NOTE: Completed 6B bonus “wishlist” functionality, on the products page

Assignment 6A

For the low-fidelity sketch, I retained my previous design elements. I kept the navigation consistent at the top, as it is throughout the site, and presented a clear 'Cart' title with products displayed on the left and summary of cost on the right. This way, the user can view both the products and the summary simultaneously, without clicking back and forth.

For the high-fidelity sketch, I also kept most of my previous design elements. I wasn't sure about including the same clear 'Cart' title (which I ended up implementing in my web prototype) but the general layout is consistent. I also changed the navigation bar, deciding clear titles such as 'About' and 'Account' might be easier to scan than icons.

Assignment 6B

1. Issues/Bugs
    a. I had trouble passing props down between parents and children, specifically props that were functions rather than individual variables. I realized that rather than passing a function, for example with “thisIsAProp,” I was calling the function, by using the incorrect syntax “thisIsAProp().”
    b. I was rusty with some Javascript syntax, and what basic functions such as splice() did. For example, I had a bug where splicing resulted in an array of no elements or one element, when one element should have been eliminated instead. I realized that this is because I was doing array = array.splice(1, 1), but since splice() returns the spliced element I was actually saving the array to be only one (or zero) element(s). I fixed this by removing the “array =,” and just using array.splice().
    c. I had issues setting the state of a React component, specifically within component functions. I realized this was because I needed to bind the “this” reference for those functions, such as “this.changePage = this.changePage.bind(this)” since Javascript does not bind class methods by default, so the callback works later on.
2. Programming Concepts
    a. The broadest concept I learned during this assignment was React as a whole, which I had not had experience with beforehand. I learned how React allows developers to build classes called components, and how those components have a render function. This render function returns HTML, which can be interjected with javascript using curly brackets. In my own project, I created multiple components, including an App.js parent component, with child components such as Header.js, Home.js, Products.js, Cart.js, and Footer.js.

b. Another concept I learned about React is the state, which are Javascript objects that hold information within its component that affect that component's rendering. By changing a component's state, using the setState() method, the component automatically re-renders itself (which is an asynchronous call). In my code, I used state to do things such as store the items currently in the cart, in my App.js component (so that the cart information is retained on all pages on the site).
c. I learned about passing variables and functions in React from parent to child components as props. These properties allow data to travel through the hierarchy of parent and child components. For example, I created an "addToCart()" method in the parent component, App.js, that changed the App.js component state based on a given parameter. I then passed this method as a prop down to a child component, Products.js, for later use.
d. I also practiced good React principles in compartmentalizing code, in that I tried to localize components and create individual .js files to hold those components and reuse that, rather than having long, repeated code. I originally had an Products.js file that contained the HTML for all products, but I then created a Pillow.js child component that held individual data for pillows that I passed props to from the parent Products.js component.
e. A useful concept I picked up using React was its method for interjected JSX into HTML, which was itself interjected into JSX with the render() function. For example, I would begin a render() method within a React component, and return HTML. Within that return statement and the HTML in it, I would need to reference, say, the components state. I would use curly brackets, which would look like this: Color: {this.props.cartItems[i].color}
This code reflects the need for text that labels the information, the "Color: ," as well as the need to fetch the value corresponding to the "color" key of the element currently pointed to by the current iterator key, "i," in the array "cartItems" (which itself is a prop passed to the current component).