Sophia Timko
11/7/2021

Homework #6B: Reflection

Website Link: https://sophiati.github.io/pui_projects/homework_6b
GitHub Code Link: https://github.com/sophiati/pui_projects.git

---

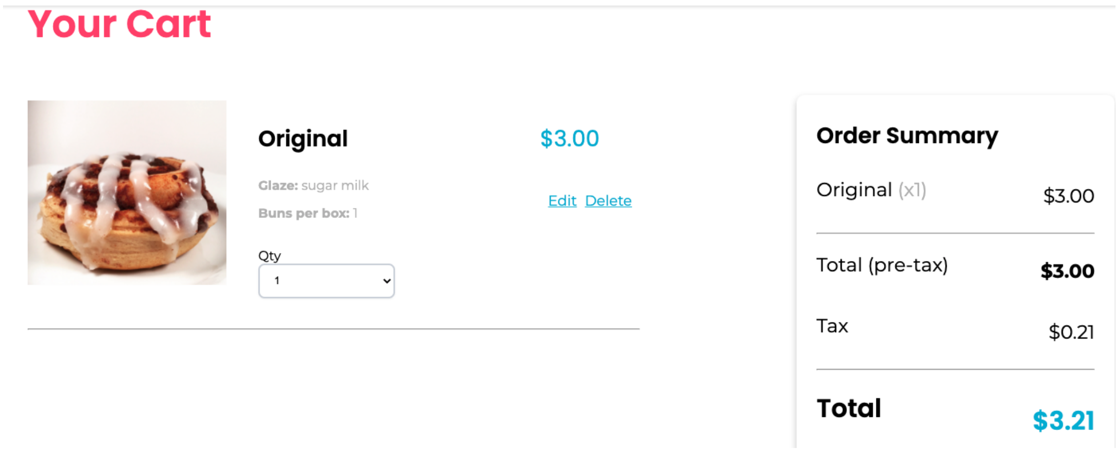
Debugging Reflection

Implementing this shopping cart functionality, I ran into a number of bugs. A challenge I ran into early on was showing the correctly selected items in the cart. After going to office hours and reading through some resources online, I decided to change my original HTML to only have one template of what the cart items would look like in the cart and one template for each receipt item. I then was able to clone this template in Javascript in order to change the content and add it to the array to display correctly.

Another bug I ran into was removing an item from the cart. This was a challenge since I used templates because the delete button clicked could not figure out which delete button it was since the delete buttons were part of the template and were not changing. In order to fix this, I added an event listener directly to the delete button in the template and an Id value to each object in the cart. The Id item then became the index to splice by in the remove function within the delete button event listener. This way, the function was called directly from the cloned item and knew which clone that button corresponded to. In the future, when cloning templates, I could either similarly use an event listener or add an “id” to the HTML element that could be changed by the Javascript for each clone, which would then allow me to access the delete button id.

Five Techniques

1. Cloning Templates - To show items in the cart and to show items in the receipt, I created a template using HTML and then cloned the template using Javascript, which added the clone to the array. The final outcome looked like as follows:



Your Cart

Original $3.00

Glaze: sugar milk
Buns per box: 1

Edit Delete

Qty 1

Order Summary

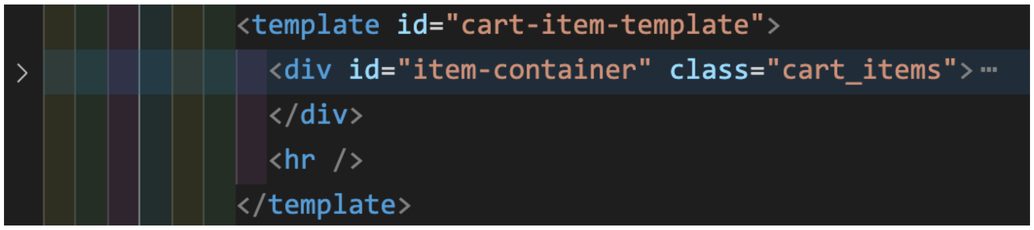
Original (x1) $3.00

Total (pre-tax) $3.00

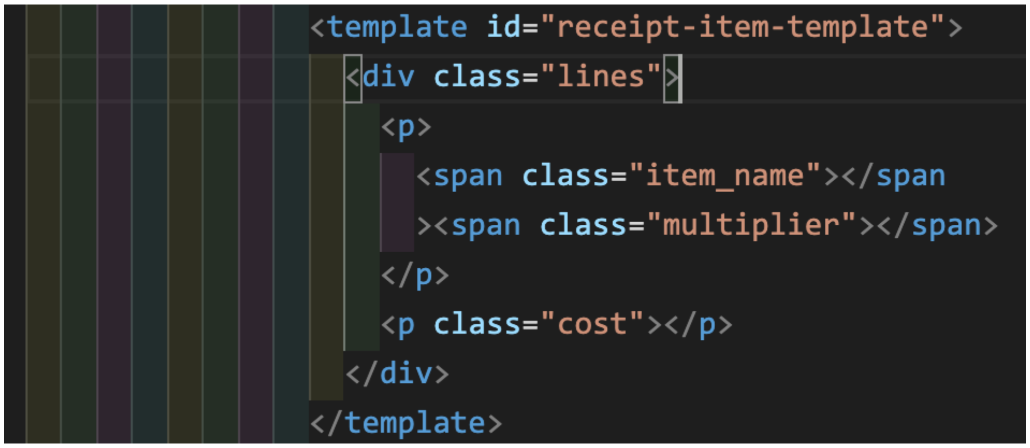
Tax $0.21

Total $3.21

Template HTML:

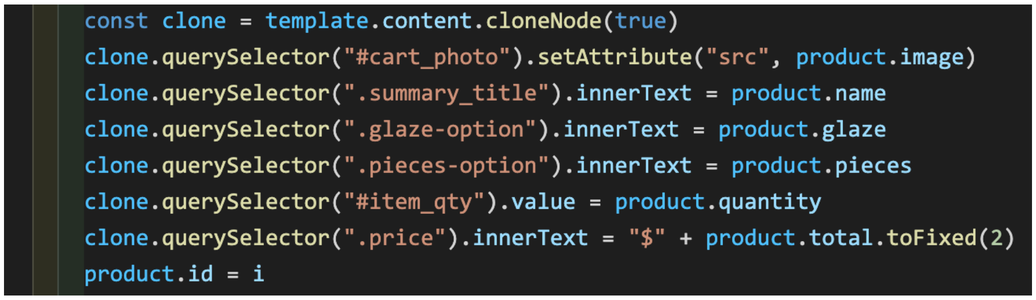


```
<template id="cart-item-template">
  <div id="item-container" class="cart_items">…
  </div>
  <hr />
</template>
```



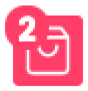
```
<template id="receipt-item-template">
  <div class="lines">
    <p>
      <span class="item_name"></span
      ><span class="multiplier"></span>
    </p>
    <p class="cost"></p>
  </div>
</template>
```

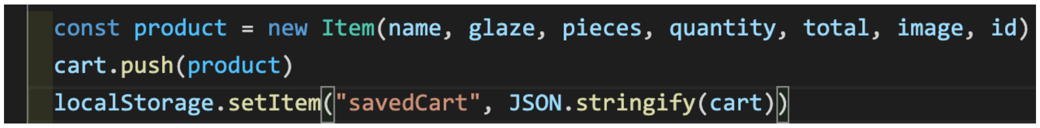
Cloning in JS:



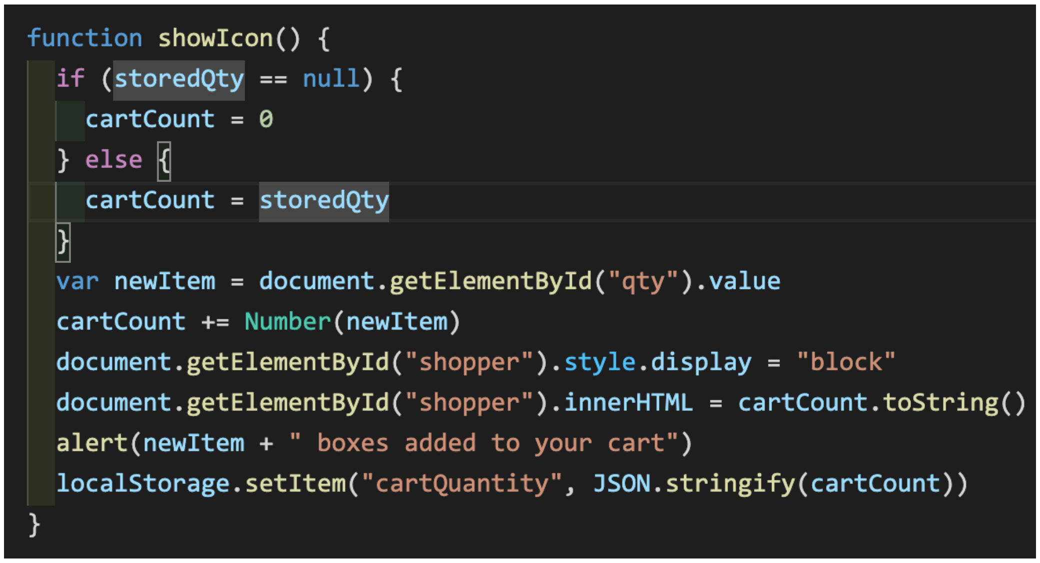
```
const clone = template.content.cloneNode(true)
clone.querySelector("#cart_photo").setAttribute("src", product.image)
clone.querySelector(".summary_title").innerText = product.name
clone.querySelector(".glaze-option").innerText = product.glaze
clone.querySelector(".pieces-option").innerText = product.pieces
clone.querySelector("#item_qty").value = product.quantity
clone.querySelector(".price").innerText = "$" + product.total.toFixed(2)
product.id = i
```

2. Local Storage - After learning this technique in lab, I used local storage twice when implementing the shopping cart. When items were added to the cart, I would use localStorage.setItem to set the cart to store the new item as part of the cart array. I would also use localStorage.setItem to set the value of the shopping cart count, which would display the number of items in the cart on each page. On each page, the cart count local storage was called to display the number of items and on the cart page, the cart local storage was called to access and display the items in the cart. Additionally, on the product page, the savedCart would be checked to see if it was empty or should be the stored cart value.



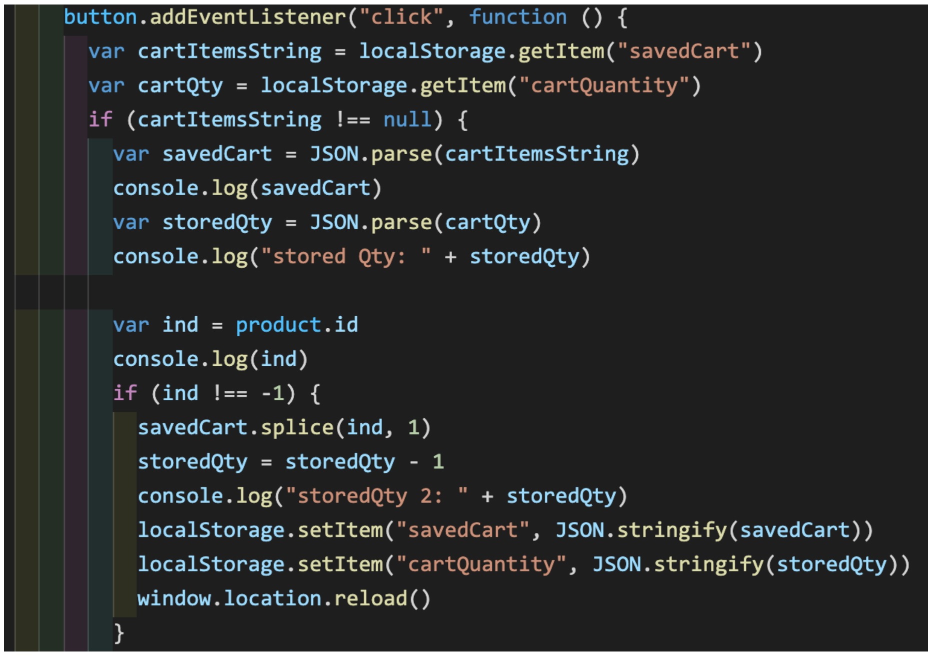


```
const product = new Item(name, glaze, pieces, quantity, total, image, id)
cart.push(product)
localStorage.setItem("savedCart", JSON.stringify(cart))
```



```
function showIcon() {
  if (storedQty == null) {
    cartCount = 0
  } else {
    cartCount = storedQty
  }
  var newItem = document.getElementById("qty").value
  cartCount += Number(newItem)
  document.getElementById("shopper").style.display = "block"
  document.getElementById("shopper").innerHTML = cartCount.toString()
  alert(newItem + " boxes added to your cart")
  localStorage.setItem("cartQuantity", JSON.stringify(cartCount))
}
```

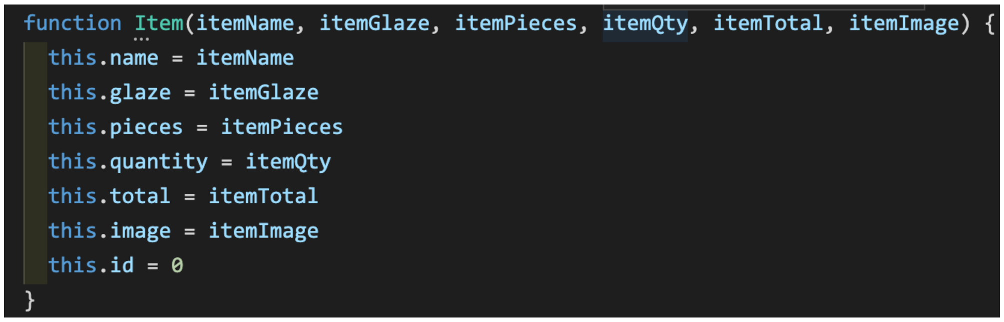
3. Adding Event Listeners - Another challenge when calling the removeItem function was that the delete button wasn't necessarily corresponding to a specific cart item. To fix this, I learned how to add an event listener within the cloned item's delete button and wrote the delete function within that element.



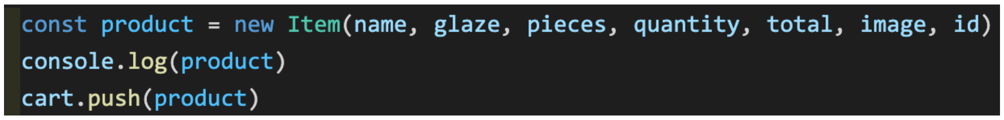
```
button.addEventListener("click", function () {
  var cartItemsString = localStorage.getItem("savedCart")
  var cartQty = localStorage.getItem("cartQuantity")
  if (cartItemsString !== null) {
    var savedCart = JSON.parse(cartItemsString)
    console.log(savedCart)
    var storedQty = JSON.parse(cartQty)
    console.log("stored Qty: " + storedQty)

    var ind = product.id
    console.log(ind)
    if (ind !== -1) {
      savedCart.splice(ind, 1)
      storedQty = storedQty - 1
      console.log("storedQty 2: " + storedQty)
      localStorage.setItem("savedCart", JSON.stringify(savedCart))
      localStorage.setItem("cartQuantity", JSON.stringify(storedQty))
      window.location.reload()
    }
```

4. Creating Objects - One of the first techniques I learned about was creating objects to add them to the cart array. To implement this, I made each cart item an

object with certain attributes: itemImage, itemName, itemGlaze, itemPieces, itemQty, and itemTotal. A new object was created with these attributes each time the addtoCart() function was called on the product page.

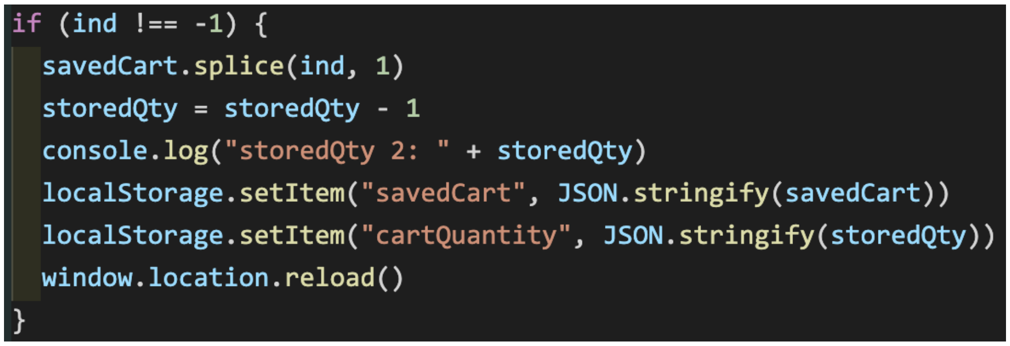


```
function Item(itemName, itemGlaze, itemPieces, itemQty, itemTotal, itemImage) {
    this.name = itemName
    this.glaze = itemGlaze
    this.pieces = itemPieces
    this.quantity = itemQty
    this.total = itemTotal
    this.image = itemImage
    this.id = 0
}
```



```
const product = new Item(name, glaze, pieces, quantity, total, image, id)
console.log(product)
cart.push(product)
```

5. Splicing an Array - The last main technique I used was using the .splice() method to remove an item from an array at a certain index. This was used within the function to remove an item, where the index (which was the Id given to the object) was used to splice the cart array at that point, removing the item at that index from the array.



```
if (ind !== -1) {
    savedCart.splice(ind, 1)
    storedQty = storedQty - 1
    console.log("storedQty 2: " + storedQty)
    localStorage.setItem("savedCart", JSON.stringify(savedCart))
    localStorage.setItem("cartQuantity", JSON.stringify(storedQty))
    window.location.reload()
}
```