

How To Setup Stripe Checkout Using Express.js & Heroku

2019-02-26

Stripe Payments Checkout Express Heroku Cart

You have a product you want to sell on your website using Stripe, without any 3rd parties, and so all you need to do is figure out how to set it all up from scratch. Ha! Obviously, this is not an easy undertaking, especially if you don't know anything about web development.

I know because I struggled trying to figure this out myself. Fortunately, you won't have to struggle as hard as I'm going to walk you through the process step-by-step! :)

Here's a few things you should know before getting started:

1. You'll most likely need to setup a sub-domain for your Stripe Checkout as we're going to be using [Express.js](#) (a server-side framework) and Heroku to create our Stripe Checkout.
2. This tutorial is for MacOS and Linux users only unfortunately as I don't know how to do this using Windows at this time. We will be using the command line (or [Terminal app](#)) throughout the guide.
3. This guide also assumes you already have Node/NPM installed on your computer already. You can learn how to do that [here](#).

Okay, now that we have that out of the way, let's dive in!

Step 1: Create an ExpressJS app using Express Application Generator

First we want to install [Express Application Generator](#) globally. Open Terminal and enter the following command:

```
$ npm install express-generator -g
```

Now enter the following command to create the ExpressJS app in whatever directory or folder you're currently in, feel free to rename the app from 'myapp' to whatever you want to call it. For the rest of the guide I'll assume you just left the name as 'myapp':

```
$ express --view=pug myapp
```

Go ahead and [cd \(change directory\)](#) into the newly made directory, 'myapp' or whatever you decide to name it. From this point forward, this guide will assume you just left it named 'myapp' for sake of simplicity.

Then run the following npm command to install all the necessary dependencies. It's ok if you don't know what this means yet!

```
$ npm install
```

Now run your new express app and view it in the browser! It's super barebones right now but it's a full functioning express app.

```
$ DEBUG=myapp:* npm start
```

Now open a web browser and enter <http://localhost:3000> into the address bar, and press enter!

You should see a generic page like this:

Express

Welcome to Express

This is your app's homepage and only page right now. We're going to replace the content with Stripe Checkout.

Step 2: Install Stripe NPM Body-Parser Package

The rest of this tutorial essentially follows [this other tutorial found in the Stripe Documentation](#) FYI.

Enter the following command in Terminal:

```
$ npm install stripe express pug body-parser
```

Next, open up app.js in your favorite text editor and replace the contents of the file with the following code:

```
myapp/app.js
```

```
var createError = require('http-errors');

var express = require('express');

var path = require('path');

var cookieParser = require('cookie-parser');

var logger = require('morgan');

var indexRouter = require('./routes/index');

var usersRouter = require('./routes/users');

const keyPublishable = process.env.PUBLISHABLE_KEY;

const keySecret = process.env.SECRET_KEY;

const app = require("express")();

const stripe = require("stripe")(keySecret);

app.set("view engine", "pug");

app.use(require("body-parser").urlencoded({extended: false}));

app.get("/", (req, res) =>
```

```
app.post("/charge", (req, res) => {

  let amount = 500;

  stripe.customers.create({

    email: req.body.stripeEmail,

    source: req.body.stripeToken

  })

  .then(customer =>

    stripe.charges.create({

      amount,

      description: "Sample Charge",

      currency: "usd",

      customer: customer.id

    )))

  .then(charge => res.render("charge.pug"));

});

app.listen(4567);

// catch 404 and forward to error handler

app.use(function(req, res, next) {
```

```

    next(createError(404));
  });

// error handler
app.use(function(err, req, res, next) {

  // set locals, only providing error in development

  res.locals.message = err.message;

  res.locals.error = req.app.get('env') === 'development' ? err : {};

  // render the error page

  res.status(err.status || 500);

  res.render('error');

});

module.exports = app;

```

You don't need to know what any of this means right now, just know that you can adjust the price of the product on line 23:

```
let amount = 500;
```

You can also add a product description to line 32, by default it just says “Sample Charge”:

```
description: "Sample Charge",
```

Also, note that your Stripe API keys are used by the app on lines 10 and 11:

```
const keyPublishable = process.env.PUBLISHABLE_KEY;
```

```
const keySecret = process.env.SECRET_KEY;
```

I’ll show you how to setup your Stripe API keys later when we deploy our app to the internet using Heroku.

Important: Never share your Stripe API test or production keys.

Step 3: Create html page and Stripe Checkout form

This code isn't going to look like HTML to you since we're using a different 'version' of HTML (if you will) called Pug. You don't need to know that much about it right now, just create a file in the /views directory and name it index.pug.

Open it up in your text editor and enter the following code and hit the save button:

```
html
  body
    form(action="/charge", method="post")
      article
        label Amount: $5.00
      script(
        src="//checkout.stripe.com/v2/checkout.js",
        class="stripe-button",
        data-key=keyPublishable,
        data-locale="auto",
        data-description="Sample Charge",
        data-amount="500")
```

Step 4: Create a final 'thank you' page

We need to send our customers somewhere after the transaction has successfully completed, for this we will create another .pug file in the /views directory, this one we'll name it 'charge.pug'.

Open it up in your text editor and enter the following code and click save:

```
h2 You successfully paid <strong>$5.00</strong>!
```

Step 5: Deploy to Heroku

Deploying your Express app can be tricky to say the least. The easiest way that I've found is to use [Heroku](#). Heroku has a real friendly [command line interface \(CLI\) tool](#) for deploying your Express app.

Follow these steps to deploy your Express App to Heroku:

Install Heroku CLI

Create [Heroku](#) Account and install the [Heroku Command Line Interface tools](#)). You can find step by step instructions on how to do that [here](#)). You'll need [Git](#) installed on our computer first but they cover that in the guide here.

Confirm Node.js version in package.json

Make sure your version of Node on your machine matches the version you have listed in your package.json file.

You can check the version of Node you're using on your computer by typing the following command in the Terminal:

```
$ node -v
```

Create Heroku App Inside Terminal

Make sure you're inside your project directory (i.e. /myapp) via your computer terminal and type the following command:

```
$ heroku create
```

After Heroku has created your new app, inside your Terminal you'll see it gives you a special URL you'll be able to access your app from after you deploy. It will usually be some weird string of words followed by .herokuapp.com.

You can also access this URL from inside your Heroku.com Dashboard.

Set Environment Variables

If your application has any environment variables, you'll need to configure those as Heroku environment variables by typing the following commands into Terminal:

```
$ heroku config:set PUBLISHABLE_KEY="pk_test_yourtestpublishablekey";
```

```
$ heroku config:set SECRET_KEY="sk_test_yourtestsecretkey";
```

Deploy to Heroku

Alright we're on the final step! Once you've configured all of your environment variables go and enter the following command in your Terminal:

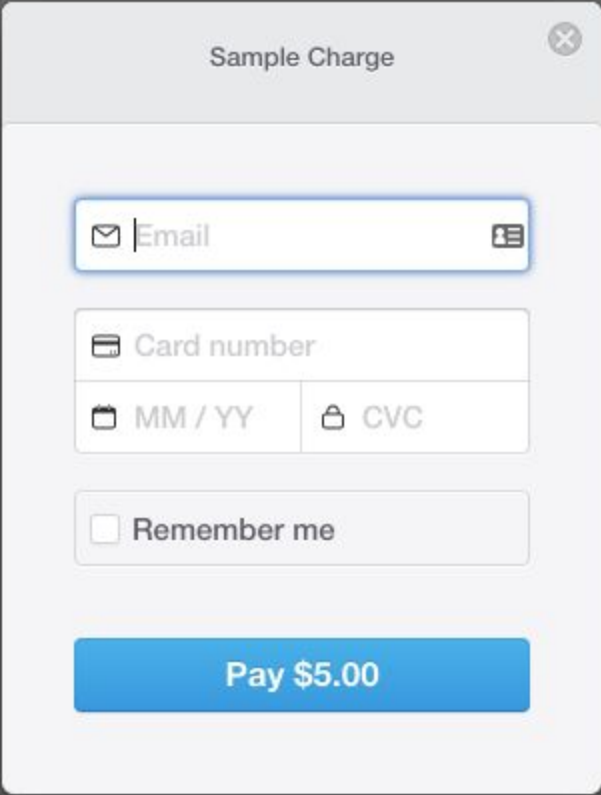
```
$ git push heroku master
```

Step 6: Perform a test transaction

Now it's time to test our new Stripe Checkout form! Visit the Heroku app URL for your new Stripe Checkout Express.js application. It looks like something like this "<https://rocky-journey-37911.herokuapp.com/>". You should now see our Stripe Checkout button:



Go ahead and click the button and you'll see the Stripe Checkout overlay take over the middle of the web page.

A screenshot of a 'Sample Charge' payment form. The form is a light gray modal with a close button (X) in the top right corner. It contains an email input field with an envelope icon and a list icon. Below it is a card number input field with a card icon. Underneath the card number are two fields: 'MM / YY' with a calendar icon and 'CVC' with a lock icon. There is a 'Remember me' checkbox. At the bottom is a blue button labeled 'Pay \$5.00'.

Sample Charge

Email

Card number

MM / YY CVC

☐ Remember me

Pay \$5.00

Enter your email address, test credit card number (4242 4242 4242 4242), any expiration date so long as it's in the future and any CVC number (I usually use 123).

Submit the test payment, if successful you'll be sent to the 'success' or 'thank you' page:



You successfully paid \$5.00!

After you've successfully completed the test transaction you only have one last step before you can process real transactions from customers.

Reset your Stripe API Heroku Environment Variables with your [production keys instead of your test keys](#) and you're all set to go!

If you'd like a custom domain for your Heroku App read [this guide](#).

Now simply style and format your new Stripe Checkout page, fill it with your sales copy and you've got a completely functioning sales page with payments.