Bun Bun Bake Shop!

Assignment 6B (Full cart functionality)
Sree Mahit Munakala
MHCI 2022

Website: https://pui-6b-mahit.netlify.app
Repo: https://github.com/SreeMahit/PUI-Assignment-6B

In assignment 6B, I began tackling some of the more critical element of the shopping cart, including a fully functional add/remove feature in both the popup and the cart page, persistent items, and a recommended products section in the product details page.

Reflections

Assignment 6B was one of the most challenging assignments in trying to build a shopping cart. Along with the shopping cart, I also implemented specific product detail pages for each product. Some of the peskiest bugs I encountered while working on this assignment:

1. A product details page that is dynamically updated through JS depending on the selected product

   This is an issue I was struggling to solve for the better part of 3 weeks, from the days of assignment 6A. There were two parts to this issue that I needed to tackle

   - Storing an identifier for which product has been selected from the product details page.
   - Updating the product-details.html file with appropriate product details

   There were 3 ways I tried to store an identifier for which product has been selected: **passing a parameter through the event listener, storing it in local storage, using URL parameters** (as suggested by Jesse). The reason why the first method didn't work was because as a new HTML file was rendered, the JS file is rerun and the parameters were lost. Instead, I used local storage to persist the identifier and retrieve it when rendering the product details page. Jesse's suggestion of using URL parameters also worked perfectly and I could retrieve identifier for the chosen product.

   The second issue was more challenging. Everytime I tried to render the product details page with updated product information, the JS file would try to access the product details page's DOM even before it loaded since the event listener calls the update function much earlier. There were two methods I used to try and solve this problem. The first was to **separate the function into JS files** to see if I can force the required JS file to run only when the DOM has loaded. This didn't work because the function still needed to be present in both filess and defeated the purpose of separating the files. The second was to have the **function run on a window.onload statement** to force it to run when the product details page has loaded. This didn't work because the function ran irrespective of what page was loaded and threw more errors.

   The workaround I employed was to include the script within the HTML file directly. This ensured the functions ran at the right time and the right place. I also needed to leave empty placeholders for information which would be updated once the DOM was fully loaded. This solved my problem but aadmittedly isn't the cleanest implementation.

2. Updating the shopping cart and the pop-up simultaneously

   Once I had built the shopping cart and the cart pop-up, I needed to ensure there was proper sync between the two versions and that updating one would autmotically update the other.

   I struggled with linking the functions without causing any unexpected changes. Sometimes, the empty cart icon wouldn't show up, sometimes removing an item wouldn't re-render the cart, updating the pop-up woudn't update the shopping cart and vice versa.

   The solution to this problem was to just **reviewing every line** and ensuring the render functions are called at the right place. It took a while to get it right but it finally worked!

3. Recommended products dynamically loading

   For the recommended products section in the product details page, I needed to display the rest of the 5 products and not the selected product. This needed to be updated with the right and left arrow keys.

   The issues I ran into while implementing this functionality were centered around the proper use of arrays. When rendering the recommended products, it showed the selected product as well.

   To circumvent this issue, I **duplicated the array using the spread operator**, ran an if/else statement to check for the selected product, remove it and then proceed to render items from the newly created array. This was a workaround that was dynamic and changed according to the user's product selection.

Programming concepts

There were several new concepts I learnt and used for the first time in this assignment.

1. The concept of URL oarameters was new and was explained to me by Jesse. Though I didn't end up using it in the final website, I understand the benefit of storing variables in the URL which can be versatile and not take up any memory.
2. Local Storage has proven to be a very useful feature to make use of in JS. Storing values which need to be accessible across multiple places was one of the biggest use cases for me. My entire shopping cart in an array format is stored in local storage.
3. This was the first time I used the spread operator when I needed to copy arrays without manipulating the original. I used the spread operator when creating the recommended products section when I needed to skip the selected product and show all the other products.
4. The window.onload statement statement proved to be very useful when I needed to update an HTML page immediately after it loads. The entire product details page works on the foundation of window.onload.
5. JSON stringify and parse statements were entirely new concepts for me when working with local storage. Even though directly storing values into local storage converted it into strings, it was always a safe bet to stringify it and made it easier to debug.