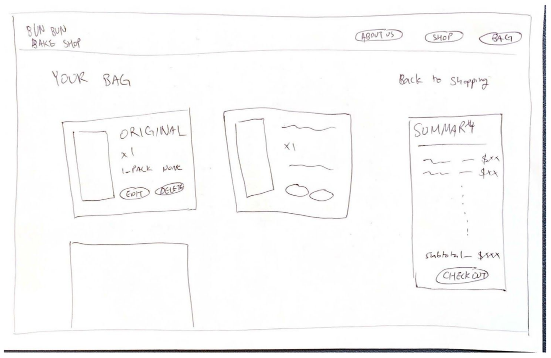
PUI Assignment 6 Reflection

Aaron Lee

Github Repo:
https://github.com/thelittlefloor/PUI-Aaron-Lee.git
Hosted Website:
https://thelittlefloor.github.io/PUI-Aaron-Lee/
https://thelittlefloor.github.io/PUI-Aaron-Lee/assignment_6/homepage.html

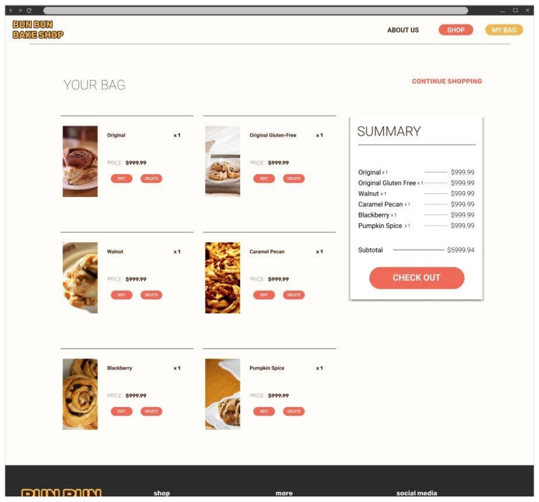
Assignment 6A Reflection (**Reflection for Assignment 6B starts at Page 5**)

Shopping Cart: Low Fi Prototype



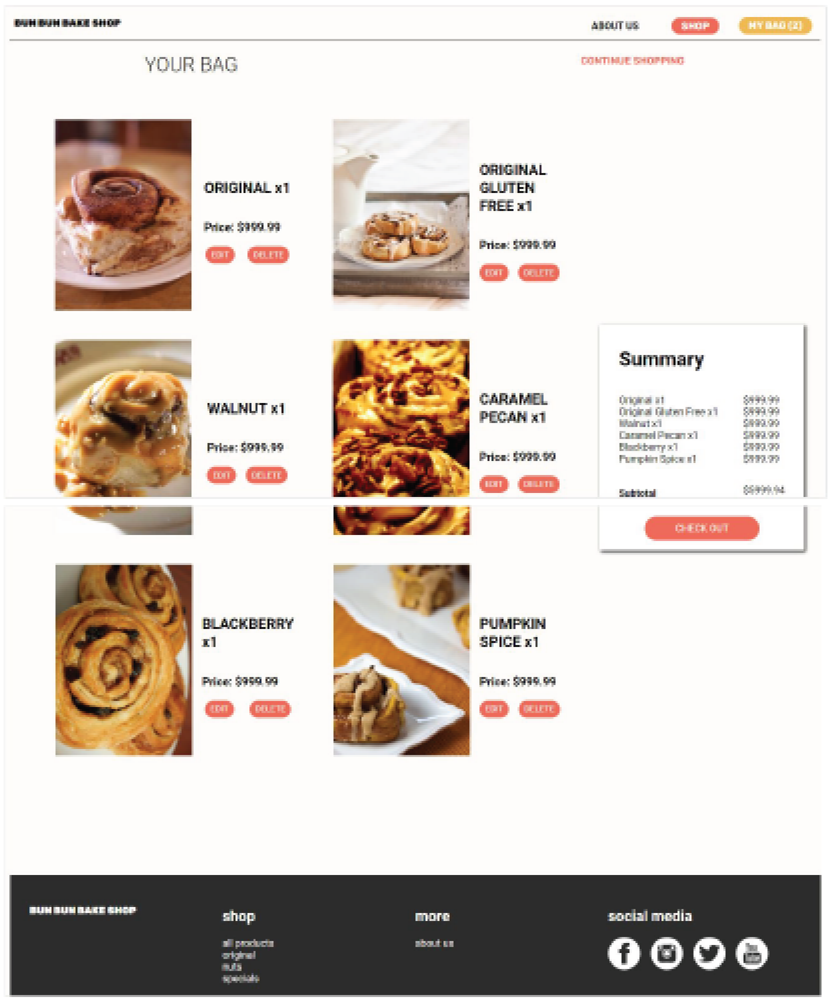
Paper prototype of shopping cart page made as a continuation of Assignment 2, which did not include the low fidelity paper prototype for the cart page. It is simple card layout design that shows the items in a grid to the left, and a summary of the order on the right. The right card layout also has the button for 'check out'.

Shopping Cart: High Fi Prototype



The high fidelity prototype of the shopping cart was included in my submission of Assignment 3. It is practically a literal translation from the paper prototype to figma. Also note that there is 'continue shopping' button that will take the user back to the browse page. The link to the figma page is given here:
[https://www.figma.com/file/tXLrl5F0lekpn6jRVISMBO/PUI-Assignment-3-Aaron-Lee?node-id=0%3A1](https://www.figma.com/file/tXLrl5F0lekpn6jRVISMBO/PUI-Assignment-3-Aaron-Lee?node-id=0%3A1)

HTML + CSS Implementation

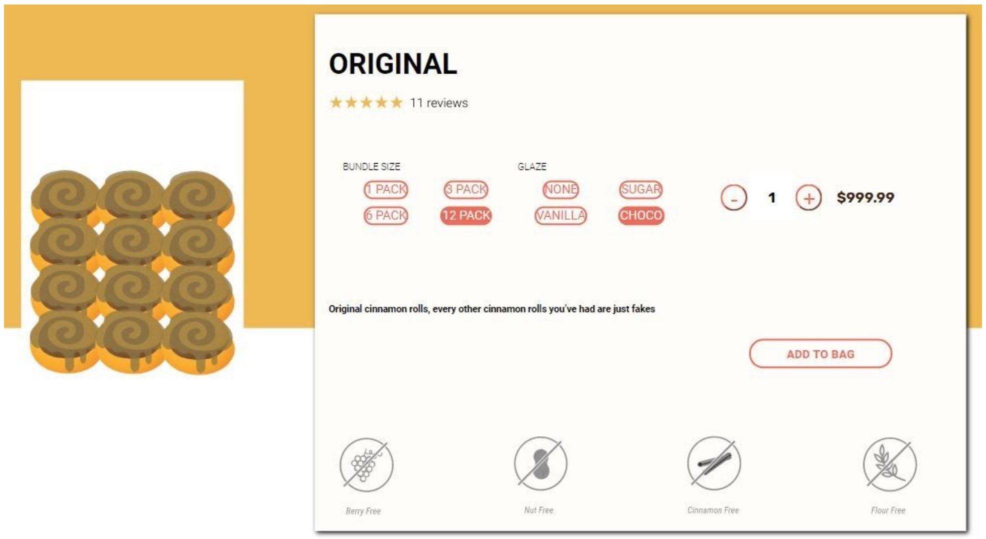


Summary

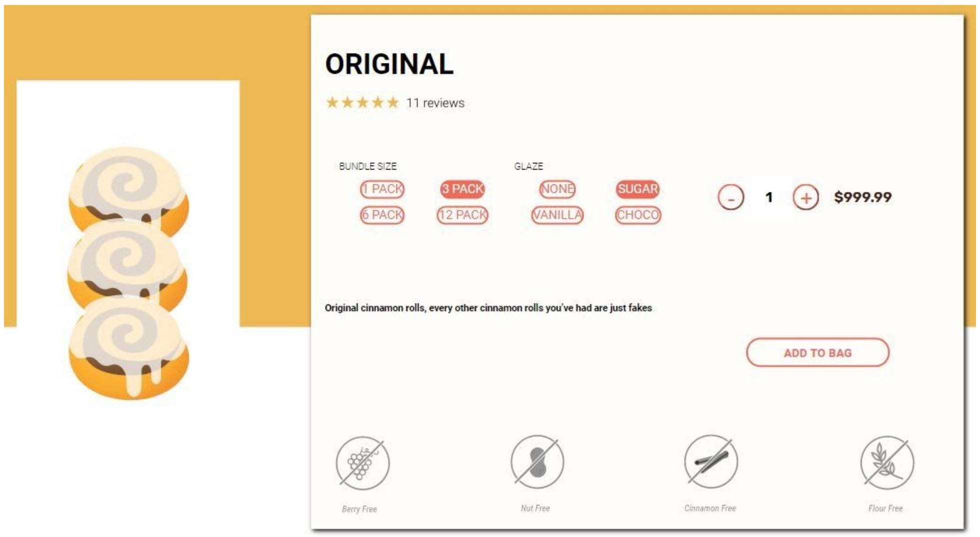
| | |
|---|---|
| Original x1 | $999.99 |
| Original Gluten Free x1 | $999.99 |
| Walnut x1 | $999.99 |
| Caramel Pecan x1 | $999.99 |
| Blackberry x1 | $999.99 |
| Pumpkin Spice x1 | $999.99 |
| Subtotal | $5999.94 |

The HTML and CSS implementation for the assignment 6A submission is a static page, adding items from the product detail page will not change it.

Product Detail Page Modify Update

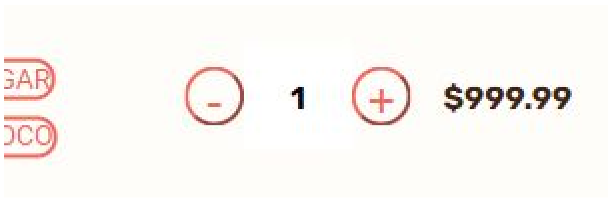


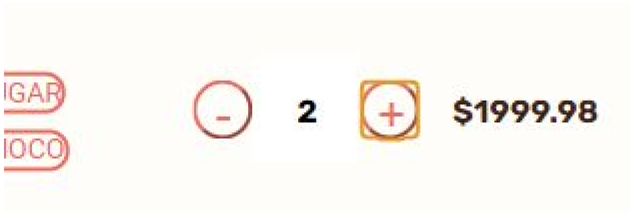
Original cinnamon rolls, every other cinnamon rolls you've had are just fakes



Original cinnamon rolls, every other cinnamon rolls you've had are just fakes

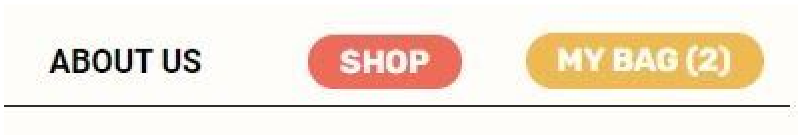
The image to the left of the product detail page will change when the user clicks on the 2 radio button groups for bundle size, and glaze type.





Incrementing the quantity also changes the calculated price.

<u>Number of Items in Bag Update</u>

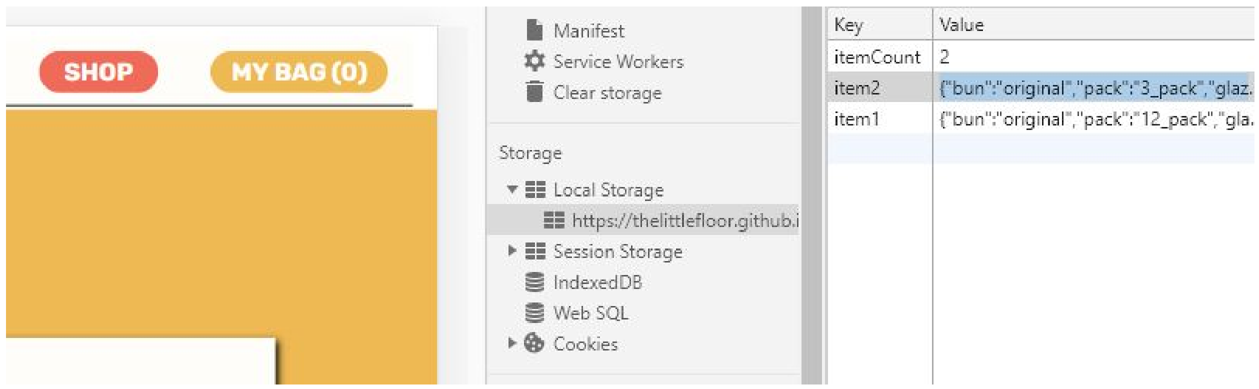


Clicking ‘ADD TO BAG’ in the product detail page will increase the number of items in the order. Note the exact items are not dynamically displayed in the shopping cart page, but they are stored as objects in local storage. Like such,
key: “item1”,
value: “{"bun":"original","pack":"3_pack","glaze":"sugar","quantity":2}”

The number of items in the order is also stored in local storage. Like such,
key: “itemCount”,
value: “2”

The way items are counted are by input, every click of ‘ADD TO BAG’ constitutes an added item in the order.

There is currently a bug to this feature, where the number of items indicated shows as ‘0’, when the user leaves the product detail page and returns to the product detail page, even when the local storage variable says otherwise.



| Key | Value |
|---|---|
| itemCount | 2 |
| item2 | {"bun":"original","pack":"3_pack","glaz |
| item1 | {"bun":"original","pack":"12_pack","gla |

<u>Other</u>

Unfortunately, I was not able to straighten out the issue regarding the css not applying evenly to github pages and vs code liverserver (this can be seen in homepage) that I mentioned in Assignment 5. Also note, that all the functionality added for Assignment 6 was applied to the **“Original” cinnamon bun only**, the other product pages will not have the added functions.

**Assignment 6B Reflection**

Issues/Bugs Encountered

There were many issues and obstacles that came up during the development of Assignment 6B, which I will describe in chronological order.

The first step to completing the assignment was to display the items added to the bag when the user navigates to the My Bag page. Currently, the item objects was stored as strings in local storage along with the total item count. When I started to code the dynamic creation of html elements in javascript, I realized that the objects did not have enough information. In the product detail page, I kept track of the product image using numerical keys mapped to the product options. While I stored the individual options chosen when adding the item to the bag, I forgot to store the image key. This was an easy fix, where I simply included the key in the item object when storing it to local storage.

Dynamically adding the html elements did not provide much in the way of obstacles, simple web searches were enough to answer any syntax questions and so on. I simply created an item_array of objects from the local storage by using JSON.parse on the serialized objects, then created html elements one at a time.

After adding the elements, it was time to implement the removal of the items by clicking the "DELETE" button that accompanies every item's display on the My Bag page. This is where a few issues came up. The original plan was to remove the item that corresponded to the triggered delete button from the item_array, then update the cart. I realized that I needed a way to distinguish between the different delete buttons. I rectified the issue by adding a value tag to the delete buttons within the for loop that created the html elements. The value would be the index of the for loop that created the button element. This way, the delete button stores its corresponding index in item_array.

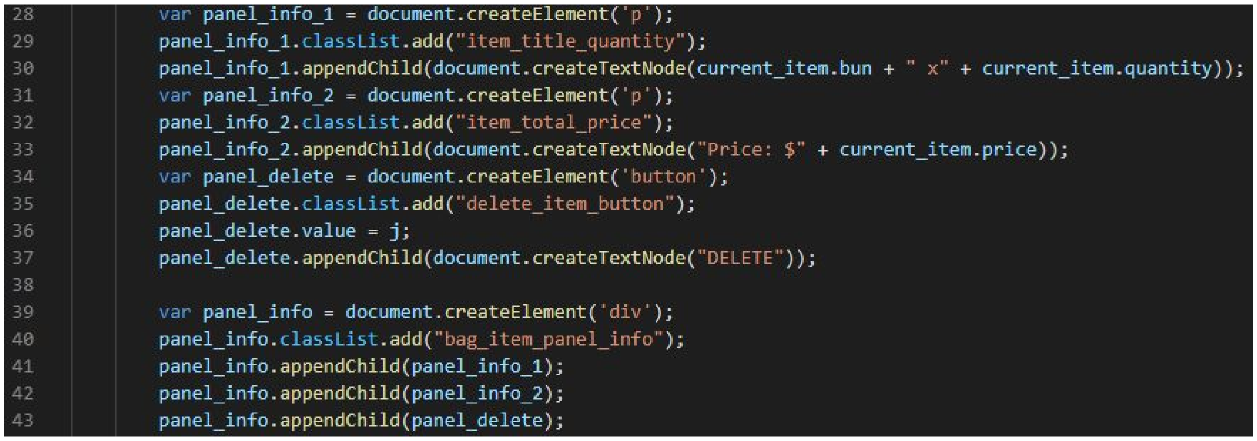
I then removed the item from the item_array using the splice method, then ran the updateCart function. Note that the updateCart function is the function that creates the html elements. A bug surfaced where the delete button triggered the creation of new items in the bag. There were two problems that were beneath the surface. The first and easily fixed one was that I needed to clear the html div before calling updateCart(). The second was that I needed to update the local storage as well as item_array, because the updateCart function works by calling JSON.parse on the serialized local storage strings. There were a couple adjustments to fix this issue. First, the item_array was a global variable while updateCart was the only function using it, but now that the delete button event listener used item_array as well, it cannot be a global variable. I worked around this issue by recreating the item_array in both the updateCart function and the event listener. Second, I updated the local storage by clearing the storage, then re-storing the serialized objects from the spliced item_array. Now the items can be removed from the My Bag Page. I also made sure to update the item count at the navigation bar as well.

There was a minor issue where the delete button would only work once on the page, but this was quickly resolved by re-adding the event listeners to the buttons after updating the cart.

A bug that I was not able to fix was that occasionally, clicking the delete button on an item would delete 2 items from the bag, not just the single one.

## Programming Concepts

1. DOM Creation and Modification

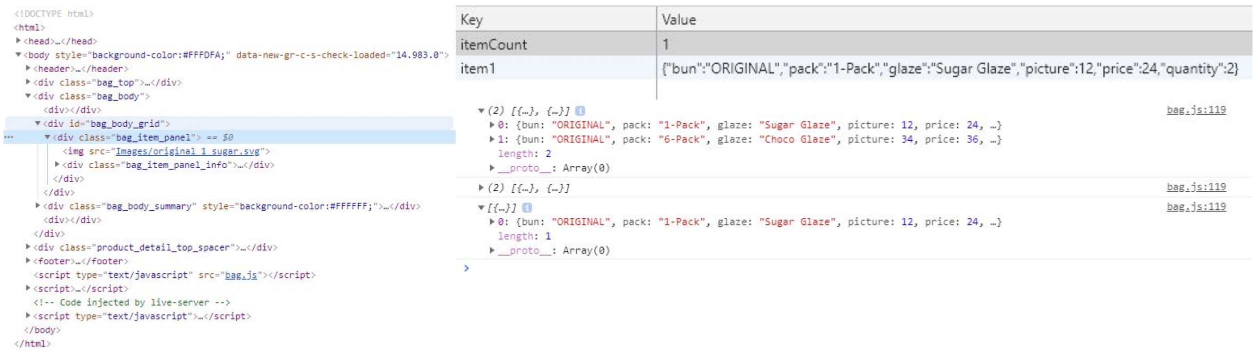


```
28          var panel_info_1 = document.createElement('p');
29          panel_info_1.classList.add("item_title_quantity");
30          panel_info_1.appendChild(document.createTextNode(current_item.bun + " x" + current_item.quantity));
31          var panel_info_2 = document.createElement('p');
32          panel_info_2.classList.add("item_total_price");
33          panel_info_2.appendChild(document.createTextNode("Price: $" + current_item.price));
34          var panel_delete = document.createElement('button');
35          panel_delete.classList.add("delete_item_button");
36          panel_delete.value = j;
37          panel_delete.appendChild(document.createTextNode("DELETE"));
38
39          var panel_info = document.createElement('div');
40          panel_info.classList.add("bag_item_panel_info");
41          panel_info.appendChild(panel_info_1);
42          panel_info.appendChild(panel_info_2);
43          panel_info.appendChild(panel_delete);
```

Figure 1. Lines 28 - 43 of bag.js

A key concept I learned and used was the dynamic creation and modification of the DOM. This concept was essential for the dynamic display of the items within the My Bag page. Figure 1, shows how the grid layout was filled with the div element that housed the item's information including quantity, price, and the delete button.

2. Dev Tool Debugging



```
<!DOCTYPE html>
<html>
 ▸<head>…</head>
 ▾<body style="background-color:#FFFDFA;" data-new-gr-c-s-check-loaded="14.983.0">
   ▸<header>…</header>
   ▸<div class="bag_top">…</div>
   ▾<div class="bag_body">
      <div></div>
     ▾<div id="bag_body_grid">
…      ▾<div class="bag_item_panel"> == $0
          <img src="Images/original 1 sugar.svg">
         ▸<div class="bag_item_panel_info">…</div>
        </div>
      </div>
     ▸<div class="bag_body_summary" style="background-color:#FFFFFF;">…</div>
      <div></div>
    </div>
   ▸<div class="product_detail_top_spacer">…</div>
   ▸<footer>…</footer>
    <script type="text/javascript" src="bag.js"></script>
   ▸<script>…</script>
    <!-- Code injected by live-server -->
   ▸<script type="text/javascript">…</script>
  </body>
</html>
```

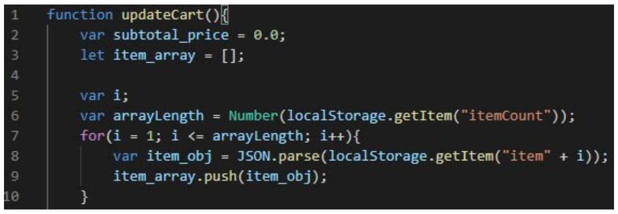
| Key | Value |
|---|---|
| itemCount | 1 |
| item1 | {"bun":"ORIGINAL","pack":"1-Pack","glaze":"Sugar Glaze","picture":12,"price":24,"quantity":2} |

```
▾(2) [{…}, {…}]                                                                          bag.js:119
  ▸0: {bun: "ORIGINAL", pack: "1-Pack", glaze: "Sugar Glaze", picture: 12, price: 24, …}
  ▸1: {bun: "ORIGINAL", pack: "6-Pack", glaze: "Choco Glaze", picture: 34, price: 36, …}
   length: 2
  ▸__proto__: Array(0)
▸(2) [{…}, {…}]                                                                          bag.js:119
▾[{…}]                                                                                   bag.js:119
  ▸0: {bun: "ORIGINAL", pack: "1-Pack", glaze: "Sugar Glaze", picture: 12, price: 24, …}
   length: 1
  ▸__proto__: Array(0)
>
```

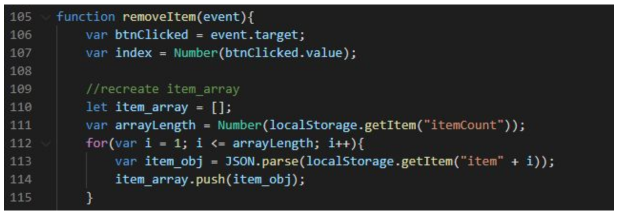
Figure 2. Elements (left), Local Storage (top right), and Console (bottom right) of Dev Tool

The developer's tool was used throughout the assignment to debug issues that came along. There are numerous examples of this, but to state one was the aforementioned issue where the delete button added items, not remove them. To resolved that issue I had to console.log() the item_array onto the console, assess the button value within the html element, and compare the item_array to the local storage at the same time. I would not have been able to do this without the developer''s tool.

3. Scope



```
1   function updateCart(){
2       var subtotal_price = 0.0;
3       let item_array = [];
4
5       var i;
6       var arrayLength = Number(localStorage.getItem("itemCount"));
7       for(i = 1; i <= arrayLength; i++){
8           var item_obj = JSON.parse(localStorage.getItem("item" + i));
9           item_array.push(item_obj);
10      }
```

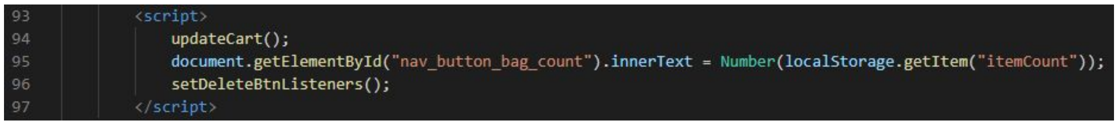


```
105 function removeItem(event){
106     var btnClicked = event.target;
107     var index = Number(btnClicked.value);
108
109     //recreate item_array
110     let item_array = [];
111     var arrayLength = Number(localStorage.getItem("itemCount"));
112     for(var i = 1; i <= arrayLength; i++){
113         var item_obj = JSON.parse(localStorage.getItem("item" + i));
114         item_array.push(item_obj);
115     }
```

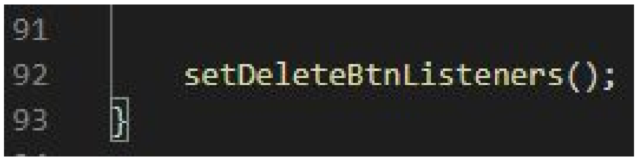
Figure 3. Lines 1 - 10 (left) and Lines 105 - 115 (right) of bag.js

The item_array was initially conceived as a global variable that would mirror the serialized item objects stored in local storage. However, I realized that the scoping of the variable would not allow for this, when both the updateCart function (lines 1 - 10) and the delete button event listener (line 105 - 115) both needed to refer to the item_array. The reference needed to be within the scope of the function. I resolved the issue after realizing that the item_array must act independently of local storage, not just a mirroring of it, and by having the updateCart function and the event listener recreate item_array separately.

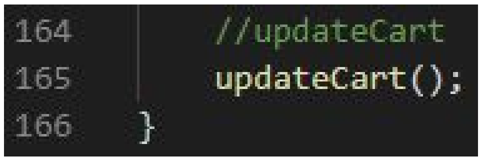
4. Function Calls



```
93          <script>
94              updateCart();
95              document.getElementById("nav_button_bag_count").innerText = Number(localStorage.getItem("itemCount"));
96              setDeleteBtnListeners();
97          </script>
```



```
91
92          setDeleteBtnListeners();
93  }
```

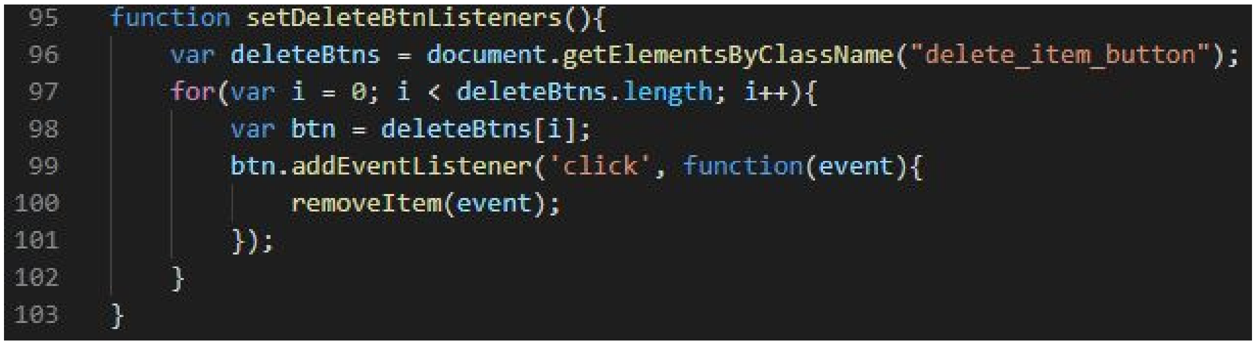


```
164         //updateCart
165         updateCart();
166     }
```

Figure 4. Lines 93 - 97 of mybag.html (top), Lines 91 - 93 (bottom left) and Lines 164 - 166 (bottom right) of bag.js

The function updateCart is first called upon the initialization of the html DOM in mybag.html. At this point the local storage would be filled with serialized item objects. Within the updateCart function the setDeleteBtnListeners function is called, which adds event listeners to the delete buttons. Within the removeItem function, which updates the item_array and local storage, the updateCart function is called again to reflect the changes made to the local storage. This interplay of function calls was important to run the interaction within the My Bag page smoothly.

5. Event Handling



```
 95  function setDeleteBtnListeners(){
 96      var deleteBtns = document.getElementsByClassName("delete_item_button");
 97      for(var i = 0; i < deleteBtns.length; i++){
 98          var btn = deleteBtns[i];
 99          btn.addEventListener('click', function(event){
100              removeItem(event);
101          });
102      }
103  }
```

Figure 5. Lines 95 - 103 of bag.js

Event listeners were added to the delete buttons within the setDeleteBtnListeners function. The triggering of the event would pass the event variable to the removeItem function which would identify which delete button was pressed by extracting the event variable's value attribute.

<u>Other</u>

Unfortunately, I was not able to straighten out the issue regarding the css not applying evenly to github pages and vs code liverserver (this can be seen in homepage) that I mentioned in Assignment 5. Also note, that all the functionality added for Assignment 6 was applied to the **"Original" cinnamon bun only**, the other product pages will not have the added functions. A minor functionality added to the My Bag page is that the "CHECK OUT" button clears all items from the bag page.