Orentlikher Tim

CS290

Activity: Practice!

7/28/19

The readings and homework assignment for this week were probably the most difficult for me to pick up conceptually. I didn't have much trouble setting up forms or creating tables dynamically in prior weeks. While I'm familiar with the concept of Asynchronous programming, my experience has been mostly with making API calls and handling asynchronous responses, as opposed to programming them server-side. This was interesting to learn about because it gave me deeper insight into what happens on the server side of things. The homework assignment this week also gave me some issues when handling the GET/POST requests, particularly when parsing the query parameters.

It follows then that my practice activity would focus on those few topics. To boost my understanding of promises, I went through several of the exercises on Google's Developer site:

https://developers.google.com/web/ilt/pwa/lab-promises

This first thing I tried to better understand is how to actually make an asynchronous function. The exercises use the setTimeout() function, however, Javascript offers several ways to create one: setInterval, XMLHttpRequest, WebSocket, etc. I played around with using setInterval in an effort to break things, but it seemed to work as I would expect. Having created an async function, I created a few other ones so I could practice chaining. Using then() is pretty straightforward and similar to how you chain together methods in languages like Python. This really didn't cause many issues for me. Having created and chained together promises, I think my understanding is much better and they're not as complicated as I initially thought – they're just weird conceptually.

The second thing I wanted to practice is handling GET/POST requests. This week's homework had us create templates and route handlers, which was interesting. I did not struggle at all with the handlebars templating part of the assignment; however, parsing the query parameters was very challenging. When I first did the assignment, I was able to get a node running and displaying the proper template; however, my query parameters were not being displayed on the page. I set console.log() events in my code to troubleshoot, which gave me some indication of where my code was breaking because my terminal would properly see the query params. The issue I was having is that instead of passing a dict of key/value pairs, I was passing a string. When it came time to iterate over the object that I was passing to my handlebars template, it was failing upon trying to iterate over the string (rightfully so). I think part of my issue is that I'm coming from a compiled language such as C++/Python, I was expecting some kind of error or feedback from the browser that would indicate that my methodology was incorrect. Regardless, this was actually great practice in debugging and helped me better understand how to parse

queries. To take this assignment further, I decided to read more of the documentation around Express(). What's interesting is that you actually don't need to create separate templates for the GET/POST/404 error pages. Using res.render(), you can actually pass in dynamic titles to your pages also. I accomplished this by changing a few lines of code in an effort to break stuff:

```
res.render('request method', { title: 'GET Request Received', query: req.query});
```

I essentially route users to a generic "request method" template and specify the title I want to display in 1 line. I don't know what impact this has on performance, but it is much cleaner and easier to read.

You can obviously send a lot more via render() as well. One potential use case would be to send the results of a database query and display them in a view. Overall, the documentation has lot of useful things when it comes to render() that I will be experimenting with:
https://expressjs.com/en/api.html#app.render