Start accepting payments in any application, framework or programming language today

This plugin is the one of the best solutions to integrate Stripe Checkout in your work. Provided with a very elaborate documentation, even novice developers can integrate this plugin within 10 minutes and start receiving earnings on their account.

The package consists of two parts:

1. Server-Side API to process the Stripe payments

The purpose of the Server-Side API is to process your payments in a secure fashion. The API functions can be accessed by making HTTP POST calls. In this way, you can use it independent of the language that your application or website is written in. This part is written in NodeJS and can be hosted on any VPS such as Heroku or Cloud9 (documentation provided) .

2. Example in an Angular/Ionic setting

We also provide an example application, written in AngularJS/Ionic to illustrate how easy it is use to consume the Server-Side API. In addition, we integrate Stripe Checkout to handle the validations and retrieval of the needed stripeToken.

Functionalities include:

- Validate Credit Card Details
- Retrieve a StripeToken for the Transaction
- Customize the meta data and image
- Charge the user by sending a HTTP Post request to the Server-Side API
- Starter App which can be easily configured following the Ionic Documentation

Try before you buy

You can try this package out by downloading Ionic View (view.ionic.io) and previewing an app with app id: **997a3b85**

Getting started

Unzip the package in your workspace. You will find two folders:

- node-server-side
- ionic-example

Setting up your Stripe Account

We first need to integrate Stripe in our work. First make sure that you have signed up for a Stripe account. From there, you need to retrieve your API keys, which you can find from your dashboard here:https://dashboard.stripe.com/account/apikeys.

These keys are defined as: `STRIPE_API_PUBLISHABLE_KEY` and `STRIPE_API_SECRET_KEY`. You will need these keys in the following step.

Preparing all the Keys and Constants

You will need to configure three constants:

- Open `ionic-example/www/js/app.js` and
  replace `STRIPE_API_PUBLISHABLE_KEY` (either test or development)
- Open `node-server-api/server.js` and replace `STRIPE_API_SECRET_KEY` (either test or development).

The third constant is the `SERVER_SIDE_URL`, which you can obtain in the following steps.

Preparing the Server-Side

As the name implies, the node-server-side folder contains all the NodeJs files needed to setup your server. It is important that **NodeJS** is installed and available in your workspace. If you are working locally on your computer, you will need to host this part of the package on a site like Heroku (for production). Alternatively, to avoid installing a lot of dependencies, we recommend that you use Cloud9 Cloud9 (for testing and development)

To get started, cd into the directory node-server-side and install all the dependencies as follows:

```
$ cd node-server-side
$ npm install
```

This will install all the required packages. After this, deploy your package to a Cloud server (such as AWS, Heroku, C9, etc.). Below are two examples:

Example 1: Heroku

You can read most of the information you need in the tutorial Deploying with Git on Heroku. What is basically boils down to the following:

```
$ git init
$ git add . // or git add -A
$ git commit -m "commit name"
$ heroku create // it will ask you to login (sign up on Heroku.com for an account)
$ git push heroku master
```

The command line `$ heroku create` will create a directory for your app, something like:

```
falling-wind-4924.herokuapp.com
```

This is the location of your server side, and from now on we will define it as `SERVER_SIDE_URL`.

Example 2: Cloud 9

To avoid the hassle of installing dependencies such as git, node, etc. Cloud9 is a perfect solution to get you quickly started. The only disadvantage is that your workspace will go in a hybernation mode if you are not active on it for more than 2 days. This can be again solved by pushing your

work to Heroku and buying a dyno that keeps your server running 24/7. You can contact me for more information about this option.

To get started with Cloud9, simply create an account on c9.io and subsequentely create a new workspace with **NodeJS** as a template. Once you open your new workspace, delete all the nodejs files in there and unpack/unzip (drag and drop) all the files from the folder node-server-api of this package. Open the file `server.js` and press on Run (green button in top). You will see something like:

```
Your code is running at https://your-workspace-name-your-username.c9users.i
o.
Important: use process.env.PORT as the port and process.env.IP as the host
in your scripts!


Debugger listening on port 15454
Magic happens on port 8080
```

If you have chosen for this option, then we define from now on `https://your-workspace-name-your-username.c9users.io` as the `SERVER_SIDE_URL`

Replace the final constant

Depending on which option you choose, now we can replace the final constant on the client-side.

- Open `ionic-example/www/js/app.js` and replace `SERVER_SIDE_URL` with the appropriate value (depending on example1 or example2)

Preparing the Client-Side

Now that we have prepared the back-end, it's time to setup the front-end. If you are familiar with Ionic Framework, then you know the structure of such apps. If you are new to Ionic, then you probably need to setup the dependencies. Here is how to get started with Ionic:

```
$ npm install -g cordova ionic
```

Usually you would follow up by typing something like ionic start myApp tabs. This is not needed anymore since all the files are in the folder ionic-example. To run the application, the only thing you need to do is:

```
$ cd ionic-example
$ ionic-serve
```

If you are working locally, you can then visit `http://localhost:8100/` in your browser to preview your application.

Questions?

If you have questions or remarks, please let me know by either typing a comment on this site, or by e-mailing me to:`noodlio@seipel-ibisevic.com`

Have fun!