Luke Breitfeller

Assignment 6B

Reflection:

One challenge I encountered while building my JavaScript was confusion over how complicated functions could be called when a button was clicked. In practicing JavaScript, I found several different ways to attach a function to a button but found some conceptual difficulty in figuring out to instruct JavaScript to assign an unexecuted function as the onclick attribute for a button and also to pass along a value to that function once the button was clicked. I had experienced that same difficulty in assignment 6A but at the time I was working only with buttons that were part of the original HTML code and could thus assign these function values to the button as it was created. In this assignment, I needed to be able to add new buttons to the webpage with each new item added to the cart, and thus had to figure out how to make the same work in JavaScript. I found out that while only the "function () { }" format allows JavaScript to interpret something as a function value to be executed on the click, function calls that take parameters can be included within the curly braces, and thus I can have each item's add/remove button make adjustments to or remove a different product from the cart. Another bug that cropped up happened when I mixed up the class name or id of an object I was seeking to modify. In one case, I mistook the id of the element I wanted to adjust for its parent node, and thus modifying the innerHTML attribute erased the other elements under that parent node. In another case, I was having trouble modifying the style of a particular element (which should have followed the existing style convention in my stylesheet) but ended up assigning that element a class name of "emptycart" when it should have had an id of "emptycart". I overcame this challenge by examining my code more closely and making sure every detail matched across HTML, CSS, and JavaScript.

Programming Concepts:

In this portion of the assignment, I needed to be able to call up the image and alt text for any of the 6 roll types. Unlike in assignment 6A, I could not extract the image information from the product page, because we can access the cart from any other page on the site. As such, I needed to build a set of class constructors (the "productInfo" class constructor, line 3 of bakeryscripts.js) that could store the necessary information about each product. I know if we were working with a larger inventory, we would probably be connecting the site to a MySQL database we could query directly, but for this assignment this set of six productInfo instances works fine.

For this assignment I had to remind myself of the difference between a function as a value and the value returned by the function, especially as it relates to JavaScript. As mentioned in the "challenges" section, many of the ways I tried to execute a clickable button that would run a function were actually linked to an executed version of the function that ran when the page was first loaded. I had to figure out how to, in JavaScript, pass along an unexecuted function as

a value to the button, even when I also needed to give that unexecuted function distinct parameters per button (particular cases are in the newChangeButton, line 452 of bakeryscript.js and newRemoveButton, 459).

I had not used the DOM structure of JavaScript before this class, and had not written script using any of the createElement or appendChild functions of JavaScript before this particular assignment. I needed to use those many times to display the individual cart items (see showCartItem, line 316 of bakeryscript.js). It was also important while using those to keep a careful eye on my intended final HTML hierarchy, to make sure I was adding each new element to the correct parent.

For my product details page, I wanted my “Add to Cart” button to change the contents of the cart and automatically redirect to the product browse page. I found I could not wrap the button in a link and still have the onclick function execute, so I needed to learn the attributes of “window” to change the user’s window location as part of the onclick function (see addToCartDet, line 158 of bakeryscript.js).

Finally, I needed to store information in localStorage and load it with each page reload. To ensure flexibility, these storage calls needed to happen every time the cart was changed in any way. Forgetting that had at times led to my cart seeming not to process certain changes. (The most common calls to localStorage were in loadPage--line 41 of bakeryscript.js and addToCart, line 128).