require('express');

const shopController = require('../controllers/shop');
const isAuth = require('../middleware/is-auth');

const router = express.Router();

router.get('/', shopController.getIndex);

router.get('/products', shopController.getProducts);

router.get('/products/:productId', shopController.getProduct);

router.get('/cart', isAuth, shopController.getCart);

router.post('/cart', isAuth, shopController.postCart);

router.post('/cart-delete-item', isAuth, shopController.postCartDeleteProduct);

router.post('/create-order', isAuth, shopController.postOrder);

router.get('/orders', isAuth, shopController.getOrders);

router.get('/orders/:orderId', isAuth, shopController.getInvoice)

module.exports = router;
```

```
//./controllers/shop.js

const fs = require('fs')
const path = require('path')

const Product = require('../models/product');
const Order = require('../models/order');

exports.getProducts = (req, res, next) => {
  Product.find()
    .then(products => {
      console.log(products);
      res.render('shop/product-list', {
        prods: products,
        pageTitle: 'All Products',
        path: '/products'
      });
    })
    .catch(err => {
      const error = new Error(err)
      error.httpStatusCode = 500
      return next(error)
    });
};

exports.getProduct = (req, res, next) => {
  const prodId = req.params.productId;
  Product.findById(prodId)
```

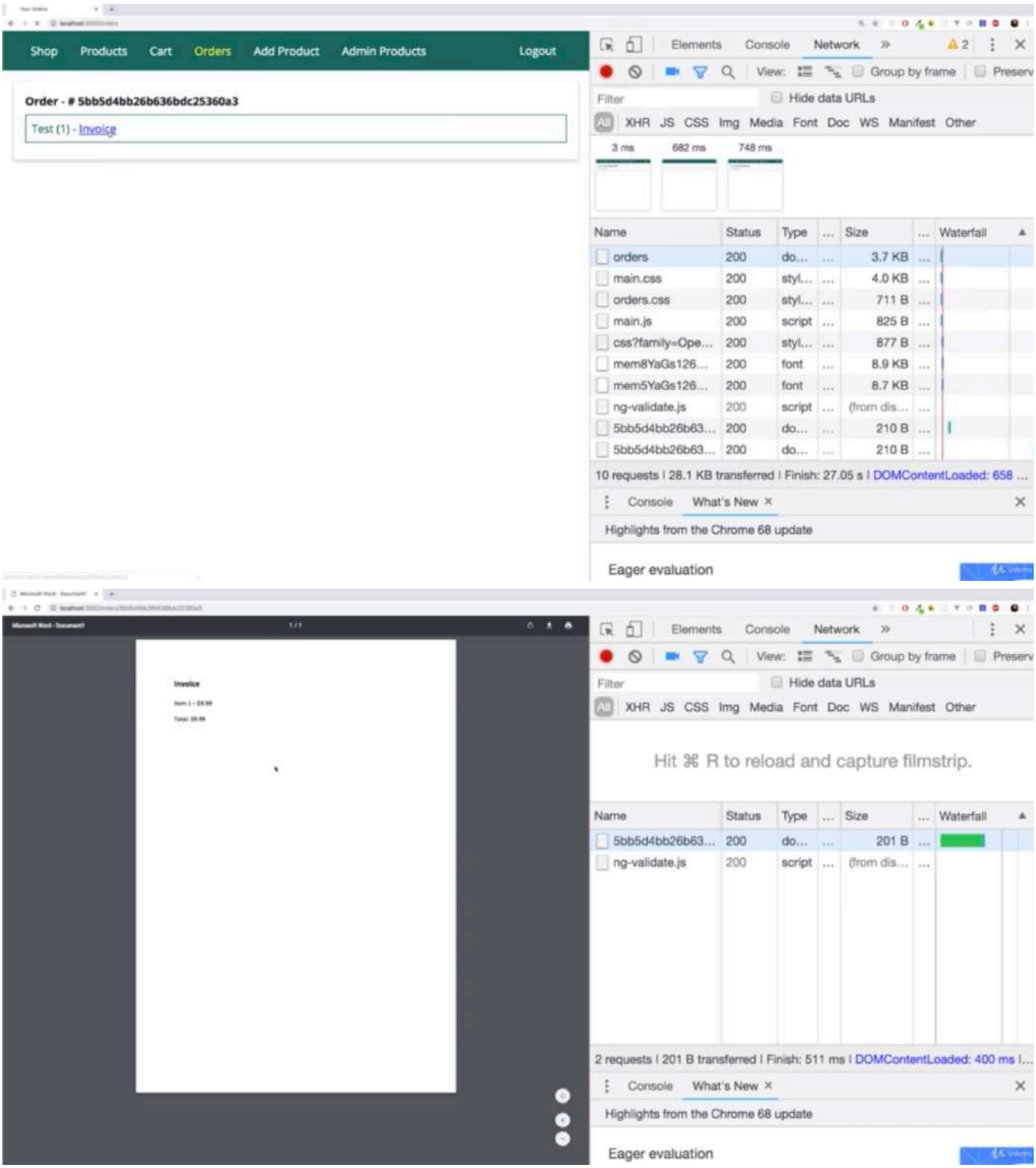
```
    .then(product => {
      res.render('shop/product-detail', {
        product: product,
        pageTitle: product.title,
        path: '/products'
      });
    })
    .catch(err => {
      const error = new Error(err)
      error.httpStatusCode = 500
      return next(error)
    });
};

exports.getIndex = (req, res, next) => {
  Product.find()
    .then(products => {
      res.render('shop/index', {
        prods: products,
        pageTitle: 'Shop',
        path: '/'
      });
    })
    .catch(err => {
      const error = new Error(err)
      error.httpStatusCode = 500
      return next(error)
    });
};

exports.getCart = (req, res, next) => {
  req.user
    .populate('cart.items.productId')
    .execPopulate()
    .then(user => {
      const products = user.cart.items;
      res.render('shop/cart', {
        path: '/cart',
        pageTitle: 'Your Cart',
        products: products
      });
    })
    .catch(err => {
      const error = new Error(err)
      error.httpStatusCode = 500
      return next(error)
    });
};

exports.postCart = (req, res, next) => {
  const prodId = req.body.productId;
  Product.findById(prodId)
    .then(product => {
      return req.user.addToCart(product);
    })
    .then(result => {
```

```
      console.log(result);
      res.redirect('/cart');
    })
    .catch(err => {
      const error = new Error(err)
      error.httpStatusCode = 500
      return next(error)
    })
};

exports.postCartDeleteProduct = (req, res, next) => {
  const prodId = req.body.productId;
  req.user
    .removeFromCart(prodId)
    .then(result => {
      res.redirect('/cart');
    })
    .catch(err => {
      const error = new Error(err)
      error.httpStatusCode = 500
      return next(error)
    });
};

exports.postOrder = (req, res, next) => {
  req.user
    .populate('cart.items.productId')
    .execPopulate()
    .then(user => {
      const products = user.cart.items.map(i => {
        return { quantity: i.quantity, product: { ...i.productId._doc } };
      });
      const order = new Order({
        user: {
          email: req.user.email,
          userId: req.user
        },
        products: products
      });
      return order.save();
    })
    .then(result => {
      return req.user.clearCart();
    })
    .then(() => {
      res.redirect('/orders');
    })
    .catch(err => {
      const error = new Error(err)
      error.httpStatusCode = 500
      return next(error)
    });
};

exports.getOrders = (req, res, next) => {
  Order.find({ 'user.userId': req.user._id })
```

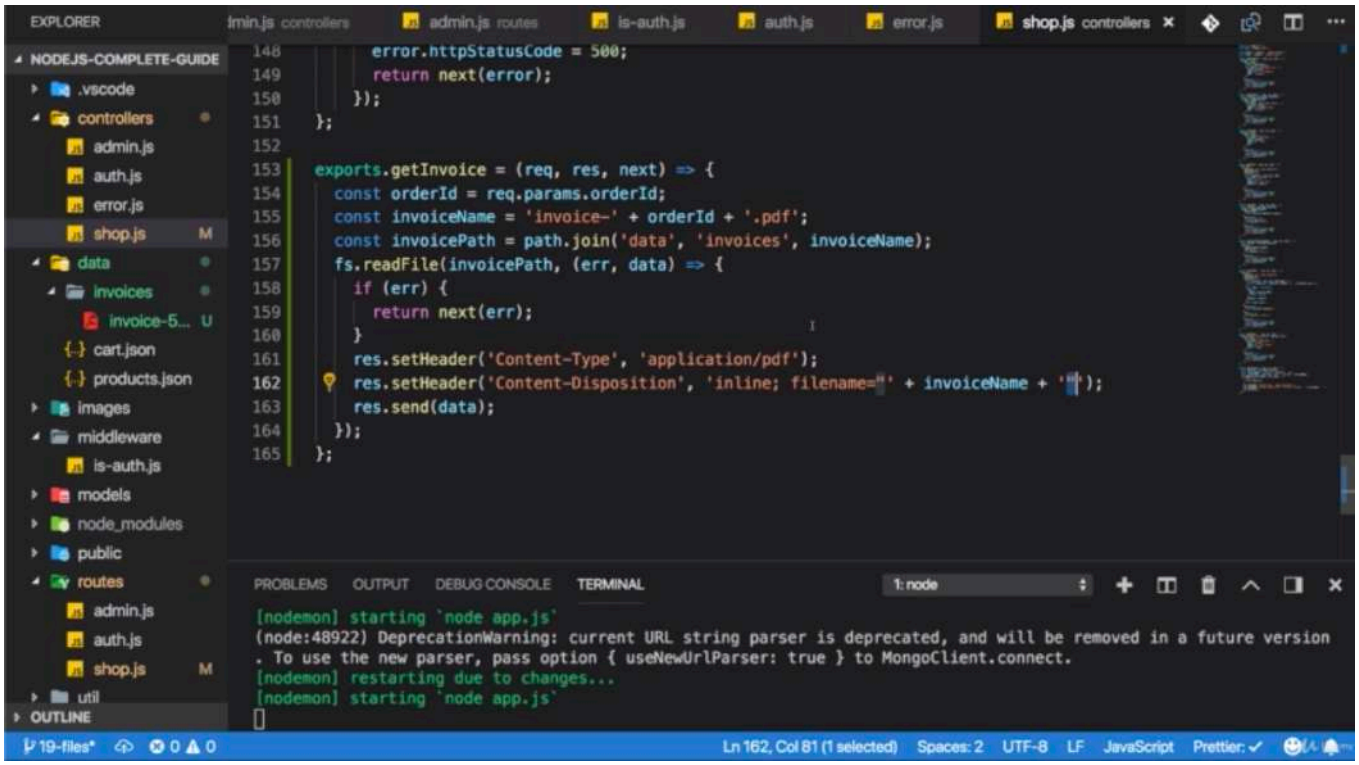
```
    .then(orders => {
      res.render('shop/orders', {
        path: '/orders',
        pageTitle: 'Your Orders',
        orders: orders
      });
    })
    .catch(err => {
      const error = new Error(err)
      error.httpStatusCode = 500
      return next(error)
    });
};

exports.getInvoice = (req, res, next) => {
  /**request and encoded in the url
   * so it's params and then orderId
   *
   * that is specified in ./routes/shop.js
   *
   * and now we need to retrieve that file
   * and we can retrieve files with nodes file system.
   * now the path should be constructed with the path core module
   * so that it works on all operating system
   */
  const orderId = req.params.orderId
  const invoiceName = 'invoice-' + orderId + '.pdf'
  const invoicePath = path.join('data', 'invoices', invoiceName)
  fs.readFile(invoicePath, (err, data) => {
    if (err) {
      return next(err)
    }
    res.send(data)
  })
}
```

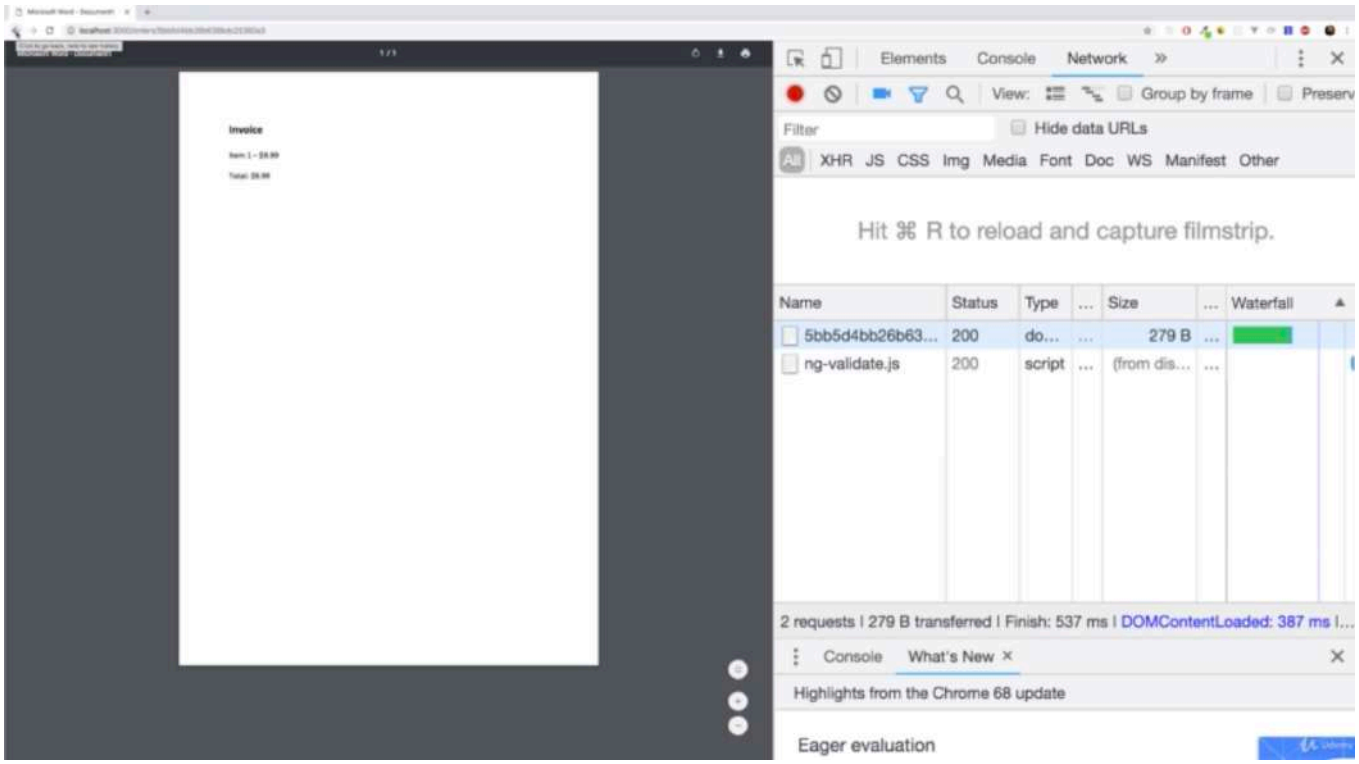
* Chapter 324: Setting File Type Headers

1. update
- ./controllers/shop.js

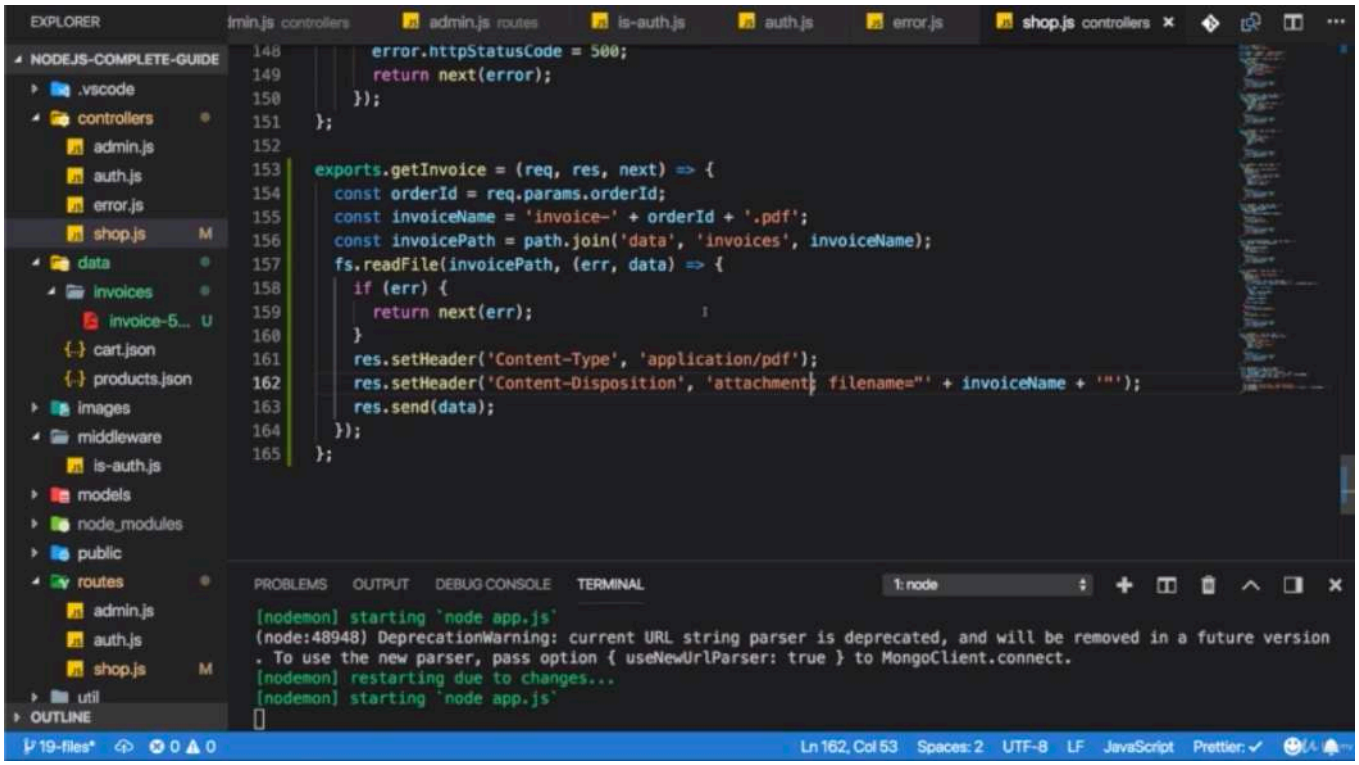



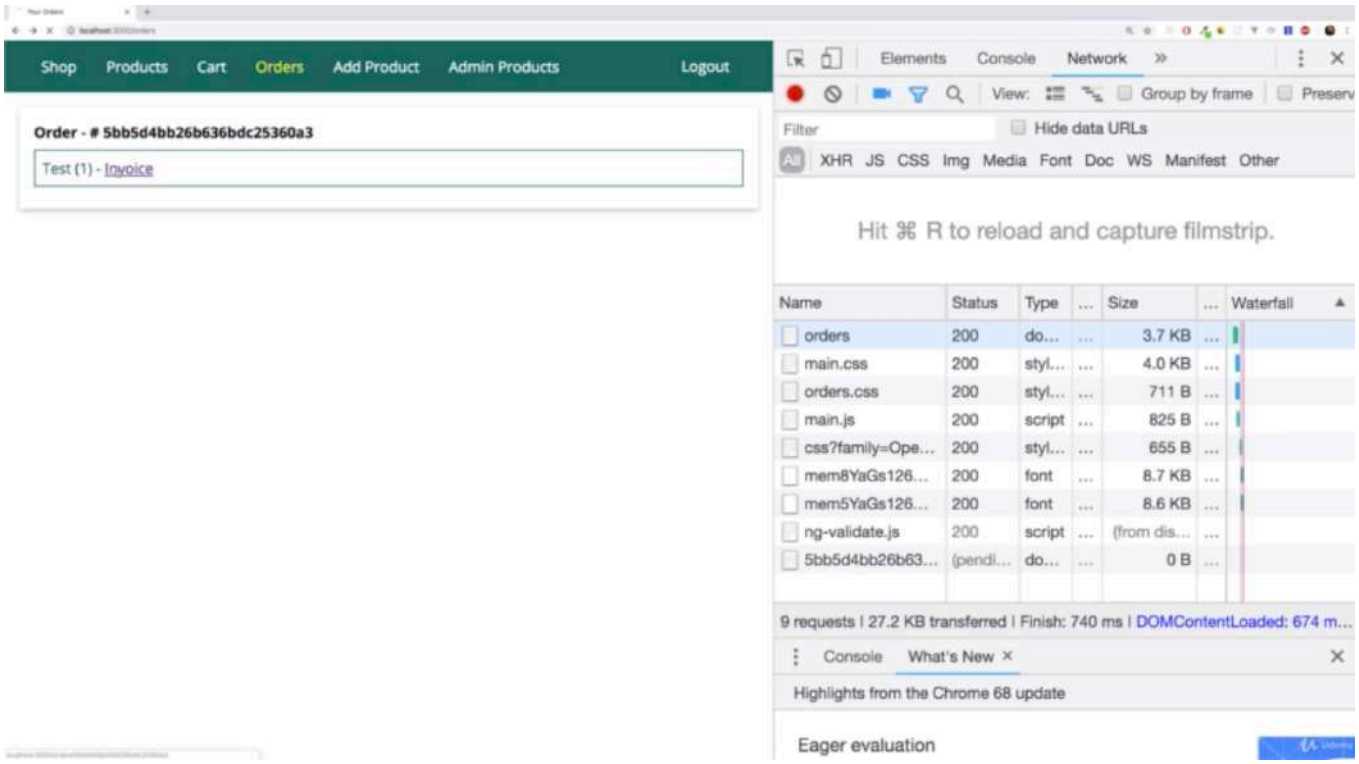
- if i click this 'invoice' button, now i open it in the browser. so this already changes the behavior. it already gives the browser some information which allows the browser to handle this in a better way.

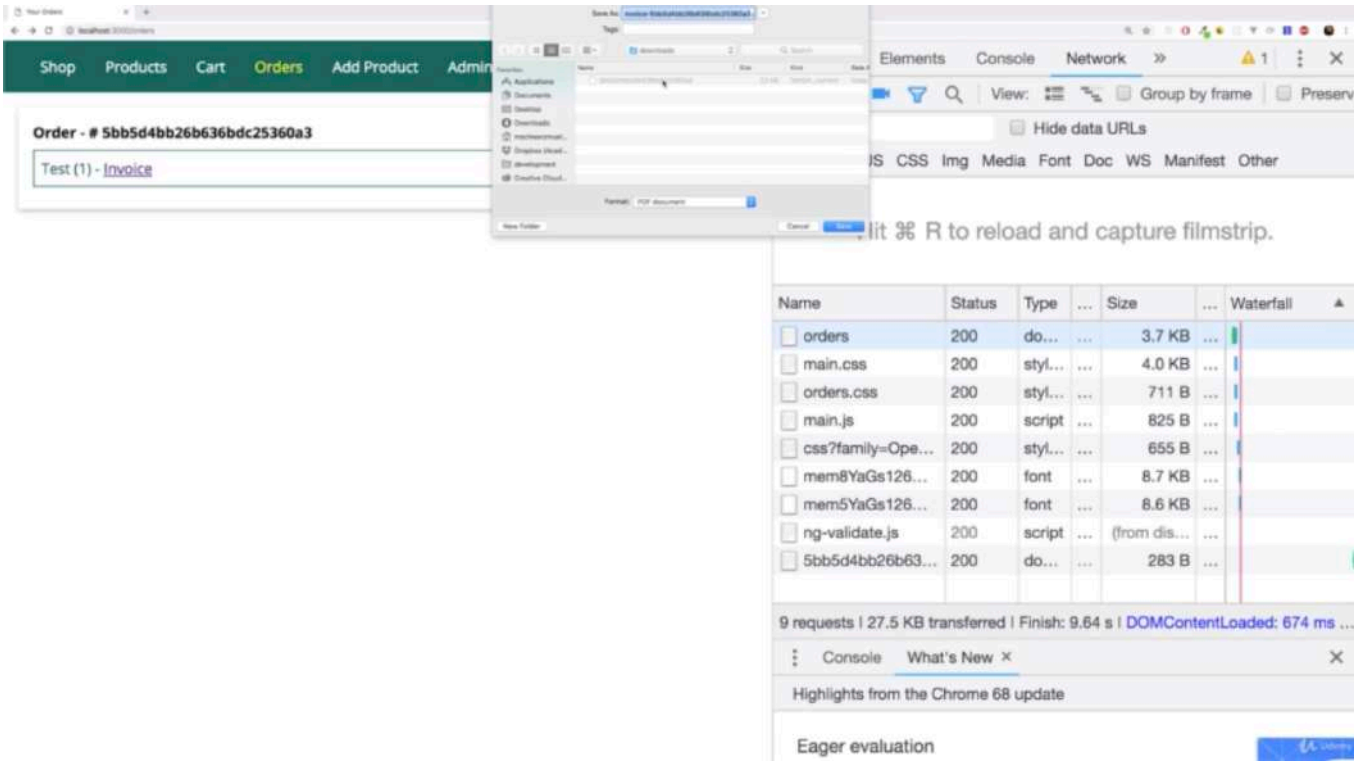

EXPLORER
NODEJS-COMPLETE-GUIDE
.vscode
controllers
admin.js
auth.js
error.js
shop.js
data
invoices
cart.json
products.json
images
middleware
is-auth.js
models
node_modules
public
routes
admin.js
auth.js
shop.js
util
OUTLINE
error.httpStatusCode = 500;
return next(error);
});
};
exports.getInvoice = (req, res, next) => {
const orderId = req.params.orderId;
const invoiceName = 'invoice-' + orderId + '.pdf';
const invoicePath = path.join('data', 'invoices', invoiceName);
fs.readFile(invoicePath, (err, data) => {
if (err) {
return next(err);
}
res.setHeader('Content-Type', 'application/pdf');
res.setHeader('Content-Disposition', 'inline; filename="' + invoiceName + '"');
res.send(data);
});
};
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
1: node
[nodemon] starting `node app.js`
(node:48922) 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.
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
Ln 162, Col 81 (1 selected)
Spaces: 2
UTF-8
LF
JavaScript
Prettier:


Invoice
Elements
Console
Network
Hide data URLs
XHR JS CSS Img Media Font Doc WS Manifest Other
Hit ⌘ R to reload and capture filmstrip.
Name
Status
Type
Size
Waterfall
5bb5d4bb26b63...
200
279 B
ng-validate.js
200
script
2 requests | 279 B transferred | Finish: 537 ms | DOMContentLoaded: 387 ms
Console
What's New
Highlights from the Chrome 68 update
Eager evaluation

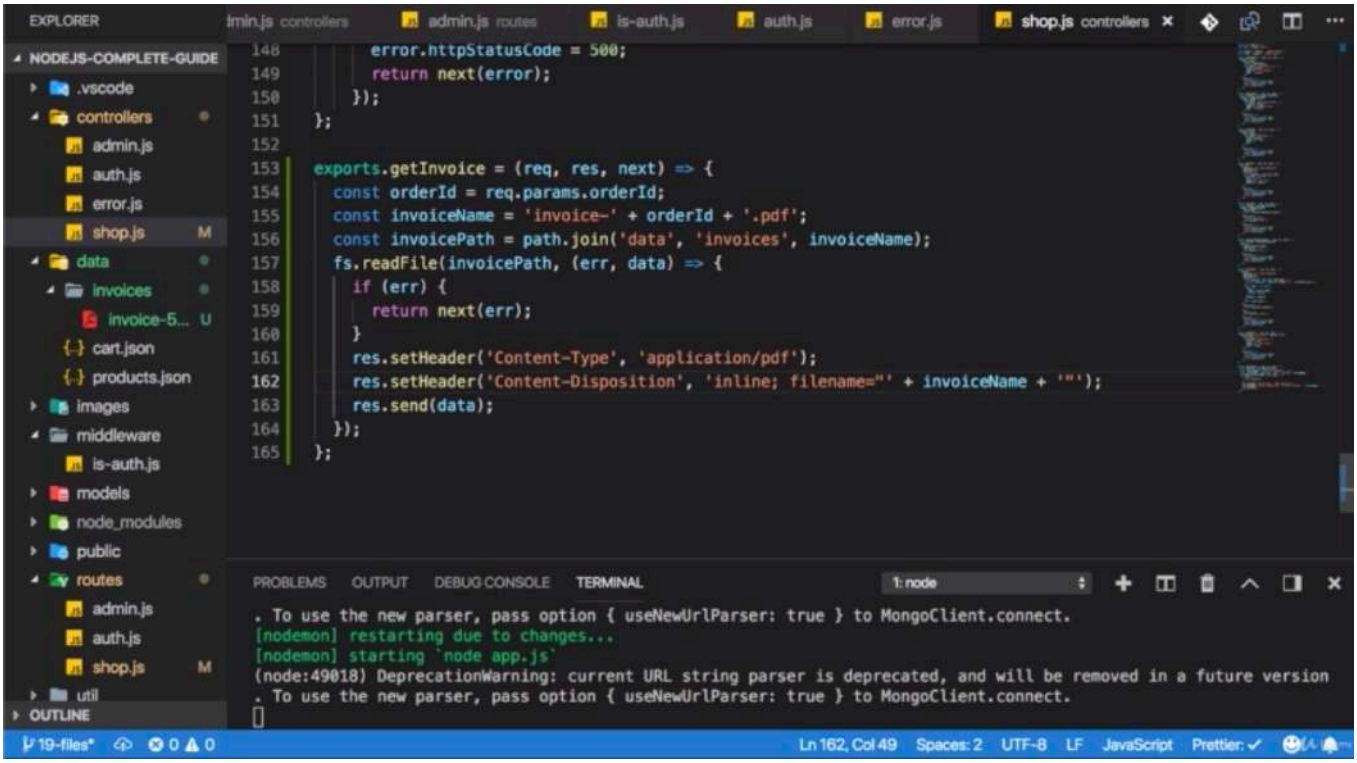
————————————————————


EXPLORER
admin.js routes
is-auth.js
auth.js
error.js
shop.js controllers
NODEJS-COMPLETE-GUIDE
.vscode
controllers
admin.js
auth.js
error.js
shop.js
data
invoices
invoice-5...
cart.json
products.json
images
middleware
is-auth.js
models
node_modules
public
routes
admin.js
auth.js
shop.js
util
OUTLINE
error.httpStatusCode = 500;
return next(error);
});
};
exports.getInvoice = (req, res, next) => {
const orderId = req.params.orderId;
const invoiceName = 'invoice-' + orderId + '.pdf';
const invoicePath = path.join('data', 'invoices', invoiceName);
fs.readFile(invoicePath, (err, data) => {
if (err) {
return next(err);
}
res.setHeader('Content-Type', 'application/pdf');
res.setHeader('Content-Disposition', 'attachment; filename="' + invoiceName + '"');
res.send(data);
});
};
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
1: node
[nodemon] starting `node app.js`
(node:48948) 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.
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
19-files*
0 0
Ln 162, Col 53
Spaces: 2
UTF-8
LF
JavaScript
Prettier:


Shop
Products
Cart
Orders
Add Product
Admin Products
Logout
Order - # 5bb5d4bb26b636bdc25360a3
Test (1) - Invoice
Elements
Console
Network
View:
Group by frame
Filter
Hide data URLs
All
XHR
JS
CSS
Img
Media
Font
Doc
WS
Manifest
Other
Hit ⌘ R to reload and capture filmstrip.
Name
Status
Type
Size
Waterfall
orders 200 do... 3.7 KB
main.css 200 styl... 4.0 KB
orders.css 200 styl... 711 B
main.js 200 script 825 B
css?family=Ope... 200 styl... 655 B
mem8YaGs126... 200 font 8.7 KB
mem5YaGs126... 200 font 8.6 KB
ng-validate.js 200 script (from dis...
5bb5d4bb26b63... (pendi... do... 0 B
9 requests | 27.2 KB transferred | Finish: 740 ms | DOMContentLoaded: 674 m...
Console
What's New
Highlights from the Chrome 68 update
Eager evaluation



- and if i save it and click 'invoice' button, the download menu opens again. and we have the proper file name with the proper extension. this is how you can control how the browser should handle the incoming data.



- now i have a set up where only authenticated users can requests this invoice. we can still improve that.

```
//./controllers/shop.js

const fs = require('fs')
const path = require('path')

const Product = require('../models/product');
const Order = require('../models/order');

exports.getProducts = (req, res, next) => {
```

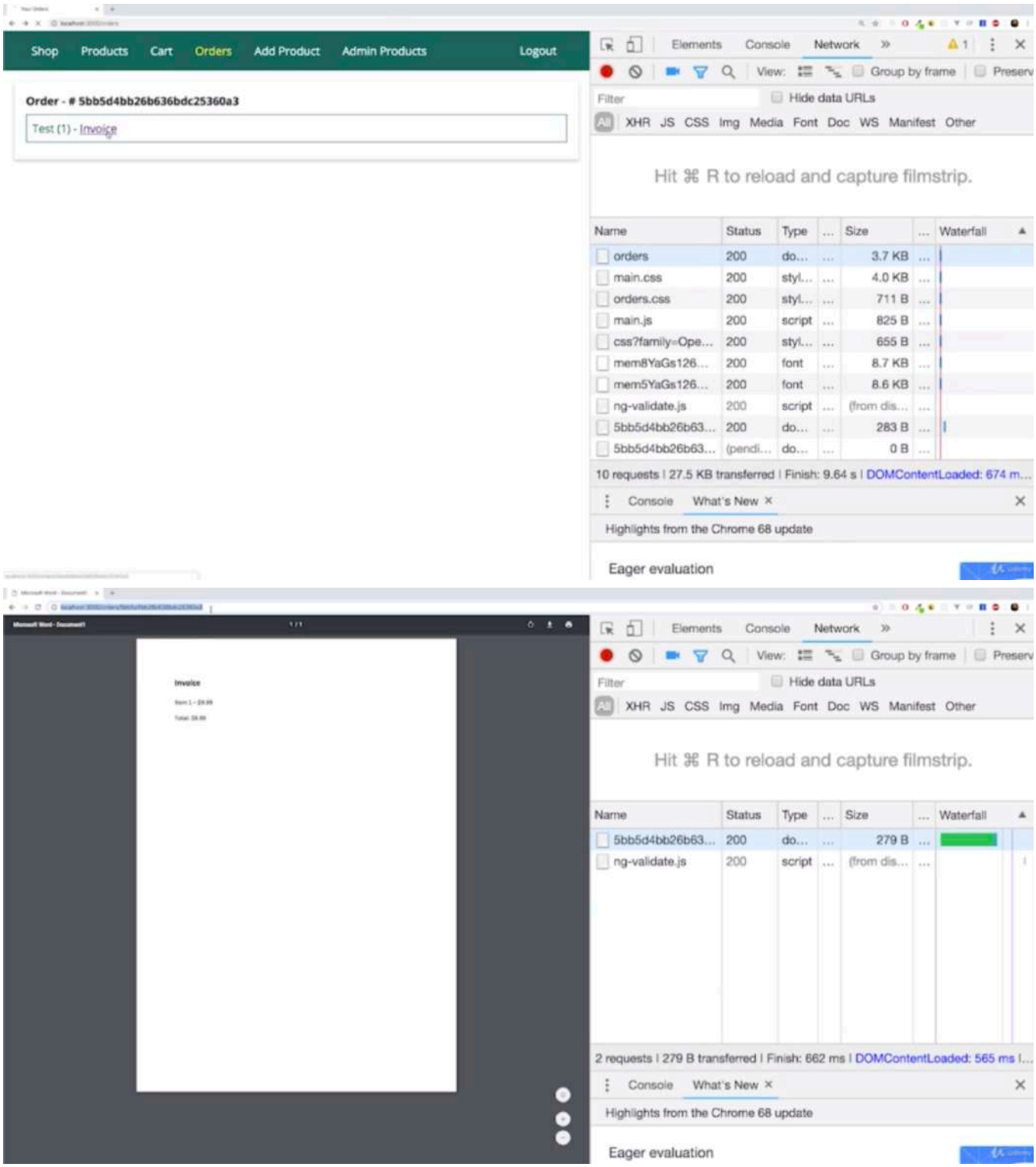
```
  Product.find()
    .then(products => {
      console.log(products);
      res.render('shop/product-list', {
        prods: products,
        pageTitle: 'All Products',
        path: '/products'
      });
    })
    .catch(err => {
      const error = new Error(err)
      error.httpStatusCode = 500
      return next(error)
    });
};

exports.getProduct = (req, res, next) => {
  const prodId = req.params.productId;
  Product.findById(prodId)
    .then(product => {
      res.render('shop/product-detail', {
        product: product,
        pageTitle: product.title,
        path: '/products'
      });
    })
    .catch(err => {
      const error = new Error(err)
      error.httpStatusCode = 500
      return next(error)
    });
};

exports.getIndex = (req, res, next) => {
  Product.find()
    .then(products => {
      res.render('shop/index', {
        prods: products,
        pageTitle: 'Shop',
        path: '/'
      });
    })
    .catch(err => {
      const error = new Error(err)
      error.httpStatusCode = 500
      return next(error)
    });
};

exports.getCart = (req, res, next) => {
  req.user
    .populate('cart.items.productId')
    .execPopulate()
    .then(user => {
      const products = user.cart.items;
      res.render('shop/cart', {
```

```
      path: '/cart',
      pageTitle: 'Your Cart',
      products: products
    });
  })
  .catch(err => {
    const error = new Error(err)
    error.httpStatusCode = 500
    return next(error)
  });
};

exports.postCart = (req, res, next) => {
  const prodId = req.body.productId;
  Product.findById(prodId)
    .then(product => {
      return req.user.addToCart(product);
    })
    .then(result => {
      console.log(result);
      res.redirect('/cart');
    })
    .catch(err => {
      const error = new Error(err)
      error.httpStatusCode = 500
      return next(error)
    })
};

exports.postCartDeleteProduct = (req, res, next) => {
  const prodId = req.body.productId;
  req.user
    .removeFromCart(prodId)
    .then(result => {
      res.redirect('/cart');
    })
    .catch(err => {
      const error = new Error(err)
      error.httpStatusCode = 500
      return next(error)
    });
};

exports.postOrder = (req, res, next) => {
  req.user
    .populate('cart.items.productId')
    .execPopulate()
    .then(user => {
      const products = user.cart.items.map(i => {
        return { quantity: i.quantity, product: { ...i.productId._doc } };
      });
      const order = new Order({
        user: {
          email: req.user.email,
          userId: req.user
        },
```

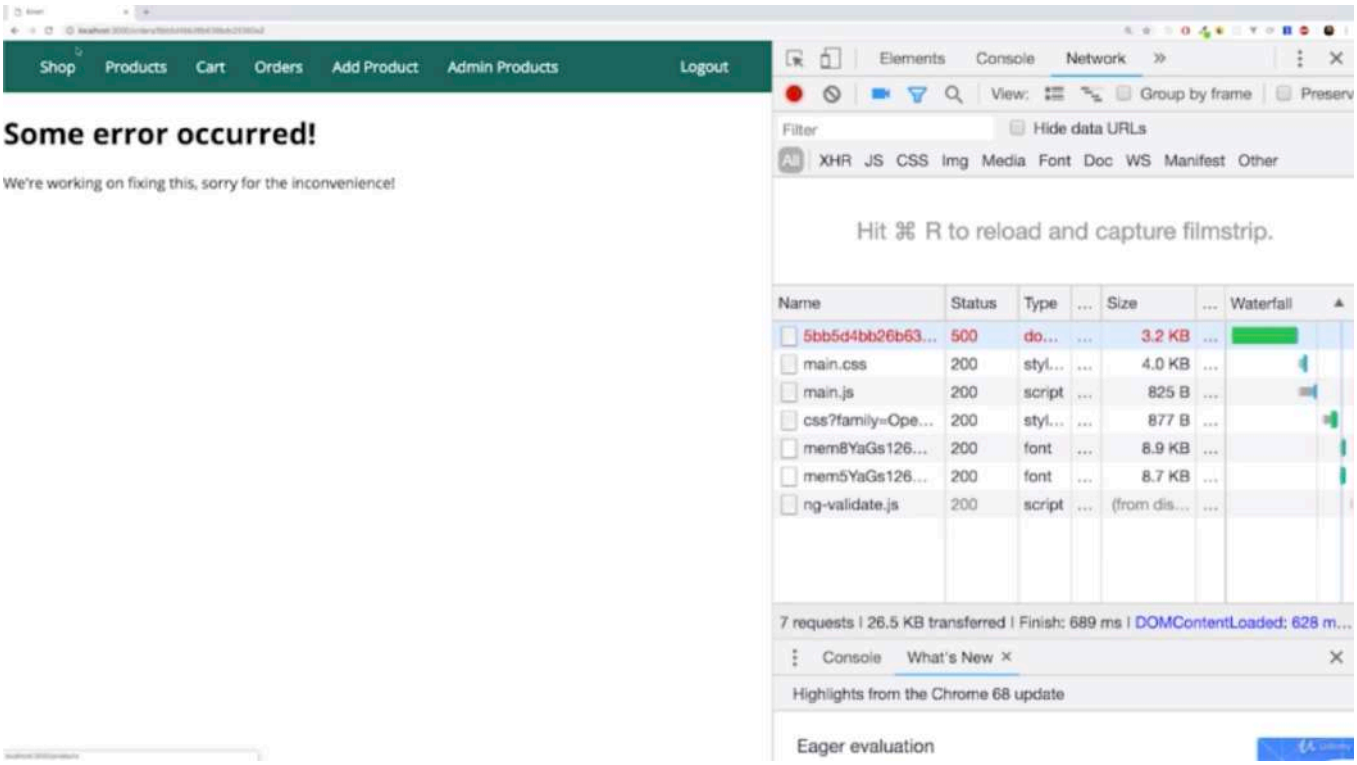
```
      products: products
    });
    return order.save();
  })
  .then(result => {
    return req.user.clearCart();
  })
  .then(() => {
    res.redirect('/orders');
  })
  .catch(err => {
    const error = new Error(err)
    error.httpStatusCode = 500
    return next(error)
  });
};

exports.getOrders = (req, res, next) => {
  Order.find({ 'user.userId': req.user._id })
    .then(orders => {
      res.render('shop/orders', {
        path: '/orders',
        pageTitle: 'Your Orders',
        orders: orders
      });
    })
    .catch(err => {
      const error = new Error(err)
      error.httpStatusCode = 500
      return next(error)
    });
};

exports.getInvoice = (req, res, next) => {
  const orderId = req.params.orderId
  const invoiceName = 'invoice-' + orderId + '.pdf'
  const invoicePath = path.join('data', 'invoices', invoiceName)
  fs.readFile(invoicePath, (err, data) => {
    if (err) {
      return next(err)
    }
    res.setHeader('Content-Type', 'application/pdf')
    /**'Content-Disposition' allows us to define how this content should be served to the client.
     * and we can set this to 'inline' to tell the browser to open it inline.
     *
     * and we can also add filename
     */
    res.setHeader('Content-Disposition', 'inline; filename="' + invoiceName + '"' )
    res.send(data)
  })
}
```

* Chapter 325: Restricting File Access

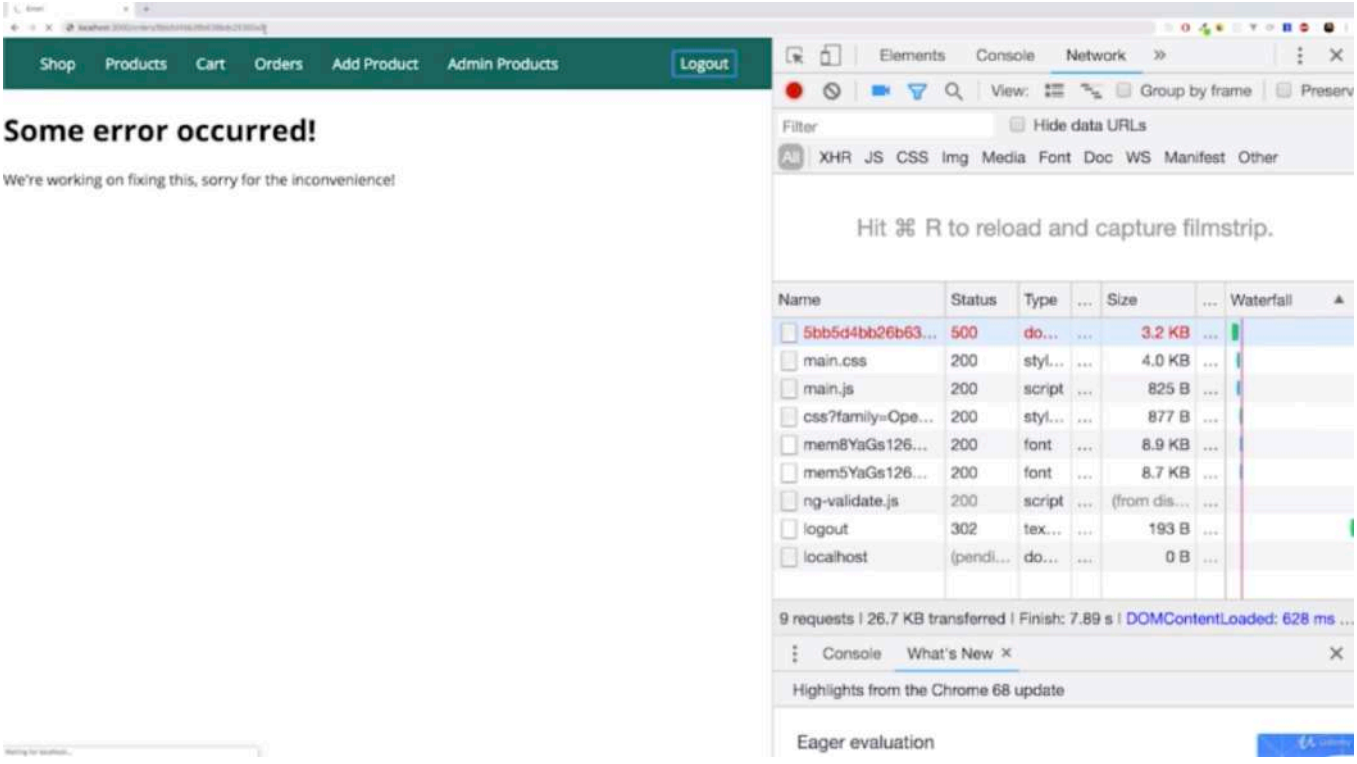
1. update
- ./controllers/shop.js

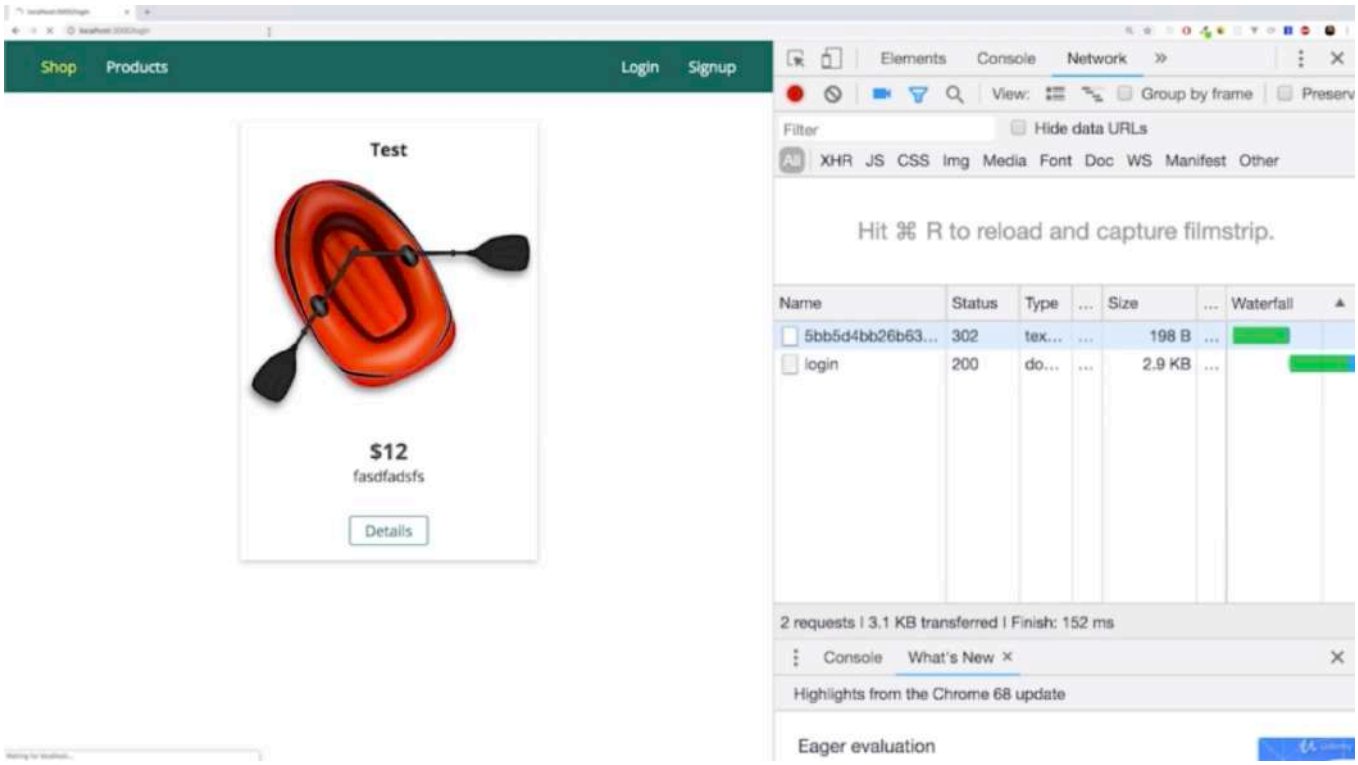


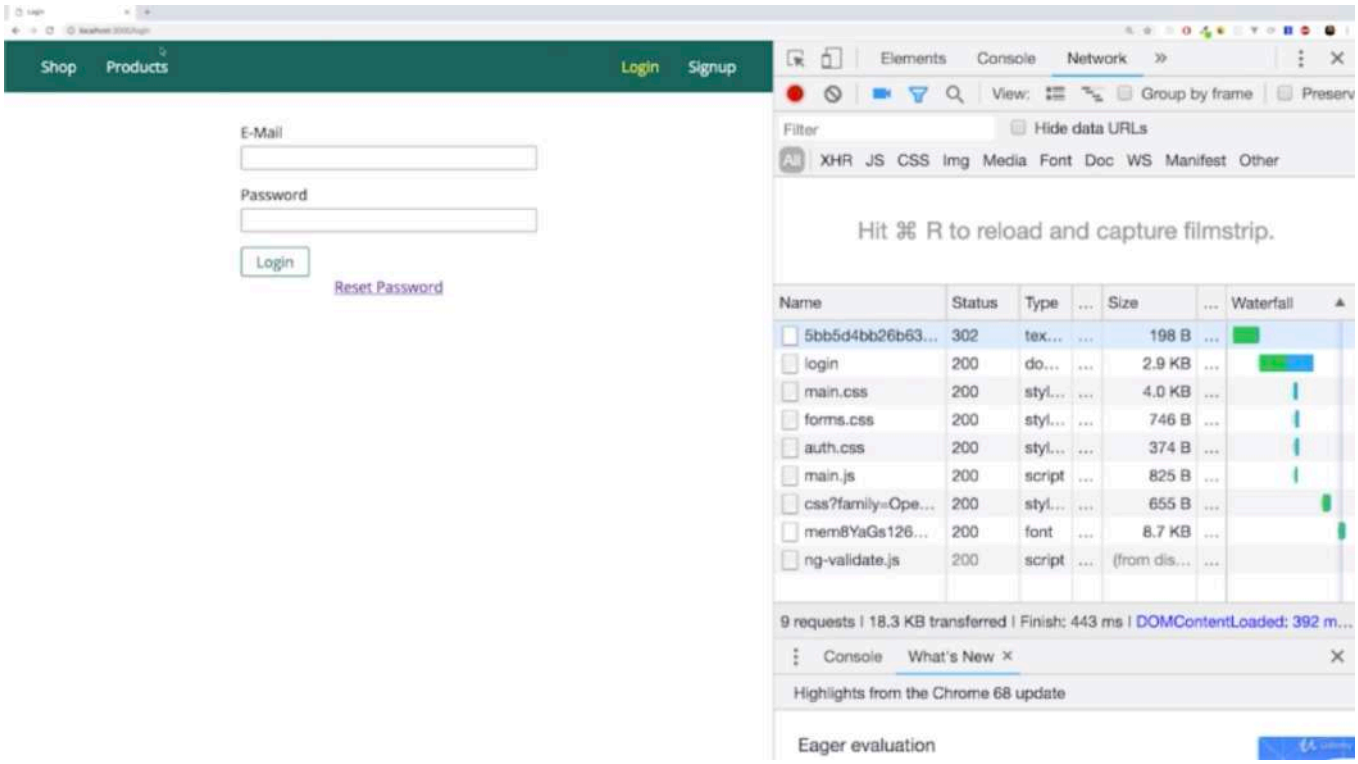
- it works



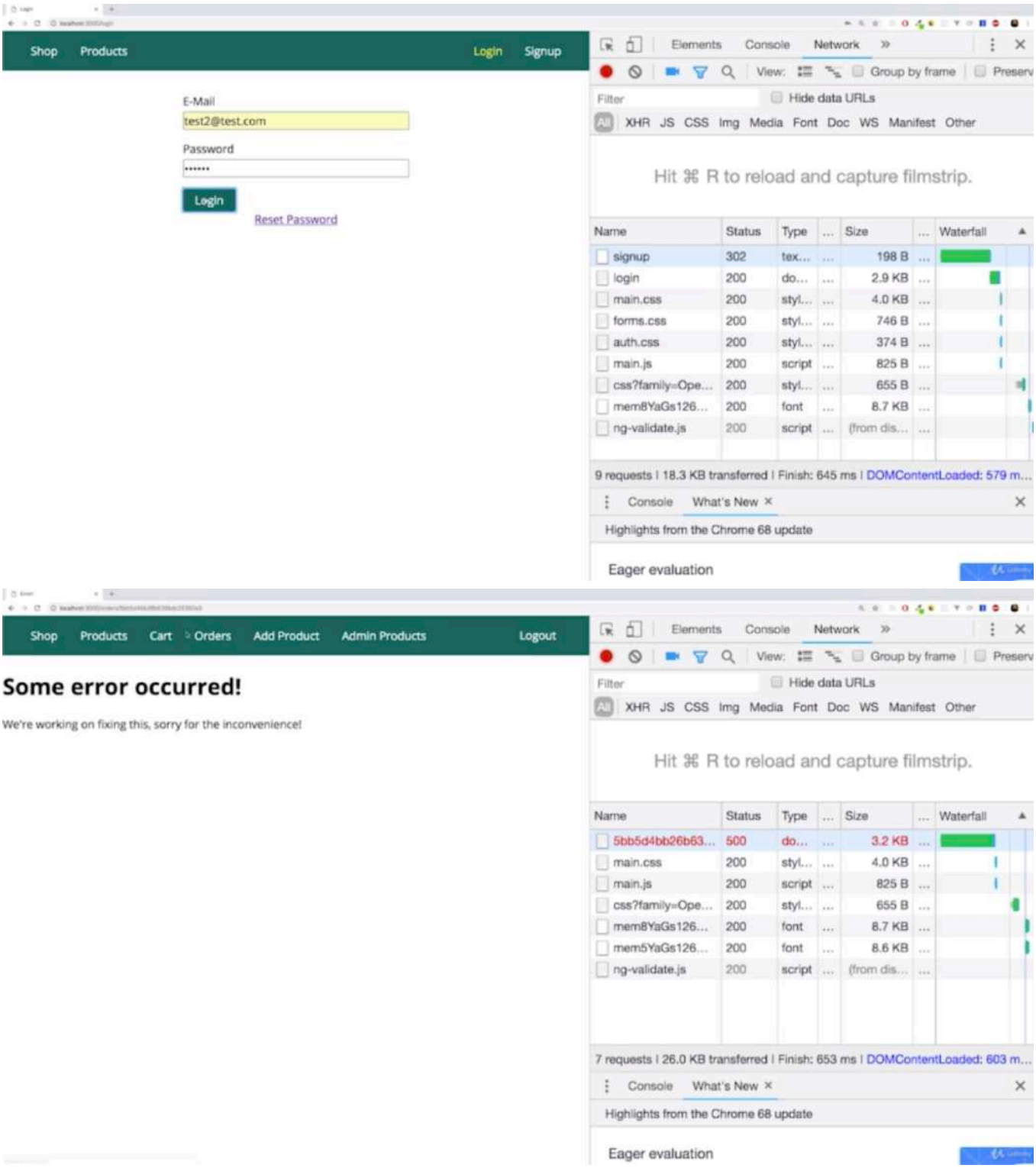
- and if i change 3 here to 2 in the URL, then i get my error because it's an invalid URL







- and if i logout and try to access this and access the original URL, i can't see that.



- if i log in with the different user, and i try to access the original order before, i can't access that. i get error.

```
//./controllers/shop.js

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

const Product = require('../models/product');
const Order = require('../models/order');

exports.getProducts = (req, res, next) => {
  Product.find()
    .then(products => {
      console.log(products);
      res.render('shop/product-list', {
```

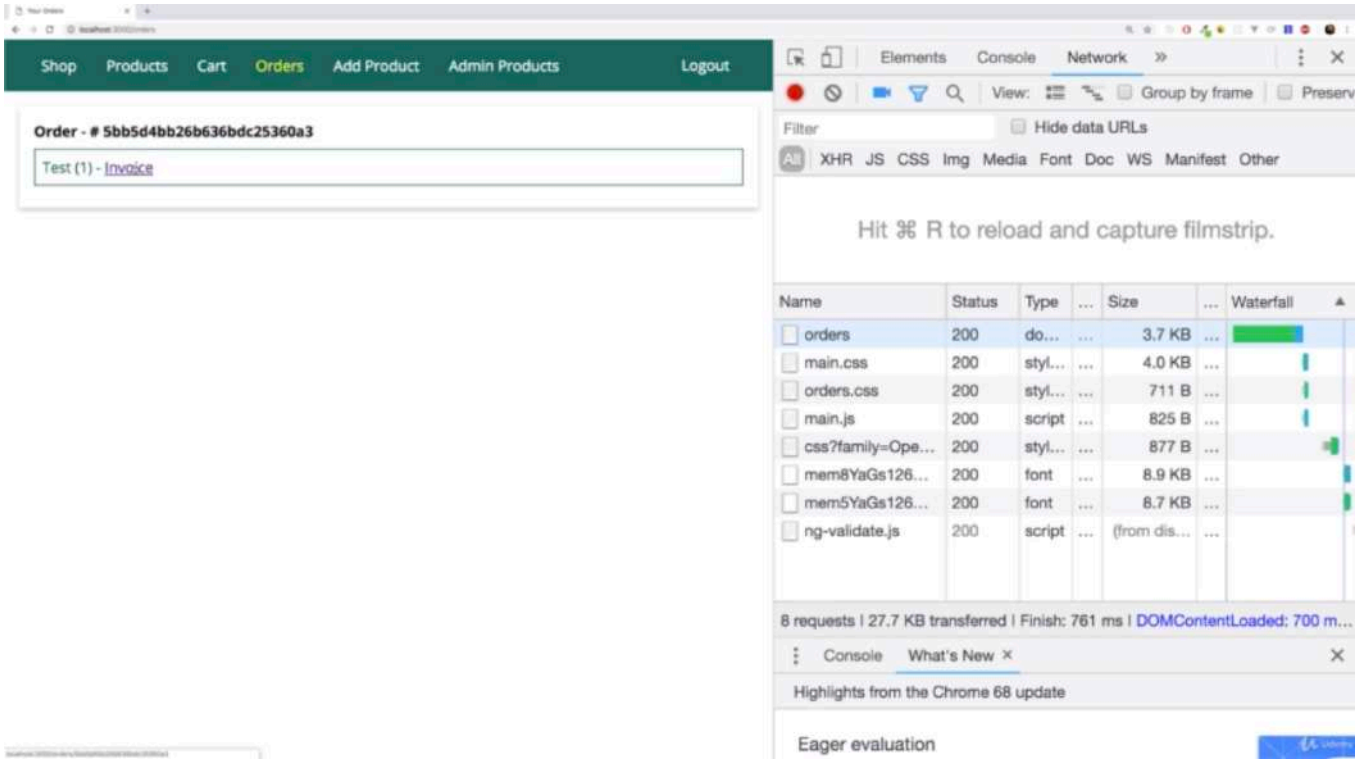
```
      prods: products,
      pageTitle: 'All Products',
      path: '/products'
    });
  })
  .catch(err => {
    const error = new Error(err);
    error.httpStatusCode = 500;
    return next(error);
  });
};

exports.getProduct = (req, res, next) => {
  const prodId = req.params.productId;
  Product.findById(prodId)
    .then(product => {
      res.render('shop/product-detail', {
        product: product,
        pageTitle: product.title,
        path: '/products'
      });
    })
    .catch(err => {
      const error = new Error(err);
      error.httpStatusCode = 500;
      return next(error);
    });
};

exports.getIndex = (req, res, next) => {
  Product.find()
    .then(products => {
      res.render('shop/index', {
        prods: products,
        pageTitle: 'Shop',
        path: '/'
      });
    })
    .catch(err => {
      const error = new Error(err);
      error.httpStatusCode = 500;
      return next(error);
    });
};

exports.getCart = (req, res, next) => {
  req.user
    .populate('cart.items.productId')
    .execPopulate()
    .then(user => {
      const products = user.cart.items;
      res.render('shop/cart', {
        path: '/cart',
        pageTitle: 'Your Cart',
        products: products
      });
```

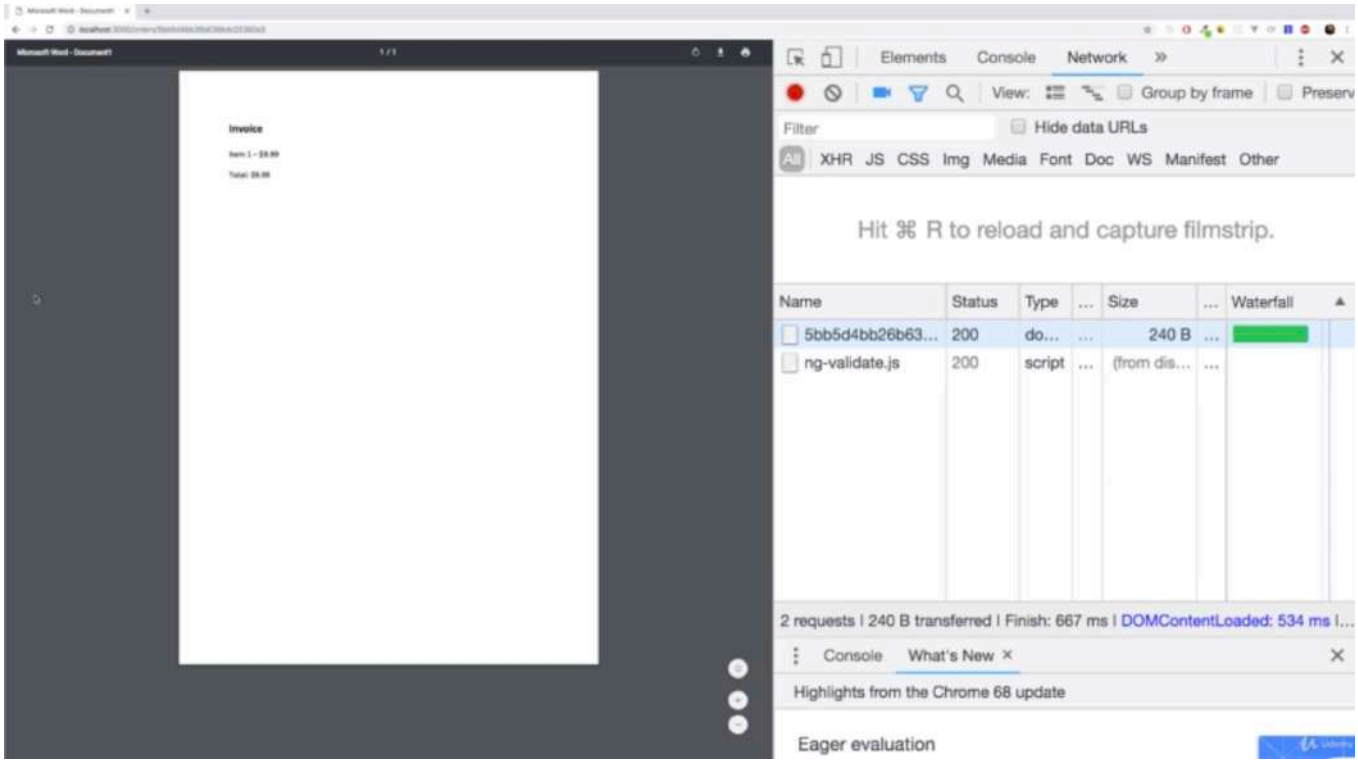
```
    })
    .catch(err => {
      const error = new Error(err);
      error.httpStatusCode = 500;
      return next(error);
    });
};

exports.postCart = (req, res, next) => {
  const prodId = req.body.productId;
  Product.findById(prodId)
    .then(product => {
      return req.user.addToCart(product);
    })
    .then(result => {
      console.log(result);
      res.redirect('/cart');
    })
    .catch(err => {
      const error = new Error(err);
      error.httpStatusCode = 500;
      return next(error);
    });
};

exports.postCartDeleteProduct = (req, res, next) => {
  const prodId = req.body.productId;
  req.user
    .removeFromCart(prodId)
    .then(result => {
      res.redirect('/cart');
    })
    .catch(err => {
      const error = new Error(err);
      error.httpStatusCode = 500;
      return next(error);
    });
};

exports.postOrder = (req, res, next) => {
  req.user
    .populate('cart.items.productId')
    .execPopulate()
    .then(user => {
      const products = user.cart.items.map(i => {
        return { quantity: i.quantity, product: { ...i.productId._doc } };
      });
      const order = new Order({
        user: {
          email: req.user.email,
          userId: req.user
        },
        products: products
      });
      return order.save();
    })
```

```
    .then(result => {
      return req.user.clearCart();
    })
    .then(() => {
      res.redirect('/orders');
    })
    .catch(err => {
      const error = new Error(err);
      error.httpStatusCode = 500;
      return next(error);
    });
};

exports.getOrders = (req, res, next) => {
  Order.find({ 'user.userId': req.user._id })
    .then(orders => {
      res.render('shop/orders', {
        path: '/orders',
        pageTitle: 'Your Orders',
        orders: orders
      });
    })
    .catch(err => {
      const error = new Error(err);
      error.httpStatusCode = 500;
      return next(error);
    });
};

exports.getInvoice = (req, res, next) => {
  const orderId = req.params.orderId;
  Order.findById(orderId)
    .then(order => {
      if (!order) {
        return next(new Error('No order found.'));
      }
      if (order.user.userId.toString() !== req.user._id.toString()) {
        return next(new Error('Unauthorized'));
      }
      const invoiceName = 'invoice-' + orderId + '.pdf';
      const invoicePath = path.join('data', 'invoices', invoiceName);
      fs.readFile(invoicePath, (err, data) => {
        if (err) {
          return next(err);
        }
        res.setHeader('Content-Type', 'application/pdf');
        res.setHeader(
          'Content-Disposition',
          'inline; filename="' + invoiceName + '"'
        );
        res.send(data);
      });
    })
    .catch(err => next(err));
};

```

* Chapter 326: Streaming Data Vs Preloading Data

1. update
- ./controllers/shop.js





- this is streaned data created with that 'createReadStream()' which is the recommanded way of getting your file data for bigger files.

```
1 //./controllers/shop.js
```

```
2
3  const fs = require('fs');
4  const path = require('path');
5
6  const Product = require('../models/product');
7  const Order = require('../models/order');
8
9  exports.getProducts = (req, res, next) => {
10   Product.find()
11     .then(products => {
12       console.log(products);
13       res.render('shop/product-list', {
14         prods: products,
15         pageTitle: 'All Products',
16         path: '/products'
17       });
18     })
19     .catch(err => {
20       const error = new Error(err);
21       error.httpStatusCode = 500;
22       return next(error);
23     });
24 };
25
26 exports.getProduct = (req, res, next) => {
27   const prodId = req.params.productId;
28   Product.findById(prodId)
29     .then(product => {
30       res.render('shop/product-detail', {
31         product: product,
32         pageTitle: product.title,
33         path: '/products'
34       });
35     })
36     .catch(err => {
37       const error = new Error(err);
38       error.httpStatusCode = 500;
39       return next(error);
40     });
41 };
42
43 exports.getIndex = (req, res, next) => {
44   Product.find()
45     .then(products => {
46       res.render('shop/index', {
47         prods: products,
48         pageTitle: 'Shop',
49         path: '/'
50       });
51     })
52     .catch(err => {
53       const error = new Error(err);
54       error.httpStatusCode = 500;
55       return next(error);
56     });
57 };
```

```
58
59 exports.getCart = (req, res, next) => {
60   req.user
61     .populate('cart.items.productId')
62     .execPopulate()
63     .then(user => {
64       const products = user.cart.items;
65       res.render('shop/cart', {
66         path: '/cart',
67         pageTitle: 'Your Cart',
68         products: products
69       });
70     })
71     .catch(err => {
72       const error = new Error(err);
73       error.httpStatusCode = 500;
74       return next(error);
75     });
76 };
77
78 exports.postCart = (req, res, next) => {
79   const prodId = req.body.productId;
80   Product.findById(prodId)
81     .then(product => {
82       return req.user.addToCart(product);
83     })
84     .then(result => {
85       console.log(result);
86       res.redirect('/cart');
87     })
88     .catch(err => {
89       const error = new Error(err);
90       error.httpStatusCode = 500;
91       return next(error);
92     });
93 };
94
95 exports.postCartDeleteProduct = (req, res, next) => {
96   const prodId = req.body.productId;
97   req.user
98     .removeFromCart(prodId)
99     .then(result => {
100      res.redirect('/cart');
101    })
102    .catch(err => {
103      const error = new Error(err);
104      error.httpStatusCode = 500;
105      return next(error);
106    });
107 };
108
109 exports.postOrder = (req, res, next) => {
110   req.user
111     .populate('cart.items.productId')
112     .execPopulate()
113     .then(user => {
```

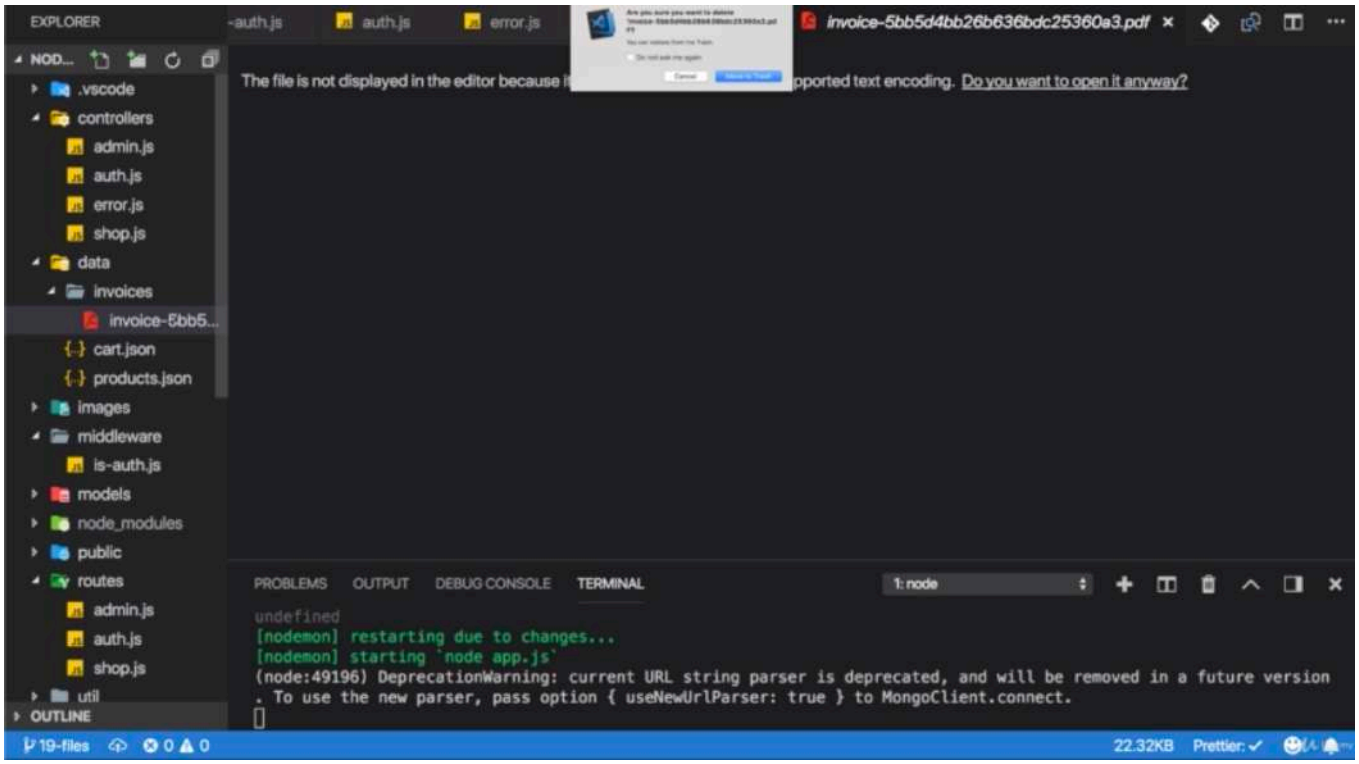
```
      const products = user.cart.items.map(i => {
        return { quantity: i.quantity, product: { ...i.productId._doc } };
      });
      const order = new Order({
        user: {
          email: req.user.email,
          userId: req.user
        },
        products: products
      });
      return order.save();
    })
    .then(result => {
      return req.user.clearCart();
    })
    .then(() => {
      res.redirect('/orders');
    })
    .catch(err => {
      const error = new Error(err);
      error.httpStatusCode = 500;
      return next(error);
    });
};

exports.getOrders = (req, res, next) => {
  Order.find({ 'user.userId': req.user._id })
    .then(orders => {
      res.render('shop/orders', {
        path: '/orders',
        pageTitle: 'Your Orders',
        orders: orders
      });
    })
    .catch(err => {
      const error = new Error(err);
      error.httpStatusCode = 500;
      return next(error);
    });
};

exports.getInvoice = (req, res, next) => {
  const orderId = req.params.orderId;
  Order.findById(orderId)
    .then(order => {
      if (!order) {
        return next(new Error('No order found.'));
      }
      if (order.user.userId.toString() !== req.user._id.toString()) {
        return next(new Error('Unauthorized'));
      }
      const invoiceName = 'invoice-' + orderId + '.pdf';
      const invoicePath = path.join('data', 'invoices', invoiceName);
      /**if you read a file like this,
       * node will fisrt of all access that file,
       * read the entire content into memory
```

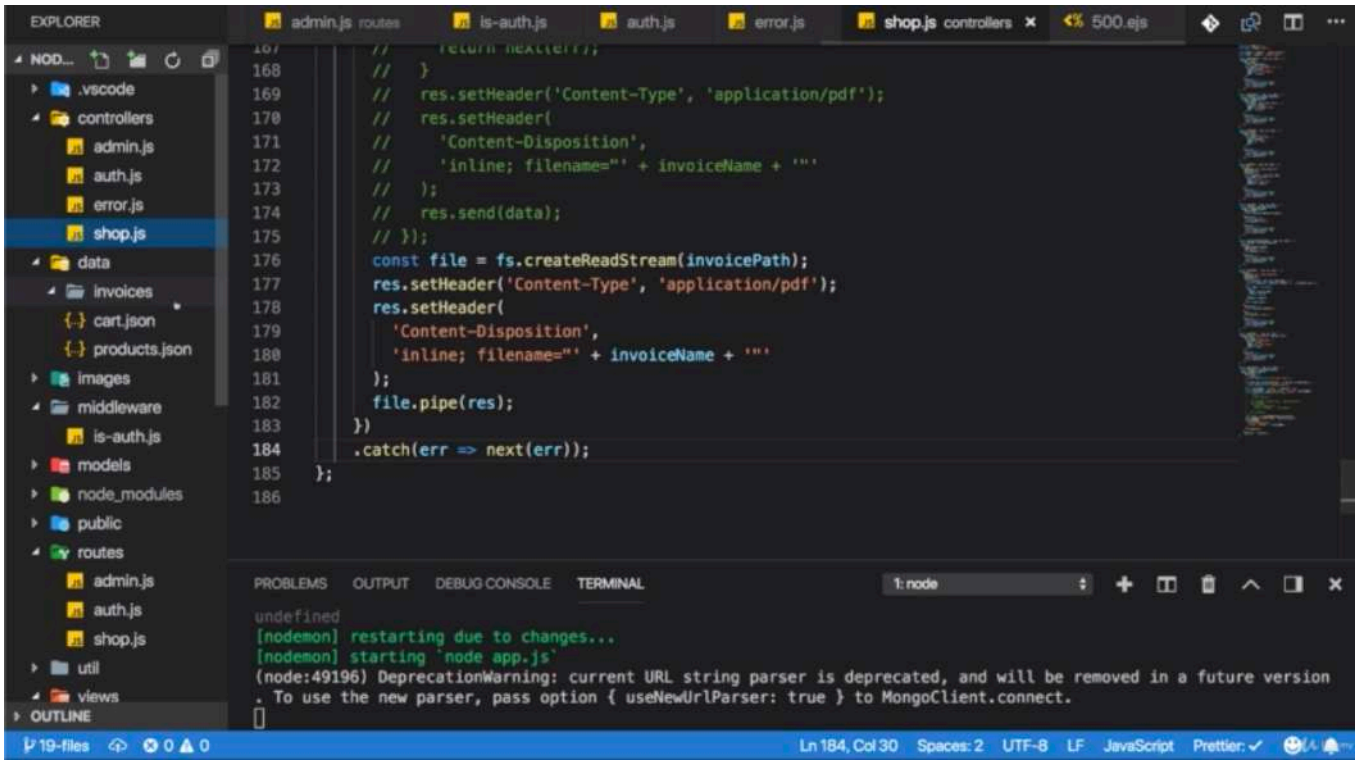
```
      * and then return it with the response.
      *
      * this means that for bigger files,
      * this will take very long before a response is sent
      * and your memory on the server might overflow at some point for many incoming
requests
      * because it has to read all the data into memory which is limited.
      *
      * instead you should be streaming your response data
      */

     //fs.readFile(invoicePath, (err, data) => {
     //  if (err) {
     //    return next(err);
     //  }
     //  res.setHeader('Content-Type', 'application/pdf');
     //  res.setHeader(
     //    'Content-Disposition',
     //    'inline; filename="' + invoiceName + '"'
     //  );
     //  res.send(data);
     //});

     const file = fs.createReadStream(invoicePath)
       res.setHeader('Content-Type', 'application/pdf');
       res.setHeader(
         'Content-Disposition',
         'inline; filename="' + invoiceName + '"'
       )
       /**and call the 'pipe()' method to forward the data
        * that is read in with that stream to my response
        * because the response object is a writable stream.
        * and you can use readable streams to pipe their output into a writable stream.
        *
        * we can pipe our readable stream
        * the file stream into the response
        * and that means that the response will be streamed to the browser
        * and will contain the data
        * and the data will be downloaded by the browser step by step.
        *
        * this is huge advantage for big file
        * because node never has to pre-load all the data into memory
        * but just streams it to the the client on the fly.
        * and the most it has to store is one chunk of data.
        *
        * the chunks are what we work with
        * the buffers gives us access to these chunks.
        *
        * we don't wait for all the chunks to come together
        * and concatenate them into one object,
        * instead we forward them to the browser
        * which is able to concatenate the incoming data pieces into the final file.
        */
       file.pipe(res)
   })
   .catch(err => next(err));
```

```
225 };
226
```

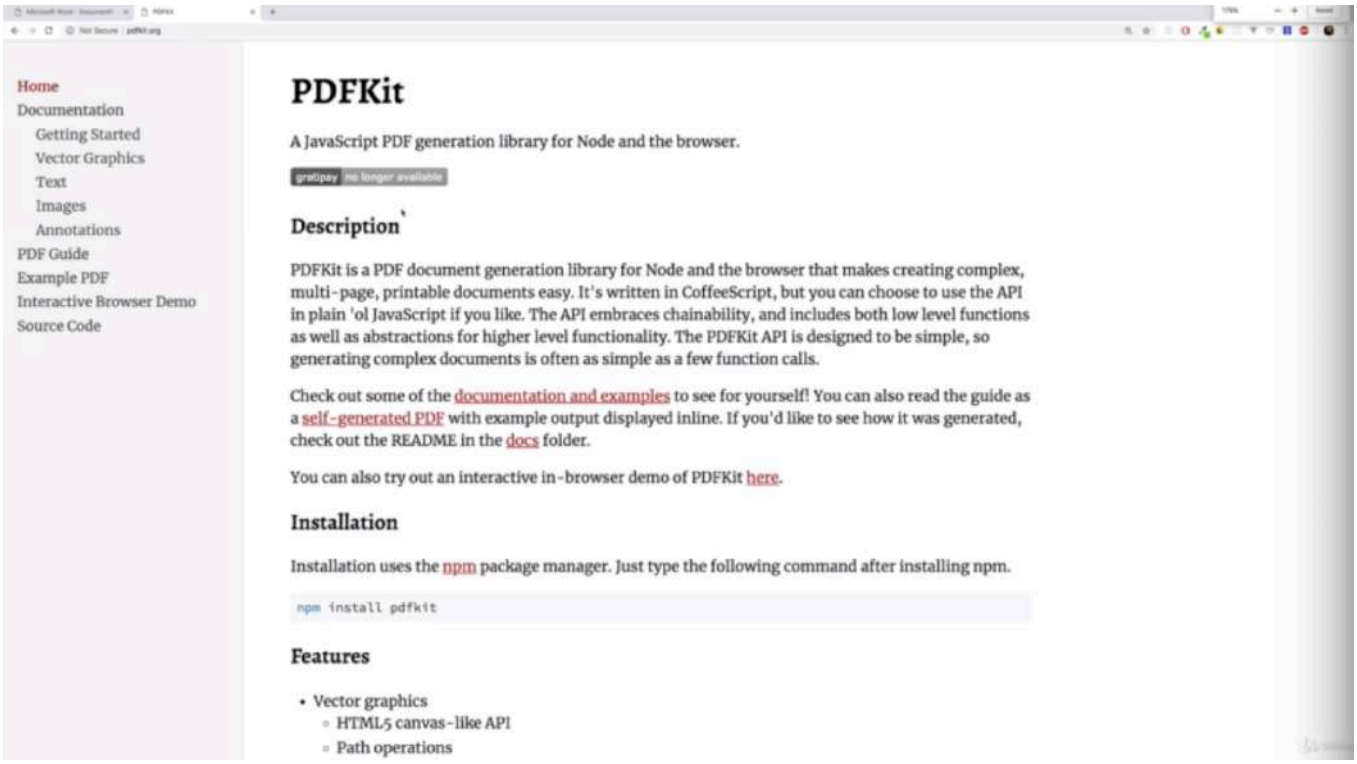
* Chapter 327: Using PDFKit For .pdf Generation

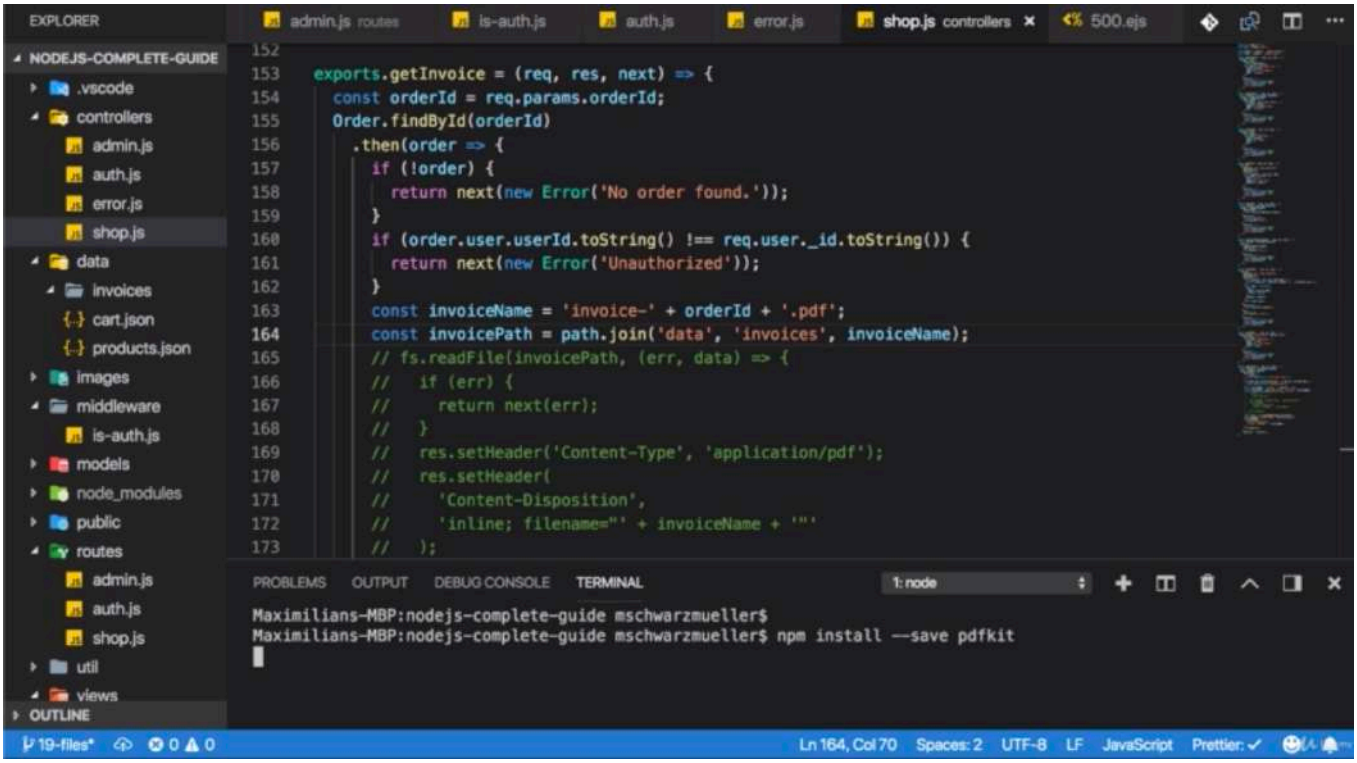
1. update
- ./controllers/shop.js

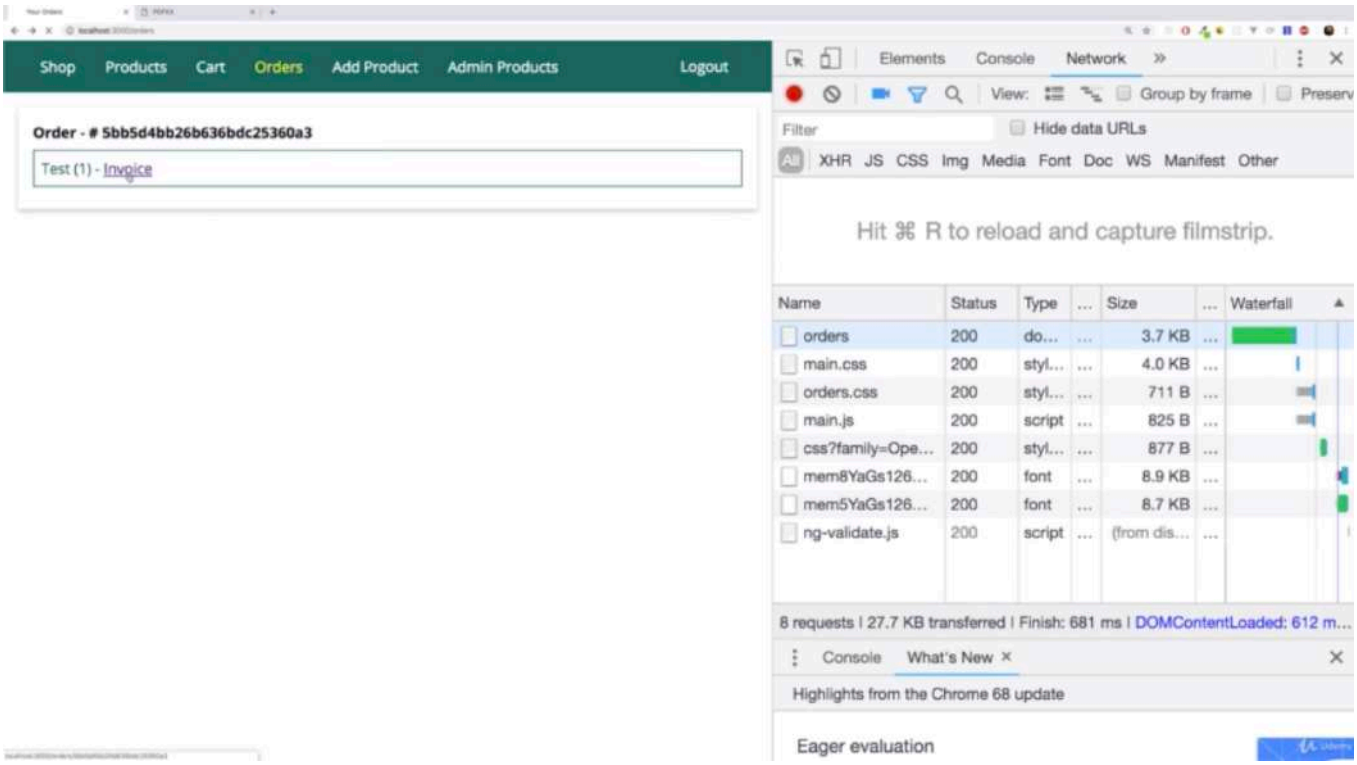


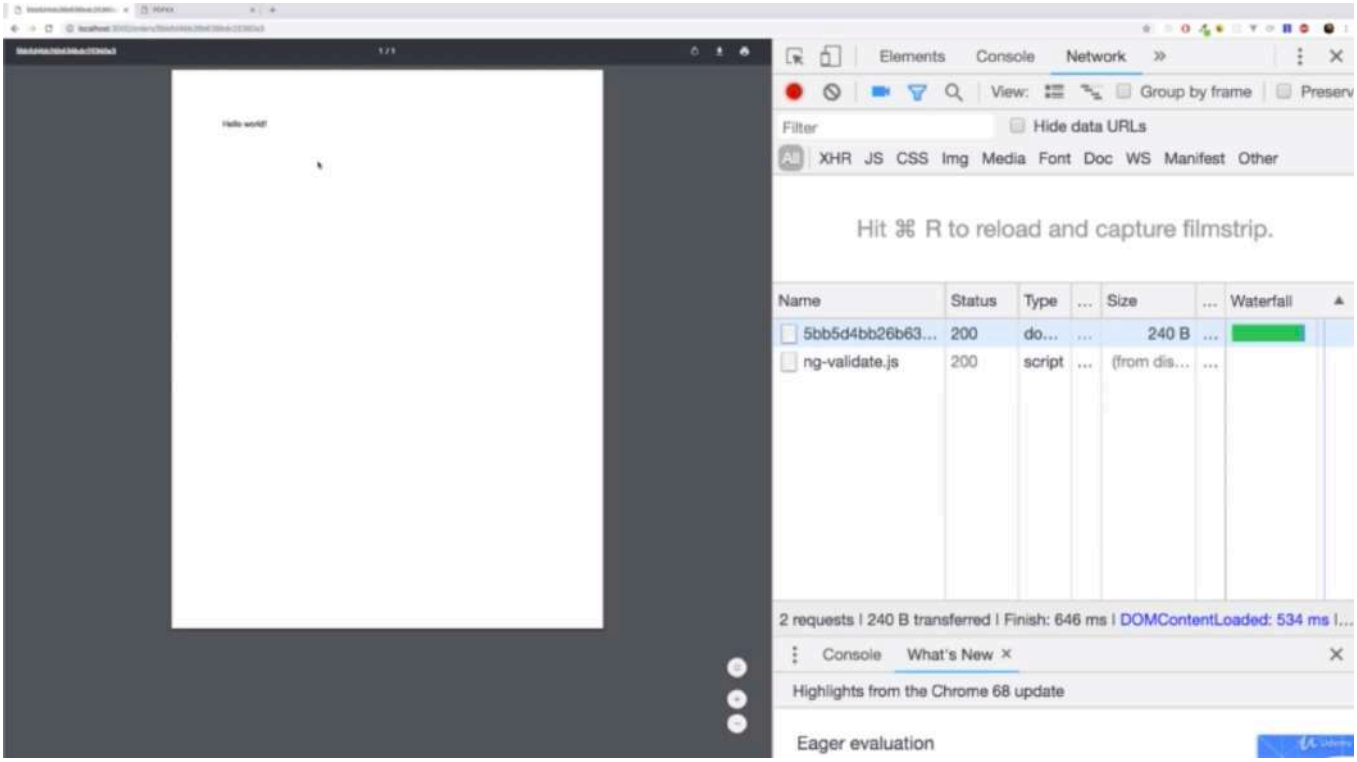


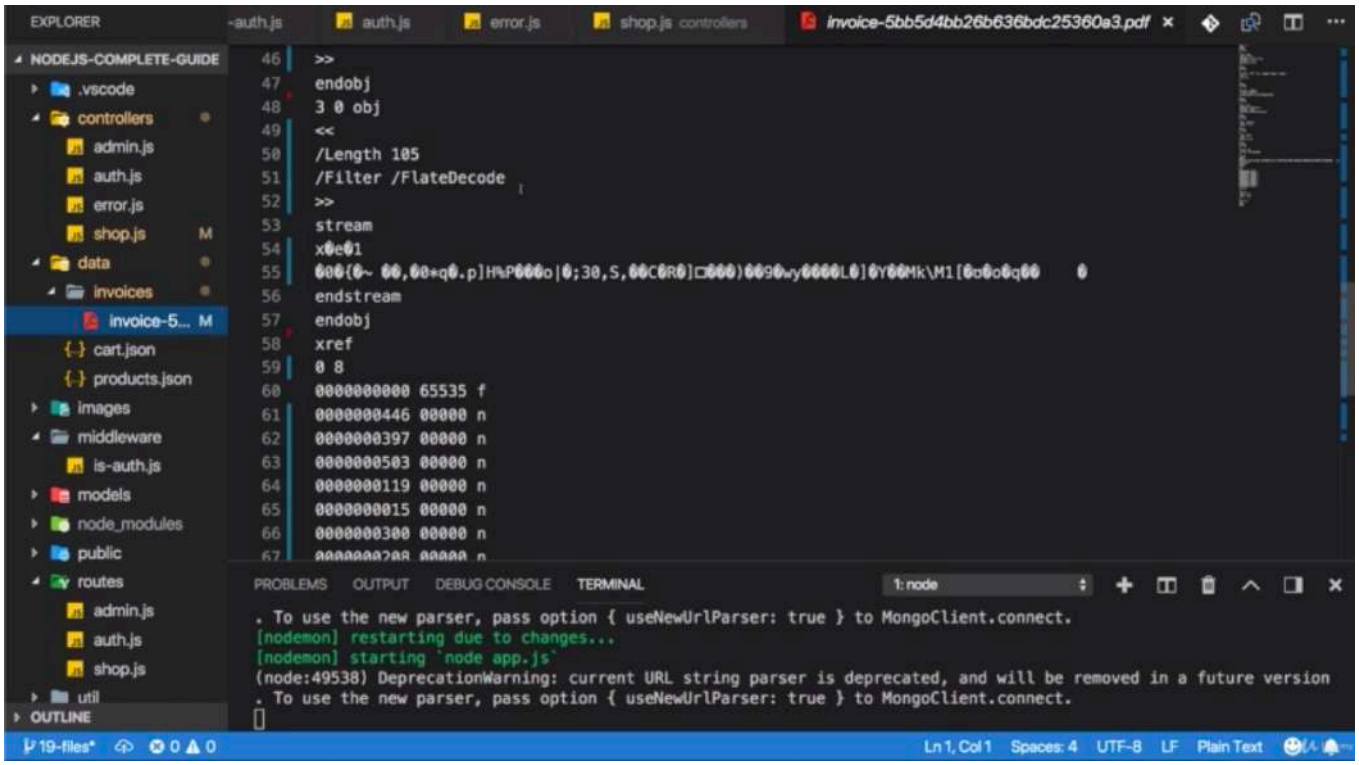
- delete my invoice.pdf file in ./invoice folder. but instead when we get the invoice for an order, i don't wanna

serve a file that already exists. but i wanna generate that file based on the real order data.


- 'PDFKit' is the 3rd party package which we can use and is very prominet or popular package for creating pdfs on a node.js server.



Shop
Products
Cart
Orders
Add Product
Admin Products
Logout
Order - # 5bb5d4bb26b636bdc25360a3
Test (1) - Invoice
Elements
Console
Network
View:
Group by frame
Filter
Hide data URLs
All
XHR JS CSS Img Media Font Doc WS Manifest Other
Hit ⌘ R to reload and capture filmstrip.
Name
Status
Type
Size
Waterfall
orders 200 do... 3.7 KB
main.css 200 styl... 4.0 KB
orders.css 200 styl... 711 B
main.js 200 script 825 B
css?family=Ope... 200 styl... 877 B
mem8YaGs126... 200 font 8.9 KB
mem5YaGs126... 200 font 8.7 KB
ng-validate.js 200 script (from dis...
8 requests | 27.7 KB transferred | Finish: 681 ms | DOMContentLoaded: 612 m...
Console
What's New
Highlights from the Chrome 68 update
Eager evaluation


Hello world!
Elements
Console
Network
View:
Group by frame
Filter
Hide data URLs
All
XHR JS CSS Img Media Font Doc WS Manifest Other
Hit ⌘ R to reload and capture filmstrip.
Name
Status
Type
Size
Waterfall
5bb5d4bb26b63... 200 do... 240 B
ng-validate.js 200 script (from dis...
2 requests | 240 B transferred | Finish: 646 ms | DOMContentLoaded: 534 ms |...
Console
What's New
Highlights from the Chrome 68 update
Eager evaluation




```
1  //./controllers/shop.js
2
3  const fs = require('fs');
4  const path = require('path');
5
6  const PDFDocument= require('pdfkit')
7
8  const Product = require('../models/product');
9  const Order = require('../models/order');
10
11 exports.getProducts = (req, res, next) => {
12   Product.find()
13     .then(products => {
14       console.log(products);
15       res.render('shop/product-list', {
16         prods: products,
17         pageTitle: 'All Products',
18         path: '/products'
19       });
20     })
21     .catch(err => {
22       const error = new Error(err);
23       error.httpStatusCode = 500;
24       return next(error);
25     });
26 };
27
28 exports.getProduct = (req, res, next) => {
29   const prodId = req.params.productId;
30   Product.findById(prodId)
31     .then(product => {
32       res.render('shop/product-detail', {
33         product: product,
34         pageTitle: product.title,
35         path: '/products'
```

```
    });
  })
  .catch(err => {
    const error = new Error(err);
    error.httpStatusCode = 500;
    return next(error);
  });
};

exports.getIndex = (req, res, next) => {
  Product.find()
  .then(products => {
    res.render('shop/index', {
      prods: products,
      pageTitle: 'Shop',
      path: '/'
    });
  })
  .catch(err => {
    const error = new Error(err);
    error.httpStatusCode = 500;
    return next(error);
  });
};

exports.getCart = (req, res, next) => {
  req.user
  .populate('cart.items.productId')
  .execPopulate()
  .then(user => {
    const products = user.cart.items;
    res.render('shop/cart', {
      path: '/cart',
      pageTitle: 'Your Cart',
      products: products
    });
  })
  .catch(err => {
    const error = new Error(err);
    error.httpStatusCode = 500;
    return next(error);
  });
};

exports.postCart = (req, res, next) => {
  const prodId = req.body.productId;
  Product.findById(prodId)
  .then(product => {
    return req.user.addToCart(product);
  })
  .then(result => {
    console.log(result);
    res.redirect('/cart');
  })
  .catch(err => {
    const error = new Error(err);
```

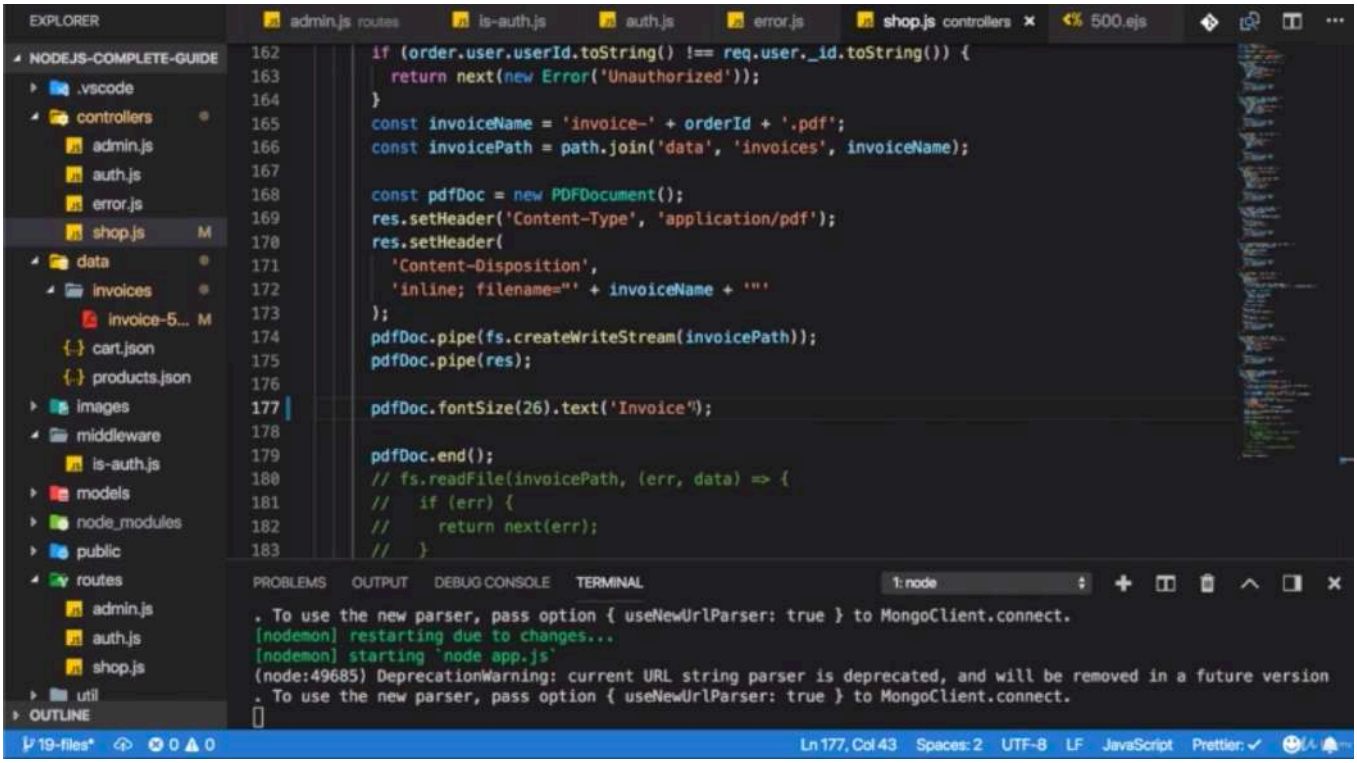
```
      error.httpStatusCode = 500;
      return next(error);
    });
};

exports.postCartDeleteProduct = (req, res, next) => {
  const prodId = req.body.productId;
  req.user
    .removeFromCart(prodId)
    .then(result => {
      res.redirect('/cart');
    })
    .catch(err => {
      const error = new Error(err);
      error.httpStatusCode = 500;
      return next(error);
    });
};

exports.postOrder = (req, res, next) => {
  req.user
    .populate('cart.items.productId')
    .execPopulate()
    .then(user => {
      const products = user.cart.items.map(i => {
        return { quantity: i.quantity, product: { ...i.productId._doc } };
      });
      const order = new Order({
        user: {
          email: req.user.email,
          userId: req.user
        },
        products: products
      });
      return order.save();
    })
    .then(result => {
      return req.user.clearCart();
    })
    .then(() => {
      res.redirect('/orders');
    })
    .catch(err => {
      const error = new Error(err);
      error.httpStatusCode = 500;
      return next(error);
    });
};

exports.getOrders = (req, res, next) => {
  Order.find({ 'user.userId': req.user._id })
    .then(orders => {
      res.render('shop/orders', {
        path: '/orders',
        pageTitle: 'Your Orders',
        orders: orders
```

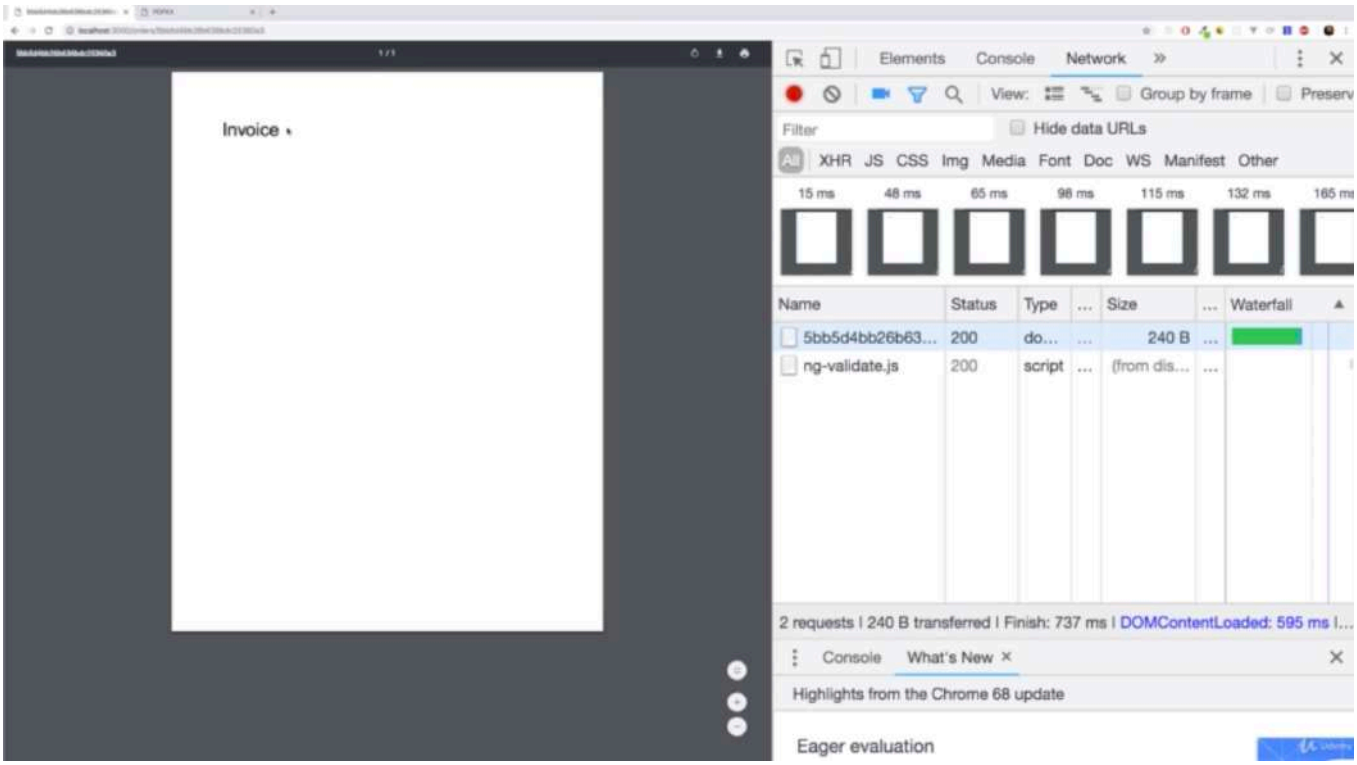
```
    });
  })
  .catch(err => {
    const error = new Error(err);
    error.httpStatusCode = 500;
    return next(error);
  });
};

exports.getInvoice = (req, res, next) => {
  const orderId = req.params.orderId;
  Order.findById(orderId)
    .then(order => {
      if (!order) {
        return next(new Error('No order found.'));
      }
      if (order.user.userId.toString() !== req.user._id.toString()) {
        return next(new Error('Unauthorized'));
      }
      const invoiceName = 'invoice-' + orderId + '.pdf';
      const invoicePath = path.join('data', 'invoices', invoiceName);

      const pdfDoc = new PDFDocument()
      res.setHeader('Content-Type', 'application/pdf');
      res.setHeader(
        'Content-Disposition',
        'inline; filename="' + invoiceName + '"'
      )
      /**this ensure that the pdf we generate also gets stored on the server
       * and not just serve to the client.
       *
       * and when you call 'end()',
       * these writable streams for creating the file and for sending the response
       * will be closed. so file will be saved
       * and the response will be sent.
       */
      pdfDoc.pipe(fs.createWriteStream(invoicePath))
      pdfDoc.pipe(res)

      /**'text()' allows us to ad a single line of text into the PDFDocumet
       * and you have to call pdfDoc to tell node
       * when you are done writing to that stream
       * because you have to be done at some point.
       */
      pdfDoc.text('Hello World!')

      pdfDoc.end()
      //fs.readFile(invoicePath, (err, data) => {
      //  if (err) {
      //    return next(err);
      //  }
      //  res.setHeader('Content-Type', 'application/pdf');
      //  res.setHeader(
      //    'Content-Disposition',
      //    'inline; filename="' + invoiceName + '"'
      //  );
```

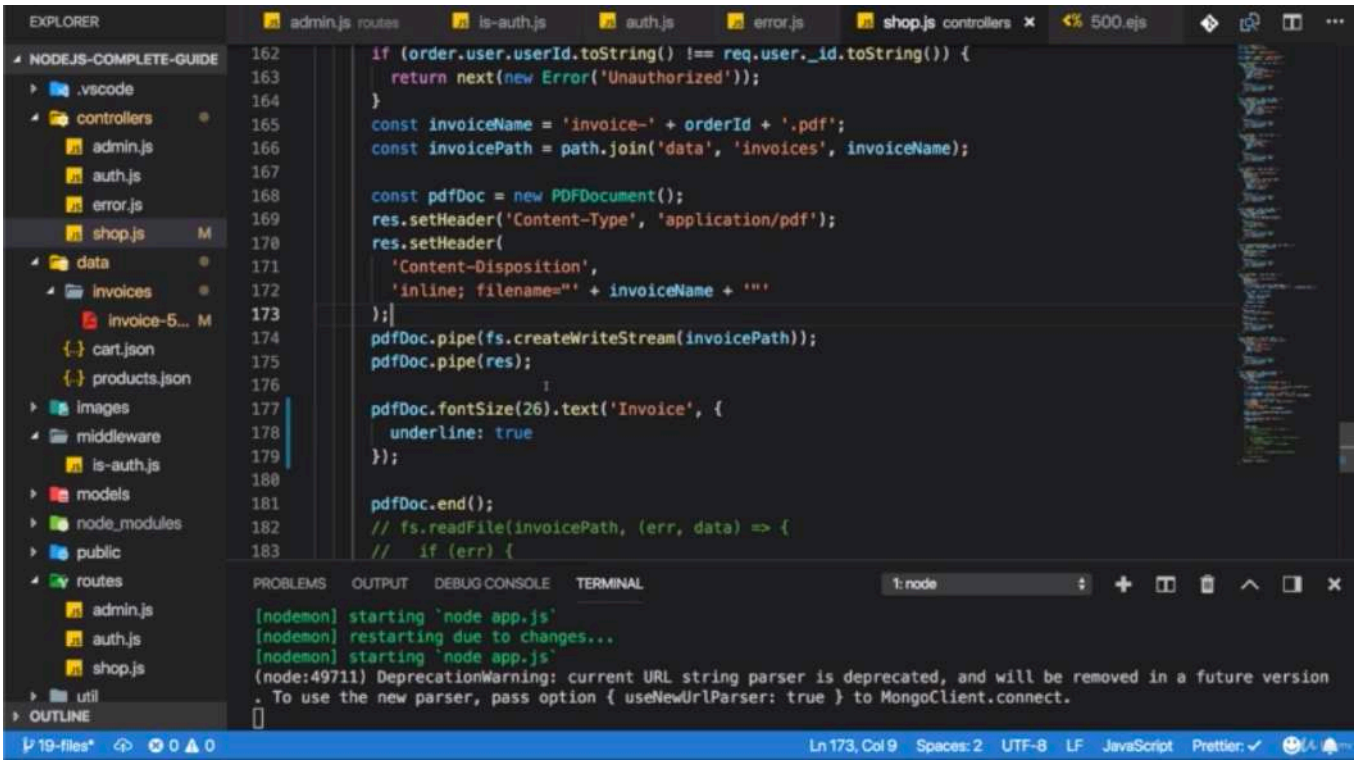
```
      //  res.send(data);
      //});

      //const file = fs.createReadStream(invoicePath)
      //file.pipe(res)
    })
    .catch(err => next(err));
};

```

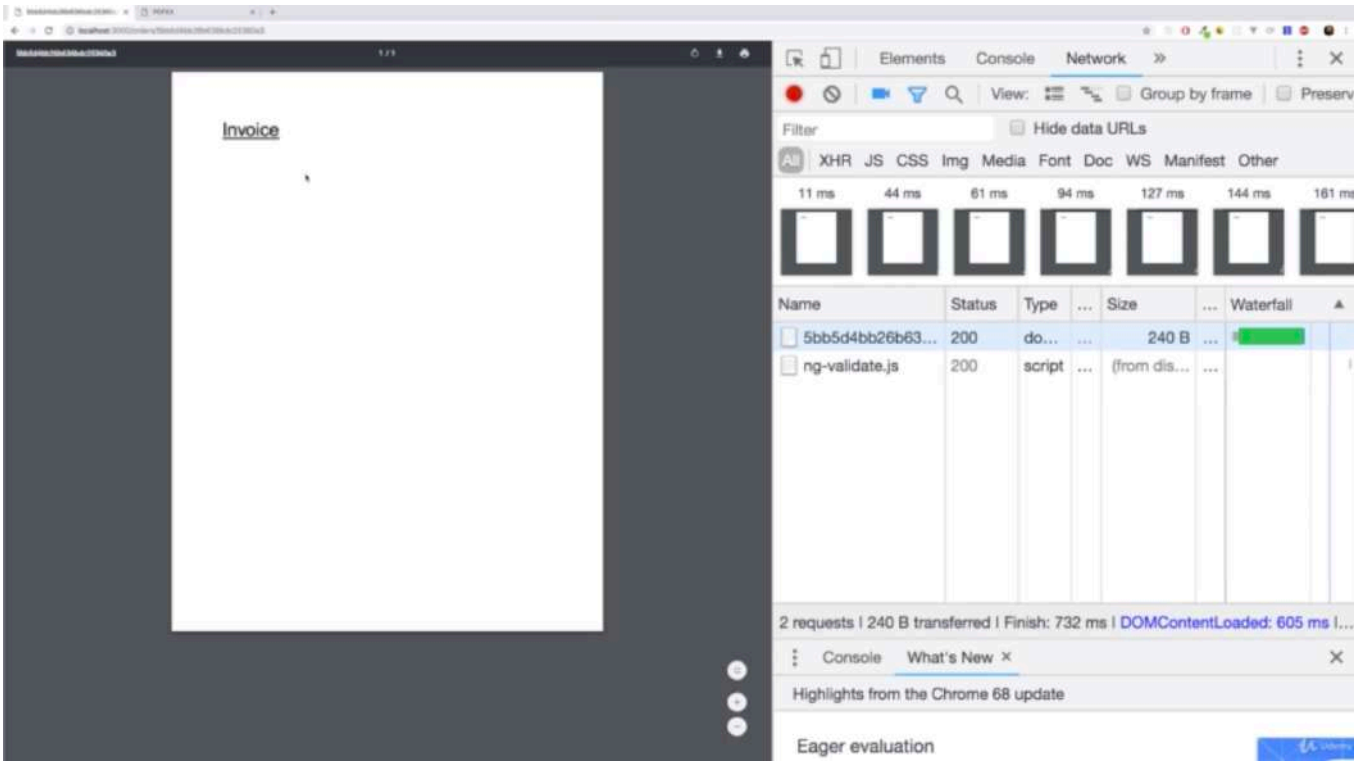
* Chapter 328: Generating .pdf Files With Order Data

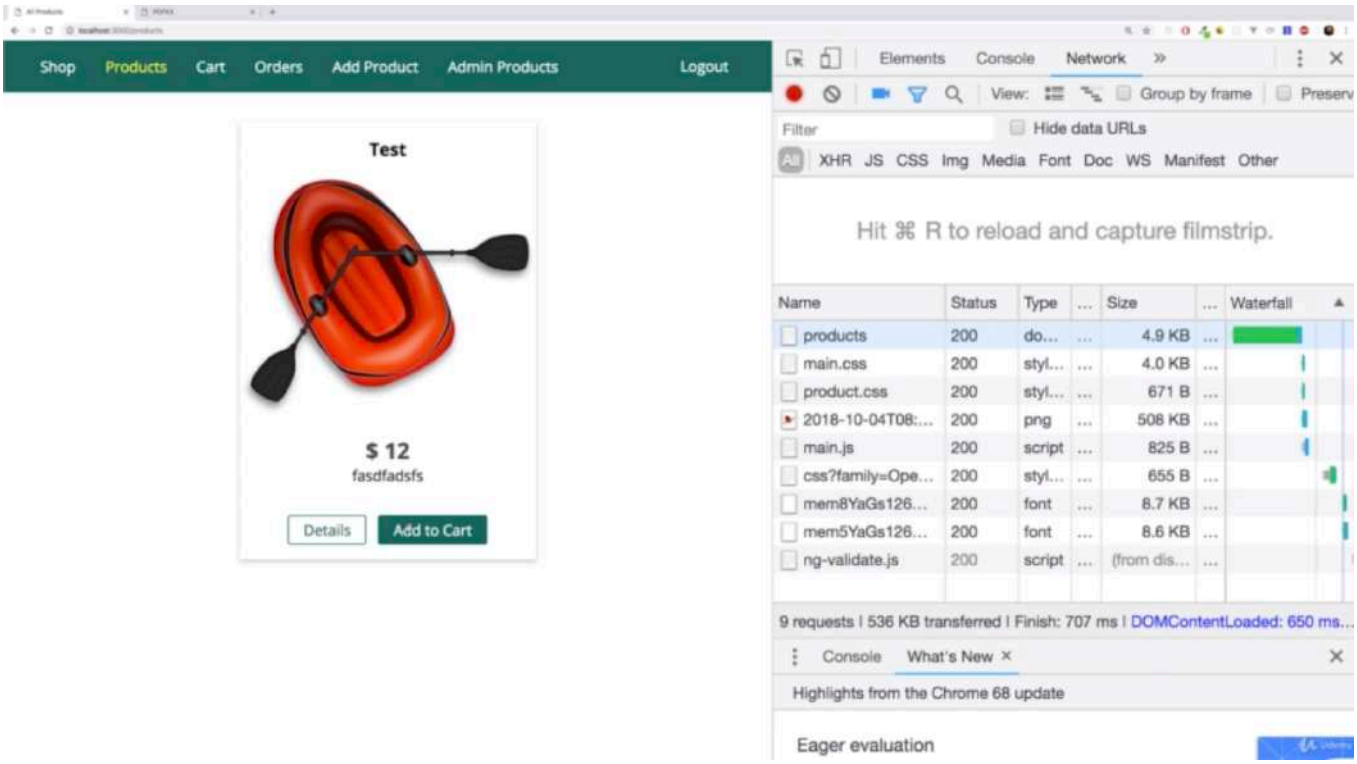
1. update
- ./controllers/shop.js

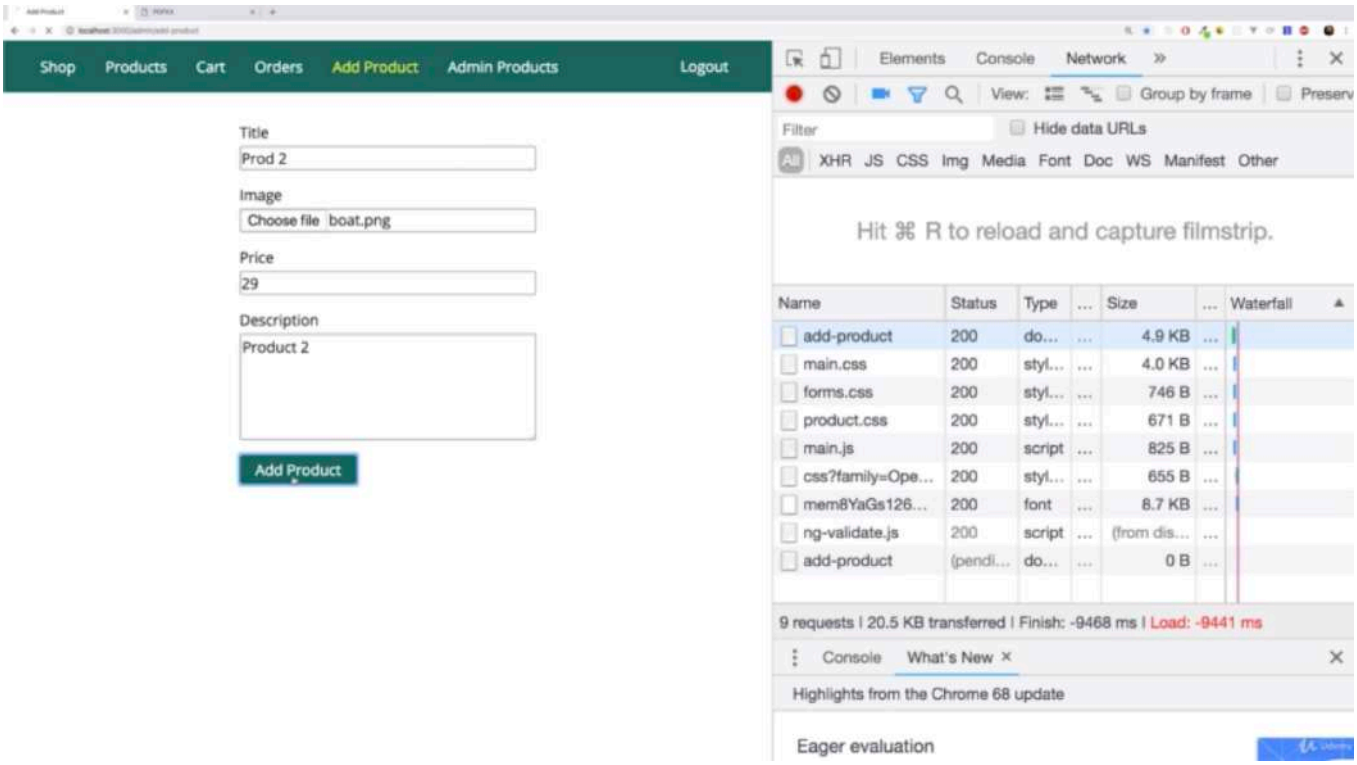


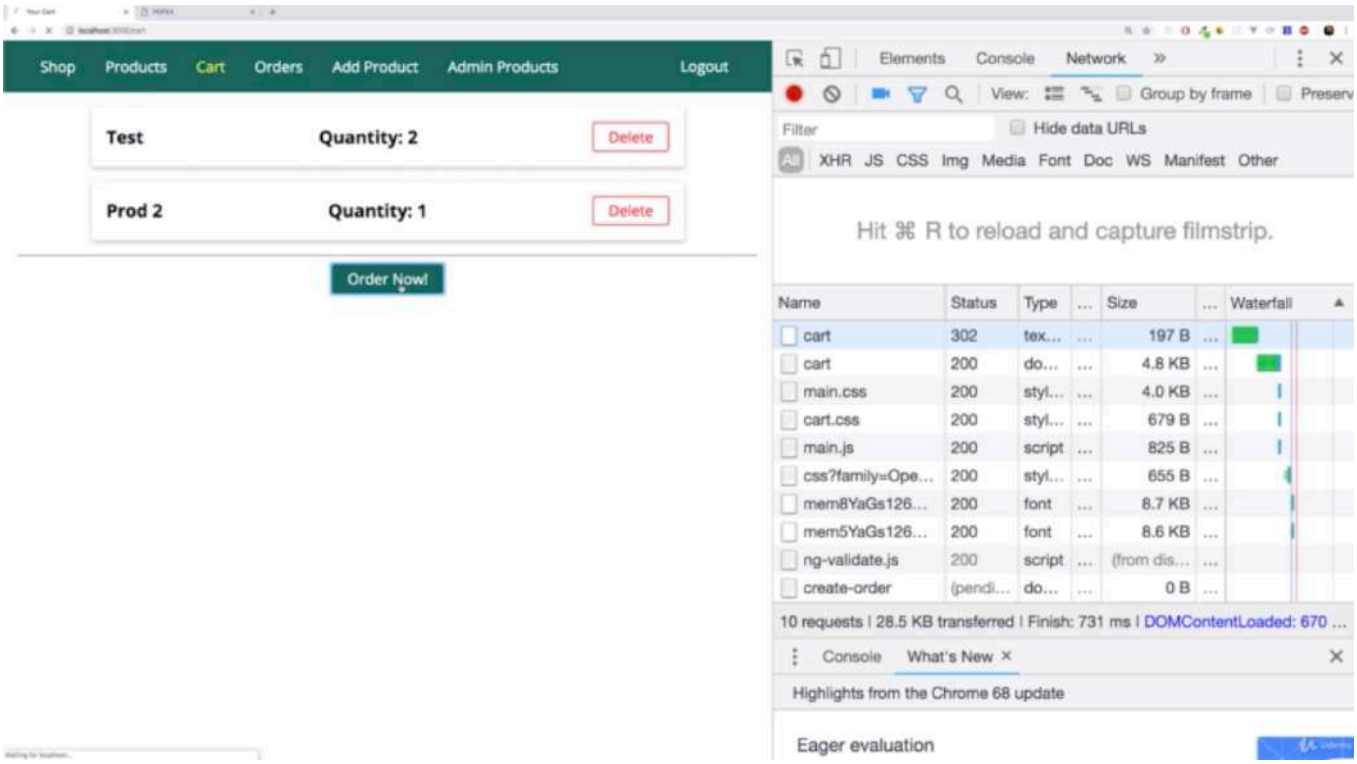

Invoice
Elements
Console
Network
View:
Group by frame
Filter
Hide data URLs
All
XHR
JS
CSS
Img
Media
Font
Doc
WS
Manifest
Other
15 ms
48 ms
65 ms
98 ms
115 ms
132 ms
Name
Status
Type
Size
Waterfall
5bb5d4bb26b63...
200
do...
240 B
ng-validate.js
200
script
(from dis...
2 requests | 240 B transferred | Finish: 737 ms | DOMContentLoaded: 595 ms
Console
What's New
Highlights from the Chrome 68 update
Eager evaluation

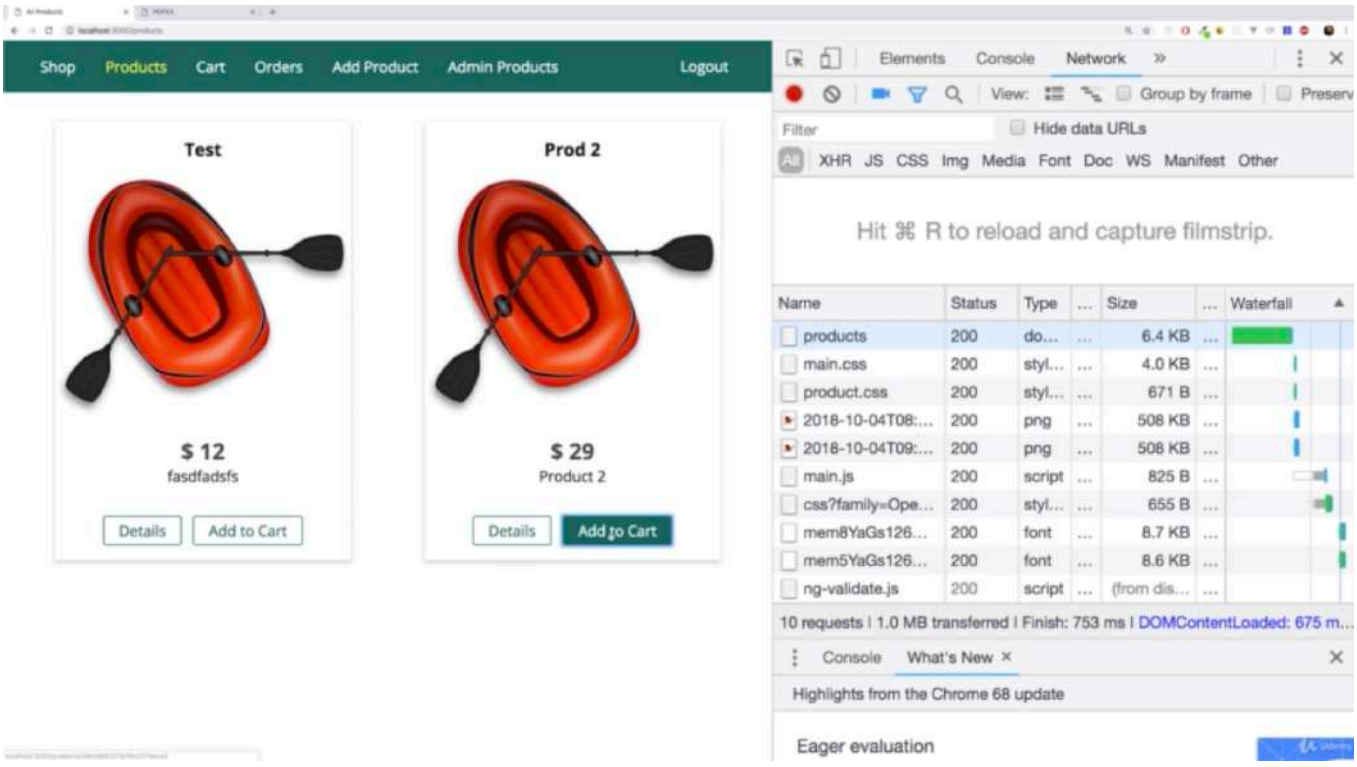

EXPLORER
admin.js routes
is-auth.js
auth.js
error.js
shop.js controllers
500.ejs
NODEJS-COMPLETE-GUIDE
.vscode
controllers
admin.js
auth.js
error.js
shop.js
data
invoices
invoice-5...
cart.json
products.json
images
middleware
is-auth.js
models
node_modules
public
routes
admin.js
auth.js
shop.js
util
OUTLINE
if (order.user.userId.toString() !== req.user._id.toString()) {
return next(new Error('Unauthorized'));
}
const invoiceName = 'invoice-' + orderId + '.pdf';
const invoicePath = path.join('data', 'invoices', invoiceName);
const pdfDoc = new PDFDocument();
res.setHeader('Content-Type', 'application/pdf');
res.setHeader(
'Content-Disposition',
'inline; filename="' + invoiceName + '"'
);
pdfDoc.pipe(fs.createWriteStream(invoicePath));
pdfDoc.pipe(res);
pdfDoc.fontSize(26).text('Invoice', {
underline: true
});
pdfDoc.end();
// fs.readFile(invoicePath, (err, data) => {
// if (err) {
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
1: node
[nodemon] starting `node app.js`
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
(node:49711) 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.
19-files*
Ln 173, Col 9
Spaces: 2
UTF-8
LF
JavaScript
Prettier:

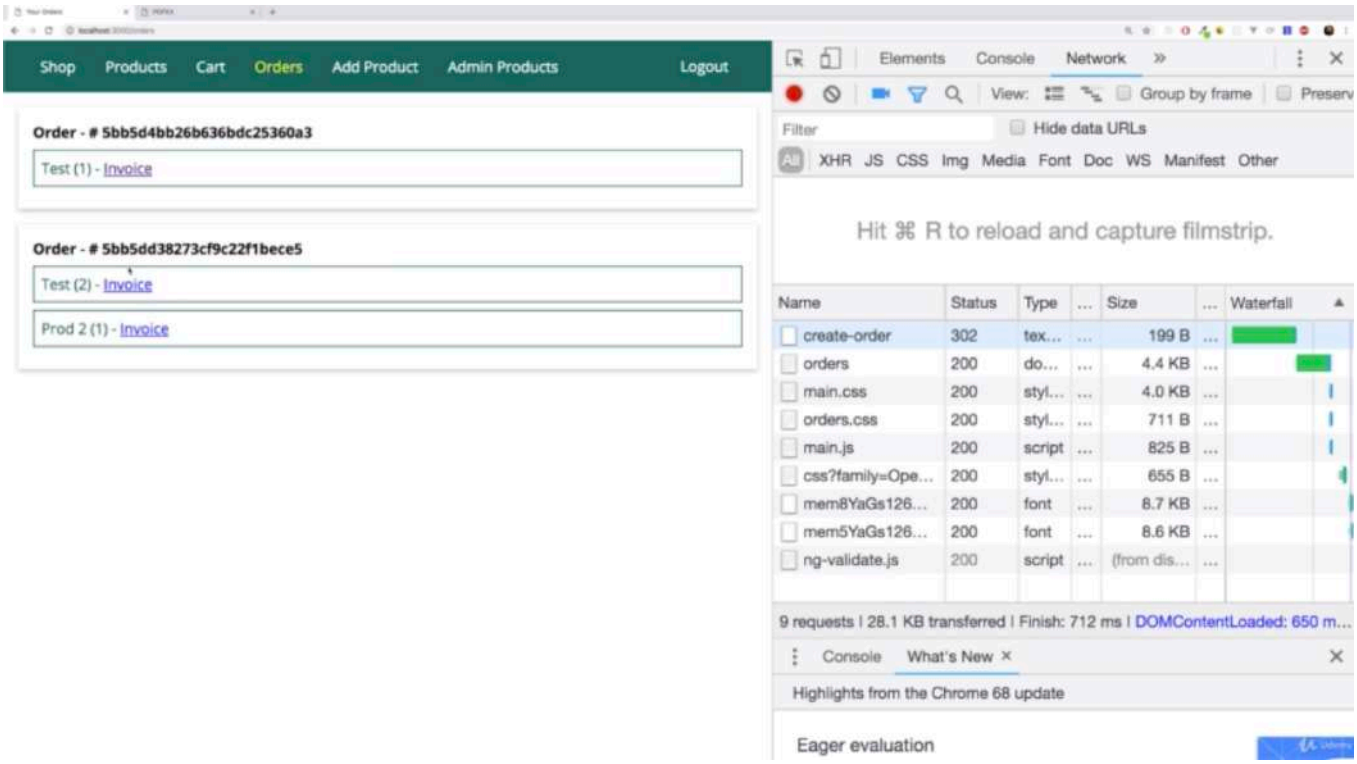

Invoice
Elements
Console
Network
View:
Group by frame
Filter
Hide data URLs
All
XHR
JS
CSS
Img
Media
Font
Doc
WS
Manifest
Other
11 ms
44 ms
61 ms
94 ms
127 ms
144 ms
Name
Status
Type
Size
Waterfall
5bb5d4bb26b63...
200
do...
240 B
ng-validate.js
200
script
(from dis...
2 requests | 240 B transferred | Finish: 732 ms | DOMContentLoaded: 605 ms
Console
What's New
Highlights from the Chrome 68 update
Eager evaluation

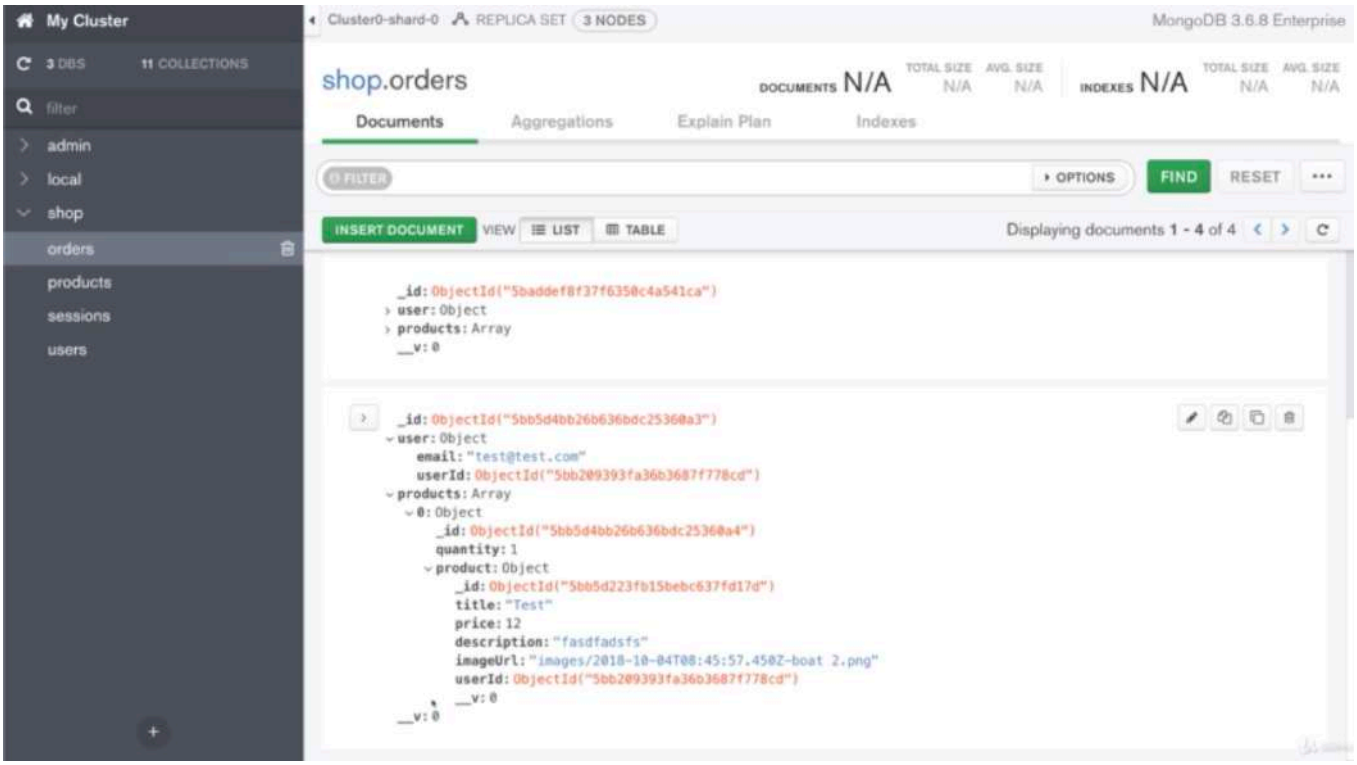

Shop
Products
Cart
Orders
Add Product
Admin Products
Logout
Test
$ 12
fasdfadsfs
Details
Add to Cart
Elements
Console
Network
View:
Group by frame
Filter
Hide data URLs
All
XHR
JS
CSS
Img
Media
Font
Doc
WS
Manifest
Other
Hit ⌘ R to reload and capture filmstrip.
Name
Status
Type
Size
Waterfall
products
200
do...
4.9 KB
main.css
200
styl...
4.0 KB
product.css
200
styl...
671 B
2018-10-04T08:...
200
png
508 KB
main.js
200
script
825 B
css?family=Ope...
200
styl...
655 B
mem8YaGs126...
200
font
8.7 KB
mem5YaGs126...
200
font
8.6 KB
ng-validate.js
200
script
(from dis...
9 requests | 536 KB transferred | Finish: 707 ms | DOMContentLoaded: 650 ms
Console
What's New
Highlights from the Chrome 68 update
Eager evaluation

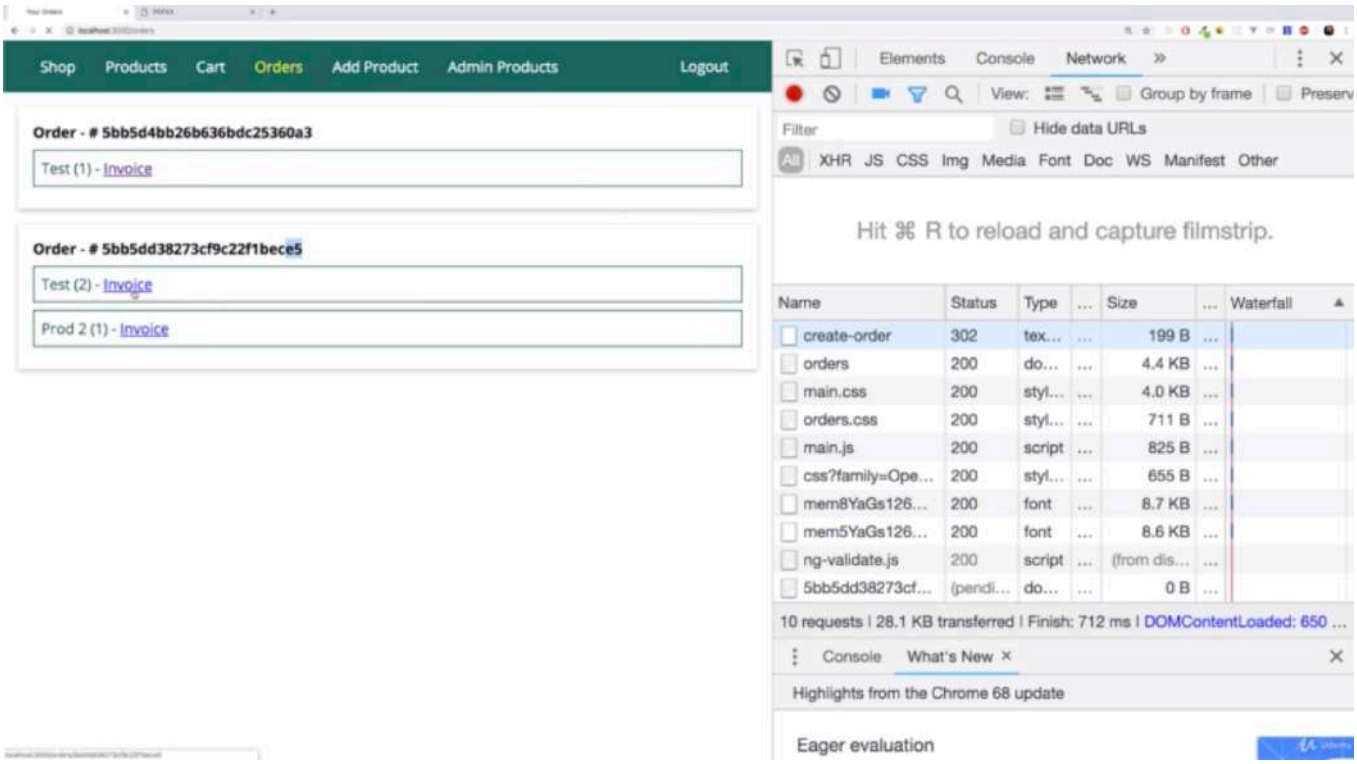

Shop
Products
Cart
Orders
Add Product
Admin Products
Logout
Title
Prod 2
Image
Choose file boat.png
Price
29
Description
Product 2
Add Product
Elements
Console
Network
Hit ⌘ R to reload and capture filmstrip.
Name
Status
Type
Size
Waterfall
add-product 200 do... 4.9 KB
main.css 200 styl... 4.0 KB
forms.css 200 styl... 746 B
product.css 200 styl... 671 B
main.js 200 script 825 B
css?family=Ope... 200 styl... 655 B
mem8YaGs126... 200 font 8.7 KB
ng-validate.js 200 script (from dis...
add-product (pendi... do... 0 B
9 requests | 20.5 KB transferred | Finish: -9468 ms | Load: -9441 ms
Console
What's New
Highlights from the Chrome 68 update
Eager evaluation

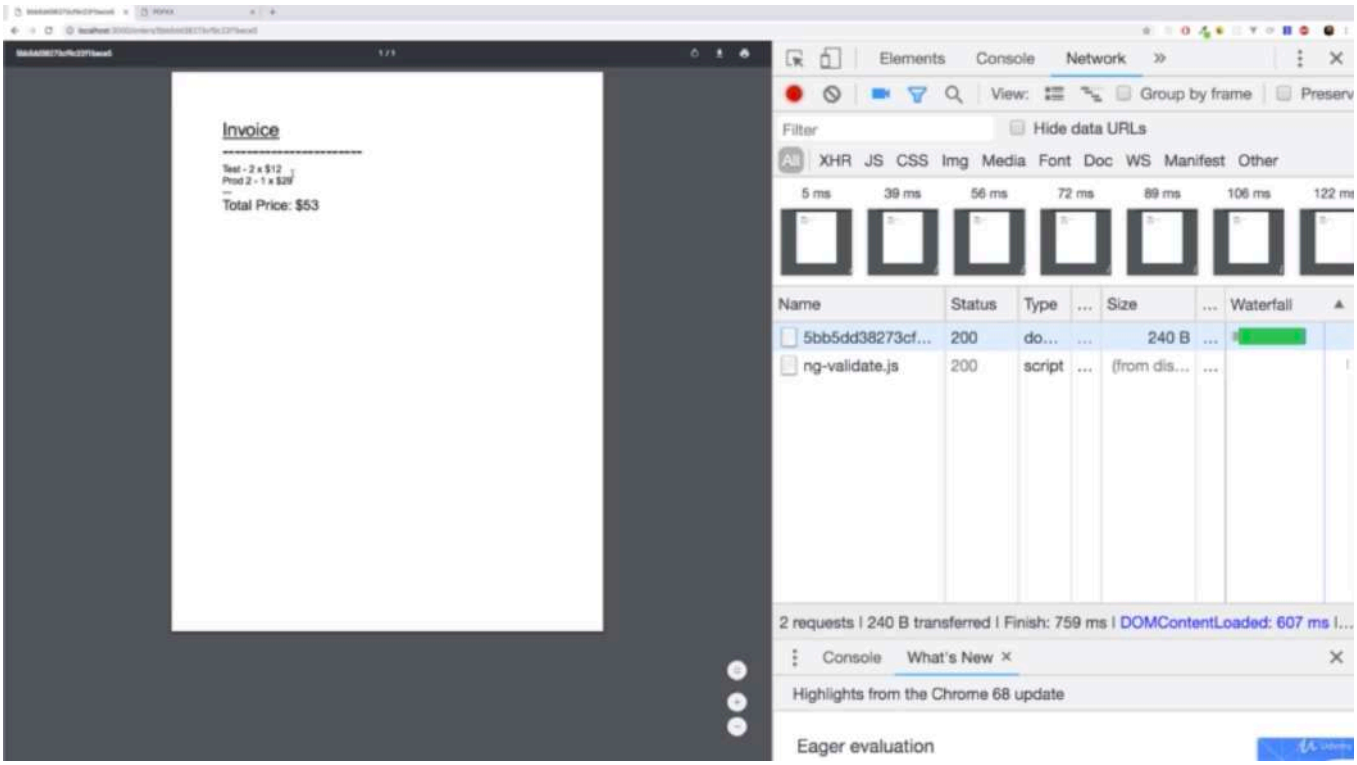

Shop
Products
Cart
Orders
Add Product
Admin Products
Logout
Test
Quantity: 2
Delete
Prod 2
Quantity: 1
Delete
Order Now!
Elements
Console
Network
Hit ⌘ R to reload and capture filmstrip.
Name
Status
Type
Size
Waterfall
cart 302 tex... 197 B
cart 200 do... 4.8 KB
main.css 200 styl... 4.0 KB
cart.css 200 styl... 679 B
main.js 200 script 825 B
css?family=Ope... 200 styl... 655 B
mem8YaGs126... 200 font 8.7 KB
mem5YaGs126... 200 font 8.6 KB
ng-validate.js 200 script (from dis...
create-order (pendi... do... 0 B
10 requests | 28.5 KB transferred | Finish: 731 ms | DOMContentLoaded: 670
Console
What's New
Highlights from the Chrome 68 update
Eager evaluation


Shop
Products
Cart
Orders
Add Product
Admin Products
Logout
Test
$ 12
fasdfadsfs
Details
Add to Cart
Prod 2
$ 29
Product 2
Details
Add to Cart
Elements
Console
Network
View:
Group by frame
Filter
Hide data URLs
All
XHR JS CSS Img Media Font Doc WS Manifest Other
Hit ⌘ R to reload and capture filmstrip.
Name
Status
Type
Size
Waterfall
products 200 do... 6.4 KB
main.css 200 styl... 4.0 KB
product.css 200 styl... 671 B
2018-10-04T08:... 200 png 508 KB
2018-10-04T09:... 200 png 508 KB
main.js 200 script 825 B
css?family=Ope... 200 styl... 655 B
mem8YaGs126... 200 font 8.7 KB
mem5YaGs126... 200 font 8.6 KB
ng-validate.js 200 script (from dis...
10 requests | 1.0 MB transferred | Finish: 753 ms | DOMContentLoaded: 675 m...
Console
What's New
Highlights from the Chrome 68 update
Eager evaluation


Shop
Products
Cart
Orders
Add Product
Admin Products
Logout
Order - # 5bb5d4bb26b636bdc25360a3
Test (1) - Invoice
Order - # 5bb5dd38273cf9c22f1bece5
Test (2) - Invoice
Prod 2 (1) - Invoice
Elements
Console
Network
View:
Group by frame
Filter
Hide data URLs
All
XHR JS CSS Img Media Font Doc WS Manifest Other
Hit ⌘ R to reload and capture filmstrip.
Name
Status
Type
Size
Waterfall
create-order 302 tex... 199 B
orders 200 do... 4.4 KB
main.css 200 styl... 4.0 KB
orders.css 200 styl... 711 B
main.js 200 script 825 B
css?family=Ope... 200 styl... 655 B
mem8YaGs126... 200 font 8.7 KB
mem5YaGs126... 200 font 8.6 KB
ng-validate.js 200 script (from dis...
9 requests | 28.1 KB transferred | Finish: 712 ms | DOMContentLoaded: 650 m...
Console
What's New
Highlights from the Chrome 68 update
Eager evaluation


My Cluster
3 DBS
11 COLLECTIONS
filter
admin
local
shop
orders
products
sessions
users
Cluster0-shard-0
REPLICA SET
3 NODES
MongoDB 3.6.8 Enterprise
shop.orders
DOCUMENTS N/A
INDEXES N/A
Documents
Aggregations
Explain Plan
Indexes
FILTER
OPTIONS
FIND
RESET
INSERT DOCUMENT
VIEW
LIST
TABLE
Displaying documents 1 - 4 of 4
_id: ObjectId("5baddef8f37f6350c4a541ca")
user: Object
products: Array
__v: 0
_id: ObjectId("5bb5d4bb26b636bdc25360a3")
user: Object
email: "test@test.com"
userId: ObjectId("5bb209393fa36b3687f778cd")
products: Array
0: Object
_id: ObjectId("5bb5d4bb26b636bdc25360a4")
quantity: 1
product: Object
_id: ObjectId("5bb5d223fb15bebc637fd17d")
title: "Test"
price: 12
description: "fasdfadsfs"
imageUrl: "images/2018-10-04T08:45:57.450Z-boat 2.png"
userId: ObjectId("5bb209393fa36b3687f778cd")
__v: 0
__v: 0


Shop
Products
Cart
Orders
Add Product
Admin Products
Logout
Order - # 5bb5d4bb26b636bdc25360a3
Test (1) - Invoice
Order - # 5bb5dd38273cf9c22f1bece5
Test (2) - Invoice
Prod 2 (1) - Invoice
Elements
Console
Network
View:
Group by frame
Filter
Hide data URLs
All
XHR
JS
CSS
Img
Media
Font
Doc
WS
Manifest
Other
Hit ⌘ R to reload and capture filmstrip.
Name
Status
Type
Size
Waterfall
create-order
orders
main.css
orders.css
main.js
css?family=Ope...
mem8YaGs126...
mem5YaGs126...
ng-validate.js
5bb5dd38273cf...
10 requests | 28.1 KB transferred | Finish: 712 ms | DOMContentLoaded: 650 ...
Console
What's New
Highlights from the Chrome 68 update
Eager evaluation




```
//./controllers/shop.js

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

const PDFDocument= require('pdfkit')

const Product = require('../models/product');
const Order = require('../models/order');

exports.getProducts = (req, res, next) => {
  Product.find()
    .then(products => {
      console.log(products);
      res.render('shop/product-list', {
        prods: products,
        pageTitle: 'All Products',
        path: '/products'
      });
    })
    .catch(err => {
      const error = new Error(err);
      error.httpStatusCode = 500;
      return next(error);
    });
};

exports.getProduct = (req, res, next) => {
  const prodId = req.params.productId;
  Product.findById(prodId)
    .then(product => {
      res.render('shop/product-detail', {
        product: product,
        pageTitle: product.title,
        path: '/products'
```

```
    });
  })
  .catch(err => {
    const error = new Error(err);
    error.httpStatusCode = 500;
    return next(error);
  });
};

exports.getIndex = (req, res, next) => {
  Product.find()
    .then(products => {
      res.render('shop/index', {
        prods: products,
        pageTitle: 'Shop',
        path: '/'
      });
    })
    .catch(err => {
      const error = new Error(err);
      error.httpStatusCode = 500;
      return next(error);
    });
};

exports.getCart = (req, res, next) => {
  req.user
    .populate('cart.items.productId')
    .execPopulate()
    .then(user => {
      const products = user.cart.items;
      res.render('shop/cart', {
        path: '/cart',
        pageTitle: 'Your Cart',
        products: products
      });
    })
    .catch(err => {
      const error = new Error(err);
      error.httpStatusCode = 500;
      return next(error);
    });
};

exports.postCart = (req, res, next) => {
  const prodId = req.body.productId;
  Product.findById(prodId)
    .then(product => {
      return req.user.addToCart(product);
    })
    .then(result => {
      console.log(result);
      res.redirect('/cart');
    })
    .catch(err => {
      const error = new Error(err);
```

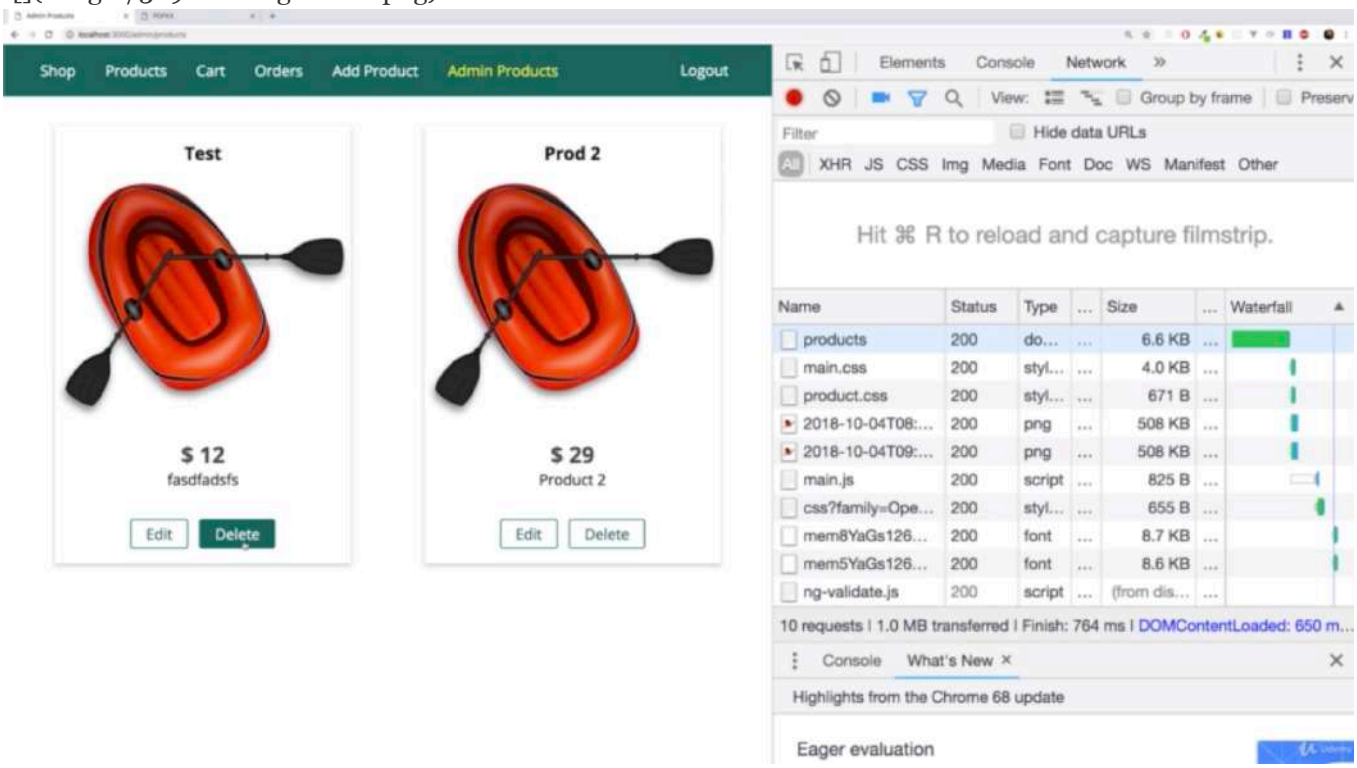
```
      error.httpStatusCode = 500;
      return next(error);
    });
};

exports.postCartDeleteProduct = (req, res, next) => {
  const prodId = req.body.productId;
  req.user
    .removeFromCart(prodId)
    .then(result => {
      res.redirect('/cart');
    })
    .catch(err => {
      const error = new Error(err);
      error.httpStatusCode = 500;
      return next(error);
    });
};

exports.postOrder = (req, res, next) => {
  req.user
    .populate('cart.items.productId')
    .execPopulate()
    .then(user => {
      const products = user.cart.items.map(i => {
        return { quantity: i.quantity, product: { ...i.productId._doc } };
      });
      const order = new Order({
        user: {
          email: req.user.email,
          userId: req.user
        },
        products: products
      });
      return order.save();
    })
    .then(result => {
      return req.user.clearCart();
    })
    .then(() => {
      res.redirect('/orders');
    })
    .catch(err => {
      const error = new Error(err);
      error.httpStatusCode = 500;
      return next(error);
    });
};

exports.getOrders = (req, res, next) => {
  Order.find({ 'user.userId': req.user._id })
    .then(orders => {
      res.render('shop/orders', {
        path: '/orders',
        pageTitle: 'Your Orders',
        orders: orders
```

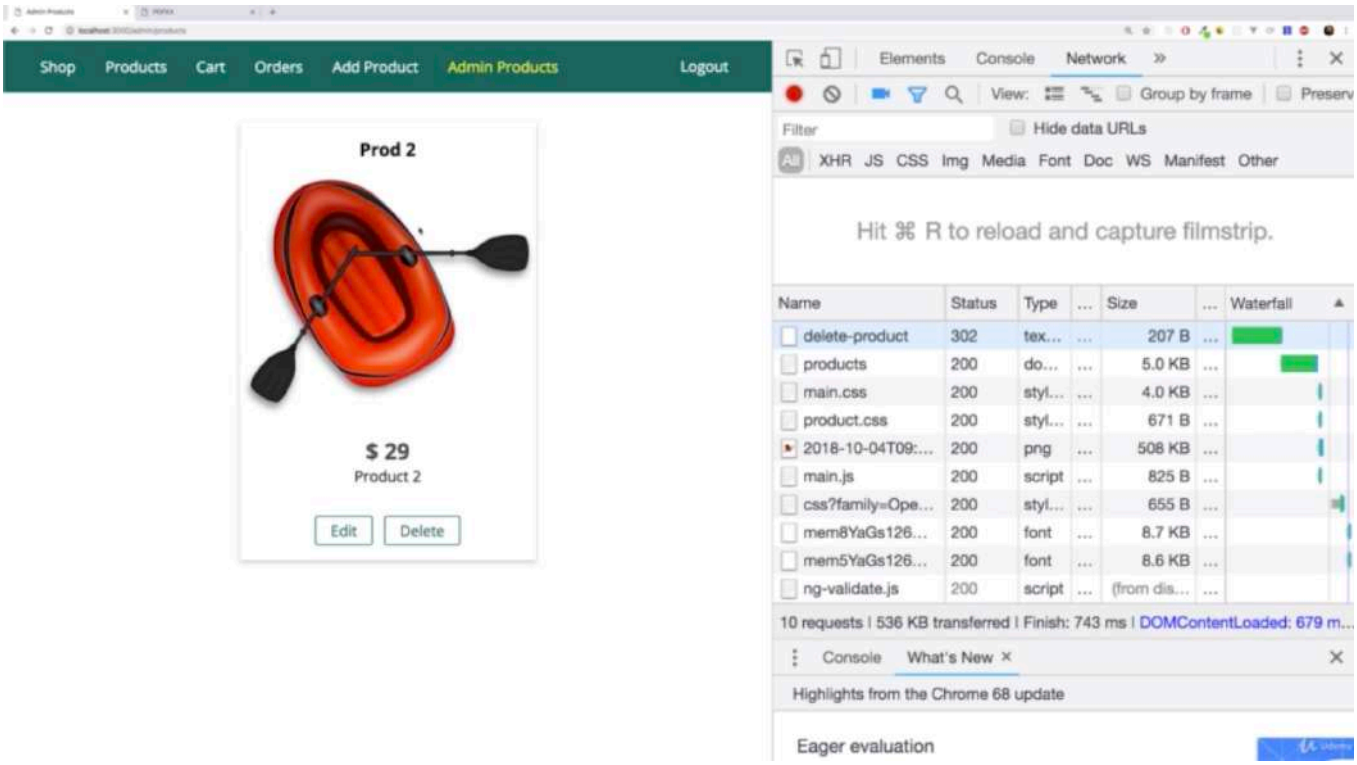
```
    });
  })
  .catch(err => {
    const error = new Error(err);
    error.httpStatusCode = 500;
    return next(error);
  });
};

exports.getInvoice = (req, res, next) => {
  const orderId = req.params.orderId;
  Order.findById(orderId)
    .then(order => {
      if (!order) {
        return next(new Error('No order found.'));
      }
      if (order.user.userId.toString() !== req.user._id.toString()) {
        return next(new Error('Unauthorized'));
      }
      const invoiceName = 'invoice-' + orderId + '.pdf';
      const invoicePath = path.join('data', 'invoices', invoiceName);

      const pdfDoc = new PDFDocument()
      res.setHeader('Content-Type', 'application/pdf');
      res.setHeader(
        'Content-Disposition',
        'inline; filename="' + invoiceName + '"'
      )
      pdfDoc.pipe(fs.createWriteStream(invoicePath))
      pdfDoc.pipe(res)

      pdfDoc.fontSize(26).text('Invoice', {
        underline: true
      })

      pdfDoc.text('-----------------------')
      let totalPrice = 0
      /**'products' is an array
       * because we store this products in a database as an array.
       */
      order.products.forEach(prod => {
        totalPrice += prod.quantity * prod.product.price
        pdfDoc
          .fontSize(14)
          .text(
            prod.product.title +
            ' - ' +
            prod.quantity +
            ' x ' +
            '$' +
            prod.product.price)
      })
      pdfDoc.text('---')
      pdfDoc.fontSize(20).text('Total Price: $' + totalPrice)

      pdfDoc.end()
```

```
204       //fs.readFile(invoicePath, (err, data) => {
205       //  if (err) {
206       //    return next(err);
207       //  }
208       //  res.setHeader('Content-Type', 'application/pdf');
209       //  res.setHeader(
210       //    'Content-Disposition',
211       //    'inline; filename="' + invoiceName + '"'
212       //  );
213       //  res.send(data);
214       //});
215
216       //const file = fs.createReadStream(invoicePath)
217       //file.pipe(res)
218     })
219     .catch(err => next(err));
220 };
221
```

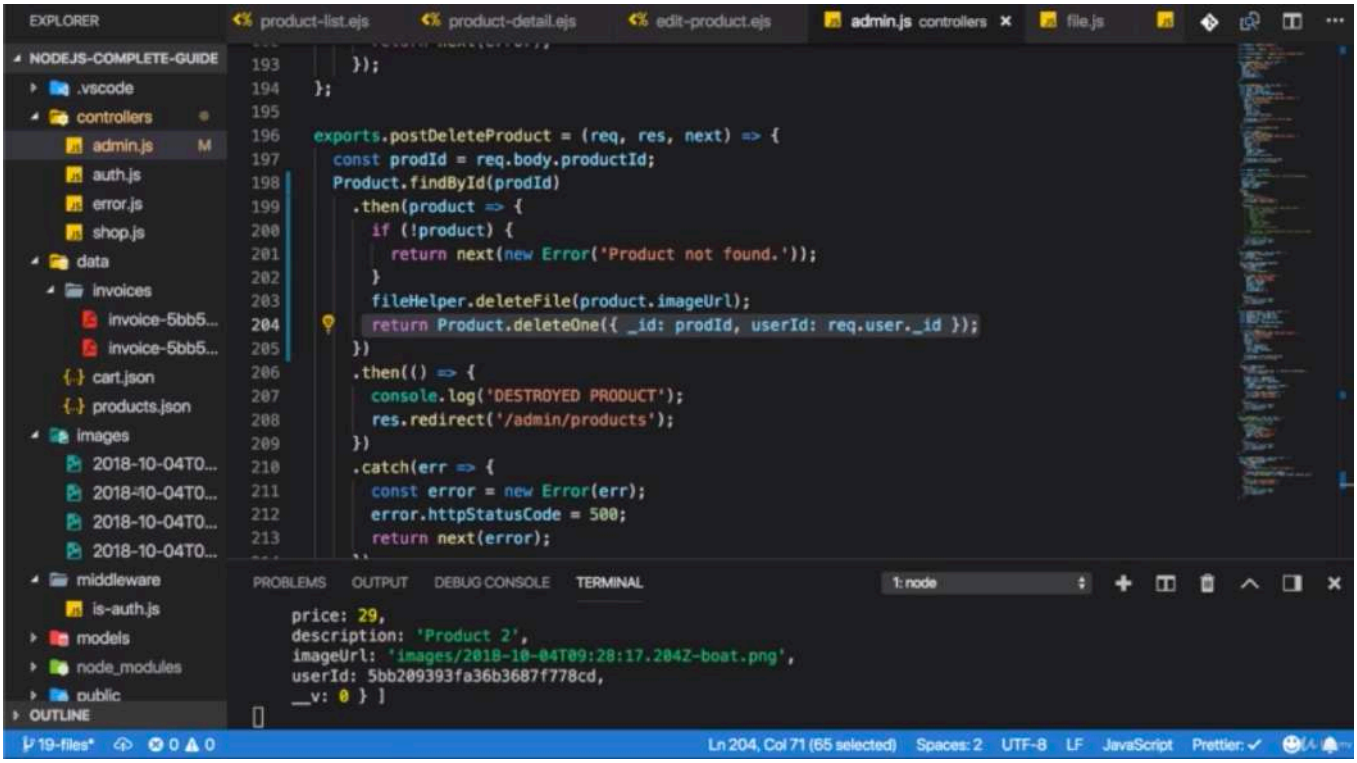
* Chapter 329: Deleting Files

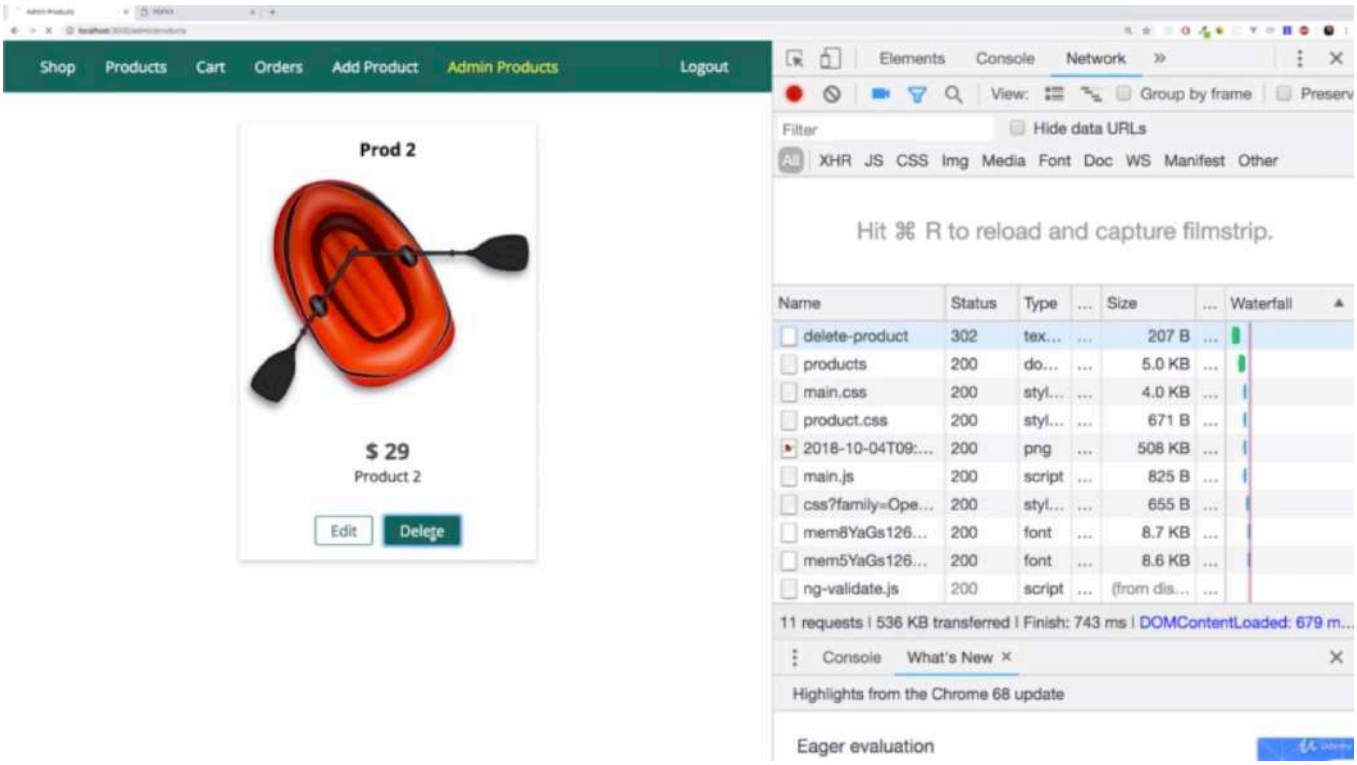
1. update
- ./util/file.js
- ./controllers/admin.js

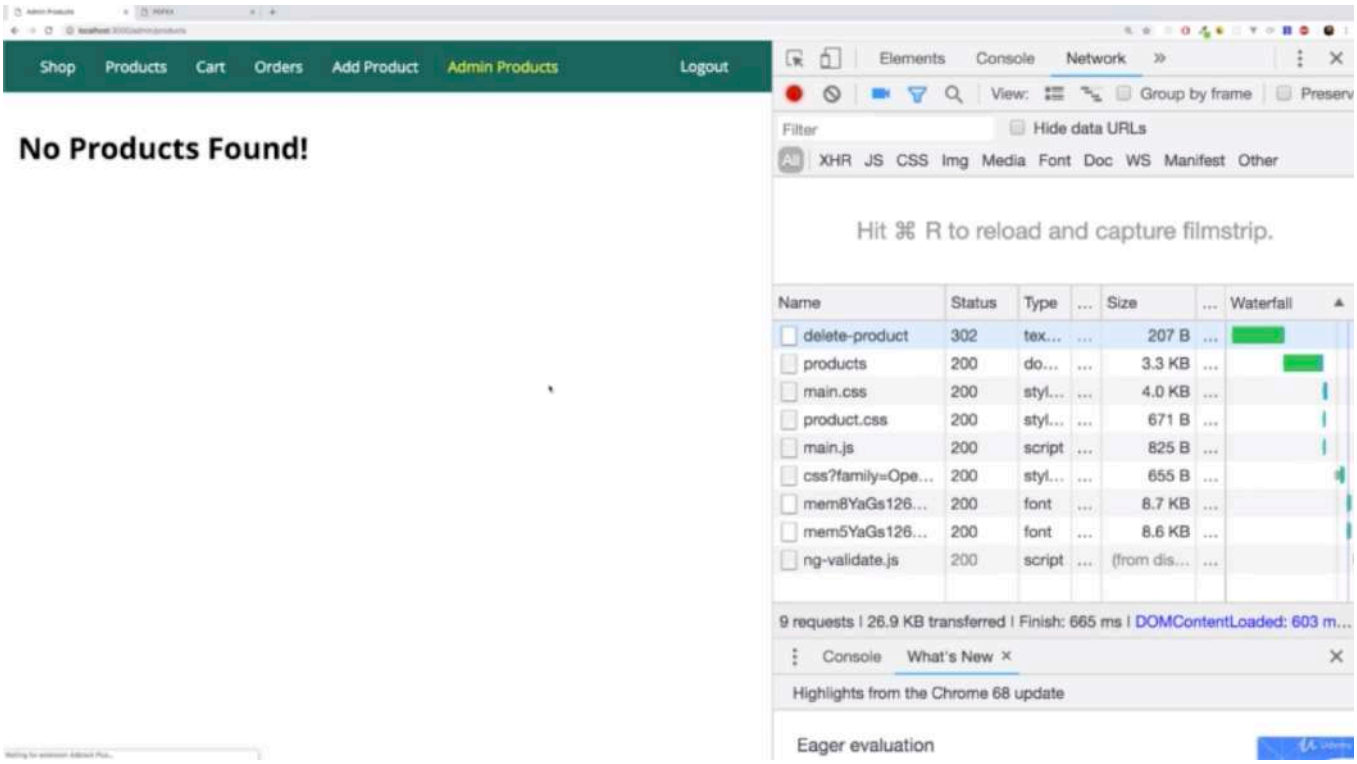


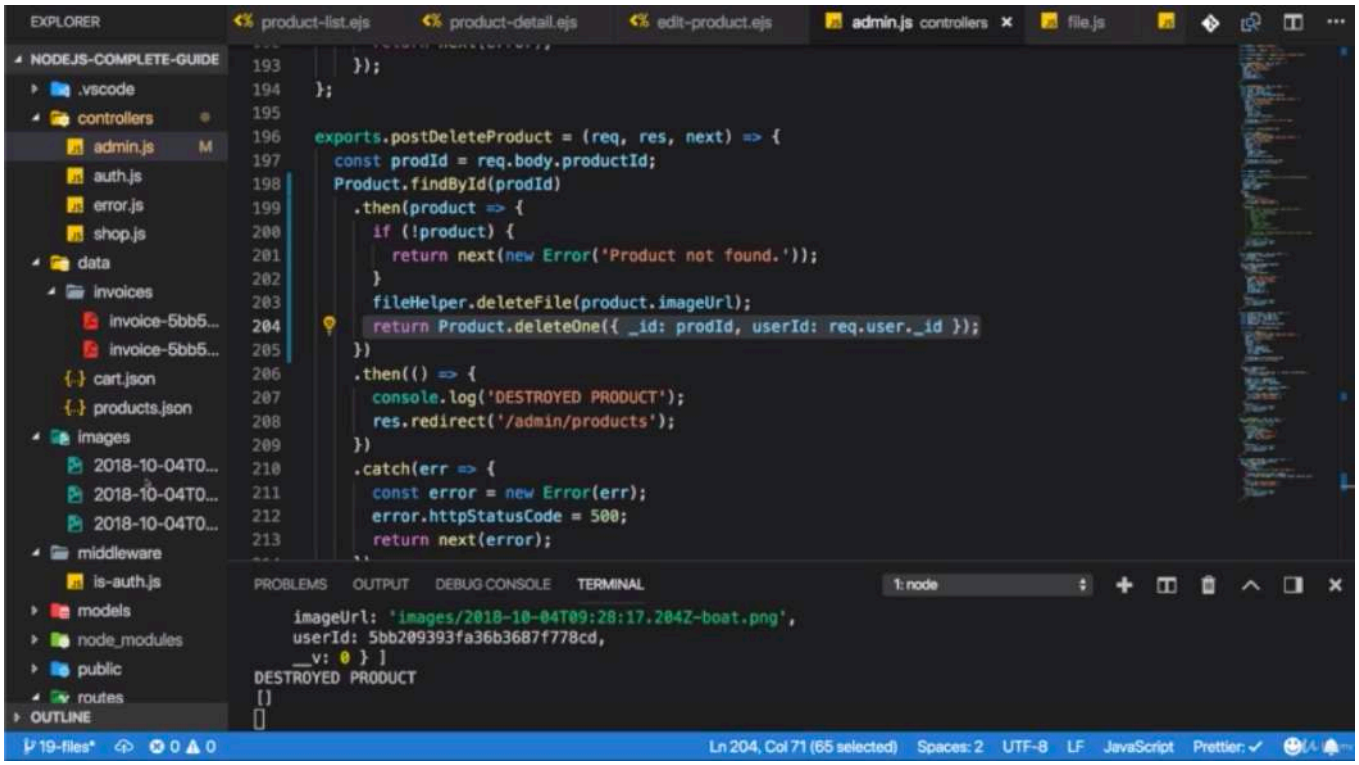


- i delete prod1, and in the images folder, one image connected to prod1 is removed.

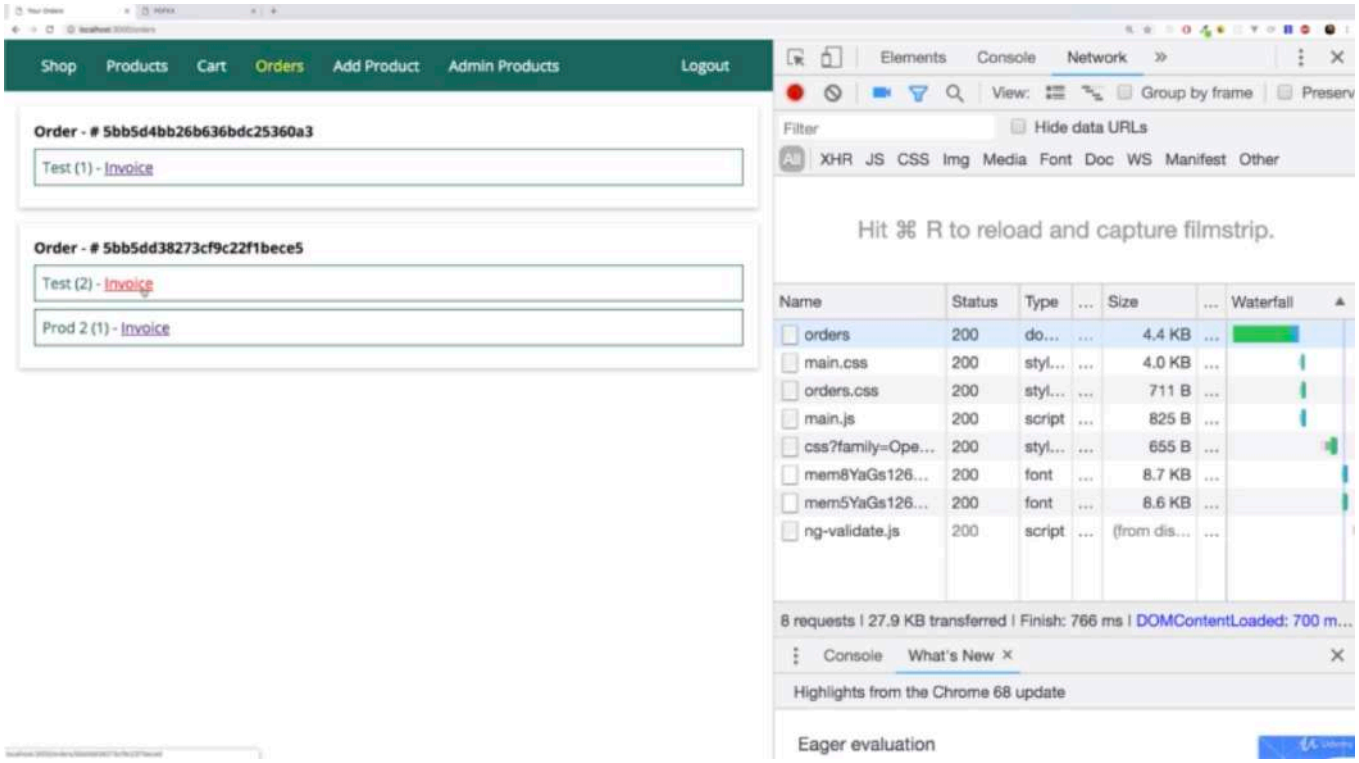



Shop
Products
Cart
Orders
Add Product
Admin Products
Logout
Prod 2
$ 29
Product 2
Edit
Delete
Elements
Console
Network
Hit ⌘ R to reload and capture filmstrip.
Name
Status
Type
Size
Waterfall
delete-product 302 tex... 207 B
products 200 do... 5.0 KB
main.css 200 styl... 4.0 KB
product.css 200 styl... 671 B
2018-10-04T09:... 200 png 508 KB
main.js 200 script 825 B
css?family=Ope... 200 styl... 655 B
mem8YaGs126... 200 font 8.7 KB
mem5YaGs126... 200 font 8.6 KB
ng-validate.js 200 script (from dis...
11 requests | 536 KB transferred | Finish: 743 ms | DOMContentLoaded: 679 m...
Console
What's New
Highlights from the Chrome 68 update
Eager evaluation

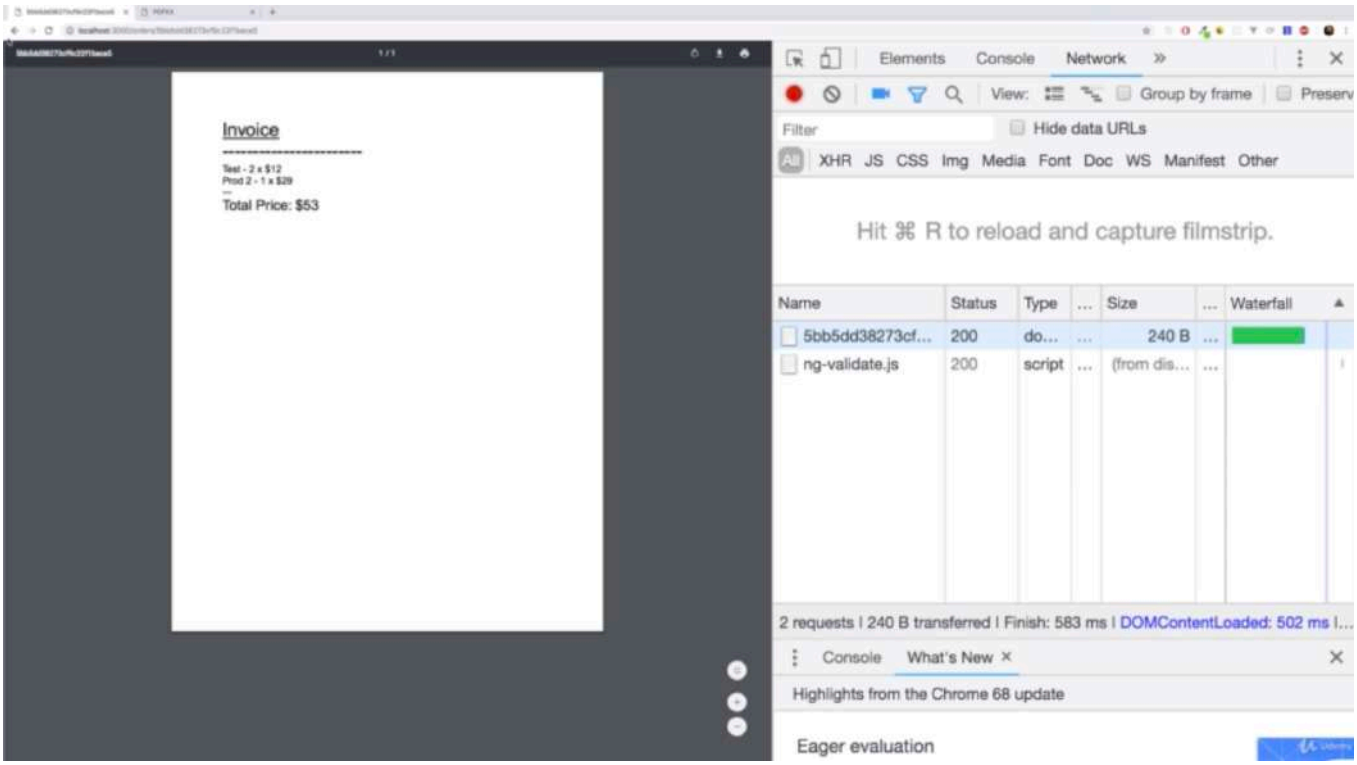

Shop
Products
Cart
Orders
Add Product
Admin Products
Logout
No Products Found!
Elements
Console
Network
Hit ⌘ R to reload and capture filmstrip.
Name
Status
Type
Size
Waterfall
delete-product 302 tex... 207 B
products 200 do... 3.3 KB
main.css 200 styl... 4.0 KB
product.css 200 styl... 671 B
main.js 200 script 825 B
css?family=Ope... 200 styl... 655 B
mem8YaGs126... 200 font 8.7 KB
mem5YaGs126... 200 font 8.6 KB
ng-validate.js 200 script (from dis...
9 requests | 26.9 KB transferred | Finish: 665 ms | DOMContentLoaded: 603 m...
Console
What's New
Highlights from the Chrome 68 update
Eager evaluation


EXPLORER
product-list.ejs
product-detail.ejs
edit-product.ejs
admin.js controllers
file.js
NODEJS-COMPLETE-GUIDE
.vscode
controllers
admin.js
auth.js
error.js
shop.js
data
invoices
cart.json
products.json
images
middleware
is-auth.js
models
node_modules
public
routes
OUTLINE
193 });
194 };
196 exports.postDeleteProduct = (req, res, next) => {
197 const prodId = req.body.productId;
198 Product.findById(prodId)
199 .then(product => {
200 if (!product) {
201 return next(new Error('Product not found.'));
202 }
203 fileHelper.deleteFile(product.imageUrl);
204 return Product.deleteOne({ _id: prodId, userId: req.user._id });
205 })
206 .then(() => {
207 console.log('DESTROYED PRODUCT');
208 res.redirect('/admin/products');
209 })
210 .catch(err => {
211 const error = new Error(err);
212 error.httpStatusCode = 500;
213 return next(error);
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
1: node
imageUrl: 'images/2018-10-04T09:28:17.204Z-boat.png',
userId: 5bb209393fa36b3687f778cd,
__v: 0 }]
DESTROYED PRODUCT
[]
Ln 204, Col 71 (65 selected)
Spaces: 2
UTF-8
LF
JavaScript
Prettier:

——————————————


Shop
Products
Cart
Orders
Add Product
Admin Products
Logout
Order - # 5bb5d4bb26b636bdc25360a3
Test (1) - Invoice
Order - # 5bb5dd38273cf9c22f1bece5
Test (2) - Invoice
Prod 2 (1) - Invoice
Elements
Console
Network
Hit ⌘ R to reload and capture filmstrip.
Name
Status
Type
Size
Waterfall
orders 200 do... 4.4 KB
main.css 200 styl... 4.0 KB
orders.css 200 styl... 711 B
main.js 200 script 825 B
css?family=Ope... 200 styl... 655 B
mem8YaGs126... 200 font 8.7 KB
mem5YaGs126... 200 font 8.6 KB
ng-validate.js 200 script (from dis...
8 requests | 27.9 KB transferred | Finish: 766 ms | DOMContentLoaded: 700 m...
Console
What's New
Highlights from the Chrome 68 update
Eager evaluation

Shop
Products
Cart
Orders
Add Product
Admin Products
Logout
Order - # 5bb5d4bb26b636bdc25360a3
Test (1) - Invoice
Order - # 5bb5dd38273cf9c22f1bece5
Test (2) - Invoice
Prod 2 (1) - Invoice
Elements
Console
Network
Hit ⌘ R to reload and capture filmstrip.
Name
Status
Type
Size
Waterfall
orders 200 do... 4.4 KB
main.css 200 styl... 4.0 KB
orders.css 200 styl... 711 B
main.js 200 script 825 B
css?family=Ope... 200 styl... 655 B
mem8YaGs126... 200 font 8.7 KB
mem5YaGs126... 200 font 8.6 KB
ng-validate.js 200 script (from dis...
8 requests | 27.9 KB transferred | Finish: 766 ms | DOMContentLoaded: 700 m...
Console
What's New
Highlights from the Chrome 68 update
Eager evaluation



- fetching my invoice or generating it on the fly works. thanks to the fact that we store a snapshot of that in the database, it even works after the product was deleted.

```
//./util/file.js

const fs = require('fs')

const deleteFile = (filePath) => {
    /**'unlink()' method deletes the name and file
     * that is connected to the name,
     * so it deletes a file at this path.
     */
    fs.unlink(filePath, (err) => {
        if (err) {
            throw (err)
        }
    })
}

exports.deleteFile = deleteFile
```

```
// ./controllers/admin.js

const mongoose = require('mongoose');

const fileHelper = require('../util/file')

const { validationResult } = require('express-validator/check');

const Product = require('../models/product');

exports.getAddProduct = (req, res, next) => {
  res.render('admin/edit-product', {
    pageTitle: 'Add Product',
    path: '/admin/add-product',
    editing: false,
```

```
    hasError: false,
    errorMessage: null,
    validationErrors: []
  });
};

exports.postAddProduct = (req, res, next) => {
  const title = req.body.title;
  const image = req.file;
  const price = req.body.price;
  const description = req.body.description;
  if (!image) {
    res.status(422).render('admin/edit-product', {
      pageTitle: 'Add Product',
      path: '/admin/add-product',
      editing: false,
      hasError: true,
      product: {
        title: title,
        price: price,
        description: description
      },
      errorMessage: 'Attached file is not an image',
      validationErrors: []
    });
  }
  const errors = validationResult(req);

  if (!errors.isEmpty()) {
    console.log(errors.array());
    return res.status(422).render('admin/edit-product', {
      pageTitle: 'Add Product',
      path: '/admin/add-product',
      editing: false,
      hasError: true,
      product: {
        title: title,
        imageUrl: imageUrl,
        price: price,
        description: description
      },
      errorMessage: errors.array()[0].msg,
      validationErrors: errors.array()
    });
  }
  const imageUrl = image.path

  const product = new Product({
    //_id: new mongoose.Types.ObjectId('5badf72403fd8b5be0366e81'),
    title: title,
    price: price,
    description: description,
    imageUrl: imageUrl,
    userId: req.user
  });
  product
```

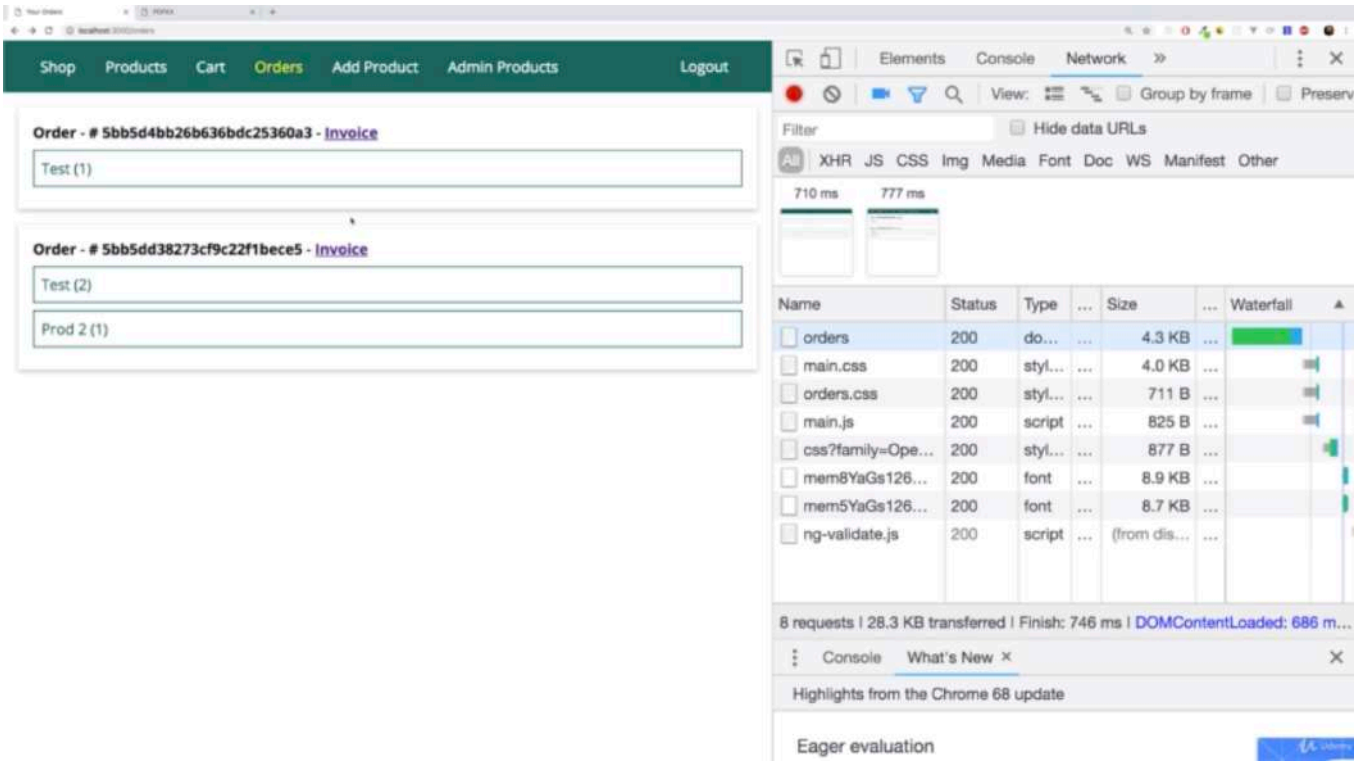
```
    .save()
    .then(result => {
      // console.log(result);
      console.log('Created Product');
      res.redirect('/admin/products');
    })
    .catch(err => {
      // return res.status(500).render('admin/edit-product', {
      //   pageTitle: 'Add Product',
      //   path: '/admin/add-product',
      //   editing: false,
      //   hasError: true,
      //   product: {
      //     title: title,
      //     imageUrl: imageUrl,
      //     price: price,
      //     description: description
      //   },
      //   errorMessage: 'Database operation failed, please try again.',
      //   validationErrors: []
      // });
      //res.redirect('/500');
      const error = new Error(err)
      error.httpStatusCode = 500
      return next(error)
    });
};

exports.getEditProduct = (req, res, next) => {
  const editMode = req.query.edit;
  if (!editMode) {
    return res.redirect('/');
  }
  const prodId = req.params.productId;
  Product.findById(prodId)
    .then(product => {
      if (!product) {
        return res.redirect('/');
      }
      res.render('admin/edit-product', {
        pageTitle: 'Edit Product',
        path: '/admin/edit-product',
        editing: editMode,
        product: product,
        hasError: false,
        errorMessage: null,
        validationErrors: []
      });
    })
    .catch(err => {
      const error = new Error(err)
      error.httpStatusCode = 500
      return next(error)
    });
};

```

```
exports.postEditProduct = (req, res, next) => {
  const prodId = req.body.productId;
  const updatedTitle = req.body.title;
  const updatedPrice = req.body.price;
  const image = req.body.imageUrl;
  const updatedDesc = req.body.description;

  const errors = validationResult(req);

  if (!errors.isEmpty()) {
    return res.status(422).render('admin/edit-product', {
      pageTitle: 'Edit Product',
      path: '/admin/edit-product',
      editing: true,
      hasError: true,
      product: {
        title: updatedTitle,
        price: updatedPrice,
        description: updatedDesc,
        _id: prodId
      },
      errorMessage: errors.array()[0].msg,
      validationErrors: errors.array()
    });
  }

  Product.findById(prodId)
    .then(product => {
      if (product.userId.toString() !== req.user._id.toString()) {
        return res.redirect('/');
      }
      product.title = updatedTitle;
      product.price = updatedPrice;
      product.description = updatedDesc;
      if (image) {
        fileHelper.deleteFile(product.imageUrl)
        product.imageUrl = updatedImageUrl;
      }
      return product.save().then(result => {
        console.log('UPDATED PRODUCT!');
        res.redirect('/admin/products');
      });
    })
    .catch(err => {
      const error = new Error(err)
      error.httpStatusCode = 500
      return next(error)
    });
};

exports.getProducts = (req, res, next) => {
  Product.find({ userId: req.user._id })
    // .select('title price -_id')
    // .populate('userId', 'name')
    .then(products => {
      console.log(products);
```

```
      res.render('admin/products', {
        prods: products,
        pageTitle: 'Admin Products',
        path: '/admin/products'
      });
    })
    .catch(err => {
      const error = new Error(err)
      error.httpStatusCode = 500
      return next(error)
    });
};

exports.postDeleteProduct = (req, res, next) => {
  const prodId = req.body.productId;
  Product.findById(prodId)
    .then(product => {
      if (!product) {
        return next(new Error('Product not found'))
      }
      fileHelper.deleteFile(product.imageUrl)
      /**i should trigger 'deleteOne'
       * after i found 'product' above
       * otherwise we have a race condition
       * where deleting could finish before finding is finished
       * and that would be bad.
       */
      return Product.deleteOne({ _id: prodId, userId: req.user._id })
    })
    .then(() => {
      console.log('DESTROYED PRODUCT');
      res.redirect('/admin/products');
    })
    .catch(err => {
      const error = new Error(err)
      error.httpStatusCode = 500
      return next(error)
    });
};

```

* Chapter 330: Fixing Invoice Links

1. update
- ./views/shop/orders.ejs




```
<!--./views/shop/orders.ejs-->

<%- include('../includes/head.ejs') %>
    <link rel="stylesheet" href="/css/orders.css">
    </head>

    <body>
        <%- include('../includes/navigation.ejs') %>
        <main>
            <% if (orders.length <= 0) { %>
                <h1>Nothing there!</h1>
            <% } else { %>
                <ul class="orders">
                    <% orders.forEach(order => { %>
                        <li class="orders__item">
                            <h1>Order - # <%= order._id %> - <a href="/orders/<%= order._id %>">Invoice</a></h1>
                            <ul class="orders__products">
                                <% order.products.forEach(p => { %>
                                    <li class="orders__products-item"><%= p.product.title %> (<%= p.quantity %>)</li>
                                <% }); %>
                            </ul>
                        </li>
                    <% }); %>
                </ul>
            <% } %>
        </main>
        <%- include('../includes/end.ejs') %>
```