PUI HW6: Adding Functionality to a Website with JS

Section A - Judith Leng

**Note to TA**

I’ve hard-coded part of the shopping cart to match the look of my static page mockup, and the scenario of a user’s interactions with the web pages would be that the user has some items in cart, and s/he is in the process of adding more. That said, the change in quantity and subtotal will be accurately reflected – the cart will just start with 4 items and a subtotal of $99.96.

**Design Choices/Reasons for Changes**

**Lo-fi**: I kept my design simple to match the overall visual language of the site I designed. I changed my slide-out cart into a full page mostly because I remember seeing the requirement on Piazza.

**Hi-fi**: I stuck to the white/dark gray color palette and chose a simple layout to minimize the visual load.

**Note**: more changes happened in the implementation process.

1. I added borders around the ‘remove” buttons to help users understand the actions they could take.
2. I added the checkout button to the top of the page because I realized with more items being added to the cart, the main call-to-action (CTA) button – checkout – goes below the fold. To help CTA remain visible, I included two buttons on the page.
3. I originally included quantity selectors in the cart page to help users make changes without having to go back to the product page. However, adding them made the jQuery code look a bit messier, and I decided to keep them as numbers for the scope of this project – I think learning about new functions and syntax in JavaScript is a better use of my time.

**Reflection**

While building the shopping cart, I’ve come across the following major challenges and bugs; below each point I’ll briefly talk about how I approached them:

1. **Limited understanding of the capabilities of JavaScript/jQuery**: I’ve learned bits and pieces of JavaScript on my own, but it’s still a relatively new language for me. When building this project, a lot of times I did not know what methods or built-in functions I could leverage to achieve the desired results (e.g. is there a way for me to check if a key-value pair exists in an array so that I can compare the product selected with those already in the cart to avoid having duplicates?)

***Solution***: I spent a significant amount of time learning about different functions and methods in JavaScript and jQuery, respectively – resources included Stack Overflow, Google searches and documentations from w3schools and jQuery, and I also learned from other people’s examples in CodePen and JSFiddle.

1. **Longer functions and more difficulty in debugging**: after making sure that some of the major functions (e.g. **localStorage**, quantity selectors) run, I went back to write “checker” conditions to make sure the function is also logically sound. Taking the example in #1, if the user selects an item that is already in the cart (both color and size match), the cart should update the quantity of the existing item instead of appending a new row. To achieve that, I wrote additional code for finding matches in color and size key-value pairs and identifying the existing item with the same characteristics. I had to learn new functions and I’ve run into bugs because I did not know how to use them properly.

***Solution***: Most of the times I managed to debug in the JS console in Chrome. I would either type commands in the console or write **console.log** after the statements where I suspected the issues were happening. Breaking down a “task” into multiple functions also made debugging easier.

1. **Hard-coded content made calculating cart subtotal slightly more difficult**: I hard-coded a static shopping cart page to match the mockup. Since the hard-coded data were not stored in **localStorage** to begin with, I could not simply add the price in the cart array together.

***Solution***: I found a work-around by extracting the price information in each table row and manipulating the data to get clean numbers for calculations. The process was a little challenging because of the nested HTML structures and the switch between different data types.

***Steps in detail***:

1. Loop through each row and find the specific column by using **.each()** and **.find()**
2. Find and take out the “$” by splitting the string, finding index and splicing the array
3. Piece the new array together into string (**.join**()) and parse it into price with a type of number using **parseInt()**
4. Use price to do calculations and update quantity/subtotal