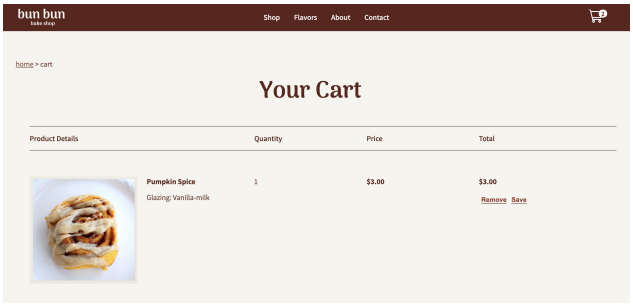
Vikitha Bezawada
11/07/2021

Assignment 6B - Adding Functionality to a Website with JavaScript

Link to Website

Link to Github Repository



Challenges / Bugs:

Since I have no prior experience with JavaScript, this entire assignment was challenging but rewarding at the same time. My initial challenge was to familiarize myself with the different kinds of event listeners in JS. While “onclick” was the most used and straightforward one, I had to understand how to use ‘onchange’ to store the input values from the select drop-down lists. At first, I tried to implement this functionality only using onclick but it didn’t work because the event listener was triggered only when I clicked on the drop-down list, not when I selected an item from the list. After realizing the problem with this logic, I implemented an onchange event listener which successfully gave me the input from the drop-down menu. Following this, I encountered another problem with understanding the order that different event listeners were triggered. From my understanding, the event listeners were fired according to the order they were called in but since my JS code is divided into multiple JS files (to prevent ‘null’ errors), it was hard to see what that order was. I sketched a flowchart of the actions I intended each event listener to perform, which helped me fix this bug. For example, I have a function that updates the number on the cart icon according to the number of items in the cart. This function was only called onload so when the user adds a new item to the cart, the number would not update in real-time but instead be updated only when the page was refreshed. Calling this function again when the ‘Add to Cart’ button was clicked fixed this problem.
Another challenge I faced was to do with adding and retrieving from localStorage. While adding to localStorage was intuitive, retrieving the data stored in localStorage was confusing. Checking the data stored in localStorage and its structure using Chrome Developer Tools (Developer Too;s -> Applications -> LocalStorage) coupled with using multiple print statements was very useful to overcome this.
Lastly, I realized the importance of well-commented and documented code because when having to revisit code I wrote for the previous assignments, I found myself getting confused about how I had implemented it. My HTML and CSS files did not contain enough comments so when I had to revisit them to restructure some of my code, I had to spend some time familiarizing myself with it again. Moreover, I found it relatively easy to implement the main functionality of the website but I had a lot of smaller bugs, such as getting the right input from the drop-down menus, updating the cart with the right item, and displaying an “Added to cart” feedback message at the right time that I had to spend a long time debugging.
Through this assignment, I realized that using a trial-and-error method to come up with code that achieves the programming logic is the relatively easy part. Writing efficient code (i.e. without duplicating code, using functions effectively, etc) is the hard part. Since I have a better understanding of programming now, I aim to write code that is concise and efficient in the future assignments.

Programming Concepts:

1. Event Listeners and Event Handling
   Events in JS are essentially actions that occur in the program to which we can respond to with a piece of code. An event can be handled by a single or multiple event handlers. If the event has multiple event handlers, all of them will be executed when the event is triggered in the order that they were called. There are many different types of event handlers of which the ones I used in my code are: onclick, onload, onchange, and onmouseover. An example of the onclick event handler is seen in my add to cart functionality. When the 'Add to Cart' button is "clicked," the item is added to the cart page, shows a feedback message that says "New Item Added to Cart," and increments the count on the cart icon. The onload event handler works in a similar way. When the Cart page is loaded, the createCart() function is triggered, which contains all the functionality to display the items in the cart.
2. JavaScript Closure
   JavaScript closures are an interesting programming concept. A JS closure is a nested function within another parent function. The inner function has three scope levels: it can access variables declared in its own scope, it can access variables in the parent (outer) function's scope, and also has access to the global variables. This concept was useful in implementing many of my functions, namely the createCart() and productInfo() functions. The createCart() function is implemented with multiple functions declared within it, which allowed me to access the variables declared in the parent function easily.
3. Array Methods
   An Array method that I used to implement the remove item from cart functionality was the .findIndex() method. This method traverses the array and returns the index of the first matching element. In my code, I used this method to find the index of the element that was being removed from the cart to then delete the item from localStorage. Another method that was useful in implementing this functionality was the .splice() method. This method changes the contents of an array in place. Other methods that I encountered while researching array methods include: map(), filter(), and find(). If I wanted to include a condition for the type or number of items added to the cart, the filter() method would have been useful. The filter method takes in a conditional function and returns a boolean value. If the value is true, it returns an array of elements where the function holds true.
4. Traversing the DOM
   The DOM (Document Object Model) is a tree-like structure that is a representation of the HTML contained in the website. The DOM can be made more dynamic using JavaScript by grabbing elements from the existing HTML pages or by adding new elements to the page. In order to build my cart from localStorage, I used this concept. For every item that was added to the cart (and added to localStorage as a consequence), I modified the DOM of the Cart.html page to display the item in the cart. I did this by first creating a new element for each corresponding element (i.e. div, h4, button, etc) and then appending these items to the body of the cart page. While using the <template> tag might have been a more efficient way to implement this functionality, it was more intuitive for me to do it by directly changing the DOM.
5. Identity Operator (==) vs. Equality Operator (===)
   The difference between these two seemingly similar operators was important to understand when implementing conditional statements. The identity operator (==) simply checks the value of the two variables by converting both the variables to the same type before performing the comparison. The Equality operator (===), on the other hand, checks if the value and type of the two variables are the

Programming Concepts:

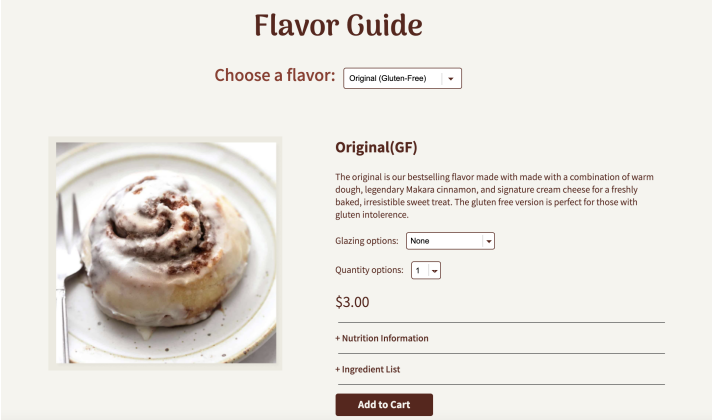
same and returns true only if they're of the same type and value.
For example:
console.log(1 == "1") //returns true
console.log(1 === "1") //returns false
Since the conditionals I used are not necessarily dependent on type, I used the == operator. When I tried to do this by using the === operator, I ran into some issues because I was not sure about the types of the values I was retrieving. Using the typeof function helped me determine the types more accurately.

Add to Cart Functionality:

Add to Cart Functionality:
Flavors.html –> select flavor, glazing, quantity from the options –> Click on Add to Cart button –> View item in Cart.html –> Remove item or add it to wishlist

Bonus:

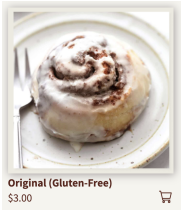
- Every product has its own product page, which was implemented using JavaScript. Each product has its own dynamically created page in which the content differs and the user is able to pick their flavor choice and add the item to cart directly from the page.



- Carousel on the product description page (i.e. flavors.html)

Bonus:

- Interesting JS functionality: Text animation on the home page and shadow on hover for the product cards in the Products page.



- Wishlist: users can add items in the cart to the wishlist by clicking on the ‘Save’ button. Items can be removed from the wishlist by clicking on the ‘Remove’ button. The item is then added to the wishlist section under the cart.

Sources:

Product Images from Unsplash.com and Google Images.