Reflection

For this assignment, I had much difficulty updating items in the cart instead of hardcoding them. Since I started off by designing the website using hardcoding, it was difficult for me to change the code while keeping the design. Although I wasn't able to complete the assignment, I realized that I had to use arrays and assign each product photo, name, price, and quantity into the array in order to update and remove items in the cart appropriately. I hope to mitigate this problem in the future by using the appropriate javascript code and assigning each span to a class in order to make the code work without hardcoding it. Another problem that I encountered was increasing the count whenever the 'add to cart' button was clicked on. To resolve this problem, I decided to include a function that increases the count whenever the button is clicked using 'onclick', and display this number next to 'cart' using '.innerHTML', which successfully resolved the issue.

Programming Concepts

Here are the 5 programming concepts that I learned in Javascript:

1. Hoisting: Hoisting is a JavaScript mechanism where variables and function declarations are moved to the top of their scope before code execution.
    - i.e. Declared variable count first:

```
var count = 0;
  function addtocart() {
        count += 1;
        document.getElementById("count").innerHTML = count;
    };
```

2. Closure: A closure is an inner function that has access to the outer (enclosing) function's variables — scope chain.
    - i.e. Function within a function:

```
function exampleFunc() {
  var name = 'Dian';
  function displayName() {
    alert(name);
  }
  return displayName;
}
```

3. IIFE: IIFE (Immediately Invoked Function Expression) is a JavaScript function that runs as soon as it is defined.
    - i.e. Immediately executed:

```
(function () {
    var name = "Dian";
}) ();
```

4. Currying: Currying is a technique of evaluating the function with multiple arguments, into a sequence of functions with a single argument.
    - I.e. Taking multiple arguments:

```
function multiply(a) {
   return (b) => {
      return (c) => {
         return a * b * c
      }
   }
}
```

5. Callback: Callback is a function that is passed to another function as a parameter and is invoked or executed inside the other function
    - i.e. myDisplayer is a function but is passed as an argument in myCalculator

```
function exampleNum(a) {
  document.getElementById("example").innerHTML = a;
}

function myCalculator(num1, num2, callback) {
  let sum = num1 + num2;
  callback(sum);
}

myCalculator(10, 10, exampleNum);
```