Bun Bun Pittsburgh

Gabriel Alvarez

1. Reflection.

Coming from a background in architecture, I experienced a steep learning curve throughout the process of turning my mock up into a functional website. Similar to architecture and structural engineering, interaction designers do not necessarily have to be experts at coding but they do need to be able to communicate with developers, have notions of programming and most importantly, know what is feasible and what is not.

- What challenges or bugs did you encounter?

One of the biggest challenges that I encountered was keeping track of everything that needed to be updated in the local storage (quantity and total), both for the home and shopping cart page.

- How did you overcome these challenges?

I overcame this challenge by using many print statements, developing code incrementally and constantly backtracking, but ultimately what helped me the most was talking out pseudocode with more experienced developers.

For the process of keeping count and total up to date, I matched the ids of the local storage to the elements on the list (shopping cart products). One of the most difficult parts was to revise the ids everytime I subtracted an item from the cart so the ids matched the new list length.

2. What programming concepts did you learn as a part of the assignment?

HTML Document Oriented Model (DOM) is a structure constructed as a tree of objects, its configuration allows JavaScript to alter the content of HTML elements using the .innerHTML property.

```
document.getElementById("cart_list").innerHTML = begin;
  document.getElementById("Amount").innerHTML = "$"+price.toFixed(2);
```

Template literals and placeholders proved to be very useful to update the product “bubbles” depending on the flavor, glazing and amount selected by the customer. In hindsight, I could have implemented the same strategy to populate the flavors on my home page, instead of duplicating (and modifying) html branches.

```
var bubble=`<div id=${count} class="productBubble_Shopping_Cart">
   <img id="Remove_Icon_Shopping_Cart" src="Remove_Icon.png" onclick="remove(event)">
   <svg class="bubble_Shopping_Cart">
       <rect id="Rectangle_4" rx="30" ry="30" x="0" y="0" width="665" height="178">
       </rect>
   </svg>
   <div id="bubbleText_Shopping_Cart">
       <span id="Name">${element.name}</span><br/><span id="Price_Glazing_Quantity"
style="font-style:normal;font-weight:normal;">${element.price}<br/>${element.glazing}<
br/>Qty: ${element.quantity}</span>
   </div>
   <!-- make image change -->
   <img id="Bubble_Image_Shopping_Cart" alt="product added to basket"
src="cinnamon.png">
   <div id="bubbleText_Total">
       <span id="SubTotal"
style="text-decoration:underline;">$${(Number(element.price.replace("$",""))*Number(el
ement.quantity)).toFixed(2)}</span>
   </div>
   </div>`
```

Web Storage API allows web applications to store data locally in the user’s browser using a key-value format. By calling out methods such as localStorage.setItem and localStorage.getItem, we can access local storage to add and retrieve data from the browser.

```
localStorage.setItem('glazing', 'Glazing');
   localStorage.setItem('quantity', 'Quantity');
   localStorage.setItem('current', 'Blackberry');
```

JavaScript Object Notation (JSON) is a text-based format used to transmit data in web applications. When adding data to the browser one has to convert the JavaScript object into a string using the method JSON.stringify. And when we want to retrieve data we do the inverse process using the method JSON.parse.

```
if (!JSON.parse(localStorage.getItem("cart"))) {
        let cart = [];
        cart = JSON.stringify(cart);
        localStorage.setItem('cart', cart);
    }
```

Target Event is a property of an event which returns the element that fired the event. Events are defined in HTML and are useful to trigger functions in JavaScript.

HTML:

```
</div>
            <button onclick="addToBasket('Blackberry','$2.75')"
onmouseover="addToBasket_Hover(event)" onmouseout="mouseOff(event)"
id="Blackberry_Add_Basket" type="button">Add to Basket</button>
        </div>
```

JavaScript:

```
function addToBasket_Hover(event) {
    if (event.target.id.split("_")[0] == localStorage.getItem("current") &&
(localStorage.getItem("glazing") != "Glazing" && localStorage.getItem("quantity") !=
"Quantity")) {
        let id = event.target.id.split("_")[0] + "_Add_Basket"
        document.getElementById(id).style.boxShadow = '0 4px 8px 0 rgba(0, 0, 0, 0.2),
0 6px 20px 0 rgba(0, 0, 0, 0.19)';
    }
}
```