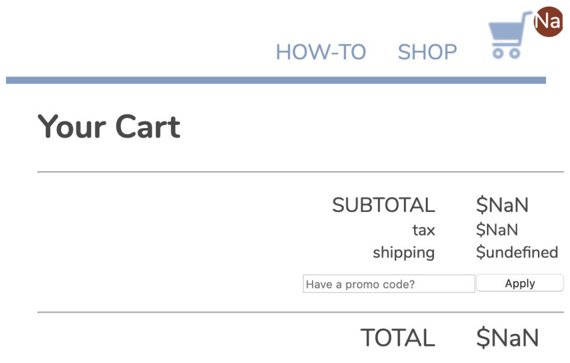
ISSUES ENCOUNTERED

While transitioning my cart page from static HTML/CSS to dynamic functionality with JS, I realized that I needed to create a standalone CSS grid for each item added to cart (initially both items were in one grid on my static page).

After fixing this, I understood that I needed to iterate through the number of items in my cart (array) and create a div for each item. I initially struggled with figuring out where to place them, and learned that I had to create a new div to call, under which I would append the new divs. While building my CSS for the static site I knew that current decisions would come back to haunt me, but did not have the experience or foresight to know how to best construct them in that moment. However, starting with the static cart page was still a valuable exercise, as it helped deepen my understanding of what the final state should look like.

Once my div “infrastructure” was in place, on my product page I used trial and error while constructing my pillow object for adding to the cart. At first I tried to use logic on the cart page to generate an image based on the product color passed through local storage, but I later moved this logic inside the add to cart function on the product page itself, as it was causing issues within my loop. I decided to do the same thing with total price for each object added to cart, passing 25*quantity added within the stringified object. These decisions simplified parsing on the cart page.

At this time I had implemented local storage from the product page to the cart page, and was able to remove items from the cart. However, my cart badge was still showing when the quantity in the cart equalled zero. Even worse, when I cleared my cache or loaded the page in an incognito window, I was met with the following sight on the cart page:



Your Cart

SUBTOTAL	$NaN
tax	$NaN
shipping	$undefined

TOTAL	$NaN

I needed to implement logic to distinguish whether the local storage value for cart quantity was null, zero, or 1+. At first I only accounted for q == 0 and q > 0, and then realized that an empty cart had a null value. After fussing with my conditionals, I realized that a conditional statement with multiple clauses needed a SECOND parentheses wrapper around the entire conditional:

```
if ( (cart_contents == null) || (cart_contents.length == 0) )
```

This enabled me to read the cart quantity or set it to 0 as appropriate.

PROGRAMMING CONCEPTS

Local storage persisting between pages

At first I only set local storage on my product page and parsed it on my cart page. I then realized that moving back and forth between shopping and cart, as users tend to do, would then not show the updated cart quantity on the product page until I called local storage on that page again. This process helped me understand that in future sites, I should make a list of functionality universal to all pages on load vs. specific to some pages, and implement accordingly. This will help me be much more intentional about code deployment in the future.

Global vs local variables

During this project I really felt like my understanding of global vs. local variables solidified. For example, at first on my product page I placed my localStorage.getItem call within my onLoad function. I then realized that it was generating variables I would need to call later, and it had to be moved outside the function. While this seems simple, this project marked a shift in my understanding.

Referencing variables within strings

```
    item.innerHTML += '<img id="img1" src=' +img +'>';
```

Initially I struggled to dynamically generate innerHTML with variables. I came across this functionality on Stack Overflow and tried to implement it; however, there was a syntax error in my code above it so my file was not even running up to this point! I was discouraged, but eventually resolved the earlier issue and was excited to see my variables populate appropriately.

“Reset” and recursion

During office hours I discussed my approach to removing cart objects before implementing code. My first idea was to remove the object from the cart array and then remove the div. The TA pointed out an even easier approach that hadn’t occurred to me: after removing the object from the array, reset the parent div to empty and simply re-execute the loop I had constructed to iterate through the array and generate divs with the cart objects. This kind of reset - recursion is a way of thinking still new to me as a beginner programmer, and I am excited by the possibilities it affords.

Dynamically creating new elements

Prior to this assignment, the functionality in Javascript to write new elements (and have them preemptively styled with CSS) was unknown to me. I used the combination of document.createElement, item.innerHTML, and appendchild(item) to iterate through my cart objects and create new “blocks” for each object in the cart.