Assignment 6B – Adding Functionality to a Website with JavaScript **Sreya Cherukuri**
Due: Sun Nov 7, 2021,11:59 pm Section C

[Website Link]()	[Github Link]()

Reflection
In the 6A assignment, I had completed the cart adding functionality independent of any implementation of local storage. Once I began to implement local storage, I ran into issues of the incorrect information being saved across pages. It proved to be a struggle to maintain the same construction of my order list along with local storage. For guidance, I attended office hours in order to better understand how to connect local storage along with my current construction of order list. Office hours proved to be the most helpful and I was able to retain the majority of how I had initially constructed my code.

In my original prototype of the quantity selector, I designed them to be visible quantities so that the user knew what the quantity options are before making a decision. In order to construct the circle design of the quantity selector, I built the selectors as multiple <div>. Unfortunately, this construction of the quantities required significantly more logic in order to retain the quantity selected when adding to the cart as well as allowing the user to only pick one option. In order to solve this problem, I explored a multitude of options but was still struggling to implement the styling of the selector in the way that I had originally designed. I considered changing the design of the quantity selector to be different so that it was easier to implement. Ultimately, after a lot of research and googling, I was able to implement and style a selector using a radio button, which also made it easier to record the selection that the user had made.

A major issue I faced with the removal process of cart items was making sure that items were also removed from the order list array, not just from the HTML DOM. Throughout the removal process, the index numbers I was using for removal were constantly changing, which meant that sometimes even if an item is removed from the DOM, it is not removed in the order list array, meaning it would show up in your cart later. In order to fix this issue, I leveraged the inspect tool to track the removal process in order to figure out where the logic error was occurring. Once I identified that the removal process was occurring based on the order that the user removes the items, I was able to understand which part of the process I needed to recode.

Programming Concepts
1. In order to keep track of items that are added to the order and their properties, I leveraged the use of constructors to create cart item objects. By creating these instances, I was able to keep track of each characteristic selected (flavor, glaze, quantity) of the order added to the cart. In my project, I have cartItem objects that keep track of the order information that the user selected.
2. I use loops in my code in order to quickly fill the cart with the users' order information. Using a loop made it simple to go through an action multiple times based on criteria given. In my project, I use a loop to fill the cart based on information in the order list so that whenever the user moves to another page, the cart is always filled with the current order information.

3. I relied heavily on static methods that only work when triggered. Using static methods was essential to constructing the functionality behind clicking the button “Add To Cart.” I use these methods to trigger the process of adding an item to the users’ order.
4. It was important that I picked the correct data type for constructing the order list. Since I created new objects cartItems for every item added, I picked an array to store order information. Since arrays can keep any type of data information in a list, it was essential to implementing the cart feature.
5. In order to minimize the number of methods used in my code for similar activities, I used parameters to a method that took in an object and index that would allow for flexible building. This allowed me to use the same method for order construction regardless of if the cart will filling up on load, or if it was being updated with a new item. In my project, I use createCartItem(cartItem, i).