Assignment 6 – Adding Functionality to a Website with JavaScript
05-430 – Programming Usable Interfaces
Spring 2020
Roy Xu

Programming Concepts

JavaScript Objects

The first JavaScript concept that I got a ton of practice with this assignment was using JavaScript objects. Initially, I created a Product object and many sub-classes that inherit from the Product class, such as apple dumplings and cinnamon rolls. Eventually I got rid of all the other objects other than cinnamon rolls since I only implemented on type of product on the product detail page.

One of the challenges I had was retrieving the stored values in the object. I forgot that every time I use JSON to stringify an object, I have to parse it to get its key-value pairs. Specifically, I had name, flavor, glazing, and quantity for my Cinnamon Roll object. Every time the user clicks Add to Cart, a new object was created and inserted into the local storage. It would be nice to implement a master class with name and quantity, and product sub classes with more specific user selections such as flavor and glazing options to make the website even more realistic.

LocalStorage

I've used Ruby on Rails before and I had the impression that to make a fully functional website with memory, I needed to use the Model-View-Controller method. LocalStorage or client-side storage was eye-opening for me to see user selections carried on from one front-end page to another. While I only used the setItem and removeItem method, corresponding to the Add to Cart button and the Remove button, I learned a lot about how the browser can come in handy to store insensitive and limited information to better user experience, especially when it comes to front-end things such as user preferred design options.

Pass in Parameters from HTML

I had the most trouble calculating prices for the items based on user input/quantity change. I wrote quite a few functions to handle these changes from changeQuantity to setItemPrice to calculateItemPrice to calculateSubtotal. Initially, I set every element individually by using the Document.getElementById method, but this became an issue once I started templating the shopping cart items. To overcome this issie, I learned that I could pass in parameters from HTML such as the id of the element being called on (using this.id) and by doing so, I was able to grab the specific element and set its values accordingly to respond to user inputs.

Customize DOM Elements/HTML Event Attributes

One of the most used methods in my JavaScript file was the Document.getElementById method. I used it to customize the page based on user inputs, from the product image to flavor to glazing to quantity selections. Coupled with HTML event attributes such as onClick and onChange, I was able to call the specific JavaScript functions to modify the DOM tree as needed to allow for front-end changes. For example, whenever the user changes the quantity on the form on the shopping cart page, I call the changeQuantity function with the id of the specific form and calculate the new itemPrice and update the innerHTML of the field accordingly.

Type Conversion

Calculating item prices required me to switch from int to string and vice versa, but after experimenting with JavaScript, I realized that the language dynamically converts types for the user whenever possible. For example, “$” + 3 in other languages such as Python would raise a TypeError, but in JavaScript it understood what I intended to do and changed the result to a string automatically. Moreover, I learned to use the toFixed method to change a number into a string with fixed number of decimals for more detailed numerical display on the shopping cart page, allowing users to better breakdown the prices of each item as well as the subtotal of the shopping cart.