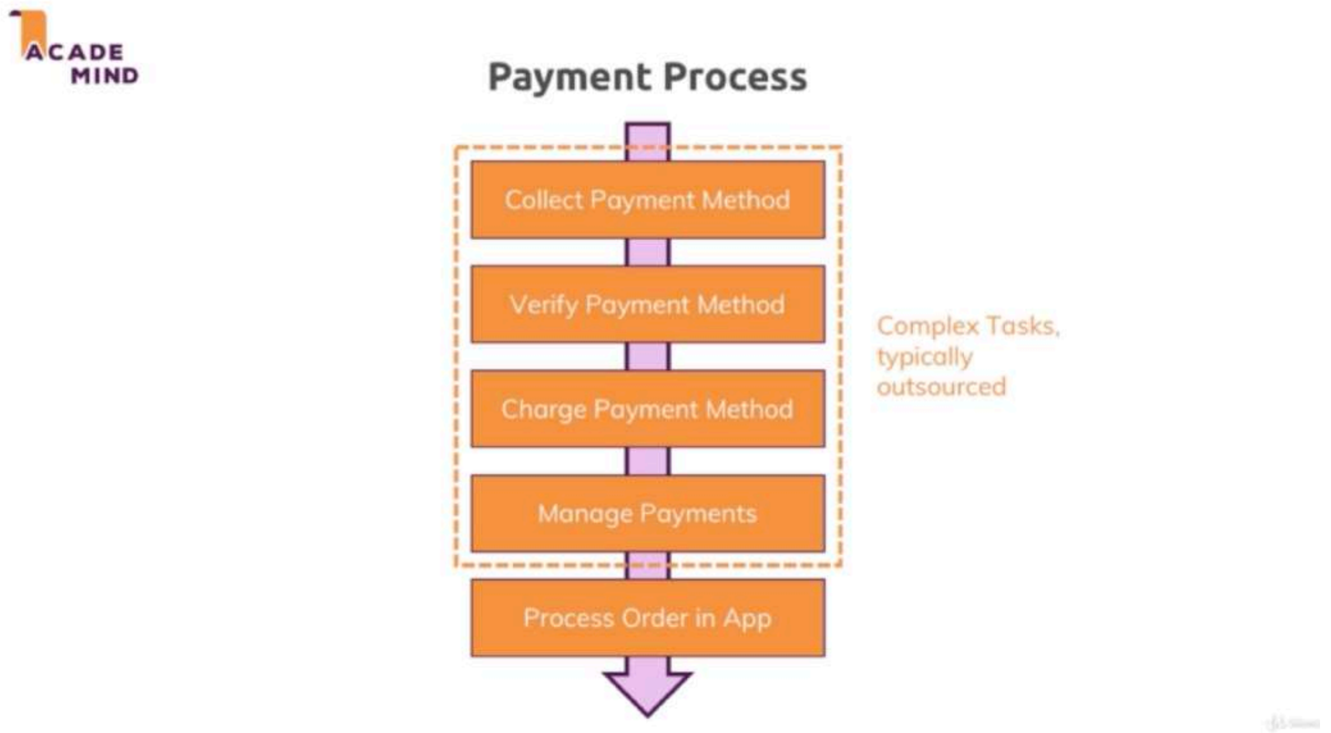
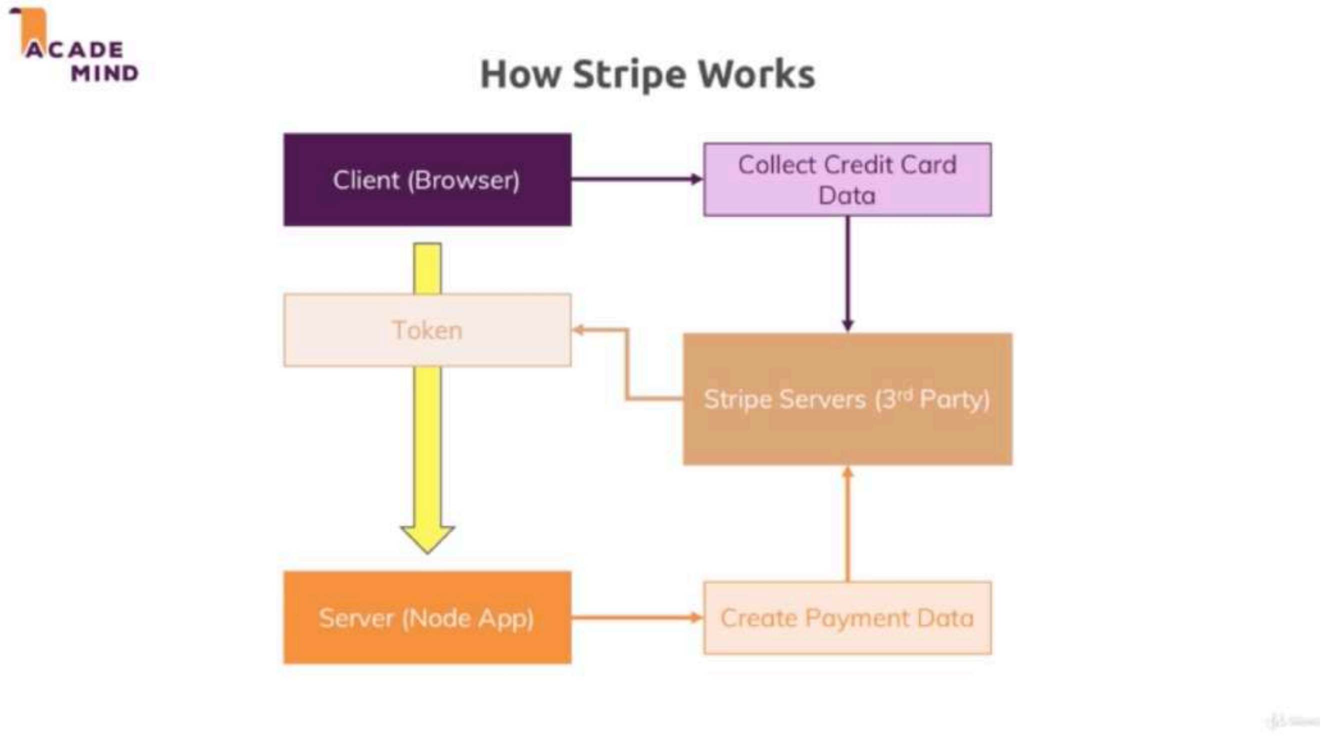
23. Adding Payments

* Chapter 350: How Payments Work

![](images/350-how-payments-work-1.png)
![](images/350-how-payments-work-2.png)

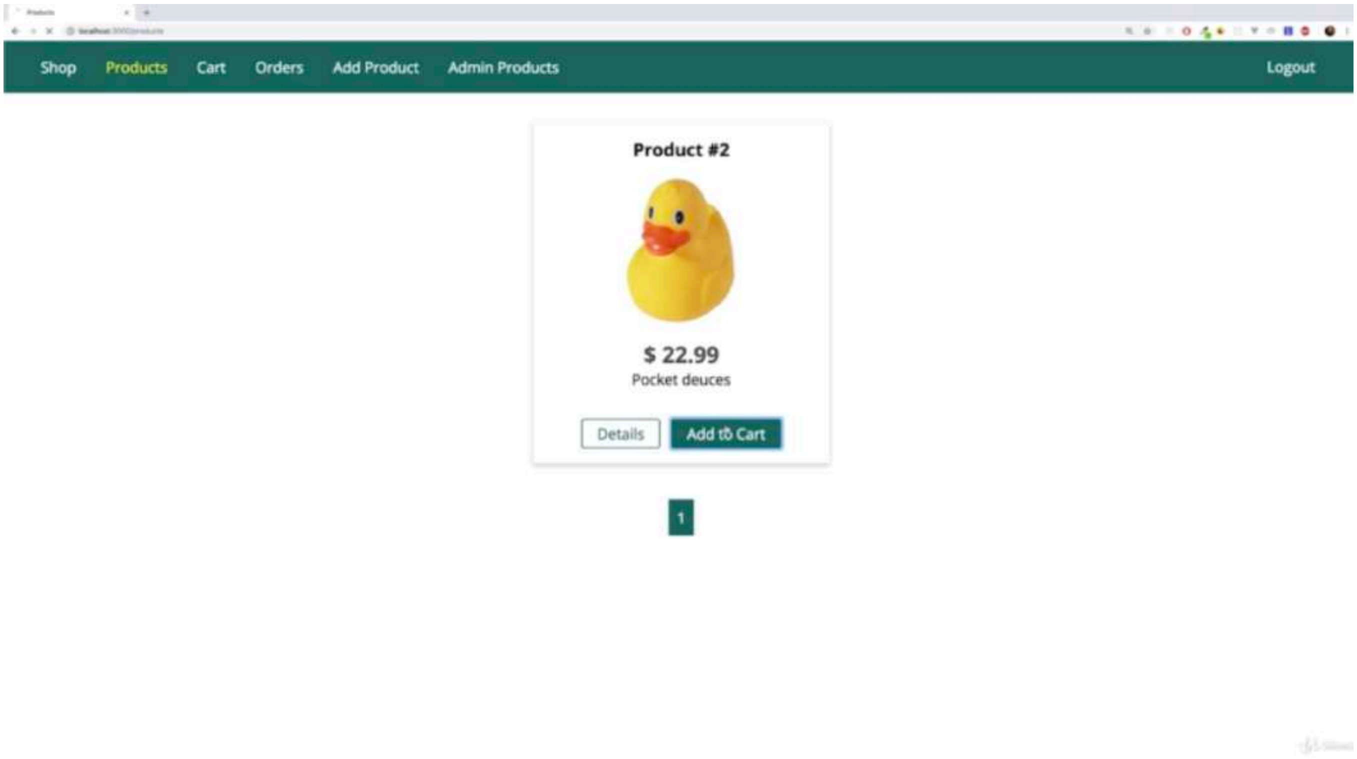


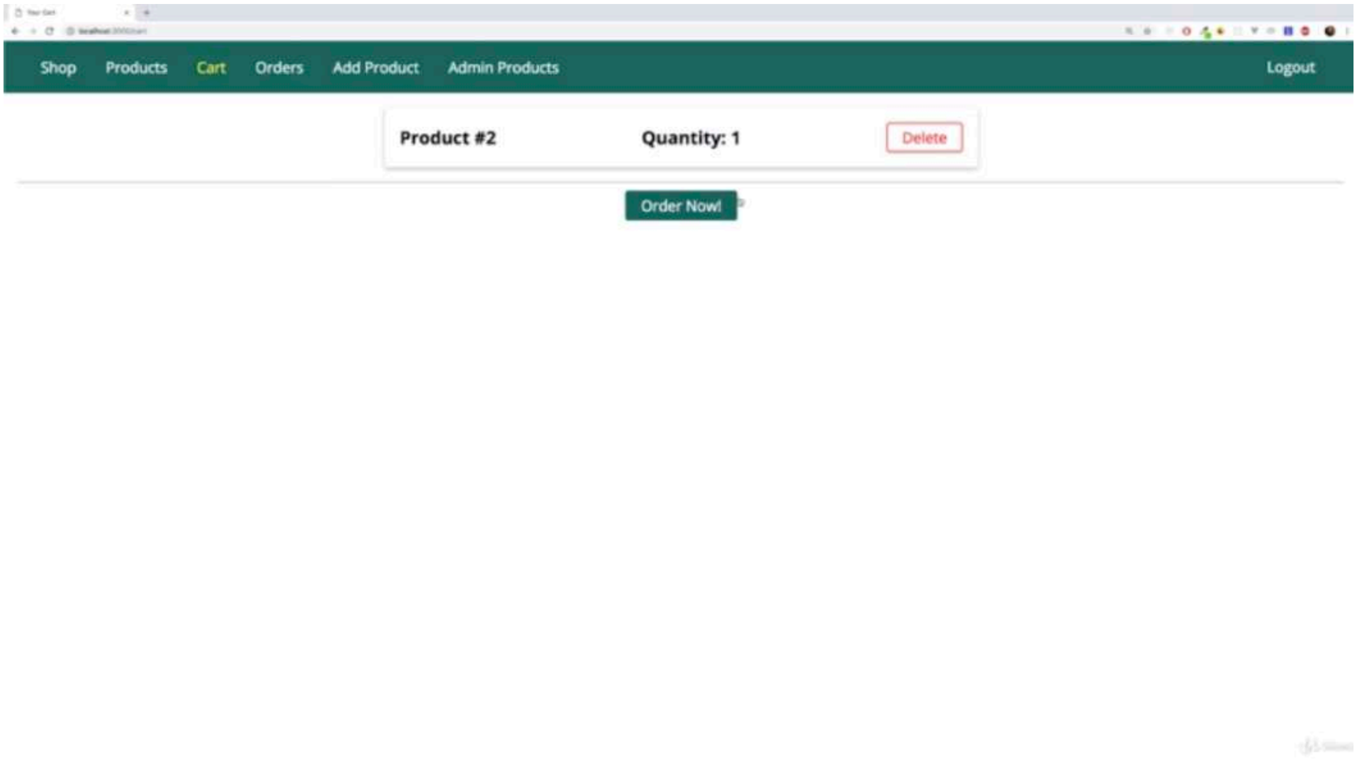


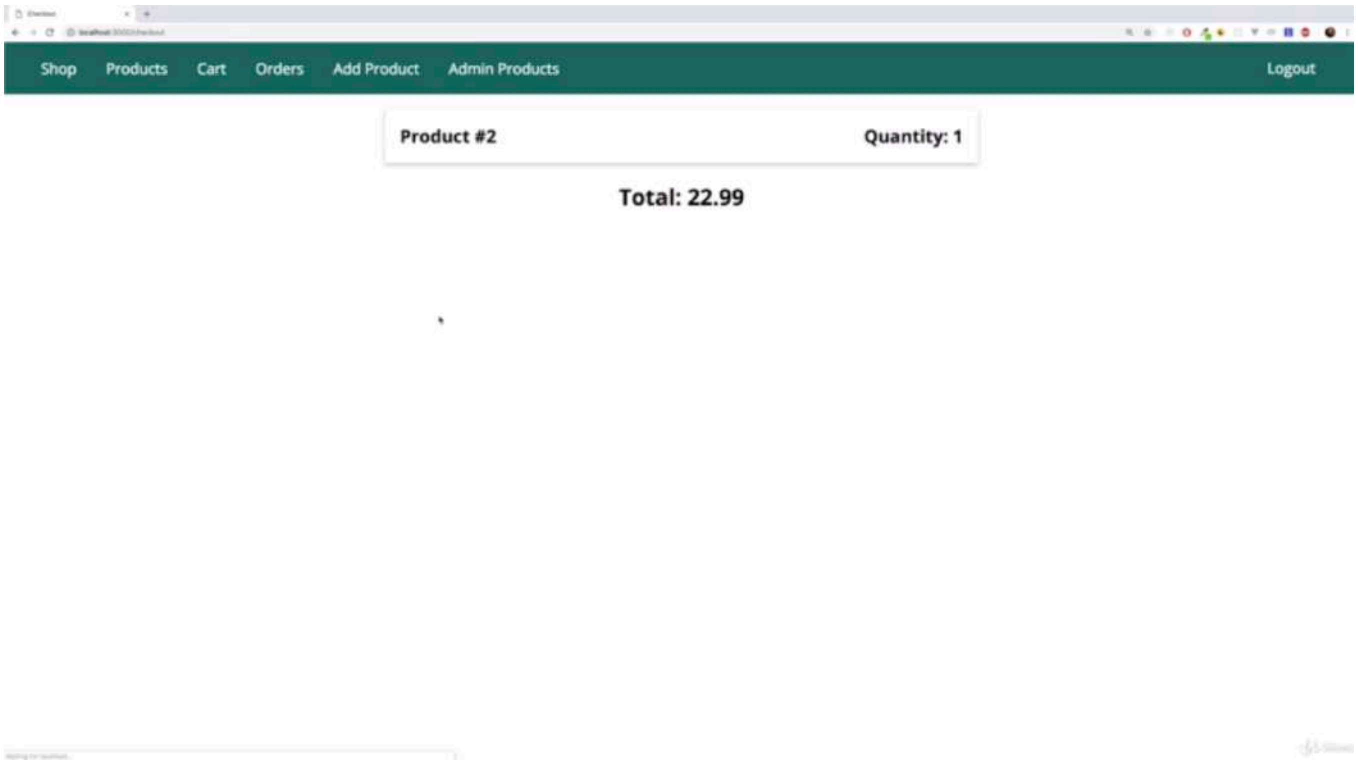
* Chapter 351: Adding A Checkout Page

1. update
- ./views/shop/checkout.ejs
- ./routes/shop.js
- ./controllers/shop.js
- ./views/shop/cart.ejs

![](images/351-adding-a-checkout-page-1.png)
![](images/351-adding-a-checkout-page-2.png)
![](images/351-adding-a-checkout-page-3.png)







Product #2 Quantity: 1

Total: 22.99

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

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

    </head>

    <body>
        <%- include('../includes/navigation.ejs') %>
        <main>
            <ul class="cart__item-list">
                <% products.forEach(p => { %>
                    <li class="cart__item">
                        <h1><%= p.productId.title %></h1>
                        <h2>Quantity: <%= p.quantity %></h2>
                    </li>
                <% }) %>
            </ul>
            <div class="centered">
                <h2>Total: <%= totalSum %></h2>
            </div>
        </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();

```

```javascript
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.get('/checkout', isAuth, shopController.getCheckout)

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

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

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

module.exports = router;
```

```javascript
//./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');

const ITEMS_PER_PAGE = 2;

exports.getProducts = (req, res, next) => {
  const page = +req.query.page || 1;
  let totalItems;

  Product.find()
    .countDocuments()
    .then(numProducts => {
      totalItems = numProducts;
      return Product.find()
        .skip((page - 1) * ITEMS_PER_PAGE)
        .limit(ITEMS_PER_PAGE);
    })
    .then(products => {
      res.render('shop/product-list', {
        prods: products,
        pageTitle: 'Products',
        path: '/products',
        currentPage: page,
        hasNextPage: ITEMS_PER_PAGE * page < totalItems,
        hasPreviousPage: page > 1,
        nextPage: page + 1,
        previousPage: page - 1,
        lastPage: Math.ceil(totalItems / ITEMS_PER_PAGE)
      });
    })
    .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) => {
  const page = +req.query.page || 1;
  let totalItems;

  Product.find()
    .countDocuments()
    .then(numProducts => {
      totalItems = numProducts;
      return Product.find()
        .skip((page - 1) * ITEMS_PER_PAGE)
        .limit(ITEMS_PER_PAGE);
    })
    .then(products => {
      res.render('shop/index', {
        prods: products,
        pageTitle: 'Shop',
        path: '/',
        currentPage: page,
        hasNextPage: ITEMS_PER_PAGE * page < totalItems,
        hasPreviousPage: page > 1,
        nextPage: page + 1,
        previousPage: page - 1,
        lastPage: Math.ceil(totalItems / ITEMS_PER_PAGE)
      });
    })
    .catch(err => {
      const error = new Error(err);
      error.httpStatusCode = 500;
      return next(error);
    });
```

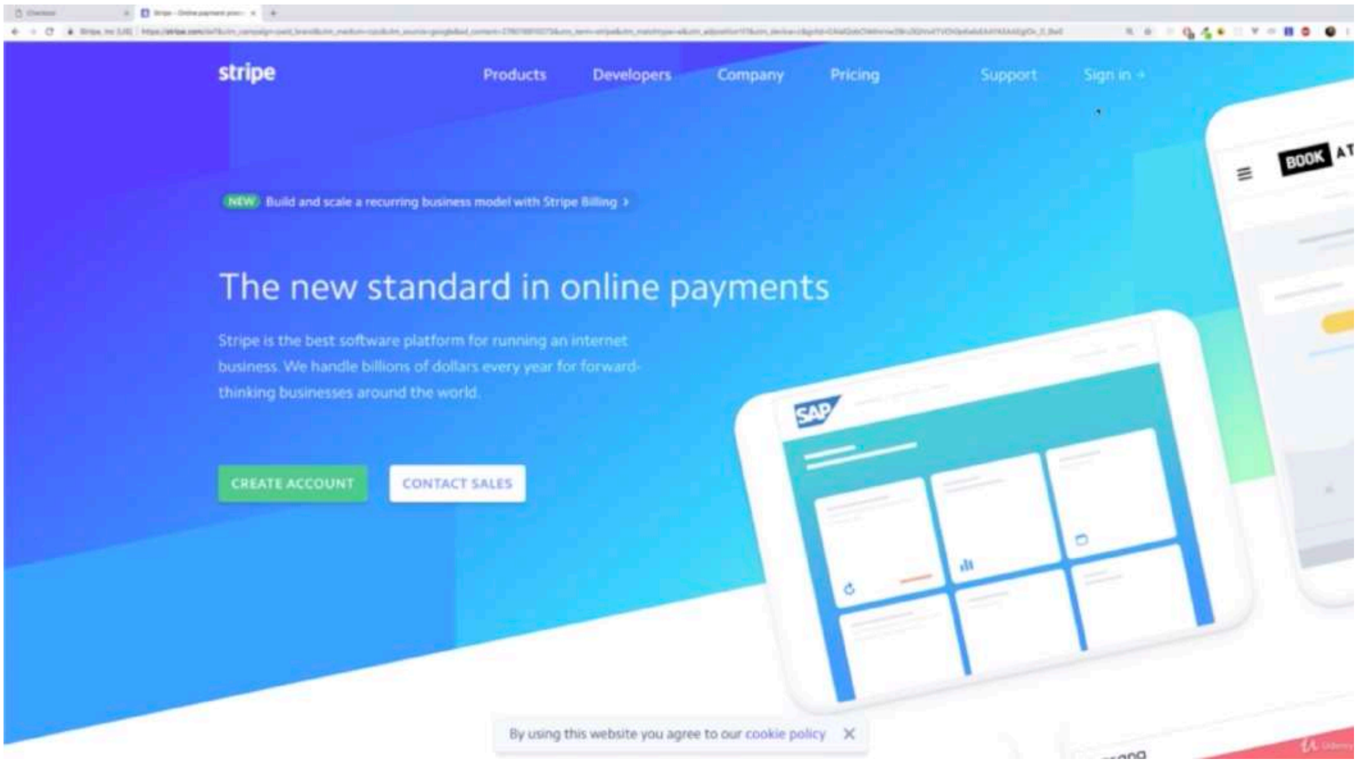
```javascript
};

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.getCheckout = (req, res, next) => {
  req.user
    .populate('cart.items.productId')
    .execPopulate()
    .then(user => {
      const products = user.cart.items;
      let total = 0
      products.forEach(p => {
        total += p.quantity * p.productId.price
      })
      res.render('shop/checkout', {
        path: '/checkout',
        pageTitle: 'Checkout',
        products: products,
        /**'products' is in an array of products
         * where we have the quantity
         * and a productId field with detail data about the products
         * because we populate this productId field with the detailed product data
         */
        totalSum: total
      });
    })
    .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;
      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();
    // 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));
};

```

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

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

    <body>
        <%- include('../includes/navigation.ejs') %>
        <main>
            <% if (products.length > 0) { %>
                <ul class="cart__item-list">
                    <% products.forEach(p => { %>
                        <li class="cart__item">
                            <h1><%= p.productId.title %></h1>
                            <h2>Quantity: <%= p.quantity %></h2>
                            <form action="/cart-delete-item" method="POST">
                                <input type="hidden" value="<%= p.productId._id %>"
name="productId">
                                <input type="hidden" name="_csrf" value="<%= csrfToken %>">
                                <button class="btn danger" type="submit">Delete</button>
                            </form>
                        </li>
                    <% }) %>
                </ul>
                <hr>
                <div class="centered">
                    <!--
                    <form action="/create-order" method="POST">
                        <input type="hidden" name="_csrf" value="<%= csrfToken %>">
                        <button type="submit" class="btn">Order Now!</button>
                    </form>
                    -->
                    <a class="btn" href="/checkout">Order Now!</a>
                </div>

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

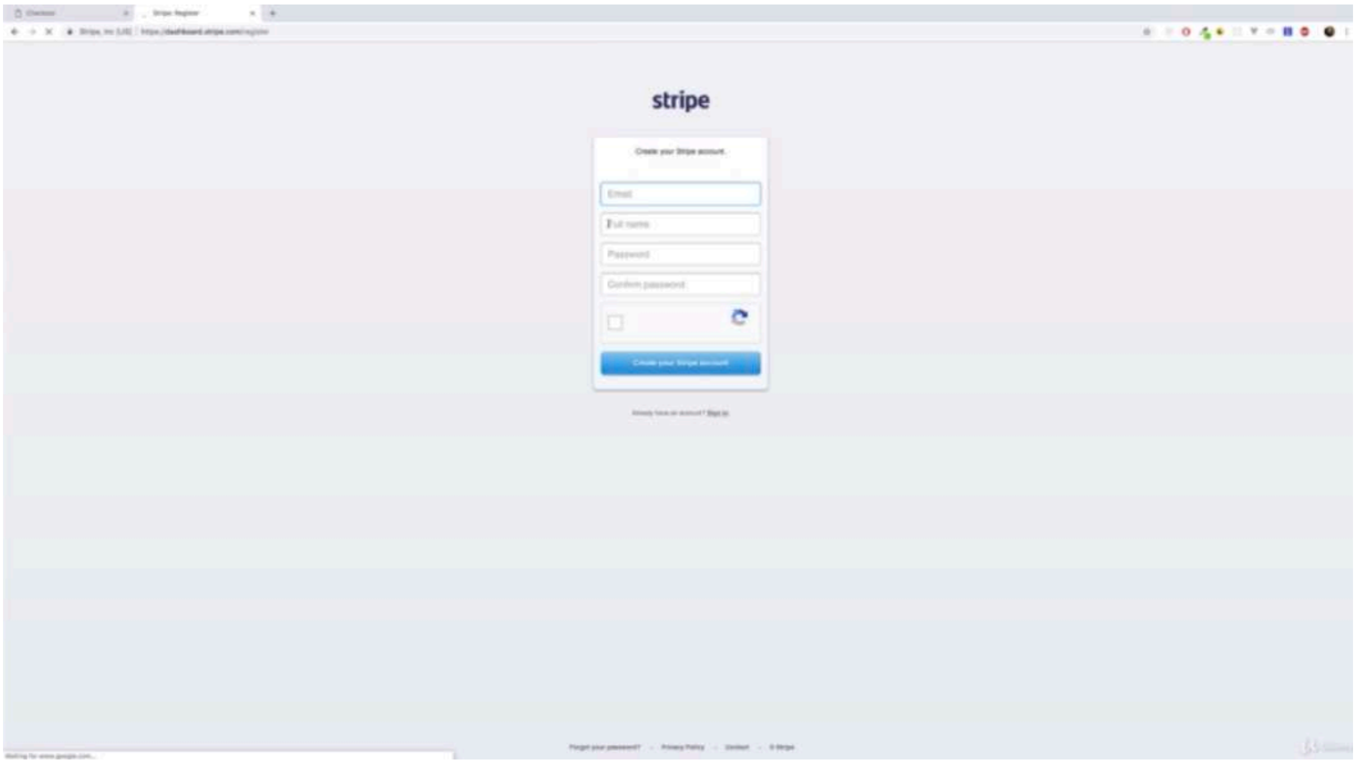
* Chapter 352: Using Stripe In Your App

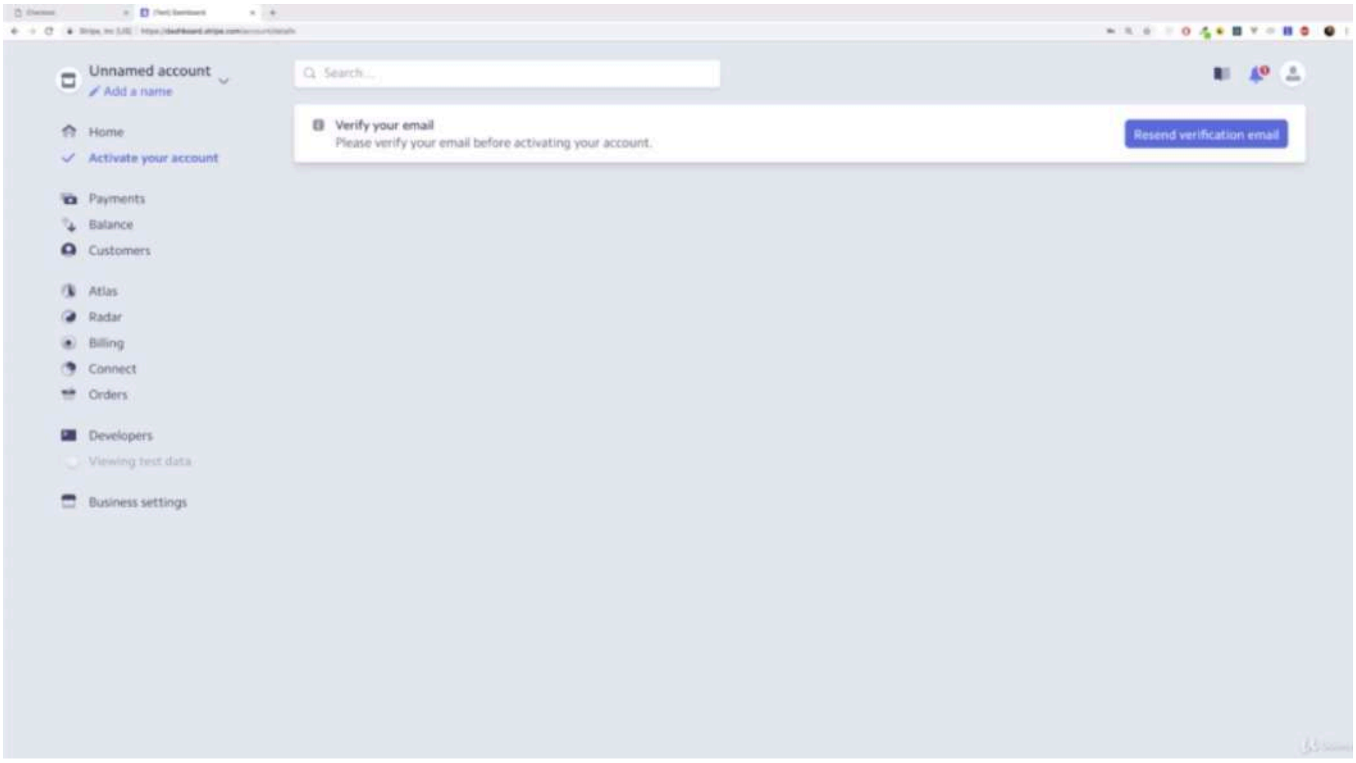
1. update
- ./views/shop/checkout.ejs
- ./controllers/shop.js
- app.js
- ./routes/shop.js

![](images/352-using-stripe-in-your-app-1.png)
![](images/352-using-stripe-in-your-app-2.png)
![](images/352-using-stripe-in-your-app-3.png)


stripe
Products
Developers
Company
Pricing
Support
Sign in →
NEW Build and scale a recurring business model with Stripe Billing ›
The new standard in online payments
CREATE ACCOUNT
CONTACT SALES
By using this website you agree to our cookie policy

Stripe is the best software platform for running an internet business. We handle billions of dollars every year for forward-thinking businesses around the world.

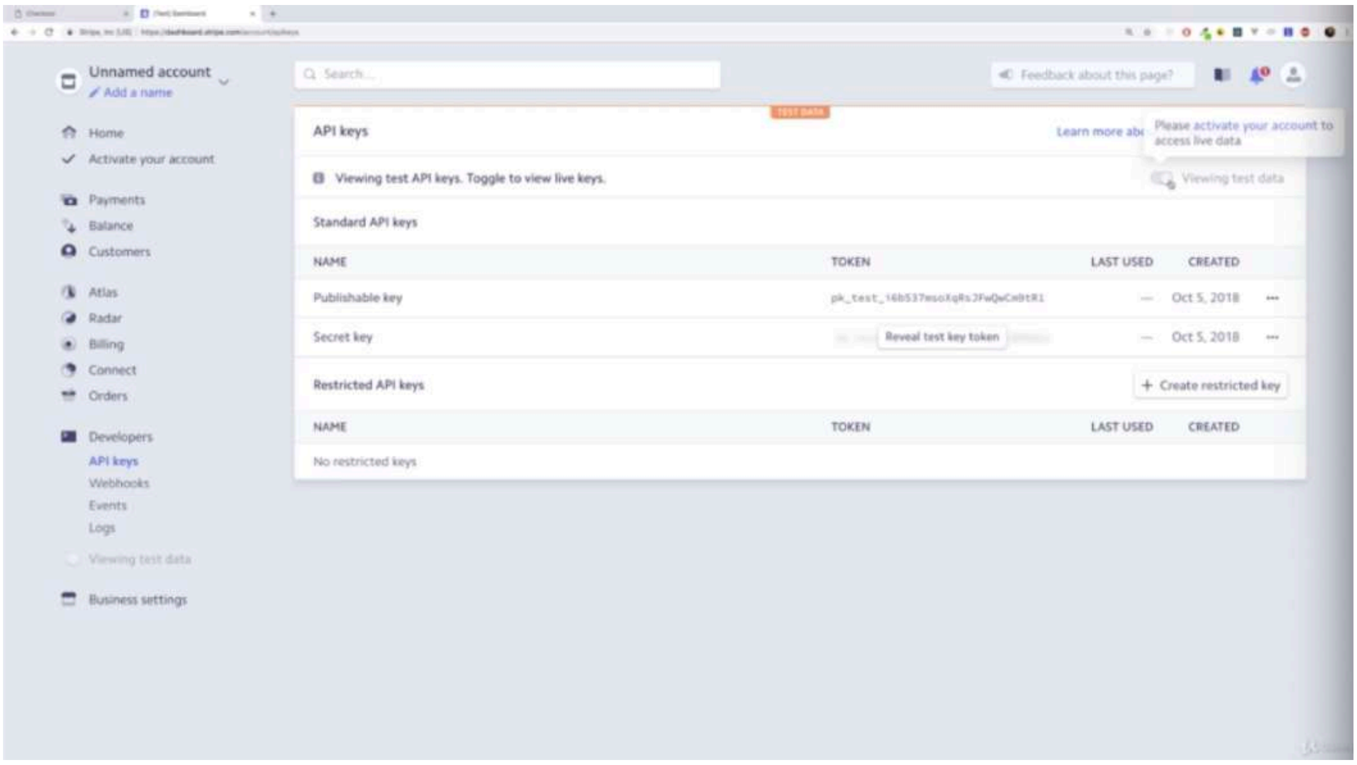

stripe
Create your Stripe account
Email
Full name
Password
Confirm password
Create your Stripe account
Already have an account? Sign in
Forgot your password? · Privacy Policy · Contact · © Stripe



Verify your email
Please verify your email before activating your account.

————————————————————————

![](images/352-using-stripe-in-your-app-4.png)



API keys

Viewing test API keys. Toggle to view live keys.

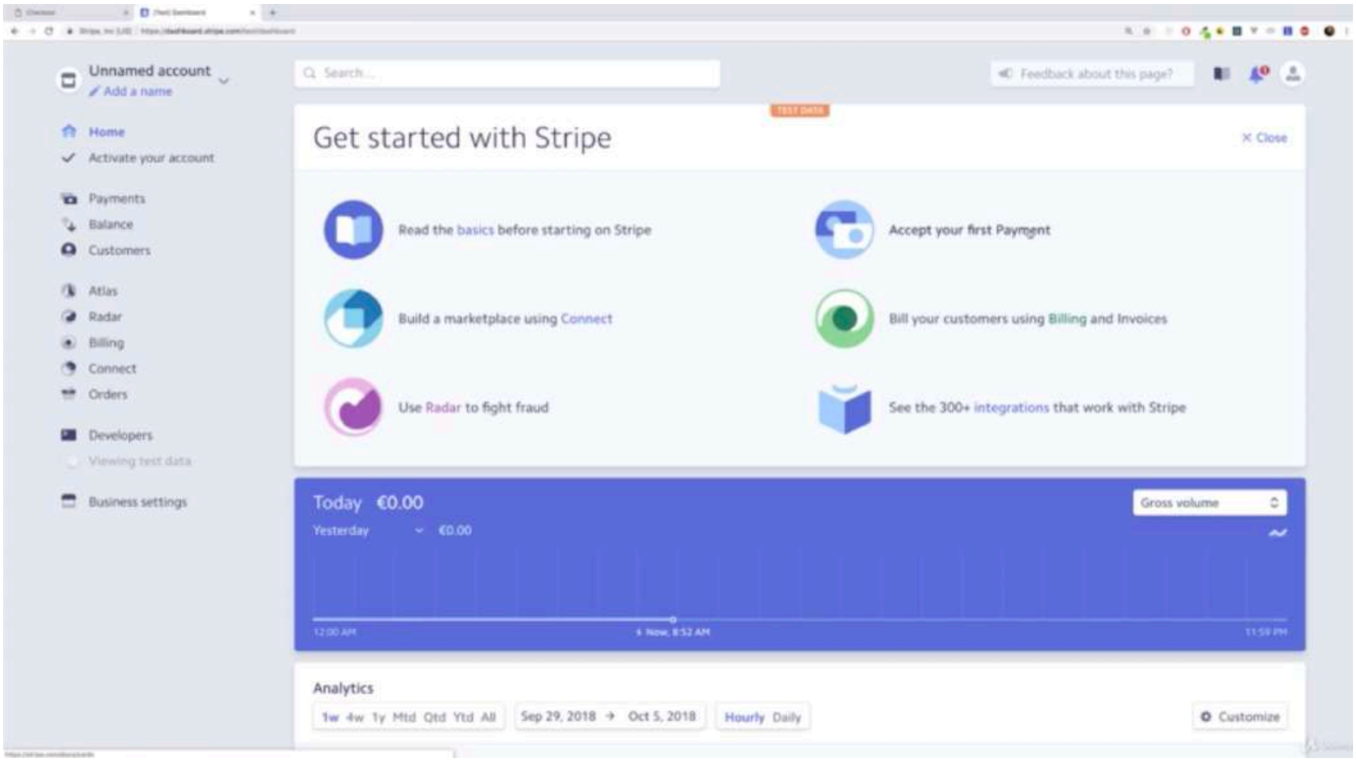
Standard API keys

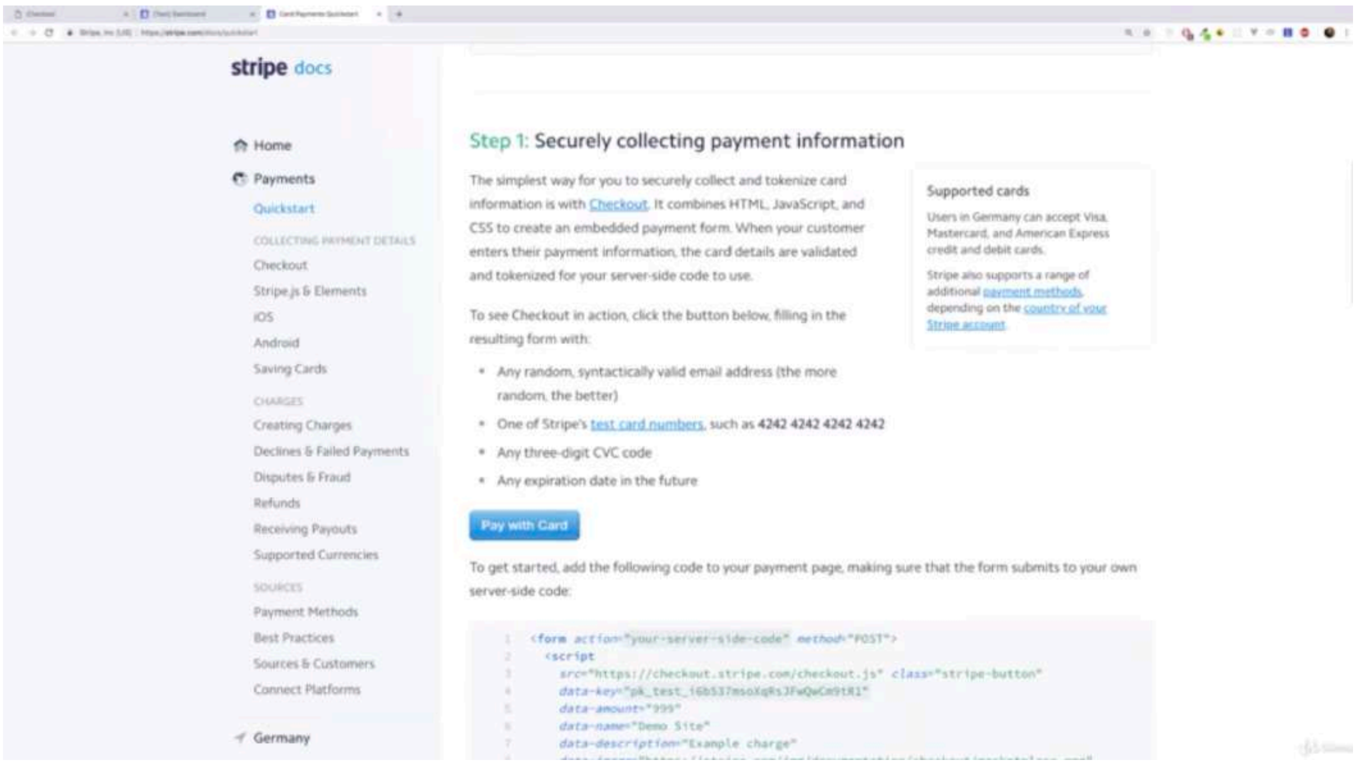
NAME	TOKEN	LAST USED	CREATED
Publishable key	pk_test_i6b537msoXqRsJFwQwCm0tR1	—	Oct 5, 2018
Secret key	Reveal test key token	—	Oct 5, 2018

Restricted API keys

NAME	TOKEN	LAST USED	CREATED
No restricted keys			

- if you wanna build a real application and you wanna push it to the production, you would switch to your live data here. for this, you have to act with your account. we will not do that here.
![](images/352-using-stripe-in-your-app-5.png)
![](images/352-using-stripe-in-your-app-6.png)





Step 1: Securely collecting payment information

The simplest way for you to securely collect and tokenize card information is with Checkout. It combines HTML, JavaScript, and CSS to create an embedded payment form. When your customer enters their payment information, the card details are validated and tokenized for your server-side code to use.

To see Checkout in action, click the button below, filling in the resulting form with:

- Any random, syntactically valid email address (the more random, the better)
- One of Stripe's test card numbers, such as **4242 4242 4242 4242**
- Any three-digit CVC code
- Any expiration date in the future

Supported cards

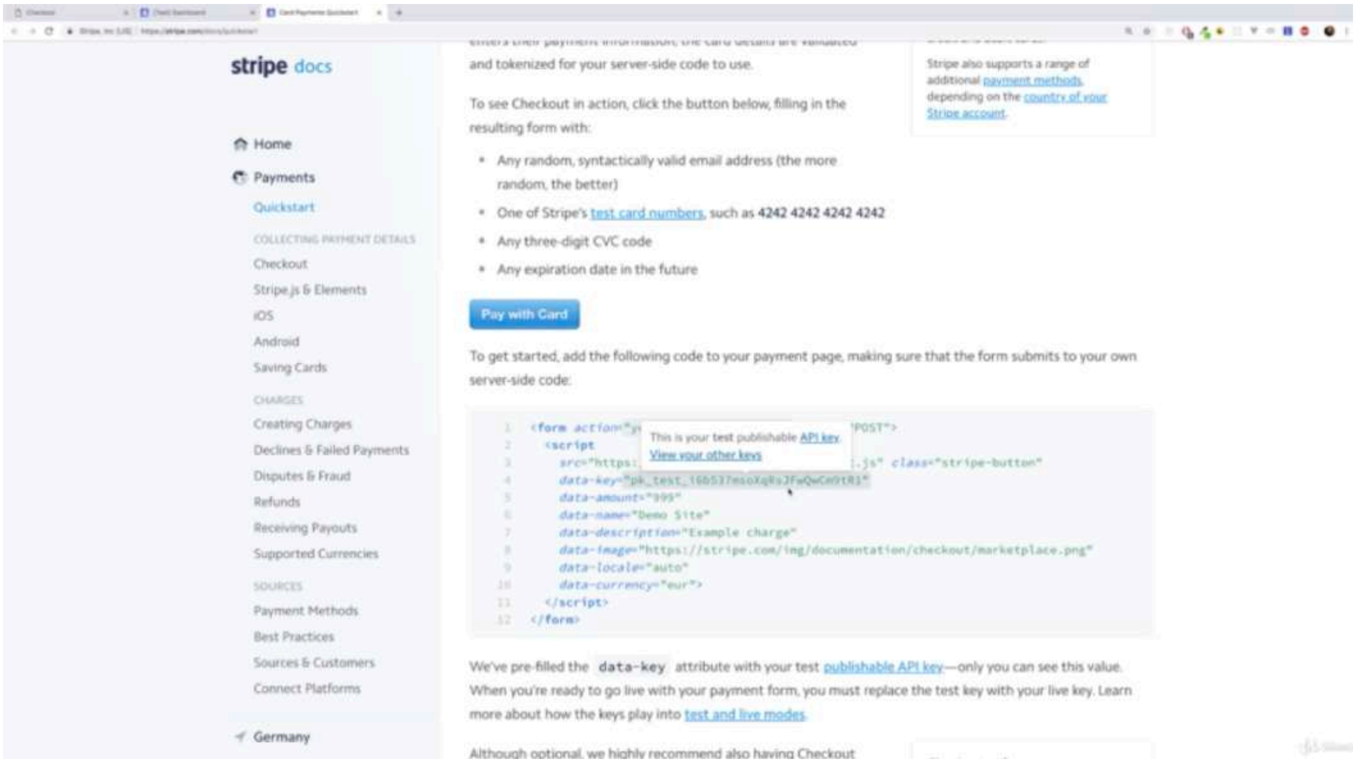
Users in Germany can accept Visa, Mastercard, and American Express credit and debit cards.

Stripe also supports a range of additional payment methods, depending on the country of your Stripe account.

To get started, add the following code to your payment page, making sure that the form submits to your own server-side code:

```
<form action="your-server-side-code" method="POST">
  <script
    src="https://checkout.stripe.com/checkout.js" class="stripe-button"
    data-key="pk_test_i6b537msoXqRsJFwQwCm9tR1"
    data-amount="999"
    data-name="Demo Site"
    data-description="Example charge"
```

- go to 'Home' and click 'Accept your first Payment'. and then official Docs and learn how we can
![](images/352-using-stripe-in-your-app-7.png)



and tokenized for your server-side code to use.

To see Checkout in action, click the button below, filling in the resulting form with:

- Any random, syntactically valid email address (the more random, the better)
- One of Stripe's test card numbers, such as **4242 4242 4242 4242**
- Any three-digit CVC code
- Any expiration date in the future

Stripe also supports a range of additional payment methods, depending on the country of your Stripe account.

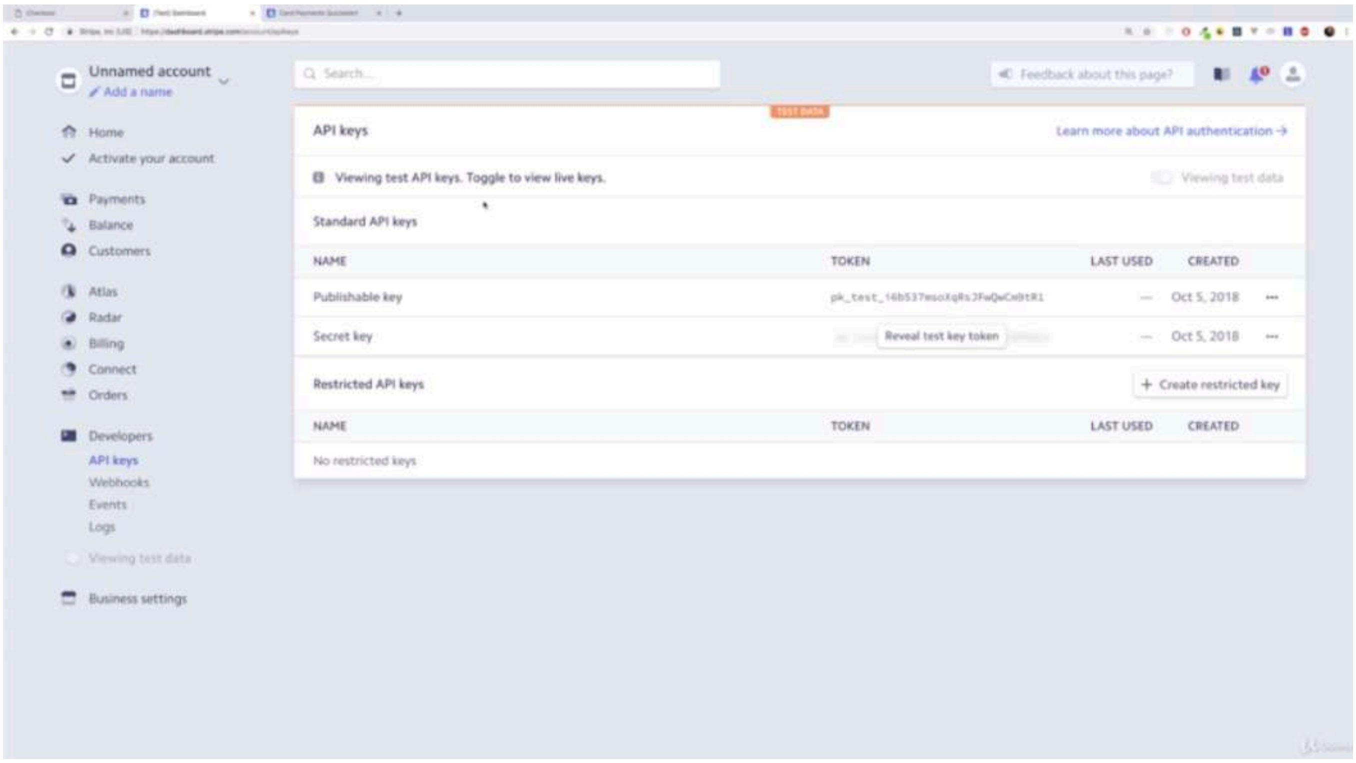
To get started, add the following code to your payment page, making sure that the form submits to your own server-side code:

```
<form action="y                                      "POST">
  <script
    src="https:                                  .js" class="stripe-button"
    data-key="pk_test_i6b537msoXqRsJFwQwCm9tR1"
    data-amount="999"
    data-name="Demo Site"
    data-description="Example charge"
    data-image="https://stripe.com/img/documentation/checkout/marketplace.png"
    data-locale="auto"
    data-currency="eur">
  </script>
</form>
```

We've pre-filled the `data-key` attribute with your test publishable API key—only you can see this value. When you're ready to go live with your payment form, you must replace the test key with your live key. Learn more about how the keys play into test and live modes.

Although optional, we highly recommend also having Checkout

- the coll thing is that this data here is already populated with our data, with our publishable key for example,
![](images/352-using-stripe-in-your-app-8.png)



API keys

Viewing test API keys. Toggle to view live keys.

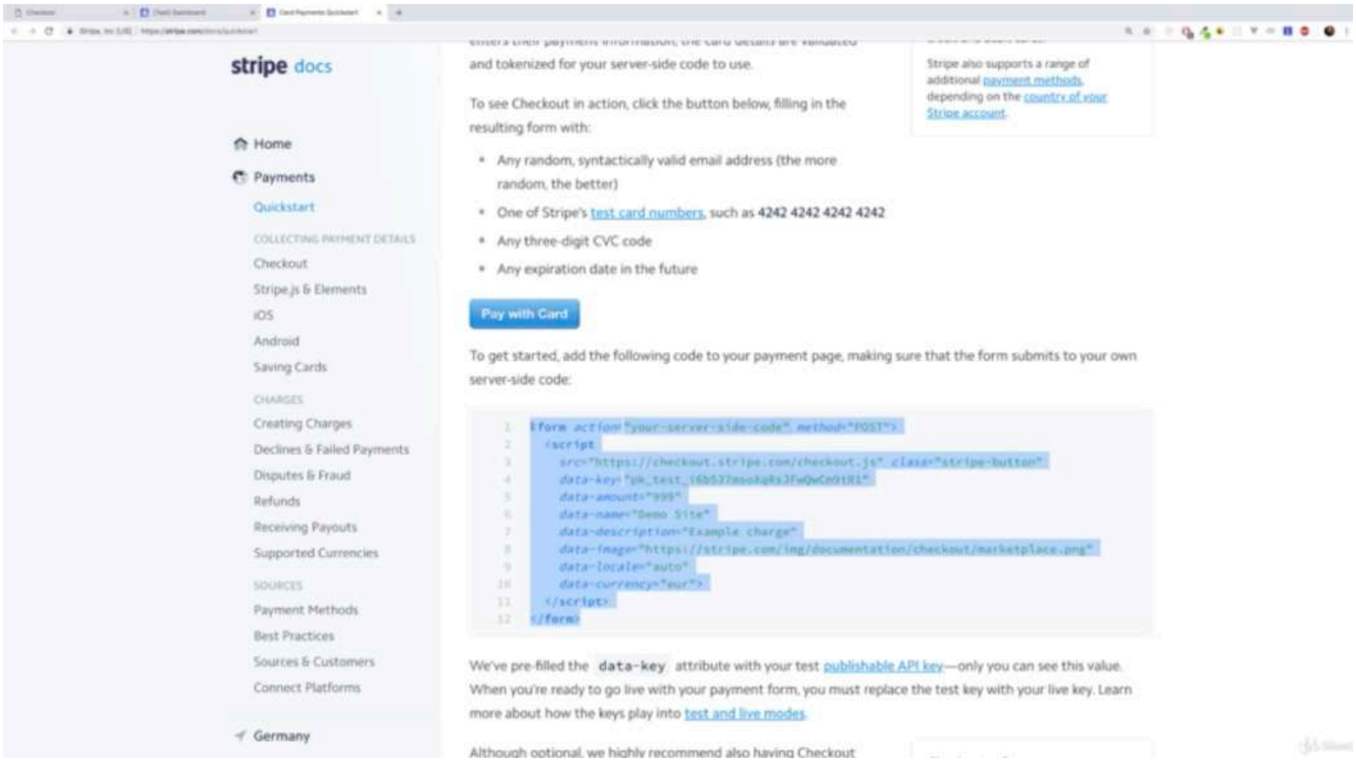
Standard API keys

NAME	TOKEN	LAST USED	CREATED
Publishable key	pk_test_16b537msoXqRsJFwQwCm9tR1	—	Oct 5, 2018
Secret key		—	Oct 5, 2018

Restricted API keys

NAME	TOKEN	LAST USED	CREATED
No restricted keys			

- that key you saw under developers API keys, that are keys you need to send with your data to let stripe know to which account this belongs.
![](images/352-using-stripe-in-your-app-9.png)
![](images/352-using-stripe-in-your-app-10.png)



and tokenized for your server-side code to use.

To see Checkout in action, click the button below, filling in the resulting form with:

- Any random, syntactically valid email address (the more random, the better)
- One of Stripe's test card numbers, such as 4242 4242 4242 4242
- Any three-digit CVC code
- Any expiration date in the future

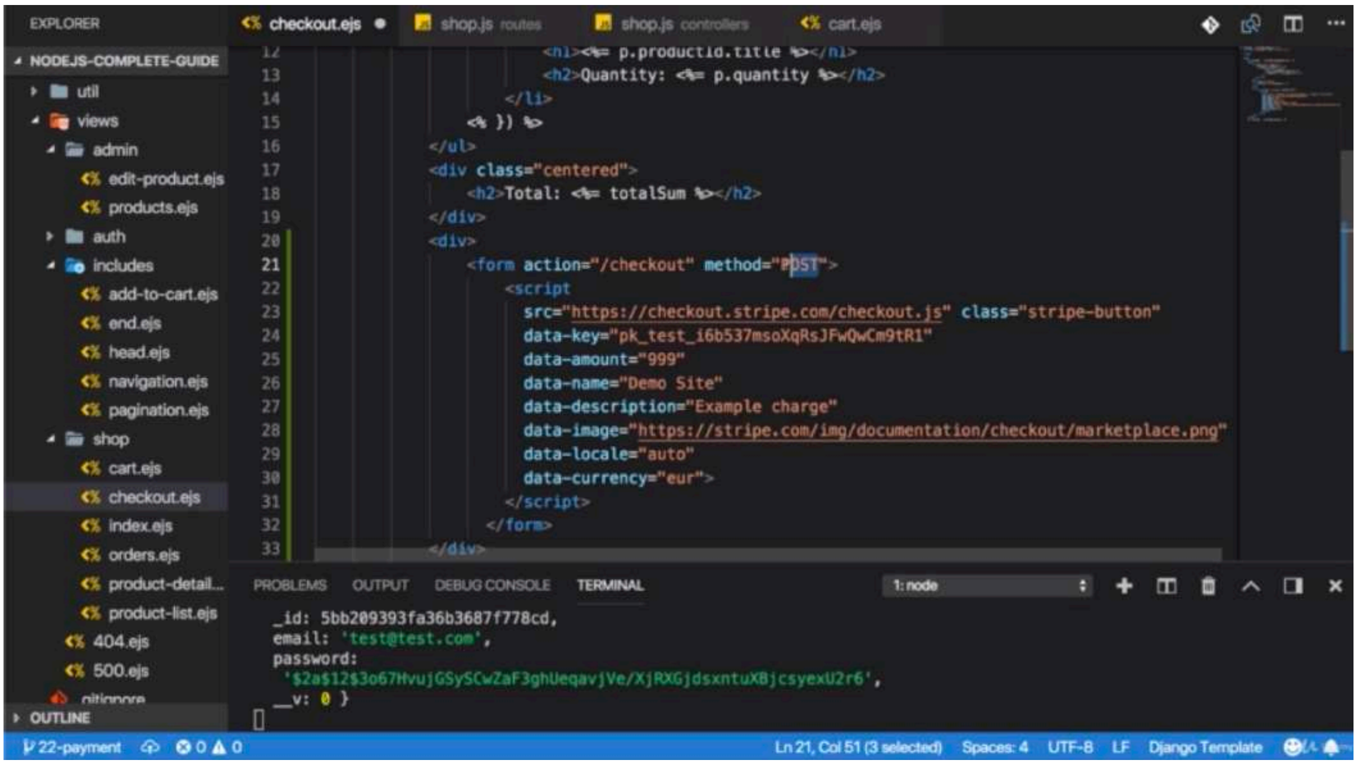
Stripe also supports a range of additional payment methods, depending on the country of your Stripe account.

To get started, add the following code to your payment page, making sure that the form submits to your own server-side code:

```
<form action="your-server-side-code" method="POST">
  <script
    src="https://checkout.stripe.com/checkout.js" class="stripe-button"
    data-key="pk_test_16b537msoXqRsJFwQwCm9tR1"
    data-amount="999"
    data-name="Demo Site"
    data-description="Example charge"
    data-image="https://stripe.com/img/documentation/checkout/marketplace.png"
    data-locale="auto"
    data-currency="eur">
  </script>
</form>
```

We've pre-filled the `data-key` attribute with your test publishable API key—only you can see this value. When you're ready to go live with your payment form, you must replace the test key with your live key. Learn more about how the keys play into test and live modes.

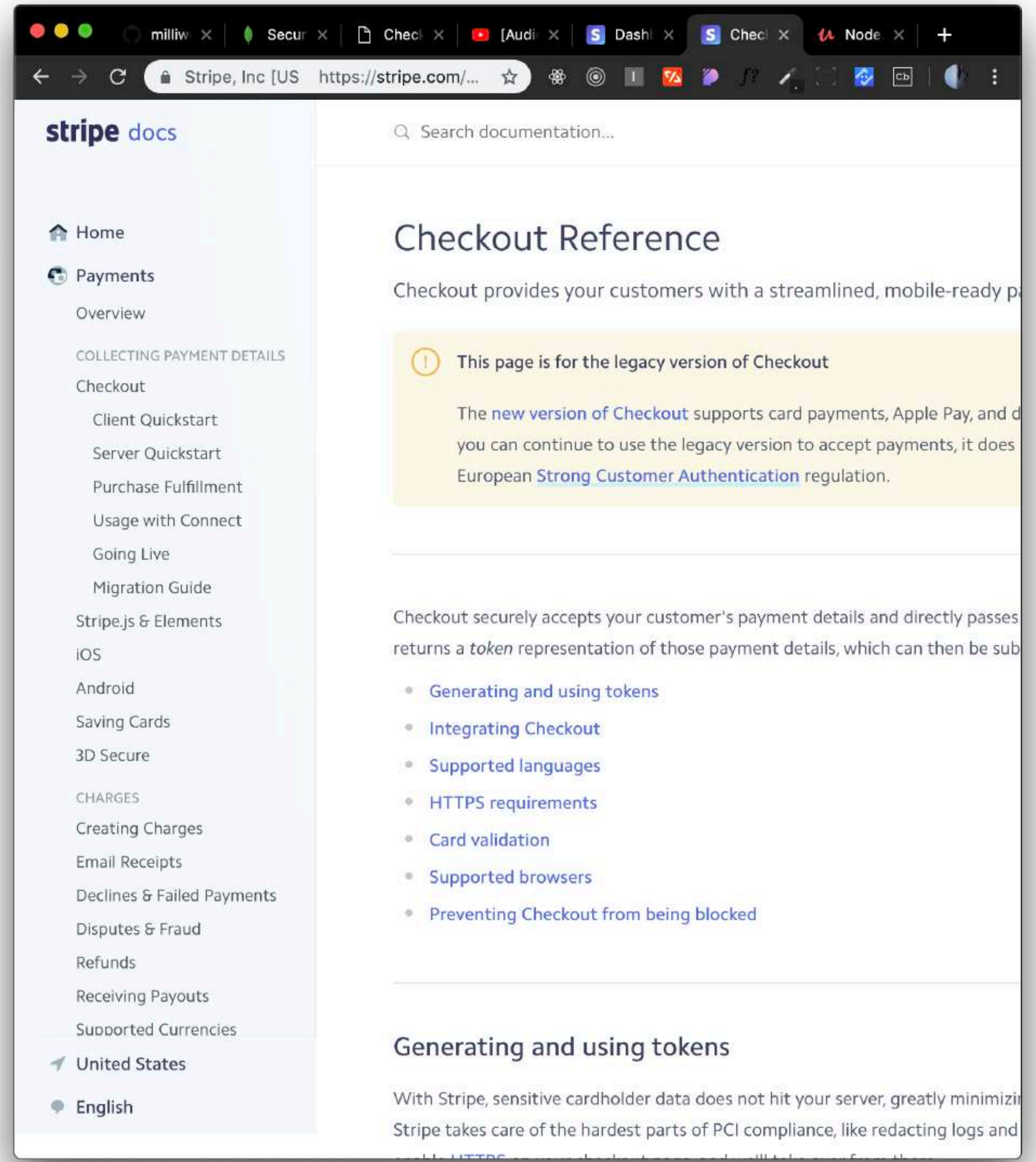
Although optional, we highly recommend also having Checkout



```html
                <h1><%= p.productId.title %></h1>
                <h2>Quantity: <%= p.quantity %></h2>
            </li>
        <% }) %>
    </ul>
    <div class="centered">
        <h2>Total: <%= totalSum %></h2>
    </div>
    <div>
        <form action="/checkout" method="POST">
            <script
              src="https://checkout.stripe.com/checkout.js" class="stripe-button"
              data-key="pk_test_i6b537msoXqRsJFwQwCm9tR1"
              data-amount="999"
              data-name="Demo Site"
              data-description="Example charge"
              data-image="https://stripe.com/img/documentation/checkout/marketplace.png"
              data-locale="auto"
              data-currency="eur">
            </script>
          </form>
    </div>
```

```
  _id: 5bb209393fa36b3687f778cd,
  email: 'test@test.com',
  password:
   '$2a$12$3o67HvujGSySCwZaF3ghUeqavjVe/XjRXGjdsxntuXBjcsyexU2r6',
  __v: 0 }
```

- a code snippet we can already use and we can add that to our ./views/shop/checkout.ejs file
![](images/352-using-stripe-in-your-app-11.png)
![](images/352-using-stripe-in-your-app-12.png)



Checkout Reference

Checkout provides your customers with a streamlined, mobile-ready p

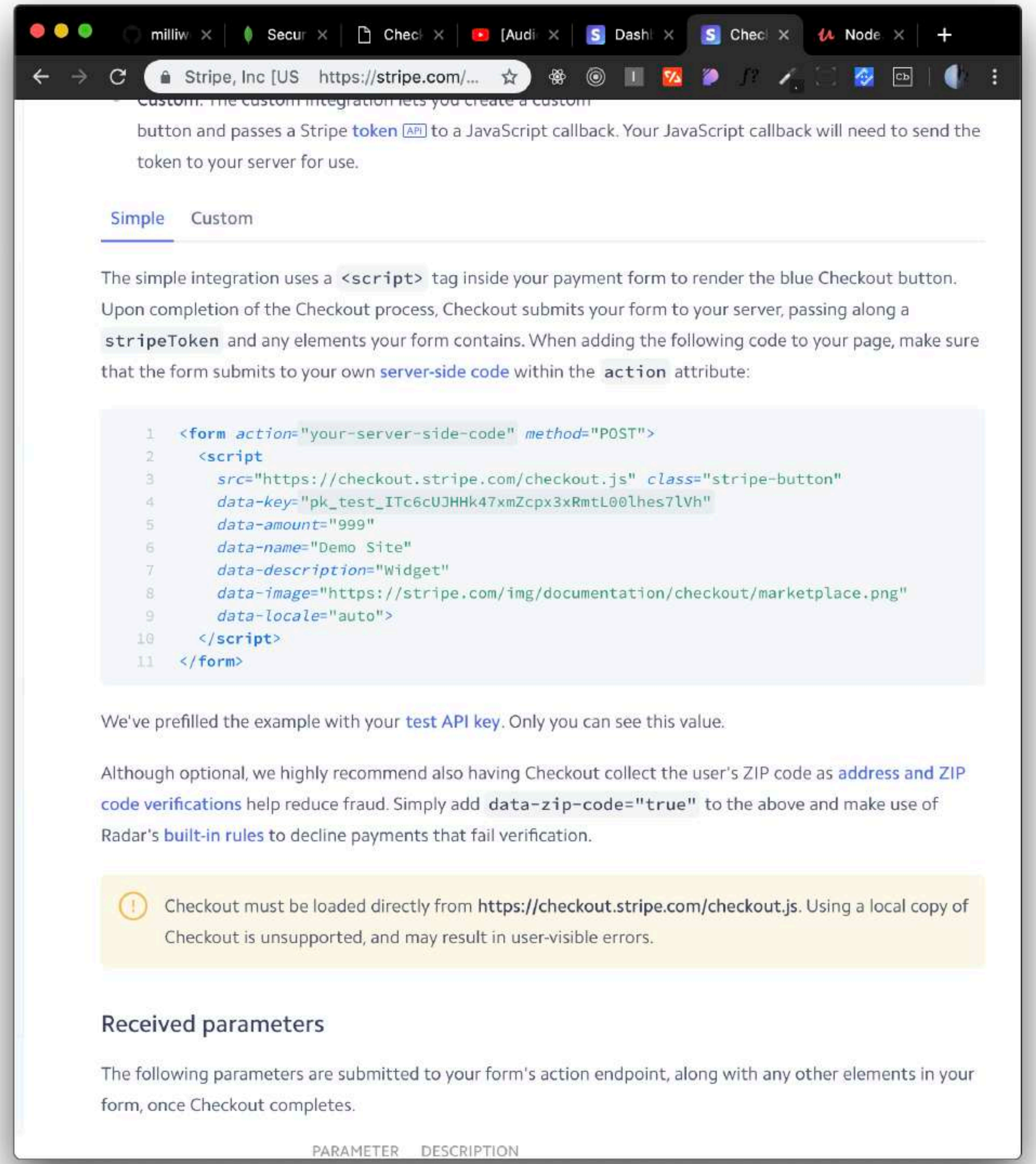
This page is for the legacy version of Checkout

The new version of Checkout supports card payments, Apple Pay, and d
you can continue to use the legacy version to accept payments, it does
European Strong Customer Authentication regulation.

Checkout securely accepts your customer's payment details and directly passes
returns a *token* representation of those payment details, which can then be sub

Generating and using tokens

With Stripe, sensitive cardholder data does not hit your server, greatly minimizi
Stripe takes care of the hardest parts of PCI compliance, like redacting logs and



Custom: The custom integration lets you create a custom button and passes a Stripe token to a JavaScript callback. Your JavaScript callback will need to send the token to your server for use.

Simple Custom

The simple integration uses a `<script>` tag inside your payment form to render the blue Checkout button. Upon completion of the Checkout process, Checkout submits your form to your server, passing along a `stripeToken` and any elements your form contains. When adding the following code to your page, make sure that the form submits to your own server-side code within the `action` attribute:

```
<form action="your-server-side-code" method="POST">
  <script
    src="https://checkout.stripe.com/checkout.js" class="stripe-button"
    data-key="pk_test_ITc6cUJHHk47xmZcpx3xRmtL00lhes7lVh"
    data-amount="999"
    data-name="Demo Site"
    data-description="Widget"
    data-image="https://stripe.com/img/documentation/checkout/marketplace.png"
    data-locale="auto">
  </script>
</form>
```

We've prefilled the example with your test API key. Only you can see this value.

Although optional, we highly recommend also having Checkout collect the user's ZIP code as address and ZIP code verifications help reduce fraud. Simply add `data-zip-code="true"` to the above and make use of Radar's built-in rules to decline payments that fail verification.

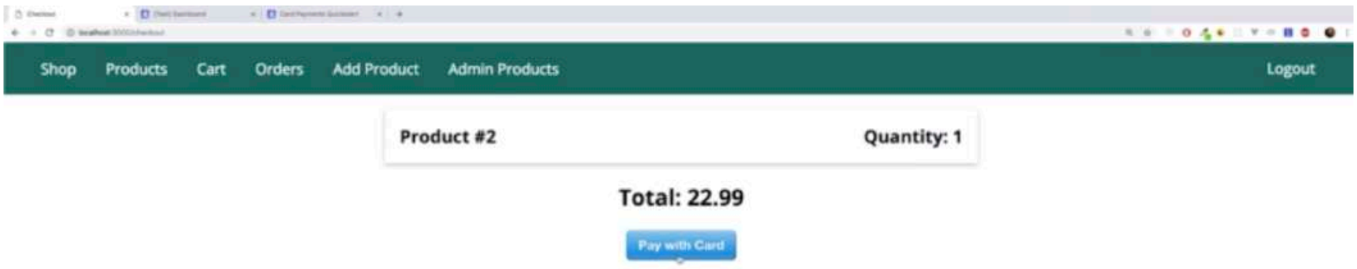
Checkout must be loaded directly from **https://checkout.stripe.com/checkout.js**. Using a local copy of Checkout is unsupported, and may result in user-visible errors.

Received parameters

The following parameters are submitted to your form's action endpoint, along with any other elements in your form, once Checkout completes.

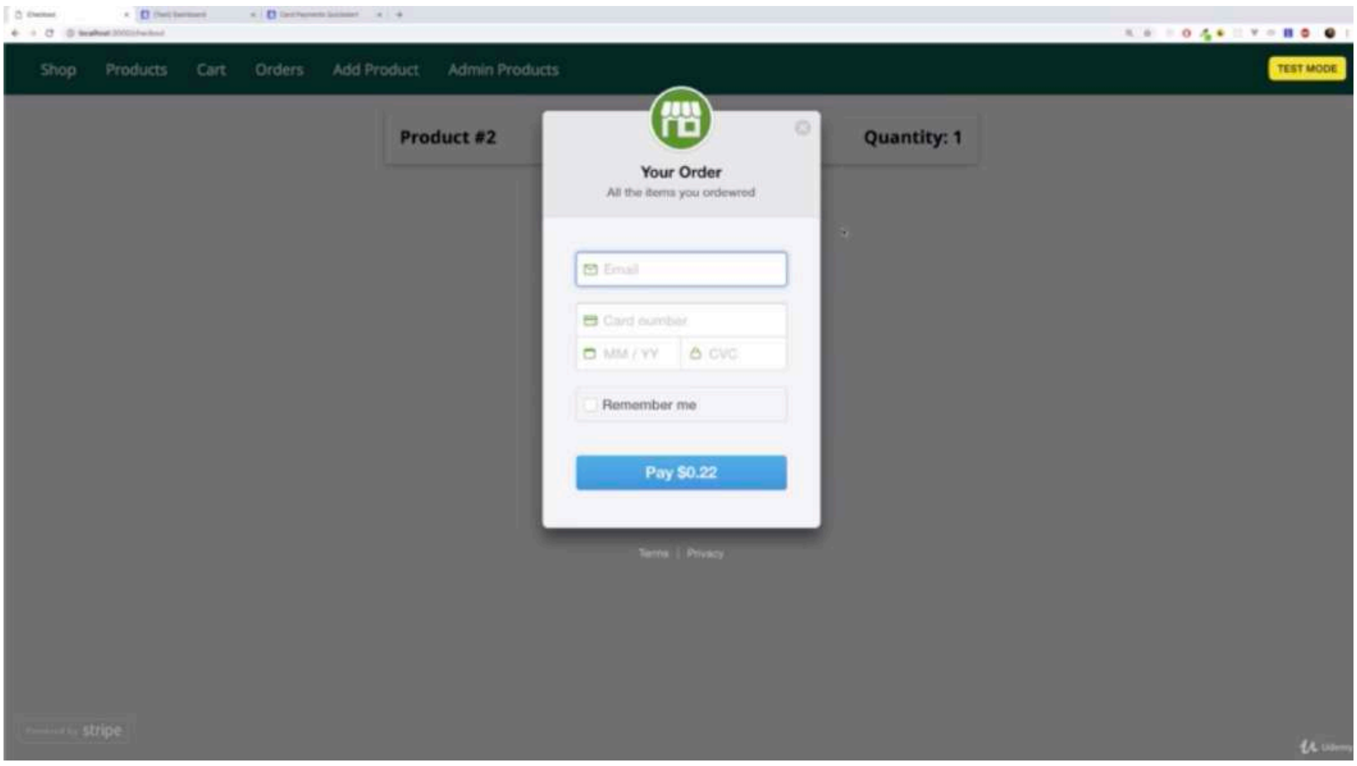
PARAMETER DESCRIPTION

- update version is above. UI is changed.
![](images/352-using-stripe-in-your-app-13.png)
![](images/352-using-stripe-in-your-app-14.png)
![](images/352-using-stripe-in-your-app-15.png)
![](images/352-using-stripe-in-your-app-16.png)
![](images/352-using-stripe-in-your-app-17.png)
![](images/352-using-stripe-in-your-app-18.png)
![](images/352-using-stripe-in-your-app-19.png)
![](images/352-using-stripe-in-your-app-20.png)



Product #2 Quantity: 1

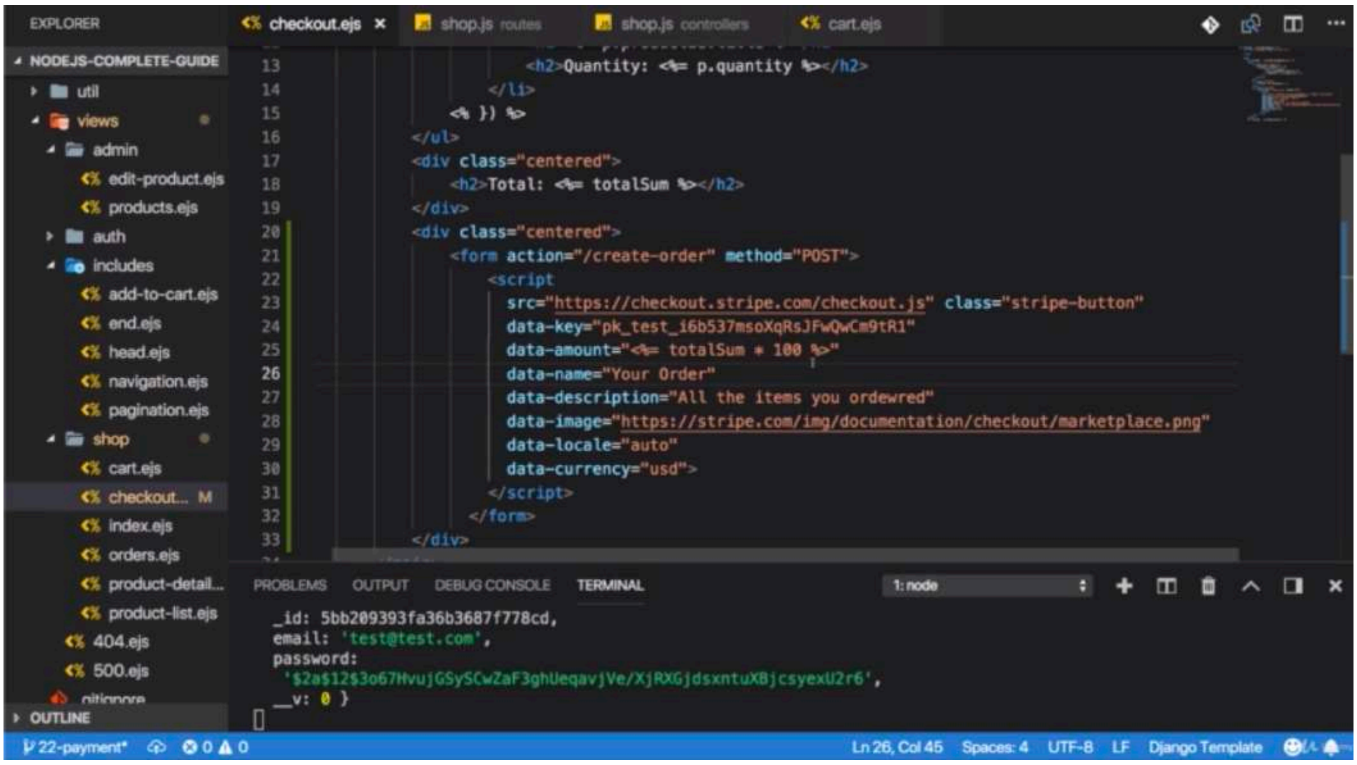
Total: 22.99



Product #2 Quantity: 1

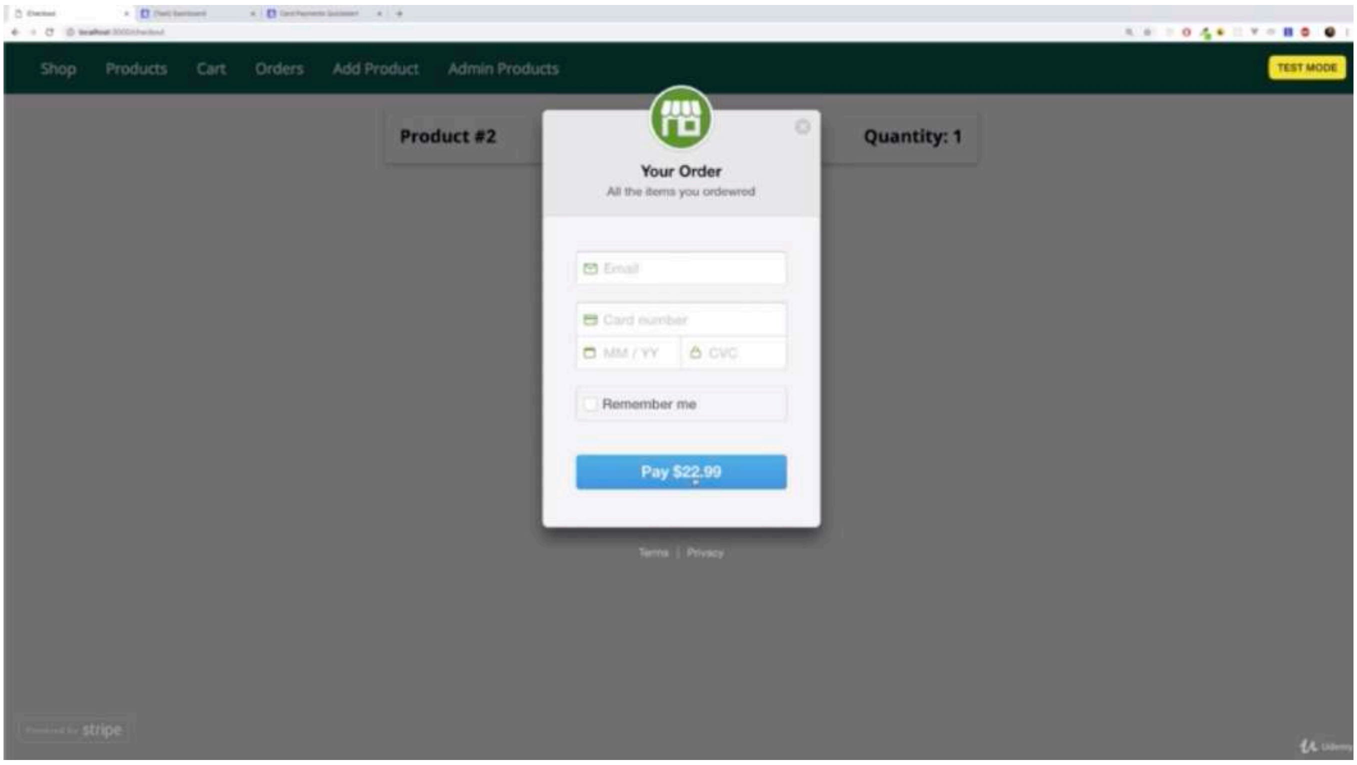
Your Order

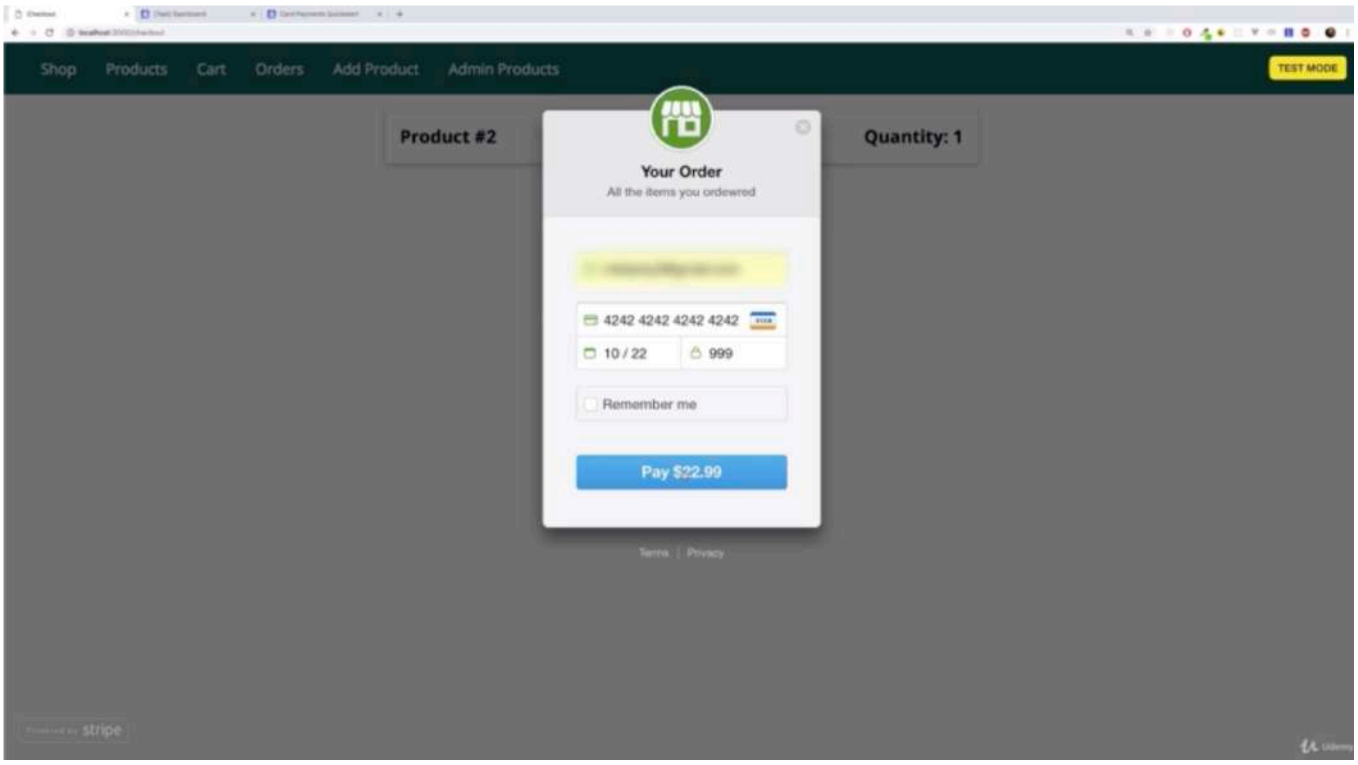
All the items you ordewred

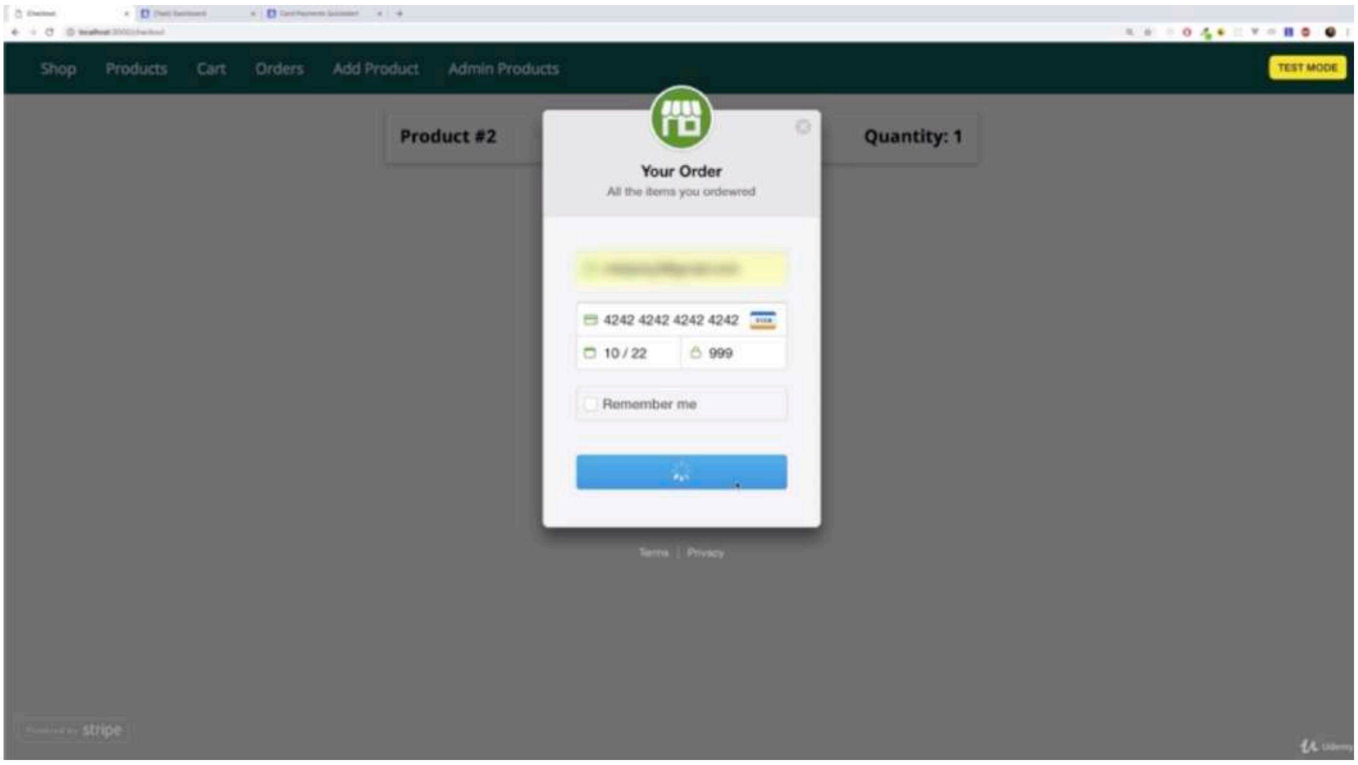


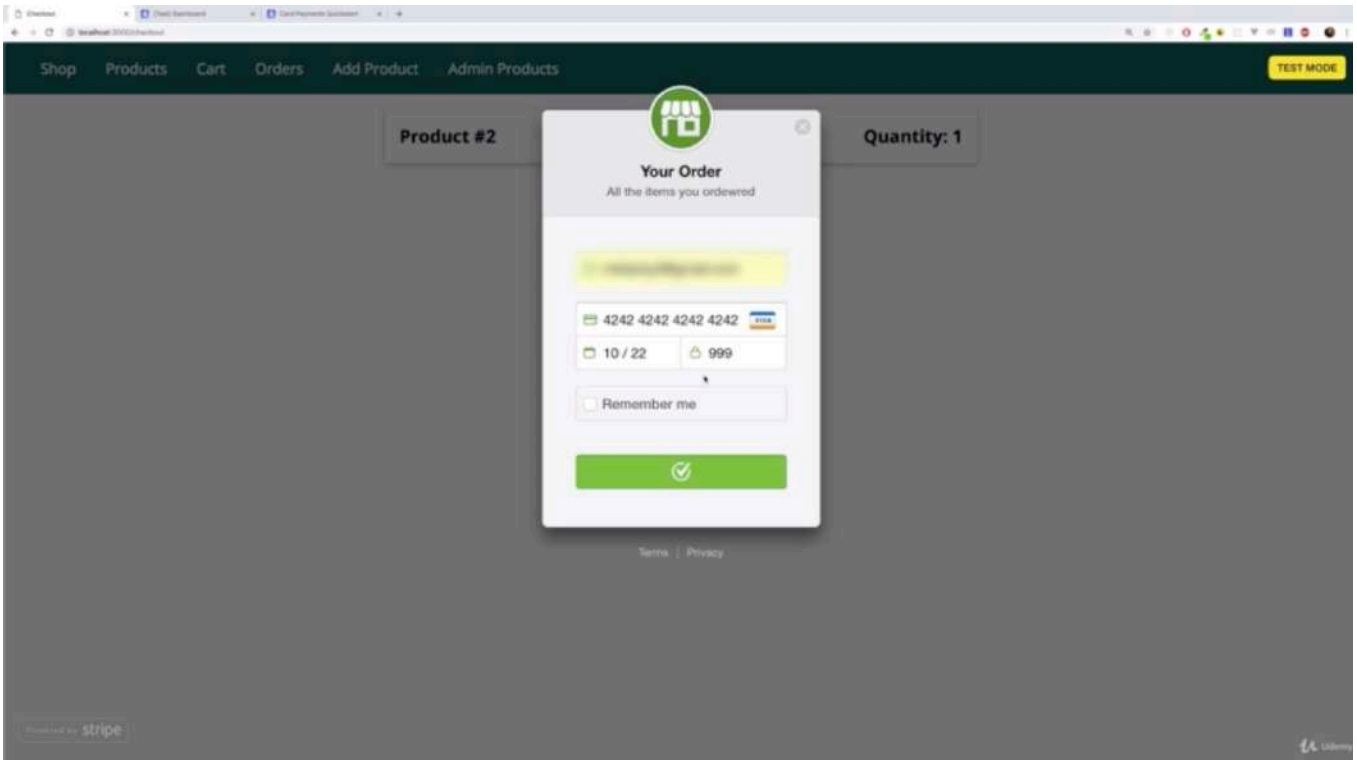
```
                <h2>Quantity: <%= p.quantity %></h2>
            </li>
        <% }) %>
    </ul>
    <div class="centered">
        <h2>Total: <%= totalSum %></h2>
    </div>
    <div class="centered">
        <form action="/create-order" method="POST">
            <script
              src="https://checkout.stripe.com/checkout.js" class="stripe-button"
              data-key="pk_test_i6b537msoXqRsJFwQwCm9tR1"
              data-amount="<%= totalSum * 100 %>"
              data-name="Your Order"
              data-description="All the items you ordewred"
              data-image="https://stripe.com/img/documentation/checkout/marketplace.png"
              data-locale="auto"
              data-currency="usd">
            </script>
          </form>
    </div>
```

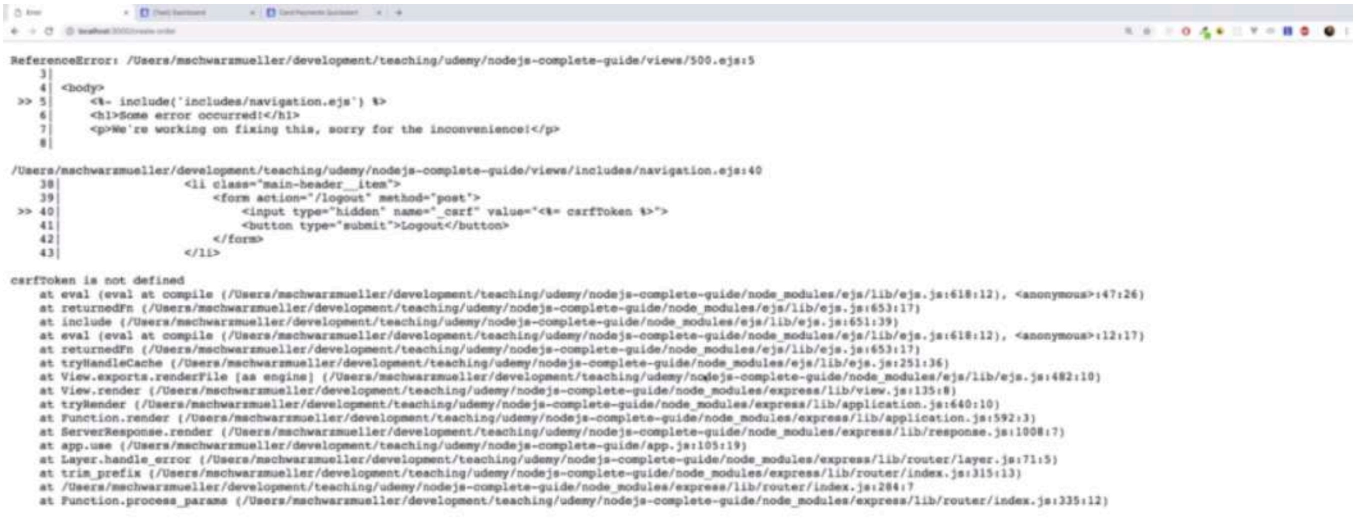
```
  _id: 5bb209393fa36b3687f778cd,
  email: 'test@test.com',
  password:
   '$2a$12$3o67HvujGSySCwZaF3ghUeqavjVe/XjRXGjdsxntuXBjcsyexU2r6',
  __v: 0 }
```




Shop
Products
Cart
Orders
Add Product
Admin Products
TEST MODE
Product #2
Quantity: 1
Your Order
All the items you ordewred
4242 4242 4242 4242
10 / 22
999
Remember me
Pay $22.99
Terms | Privacy
stripe

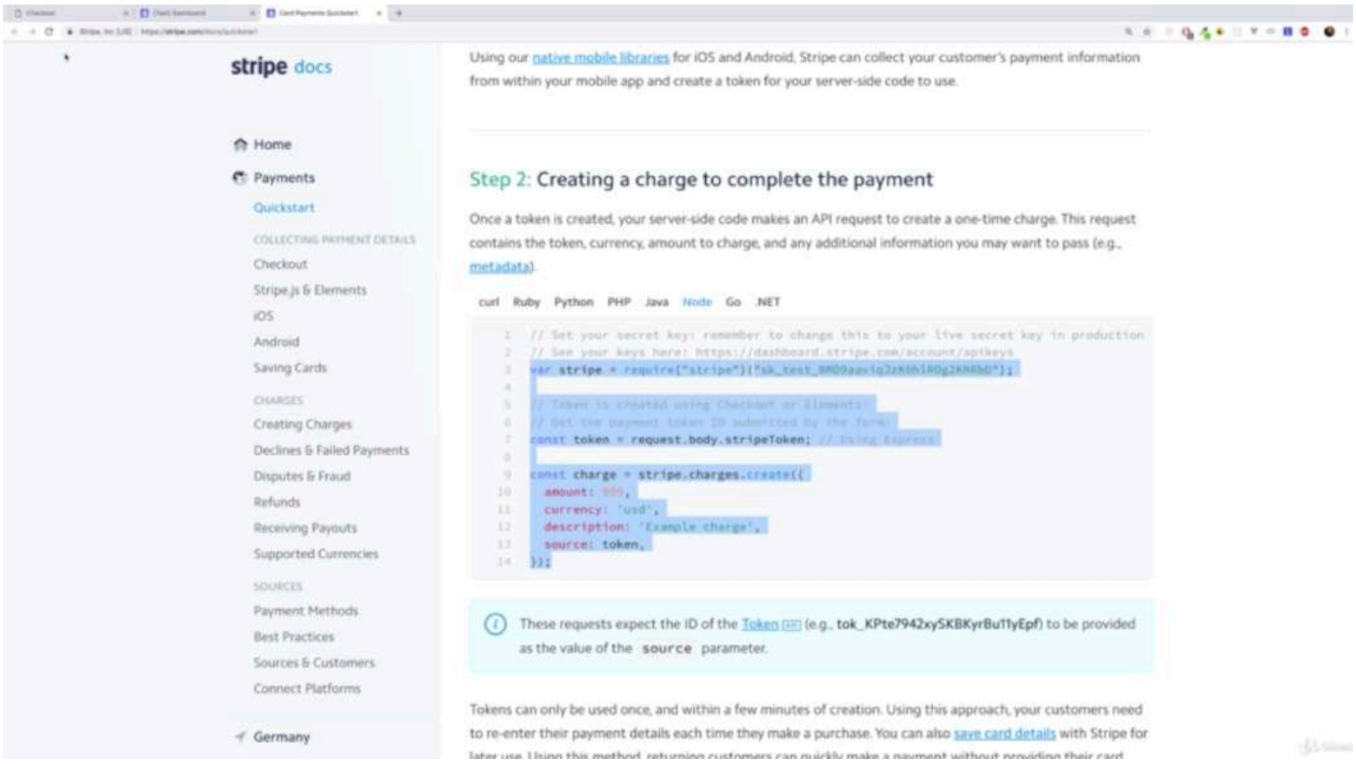

Shop
Products
Cart
Orders
Add Product
Admin Products
TEST MODE
Product #2
Quantity: 1
Your Order
All the items you ordewred
4242 4242 4242 4242
10 / 22
999
Remember me
Terms | Privacy
stripe





```
ReferenceError: /Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/views/500.ejs:5
    3|
    4| <body>
 >> 5|     <%- include('includes/navigation.ejs') %>
    6|     <h1>Some error occurred!</h1>
    7|     <p>We're working on fixing this, sorry for the inconvenience!</p>
    8|

/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/views/includes/navigation.ejs:40
    38|                 <li class="main-header__item">
    39|                     <form action="/logout" method="post">
 >> 40|                         <input type="hidden" name="_csrf" value="<%= csrfToken %>">
    41|                         <button type="submit">Logout</button>
    42|                     </form>
    43|                 </li>

csrfToken is not defined
    at eval (eval at compile (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/node_modules/ejs/lib/ejs.js:618:12), <anonymous>:47:26)
    at returnedFn (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/node_modules/ejs/lib/ejs.js:653:17)
    at include (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/node_modules/ejs/lib/ejs.js:651:39)
    at eval (eval at compile (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/node_modules/ejs/lib/ejs.js:618:12), <anonymous>:12:17)
    at returnedFn (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/node_modules/ejs/lib/ejs.js:653:17)
    at tryHandleCache (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/node_modules/ejs/lib/ejs.js:251:36)
    at View.exports.renderFile [as engine] (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/node_modules/ejs/lib/ejs.js:482:10)
    at View.render (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/node_modules/express/lib/view.js:135:8)
    at tryRender (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/node_modules/express/lib/application.js:640:10)
    at Function.render (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/node_modules/express/lib/application.js:592:3)
    at ServerResponse.render (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/node_modules/express/lib/response.js:1008:7)
    at app.use (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/app.js:105:19)
    at Layer.handle_error (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/node_modules/express/lib/router/layer.js:71:5)
    at trim_prefix (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/node_modules/express/lib/router/index.js:315:13)
    at /Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/node_modules/express/lib/router/index.js:284:7
    at Function.process_params (/Users/mschwarzmueller/development/teaching/udemy/nodejs-complete-guide/node_modules/express/lib/router/index.js:335:12)
```

- now this would send a request to the backend, to the create-order route and there we have to continue because now there we will receive some special data and how to continue is something stripe tells you.
![](images/352-using-stripe-in-your-app-21.png)
![](images/352-using-stripe-in-your-app-22.png)



Using our native mobile libraries for iOS and Android, Stripe can collect your customer's payment information from within your mobile app and create a token for your server-side code to use.

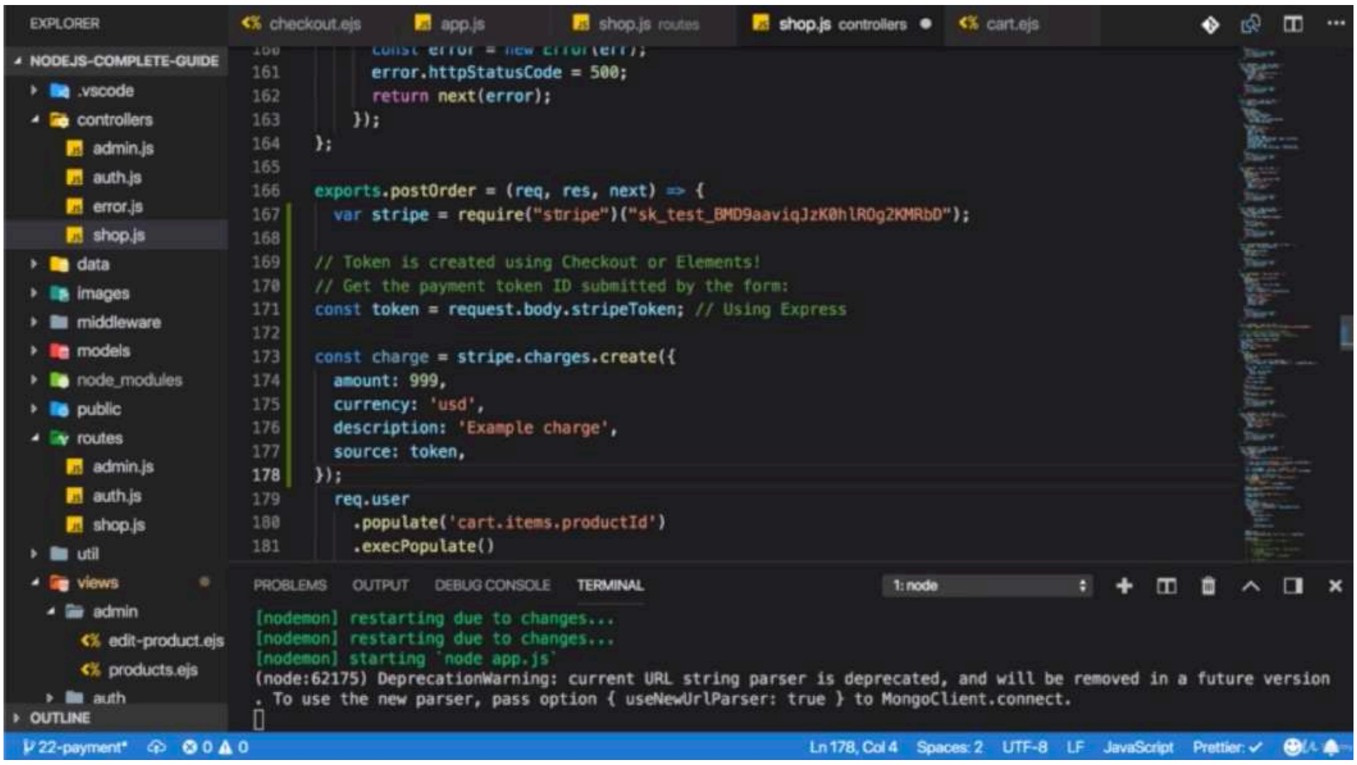
Step 2: Creating a charge to complete the payment

Once a token is created, your server-side code makes an API request to create a one-time charge. This request contains the token, currency, amount to charge, and any additional information you may want to pass (e.g., metadata).

```
// Set your secret key: remember to change this to your live secret key in production
// See your keys here: https://dashboard.stripe.com/account/apikeys
var stripe = require("stripe")("sk_test_BMD9aaviqJzK0hlROg2KMRbD");

// Token is created using Checkout or Elements!
// Get the payment token ID submitted by the form:
const token = request.body.stripeToken; // Using Express

const charge = stripe.charges.create({
  amount: 999,
  currency: 'usd',
  description: 'Example charge',
  source: token,
});
```

These requests expect the ID of the Token (e.g., tok_KPte7942xySKBKyrBu11yEpf) to be provided as the value of the `source` parameter.

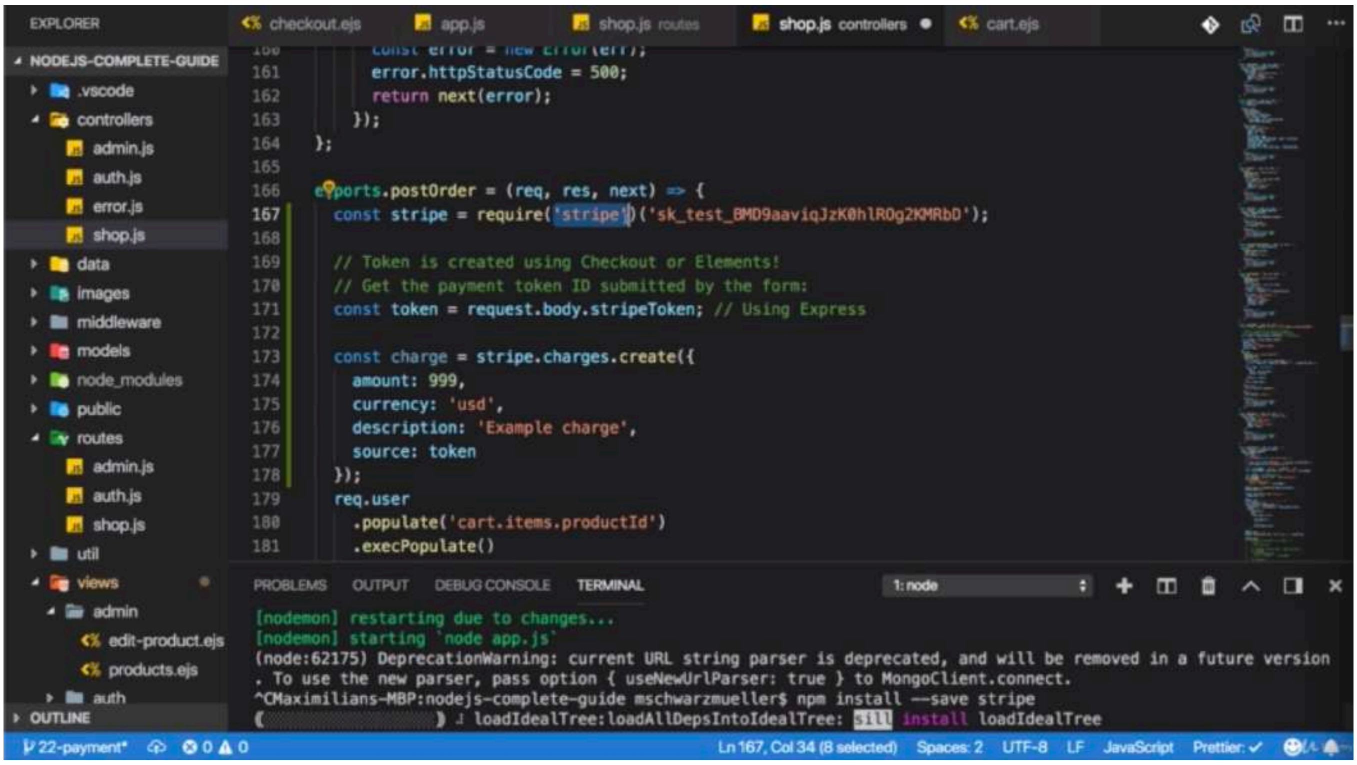
Tokens can only be used once, and within a few minutes of creation. Using this approach, your customers need to re-enter their payment details each time they make a purchase. You can also save card details with Stripe for later use. Using this method, returning customers can quickly make a payment without providing their card



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

exports.postOrder = (req, res, next) => {
  var stripe = require("stripe")("sk_test_BMD9aaviqJzK0hlROg2KMRbD");

// Token is created using Checkout or Elements!
// Get the payment token ID submitted by the form:
const token = request.body.stripeToken; // Using Express

const charge = stripe.charges.create({
  amount: 999,
  currency: 'usd',
  description: 'Example charge',
  source: token,
});
  req.user
    .populate('cart.items.productId')
    .execPopulate()
```

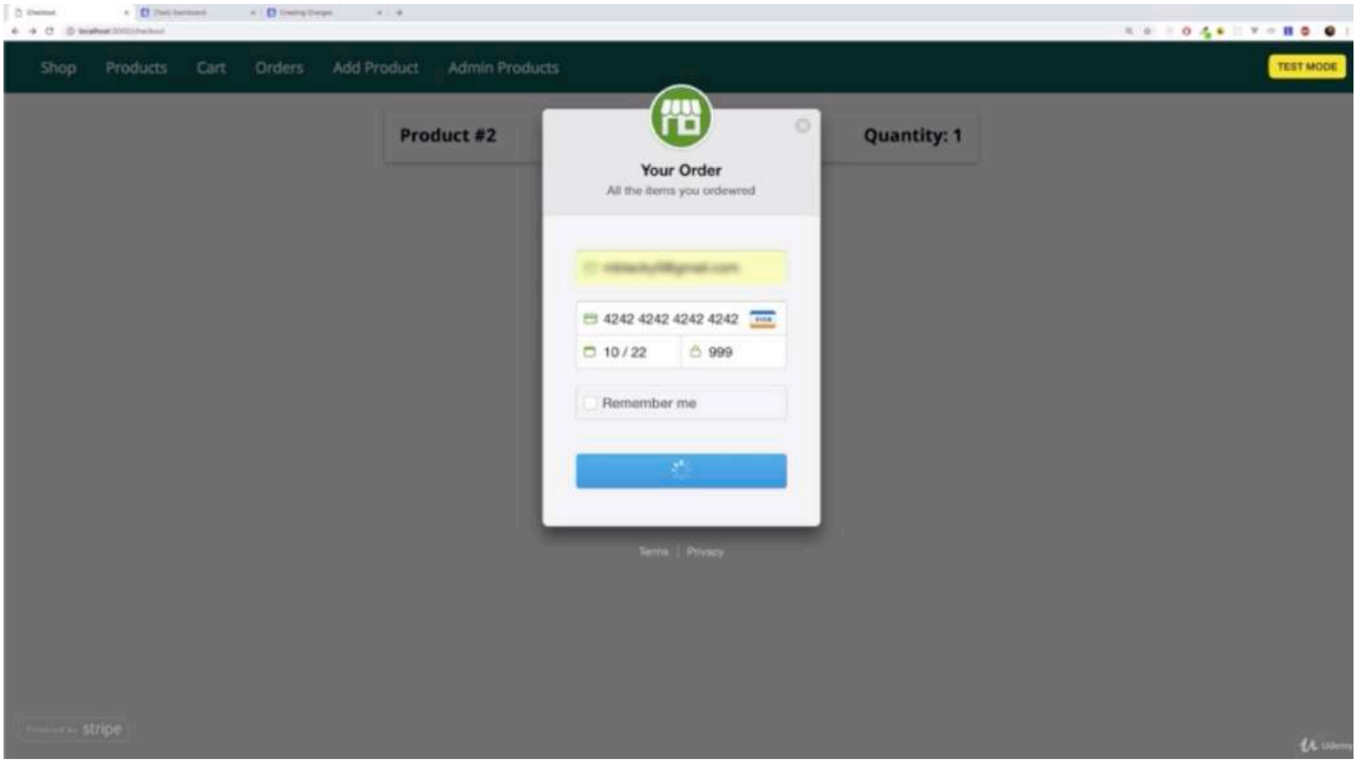
```
[nodemon] restarting due to changes...
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
(node:62175) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version
. To use the new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
```

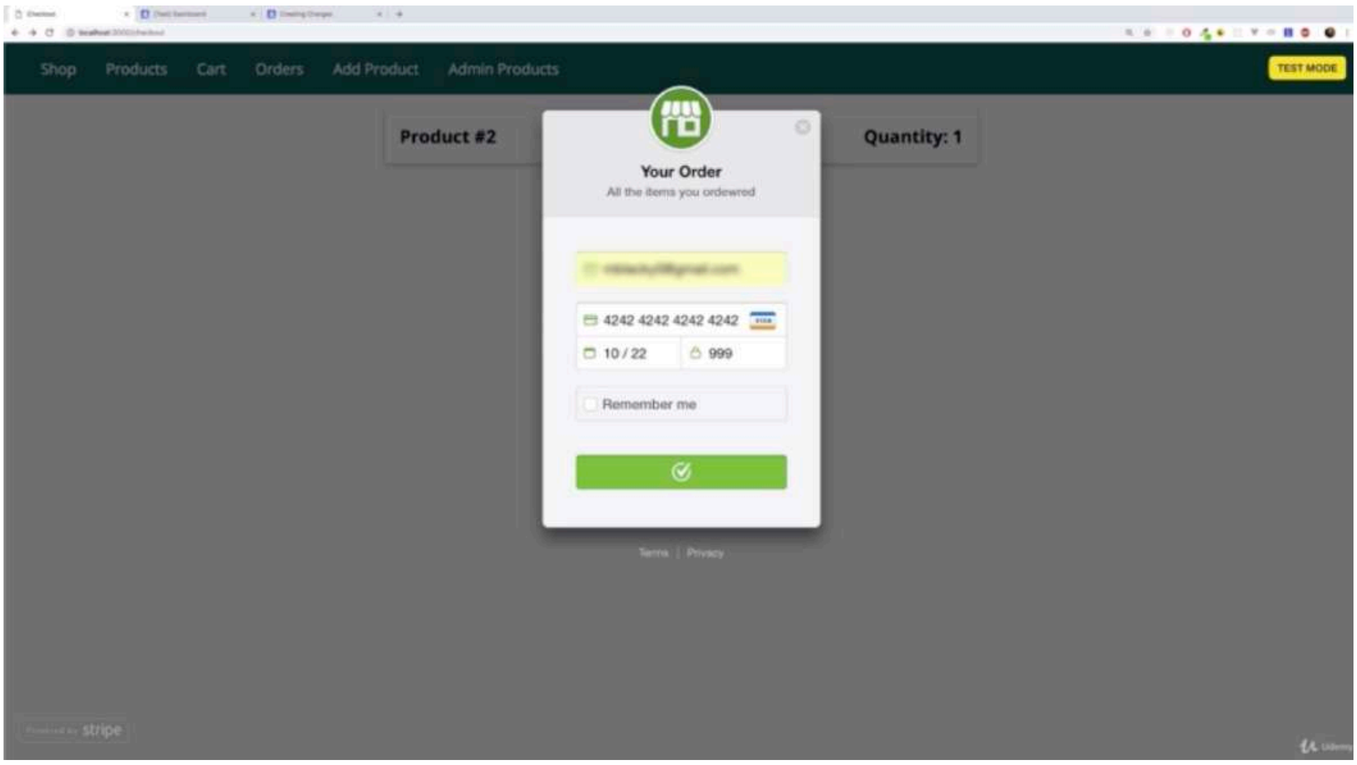
- you can see how to handle the data on the incoming request and you will see that the incoming request will have a stripe token in its body.
![](images/352-using-stripe-in-your-app-23.png)
![](images/352-using-stripe-in-your-app-24.png)
![](images/352-using-stripe-in-your-app-25.png)
![](images/352-using-stripe-in-your-app-26.png)
![](images/352-using-stripe-in-your-app-27.png)
![](images/352-using-stripe-in-your-app-28.png)

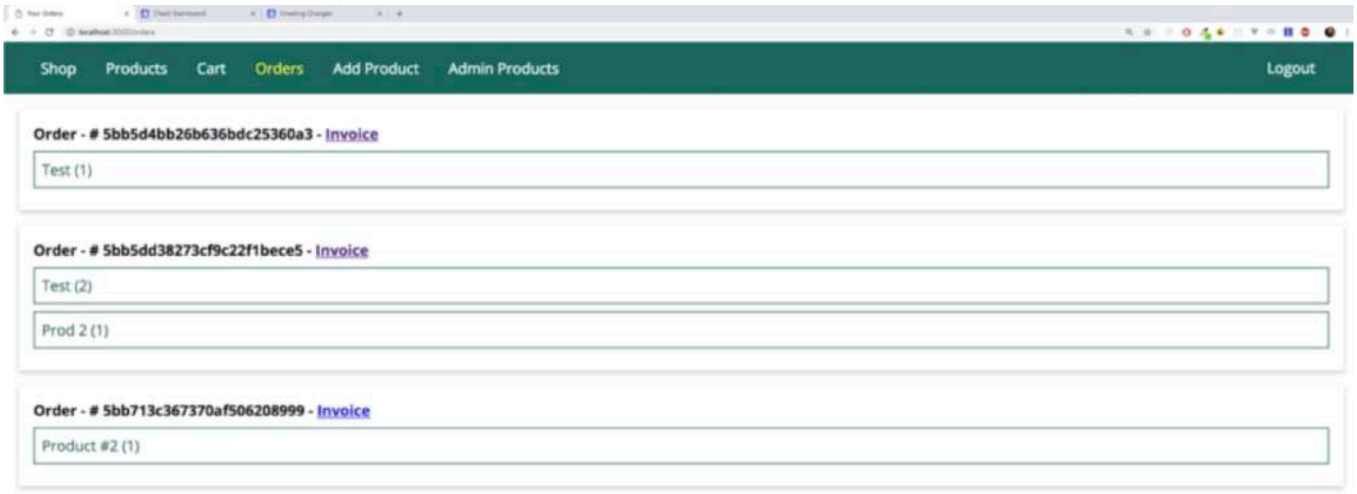


```
      const error = new Error(err);
      error.httpStatusCode = 500;
      return next(error);
    });
};

exports.postOrder = (req, res, next) => {
  const stripe = require('stripe')('sk_test_BMD9aaviqJzK0hlROg2KMRbD');

  // Token is created using Checkout or Elements!
  // Get the payment token ID submitted by the form:
  const token = request.body.stripeToken; // Using Express

  const charge = stripe.charges.create({
    amount: 999,
    currency: 'usd',
    description: 'Example charge',
    source: token
  });
  req.user
    .populate('cart.items.productId')
    .execPopulate()
```

```
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
(node:62175) 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.
^CMaximilians-MBP:nodejs-complete-guide mschwarzmueller$ npm install --save stripe
(                    ) ⠼ loadIdealTree:loadAllDepsIntoIdealTree: sill install loadIdealTree
```







Order - # 5bb5d4bb26b636bdc25360a3 - Invoice

Test (1)

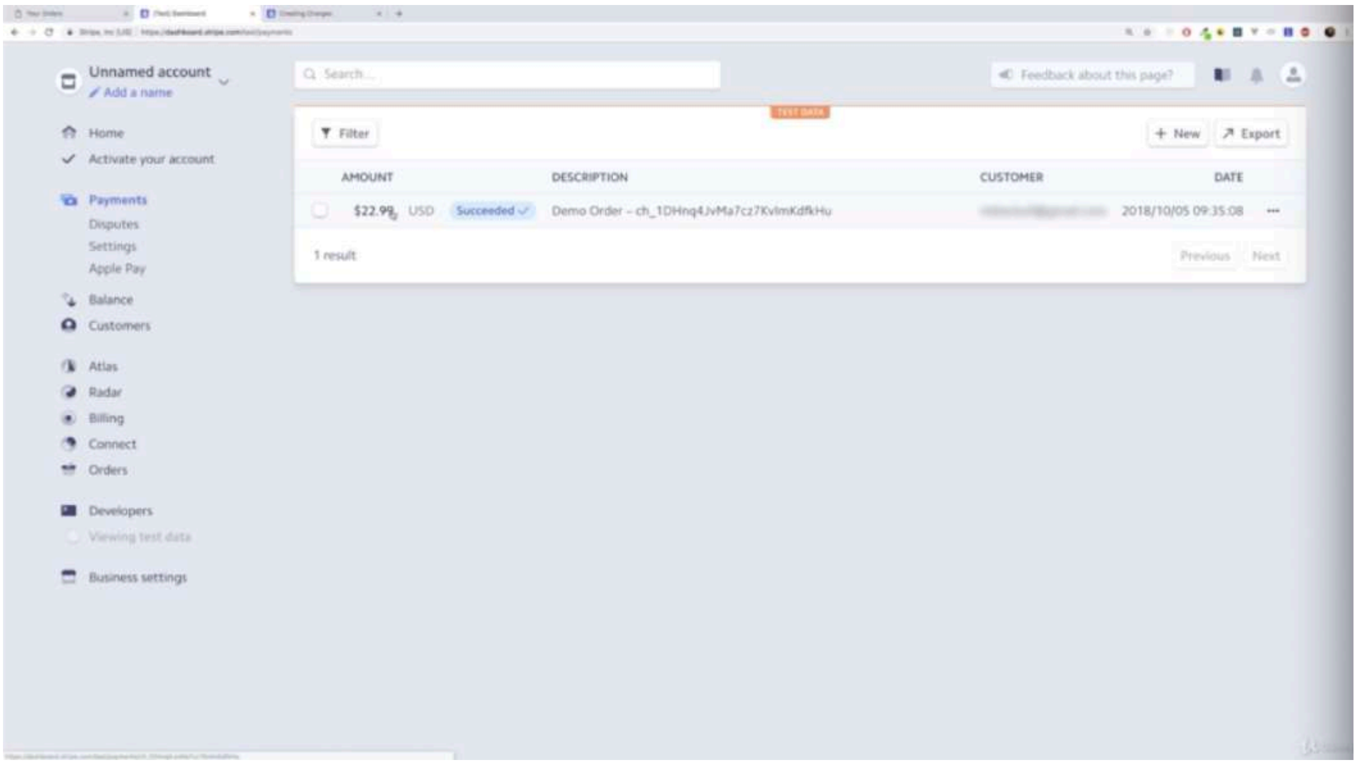
Order - # 5bb5dd38273cf9c22f1bece5 - Invoice

Test (2)

Prod 2 (1)

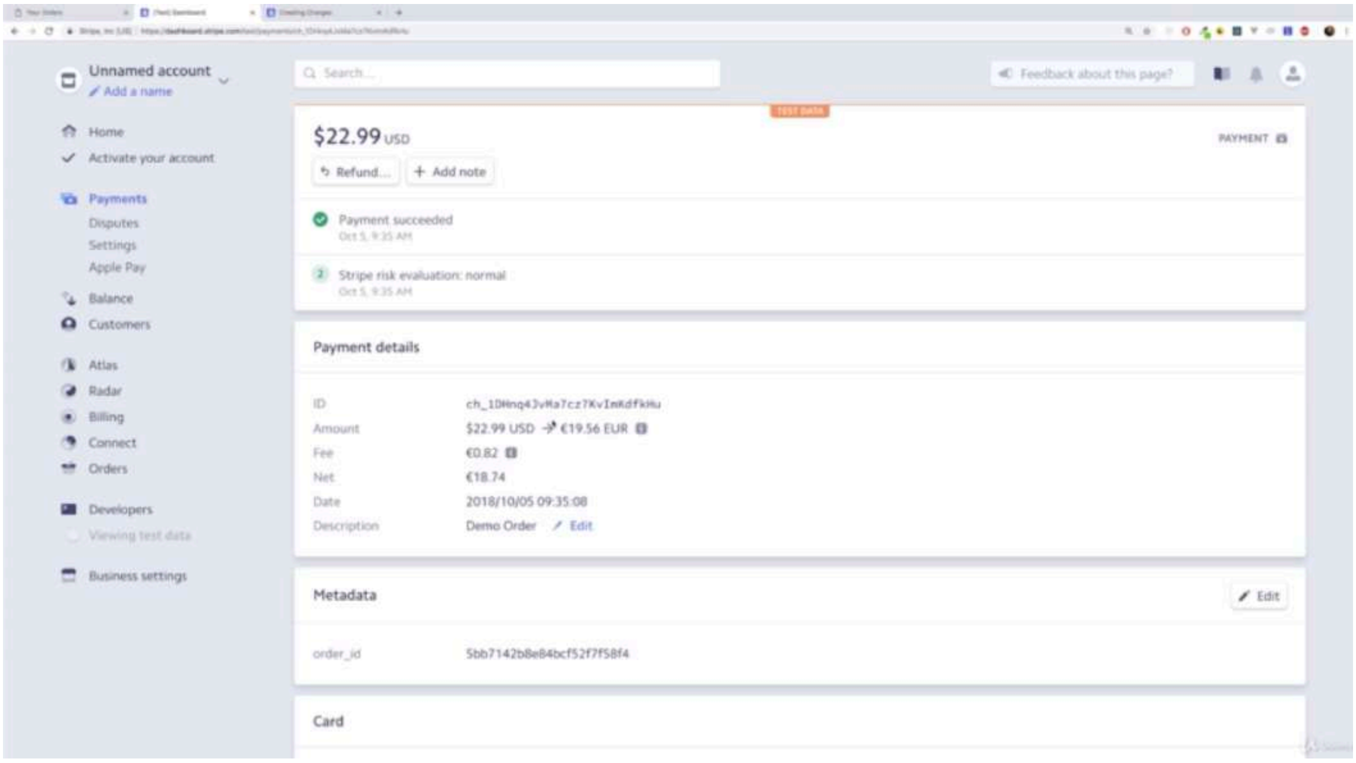
Order - # 5bb713c367370af506208999 - Invoice

Product #2 (1)



	AMOUNT			DESCRIPTION	CUSTOMER	DATE
	$22.99	USD	Succeeded	Demo Order – ch_1DHnq4JvMa7cz7KvlmKdfkHu	[redacted]	2018/10/05 09:35:08

1 result



$22.99 USD

Payment succeeded
Oct 5, 9:35 AM

Stripe risk evaluation: normal
Oct 5, 9:35 AM

Payment details

ID	ch_1DHnq4JvMa7cz7KvImKdfkHu
Amount	$22.99 USD → €19.56 EUR
Fee	€0.82
Net	€18.74
Date	2018/10/05 09:35:08
Description	Demo Order Edit

Metadata

order_id	5bb7142b8e84bcf52f7f58f4

Card

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

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

    </head>

    <body>
        <%- include('../includes/navigation.ejs') %>
        <main>
            <ul class="cart__item-list">
                <% products.forEach(p => { %>
                    <li class="cart__item">
                        <h1><%= p.productId.title %></h1>
                        <h2>Quantity: <%= p.quantity %></h2>
                    </li>
                <% }) %>
            </ul>
            <div class="centered">
                <h2>Total: <%= totalSum %></h2>
            </div>
            <div class="centered">
                <!--
                    from app.js file,
                    for the request sent by the stripe modal,
                    by that modal we entered our credit card data in,
                    this request doesn't include our CSRF token.
                    so '/checkout' form doesn't include it.

                    you could include your hidden input field
                    but that wouldn't do anything
                    because stripe creates its own form in the end in an ifram which it loads
                    but that also means that stripe takes care about securing that form
                    so this is not this form
                    which you embed into your document there,
                    stripe renders and loads and secures its own form
                    so you don't need to pass the CSRF token.

                    so this just means we have to disable CSRF token protection for this POST create-order route
                    to do that, i will cut the route from my ./routes/shop.js
                    and i will moe it directly into app.js
                -->
                <form action="/create-order" method="POST">
                    <script
                      src="https://checkout.stripe.com/checkout.js" class="stripe-button"
                      data-key="pk_test_ITc6cUJHHk47xmZcpx3xRmtL00lhes7lVh"
                      data-amount="<%= totalSum %>"
                      data-name="Your Order"
                      data-description="All the items you ordered"
                      data-image="https://stripe.com/img/documentation/checkout/marketplace.png"
                      data-locale="auto"
                      data-currency="usd">
                    </script>
                  </form>            </div>
        </main>
        <%- include('../includes/end.ejs') %>
```

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

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

const PDFDocument = require('pdfkit');
const stripe = require("stripe")("sk_test_dAPYh6CKcipsTX13kbjOFklT00NICMWhhg");

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

```

```javascript
const ITEMS_PER_PAGE = 2;

exports.getProducts = (req, res, next) => {
  const page = +req.query.page || 1;
  let totalItems;

  Product.find()
    .countDocuments()
    .then(numProducts => {
      totalItems = numProducts;
      return Product.find()
        .skip((page - 1) * ITEMS_PER_PAGE)
        .limit(ITEMS_PER_PAGE);
    })
    .then(products => {
      res.render('shop/product-list', {
        prods: products,
        pageTitle: 'Products',
        path: '/products',
        currentPage: page,
        hasNextPage: ITEMS_PER_PAGE * page < totalItems,
        hasPreviousPage: page > 1,
        nextPage: page + 1,
        previousPage: page - 1,
        lastPage: Math.ceil(totalItems / ITEMS_PER_PAGE)
      });
    })
    .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) => {
  const page = +req.query.page || 1;
  let totalItems;

  Product.find()
```

```
    .countDocuments()
    .then(numProducts => {
      totalItems = numProducts;
      return Product.find()
        .skip((page - 1) * ITEMS_PER_PAGE)
        .limit(ITEMS_PER_PAGE);
    })
    .then(products => {
      res.render('shop/index', {
        prods: products,
        pageTitle: 'Shop',
        path: '/',
        currentPage: page,
        hasNextPage: ITEMS_PER_PAGE * page < totalItems,
        hasPreviousPage: page > 1,
        nextPage: page + 1,
        previousPage: page - 1,
        lastPage: Math.ceil(totalItems / ITEMS_PER_PAGE)
      });
    })
    .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.getCheckout = (req, res, next) => {
  req.user
    .populate('cart.items.productId')
    .execPopulate()
    .then(user => {
      const products = user.cart.items;
      let total = 0
      products.forEach(p => {
        total += p.quantity * p.productId.price
      })
      res.render('shop/checkout', {
        path: '/checkout',
        pageTitle: 'Checkout',
        products: products,
        totalSum: total
      });
    })
    .catch(err => {
      const error = new Error(err);
      error.httpStatusCode = 500;
      return next(error);
    });
}

exports.postOrder = (req, res, next) => {
  // Token is created using Checkout or Elements!
  // Get the payment token ID submitted by the form:
  const token = req.body.stripeToken; // Using Express
  let totalSum = 0

  req.user
  .populate('cart.items.productId')
  .execPopulate()
  .then(user => {
    user.cart.items.forEach(p => {
      totalSum += p.quantity * p.productId.price
    })
    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 => {
    const charge = stripe.charges.create({
      amount: totalSum * 100,
      currency: 'usd',
      description: 'Demo Order',
      /**The source is the token
       * which we extract up
       * so that token which includes the validated credit card data.
       */
      source: token,
      /**some metadata which is a javascript object
       *  where you can pass any arbitrary data you wanna match this order with the charge
       * that will be stored in your stripe account.
       *
       * 'result_id' should be Id of the created order.
       *
       * itry to send some extra data in my ./controllers/shop.js
       * to stripe, the resultId should convert to string
       * otherwise this can't be added
       * and therefore my charge failed.
       */
      metadata: {order_id: result._id.toString()}
    })
      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;
      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();
      // 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));
};

```

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

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

/** from app.js file,
  for the request sent by the stripe modal,
  by that modal we entered our credit card data in,
  this request doesn't include our CSRF token.
  so '/checkout' form doesn't include it.

  you could include your hidden input field
  but that wouldn't do anything
  because stripe creates its own form in the end in an ifram which it loads
  but that also means that stripe takes care about securing that form
  so this is not this form
  which you embed into your document there,
  stripe renders and loads and secures its own form
  so you don't need to pass the CSRF token.

  so this just means we have to disable CSRF token protection for this POST create-order route
  to do that, i will cut the route from my ./routes/shop.js
  and i will moe it directly into app.js

  i will replace 'route.post' with 'app.post'
  because we only have the app available in this file
  then we also have a POST method to only trigger this middleware upon POST request

 */

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

/**the idea is that
 * i initialize this middleware
 * after this route which should not be checked
 * so that only the routes after this will be affected
 * because remember the requests travels through the middleware from top to bottom
 * so CSRF protection only gets added after this routes
 *    'app.post('/create-order', isAuth, shopController.postOrder);'
 *
 * so if you have exception of which you typically don't have too many,
 * place it before you enable this middleware
 */
app.use(csrfProtection);
app.use((req, res, next) => {
  res.locals.csrfToken = req.csrfToken();
  next();
});

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) => {
  /**for the request sent by the stripe modal,
   * by that modal we entered our credit card data in,
   * this request doesn't include our CSRF token.
   * so '/checkout' form doesn't include it.
   */
  //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);
  });

```

```
// ./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.get('/checkout', isAuth, shopController.getCheckout)

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

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

module.exports = router;
```