Meo Tianyi Zhang
PUI Assignment 6: JavaScript Functionalities
Nov.8 2020

Live site URL:
https://meozhang.github.io/homework_6/bunbun-home.html

Github page:
https://github.com/meozhang/meozhang.github.io/tree/master/homework_6

New Functions added In the cart page:

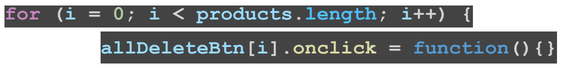
1. Delete item by clicking the “x” button;
2. Change number of items by clicking “-” or “+” buttons;
3. Change the number of an item to “0” would delete the item from the list;
4. Change glazing type of an item;
5. Items that have the same product name and glazing type would merge into one in the list, to maintain clarity and avoid duplicates.
6. Subtotal and the total number of items in the car will update based on the changes;
7. The list of items in the cart will update local storage;

Reflection and Concepts Learned:

1. Reflection (3 pts total)
- You should clearly demonstrate what issues / bugs you encountered, what you learnt from them and how did you resolve them. A good reflection will demonstrate a clear understanding of the issue, and how it may be mitigated in the future.
- Writing should use appropriate style and clearly to convey the writer’s concepts (this includes grammar).
- Writing should demonstrate reflection on actual events and analyze these events to draw appropriate conclusions.

Issue1:

When I created a “for” loop and made a function within it, the index “i” would not get passed through to the function.



```
for (i = 0; i < products.length; i++) {
        allDeleteBtn[i].onclick = function(){}
```

The solution I found was to make a variable “var currentIndex = i;” and later pass the “currentIndex” through the function. I didn’t end up having to do this, because my function changed, but it’s a valuable lesson to learn, as it taught me about what types of variables can be called at a certain point. It’s important to set “global variables” that can be accessed anywhere and the variables created within any curly brackets will only exist within it and can’t be called outside.

Issue2:

I had a hard time making the “delete” function work. As I used a for loop to loop through every item in the cart and tried to delete using the index “i”, the for loop was messed up after one item is deleted.
The solution I found was that, instead of passing the index, I constructed a “key” and used this key to found the specific item in my “cartList”, which is an Array of Objects(The Object also has a “key” property, the combination of product name and product glazing type). This was an eye-opening experience and made me more open to finding an element through other methods than purely relying on the index. Index could be challenging to work with in many circumstances, especially when it comes to deleting an item from an Array.

```
allDeleteBtn[i].onclick = function(){
    const name = this.parentElement.getElementsByClassName("product-name")[0].innerHTML;
    const glazing = this.parentElement.getElementsByClassName("product-glazing")[0].innerHTML;
    const key = name + "/" + glazing;

    const indexToDelete = cartList.findIndex(item => item.key === key);
    cartList.splice(indexToDelete, 1);
```

Issue3:

When creating an “onclick” function for a button, for example “add to cart” button, earlier I was using “getElementsByClassName” to put all the “add to cart” buttons on that page in one array, and loop through it, and add an “onclick” function to each one. I found it to be confusing and overly complicated.
Later I tried a different method, and added the “onclick” function within HTML, and pass “this”, referring to the element that has the “onclick” attribute. And in JavaScript, pass it as a parameter of the function and assign that value to a variable. This works and looks cleaner to me, although I’m not sure which way is the more professional or conventional way to approach the problem. I think it’s worth exploring options and methods.

```
<div>
  <span class="choice">quantity</span>
  <span class="sm-button minus" onclick="quantMinus(this)">&minus;</span>
  <p class="product-quant">3</p>
  <span class="sm-button plus" onclick="quantPlus(this)">&plus;</span>
</div>
```

```
function quantMinus(event){
    var minus = event;
    currentNum = parseInt(minus.parentElement.getElementsByClassName("product-quant")[0].innerHTML);
```

Issue4:

It feels so insignificant and almost stupid to address this here, but I did encounter this multiple times -- Spelling issues!! I once spelled “getELementById” and in the console it kept showing me that it’s not a function. It took me a great while to realize that I accidentally capitalized “L” in “Element”. There was also a time that I set the Class = “product-quant” in HTML but trying to get the element by using “product-quantity” in JavaScript, the inconsistency of naming gave me a hard time problem shooting. Typo happens, but in the future, I will set a better naming standard and be more careful checking the spelling issues as my first step of problem shooting.

2. Programming Concepts (5 points)
    - Demonstrate 5 programming concepts that you learned in Javascript and used in this assignment with an example.
    - Writing should use appropriate style and clearly to convey the writer's concepts (this includes grammar).

Concept 1: The use of "for" loop

"For" loop was used many times in my javascript, it's especially helpful when I need to do the same function for multiple times. For example I created a "for" loop to display the cart item information on the checkout page. It loops through an array of objects, and within the loop a function is called to create a "div" element in HTML that contains information of that specific object.

```
const cartItem = document.getElementById("cart-item");
function updateCart(){
  //clean the HTML cart list
  for(i=0; i<cartList.length; i++){
    product = cartList[i];
    // take product values
    const productName = product.name;
    const productGlaze = product.glazing;
    const productQuant = product.quantity;
    const productPrice = product.price;
    const productImage = product.image;

    //assign values to update cart-item
    cartItem.getElementsByClassName("product-name")[0].innerText = productName;
    cartItem.getElementsByClassName("product-glazing")[0].innerText = productGlaze;
    cartItem.getElementsByClassName("product-quant")[0].innerText = productQuant;
    cartItem.getElementsByClassName("product-price")[0].innerText = productPrice;
    cartItem.getElementsByClassName("product-picture")[0].src = productImage

    //make a clone of cart-item, make visible
    var newCartItem=cartItem.cloneNode(true);
    newCartItem.style.display = "block";
```

```
      document.getElementById("cartList").appendChild(newCartItem);

   }

}
```

Concept 2: Creating an object

An Object comes in handy when I need to structure a set of information (keys and values) associated with one item. The visual format(“key : value”) of it is clear to read and understand. It’s easy to retrieve value of a key, or the key to certain value. I was able to also create additional properties by combining two set properties. For example, I created “key” by combining the “name” and the “glazing”.

```
// Items adding to the card

class CartItem {

   constructor(name, price, quantity, glazing, image) {

     this.name = name;

     this.price = price;

     this.quantity = quantity;

     this.glazing = glazing;

     this.image = image;

     this.key = name + "/" + glazing;

   }

}
```

Concept 3: “findIndex” function

I learned this simplified way to write the findIndex function, which checks each item against a condition and returns the index of the first item that complies with the given condition. This saved me a lot of trouble finding a certain item in an array, and therefore avoided excessive use of “for” loops.

```
const indexToChange = cartList.findIndex(item => item.key === key);
```

Concept 4: setting and getting Local Storage

I learned the concept of using local storage to transfer information from one page to another. This helps me save the objects added to cart in the home page, and retrieve these objects in the checkout page to dynamically build the cart. One important limitation I found was that the structure of the object seems to be lost during the transfer process. I'm able to call the key of the object to get the value, but I wasn't able to create new keys or properties. The object seem to lose the dynamic quality.

```
function setLocalStr(){

  localStorage.setItem("totalCartNum", JSON.stringify(totalCartNum));

  localStorage.setItem("cartList",JSON.stringify(cartList));

}
```

```
function onLoad(){

  //get local storage

  cartList = JSON.parse(localStorage.getItem ("cartList"));

      totalCartNum = JSON.parse(localStorage.getItem ("totalCartNum"));

}
```

Concept 5: "if" and conditions

I used the conditions to switch between different state of an object. For example, I have a "div" block in HMTL that's supposed to show up when the cart is empty. I call this function every time the cart changes. It will check if the "totalCartNum" is 0, and change the display of that div block to "block's " if it is, or "none" if it's not.

```
function emptyCartBlock(){

  const emptyCart = document.getElementById("emptyCart");

  if(totalCartNum === 0){

    emptyCart.style.display = "block";

  } else {

    emptyCart.style.display = "none";

  }

}
```