Clara Cook
05-630
Assignment 6B

1. Adding more functionality to the website:
Github: https://github.com/claracoo/PUI/tree/main/homework_6b
Netlify deployed site: https://hopeful-khorana-b91448.netlify.app/homework_6b/

2. Reflection on Bugs/Challenges
One of the major bugs I encountered was that I could not figure out how to overlay the cart over the other html elements. I could render it over some elements, but not others, meaning that I could see parts of it poking out behind the other elements. At first to fix this, I tried just reordering the elements in the document, but this did more harm than good because it messed up the positionality in the x and y directions. So, next I turned to w3schools. Here, they explained how the z-index is related to the position attribute. I learned where to use the z-index and with what position. I also struggled with the cart page in that I did not know that I should apply JSON in this way. I continued to try to break down the returned string from a localStorage.getItem() call, but was unsuccessful in the crazy ways I tried to parse and restructure the string. I even looked into regex expressions to figure out what would be extracted. Eventually I looked through the lab slides and realized that JSON could be applied to local storage, and it worked nearly instantaneously.

3. 5 Programming Concepts
   a. localStorage

In this assignment, I was able to gain experience with localStorage. I have dealt with sessionStorage before since I am used to dealing with account based websites, but localStorage was an interesting change of pace because it does not reset when the tab closes. For example, I wanted to set which cinnabon's details page the user was on, such that I could gather information from the object variable I set up. When clicking on a particular item in the products page, I set a variable in local storage to store which item I clicked on, like so:

```
localStorage.setItem("name", [a parameter from the onclick function])
```

. Then, on the load of the details page, I could use

```
localStorage.getItem("name")
```

to retrieve the particular item I wanted and could load the associated pictures, prices, and ingredients.

   b. JSON

I also learned about where to use JSON. I have used yaml files, meaning that I am familiar with how to parse strings and objects when presented in a yaml-file-looking structure. That being said, since yaml files on code editors are usually nicely indented and formatted, I was very uncomfortable with this block of unformatted text. That being said, I could see the similarities in how to parse and encode the strings in JSON. I also learned where to apply it. I had never

thought to include anything more complex in local storage than a list, which can be easily dealt with using string manipulations. So the insight that I could put my yaml knowledge into localStorage was quite mindblowing. It would work something like this:

```
let cart = JSON.parse(localStorage.getItem("cart"));
//do stuff to cart
localStorage.setItem('cart', JSON.stringify(cart));
```

c. Regex

In my failed attempt to deal with the complicated structure left in local storage, I tried for hours upon hours to make a regex oriented solution work. I thought that maybe I could parse out the commas, brackets, and keys. For this reason, I am proudly puting regex as a programming topic which I learned, because while unrelated to the assignment, it is because of this assignment that I gained confidence with it. What it looked like before I realized that JSON could solve most of my problems was like this:

```
localStorage.getItem("cart").replace(',(?=[^"]*"[^"]*(?:"[^"]*"[^"]*)*$)',"")
```

I am trying to replace all of the quotation marks and colons present based on a certain structure of the cart object I had created.

d. Tick Mark Notation

I quickly figured out that I had to load each object in the cart dynamically. For this reason, I created a for loop in the javascript in which I would loop through each item in my cart and then go back and identify all of the elements that would need to be updated to reflect the particular item I had placed in the cart. While this did work, it was incredibly tricky to debug since I had to check every individual id for every individual item. The solution was tick mark notation, also known as template strings. This notation allows the programmer to place variables inside the string that will actually load as normal string literals. This looked something like this:

```
<button onClick="subtractNumFromCart('${cinn.name}', '${cinn.glaze}',
'${count}')">-</button>
```

I load a button that will let the user subtract and eventually remove items from the cart, which takes variable parameters, meaning that I could not simply plug in values. For this reason, I can substitute those strings with variables from which item I am loading.

e. Event Handlers

The last main topic I learned about was event handlers. The event handler I has used the most was onload, which I was pretty comfortable with since it took no user interaction other than entering the page, but I started to realize that the small user feedback I wanted would need user interactive events. For this reason, I started looking into event handlers. I was surprised to find how many event handlers there could be. There was even one for ontimeupdate, which I did not use, but this example demonstrated how exactly we are able to handle user input. I was excited to use not just onclick, but onmouseover and onmouseout as well. I was able to deal with the exact hovers that I wanted. It looked something like this:

```
<div class="cartDiv" onmouseover="showHoverCart()" onmouseout="hideHoverCart()">
```