Homework 6B Jamie Dorst

Site: https://fluffstuff6b.netlify.app

Repo: https://github.com/jamiedorst/pui/tree/main/homework_6b

Reflection:
This was a challenging assignment for me, but I think I learned a lot from it and am much more confident in my understanding of React now. There were many things I struggled with, almost everything in fact, but here are a selection of issues I resolved:

One thing I struggled with was sorting out how all of my functions and classes were passing information to and from one another. I didn't have a great grasp on how props and state worked, so I was really lost as far as how to get the right information to the right place. This affected a lot of my project, but a specific example of this is that I was unsure how to tell the cart *which* type of product has been added. At first, I was trying to add an item based on its id, but that was only telling me information about pillow type, not color or fill. Eventually I realized that I could pass the function multiple props, which would allow me to specify which characteristics the selected item had, and add that specific item to the cart. This also helped me clean things up overall, such that I didn't have unused states or props anywhere.

Another challenge for me had to do with prices. Each of the pillows' price is constant, but I wasn't sure how to get those numbers to add up to a total in the cart. This was especially hard because my price value was a string with a dollar sign in front, rather than a number. I learned about how to remove a character from a string, and then turn that string into a number. Then, I realized that I had to ensure my prices did not go past two decimal points, so I used another JavaScript function to do that. Then, I was able to pass this all to my Cart page and do some basic math functions to calculate tax and estimated total.

Learned Programming Concepts:

1. Maintain values across pages using props
    a. An example of this is in my Navbar, where the Cart is labeled with the number of items in the cart. In order to make it stay constant across pages, I had to first get the length of the list of the items in the cart. Then, I set the state within my parent function, App.js, so it stayed updated, and then passed that value as props to the Navbar component that was rendered on every page.

```
<Route
  exact
  path="/"
  render={(props) => (
    <Homepage {...props} cartNum={this.state.cartItems.length} />
  )}
/>
```

Example of how it's passed as props to each component, which has the Navbar component

```
<header>
  <Navbar cartNum={this.props.cartNum} />
</header>
```

Example of how props is passed to Navbar component

```
<li className="navItem">
  <h2>
    <Link style={{ textDecoration: "none" }} to="/Cart">
      Cart {this.props.cartNum}
    </Link>
  </h2>
</li>
```

How props is used in the Navbar component

2. Create new object and add to array
    a. An example of this is from my product page, where I had to take the options selected by the user, make a product object based on this, and pass it into the array. When the add to cart function is activated, it gets passed the type of pillow, color, and fill. Then, within the function, it first constructs an object using these parameters. Then it sets the state of the cart array using the spread operator, and then adds on this new object.

```
AddToCart = (id, idC, idF) => {
  const newPur = {
    ...ProductList[id],
    color: this.state.defaultColorID[idC].commonName,
    fill: this.state.defaultFillID[idF].fill,
    photo: Images[this.state.colorCounter][id][0],
    itNum: this.state.itNum,
  };
  this.setState({
    cartItems: [...this.state.cartItems, newPur],
    itNum: this.state.itNum + 1,
    totalPrice:
      this.state.totalPrice + parseInt(newPur.price.replace("$", "")),
  });
};
```

How the new variable is made, and then appended to the cart array

3. Display all items in an array based on selection using map
    a. An example of this is through my product page, where I wanted the images to change based on which color was chosen. My nested images array was organized by color and then product, so based on the selected color and the product type id I was able to choose the right array that I wanted to display. I then used the map function to cycle through that array and display all of the images.

```
<div className="detailImgs">
  {this.props.currentPhotos.map((jp) => (
    <div key={jp.id}>
      <img
        className="detailImg"
        src={jp}
        alt="Bright Fluff Stuff pillow"
      ></img>
    </div>
  ))}
</div>
```

How map cycles through and renders each image in the array

4. Change appearance based on click using classes
    a. An example of this is on my product page, in the color and fill selections. I wanted the selected color/fill to have a larger stroke, so it is easily distinguishable from the other options. To do this, I created two different classes, one for “selected” and one for “unselected.” When a new color/fill was clicked, a function would run to change the status of a boolean within the object. Then, based on that boolean, the class would get changed which would alter the CSS appearance of the button.

```
HandleColorToggle = (id) => {
  const newColors = this.state.defaultColorID.map((button) => {
    if (button.id === id && button.expanded === false) {
      return {
        ...button,
        expanded: true,
      };
    }
    if (button.expanded === true && button.id !== id) {
      return {
        ...button,
        expanded: false,
      };
    } else {
      return button;
    }
  });

  this.setState({
    ...this.state,
    defaultColorID: newColors,
    colorCounter: id,
  });
};
```

How the boolean gets changed in the App function

```
<div className="pillowOptions">
  {this.props.defaultColorID.map((el) => {
    return (
      <div
        key={el.id}
        onClick={() => {
          this.props.HandleColorToggle(el.id);
        }}
        className={[
          "colorOption",
          el.simpleName,
          el.expanded ? "selected" : "unselected",
        ].join(" ")}
      />
    );
  })}
</div>
```

How the class gets changed based on that boolean

5. Show a banner based on click
    a. An example of this is on my product page, because when the add to cart or add to wishlist buttons are clicked, a banner appears at the top indicating to the user that their action was successful. To achieve this, I created a function using the document method get element by id. The banners themselves start with a display value of none. When either of the buttons are clicked, this function runs, and changes the display value to flex, and the corresponding banner will appear on the page.

```
function showBanner() {
  document.getElementById("bannerDiv").style.display = "flex";
}
```

How the banner div display style gets changed based on a function