Lia Slaton
Assignment 6B
GitHub Repository: https://github.com/lslaton/assignment_6 (branch is assignment_6b)
GitHub Pages: https://lslaton.github.io/assignment_6/

Reflection

One of the biggest challenges that I faced when coding this project was the remove function in the cart. I knew I would have to remove the roll object from the array that was stored in local storage, but wasn't sure how to get the index of the object that needed to be removed. I had originally tried to make a property of the Roll class that contained the index of that object in the cartArray. It became clear to me, however, that that wasn't the best solution; I would have to move the other objects in the array to replace the hole that removing that object would leave. The index of each object still in the cart would also have to be updated with its new spot in the array. The solution that I ended up with iterates through the children of the larger container that holds the card for each object in the cart, creating a new Roll object for each child. Each of those new items was then pushed into a new array, which then replaced the previous array that had been saved in local storage. I'm not sure if this was the best implementation of a solution (it requires a lot of work to make new items each time), but I was concerned that any other implementation would leave holes in the array. This solution at least ensures that the array is accurate and up-to-date.

Once I had the remove function working, I had to reflect the changes in cart size in the counter in the navigation bar as well as update the cart total on the right hand side of the page. When looking at the way I counted the number of items in the cart in 6A, I hadn't used the array to save the objects at all and had to rely on local storage to keep the count which involved a lot of extra variable and conditionals. Because I was using an array now, I was able to reduce that portion of the code to simply showing the length of the array. It also made it much easier to update the cart count in the remove function, because the array was being updated with each remove call. With the displayed cart total, I had originally written code within the onLoadCart function to help with adding up the price of the items in the cart, converting it to a string, and assigning it to the HTML element. I realized I could reuse that code in a separate function, then call that separate totalCartPrice() and totalPriceString() functions both within the onLoadCart function and the remove function.

When testing this solution, however, I noticed that when the cart was filled and then emptied using the remove function, it would only display a white space and the total cart price would read "Total: $.0". To make sure I could have two decimal places in the price, I used whole numbers to represent the price (for example, $4.50 was represented as 450). The way I had converted the int value of price split the int into the whole number and the last two decimal places (for example, if the int value was 450, that would be split into two string of "4" and "50", then concatenated into "$4.50"). When the cart was emptied and the price was $0, the concatenation was thrown off. To solve this, I made conditionals about what to do when the cart was empty and assigned the value of the string to "$0.00" rather than trying to convert an int. I also made a separate function that would display an empty cart message in the div that holds the item cards, and called it in the empty cart conditionals in onLoadCart and the remove functions.

Programming Concepts

- I learned how to make classes in JavaScript, and how to instantiate those classes to make individual objects. I made a class called Roll that has properties for the description, price, and image (lines 2-8 in bb_main.js). These properties corresponded to the components of the cards in the cart that displayed each added item. For example, if I wanted to add a 3 caramel pecan cinnamon rolls with no glaze for pickup to my cart, that would be saved in a Roll object whose properties would be the description of the rolls (caramel pecan, 3 rolls, no glaze, and pickup), the price of the item ($10), and the image of the caramel pecan cinnamon rolls.
- Local storage was really helpful in saving items to the cart, but particularly in displaying the number of items in the cart in the navigation bar. When I had first started coding the page, the cart count in the navigation bar would reset back to zero whenever navigating between pages. I was able to save the cart count in local storage (lines 45 in bb_main.js), then get that item from local storage each time a new page was loaded so that the navigation bar had the correct count in it (lines 148-160 in bb_main.js).
- In order to save those items in local storage, however, I had to convert the object into a string using JSON.stringify(). I would convert them using JSON.stringify() every time I passed the array that held the items in the cart or the cart count that was displayed in the navigation bar into local storage. When I had to call the values from local storage, I also had to JSON.parse() them back into their original object form (lines 32-48 in bb_main.js).
- When making cards to display each item in the cart, I used for loops to iterate through the array that held the objects. In each loop, I made a new card for the object and added the price to the total price of the cart (lines 180-186 in bb_main.js).
- There are several places within my JavaScript code that use conditionals to accommodate specific scenarios in both the objects and the cart. They came in handy particularly in calculating the total price of the cinnamon rolls (lines 211-250 in bb_main.js). Because it wasn't a proportional increase in price as you bought more cinnamon rolls (i.e., the price of each cinnamon roll decreases the more you buy), I used if/else statements to set the price based on the type of cinnamon roll and the quantity.
- I was also able to use both the grid and flexbox to format my site and the cards that held the items in the cart. For the cards that displayed the items in the cart (lines 36-42 in cartStylesheet.css), I was able to set it as a flexbox with flex-direction set as a row to display the image, description, and price. Then, as each card is being made, the elements are easily aligned and set up to create consistent formatting of the card in the container.