Reflection

I encountered many problems in all three areas-HTML, CSS, and javascript. I spent a good amount of time figuring out how to make the three different files communicate with each other. For example, when using the HTML template, I had trouble displaying the added items. The new item would overlap with the old ones. My initial reaction was to give each new item an id and individually edit their position, but then I took another look at the styling structure I built for 6A and realize that I was using flexbox ul, which makes everything on the same horizontal level. Although it worked well for a webpage mockup, using ul as opposed to table suggests that I did not plan for the next step in the early stage.

Another issue that I struggled with was removing an item from both the local storage and the display. It was hard to wrap my head around which function needs to be executed before which. My code still has this problem that it would append an additional existing item after deleting one item, which is because I did not remove all the HTML and data in the local storage before adding in the updated list, but I do not yet know how to solve that.

Programming Concepts

MVC: It was helpful to always keep in mind that my code works with the three models-model, view, and control. It guided me to think about what functions are missing. For example, if, in the console, I see that my local storage contains all the objects I added, but they are not displayed on my screen, then I would know to try accessing the HTML elements or to add an event listener to capture the input.

Loop: I used loops to find elements. For example, while retrieving all the remove button with a for loop, I gave each of them an event Listener to monitor activities constantly.

Object: Using object constructor helped me to better organize related information. Using objects made it easier to retrieve a specific attribute.

DOM: Constantly reminding myself that I'm working with Document Object Model helped me to understand the relationship between different elements. For example, to delete an item in the

cart, I could only engage with the delete button, but tracing back to its parent elements/ parent Node allows me to change elements that are related to the delete button.

Lose-coupling: Separating specific functions is great when it comes to debugging.