

Project 4

Art Store - Dynamic Shopping Cart Project

Overview:

This project is part of the Art Store website. You will create a **new HTML page (cart.html)** that dynamically displays a shopping cart using data from a provided data.js file. In addition to calculating totals, you will enhance the cart with DOM element manipulation and event handling, allowing interactive user actions like removing items and updating totals.

Instructions:

1. Create cart.html

- This file will serve as the new dynamic shopping cart page.
- In the <body>, include a table structure with the following sections:
 - Table header with columns: Product, Price, Quantity, Total, and Actions.
 - <tbody> section (initially empty) where cart rows will be generated dynamically using JavaScript.
 - Table footer displaying Subtotal, Tax, Shipping, and Grand Total, with placeholders for dynamic updates.
- Include two input fields for user input:
 - Tax Rate (%): an input field where the user can enter a tax rate.
 - Shipping Threshold: an input field where the user can enter the subtotal threshold for free shipping.
- Add buttons for cart actions:
 - "Clear Cart" button to remove all items and reset totals.

2. Link Required Files

- Link to a CSS file for styling the cart (reuse from the previous gallery project or create a new one).
- Link data.js, which contains the cart array (The data.js file will be provided and uploaded along with the project on Moodle.).
- Link cart.js, where you will write the dynamic functionality for the cart.

3. In cart.js

- Add a DOMContentLoaded event handler to ensure your code runs after the page loads.
- Inside this handler, use JSON.parse() (if necessary) to parse data from data.js.
- Create a function calculateTotal(quantity, price) that multiplies quantity by price and returns the result. This function will be used inside the loop to calculate each item's total.
- Create a function outputCartRow(item, total) that dynamically generates a table row (<tr>), using createElement() and appendChild(), containing:
 - Product name (from item.product).
 - Price (formatted with toFixed(2)).
 - Quantity (from item.quantity).
 - Total (formatted with toFixed(2) from total).
 - An action cell with a "Remove" button.
- Loop through the cart array and for each item:
 - Call calculateTotal() to compute the total.
 - Call outputCartRow() to create and insert the row into <tbody>.

4. Calculate Totals Dynamically

- Subtotal: Sum of all item totals.
- Tax: Subtotal \times tax rate (from user input).
- Shipping: \$40 if subtotal is below shipping threshold (from user input); otherwise, \$0.
- Grand Total: Subtotal + tax + shipping.
- Use `document.getElementById().textContent` to dynamically update these values whenever the cart changes.

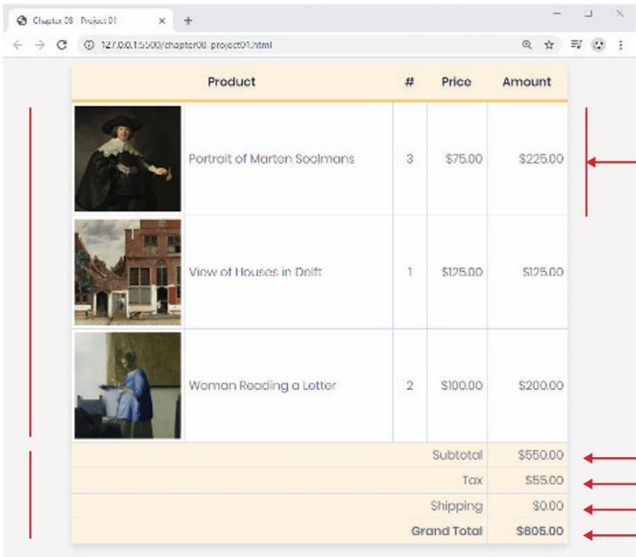
5. Add DOM Element Manipulation and Event Handling

- Replace `document.write()` with `createElement()` and `appendChild()` to dynamically build and insert elements.
- Add an event listener to each "Remove" button in the action column. When clicked:
 - Remove the respective `<tr>` from the table.
 - Update the cart array to remove the corresponding item.
 - Recalculate subtotal, tax, shipping, and grand total.
- Add event listeners to the tax rate and shipping threshold input fields. On input or change events:
 - Recalculate totals dynamically based on the updated inputs.
- Implement the "Clear Cart" button functionality:
 - Remove all rows from the table body.
 - Clear the cart array.
 - Reset totals to zero.




Test Your Work

1. Verify that the cart table is generated dynamically from data.js.
2. Check that totals are calculated and displayed correctly based on cart items.
3. Change the tax rate and shipping threshold inputs and confirm that totals update dynamically.
4. Click the "Remove" button for individual items to ensure the row is removed and totals update accordingly.
5. Click the "Clear Cart" button to remove all items and reset totals.

Here is a **preview image of the expected final layout** of your shopping cart page. Keep in mind that your design might be a little different, and that's completely fine. See the image below.



The screenshot shows a web browser displaying a shopping cart page. The page has a table with columns: Product, #, Price, and Amount. It lists three items: 'Portrait of Marten Soolmans' (3 units at \$75.00 each, totaling \$225.00), 'View of Houses in Delft' (1 unit at \$125.00, totaling \$125.00), and 'Woman Reading a Letter' (2 units at \$100.00 each, totaling \$200.00). Below the items, there are rows for Subtotal (\$550.00), Tax (\$55.00), Shipping (\$0.00), and Grand Total (\$605.00). Annotations with red arrows point to specific parts of the table: 'Replace markup with JavaScript loop using supplied array data' points to the item rows; 'Replace markup with calls to functions' points to the subtotal, tax, shipping, and grand total rows; 'Create function to output single cart row' points to the first item row; and 'Create functions to calculate these values' points to the subtotal, tax, shipping, and grand total rows.

Product	#	Price	Amount
 Portrait of Marten Soolmans	3	\$75.00	\$225.00
 View of Houses in Delft	1	\$125.00	\$125.00
 Woman Reading a Letter	2	\$100.00	\$200.00
Subtotal			\$550.00
Tax			\$55.00
Shipping			\$0.00
Grand Total			\$605.00

Additional Notes:

- Make sure to **place data.js in the same directory as your HTML and JavaScript files** for the cart to load data correctly.
- The **provided preview image** is an example of the final result. Your implementation may vary slightly depending on your design and style choices.
- Always test your project by **changing values and interacting with buttons** to ensure everything works as expected.