Jennifer Lee

Professor Alexandra Ion

05430 – PUI

7 November 2021

Assignment 6B

**Reflection**

Since this is my first time working with JavaScript, I encountered a lot of difficulties and bugs in my code. I tried working on this assignment in many different ways, but I couldn't get my code to work so I unfortunately didn't complete all the tasks. I know I may be making excuses; however, I felt that our time in lab didn't exactly prepare us to make this huge jump into incorporating interactive elements on our website. It is my fault I didn't attend office hours — the past few weeks have been very tough on me mentally and physically. I mentioned to Professor Ion a few weeks ago that I've been having to deal with personal matters back home and it's been having a huge impact on my motivation and work. I'm really sorry that my code is practically empty — I erased most of the functions that weren't working (basically everything) and will be talking about the problems I encountered instead.

One of the problems I first came across was getting my buttons to be interactive. I originally had each element of my buttons (to choose glazing and quantity) separated from one another. Each box and text had its own span class. However, I decided to fix that through CSS (combining text + box) and had each span class be a choice (ex. None, Sugar Milk, Vanilla Milk, Double Chocolate). Each span class (either glazing or item quantity) also had data within it so I could specifically call it in my JS file. I was able to use JS to make my buttons interactive — so when the user clicks on a choice, the background of the button will turn orange. This was done through having a glazeActive and quantityActive function to detect when the button is clicked or not.

Another problem was getting my added items to show up in my cart, while changing the number of the cart to change (showing "Cart (1)" if user adds an item to the cart). I decided to fix this by changing the Glazing and Quantity choice buttons to drop down instead. This helped me take in the chosen data a lot easier. I also changed the Add to Cart button to be interactive with a hover function and added a onclick=”addOrder()” function in HTML so when the user clicks Add to Cart, the addOrder function on my JS file will be called. However, my addOrder function wasn’t working properly because I didn’t see the numbers in my cart change, and the items weren’t showing up in my cart. I first had a constructor function to place all the chosen options that I call from document.getElementbyId into the constructors to add it as a new item. I also set a variable for itemsInCart = document.getElementById(“cartItems”).textContent; to get the current number of items in cart and then wrote itemsInCart = parseInt(itemsInCart)+1; to change the number in cart when the user clicks the “Add to Cart” button. I called sessionStorage.setItem(“itemsInCart”, itemsInCart); after to store the number and remember it even as the user goes to a different page. However, this function didn’t work no matter how much I tried to debug it, so I ended up erasing the code. I couldn’t even attempt to incorporate the delete function because I couldn’t get my items to show up in my cart.

I apologize again that I wasn’t able to showcase my final code with working elements on my website; but I still learned a lot through the process of attempting the assignment. I still wanted to turn in my reflection to explain my thought process and struggles with the assignment.

**Programming Concepts**

1. Local Storage
   - We learned how to incorporate localStorage into our code so that we can save key-value pairs in a web browser without it expiring. The user’s data is stored in the browser even when they go to a different page. To use localStorage, we have to implement it through setItem, getItem, or removeItem. setItem allows us to store values in the localStorage object — it takes in a key and value. getItem allows us to retrieve items and data stored in the browser’s localStorage. removeItem allows us to delete local storage sessions.
2. Change price (when clicking on a specific quantity)
   - I was able to successfully complete this by having an updateQuantity function be called when there was a change in the selection of a quantity. I was able to take in the data of the quantity the user selected and parse the text into an integer

(number). I then set the new price to be the quantity the user selected * the original bun price ($2).

3. Change cart item number (when adding item to cart)
    - I wasn't able to complete this functionality — but I explained earlier on how it should've been done. This functionality uses the parseInt concept where it takes in the text of the current number of items in the cart and turns it into a number so that you can add one every time the Add to Cart button is clicked.
4. Making the customized item show up in the shopping cart
    - I also wasn't able to successfully complete this, but this should be done by creating variables that take in the chosen data that the user selected (through document.getElementById). These variables can then be set into the constructor that takes in different variables (glaze, quantity, price) so that it can be created as a whole new item. The item is then set into a new variable and that variable can be added to an array of products in the cart. An if else statement can be called to check how many items are in the cart. If there is 1 (first item), the var of cartProducts should call a new array to add the item. If there is 2 or more items in the cart, the cartProducts can call the sessionStorage.getObj("cartProducts") so that it can push the new item into it.
5. Deleting an item from cart
    - I wasn't able to attempt this functionality but first you would have to retrieve the stored value of the cart item so we can modify it. This could be done through localStorage.getItem("cartItem"). Then, you need to find the index of the object in the list and this could be done through the .findindex method. Then, you can use the .splice method to remove the specific item from the list and update the stored value by calling localStorage again and setting the updated list of items.