Mandi Grant (onid: granaman)
CS 290 Summer 2018

I've worked as a web developer for a few years now, so many of the concepts introduced in this class are not new to me. The things I find challenging are some of the usual pitfalls of developing for the web - versioning, adapting example code, learning a new framework's conventions, etc.

Challenge #1: Versioning discrepancies
Technologies move fast in web - it's easy to Google a question and end up on old docs or a Stack Overflow question that looks relevant but is actually from a couple versions ago, making it difficult to adapt the code or introducing failures that are difficult to figure out the cause of.

For the final project I wanted to use Bootstrap's carousel, but *only* wanted the carousel (nothing else from bootstrap, as I already had styled the page the way I wanted it). I grabbed the carousel HTML and the carousel-specific JS from Bootstrap's neat customizer feature, but I didn't catch that the customizer was still on version 3.3 and the carousel template code I copied from the Bootstrap docs page was from the latest version, 4.0. Whoops. They didn't work together, and it took me a few minutes to figure out *why*. Easy mistake to make, even with a few years of experience.

Challenge #2: Oh, we call that a "helper" here…
Sometimes, it can be hard to know what name a framework uses for a particular concept. For the final project, I wanted to pass data to the 'main' template itself (the one used as a sort of default 'frame' around the other routes' templates). My first stop was the official docs, but even with the helpers section staring me in the face it wasn't obvious that they were the solution to my problem.

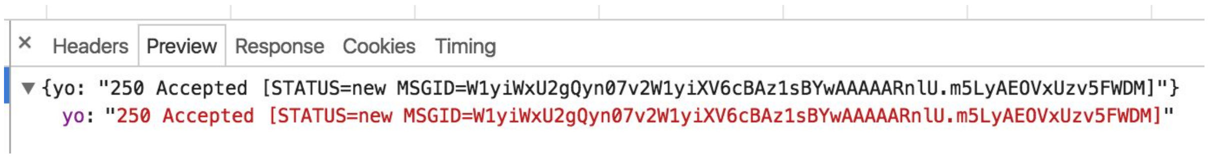
Coming from Ember, I might look for "components" or a data model. Coming from Angular, I might look for "directives" and try to grab data from the controller. It is like learning a new language every time! The code sample in the docs doesn't show the setting of the default template, so I wasn't quite sure their "create" code was anything like *my* "create" code. I only pieced it together thanks to this particular Stack Overflow question.

Challenge #3: Doing something new!
To challenge myself a bit on the final project I added a "contact us" form that would fire off an email. I knew that I couldn't just send an email from HTML or front-end JS (and I didn't want the clunkiness of opening the user's email client), but I suspected someone would've written a server side JS package that could help with the task. I found nodemailer, which was perfect.

Getting it to work with a post route was similar route work in this class, so that part was quick.

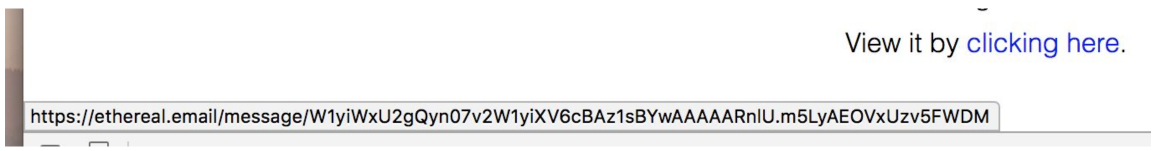
The real trick, it turned out, was in grabbing the response in the front-end JS so I could make a link out of the message ID code (so the TA who grades it can hop on over to ethereal mail's page and see the message was actually sent).

The ID I needed was actually part of the request that came back (along with status) but it was all in one string together.



```
▼{yo: "250 Accepted [STATUS=new MSGID=W1yiWxU2gQyn07v2W1yiXV6cBAz1sBYwAAAAARnlU.m5LyAEOVxUzv5FWDM]"}
   yo: "250 Accepted [STATUS=new MSGID=W1yiWxU2gQyn07v2W1yiXV6cBAz1sBYwAAAAARnlU.m5LyAEOVxUzv5FWDM]"
```

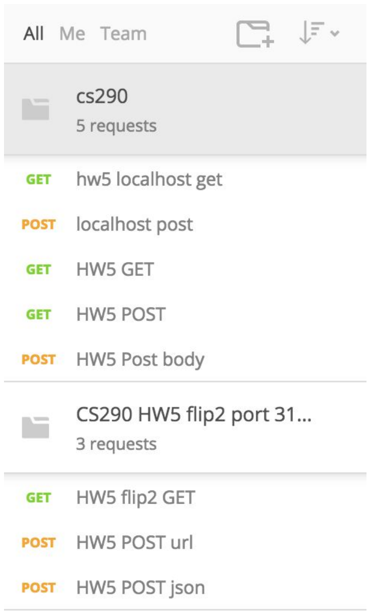
I was able to use a regex to pluck out just the relevant part (everything after MSGID=) and form my link:



View it by clicking here.

https://ethereal.email/message/W1yiWxU2gQyn07v2W1yiXV6cBAz1sBYwAAAAARnlU.m5LyAEOVxUzv5FWDM

Postman: a useful tool for checking my work in this class
I've used Postman before, so this isn't some huge revelation on my part, but it was the perfect tool for the POST/GET assignment. I have all my test data saved and organized (data sent as json, data as query params, sent to localhost, sent to flip) so every time I make a change, I can quickly hit the routes again and make sure each one is still working. I remember how amazed I was when someone first showed me this tool a few years ago, and I've been recommending it to anyone who seems lost as to how to send data to routes in this class.



cs290
5 requests
GET hw5 localhost get
POST localhost post
GET HW5 GET
GET HW5 POST
POST HW5 Post body
CS290 HW5 flip2 port 31...
3 requests
GET HW5 flip2 GET
POST HW5 POST url
POST HW5 POST json