Shopping Cart Example

This is a simple shopping cart implementation using vanilla JavaScript. It fetches product data from an API, displays products, and allows users to add/remove items from their cart, view details via a modal, and manage quantities.

Features

- **Fetch Products**: Fetch product data from a remote API (fakestoreapi.com).
- **Cart Management**: Add/remove products from the cart, update quantities, and view total price.
- **Product Quick View**: View detailed information about a product in a modal.
- **Cart Persistence**: Cart data is saved to localStorage for persistence across page reloads.
- **Interactive UI**: Dynamic updates to the cart indicator, total price, and cart items.

How It Works

Fetching Products

Upon loading the page, the fetch function is used to request product data from the API (https://fakestoreapi.com/products). This data is then mapped to a simplified structure to be used in the cart, with each product including the following properties:

- id: Product ID.
- product_name: Title of the product.
- product_description: Description of the product.
- product_price: Price of the product, formatted as a string with two decimal places.
- product_image: Image URL of the product.
- added_to_cart: Boolean to track if the product has been added to the cart.
- quantity: Number of items added to the cart.

Cart Management

- **Add/Remove Items**: Users can add or remove items from the cart by clicking the "Add to Cart" or "Remove" buttons.

- **Quantity Update**: The cart allows users to increase or decrease the quantity of items using the "+" and "-" buttons.
- **Update Cart Indicator**: The number of items in the cart is shown in a cart indicator, which updates dynamically as items are added or removed.
- **Total Price**: The total price of all items in the cart is calculated and displayed at the bottom of the cart.

Quick View Modal

- **Product Details**: A modal is displayed when the "Quick View" button is clicked. This modal shows the product image, name, price, and description.
- **Add/Remove in Modal**: Users can also add or remove items from the cart directly from the modal.

Cart Persistence

- **localStorage**: Cart data (items and quantities) is stored in the browser's localStorage to persist the cart state across page reloads.

Events Handled

- **Click Events**: Several click event listeners are used for:
    - Adding/removing products from the cart.
    - Opening and closing the quick view modal.
    - Handling cart quantity changes.
    - Toggling the shopping cart sidebar visibility.

Functions

- **updateCartIndicator**: Updates the cart indicator with the current number of items in the cart.
- **updateTotalPrice**: Calculates the total price of items in the cart and displays it in the shopping cart.
- **toggleCartButton**: Toggles the "Add to Cart" and "Remove" buttons based on whether the item is in the cart.
- **renderCartItem**: Renders an individual product item in the shopping cart.
- **renderProductCard**: Renders an individual product card in the product list.
- **openQuickViewModal**: Opens a modal with product details.
- **closeQuickViewModal**: Closes the quick view modal.
- **manageCartItems**: Adds or removes items from the cart and updates the UI.
- **initializeCart**: Initializes the cart from localStorage (if any).

- **handleQuantityChange**: Handles changing the quantity of items in the cart (increase or decrease).

Local Storage Usage

The cart items are saved in the browser’s localStorage so that if the user refreshes the page or returns to the site, their cart will persist. The data is stored as a JSON string under the key cartItems.

How to Run

1. Clone this repository or download the project files.
2. Open index.html in a browser to view the shopping cart in action.
3. The cart and products will be fetched from the API, and you can interact with the cart and product list as described above.

Customization

You can modify the following:

- **API Endpoint**: Change the fetch URL to point to a different API for products.
- **UI Design**: Modify style.css to customize the appearance of the product list, cart, and modal.
- **Product Data**: Update the product data mapping if the API response structure changes.