COMSATS UNIVERSITY ISLAMABAD

ATTOCK CAMPUS

DEPARTMENT OF COMPUTER SCIENCES

SUBMITTED TO:

SIR MUHAMMAD KAMRAN

SUBMITTED BY:

NIMRA AROOJ

(SP22-BSE-046)

COURSE NAME:

MOBILE APPLICATION AND DEVELOPMENT

DATED:

26-SEP-2024

QUESTION:01

Develop a JavaScript-based mobile shopping cart feature that uses ES6 arrow functions, array methods (map, filter, reduce), and object manipulation to manage items in a shopping cart. The application should:

1. Add Items to the Cart: o Create a function to add products to the cart using the push method. o Each product should be an object containing at least the following properties: productId, productName, quantity, and price.

2. Remove and Update Items: o Implement a function to remove items from the cart using the splice method by product ID. o Implement another function to update the quantity of items in the cart using array methods like map or find.

3. Calculate Total Cost: o Using reduce, create a function that calculates the total price of the items in the cart, taking into account the quantity of each item.

4. Display Cart Summary: o Develop a feature that generates a summary of the cart using map and displays each product's name, quantity, and total price for that product. o Additionally, include a function to filter out items with zero quantity from the cart.

5. Bonus (Optional): o Add a feature that allows applying a discount code to the total price. o Use arrow functions extensively and ensure code readability by commenting key parts

INTRODUCTION:

The main objective of this code is to simulate a **shopping cart system** in JavaScript. It allows users to add, remove, and update items in the cart, view the cart summary, calculate the total cost, and apply a discount. Here's a breakdown of its key functions:

1. **addItemToCart**: Adds a product with its details (ID, name, quantity, price) to the cart.
2. **removeItemFromCart**: Removes a product from the cart using its product ID.
3. **updateItemQuantity**: Updates the quantity of a specific product in the cart.
4. **calculateTotalCost**: Calculates the total cost of the products in the cart based on their quantity and price.
5. **displayCartSummary**: Displays the list of products in the cart, their quantities, and total prices.
6. **applyDiscount**: Applies a percentage-based discount to the total cost of the cart.

Overall, this code provides the essential operations for managing a shopping cart in an e-commerce application.

OPERATION IMPLEMENTED:

The operations implemented in the code are:

1. **Add Item to Cart (addItemToCart)**:
   - Adds a product to the cart with specific details such as productId, productName, quantity, and price.
2. **Remove Item from Cart (removeItemFromCart)**:
   - Removes a product from the cart based on its productId. If the product is found, it's removed using the splice method.
3. **Update Item Quantity (updateItemQuantity)**:
   - Updates the quantity of a product in the cart using its productId. It maps through the cart and updates the matching product's quantity.
4. **Calculate Total Cost (calculateTotalCost)**:
   - Calculates the total cost of all items in the cart by multiplying each product's quantity with its price and summing the results.
5. **Display Cart Summary (displayCartSummary)**:
   - Displays the details of each product in the cart, including the product name, quantity, and total price for each product. It also filters out any products with a quantity of zero.
6. **Apply Discount (applyDiscount)**:
   - Applies a discount to the total cart cost based on a discount percentage, and shows the total after applying the discount.

These functions simulate common actions a user would perform in an online shopping cart system.

Code Explaination:

Logic behind each function implemented in code.

1. Initialization:

Purpose:

This line initializes an empty array called cart, which will be used to store the items added to the shopping cart.

2. **Adding Items to the Cart:**

Parameters:
The function takes four parameters: productId, productName, quantity, and price.
Logic:

- A product object is created with the provided details.
- The product object is pushed into the cart array.
- A message is logged to the console to confirm the addition of the product.

3. **Removing Items from the Cart:**

- P**arameters**:
  Takes `productId` as a parameter.
- **Logic**:

  - The function searches for the index of the product in the cart using `findIndex`.
  - If the product is found (index is not -1), it logs a removal message and uses `splice` to remove the product from the cart.
  - If the product is not found, it logs a message indicating that the product does not exist in the cart.

4. **Updating Item Quantity:**

- **Parameters**:
  Takes productId and newQuantity as parameters.
- **Logic**:

  - The map function is used to iterate over the cart array.
  - If the productId matches, it creates a new object with the updated quantity.
  - The cart is then reassigned to this new array with updated quantities.
  - A confirmation message is logged.

5. **Calculating Total Cost:**

Logic:

- The reduce method is used to calculate the total cost of items in the cart.
- It multiplies each item's price by its quantity and accumulates the total.
- The total cost is logged to the console and returned for further use.

6. **Displaying Cart Summary:**

Logic:

- The filter method is used to get only the items with a quantity greater than zero.
- If there are valid items, it maps over them and logs each item's name, quantity, and total price.
- If no items are valid, it logs a message indicating that the cart is empty.

7. **Applying Discount:**

- **Parameters**: Takes discountPercentage as a parameter.
- **Logic**:

  - It first calculates the total cost of the cart.
  - The discount amount is calculated based on the total cost and the provided percentage.

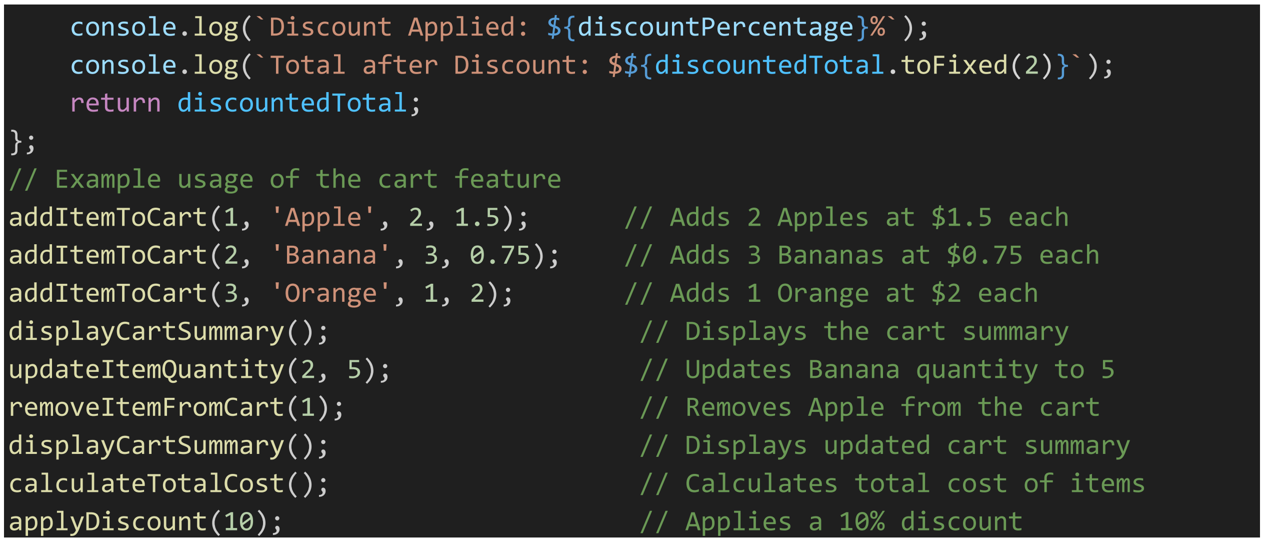
- The discounted total is computed by subtracting the discount amount from the total.
- Messages indicating the discount applied and the new total after the discount are logged.

Code:

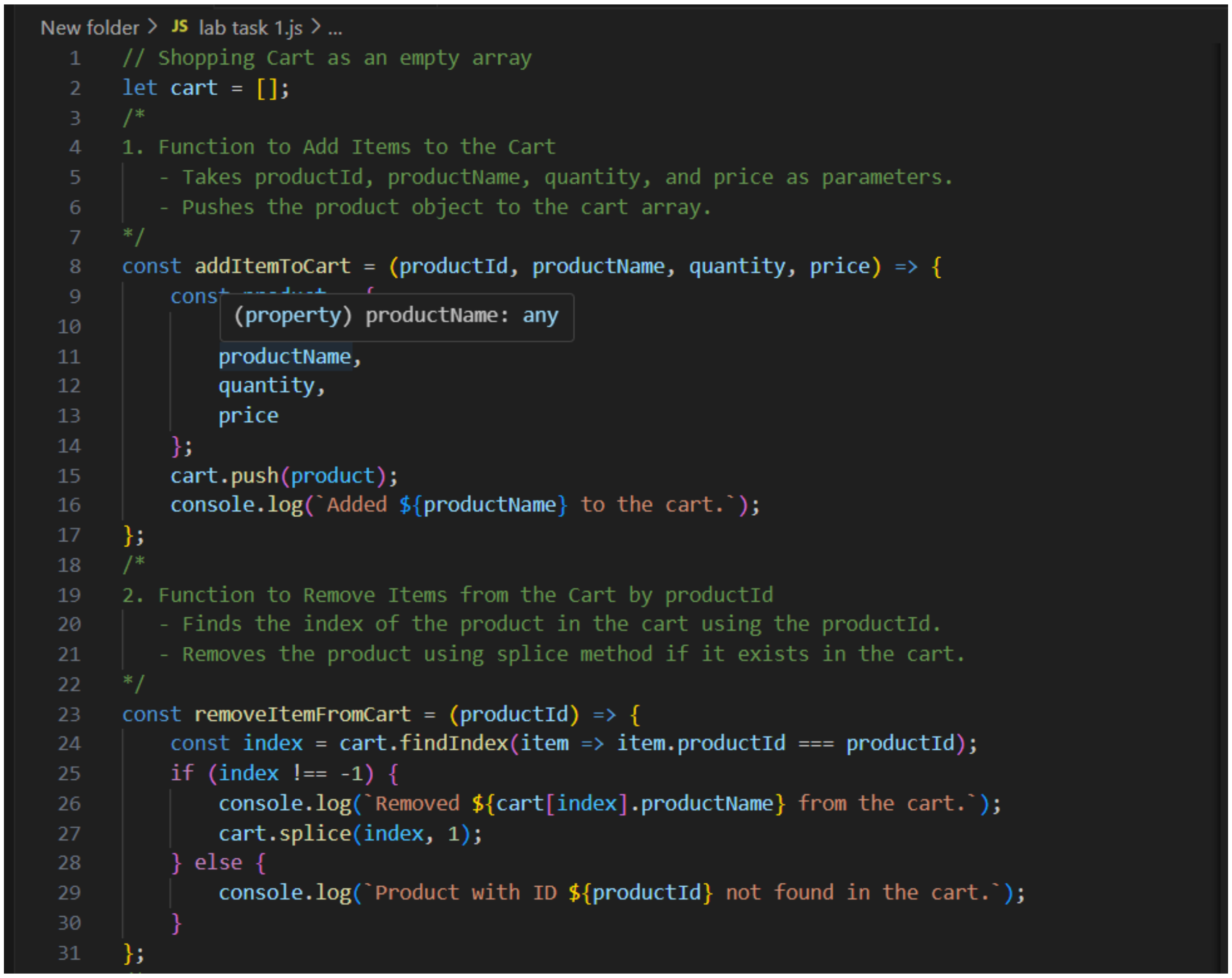
```javascript
let cart = [];
/*
1. Function to Add Items to the Cart
   - Takes productId, productName, quantity, and price as parameters.
   - Pushes the product object to the cart array.
*/
const addItemToCart = (productId, productName, quantity, price) => {
    const product = {
        productId,
        productName,
        quantity,
        price
    };
    cart.push(product);
    console.log(`Added ${productName} to the cart.`);
};
/*
2. Function to Remove Items from the Cart by productId
   - Finds the index of the product in the cart using the productId.
   - Removes the product using splice method if it exists in the cart.
*/
const removeItemFromCart = (productId) => {
    const index = cart.findIndex(item => item.productId === productId);
    if (index !== -1) {
        console.log(`Removed ${cart[index].productName} from the cart.`);
        cart.splice(index, 1);
    } else {
        console.log(`Product with ID ${productId} not found in the cart.`);
    }
};
/*
3. Function to Update Item Quantity
   - Updates the quantity of a product based on its productId.
   - Uses map to create a new array with the updated product quantity.
*/
const updateItemQuantity = (productId, newQuantity) => {
    cart = cart.map(item => item.productId === productId ?
        { ...item, quantity: newQuantity } : item);
    console.log(`Updated quantity for Product ID ${productId} to
${newQuantity}.`);
};
/*
4. Function to Calculate Total Cost of the Cart
   - Uses the reduce method to calculate the total price based on quantity and
price.
*/
const calculateTotalCost = () => {
    const total = cart.reduce((acc, item) => acc + item.price * item.quantity,
0);
    console.log(`Total Cost: $${total.toFixed(2)}`);
    return total;
};
/*
5. Function to Display Cart Summary
   - Uses map to display each product's name, quantity, and total price for that
product.
   - Filters out products with zero quantity.
*/
const displayCartSummary = () => {
    const validCartItems = cart.filter(item => item.quantity > 0);
    if (validCartItems.length > 0) {
        validCartItems.map(item =>
            console.log(`${item.productName} - Quantity: ${item.quantity}, Total
Price: $${(item.price * item.quantity).toFixed(2)}`)
        );
    } else {
        console.log("Your cart is empty.");
    }
};
/*
6. Bonus: Apply Discount Code
   - Applies a discount to the total cost based on the provided discount
percentage.
*/
const applyDiscount = (discountPercentage) => {
    const totalCost = calculateTotalCost();
    const discountAmount = totalCost * (discountPercentage / 100);
    const discountedTotal = totalCost - discountAmount;
    console.log(`Discount Applied: ${discountPercentage}%`);
    console.log(`Total after Discount: $${discountedTotal.toFixed(2)}`);
    return discountedTotal;
};
// Example usage of the cart feature
addItemToCart(1, 'Apple', 2, 1.5);       // Adds 2 Apples at $1.5 each
addItemToCart(2, 'Banana', 3, 0.75);     // Adds 3 Bananas at $0.75 each
addItemToCart(3, 'Orange', 1, 2);       // Adds 1 Orange at $2 each
displayCartSummary();                    // Displays the cart summary
updateItemQuantity(2, 5);                // Updates Banana quantity to 5
removeItemFromCart(1);                   // Removes Apple from the cart
displayCartSummary();                    // Displays updated cart summary
calculateTotalCost();                    // Calculates total cost of items
applyDiscount(10);                       // Applies a 10% discount
```

Screenshots:

```
let cart = [];
/*
1. Function to Add Items to the Cart
   - Takes productId, productName, quantity, and price as parameters.
   - Pushes the product object to the cart array.
*/
const addItemToCart = (productId, productName, quantity, price) => {
    const product = {
        productId,
        productName,
        quantity,
        price
    };
    cart.push(product);
    console.log(`Added ${productName} to the cart.`);
};
/*
2. Function to Remove Items from the Cart by productId
   - Finds the index of the product in the cart using the productId.
   - Removes the product using splice method if it exists in the cart.
*/
const removeItemFromCart = (productId) => {
    const index = cart.findIndex(item => item.productId === productId);
    if (index !== -1) {
        console.log(`Removed ${cart[index].productName} from the cart.`);
        cart.splice(index, 1);
    } else {
        console.log(`Product with ID ${productId} not found in the cart.`);
    }
};
/*
3. Function to Update Item Quantity
   - Updates the quantity of a product based on its productId.
   - Uses map to create a new array with the updated product quantity.
*/
const updateItemQuantity = (productId, newQuantity) => {
    cart = cart.map(item => item.productId === productId ?
        { ...item, quantity: newQuantity } : item);
    console.log(`Updated quantity for Product ID ${productId} to
${newQuantity}.`);
};
/*
4. Function to Calculate Total Cost of the Cart
   - Uses the reduce method to calculate the total price based on quantity and
price.
*/
const calculateTotalCost = () => {
    const total = cart.reduce((acc, item) => acc + item.price * item.quantity,
0);
    console.log(`Total Cost: $${total.toFixed(2)}`);
    return total;
};
/*
5. Function to Display Cart Summary
   - Uses map to display each product's name, quantity, and total price for that
product.
   - Filters out products with zero quantity.
*/
const displayCartSummary = () => {
    const validCartItems = cart.filter(item => item.quantity > 0);
    if (validCartItems.length > 0) {
        validCartItems.map(item =>
            console.log(`${item.productName} - Quantity: ${item.quantity}, Total
Price: $${(item.price * item.quantity).toFixed(2)}`)
        );
    } else {
        console.log("Your cart is empty.");
    }
};
/*
6. Bonus: Apply Discount Code
   - Applies a discount to the total cost based on the provided discount
percentage.
*/
const applyDiscount = (discountPercentage) => {
    const totalCost = calculateTotalCost();
    const discountAmount = totalCost * (discountPercentage / 100);
    const discountedTotal = totalCost - discountAmount;
```

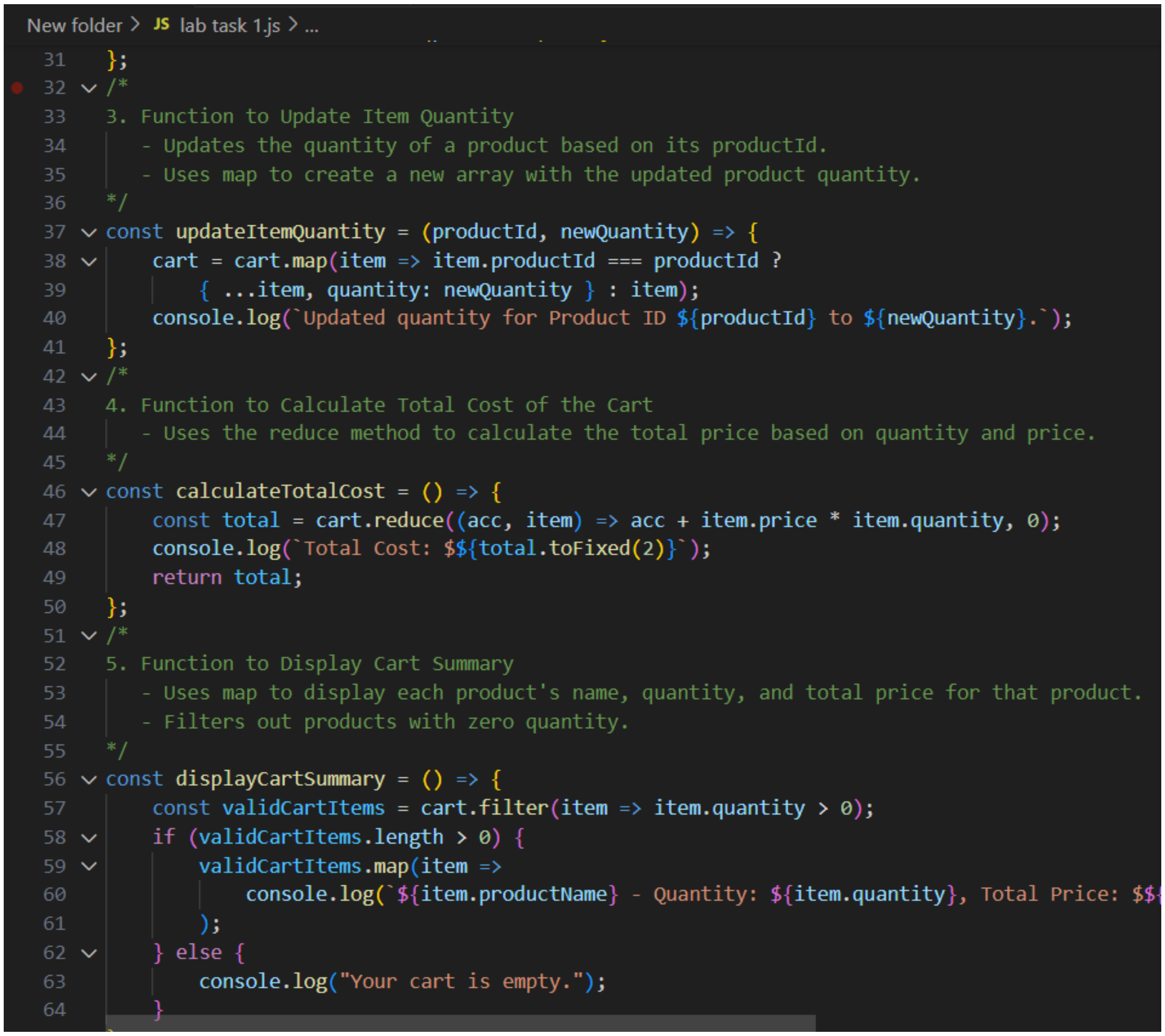


```javascript
    console.log(`Discount Applied: ${discountPercentage}%`);
    console.log(`Total after Discount: $${discountedTotal.toFixed(2)}`);
    return discountedTotal;
};
// Example usage of the cart feature
addItemToCart(1, 'Apple', 2, 1.5);       // Adds 2 Apples at $1.5 each
addItemToCart(2, 'Banana', 3, 0.75);     // Adds 3 Bananas at $0.75 each
addItemToCart(3, 'Orange', 1, 2);        // Adds 1 Orange at $2 each
displayCartSummary();                     // Displays the cart summary
updateItemQuantity(2, 5);                 // Updates Banana quantity to 5
removeItemFromCart(1);                    // Removes Apple from the cart
displayCartSummary();                     // Displays updated cart summary
calculateTotalCost();                     // Calculates total cost of items
applyDiscount(10);                        // Applies a 10% discount
```

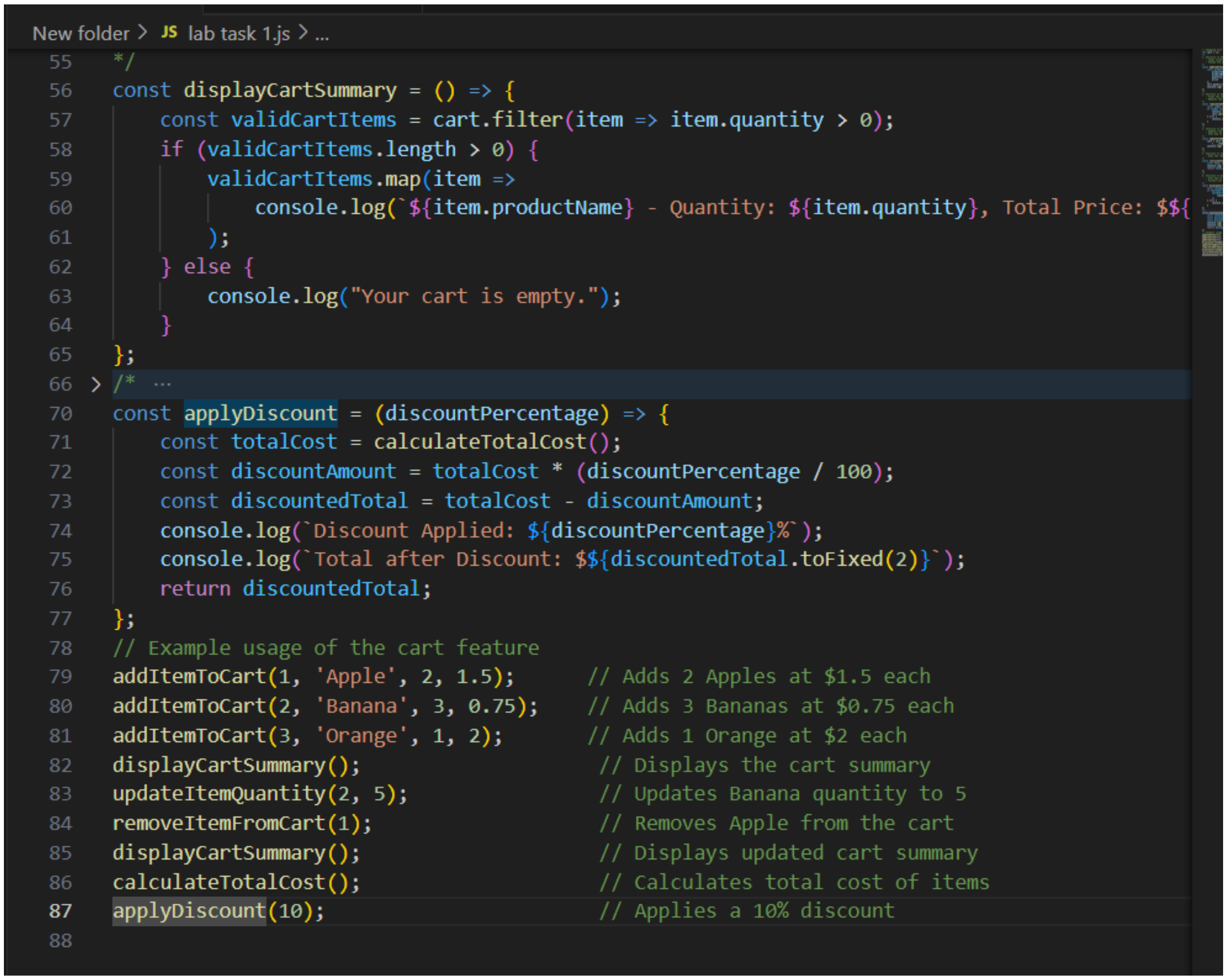

New folder > JS lab task 1.js > ...
(property) productName: any

```javascript
// Shopping Cart as an empty array
let cart = [];
/*
1. Function to Add Items to the Cart
    - Takes productId, productName, quantity, and price as parameters.
    - Pushes the product object to the cart array.
*/
const addItemToCart = (productId, productName, quantity, price) => {
    const product = {

        productName,
        quantity,
        price
    };
    cart.push(product);
    console.log(`Added ${productName} to the cart.`);
};
/*
2. Function to Remove Items from the Cart by productId
    - Finds the index of the product in the cart using the productId.
    - Removes the product using splice method if it exists in the cart.
*/
const removeItemFromCart = (productId) => {
    const index = cart.findIndex(item => item.productId === productId);
    if (index !== -1) {
        console.log(`Removed ${cart[index].productName} from the cart.`);
        cart.splice(index, 1);
    } else {
        console.log(`Product with ID ${productId} not found in the cart.`);
    }
};
```



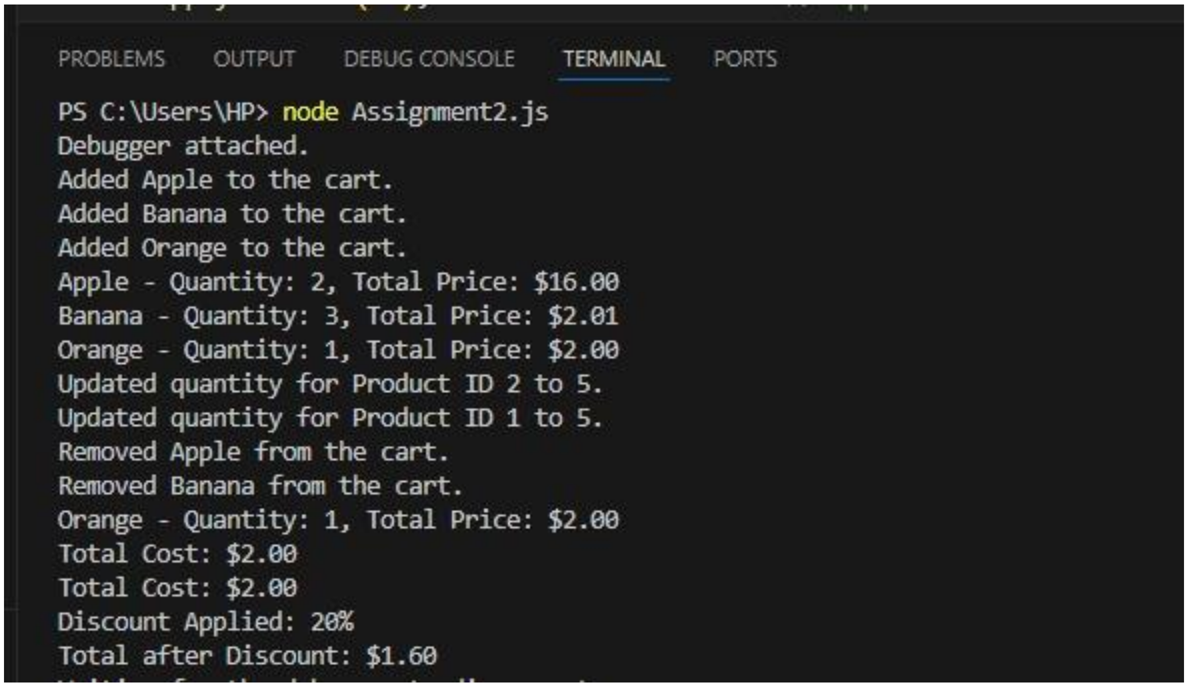
```javascript
};
/*
3. Function to Update Item Quantity
   - Updates the quantity of a product based on its productId.
   - Uses map to create a new array with the updated product quantity.
*/
const updateItemQuantity = (productId, newQuantity) => {
    cart = cart.map(item => item.productId === productId ?
        { ...item, quantity: newQuantity } : item);
    console.log(`Updated quantity for Product ID ${productId} to ${newQuantity}.`);
};
/*
4. Function to Calculate Total Cost of the Cart
   - Uses the reduce method to calculate the total price based on quantity and price.
*/
const calculateTotalCost = () => {
    const total = cart.reduce((acc, item) => acc + item.price * item.quantity, 0);
    console.log(`Total Cost: $${total.toFixed(2)}`);
    return total;
};
/*
5. Function to Display Cart Summary
   - Uses map to display each product's name, quantity, and total price for that product.
   - Filters out products with zero quantity.
*/
const displayCartSummary = () => {
    const validCartItems = cart.filter(item => item.quantity > 0);
    if (validCartItems.length > 0) {
        validCartItems.map(item =>
            console.log(`${item.productName} - Quantity: ${item.quantity}, Total Price: $${
        );
    } else {
        console.log("Your cart is empty.");
    }
```



New folder > JS lab task 1.js > ...

```
*/
const displayCartSummary = () => {
    const validCartItems = cart.filter(item => item.quantity > 0);
    if (validCartItems.length > 0) {
        validCartItems.map(item =>
            console.log(`${item.productName} - Quantity: ${item.quantity}, Total Price: $${
        );
    } else {
        console.log("Your cart is empty.");
    }
};
/* ...
const applyDiscount = (discountPercentage) => {
    const totalCost = calculateTotalCost();
    const discountAmount = totalCost * (discountPercentage / 100);
    const discountedTotal = totalCost - discountAmount;
    console.log(`Discount Applied: ${discountPercentage}%`);
    console.log(`Total after Discount: $${discountedTotal.toFixed(2)}`);
    return discountedTotal;
};
// Example usage of the cart feature
addItemToCart(1, 'Apple', 2, 1.5);      // Adds 2 Apples at $1.5 each
addItemToCart(2, 'Banana', 3, 0.75);    // Adds 3 Bananas at $0.75 each
addItemToCart(3, 'Orange', 1, 2);       // Adds 1 Orange at $2 each
displayCartSummary();                    // Displays the cart summary
updateItemQuantity(2, 5);                // Updates Banana quantity to 5
removeItemFromCart(1);                   // Removes Apple from the cart
displayCartSummary();                    // Displays updated cart summary
calculateTotalCost();                    // Calculates total cost of items
applyDiscount(10);                       // Applies a 10% discount
```

Output:



```
PS C:\Users\HP> node Assignment2.js
Debugger attached.
Added Apple to the cart.
Added Banana to the cart.
Added Orange to the cart.
Apple - Quantity: 2, Total Price: $16.00
Banana - Quantity: 3, Total Price: $2.01
Orange - Quantity: 1, Total Price: $2.00
Updated quantity for Product ID 2 to 5.
Updated quantity for Product ID 1 to 5.
Removed Apple from the cart.
Removed Banana from the cart.
Orange - Quantity: 1, Total Price: $2.00
Total Cost: $2.00
Total Cost: $2.00
Discount Applied: 20%
Total after Discount: $1.60
```

<u>**Conclusion:**</u>

In this assignment, I developed a simple shopping cart system using JavaScript. This project provided valuable insights into basic programming concepts and functional programming patterns. This assignment reinforced my understanding of JavaScript fundamentals and provided practical experience in building an interactive feature. The challenges I faced pushed me to enhance my problem-solving skills and attention to detail. Moving forward, I aim to expand this project by implementing a user interface using HTML and CSS, integrating this JavaScript functionality into a complete web application.