Molly Schaefer

Assignment 6B

GitHub Repository

https://github.com/molschaef/molschaef.github.io/tree/master/assgn6/6B

Website Links

Home: https://molschaef.github.io/assgn6/6B/index.html
Product Details Page: https://molschaef.github.io/assgn6/6B/original.html
Shopping Cart: https://molschaef.github.io/assgn6/6B/cart.html

Functionality Added

1. Users can now add multiple products to their shopping cart from the products details page. The options they selected are stored and displayed on the shopping cart page, such as the image, flavor, quantity, glaze, and price.
2. Users can remove an item from their shopping cart by clicking "Remove".
3. The price for individual items displayed on the shopping cart reflects the quantity they selected multiplied by the unit price. For example, the unit price of a Pumpkin Spice bun is $10.00. If they selected a quantity of three Pumpkin Spice buns, then the shopping cart will show a price of $30.00 for that item.
4. The total price is calculated on the shopping cart page. The total is equal to the sum of the price of each individual item. When an item is removed from the cart, the total is updated to reflect that change.

Reflection

Throughout this project there were multiple bugs I encountered. The first issue happened when I was trying to style the selected ("active") flavor differently from the others on the product details page. Initially, whenever I would click a different flavor, it would set that flavor to active and keep the previous flavors that were clicked active as well. I resolved this issue by looking into how classes can be modified with JavaScript. I knew that I had used the class "active" to set the style for active links. Since I was able to add a class to an element with JavaScript, I considered the possibility that I could also remove a class. I searched online to find ways to remove classes in JavaScript and quickly found the replace function. Through this bug I learned that you can modify a class in JavaScript and how to apply that method.

The second bug I encountered was that the correct flavor was not being selected when I called my JavaScript function to set a selection to "active". Initially, I was unsure how to pass the specific element containing the function call to that function in JavaScript. I tackled this issue as

I did with most: with a Google search. I played around by passing the element using “this” and “this.id”. I discovered that “this” was the appropriate approach because in JavaScript I can check the id of the element passed in. I remembered we had learned about “this” for JavaScript but initially I was unsure if “this” could be used in HTML as well. From this challenge I learned that “this” can be used in both HTML and JavaScript.

Another bug I encountered was when I was calculating the price on the shopping cart page. Since I was using the inner HTML to retrieve the price, it included the dollar sign at the beginning and combined my prices as a string. For example, the total price of two items that were $10.00 came out to be “$10.00$10.00” when added together. In section we learned that you can use parseInt() to convert a string to an integer but I still needed a way to remove the dollar sign at the beginning. I searched on Google for how to remove part of a string and found the slice() function that will remove a defined portion of the string. Once I was able to remove the dollar sign I converted the string to an integer and the prices came out as a single number.

Programming Concepts

1: Object and prototype

The first programming concept I learned during this assignment was prototypes and objects. The prototype contains the repeat information across all the objects so that I do not have to duplicate information in each object. For the product details page, I used a prototype for all cinnamon rolls and then created a new roll for each individual flavor. This allowed me to simplify my code by using one prototype rather than defining the same key for each flavor object.

2: “This”

I learned how to use “this” to refer to the object or element that it appeared in. As I explained in my reflection, I was having trouble figuring out how to identify the flavor that had been selected by the user on the product details page. I had used “this” in the prototype of my objects but had not used it in other contexts or in HTML. I used “this” in my product details page HTML when I called the JavaScript function to update the information displayed and set the flavor to active. Understanding the “this” keyword was a powerful tool that I ended up using in multiple places.

3: Local Storage

The third concept I learned in this project was local storage. Through applying this concept I understood that it could save user input on the browser and that local storage keeps that information stored without an expiration. An example of where I applied this concept is on the product details page when I save the user input for product selections in local storage. I retrieve the information based on what is in the inner HTML and push that to an array of products. I then save the array of products to local storage and use that in my cart.js file to retrieve the array of products.

4: Variable Scope

Another programming concept I learned was global and local variable scope. Global variables are declared outside of functions and can be accessed anywhere in the program. Local variables are defined within a function and can only be accessed by that function. There are a few instances in which I used global variables in my JavaScript files. One example is my cartQuant variable that stores the quantity of items added to the cart from the product details page. When I initially put this variable into the function to update the cart quantity, it reset the variable to 0 each time since I initialized it at 0. By initializing this variable to 0 outside of the function I was able to set it once without resetting it every time that function was called.

5: Template Literals

I felt as though some of my code was messy and went on for multiple lines when it did not need to. One way I looked to simplify my code was by using variables inside strings. I googled how to do that and found the concept of template literals. By using a specific syntax I was able to call variables from within a string without having to use concatenation. An example of where I used this is in my cart.js file. I wanted to add stored information about products to a string of HTML so I used template literals to embed the variable within the string.