Issues/Bugs

To understand my issue, you must first understand the way I was saving information. I would create a unique “Bun Order” each time the user added an item to the cart. Initially, this would only contain the glaze type and quantity for the bun. I would then save these “Bun Order” objects to local storage as a unique id, based on the length of local storage (I ended up with “bunOrder0, bunOrder1, bunOrder2, ...bunOrderN). Then, whenever I would load a page, I would iterate on the local storage, for(var i = 0; i<localStorage.length; i++) and push each bunOrder to a new array I would create. I could then iterate through the array, and populate the html with each order. It worked! Or so I thought. While I was in the shower, I was thinking about how I would implement the delete() function and immediately knew I was doomed. I had no way to tell which index to delete! When a user would click on the X button next to the order they wanted to remove, I called the delete function but could not figure out where to delete that item in localStorage.

To fix this issue, I had to rethink the entire way I was saving my bunOrders. Instead of saving each individual bunOrder to local storage, I discovered I could simply push objects to an array based on what already existed in localStorage, push the new Order to the array, and then set what existed in localStorage to the array. This was great because now I didn’t have to loop through every object in localStorage as well as every object in the array. I could just use the key ‘buns’ and save that to localStorage.

After fixing the issue with localStorage, I went back to how to remove orders in the shopping cart. I fixed the problem by adding a parameter to the delete() function, so now I would pass delete(orderNumber) whenever a user clicked on the delete sign. The orderNumber referred to which index in the Array I needed to delete. This is different from when I was trying to delete in localStorage because every time I deleted an item in localStorage, none of the other elements would “shift” their index. So if I had bunOrder0, bunOrder1, and bunOrder2, I would delete bunOrder1. I would then have bunOrder0, and bunOrder2 saved in localStorage. I can’t reach bunOrder2 based on localStorage.length because I only iterate 2 times, bunOrder0 and bunOrder1, but not bunOrder2. I would also get an error because bunOrder1 did not exist after I deleted it.

I used the Array.splice(orderNumber, 1) function in order to nicely remove the desired element, and shift all other indexes down by 1 space. I would then set the new array to localStorage.

In the future I can avoid all of these problems by reading more carefully how each component works. I didn't realize it was a bad idea to save multiple objects of the same type to localStorage with different keys. I now know I can just save a single key referring to an array of objects of that type.

Programming Concepts

1. Arrays, array.push(), array.length, array.splice() - I used an array to store all of the orders a user made when they pressed the "Add to cart" button. The array was useful because it could hold as many objects as I needed, and also because it could be set and retrieved in localStorage. The most useful feature of the array was the ability for me to remove an item at a specific index, and then shift all the other objects down an index. Here are a few examples I created in my project:
   Creating an array to hold my bun Orders
   var bunList = Array();
   Adding a new order to the array
   bunList.push(order);
   Deleting a certain order from the array
   bunList.splice(bunDelete, 1);
   Iterating based on how many elements were in the array
   for(var j=0; j<bunList.length; j++){}
2. localStorage, localStorage.setItem('key', JSON.stringify(array)), JSON.parse(localStorage.getItem('key') - I used localStorage in order to save bun orders across all HTML on my website. I decided to go with localStorage instead of sessionStorage because I didn't get the difference other than sessionStorage holds less memory. I was confused on localStorage at first, but after playing around with it for a bit and using dev tools on google chrome I learned to really appreciate how simple it is. Some examples in my code were:

```
Saving content in localStorage to bunList array
if(localStorage.length > 0){
        bunList = JSON.parse(localStorage.getItem('buns'));
}
Setting the content in localStorage to orders in bunList array
localStorage.setItem('buns', JSON.stringify(bunList));
```

3. For loops - I used for loops whenever I needed to populate the cart page. They made iterating through each index of my bunOrder array very simple,

and I got it to work functionally. Some examples of for loops in my code were:

```
Iterating through each item in the bunList
for(var j=0; j<bunList.length; j++){
Each image I used was named bun0, bun1, bun2, bun3, so this created an image Array which held that information
for(var i=0; i < 4; i++){
    imageList[i] = new Image(333,272);
    imageList[i].src = "Assets/bun" + i + ".jpg";
    }
```

4. HTML Changing, document.getElementById, document.createElement, element.innerHTML = , element.appendChild(), - Learning these concepts was probably where I spent the most amount of time on my code. I had to figure out all the different functions which would actually change the content of my website. The appendChild() function helped me significantly when trying to understand how to add seperate orders to my cart page. Some examples of code I used in my program were:

```
Changing the amount of items in the cart as a visual for the user
var y = document.getElementById("clicks");
y.innerHTML = "(" + bunList.length + ")";
```

```
Creating a new <p> element for each new order in the cart page.
```

Michael in our class helped me figure this out.

```
var buns = document.getElementById("rightPage")
var z = document.createElement('P');
z.innerHTML ='<img onclick="deletion('+j+')" id="remove" src="Assets/remove.jpeg">'+bunOrder.quantity + " Original Bun - " + glaze + " "
+ '<img id="regular" src="Assets/bun' + bunOrder.glaze +'.jpg">';
buns.appendChild(z);
```

5. Class, constructor - Lastly I learned about the concept of classes and the constructor in javascript. Not all javascript is functions! I found that creating an Order class would make the most sense when I needed to make multiple different Orders. I implemented this by doing the following in my program:

```
class Order{
        constructor(orderNumber, glaze, quantity){
                this.glaze = glaze;
                this.quantity = quantity;
                this.orderNumber = orderNumber;
        }
}
```

Credit:

I would like to give credit to Michael Silvestre for helping me understand how to modify HTML within my javascript.

I would also give credit to https://www.w3schools.com/js/default.asp
W3 schools for giving me information on literally every single function I implemented in my program.