Divya Mohan

PUI Section D

Github Repository: https://github.com/divmohan7/divmohan7.github.io/tree/main/assignment_6

Github Pages Link: https://divmohan7.github.io/assignment_6/index.html

Reflection

A majority of the bugs I encountered were related to minute syntax errors. Forgetting an end bracket was the most common reason for my code not working. Another was neglecting proper capitalization, particularly when using Camel Case. One practice I also neglected until part way through my code was using semicolons following each line of JavaScript. I also found that I ended up switching between “glaze,” “glazes,” and “glazing.” In general, I want to be better about planning my code before I execute it. I tend to get fixated on finding the solution before properly annotating my notes or considering the process. I look for quick fixes. In one example, I set the values of the option selections on my product page to be image links in order to have the image change when glaze flavor was changed. This helped me complete 6A, but when I had to return to 6B, I couldn’t find a clear way to grab the glaze info as it was linked to the image. Planning out the separate steps involved in creating a shopping cart early on would have helped to mitigate this. On trying to resolve this issue, I also found myself faced with a giant wall of code, unable to properly retrace my steps. Annotating my code originally would have helped me better pin point issues during the debugging process.

Generally, much of my debugging involved including console.log statements. When trying to get my product info to populate on the shopping cart page, I originally couldn’t find the best way to display the data. Using console.log to print the array in the DevTools bar showed me that my array was being created and stored properly. It let me know that the issue wasn’t in forming the array or pushing objects to it, but on creating a display to be populated. From here I was able to create innerHTML elements to rectify the issue. I also turned to stackoverflow and W3schools frequently during the process. Because the shopping cart functionality was a basic function, there was a wealth of information from new and practicing developers on stackoverflow that helped me solve basic problems, like knowing to use “splice” rather than “delete” to remove items from an array. The tutorials on W3schools were also incredibly helpful in laying out the logic of JavaScript statements for new programmers.

Programming Concepts

Unless separated by brackets, for each instance listed below, the examples given are uses of the same concept over different functions

1. Local Storage

I used local storage paired with a product array and the “add to cart” button to store and recall the user’s choices from the product page. Local storage allowed me to pull the user-selected information from that session and use it to populate the shopping cart page. Local storage was particularly helpful to use because, as we learned over lab, session storage would have caused the information to “expire” once the page session ended. It was an effective callback for the information that was entered.

```
localStorage.setItem('order', JSON.stringify(productArr))


var loadedProductArr = localStorage.getItem('order')
```

2. Accessing, pulling from, and overwriting HTML elements

I found the built-in DOM methods extremely helpful in modifying HTML elements. “document.getElementById” was the most frequent method I used found to pull information related to the glaze and quantity choices made on the product detail pages, retain their values, and output related images and information to the listed price and glaze image.

```
function priceCounter(){
    var price = document.getElementById("price-details");
    price.innerHTML =  document.getElementById("quantity").value;
}
function setImage(select){
  var image = document.getElementsByName("image-swap")[0];
  image.src = select.options[select.selectedIndex].value;
}
```

3. Adding new HTML elements dynamically via JS

This was an aspect of JavaScript that was messy and I was partially uncomfortable with it, yet it proved very helpful in populating my cart page with the product information pulled from local storage. I used innerHTML to directly and dynamically change the content of the checkout page depending on the objects that were created in the array, adding text and classes around them. I was unaware that this level of flexibility in content and even styles could be implemented using primarily JavaScript.

```
cart.innerHTML += "<div class='cart-img'> <img src="+ cinbunGlaze +"> </div>"
cart.innerHTML += "<h3><b>Pumpkin Spice</b> </br> <b>Glaze: </b></br> <b>Price:</b> $"+
cinbunQuantity +"</h3>"
cart.innerHTML += "<span class='delete' onclick= 'deleteProduct(" + i + ")'> Delete </span>"
```

4. For loops

A for loop was the main method I used to retrieve the details of the product array and fill the cart page. I iterated through each item in the product array that had been created to retrieve the individual characteristics – glaze, quantity, price, etc. It was a complicated concept for me to wrap my head around, particularly the use of the “i” in “i = 0” and “i++.” It was the most “mathematical” part of JavaScript to me and something I struggled with. Nevertheless, it felt extremely satisfying and allowed the most daunting part of the assignment – parsing out a variety of separate product details – to be done cleanly with a few lines of code.

```
for(var i = 0; i < productArr2.length; i++) {
        var cinbun = productArr2[i]
        var cinbunGlaze = cinbun.glaze
        var cinbunQuantity = cinbun.quantity
```

5. Event handling

I familiarized myself with onclick and onload event handling which was an integral part of the way my cart page materialized. I used the “onload” on the cart page which meant that the element it was attached to would call the function once the elements prior to it were loaded. It worked well as a means to have the cart page load automatically with the correct information without needing to have another click or initialization which would create extra steps for a customer. “Onclick” worked similarly to call a function that populated the array when “Add to Cart” was pressed.

```
<body onload=checkoutPageLoaded()>


cart.innerHTML += "<span class='delete' onclick= 'deleteProduct(" + i + ")'> Delete </span>"
```