Andrew Kyroudis
PUI
Assignment 6B

Reflection

The first bug I had trouble detecting and solving was trying to access the contents of an empty array. I couldn't figure out why my product array was full in one moment, but couldn't be accessed in the next moment. I finally realized that the variable has to be within the scope of the loop – meaning that if it was for certain for-loop, for instance, it needed to be within that for-loop to be accessed. This realization was aided by the realization that I could declare global variables at the top of my Javascript file and these variables would then be available in all of the functions in the file. I needed to understand this in order to get the cart to properly load and display its contents.

Similarly, I faced a bug that kept inputting the wrong quantity of each item into the product array. I noticed that the quantities started at zero and counted up by one, so there was a consistent counter responsible for this activity somewhere… I soon realized that when using nested loops, one has to be very careful when using the conventional 'i' counter variable. If you use 'i' in a series of nested loops, you are liable to mix up which 'i' you are actually accessing. The fix was to use different counting variables.

The most frustrating bug that I faced was an infinite loop that started when I tried to navigate to the cart page. If I had multiple products that I was buying quantity of more than one, then clicking on the cart button initiated an infinite loop that prohibited the cart page from actually loading. I used console.log to track the contents of my product array and noticed that when I clicked the cart button, the product array grew and grew, becoming infinitely large. After scouring the code for a while, I realized that for the products with a quantity of more than one, I was inputting each product as many times as the quantity. This then looped back and did the same for the updated product array, meaning that the size of the array grew every time that the loop was completed, and the loop never completed because the ending index was never reached – since the array kept on growing. This was a very difficult bug to understand, but I am proud that I finally figured it out.

Programming Concepts

1. **Using local storage** – in this assignment we used local storage to store information while transitioning to a new page. Specifically, the product detail page allows users to add items to the cart, and when the cart is clicked on, the array of products is set in local storage, and then loaded into a new array upon the page loading. Thus, information gathered on one page could be accessed later on another page.
2. **Writing HTML in JS** – in order to display items in the cart, the Javascript code must be able to modify the HTML DOM. When displaying cart contents, I had to write the HTML code into a JS string that was then set to the innerHTML of an HTML element. This was very clunky and hard to understand at first, but I figured out that writing the HTML code

in an HTML file and then transferring it to my JS file and pasting it into a string worked well and minimized mistakes.

3. **Accessing HTML DOM elements in JS** – similar to writing HTML with JS code, I also needed to access elements of the HTML DOM in JS. Instead of simply using innerHTML, I figured out that I needed to utilize the parentNode attribute after accessing a certain HTML element. This allowed me to delete items from the cart by selecting the specific HTML DOM element by its id, and then using innerHTML to change the contents of that specific HTML DOM element.
4. **Double indexing** – In order to access the contents of an array with objects that have multiple attributes, you first need to access the index of the overall object, *and then* access the key of the attribute. For instance, if I wanted to access the 'style' of the second object in an array, then I would use the following double index: [1]['style']. In other words, this method of double indexing allows you access the multidimensional data within a complex array of objects. Before this assignment, I was unaware of the concept, and it seemed impossible to access those nested elements within an array.
5. **Variable scope** – I learned that the scope of variables depends upon where they are declared. Just because a variable is declared in one part of a function, does not mean that the variable will be accessible in another part of the function. Thus, I learned that for things like the product array, I needed to declare these variables globally, and then I could modify them within functions. Therefore, the variable was accessible throughout the function. Similarly, I learned that if a variable is declared within a loop, then a different loop (within the same program) will not have access to the variable unless it is declared within the function (or globally), and not within a specific loop within the function.