PUI HW6B: Ally Hopping

Reflection

Admittedly, I ran into a lot of challenges while building this portion of the site, so I'll focus on describing the biggest, most time-consuming ones. My first challenge was figuring out how to properly create an object containing the necessary attributes for each Dog Harness a user added to their bag. The size and color variables were initially showing up as undefined in the object array, and I couldn't figure out why since it seemed I was properly referencing the clicked item in my code. After troubleshooting for a while using console.log, I realized that rather than trying to grab the value or name of the item, I could just grab and save the HTML text that was properly changing to display the selected size and color on the product page. There is probably a cleaner way to do this, however I wasn't able to find one in the time I had to complete the assignment so felt this was the best solution for the time being.

Once I had the items properly adding to the Bag and displaying on the Bag page, I realized that when I went back to the product page, the existing items were disappearing from the Bag even though they were still being stored in local storage. After using a series of console.log statements to try to figure out what each of my variables held and where the error was being introduced, I realized that I was only calling the items to save to the Bag when the user clicked on the Bag icon in the main nav. To fix this, I instead needed to save the items to the Bag (i.e. the item array) when the user clicked the Add to Bag function.

After I had fixed this, I then needed to figure out how to make the Remove buttons work on the Bag page. In 6A I had a lot of trouble styling the Bag items, and my workarounds for that came back to bite me due to the complexity of the scaffolding of divs and classes/IDs assigned to the elements in each Bag item; through a series of trial and error, I realized I had to reference .parentElement three times before being able to access the top div of each item. bI first tried to simply remove the selected item from the item array using .splice and calling the bagLoaded() function to display the updated bag, however this did not work as I intended. At first, when I clicked the remove button, the items were duplicating instead of removing which was very frustrating since this was the opposite of what I was trying to do. I walked through my bagLoaded() function several times and used a series of console.log statements to debug. This helped me realize that I needed to re-save the updated array into local storage, as it was currently being duplicated rather than edited. I then called 2 additional functions to update the bag subtotal and number of Bag items displayed in main nav.

Overall this assignment taught me an enormous amount about programming principles and how to think through the logic of the coding language both before I start to write the program and after I realize it's not working as I intended. Since this can certainly be frustrating at times, this assignment also taught me how to use my resources (especially console.log statements and stackoverflow.com) to debug and find new ways to solve problems.

Programming Concepts

1. Local Storage

   I used local storage (stringify) to save the items a user added to their bag from the Dog Harness page and to parse them out to access them and display them on the Bag page. I also used it to re-save the edited Bag items when a user adds or removes items from their Bag. Using local storage allowed me to preserve the items a user had added to their Bag during that browsing session, so they could browse freely throughout the rest of the site, and more items to their Bag, and still return to their Bag to checkout.

2. How to access and manipulate objects in an array

   I created an object for each Dog Harness a user adds to their bag and added them to an array. I used local storage to save them and a for loop to iterate through each object in the array to access them. To remove the objects, I used a for loop to iterate through each item and remove the one the user selected from the existing array of items.

3. Built-in HTML DOM methods like getElementById and innerHTML

   These methods proved extremely helpful in accessing and modifying specific elements in the HTML. For example, I used .getElementById to access each of the size buttons on the Dog Harness product page and .innerHTML to change the displayed size to the one the user last clicked on. Practicing these methods also gave me a deeper understanding of how to use IDs in the HTML, as for previous assignments, I had mostly created classes for styling because many of my elements were repeated.

4. Dynamically adding new elements to the DOM

   I didn't know that you could add entire blocks of HTML including references to the CSS (i.e. classes and IDs) in the JavaScript file before completing this assignment. I applied this concept using the .innerHTML method to display and properly style each item in the Bag. I could then break the HTML to dynamically add attributes from the object saved in local storage (i.e. itemPrice) so each item displayed the correct information as saved in the item array.

5. How to use properly use local and global variables

   Before this assignment, I didn't have a comprehensive understanding of when to use local and global variables. Other than commonly used variables like "i" for for loops, I thought most variables should be global in case they need to be referenced later in the code. However, I realized local variables are crucial to programming because they allow you to reuse the same variable later in the code without worrying about what was done earlier. For example, using local variables for all of my item attributes allowed me to

create new items using the same attributes with different values for each product a user adds to their Bag.

Other Notes

I wanted to note that I have the following error showing up in my CSS, however this is not an error because I purposely made the border match the background color for the hover state of these buttons.

URI : https://hoppinga.github.io/PUI/HW6/6B/style.css

106	`.hero-image .button-white:hover`	Same color for `background-color` and `border-color`
177	`.community-adventure-home .black-button:hover`	Same color for `background-color` and `border-color`
216	`.bottom-banner .button-white:hover`	Same color for `background-color` and `border-color`
539	`.add-to-bag-button:hover`	Same color for `background-color` and `border-color`
539	`.continue-to-checkout:hover`	Same color for `background-color` and `border-color`
600	`#black-button:hover`	Same color for `background-color` and `border-color`

I also have the following error in my HTML, however I do not want my color buttons to have a value because they should not be labeled. The color of the button serves as the label, and the name of the color appears when each button is clicked.

1. **Error** **Element** `input` **with attribute** `type` **whose value is** `button` **must have non-empty attribute** `value`.
   From line 94, column 17; to line 94, column 114
   `<input type="button" id="color-button-strawberry" class="color-buttons" name="colors-dog-harness">↵`
2. **Error** **Element** `input` **with attribute** `type` **whose value is** `button` **must have non-empty attribute** `value`.
   From line 95, column 17; to line 95, column 124
   `<input type="button" id="color-button-fire-orange" value="" class="color-buttons" name="colors-dog-harness">↵`
3. **Error** **Element** `input` **with attribute** `type` **whose value is** `button` **must have non-empty attribute** `value`.
   From line 96, column 17; to line 96, column 123
   `<input type="button" id="color-button-crazyberry" value="" class="color-buttons" name="colors-dog-harness">↵`
4. **Error** **Element** `input` **with attribute** `type` **whose value is** `button` **must have non-empty attribute** `value`.
   From line 97, column 17; to line 97, column 123
   `<input type="button" id="color-button-blackberry" value="" class="color-buttons" name="colors-dog-harness">↵`