Assignment 6 Adding Functionality to a Website with JavaScript
Programming Usable Interfaces
Ember Liu

A link to your live version of the site hosted on Github pages:
https://emberliu1997.github.io/homework_6/

A link to the repository where the code is hosted:
https://github.com/emberliu1997/homework_6

Reflection

Programming Bugs & Challenges

One of the first big issues I had with my website prototype was its inconsistent layout. I am much more used to programming with libraries such as Bootstrap that takes care of the layout of the grid system for me. Without Bootstrap, a lot of the elements on my page were positioned relatively with customized margins and paddings. When the window size changes, the elements would go everywhere since they are not positioned according to the grid. To make my website more consistent and responsive, I implemented a simple grid system myself and went back to change all my previous code to fit the new method. This helped me tremendously when it came to format my pages, thus allowing users to see the content and options better.

Using LocalStorage was another big challenge for me. I kept forgetting to use JSON parser to parse the objects stored, which made it hard for me to retrieve the values from the storage. Moreover, I had trouble templating the items in the shopping cart page. It was easy to simply duplicate or add items to the page, but I had trouble customizing the user selected options and images based on these selections. To overcome this issue, I manually changed the ids of most of the customizable fields based on a common object name in the storage. I was then able to change these images and texts before I insert the template into the page.

Lastly, calculating and updating prices was a huge challenge for me. I wanted to make the shopping cart page as accurate and user friendly as possible, and one of the must haves to make this happen was to update prices based on quantities. This was hard since I had to

simultaneously update individual item price, as well as the subtotal of the entire shopping cart. Initially, every templated item had the same id for the price elements so I couldn't update them individually. Eventually I solved this issue by looping through a class to find the subtotal of every item price and change individual ids before inserting them into the document to allow for individual pricing changes.

Programming concepts

1. Finding HTML Elements:

   I used the following methods very frequently in this assignment to achievement functions like finding an element by element id, by tag name or by class name and adding specific calculation on that

   a. document.getElementById(id)

   b. document.getElementsByTagName(name)

   c. document.getElementsByClassName(name)

```
// Calculate what the cart subtotal is
function calculateSubtotal() {
  var numItems = localStorage.length + 1;
  var items = document.getElementsByClassName("item-price");
  var subtotal = 0;
  for (var i = 0; i < items.length; i++) {
    subtotal += parseInt(items[i].innerHTML.substr(1));
  }
  document.getElementById("subtotal-price").innerHTML =
    "$" + subtotal.toFixed(2);
}
```

2. Changing HTML Elements: element.setAttribute(attribute, value)

In this assignment I also used element.setAttribute(attribute, value) to change the attribute value of an HTML element as shown in the following screenshot.

```
    clon.getElementById("itemQty").selectedIndex = currItem.quantity / 3;
    var itemTotal = currItem.quantity * 3;
    clon.getElementById("itemQtyChange").innerHTML = "$" + itemTotal.toFixed(2);
    var itemPriceSpecific = currItem.name + "Price";
    clon.getElementById("itemQty").setAttribute("id", itemPriceSpecific);
    var itemPriceSpecificChange = itemPriceSpecific + "Change";
    clon
      .getElementById("itemQtyChange")
      .setAttribute("id", itemPriceSpecificChange);
    clon.getElementById("template").setAttribute("id", currItem.name);
    var deleteButtonID = currItem.name + "Delete";
    clon.getElementById("delete-button").setAttribute("id", deleteButtonID);
    var subtotalDivider = document.getElementById("subtotal-divider");
    subtotalDivider.parentNode.insertBefore(clon, subtotalDivider);
  }
  calculateSubtotal();
}
```

3. Adding and deleting element: document.removeChild(element)

   I used document.removeChild(element) to achieve the removing item from chart function

```
function removeItem(id) {
  var rowID = id.substr(0, id.indexOf("Delete"));
  var row = document.getElementById(rowID);
  row.parentNode.removeChild(row);
  localStorage.removeItem(rowID);
  calculateSubtotal();
  onLoad();
}
```

4. HTML DOM querySelectorAll() Method

   The querySelectorAll() method returns all elements in the document that matches a specified CSS selector(s), as a static NodeList object. I used this to highlight selected flavor in product detail page.

5. Toggle Class

   Toggle between adding and removing a class name from an element with JavaScript.

```
// Highlight selected flavor option on product detail page
function selectFlavorOption(id) {
  var element = document.getElementById(id);
  element.classList.toggle("flavor-selected");

  var selectedOptions = document.querySelectorAll(".flavor-selected");
  for (var i = 0; i < selectedOptions.length; i++) {
    var previousSelected = document.getElementById(selectedOptions[i].id);
    if (previousSelected != element) {
      previousSelected.classList.toggle("flavor-selected");
    }
  }
}
```