Sarah Hand

PUI Homework 6b Write Up

Git Hub Repo Link: https://github.com/sarahj930/sarah_hand_pui/tree/main/homework_6b

Home Page Link*: https://sarahj930.github.io/sarah_hand_pui/homework_6b/MuddyPaws.html

*Note: Please open this in an Incognito window in Chrome to clear local storage from last session (6A).

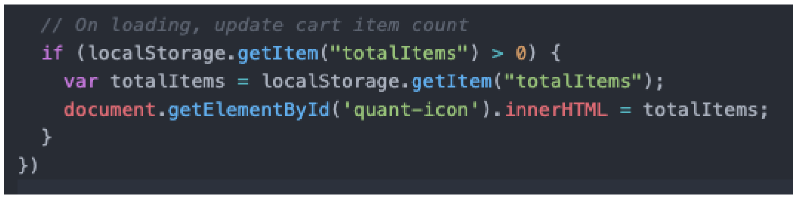
Reflection:

The biggest problem I had to figure out during this process was how to accurately store and update the cart in local storage. I started with just storing the total number of items in the cart and displaying this in an icon on every page of the site. The first couple of ways I tried to do this ended up with a huge number being displayed instead of the accurate count when I would go to other pages. I figured out that I needed to use parseInt() when accessing data in local storage and doing arithmetic with it. Otherwise, it would be treated as a string and concatenate the numbers together rather than add them. In the future, I will know to use parseInt() when handling numbers through local storage.

Then, when trying to store the whole cart in local storage, I had some trouble trying to figure out what kind of data structure would be best to use. I initially tried a 2D array (cart[][]) but found that 2D arrays in JavaScript did not behave the way I wanted them to. So, I decided to create a kind of table that uses an array for indexing and has an object with fields in each index. I did this because I needed a way to store the color, size, and quantity of each product. Each index of the array “cart” had fields for product name, color, size, and quantity that could then later be accessed as such: cart[index].field. I then stored this whole array in local storage and would access it, update it, and then reset it in local storage whenever and wherever needed. In the future, I will definitely use this method to store more complex tables of data because it was a very intuitive way to handle the data.

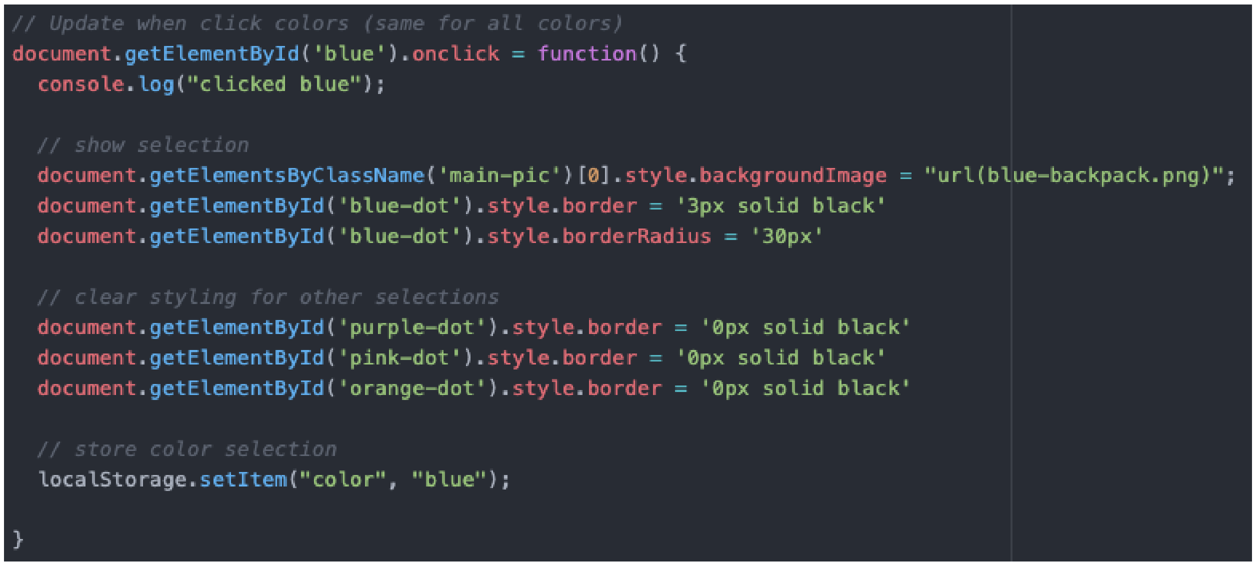
Another issue I came across was trying to figure out how to remove an element from the cart. I initially tried to only remove the element from the DOM when clicking a button, but this did not work because the underlying local storage was still the same and the way I had organized the DOM made it very complicated to remove all of the right elements and not mess everything up. So, I decided to simply use a splice() method on my locally stored cart array to remove the desired element and then repopulate the field using the updated array. In the future, however, I might try to organize my DOM so that all of the elements I may want to add or remove are inside of one div, which makes it easier to just access the desired div and remove it from the DOM.

One programming concept I learned to use was local storage. I had never used this before this assignment and found, once I knew about the little quirks such as parseInt() and needing to use JSON.stringify, it very simple and helpful to keep data consistent across pages in the website. For example, I used it to consistently update the number of items in the cart when opening every page.



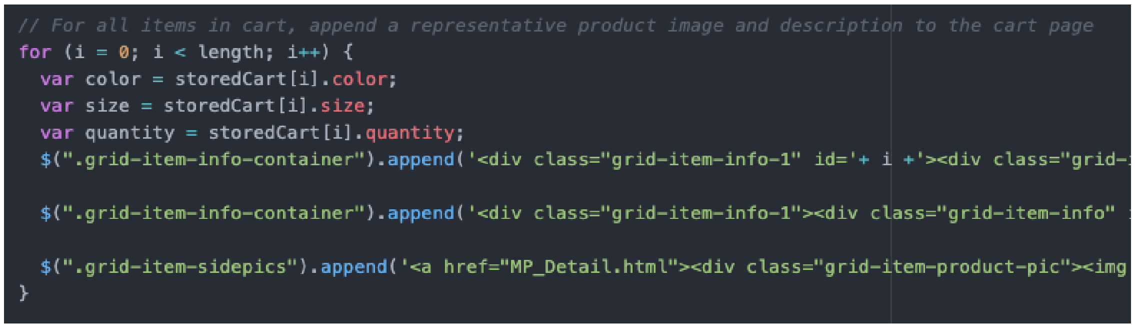
```
  // On loading, update cart item count
  if (localStorage.getItem("totalItems") > 0) {
    var totalItems = localStorage.getItem("totalItems");
    document.getElementById('quant-icon').innerHTML = totalItems;
  }
})
```

Another programming concept I learned was creating functions that are activated when you click a certain item (with a certain id) on the page. I used this quite a lot on the product details page to dynamically change the page when the user interacted. For example, I used functions that would activate when you clicked each color and size that would update the DOM and the locally stored value.



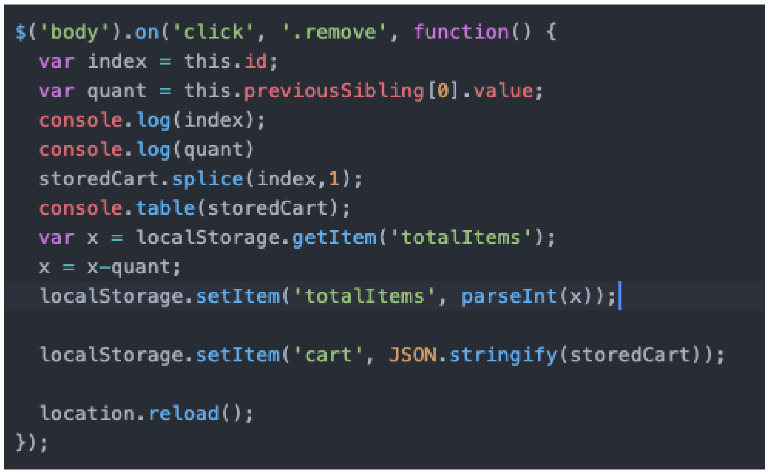
```
// Update when click colors (same for all colors)
document.getElementById('blue').onclick = function() {
  console.log("clicked blue");

  // show selection
  document.getElementsByClassName('main-pic')[0].style.backgroundImage = "url(blue-backpack.png)";
  document.getElementById('blue-dot').style.border = '3px solid black'
  document.getElementById('blue-dot').style.borderRadius = '30px'

  // clear styling for other selections
  document.getElementById('purple-dot').style.border = '0px solid black'
  document.getElementById('pink-dot').style.border = '0px solid black'
  document.getElementById('orange-dot').style.border = '0px solid black'

  // store color selection
  localStorage.setItem("color", "blue");

}
```

I also learned how to use for loops to append items to the DOM for every item in an array. I initially had some trouble with this as it was difficult to get the appended items to appear how I wanted them to appear. However, I took what I designed in the HTML and copied and pasted it into the JavaScript in quotes and added some components that would change with each rotation of the loop (assigning a unique ID to each component based on the i value in the loop).



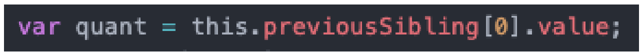
```
// For all items in cart, append a representative product image and description to the cart page
for (i = 0; i < length; i++) {
  var color = storedCart[i].color;
  var size = storedCart[i].size;
  var quantity = storedCart[i].quantity;
  $(".grid-item-info-container").append('<div class="grid-item-info-1" id='+ i +'><div class="grid-

  $(".grid-item-info-container").append('<div class="grid-item-info-1"><div class="grid-item-info"

  $(".grid-item-sidepics").append('<a href="MP_Detail.html"><div class="grid-item-product-pic"><img
}
```

I also learned how to use ajax to allow different buttons that were being dynamically added and removed from the DOM to call the same function. I used this line (below) to connect all of the buttons with the class “remove” to this function which removes the desired item from the cart array. I found that using ajax in this way was the only way to attach multiple buttons that were being dynamically added and removed from the DOM to a single function.



```
$('body').on('click', '.remove', function() {
  var index = this.id;
  var quant = this.previousSibling[0].value;
  console.log(index);
  console.log(quant)
  storedCart.splice(index,1);
  console.table(storedCart);
  var x = localStorage.getItem('totalItems');
  x = x-quant;
  localStorage.setItem('totalItems', parseInt(x));

  localStorage.setItem('cart', JSON.stringify(storedCart));

  location.reload();
});
```

I also learned how to use JavaScript to traverse the DOM and access an item related to the item the user is interacting with. For example, to remove an item from the cart, I had buttons next to every item. When the user clicks the button, I used this.previousSibling[0].value to get the value of the item next to the button in the DOM on the same level.



```
var quant = this.previousSibling[0].value;
```

Aid Cited:

w3schools.com