Reflection

During the creation of these HTML, CSS, and Javascript files, I encountered countless common syntax errors: using the wrong capitalization in a variable, forgetting a semicolon, parenthesis, or bracket, switching IDs and classes, etcetera. All of these errors simply require sifting through your code endlessly until you find the missing or incorrect piece. Beyond common syntax errors, I also encountered several primary issues or bugs in Javascript that required notable time to discover or Googling to fix.

First, on the product details page, I noticed that clicking “Add to Cart” added the correct total quantity to the upper-right-hand cart symbol, but selecting a new quantity to add next added both it and its predecessor’s quantity to the existing cart number. For example:

1. Select x2 “No Glaze” buns → click “Add to Cart” → cart number = 2.
2. Then, select x1 “Vanilla Milk” buns → click “Add to Cart” → cart number = 2 + 2 + 1 = 5. (Old cart number + previous order addition + current order addition = new cart number.)

I soon realized that the detail page’s quantity tracker (orderQuant) was not resetting after each “Add To Cart” click. In order to fix this and similar issues, I included a number of resets at the end of the addToCart() function. There, I reset the orderQuant to 0, the details page pricing to 0, and each of the dropdown selections’ values to 0 in order to have the dropdown selection revert to default ‘Quantity’ text. This process showed me the value in resetting variables to zero after each use or after their intended usage.

Next, when calculating the cost of buns (quantity * $3.95), I was getting an incorrect and recurring number. Googling this error taught me that multiplying by decimals in Javascript causes incorrect answers due to errors in the floating point multiplication precision. In order to resolve this, I found solutions online by adding .toFixed(2) to my pricing, which brought the decimal places down to two (e.g., $3.95) and solved the problem. I now know to use .toFixed(2) on all future multiplications of monetary costs, and to adjust the number (2) for different numbers of decimal places used by non-USD values.

Furthermore, I had numerous issues trying to get document.getElementById() to produce or be re-written as the type of content I wanted. At first, I Googled for solutions of how to call or rewrite the correct type of content. When I became more familiar with some common code

additions, I often just cycled through different additions of .value, parseInt(), .innerHTML, or setAttribute() until I got/created the string, code, or number that I wanted. The more times I used one particular method successfully in previous code, the faster and easier it became to use the correct method on the first try.

Finally, I struggled with having my cart display as empty when I had no items in it. I had a “new-item” in my HTML under a parent “all-items” that I wanted to clone and populate with each new item’s information. I knew there was a way to make HTML visible and invisible using Javascript, but on a wild attempt that I half-expected to fail, I tried copying the whole “new-item” div into my Javascript file (within quotes). For each bun item added, the HTML of a “new-item” would be added to the .innerHTML of “all-items”. But how would I keep track of each “new-item” and give it the correct attributes? I added an index variable *i* within each ID in the HTML, thus creating a custom “new-item” with each cart addition. While a bit unusual, it works! This showed me the flexibility of HTML and Javascript, and how each can be used within the other (via <script> in HTML or .innerHTML in Javascript) as suits your needs.

Programming Concepts

Five programming concepts that I learned in Javascript through this exercise include:

1. Objects and Arrays
   Objects are unordered lists of properties that are stored as name-value pairs, while arrays are ordered collections of properties or objects. For example: my Cart array filled with new Original flavor buns from my details page.

```
// For example: Making an empty array to fill:
var myCart = [];

// For example: Making a new object / instance for the array
new originalBun("No Glaze", document.getElementById("quantNG").value)
```

2. Javascript array push()
   Adding new items to the end of an array.

// For example: Adding a cinnamon roll object with Original flavor, No Glaze, and some quantity (pulled from its Details page dropdown menu) to the total array of rolls in the cart.

```
myCart.push(new originalBun("No Glaze", document.getElementById("quantNG").value));
```

3. Local storage
   Client-side storage of data made by user interacting with Javascript elements.

// For example: pulling the myCart array from local storage

```
var myCart = JSON.parse(localStorage.getItem("myCart"));
```

// Storing an object in localStorage, made into a string:

```
localStorage.setItem("myCart", JSON.stringify(myCart));
```

4. Change attributes within HTML (.setAttribute)
   Changing parts of the HTML by attribute label.

// For example: changing the main photo by clicking on my thumbnail pictures.

```
document.getElementById('main-image').setAttribute("src", "Images/noicing.jpg")
```

5. If / else statements
   Making conditions for when certain Javascript code is run.

// For example: If selected no quantities of any glaze from the Details page dropdowns ...

```
 if (orderQuant==0){
  // Then don't let anyone add an empty div to cart (disable "Add to Cart" button) ...
   document.getElementsByClassName("addtocart").disabled=true;
 } else {
```

// Or if quantities are selected, add them to the total Details-page-specific quantity that will update the number in the top right of your screen.

```
   cartNumber = parseInt(document.getElementById("cartNumber").innerHTML);
   cartNumber = cartNumber + orderQuant;
document.getElementById("cartNumber").innerHTML = cartNumber;
```

6. For Loops

   Continually implementing a segment of Javascript code until the index number (i) is undefined or hits a certain point.

```
// For example: For the first item added to cart, and up by one until equal to or over myCart size:

for (i=0; i<myCart.length; i++){

    // bunItem is an object in myCart array (at index number i in line):

    var bunItem = myCart[i];

      document.getElementById("all-items").innerHTML+=` (HTML here) `;

       }
```