# PUI Homework 6A/B

**Yvonne(Yiwen) Hou | Section C**

## Challenges Reflection

1. The first challenge is to add image into the dropdown selection so that users can have a clear idea/preview of what pillow color are available before they make a decision. Thus, I tried tutorials where teaches me to insert an image into the dropdown fields: https://jqueryui.com/selectmenu/#custom_render
   Eventually, I landed on an easier solution that utilize the emoji. I was surprised that it worked perfectly well and it was a fun experience going through the emoji library and putting it inside the code: https://unicode.org/emoji/charts/full-emoji-list.html

```
<label for="pillowColor">Pillow Color</label>
<select name="pillow color" id="color" onchange="changeColor(value);">
<option value="color">Select a pillow color</option>
<option value="Rainy day"> Rainy day</p></option>
<option value="Morning haze"> Morning haze</option>
<option value="Cozy denim"> Cozy denim</option>
<option value="After school special"> After school special</option>
</select>
```

2. The second challenge that I encountered is that I realized the onclick is not applicable to my <option> tag in the dropdown! After receiving errors, I searched on stack overflow and saw this post describing a similar situation: https://stackoverflow.com/questions/3487263/how-to-use-onclick-or-onselect-on-option-tag-in-a-jsp-page/16432178
   My solution was to instead use onchange="changeColor(value); so the value will be retained and passed on when the user selected a dropdown.

3. The next challenge is debugging a syntax error. Previously, I miswrote the *for ( const item of cart )* into *for ( const item in cart )* and did not realize that it was a syntax error. I solved this problem by writing another for loop with the same function but in a different format. ( see below ).
   Thus, I was able to identify that the loop function is good and the debug scope is narrowed down to the syntax, and it was then that I found out the error. It should be "of", not "in."

```
27
28      console.log("cart is " + JSON.stringify(cart))
29      for (const item of cart) {
30          console.log("item is " + JSON.stringify(item))
31          showProductInCart(item);
32
33      }
34
35      // for (var i = 0; i < cart.length; i++) {
36      //     var item = cart[i]
37      //     console.log("item is " + JSON.stringify(item))
38      //     showProductInCart(item)
39      // }
40      calculateTotal();
41  }
42
```

## What programming concepts did you learn as a part of the assignment?

1. **Separating Javascript files**
   I learned that we want javascript files to run after the page loaded so we should use “defer” when linking the js file in the header. Also, it is a good practice to have multiple javascript files so that it won’t interfere with other sub-pages that do not involve a certain part of the js code.
   From the previous 6A, I placed all my js code in the script.js file and the console will alert me with errors when loading the customization page where the slider js code does not apply there and the script is looking for carousel images on the customization page. But since those images don’t exist on the customization page (they are only on the detail page) I am getting an error.
   Lesson learned and for this 6B assignment, I separated the shared js code into the common.js file and have page-specific js files.

2. **Template & clone**
   I found the template is very handy when writing code that requires populating cards that are in the same format. Writing the Html only one time, I can clone the template and change the properties on the template according to the user selection, and then display it so it’s changing in real-time. This keeps my code easy to understand and concise, solving the need within 15 lines of code!

3. **Local Storage**
   I wrote the code that can get the saved cart from the local storage and parse the result inside the common.js file so that it is accessible globally. I also used *const cart = storedValue ? storedValue : [ ]* to check if the storedValue exists before assigning it to the cart (check null) It was new learning for me to getItem and setItem from local storage and it is amazing to see that my browser remembered my previous choice even though I closed/ refreshed the tab.

4. **cart.findIndex and cart.splice**
   I learned how to first identify the position of the target that I want to modify and then delete it with the built-in function .splice ( index, num ). For instance, in my code, I find the position index of the item in the cart by comparing all its properties type, color, fill, and quantity. If all of them are identical, then I returned a boolean of true and return that index. As long as the index is a non-negative number ( aka cart is not empty ), I can then delete that target by using the .splice (index, numOfHowManyYouWantToDelete ).

5. **window.location.reload()**
   After I implemented all remove functions, I tried to delete an existing product in the cart. When the button is clicked, I found out that the console correctly displayed the console.log info of which card is identified ( meaning that my search part is correct ), but the product is still there. And when I refresh the page, the product is correctly deleted. ( meaning that my remove part is also correct ). So I realized that I should write something to let the page automatically re-render after the changes happen. Therefore, I employed the window.location.reload( ) and it worked well!