Michelle Lu
Programming Usable Interfaces

Assignment 6B – Adding Functionality to a Website with Javascript

What challenges or bugs did you encounter?

Adding functionality to the shopping cart feature has been the most challenging part of the overall assignment so far. I encountered many bugs throughout the process, starting with implementing the visual indication of how many items are currently in a user's shopping cart. My cart kept displaying NaN as the total count regardless of the type or quantity of item being added. I finally realized that this bug was happening because the values for my quantity select options were written as strings (e.g. "one") as opposed to numbers (e.g "1"). Therefore, when I tried aggregating quantities, NaN was always the result because strings cannot be summed. The next bug I encountered was storing the selected quantity and glaze options from the product details page and displaying them in the shopping cart. I had trouble figuring out how to extract and remember the selected data from the individual product pages, but I eventually solved this problem by using online resources to find the `selectedIndex` property, which allowed me to return the index of the selected option in my glaze and quantity dropdown lists. Another problem I ran into was styling the table I used to display my cart items. Since the table is being modified dynamically, as users add items to their cart, it took some trial and error to format the item details in the way I intended. I wanted the item name and glaze choice to be displayed in a column view with the rest of the item details in a row, and I ultimately did so by assigning a class to both the item name and glaze option, then setting the `flex-direction` of that class to `row`. Lastly, I found difficulty with implementing the remove item functionality. I noticed that the items were not being removed because the `data-name` attribute that was being retrieved whenever the delete button was pressed was getting truncated, so instead of "Caramel Pecan Cinnamon Bun" only "Caramel Pecan" was fetched. This was because I had missed a quotation mark after `data-name=` in the line where each row of the table was being written in.

How did you overcome these challenges?

To overcome these challenges, I mostly used the browser inspect tool to help debug my code by putting in breakpoints and stepping through each line of the javascript file. This helped me see exactly what was happening when event listeners got triggered, which allowed me to pinpoint where the error was occurring in my code. I also used the `console.log()` method to output certain values if a function was working improperly. This was especially useful as I was debugging my remove item button because the debugger showed that the values being retrieved by the `data-name` attribute did not match the value of the actual items' data-name. I also used `console.log()` to see whether functions were being executed, as many of my bugs stemmed from the fact that the intended function was not being called or the body of an `if` statement was being skipped over. I often referenced online resources as well, such as Stack Overflow or

W3Schools, which were helpful for figuring out ways to implement particular functionalities if I was stuck.

Programming Concepts

From this assignment I learned the following 5 new programming concepts:

1. **Javascript array splice**
    a. In order to implement the remove item function properly, I had to make use of the `splice()` method. I had never used this method before, but I referenced lab slides and online resources to figure out how it is used to delete existing elements from an array. In my `removeItemFromCartAll()` function, I used the `splice()` method with one parameter that indicated the starting position from which to delete and the second parameter that indicated the number of items to delete.
2. **Web storage**
    a. When we were first tasked to implement a functional shopping cart, I knew that I needed a way for the cart to remember information across different pages of the website; however, I was unsure as to how to do this. From lab, I learned how to store items with JSON by leveraging the `sessionStorage` property. I used sessionStorage instead of localStorage so that users could start fresh with their shopping cart when they closed their browser, but their shopping cart information would be remembered for as long as their browser stayed open.
3. **innerHTML**
    a. Learning how to leverage the `innerHTML` property to dynamically update the shopping cart was critical for completing this assignment. This javascript property allowed me to manipulate the content of HTML elements, which was useful for displaying the cart, updating the number of items in the cart, and updating the total price whenever a user added to/removed from their cart.
4. **SelectElement.selectedIndex**
    a. Using web resources, I learned that the `selectedIndex` property returns the index of the selected option in a dropdown list. I needed to use this property in order to remember the values for the quantity and glaze options chosen by the user on the product details pages and carry that data over to the shopping cart page. Thus, the function attached to my add to cart button uses the `selectedIndex` property to add an item to the cart with the specified quantity and glaze options.
5. **Targeting the childSelector for .on()**
    a. To remove an item from the shopping cart, I needed to target the `.delete-item` class that characterized the remove item button. However, this class was a child of the `.show-cart` class that specified the table displaying the shopping cart content. Thus, I used web resources to figure out how to define a

function that attaches an event handler only to a specified child element (`.delete-item`) rather than the selector itself (`.show-cart`). This is exemplified by the `on()` function I implemented.