Yunmin Oh
PUI

# Assignment 6B
# Reflection

Since this was my first time using javascript to make a functional ecommerce website, Assignment 6 was a bit challenging. The biggest challenge was to store the selected items into the local storage and find a way to retrieve it back by displaying the selected items on the shopping cart. Most importantly, I could not figure out how to modify the table contents I created in the cart.html file. After attending office hours and asking questions to my peers whose expertise is on web development, one solution I found was to modify the original cart.html file by relocating the contents of the table (class="itemdisplay") in the productdetail.js file as an innerHTML text. I also realized it is important to place functions such as updateCartNumber() and updateCartList() in the other functions as these are constantly getting updated.

## Programming Concepts

1. I learned that document method getElementbyId() can change the matching element (style including background color, color, background image or innerHTML text, etc). In order to allow the user to modify options on the product detail page (the size and color), these methods allow the product detail page to be updated according to the user's selection.

Ex1)
```
document.getElementById('p-color').innerHTML = ": Blue";
```
Ex2)
```
document.getElementById('main_img').style.backgroundImage =
"url(muddypaw_img/item3.jpg)";
```

2. In order to ensure that previous information can be saved and updated when the new page gets loaded, I learned that using Window's load event can make this work. This load event executes when the whole webpage (HTML) has fully loaded including all dependent sources including JS files, CSS files, and images. By placing the updateCartNumber () and updateCartList () functions inside the window's load event, these functions will be updated as the new page is fully loaded.

Ex)
```
window.addEventListener("load", () => {
 updateCartNumber();
 updateCartList();})
```

3. In order to save the selected items into the storage and make it available to be retrieved, I used localStorage.setItem(key, JSON.stringify(array)) and localStorage.getItem(key). First, localStorage only supports strings so either JSON.stringify() or JSON.parse(key) is needed. Since the key is "cart" and an array is "cart.list," I placed them respectively.

Ex)
```
localStorage.setItem("cart", JSON.stringify(cart.list));
```
Ex)
```
return cart.list = JSON.parse(localStorage.getItem("cart") ||
'[]');
```

4. I learned that using document method querySelector() returns the first element within the document that matches the specified selector. Since I had to get the first span element in the class total to change the value of the total price, I used document.querySelector('.total span').

Ex)
```
document.querySelector('.total span').innerHTML =
`$${totalPrice.toFixed(2)}`
```

5. I learned different types of array methods. Also known as Iteration Methods, these methods operate on every item in an array one at a time. First, I learned that this arrow function expression (=>) is used in the latest version of JS instead of a traditional function expression. In order to make a function that allows the user to add the item to the list, it first needs to check if the selected color and size are matching to the ones already existing in the array. I learned the find() method executes the function once for each element present in the array and if it finds an array element where the function returns a true value, it returns the first value in an array that passes a given test. In order to remove an item from the array, I used a filter() method that creates a new array with the elements that pass the result of a given test. Once the item is removed from the array, the index of an item is removed by this filter method.

Ex)
```
let _item = cart.list.find(({ color, size }) => item.color
=== color && item.size === size);
```
Ex)
```
cart.list = cart.list.filter((_, _index) => index !== _index);
```