Kate Zhao
Assignment 6B: Finishing the shopping cart

Github Repo: https://github.com/k8zhao/pui-hw-6
Github Pages: https://k8zhao.github.io/pui-hw-6/

Reflection

The biggest challenge I ran into was thinking about what approach to use to update the cart. I decided to create a 'Pillow' object so that I could save a lot of information (e.g. color, material, type) each time a user wanted to add a pillow to the cart. I created a pillow object constructor at the beginning of my code that took in all of the necessary attributes. I created a pillow object every time the 'add to cart' button was clicked and saved this object in local storage. However, I quickly realized that my cart could only hold one pillow at a time. If I added more than one pillow to the cart, only the last pillow would be displayed when I went into my cart. This was because every time I clicked 'add to cart', I would overwrite the previous pillow and save the most recent pillow in local storage. To solve this problem, I declared a global array called "cart_items" that would store all the pillows added to cart. Now, when the 'add to cart' button is clicked, a new pillow object will be created, pushed into the "cart_items" array and then saved to local storage.

Another problem I encountered was figuring out how to delete the product from the cart. I was able to visually delete it on the website by using item.removeChild() but when I refreshed the page the deleted pillow would show up again. This was because the pillow wasn't deleted from the local storage. I worked with my TA Kirabo to add another attribute in my pillow constructor that took in an id. This way, I could identify which pillow to delete from the array when the delete button was pressed.

Programming concepts

Arrays
I learned that arrays could be used to store multiple objects. I used a global array to store my pillow objects. Everytime the 'add to cart' button was clicked, a new pillow object would be created using the information inputted on the screen (e.g. color, material, type) and pushed into this array. Because this array was populated with all the pillow objects users added to the cart, I could display all of the pillow objects in my cart.html file.

Loops
Loops can be used to cycle through things and make your code more organized and easier to manage. I used a for loop to iterate through my array 'cart_items'. This array held all of the pillow objects users added to cart. In my cart page, I wanted to display all of the pillows added to cart, so I looped through this array and created elements with document.createElement so I could show each pillow's details in an aesthetically pleasing way to users in the cart.

DOM Navigation

We learned in class that everything in an HTML document is a node. I used javascript to traverse through the node tree and access different elements. For example, to display pillows in the cart, I created many elements (e.g. divs, buttons, headers) to display my pillow objects in a specific way. To add it to the existing cart page, I needed to use .appendChild to add all the different divs as children to the larger container on the page. Another example of this is when I had to delete pillows from the cart. My delete button was nested within its own div and that div was placed in a larger div that held all the details of a singular product. In order to delete all the details of the product, I needed to delete the largest div, so this translates to: onclick = "deleteItem(this.parentNode.parentNode)" where 'this' refers to the button, '.parentNode' refers to the div that held the button, and the second '.parentNode' refers to the largest div that held all the pillow details. In short, the outline code to display a singular pillow looked like this:

```
<div class= "product_wrapper">
        <div> some info </div>
        <div> some more info </div>
        <div>
                <button> MY BUTTON </button>
        </div>
</div>
```

Objects

Objects allow you to store multiple pieces of information. I decided to use objects in this project because there were many properties I needed to keep track of and access for each pillow. I have a pillow constructor that takes in id, type, color, material, quantity, and image. This made it easy for me to quickly create a new pillow every time a user clicked on the 'add to cart' button. I would use all the information inputted by the user on the detail page to create these pillows before pushing them into the array and saving to local storage.

Local Storage

I used local storage to save:

1. The cart counter (number that indicates how many items in the cart)
2. Pillows added to cart

Local storage was really useful here because without it there would be no way to display the pillows added to cart. Before I used local storage, I would click the add to cart button but when I clicked into the cart page, it would be empty. Without saving to local storage, all that information would just disappear. After pushing new pillows into my cart_items array, I used JSON.stringify() to convert my array to a JSON string and localStorage.setItem() to save that array to local storage. In a separate function (onload function for the cart page), I retrieved the array from local storage using localStorage.getItem() and JSON.parse to parse it. I used the same logic for the cart counter.