# Stripe HW Friction Log Gabriel Hendel

**Overall takeaways**

Overall, I found the documentation for the Stripe PaymentIntents fantastic! The explanations were clear, well-structured, and included many ways to reduce coding friction. The guided sample integrations were superb, with support for major platforms and programming languages.

I found a couple nitpicky changes I’d make to the documentation, but otherwise most of the “friction” I experienced during this exercise (building a Bikeshed business!) came during setting up my development environment and working off my own rust as a coder.

**Friction Log for Stripe PaymentIntents Integration**

|  |  |
| --- | --- |
| **📛 Who’s doing this?** | Gabriel Hendel, applicant for Climate PM role |
| **📅 Date of friction log** | 13th February 2021 |
| **💡 What’s the use case?** | Integrate Stripe PaymentIntents to my website to create a custom payments flow to support one-time payments. |
| **🖥️ OS System** | MacBook Pro 2013, 3 GHz Dual-Core Intel Core i7  MacOS Catalina  Safari (with a little Chrome dev tools at the end) |
| **🌈 Logging key** | Green: Awesome, easy.  Amber: Hmm, bit tricky but getting through it.  Red: Would give up if it wasn’t my job \*bashes keyboard\* |
| **📛 Related docs** | Full [assignment description](https://docs.google.com/document/d/16SgZH2lFcmh1U0ABcdMlkfvupGe2MgxUMDy31qhCVY0/edit)  [Github repo](https://github.com/gabrielhendel/Stripe-Integration-HW) with answer code and README  (ask [gabrielhendel@gmail.com](mailto:gabrielhendel@gmail.com) for access to any of these) |

[Friction log template](https://www.trychameleon.com/blog/friction-logs)

## **My experience completing Stripe PaymentIntents Integration**

Ready, set, log!

1. Read take-home [assignment doc](https://docs.google.com/document/d/16SgZH2lFcmh1U0ABcdMlkfvupGe2MgxUMDy31qhCVY0/edit):
   1. Minor nits about the assignment documentation: Clicking on the 4 steps below (copied from the assignment)...
      1. [Create a PaymentIntent on the server](https://stripe.com/docs/payments/accept-a-payment?integration=elements)
      2. [Collect payment method details on the client](https://stripe.com/docs/payments/accept-a-payment?integration=elements#web-collect-card-details)
      3. [Submit the payment to Stripe from the client](https://stripe.com/docs/payments/accept-a-payment?integration=elements#web-submit-payment)
      4. [Asynchronously fulfill the customer’s order](https://stripe.com/docs/payments/accept-a-payment?integration=elements#web-fulfillment)

...leads to the same place (the top of the “Accept a payment” page with the “Prebuilt Checkout page” flow selected) whereas I’d expect it to open to:

* + 1. The “Custom payment flow” given this assignment’s instructions
    2. The relevant documentation page sections for each of the 4 links, instead of the top of the documentation page
  1. Also, the last step “Asynchronously fulfill the customer’s order” doesn’t have an obvious corresponding section in the API docs for custom payment integration, while the other 3 steps do.

1. Read Stripe’s “[Accept a Payment](https://stripe.com/docs/payments/accept-a-payment)” documentation that these links all led to:
   1. The documentation is self-explanatory, well laid-out, and has some nice touches like:
      1. clear documentation for Web vs other platform integrations, in key programming languages
      2. Tables of contents (left side), with an overview of the current page (right side)
      3. Example code and architecture overview
      4. Collapsible sections with clear labeling for server-side vs client-side integration
      5. Functioning integration UI with “Prefill card details”
   2. Potential improvements to documentation:
      1. Step 2 of the (“Create a PaymentIntent) of the [Custom payment flow](https://stripe.com/docs/payments/accept-a-payment?ui=elements) overview has a few code samples. It was not clear to me that “paymentintent” from one code sample has to be pasted into “intent=” from the previous code snippet. It took me a couple reads to realize that. This didn’t ultimately affect my work since I copied the sample code, but it made it harder to follow the explanations.
      2. If I were a partner unsure of whether to choose a “prebuilt checkout page” or “custom payments flow,” the documentation could have linked to pages helping me choose between the two (e.g. do partners see different levels of conversion between the two options?) This didn’t affect me since the assignment required a “custom payments flow” integration
2. Created a Stripe account
   1. Forgot my Stripe password but password recovery flow was slick
3. Installed Node.js
   1. Searched for [download link](https://nodejs.org/en/download/)
   2. Took a while to complete download but it was straightforward
   3. I had to read more about Node and React since I’d used it only once years ago. This led to a snafu… I first tried to install nvm from my Terminal based on recommendations from a Stack Overflow page, only to find out (from reading other documentation) that npm is much preferred over nvm. Oops!
4. Installed updated version of Github from the web
   1. After a failed install of xcode from App Store after 10 min of download time
5. Downloaded Stripe [sample code](https://stripe.com/docs/payments/integration-builder) “integration builder” from Custom Payment Flow
   1. Super easy and straightforward:
      1. Finding the docs from the main Accept a Payment documentation
      2. Platform/backend/frontend of my choice (chose web, react, node)
      3. Download feature
      4. Explanation of the different sections of code with code pointers (really handy for someone like me who hasn’t done this kind of coding in a while!)
      5. Gave me the brunt of the code I need for the PaymentIntents integration, so I didn’t have to do much work. YAAY!
   2. Only one nit about the sample code:
      1. When comparing the code sample integration doc and the Accept Payments API integration page, there was an app.post on the former that corresponded to an app.get on the other, which threw me off.
   3. Made some edits to personalize the code using Sublime Text v3
   4. Named overall folder “Stripe-Integration-HW”
   5. Found my secret key from Stripe dashboard and replaced it in local server.js code
      1. The code comments could have made it more clear to get the secret key to test, and where to find it, but once I looked it was easy to find on my Stripe dashboard
   6. Added a .gitignore file to make sure all the files in the “node\_modules” folder don’t get uploaded to Github every time since they’re not necessary
6. Uploaded files to Github in my first commit, added recruiter as collaborator on new repo
7. Customized look of website and made iterative improvements
   1. Updated the checkout.html to add a header and background image
   2. Updated global.css to make sure the background image didn’t distract from the main payments entry flow
   3. With each iterative change, went through this loop:
      1. saved changes
      2. entered “npm start” from Terminal
      3. reloaded the test webpage at <http://localhost:4242/checkout.html>
      4. Entered one of the test credit cards from Stripe’s [testing page](https://stripe.com/docs/testing)
      5. Checked response on test webpage, Terminal for errors, Stripe dashboard for payment logs, and log file for logs (see next step)
      6. Uploaded changes to Github repo once they worked
      7. This iteration involved a lot of context switching, and finding the right Stripe documentation browser tab was frustrating when I had 6 of them open with the same favicon
8. Created a log file for logging amount and currency from successful payments
   1. Googled “log file node js” because I didn’t know how to create a log file in Node
      1. Note: I didn’t follow the Webhooks explanation for this section of HW, since I had to return my partners’ laptop to her imminently and was looking for shortcuts to finish this assignment
   2. Found [this article](https://stackoverflow.com/questions/40880094/create-log-file-in-nodejs) that took a little while to understand, then copied the relevant code into:
      1. server.js
      2. client.js as fetch("/log-successful-payment"...
         1. I had to inspect the results of testing [my integration](http://localhost:4242/checkout.html) from Chrome to find the format that the amount and currency were logged in, and insert a breakpoint to debug an issue (due to a misspelling)
   3. Ran out of time before making tweaks to improve formatting of log file or web page