Reflection

The biggest challenge was having to redo my coding part after completing a version without local storage. Because of my single-page web design and an overlay shopping cart design, I was able to fulfill homework 6b instructions without using local storage. I finished the coding part early and later found out in the slack channel that we had to use local storage in this homework. It was mentally hard to accept the fact that I had to redo something I already checked off in my to-do list. With a reluctant attitude, I tried to keep my original structure and adapt it to local storage. I was not able to make any progress because my mind was still stuck in the previous framework. Finally, I decided to start with a clean slate on a new js file and got it to work quickly after.

In situations like this, It is important to first set my mind straight. If I can't change the situation, I need to quickly adapt to the new one and find the best approach. If I still get stuck for a long time, start with a clean slate. Most importantly, keep a positive attitude. These tips can really help in unexpected situations.

Programming Concepts

Programming Concept 1: Variables and Constants
Variables and constants define areas in memory in which values (data) are stored. Variables hold values that can be modified, whereas constants hold values that remain unchanged.
In my case, I use constants (*let*) to store cart items retrieved from local storage in *displayCart()* since I only need to read the information. On the other hand, I use variables (*var*) to store cart items in *updateCartItem(index)* because I need to update the items.

Programming Concept 2: Casting
Casting means changing the data type of a piece of data from one type to another.
For example, I use the *JSON.stringify()* method every time I need to store an item in local storage, because only strings can be sent to a web server. The *stringify()* method converts a JavaScript object or value to a JSON string.

Programming Concept 3: Arithmetic Operators
Arithmetic operators are used to perform calculations. For example, a single item's price is calculated using multiplication (*) operator depending on the quantity selected. To calculate cart total price, addition (+) operator is used to add prices of every item in the cart.

Programming Concept 4: *.map()*
.map() creates a new array with the results of calling a provided function on every element in the calling array. It is similar to *forEach(),* which changes the original array by executing a provided function once for each array element. Normally, I would always prefer non destructive methods.

However, it does not matter which method to use in my case (*displayCart()*) because I only read the data rather than alter it. I still chose *.map()* because it is faster.

Programming Concept 5: *.splice()*

.splice() removes, adds, or replaces elements in an array in place, or in other words, destructively. In my code, I use *.splice()* to update a specific item in the cart by first getting the element, updating the element, and replacing the element with the updated element. The example code for replacing an element in cart is *cart.splice(index, 1, element)*. For reference, the syntax is *splice(start, deleteCount, item1)*.