Assignment 6B – John Han

<u>Reflection</u>

Honestly, everything about JavaScript was difficult. One of the biggest problems I faced when coding for JS was trying to create each row of the items selected in the shopping cart page. At first it was just lines of HTML and CSS of one row of the item in the cart page where it had no relationship with the previous product detail page, where users would be able to see the items selected. Therefore, it was necessary to bring the options selected from the previous page (quantity and glazing) and display it in the shopping cart page. I started by putting in the selection in the localStorage with each array holding both values for the quantity and glazing option. Thus, localStorage would have an array in an array like [["3", "Vanilla-Milk"]].

However, it was hard to bring out the HTML row for each array that was stored from the product detail page. From here, I decided to move the HTML code for the cart list into the JS where I was able to make direct changes. For every index number of the array, I gave an ID with '+i+' for every element that needed to change according to the selected value in the localStorage. In this case, the picture, quantity, glazing option and the price had to show accordingly from the localStorage. Therefore, I defined each element to variable to have the correct array for every index number. If I picked a variable to be [i][0], this would bring out the quantity number for every available index number inside of the localStorage.

As for [i][1], I wrote the code so that each selected glazing option would be represented in the dropdown choice as well as the image displayed in the cart page. At first, the image was the only thing that didn't show up. As I was debugging, I realized the name of the value of the dropdown list and case value were different because of capitalization. For example, one was written as "Double-chocolate" while the other had "Double-Chocolate". This made me notice how important it is to define a value with specific letters whether it's capital or lower case.

Programming Concepts

- Variables – I learned that setting up these containers for storing data values is concept I used the most when writing JavaScript. Almost every one of my functions started off with ‘var’ to set up the value which I reuse it very often. In the example, I would set the variable x as the integer value from the quantity ID which was the selected dropdown option.

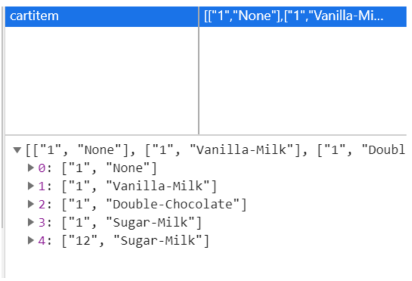
```
function myFunction() {
  var x = document.getElementById("quantity").value;
  document.getElementById("total").innerHTML=`Subtotal:$${3*parseInt(x)}.00`
   }
```

- HTML5 Web storage – This allowed me to save key and values in a web browser. This was a better choice for me since it stored data with no expiration date. Therefore, even when I close the tab, the data is still stored. I used the localStorage so that the number of the quantity of the item and the glazing option still remained even when I closed the screen. My example let me store the string value of quantity and glaze in a key called ‘cartitem’ only when the key itself had no value inside.

```
if (localStorage.getItem('cartitem') == null) {
     localStorage.setItem('cartitem', JSON.stringify([[qtynumber, glaze]]) )
}
```

- Array – Creating an array allowed me to put multiple values in one variable at a time. This was helpful in using different orders the user chose. Each order would be an array and the quantity & glaze options would be an array inside the order array. Adding from the equation above, I added:

```
else {var noempty = JSON.parse(localStorage.getItem('cartitem'))
      noempty.push([qtynumber, glaze])
      localStorage.setItem('cartitem', JSON.stringify(noempty))
   }
```



```
cartitem    [["1","None"],["1","Vanilla-Mi...

▼[["1", "None"], ["1", "Vanilla-Milk"], ["1", "Doubl
  ▶0: ["1", "None"]
  ▶1: ["1", "Vanilla-Milk"]
  ▶2: ["1", "Double-Chocolate"]
  ▶3: ["1", "Sugar-Milk"]
  ▶4: ["12", "Sugar-Milk"]
```

  - I used the push method to make another array from the existing one for quantity and glaze options.
  - This created an array that was easy to detect in the google inspector.

- DOM Events with HTML – When I reload the page or click a button, HTML DOM allowed JavaScript to react. These events helped me link these two languages whenever I’m interacting with the page. One example to use onload() to make the number of the items in cart icon keep staying there when reloading the page.

On HTML

```
<body onload="showcart()”>
```

On JavaScript

```
function showcart() {
   var cart = localStorage.getItem('cart');
   if (localStorage.getItem('cart') > 0) {
      document.getElementById(`cart-count`).innerHTML = `${cart}`
   }
}
```

- JSON – The JavaScript Object Notation helped me format the code to store data more efficiently across the network. I used JSON.stringify() to send data to the web server as it has to be a string. When receiving the data, I used JSON.parse() to turn it into JS object. Going back to the Web storage and array example, I made the values into a string when sending the data to the server by using JSON.stringify(). On the other hand, I used JSON.parse() when receiving data from the storage to add a new array into the existing server. I would stringify the data once more to store the data to the server, forming a new array.

```
     if (localStorage.getItem('cartitem') == null) {
           localStorage.setItem('cartitem', JSON.stringify([[qtynumber, glaze]]) )
        }
        else {var noempty = JSON.parse(localStorage.getItem('cartitem'))
           noempty.push([qtynumber, glaze])
           localStorage.setItem('cartitem', JSON.stringify(noempty))
   }
```