Reflection

One of the main challenges I encountered was how to uniquely identify each item in the shopping list for removing & updating an item. I overcame this by using *data-* attributes*. With this, I could easily store and retrieve product names and other useful information. After overcoming this challenge, I ran into another problem. I originally thought product names would be sufficient for uniquely identifying a row, but then I realized that color and fill selection are also two important factors that I need to take into account. For example, if the user adds two couch pillows with two different colors and/or fills, then they are technically two different items and should belong to two different rows in the item list in the shopping cart. Last but not least, another bug I had was getting null values from *document.getElementById("...")*. I then realized that this happens because the html was not generated yet when I ran the line. I solved this by moving the script link to the end of the file.

Programming Concepts

1. Web storage
    a. In this assignment, I used local storage to dynamically store and remove items that the users add to the shopping cart.
    b. Ex: `localStorage.setItem(`${name},${color},${fill}`, [price, quantity]);`
        i. In this example, the user is adding an item to the cart. I used the combination of product name, color, and fill as a unique key for each item, and price and quantity as the value. Then, when the user clicks on the cart, the showItems function will loop through the local storage and display every item.
2. Event
    a. In this assignment, I used the concept of event listener to monitor if the user is trying to remove or update an item from the shopping cart.
    b. Ex:

```
itemList.onclick = function (e) {
if (e.target && e.target.classList.contains("remove")) {
      const name = e.target.dataset.name;
     const color = e.target.dataset.color;
     const fill = e.target.dataset.fill;
     const key = `${name},${color},${fill}`;
     removeItem(key);
   }
 }
```

        i. This function will be called when the user clicks on the item list in the shopping cart. If they clicked on the remove button, then the function will remove the corresponding item.
3. Scope

a. In this assignment, I used both global and function scope.
b. Ex: `const itemList = document.getElementById("item-list");`
    i. I declared this variable as a global variable since the event listeners are constantly "listening" to the changes made to the itemList.
c. Ex: `const name = document.getElementById("product-name").innerHTML;`
    i. Whereas I declared this variable as a local variable inside the addToCart() function since we only have access to the product-name element when the user wants to add an item to the cart (from the product-details page).

4. Popup alert
    a. The three main popup alert functions in Javascript are alert(), confirm(), and prompt(). In this assignment, I used the alert function to notify the user whenever a new item has been added to the cart. Other popup alert
    b. Ex: `alert ("An item has been added to cart");`
        i. I called this function in my addToCart() function to give feedback to the user's action.
5. DOM
    a. I used Javascript DOM methods to access and modify the HTML DOM elements.
    b. Ex:
```
const divTotal = document.getElementById('total');
divTotal.innerHTML = "Total: $" + getTotal();
```
        i. In this example, I first used the *getElementById* method to access the element with ID "total" from the HTML DOM. Then, I used the *innerHTML* property to update the HTML element with the total price of all cart items.