Sara Jinfan Bai
PUI Assignment 6B
04/08/2021

Bug Fix #1

One of the main difficult coding part was limiting the number of rolls that users can add roll selections to the show box. For example, when they chose “box of 4” in the previous step, they should not be adding more than 4 to show box. In the “function countRoll(index, flag)”, I mistakenly compared the “restNum” with “count” to ensure the number limit, but it ended up with a bug of not being able to add more rolls after removing some rolls. I fixed it by comparing “restNum” with “flag”, which is the plus and minus action that users take, so that the “restNum” change as users select and deselect rolls while ensuring the number limit.

Bug Fix #2

At first, I put the images of different rolls into the show box but the positions are all off except for image #1 that I used to adjust in css. So I came up with the idea of putting each “<img>” in a “ ” and justify/align their positions inside of the div first, and then adjust the div sizing and positions so that each roll is well placed on the center of each dashed-line square in the show box.

Bug Fix #3

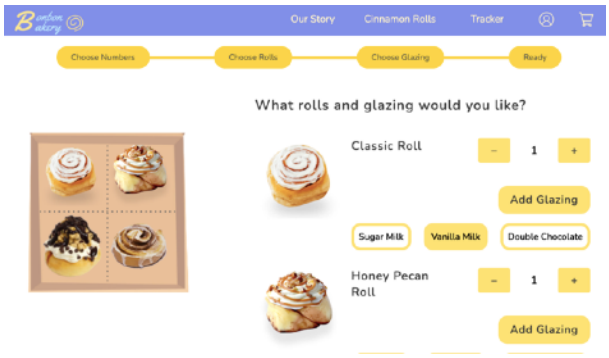
“boxSelect.num” was set to 1 so that when users accidentally enter the next step in building rolls, the show box is set to the Box of 1, instead of no box. But this setting was mistakenly taken over to the the “function openCart()”, so the shopping cart always shows “Box of 1” and its price “$3.25”. I changed it to 0 so that the shopping cart shows as completely empty before users add anything in.

Bug Fix #4

The “openCart()” function is used to manipulate a Dom node. Initially it was called when the js file loaded, but console alert shows an error that the Dom node cannot be found. Then I realized that it’s because the js is called before the HTML file finished loading. So I added the “openCart()” function to “onLoad()” and the cart was correctly displayed.

Programming Concept (in ordering process example)

```
let restNum = boxSelect ? boxSelect.num : null

function countRoll(index, flag){
    let count = Rolls[index].count;
    if (restNum == 0 && flag ==1) return;
    if(count == 0 && flag == -1) return;

    Rolls[index].count = Rolls[index].count + flag;

    let inputs = document.getElementsByClassName("input-number");
    inputs[index].value = Rolls[index].count

    if (flag == 1) {
        restNum--;

        box.innerHTML = box.innerHTML + `<div class="box-${boxSelect.index}-
    } else {
        restNum++;

        let children = box.childNodes // get all rolls in box
        for (let i = 0; i < children.length; i++){
            if(children[i].getAttribute("type") == `${index}`){
                box.removeChild(children[i])
                break;
            }

        }
    }
    let progressDom = document.getElementsByClassName("task-undone")
    if (restNum == 0) {
        progressDom[0].style.backgroundColor = "#FFD433"
    } else {
        progressDom[0].style.backgroundColor = "#FFFFFF"
    }

    localStorage.setItem("Rolls", JSON.stringify(Rolls))
```



1. Use Web Storage API (setItem, getItem, removeItem, key, length).
I stored the the dictionary “Rolls” of each roll in this step so that their key and values can be passed to the openCart() function later.

2. Scope
I used let to declared “restNum” outside of any functions so that it’s used throughout the the entire js file, while declaring “children” and “count” inside of this function to be accessible only in the function.

3. Create and Modify Dom
Access the Dom nodes and make changes to the html for example with “.innerHTML” to add block of elements to the html.

4. Event Handler
Added onClick event to html such as onClick = “addGlazing()” and event handler to add how users can add and remove glazing.

5. Conditional Statement
Used “if else” to check if the restNum =0, the background color of the progress bar on the top will change to solid yellow to indicate process.