Stripe Payments with AngularJS and Flask

Kristin Holmbeck rev. by Yun Wang

15 July 2019 | 13 May 2020

# Outline

Stripe is the best software platform for running an internet business.

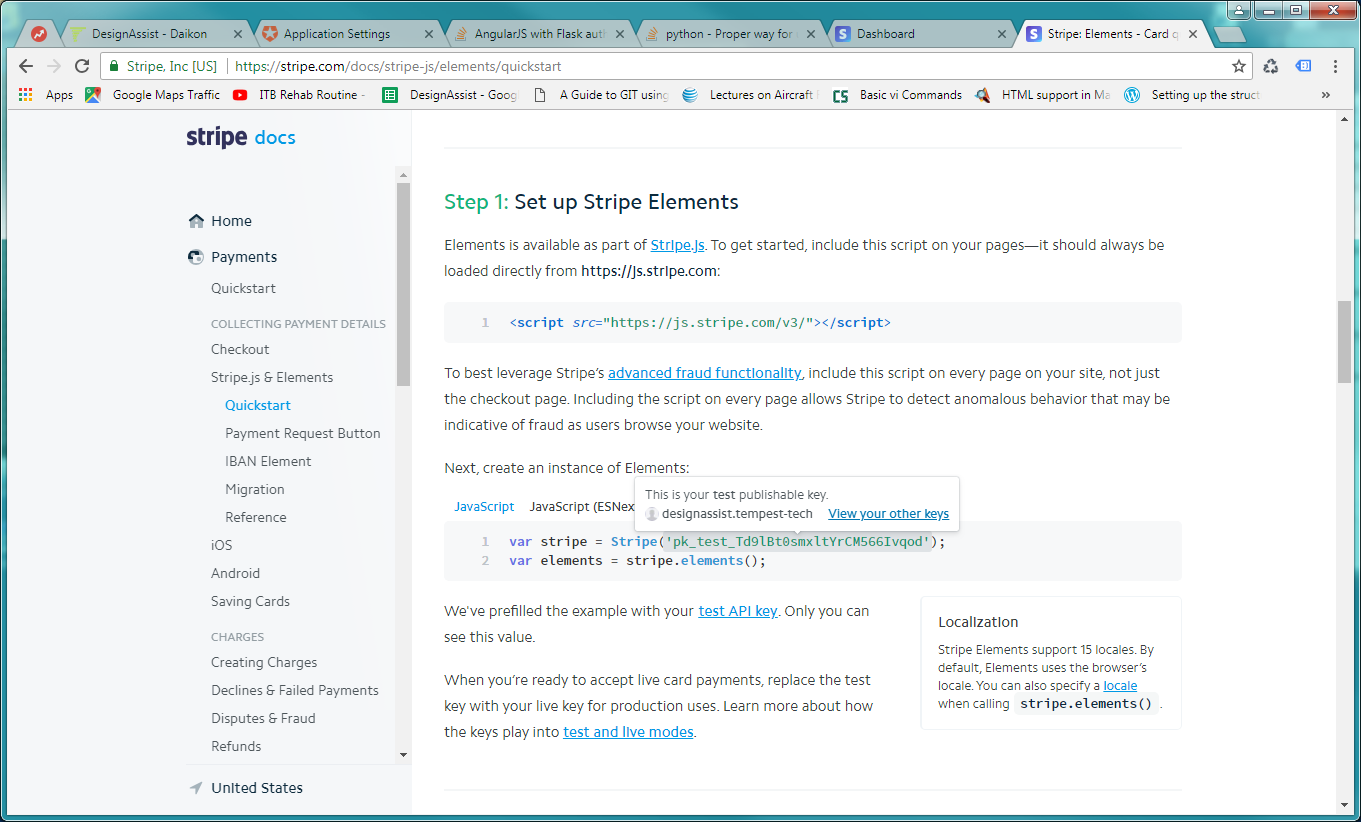
This document will outline the steps needed for Stripe integration into a SPA served by a Python Flask backend. As a preliminary step, service on HTTPS is required for Stripe integration. Refer to the document on HTTPS for more information.

First, we implement Stripe Elements: pre-built user interface components. Stripe Elements is a way of creating user payment forms to securely collect credit card information without the need for us (Tempest) to handle the data. The details are then converted to a *token* that we use to send a request via the servers.

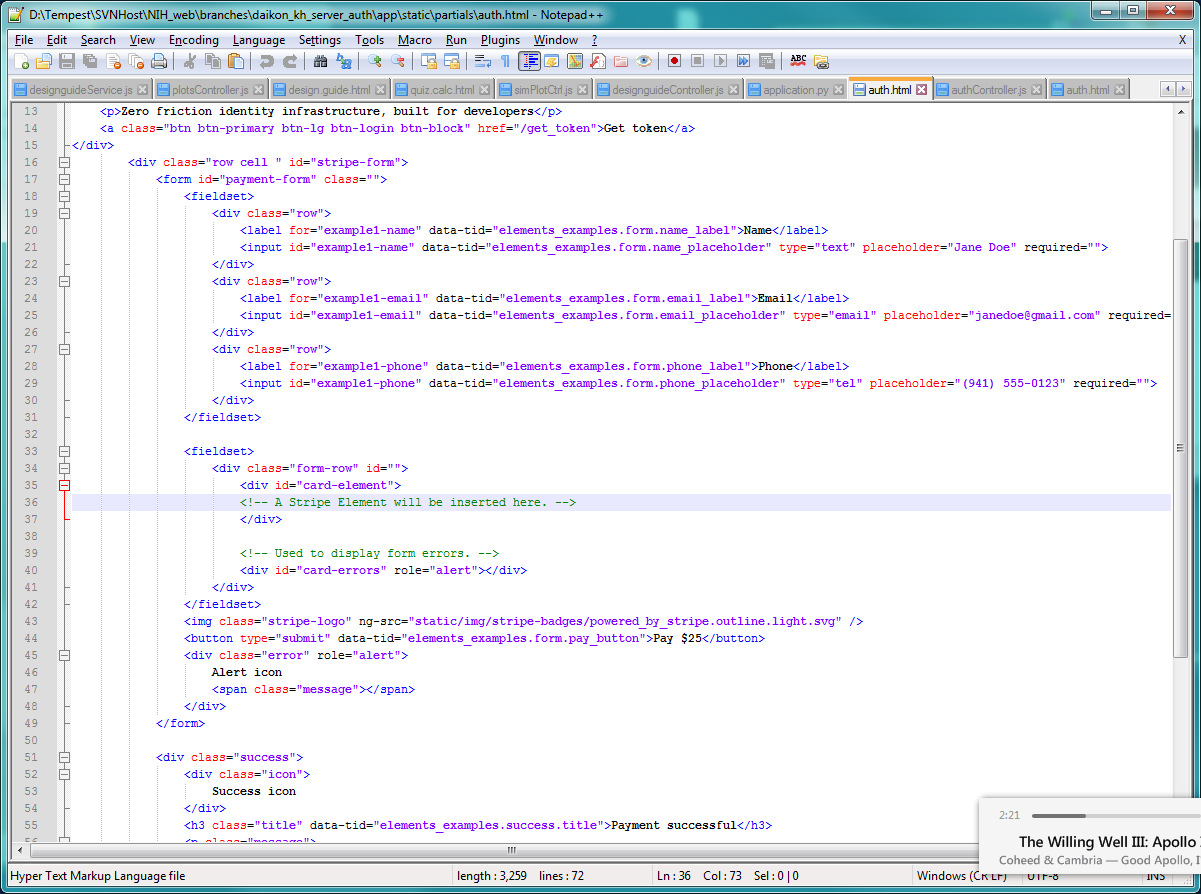
# Stripe Elements

The way Stripe works is by encoding a user's payment information and sending a promise directly to the Stripe server. That way, the Stripe server handles all sensitive information. The following information can be found at <https://stripe.com/docs/payments/quickstart>.

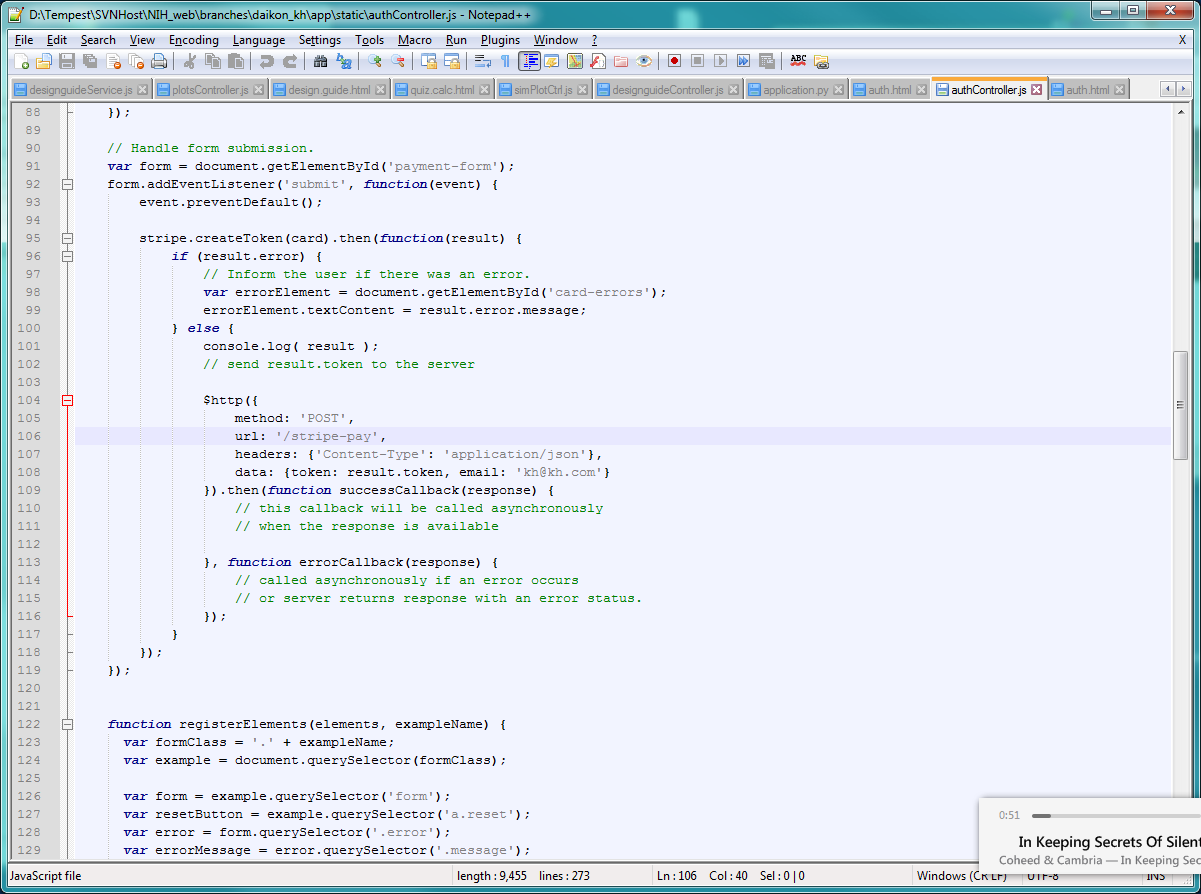
The first step is to set up Stripe Elements by including the script source and creating an instance using an API key that can be found in the Stripe <https://dashboard.stripe.com/>. In this instance, we have added a .js file called *myAccountController* to handle Stripe authentication.



Next, we create a simple payment form.

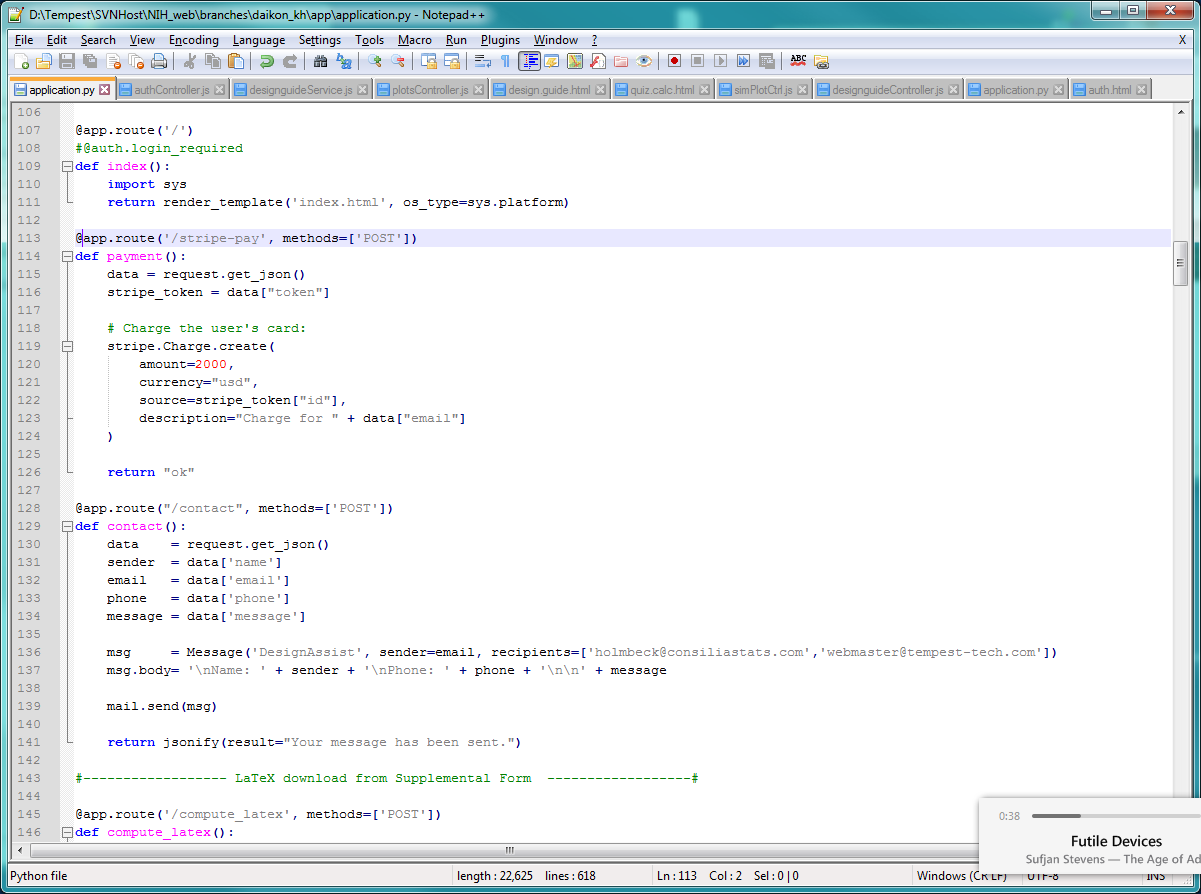
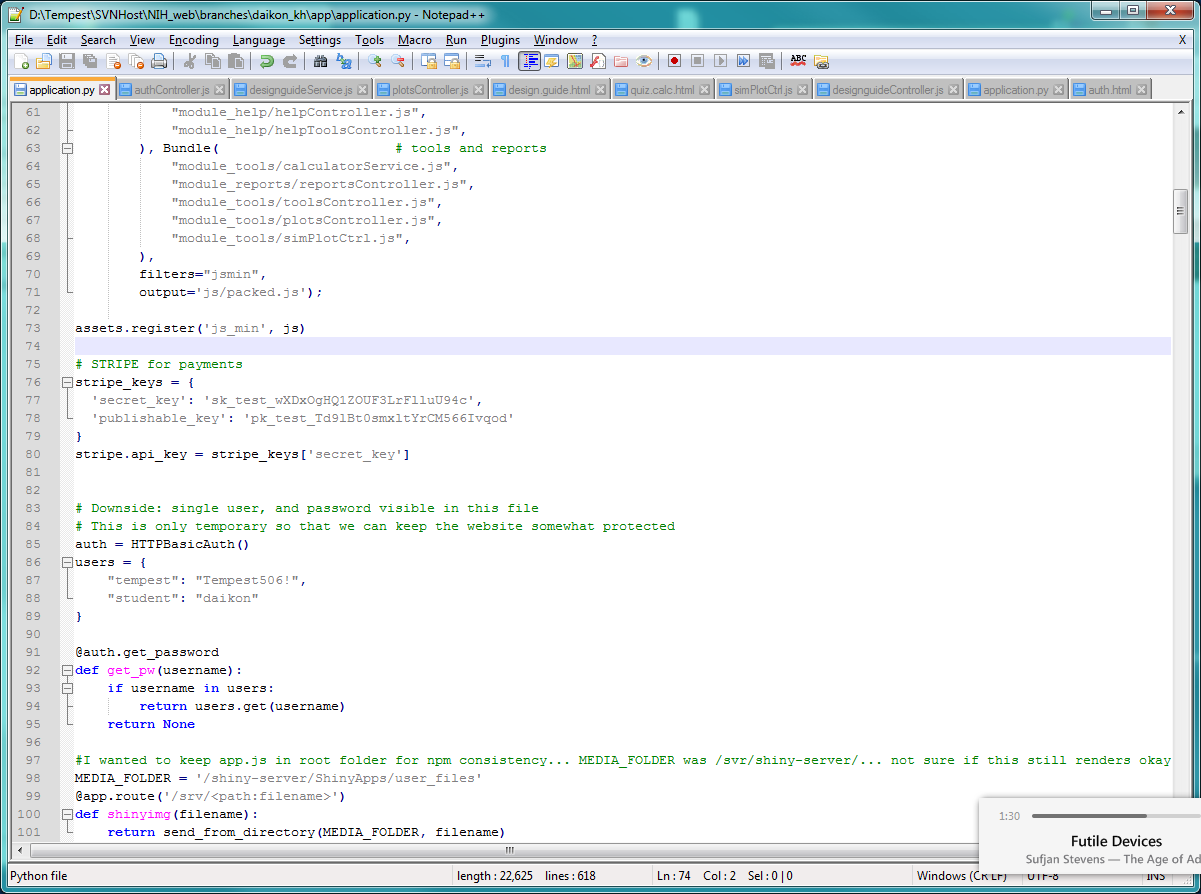


In the .js file, we utilize jQuery to append information to the form dependent on the id names. After the form is loaded, we mount the Element instance to the container with id *card-element* (line 35). On a button click, we submit the form and token to the server:



# Backend and Charge Requests

For our backend, we need to pip install stripe in order to use a Python interface to Stripe. The payment processing has two parts: (1) collecting payment details, and (2) using the collected payment method in a charge request. Stripe itself collects payment details by converting the details to a unique ID. In other words, Stripe (through Stripe Elements) collects all details and we, the company providing the service, do not see any user credit card information. The Python server (us) handles the charge request (2),



Configure Stripe to use the API keys specified in the Stripe Dashboard. Get the JSON information from the POST call, which must include the token and any other information, such as email.

There are several steps to signing up a customer:

1. Create a product
   1. This happens through the Stripe dashboard (online), and is only done once. For testing, we will create a DesignAssist product – a subscription plan of $20/month.
2. Create a new customer
   1. On success, this will return a new, unique customer ID
3. Create a new subscription
   1. This subscription is associated with a customer

Of course, error checking is important for notifying the user of things such as wrong zip code, card processing error, etc. These are currently in place through our Python Stripe processing backend.