Timothy Thomas
CS 290 – Spring 2015 Activity: Practice!

For this week's activity I decided to make a page that serves as a checkbook ledger for a bitcoin account. It allows one to add and delete transactions (either credits or debits). It displays a balance that is automatically updated as transactions are added or deleted. The functionality is very similar to the to-do list example provided in this week's module. What follows is discussion regarding how I checked my understanding of the different topics and what difficulties I had.

Create sessions, add and update different kinds of data to sessions and end sessions. Set up pages to accept data from forms.

What I did: enabled user to create a new session similar to the to-do list example. Used forms to allow users to add transactions to the ledger and delete transactions from the ledger.

Difficulties: Keeping track of variable names in the session object, local variables in the javascript code, views and the form submission variables and ensuring consistency was the most difficult part in getting a basic app up and running. It took some time to get all the bugs worked out, but was a good exercise in learning how the javascript code and session object is linked to the views and data displayed on the page. For example, it took some time to figure out why the account balance was showing up as NaN, even though the amounts in the transactions looked correct. I finally figured out that I was trying to access the individual transaction amounts with req.session.amount instead of req.body.amount. The second one is correct since when a new transaction is submitted, the amount data is contained in the request body.

Make HTTP requests, including POSTs, from the server and deal with the data you get back.

What I did: Since the user tracks all transactions using bitcoin as the currency, I added a request to the http://coinbase.com API to fetch the spot price of bitcoin in US dollars. I then used this data to calculate and display the account balance in dollars in addition to the already-displayed bitcoin amount.

Difficulties: this was the most difficult part of the assignment. I was banging my head against the wall trying to make this work as the USD account balance was not displaying. I couldn't even reliably log the returned spot price to the console. I knew that I was correctly parsing the JSON object returned, because sometimes when I would refresh the page or hit the back button it would magically show up in the console. Finally, after re-reading the class material, it dawned on me that I was not rendering the page inside the request call-back function. Once I fixed this in both the get and post handlers, it worked fine.

Make sure you can handle nested asynchronous requests and process errors if they crop up (force errors to happen by doing things like entering in the wrong URL).

What I did: I then added a nested asynchronous post request to my still running GET-POST checker from last week. I just put in some garbage data and logged it to the console to make sure I could get it working properly. Lastly I did some testing to makes sure errors were correctly handled by entering the wrong URL and doing things like page refreshes, back button, etc.

Difficulties: None with this step since I had figured out how to properly nest the requests and where to locate the page render statements.