Exercise 9 - Implementing a Shopping Cart with Closures

Introduction

This exercise focuses on using JavaScript to add dynamic behavior to a shopping cart. The HTML and CSS code will be provided, and you will be responsible for adding the JavaScript functionality. Specifically, we will explore the use of closures to add event listeners.

Background

In this Shopping Cart exercise, closures are leveraged to effectively manage the individual products and their associations with specific actions within the cart. Here's how closures bring advantages in this context:

1. **Simplicity**: Closures enable the connection of specific products to their corresponding actions, without the need for complex global variables or structures. This simplifies the process of identifying which product a particular button click relates to.

2. **Memory Efficiency**: By enclosing variables within functions, closures help maintain a clean and efficient memory usage. Variables are not left in the global scope and can be garbage collected when they are no longer needed, promoting optimal performance. Read more about it here.

3. **Reusability**: The use of closures fosters the creation of reusable and customizable behavior for various elements. This not only enhances code modularity but also prevents repetitive code, making future modifications or extensions more manageable.

In sum, closures in this Shopping Cart scenario streamline the code structure, enhance memory utilization, and promote reusable patterns, thus contributing to a robust and efficient implementation.

Instructions

Step 1: Understanding the HTML Structure

Before you start coding, analyze the HTML structure. Note the product divs, buttons, and the cart. Understand how the `data-id` attribute in the product divs can be used to uniquely identify products.

Step 2: Initialize the Cart Object

In the provided JavaScript area, create a `cart` object that will store the product IDs and their respective quantities.

Step 3: Create a Function to Handle Adding Products to the Cart

Now, let's create a function `addProductToCart(productId)`, which will return another function. The inner function will be the event listener, and it needs access to the `productId`.

```
function addProductToCart(productId) {
  return function () {
    // Logic for adding product to cart
  };
}
```

A. Check If the Product Exists in the Cart

Inside the inner function, check if the `productId` exists in the `cart` object.

B. Increment the Quantity or Add a New Entry

If the product already exists, increment the quantity by 1. If not, add a new entry for the product with a quantity of 1.

Step 4: Implement a Function to Update the Cart Display

Create a separate function `updateCart()` that:

A. Selects the Cart Element

Use the `querySelector` method to select the cart items div.

B. Clears Previous Entries

Set the inner HTML of the cart items div to an empty string to clear previous entries.

Step C: Iterate Through the Cart Object

Now you'll create the new cart items.

1. Use a for-of loop to iterate through the `cart` object using `Object.entries(cart)`.

```
for (const [productId, quantity] of Object.entries(cart)){
    // Logic goes here...
 }
```

This will return an array of the object's key-value pairs, where the key is the `productId` and the value is the `quantity`. 2. For each key-value pair (use destructuring to name the key as `productId` and the value as `quantity`), perform the following steps:

Step D: Create a New Div for Each Cart Item

1. Use `document.createElement` to create a new div element, and store it in a variable, such as `cartItemDiv`.
2. Set the `className` property of `cartItemDiv` to `'cart-item'`. This will apply the corresponding styling defined in the CSS.
3. Set the `innerHTML` property of `cartItemDiv` to a template literal displaying the product ID and quantity. This will add the text content to the div. Make sure to follow the provided format, utilizing the `productId` and `quantity` variables.

Step E: Append the New Div to the Cart Items Div

1. Use the `appendChild` method of `cartItemsDiv` to append `cartItemDiv`. This will add the new div as a child element inside the cart items div.

The loop will repeat steps C to E for each entry in the `cart` object.

Step 5: Attach Event Listeners to the Products Using Closures

Create a loop that goes through each product:

A. Select the Product's Button

Inside the loop, use the `querySelector` method to select the button inside each product div.

B. Add an Event Listener

Attach an event listener to the button, using the `addProductToCart` function and passing in the `data-id` of the product. This is where the closure comes into play, as it retains the specific `productId` for each button.

Step 6: Test the Application

Click the "Add to Cart" buttons and ensure that the cart updates correctly.

Conclusion

This exercise challenges you to build a shopping cart feature using closures and event listeners in JavaScript. By following these steps, you'll create a dynamic and interactive experience. Pay attention to how closures enable you to associate specific actions with specific products, making the code more organized and efficient.

Good luck, and feel free to ask questions if you need further clarification!