Harkiran Saluja
HW 6B: Adding Functionality with JavaScript

- Link to repository: https://github.com/harkiran-kaur-saluja/homework_6b
- Link to site: https://harkiran-kaur-saluja.github.io/homework_6b/

Bugs

There were three major bugs I encountered as I was completing this assignment. One of them was that in local storage, the elements were not loading to the cart. This was fixed by adding `'cart = JSON.parse(cart || '[]');'` to the `orderDetails.js` file, as JSON cannot parse an empty string, and if the cart does not exist in the local storage, it will return an empty string. The `||` lets the method parse an empty array, which is a JSON object rather than an empty string. Another bug I ran into was that when hitting the delete button in the cart, the page would not update. To address this, I added two lines to the event handler in `checkout.js` and `wishlist.js`, which were `location.reload();` and `loadAllBuns();`. In turn, this was able to refresh the cart page and then reload all of the new buns after updating the cart and wishlist. Lastly, when I was implementing the wishlist, the cart lost elements from the wishlist once the button "add all items to cart" was clicked. Here, I added a `window.onload = function()` in `wishlist.js` to ensure the page would refresh and show all cart elements instead of just changing the location of the page.

Bonus Features

- Implemented a wishlist: from the original bun, the user can add it to the wishlist and also view the wishlist from the navigation bar
- Interesting functionality: the wishlist can be edited and deleted, and there is an option to add the items from the wishlist into the cart, which in turn adds the items to the cart and deletes them from the wishlist

5 Learned Programming Concepts

1. Using local storage to store the elements in the cart through `localStorage.setItem('cart', JSON.stringify(cart));`. Here, I learned that local storage takes in objects as a string and JSON cannot parse objects as a string, so I had to use `cart = JSON.parse(cart || '[]');` to allow the cart to parse the empty array.
2. I was able to use templates to make clones of HTML elements, as `let currentBun = template.content.cloneNode(true);` is an example of creating a clone of the buns found in the template below.

```
<template id="cart-item-template">
                <div class="card1">
                    <div class="card-details row">
```

```
                        <img id="cart-image" alt="original cinnamon roll"
class="col">
                        <div class="col text">
                            <span
class="title">Original   </span>
                            <span class="price"></span>
                            <br>
                            <span class="quantity"></span>
                            <span class="quantity quantity-value"></span>
                            <br>
                            <span class="glaze"></span>
                            <span class="glaze glaze-value"></span>
                            <br>
                            <button style="color: blue; background-color:
none; text-decoration: underline;">Edit</button>

<span>      </span>
                            <button id="delete" style="color: blue;
background-color: none; text-decoration: underline;">Delete</button>
                        </div>

                    </div>
                </div>
            </template>
```

3. Using event listeners in the JS files to handle events when buttons are clicked. See example below.

```
three.addEventListener('click', function(){
    console.log('clicking');
    numClicksThree ++;
    if(numClicksThree % 2 === 1  && sizeSelectedOne === false &&
sizeSelectedSix === false && sizeSelectedTwelve === false) {
        three.style.backgroundColor = '#CFCFCF';
        sizeSelectedThree = true;
    }
    else {
        three.style.backgroundColor = '#EFC65B';
        sizeSelectedThree = false;
    }

 });
```

4. Using variables and mod in the `orderDetails.js` file to make the page elements change color to gray and yellow when clicked. Here, I kept track of a variable for the

number of times each element was clicked and used an if/else check to change color appropriately (see below).

```
  if(numClicksSix % 2 === 1  && sizeSelectedThree === false &&
sizeSelectedOne === false && sizeSelectedTwelve === false) {
        six.style.backgroundColor = '#CFCFCF';
        sizeSelectedSix = true;
    }
    else {
        six.style.backgroundColor = '#EFC65B';
        sizeSelectedSix = false;
    }
```

5. Lastly, I learned how to delete elements from an array in JS, as I don't need to iterate through the whole array in order to find the right element. Rather, I can just use a callback function to find the item with the same name as the bun (see below).

```
const index = wish.findIndex( function(item) {
            if (item.name == bun.name) {
                return true;
            }
        });
```

When the index is found, use splice to delete the element: `wish.splice(index, 1);`.