🛍️ Build Your First Online Store

Complete E-Commerce Tutorial - Learn JavaScript by creating a real shopping website!

Table of Contents

🚀 Getting Started

Lesson 1: Setting Up Your Project

Let's create our very first online store! We'll call it "TechVibe" and it will sell cool gadgets.

Step 1: Create your project folders

```
TechVibe-Store/
├── index.html
├── products.html
├── cart.html
├── checkout.html
├── contact.html
├── style.css
├── script.js
└── images/
    └── (we'll add images later)
```

Step 2: Create all these empty files in your folder

- Just create empty files for now - we'll fill them in the next lessons!

> 💡 **What we learned:** Organization is super important! By creating a clear folder structure, we make our project easy to understand and maintain.

🏠 Creating Our Homepage

Lesson 2: Basic HTML Structure

Let's start with the homepage. HTML is like the skeleton of our website - it gives structure to everything.

Create `index.html`:

html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>TechVibe - Electronics Store</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <!-- This is our navigation bar -->
    <nav class="navbar">
        <div class="nav-container">
            <h2 class="logo">TechVibe</h2>
            <ul class="nav-menu">
                <li><a href="index.html">Home</a></li>
                <li><a href="products.html">Products</a></li>
                <li><a href="contact.html">Contact</a></li>
                <li><a href="cart.html" class="cart-link">Cart (0)</a></li>
            </ul>
        </div>
    </nav>

    <!-- This is our hero section - the big banner at the top -->
    <section class="hero">
        <div class="hero-content">
            <h1>Welcome to TechVibe!</h1>
            <p>Find the best electronics at amazing prices</p>
            <a href="products.html" class="btn">Shop Now</a>
        </div>
    </section>

    <!-- This is where we'll show featured products -->
    <section class="featured">
        <div class="container">
            <h2>Featured Products</h2>
            <div class="products-grid" id="featured-products">
                <!-- Products will appear here with JavaScript -->
            </div>
        </div>
    </section>

    <!-- Footer -->
    <footer class="footer">
        <p>&copy; 2024 TechVibe. All rights reserved.</p>
    </footer>
    <script src="script.js"></script>
</body>
</html>
```

> 🎓 **What we learned:**
>
> - **HTML tags** create the structure of our page
> - **Comments** (<!-- -->) help us remember what each section does
> - **Links** connect our pages together
> - **IDs** (like "featured-products") let JavaScript find specific elements
> - The **script tag** at the bottom loads our JavaScript

🎨 Making It Look Good

Lesson 3: Basic Styling with CSS

CSS is like the paint and decorations for our website. Let's make it look professional!

Create `style.css`:

```css
/* Reset - this removes default browser styling */
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

/* Basic body styling */
body {
    font-family: Arial, sans-serif;
    line-height: 1.6;
    color: #333;
}

/* Container for centering content */
.container {
    max-width: 1200px;
    margin: 0 auto;
    padding: 0 20px;
}

/* Navigation Bar */
.navbar {
    background-color: white;
    box-shadow: 0 2px 5px rgba(0,0,0,0.1);
    position: fixed;
    top: 0;
    width: 100%;
    z-index: 100;
}

.nav-container {
    max-width: 1200px;
    margin: 0 auto;
    padding: 0 20px;
    display: flex;
    justify-content: space-between;
    align-items: center;
    height: 70px;
}

.logo {
    color: #3b82f6;
    font-size: 24px;
}
```

```css
.nav-menu {
    display: flex;
    list-style: none;
    gap: 30px;
    align-items: center;
}

.nav-menu a {
    text-decoration: none;
    color: #333;
    font-weight: 500;
}

.nav-menu a:hover {
    color: #3b82f6;
}

.cart-link {
    background-color: #3b82f6;
    color: white !important;
    padding: 8px 16px;
    border-radius: 20px;
    font-size: 14px;
}

.cart-link:hover {
    background-color: #2563eb !important;
}
```

> 🎓 **What we learned:**
>
> - **CSS Reset** removes inconsistent default browser styles
> - **Flexbox** helps us arrange items in rows (like our navigation)
> - **Box-shadow** creates nice drop shadows
> - **Position: fixed** keeps the navbar at the top when scrolling
> - **Hover effects** make our site feel interactive
> - **Colors** can be written as names (white) or hex codes (🟦 `#3b82f6`)

Lesson 4: Hero Section Styling

The hero section is the big, eye-catching area at the top of our homepage.

Add to `style.css`:

```css
/* Hero Section */
.hero {
    background: linear-gradient(135deg, #3b82f6, #1d4ed8);
    color: white;
    text-align: center;
    padding: 150px 20px 100px;
    margin-top: 70px; /* Space for fixed navbar */
}

.hero h1 {
    font-size: 48px;
    margin-bottom: 20px;
    font-weight: bold;
}

.hero p {
    font-size: 20px;
    margin-bottom: 30px;
}

/* Button styling */
.btn {
    display: inline-block;
    background-color: white;
    color: #3b82f6;
    padding: 12px 30px;
    text-decoration: none;
    border-radius: 5px;
    font-weight: bold;
    transition: all 0.3s ease;
}

.btn:hover {
    background-color: #f8fafc;
    transform: translateY(-2px);
}
```

> 🎓 **What we learned:**
>
> - **Linear-gradient** creates beautiful color transitions
> - **Margin-top** pushes content down to avoid overlapping the fixed navbar
> - **Transition** makes hover effects smooth instead of instant
> - **Transform: translateY** moves elements up or down (great for hover effects)

Lesson 5: Featured Products Section

Add to `style.css`:

css

```css
/* Featured Products Section */
.featured {
    padding: 80px 0;
    background-color: #f8fafc;
}

.featured h2 {
    text-align: center;
    font-size: 36px;
    margin-bottom: 50px;
    color: #1e293b;
}

/* Products Grid */
.products-grid {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));
    gap: 30px;
}

/* Product Card */
.product-card {
    background: white;
    border-radius: 10px;
    box-shadow: 0 4px 6px rgba(0,0,0,0.1);
    overflow: hidden;
    transition: transform 0.3s ease;
}

.product-card:hover {
    transform: translateY(-5px);
}

.product-image {
    width: 100%;
    height: 200px;
    object-fit: cover;
    background-color: #e2e8f0;
}

.product-info {
    padding: 20px;
}

.product-title {
    font-size: 18px;
    font-weight: bold;
    margin-bottom: 10px;
    color: #1e293b;
}

.product-description {
    color: #64748b;
    font-size: 14px;
    margin-bottom: 15px;
}

.product-price {
    font-size: 20px;
    font-weight: bold;
    color: #3b82f6;
    margin-bottom: 15px;
}

.product-actions {
    display: flex;
    gap: 10px;
}

.btn-small {
    padding: 8px 16px;
    font-size: 14px;
    flex: 1;
}

.btn-primary {
    background-color: #3b82f6;
    color: white;
}

.btn-primary:hover {
    background-color: #2563eb;
}

.btn-secondary {
    background-color: #e2e8f0;
    color: #334155;
}

.btn-secondary:hover {
    background-color: #cbd5e1;
}
```

> 🎓 **What we learned:**
> - **CSS Grid** automatically arranges products in rows and columns
> - **auto-fit** and **minmax()** make our grid responsive (adapts to screen size)
> - **Object-fit: cover** keeps images from stretching weirdly
> - **Overflow: hidden** prevents content from spilling out of rounded corners
> - **Flex: 1** makes buttons grow to fill available space equally

Lesson 6: Footer Styling

Add to `style.css`:

```css
/* Footer */
.footer {
    background-color: #1e293b;
    color: white;
    text-align: center;
    padding: 40px 0;
}
```

> 🎓 **What we learned:**
> - Simple, clean footer styling
> - Dark background with light text for contrast

🛍️ Products Page

Lesson 7: Products Page HTML

Now let's create a page to show all our products with filtering options.

Create `products.html`:

```
html
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Products - TechVibe</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <!-- Same navigation as homepage -->
    <nav class="navbar">
        <div class="nav-container">
            <h2 class="logo">TechVibe</h2>
            <ul class="nav-menu">
                <li><a href="index.html">Home</a></li>
                <li><a href="products.html">Products</a></li>
                <li><a href="contact.html">Contact</a></li>
                <li><a href="cart.html" class="cart-link">Cart (<span id="cart-count">
            </ul>
        </div>
    </nav>

    <main class="main-content">
        <div class="container">
            <h1 class="page-title">Our Products</h1>

            <!-- Filter buttons -->
            <div class="filters">
                <button class="filter-btn active" data-category="all">All Products</bu
                <button class="filter-btn" data-category="phones">Phones</button>
                <button class="filter-btn" data-category="laptops">Laptops</button>
                <button class="filter-btn" data-category="accessories">Accessories</bu
            </div>

            <!-- Products will appear here -->
            <div class="products-grid" id="products-grid">
                <!-- JavaScript will fill this -->
            </div>
        </div>
    </main>

    <footer class="footer">
        <p>&copy; 2024 TechVibe. All rights reserved.</p>
    </footer>

    <script src="script.js"></script>
```

```
</body>
</html>
```

> 🎓 **What we learned:**
>
> - **data-category** is a custom attribute that JavaScript can read
> - **span** with id="cart-count" will be updated by JavaScript
> - **main** tag is semantic HTML for the main content area

Lesson 8: Products Page Styling

Add to `style.css`:

```css
/* Main content area */
.main-content {
    margin-top: 70px;
    padding: 60px 0;
    min-height: calc(100vh - 140px);
}

.page-title {
    text-align: center;
    font-size: 36px;
    margin-bottom: 40px;
    color: #1e293b;
}

/* Filter buttons */
.filters {
    display: flex;
    justify-content: center;
    gap: 15px;
    margin-bottom: 40px;
    flex-wrap: wrap;
}

.filter-btn {
    padding: 10px 20px;
    background-color: #e2e8f0;
    color: #64748b;
    border: none;
    border-radius: 25px;
    cursor: pointer;
    font-weight: 500;
    transition: all 0.3s ease;
}

.filter-btn:hover, .filter-btn.active {
    background-color: #3b82f6;
    color: white;
}
```

> 🎓 **What we learned:**
>
> - **calc()** lets us do math in CSS (100vh = full screen height)
> - **flex-wrap** allows items to wrap to next line on small screens
> - **cursor: pointer** shows a hand cursor on buttons

> - **.active** class indicates the currently selected filter

---

🚀 Introduction to JavaScript

Lesson 9: Your First JavaScript - Creating Product Data

JavaScript brings our website to life! Let's start by creating our product information.

Create `script.js`:

```
javascript
```

```javascript
// Welcome to JavaScript!
// This is a comment - it doesn't run, it just explains what we're doing

// Step 1: Create our product data
// Think of this like a digital catalog of all our products
const products = [
    {
        id: 1,
        name: "iPhone 15 Pro",
        price: 999,
        category: "phones",
        image: "https://via.placeholder.com/300x200/3b82f6/white?text=iPhone+15+Pro",
        description: "The latest iPhone with amazing camera and performance"
    },
    {
        id: 2,
        name: "MacBook Air",
        price: 1199,
        category: "laptops",
        image: "https://via.placeholder.com/300x200/10b981/white?text=MacBook+Air",
        description: "Lightweight laptop perfect for work and creativity"
    },
    {
        id: 3,
        name: "AirPods Pro",
        price: 249,
        category: "accessories",
        image: "https://via.placeholder.com/300x200/f59e0b/white?text=AirPods+Pro",
        description: "Wireless earbuds with noise cancellation"
    },
    {
        id: 4,
        name: "Samsung Galaxy S24",
        price: 899,
        category: "phones",
        image: "https://via.placeholder.com/300x200/8b5cf6/white?text=Galaxy+S24",
        description: "Android phone with incredible features"
    },
    {
        id: 5,
        name: "Dell Laptop",
        price: 799,
        category: "laptops",
        image: "https://via.placeholder.com/300x200/06b6d4/white?text=Dell+Laptop",
        description: "Reliable laptop for everyday computing"
    },
    {
        id: 6,
        name: "Wireless Mouse",
        price: 49,
        category: "accessories",
        image: "https://via.placeholder.com/300x200/ec4899/white?text=Wireless+Mouse",
        description: "Ergonomic wireless mouse for productivity"
    }
];

// Step 2: Create our shopping cart
// This will store items that customers want to buy
let cart = [];

// Step 3: Get references to HTML elements
// This connects our JavaScript to specific parts of our webpage
const cartCountElement = document.getElementById('cart-count');
const productsGrid = document.getElementById('products-grid');
const featuredProducts = document.getElementById('featured-products');

// Step 4: Utility function to format prices
// This makes "$999.00" instead of just "999"
function formatPrice(price) {
    return '$' + price.toFixed(2);
}

console.log('JavaScript loaded successfully!');
console.log('We have', products.length, 'products');
```

> 🎓 **What we learned:**
> - **const** creates variables that don't change (our product list)
> - **let** creates variables that can change (our cart)
> - **Arrays** (the [ ] brackets) store lists of things
> - **Objects** (the { } brackets) store related information together
> - **console.log()** helps us debug by showing messages in browser tools
> - **Functions** are reusable pieces of code that do specific tasks

Lesson 10: Displaying Products with JavaScript

Now let's make our products appear on the webpage!

Add to `script.js`:

```
javascript
```

```
// Function to create HTML for one product card
function createProductCard(product) {
    // Template literals (backticks) let us create HTML with JavaScript
    return `
        <div class="product-card">
            <img src="${product.image}" alt="${product.name}" class="product-image">
            <div class="product-info">
                <h3 class="product-title">${product.name}</h3>
                <p class="product-description">${product.description}</p>
                <div class="product-price">${formatPrice(product.price)}</div>
                <div class="product-actions">
                    <button class="btn btn-primary btn-small" onclick="addToCart(${pro
                        Add to Cart
                    </button>
                    <button class="btn btn-secondary btn-small" onclick="viewProduct($
                        View Details
                    </button>
                </div>
            </div>
        </div>
    `;
}

// Function to display products on the page
function displayProducts(productsToShow = products) {
    // If we're on the products page
    if (productsGrid) {
        // Create HTML for each product and join them together
        const productsHTML = productsToShow.map(createProductCard).join('');
        productsGrid.innerHTML = productsHTML;
    }

    // If we're on the homepage, show first 3 products as featured
    if (featuredProducts) {
        const featuredHTML = productsToShow.slice(0, 3).map(createProductCard).join(''
        featuredProducts.innerHTML = featuredHTML;
    }
}

// Function to add product to cart (we'll build this next)
function addToCart(productId) {
    alert('Adding product ' + productId + ' to cart! (We\'ll build this feature next)'
}

// Function to view product details
function viewProduct(productId) {
    const product = products.find(p => p.id === productId);
    alert('Product: ' + product.name + '\nPrice: ' + formatPrice(product.price) + '\n\
}

// Wait for the page to load, then display products
document.addEventListener('DOMContentLoaded', function() {
    console.log('Page loaded, displaying products...');
    displayProducts();
});
```

> 🎓 **What we learned:**
> - **Template literals** (`backticks`) let us mix JavaScript variables with HTML
> - **${variable}** inserts JavaScript values into our HTML strings
> - **map()** transforms each item in an array (turns products into HTML)
> - **join('')** combines an array of strings into one big string
> - **innerHTML** puts our HTML into the webpage
> - **onclick** runs JavaScript when someone clicks a button
> - **find()** searches an array for something specific
> - **DOMContentLoaded** waits for the webpage to be ready before running our code

Lesson 11: Product Filtering

Let's make our filter buttons work!

Add to `script.js`:

```javascript
// Function to handle filter button clicks
function setupFilters() {
    // Get all filter buttons
    const filterButtons = document.querySelectorAll('.filter-btn');

    // Add click event to each button
    filterButtons.forEach(button => {
        button.addEventListener('click', function() {
            // Remove 'active' class from all buttons
            filterButtons.forEach(btn => btn.classList.remove('active'));

            // Add 'active' class to clicked button
            this.classList.add('active');

            // Get the category from the button's data-category attribute
            const category = this.getAttribute('data-category');

            // Filter products based on category
            let filteredProducts;
            if (category === 'all') {
                filteredProducts = products; // Show all products
            } else {
                filteredProducts = products.filter(product => product.category === cat
            }

            // Display the filtered products
            displayProducts(filteredProducts);

            console.log('Showing', filteredProducts.length, 'products in category:', c
        });
    });
}

// Update our page load function
document.addEventListener('DOMContentLoaded', function() {
    console.log('Page loaded, displaying products...');
    displayProducts();
    setupFilters(); // Add this line
});
```

> 🎓 **What we learned:**
>
> - **querySelectorAll()** finds all elements that match a pattern
> - **forEach()** runs code for each item in a list

> - **addEventListener()** listens for events (like clicks)
> - **this** refers to the element that was clicked
> - **classList** lets us add/remove CSS classes
> - **getAttribute()** gets the value of HTML attributes
> - **filter()** creates a new array with only items that match a condition

---

🛒 Building the Shopping Cart

Lesson 12: Adding Items to Cart

Now for the exciting part - let's build a real shopping cart!

Replace the simple addToCart function in `script.js`:

```
javascript
```

```javascript
// Update cart count display
function updateCartCount() {
    // Calculate total number of items in cart
    const totalItems = cart.reduce((sum, item) => sum + item.quantity, 0);

    // Update the cart count in navigation
    if (cartCountElement) {
        cartCountElement.textContent = totalItems;
    }

    console.log('Cart now has', totalItems, 'items');
}

// Save cart to browser storage so it persists between page visits
function saveCart() {
    localStorage.setItem('techvibe-cart', JSON.stringify(cart));
}

// Load cart from browser storage
function loadCart() {
    const savedCart = localStorage.getItem('techvibe-cart');
    if (savedCart) {
        cart = JSON.parse(savedCart);
        updateCartCount();
        console.log('Loaded cart with', cart.length, 'different products');
    }
}

// Add product to cart
function addToCart(productId) {
    // Find the product in our products array
    const product = products.find(p => p.id === productId);
    if (!product) {
        console.error('Product not found!');
        return;
    }

    // Check if product is already in cart
    const existingItem = cart.find(item => item.id === productId);

    if (existingItem) {
        // If it's already in cart, increase quantity
        existingItem.quantity += 1;
        console.log('Increased quantity of', product.name, 'to', existingItem.quantity
    } else {
        // If it's new, add it to cart
        cart.push({
            id: product.id,
            name: product.name,
            price: product.price,
            image: product.image,
            quantity: 1
        });
        console.log('Added', product.name, 'to cart');
    }

    updateCartCount();
    saveCart();
    showNotification(product.name + ' added to cart!');
}

// Show notification when item is added
function showNotification(message) {
    // Create notification element
    const notification = document.createElement('div');
    notification.className = 'notification';
    notification.textContent = message;

    // Add to page
    document.body.appendChild(notification);

    // Remove after 3 seconds
    setTimeout(() => {
        notification.remove();
    }, 3000);
}

// Update our page load function to load the cart
document.addEventListener('DOMContentLoaded', function() {
    console.log('Page loaded, displaying products...');
    loadCart(); // Load saved cart
    displayProducts();
    setupFilters();
});
```

> 🎓 **What we learned:**
>
> - **reduce()** combines all items in an array into a single value (like adding up quantities)
> - **localStorage** saves data in the browser so it persists between visits
> - **JSON.stringify()** converts JavaScript objects to text for storage
> - **JSON.parse()** converts stored text back to JavaScript objects

> - **push()** adds new items to an array
> - **createElement()** creates new HTML elements with JavaScript
> - **setTimeout()** runs code after a delay
> - **remove()** deletes elements from the page

Lesson 13: Notification Styling

Let's style our notification popup!

Add to `style.css`:

```css
/* Notification popup */
.notification {
    position: fixed;
    top: 90px;
    right: 20px;
    background-color: #10b981;
    color: white;
    padding: 15px 25px;
    border-radius: 5px;
    box-shadow: 0 4px 6px rgba(0,0,0,0.1);
    z-index: 200;
    animation: slideIn 0.3s ease;
}

/* Animation for notification */
@keyframes slideIn {
    from {
        transform: translateX(100%);
        opacity: 0;
    }
    to {
        transform: translateX(0);
        opacity: 1;
    }
}
```

> 🎓 **What we learned:**
> - **position: fixed** keeps the notification in the same spot even when scrolling
> - **z-index** controls which elements appear in front of others
> - **@keyframes** creates custom animations
> - **transform: translateX** moves elements left or right

> - **opacity** controls transparency (0 = invisible, 1 = fully visible)

🛒 Cart Page

Lesson 14: Cart Page HTML

Create `cart.html`:

```
html
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Shopping Cart - TechVibe</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <nav class="navbar">
        <div class="nav-container">
            <h2 class="logo">TechVibe</h2>
            <ul class="nav-menu">
                <li><a href="index.html">Home</a></li>
                <li><a href="products.html">Products</a></li>
                <li><a href="contact.html">Contact</a></li>
                <li><a href="cart.html" class="cart-link">Cart (<span id="cart-count">
            </ul>
        </div>
    </nav>

    <main class="main-content">
        <div class="container">
            <h1 class="page-title">Shopping Cart</h1>

            <div class="cart-layout">
                <!-- Cart items will go here -->
                <div class="cart-items" id="cart-items">
                    <!-- JavaScript will fill this -->
                </div>

                <!-- Order summary -->
                <div class="cart-summary">
                    <div class="summary-card">
                        <h3>Order Summary</h3>
                        <div class="summary-row">
                            <span>Subtotal:</span>
                            <span id="cart-subtotal">$0.00</span>
                        </div>
                        <div class="summary-row">
                            <span>Shipping:</span>
                            <span id="cart-shipping">$9.99</span>
                        </div>
                        <div class="summary-row">
                            <span>Tax:</span>
                            <span id="cart-tax">$0.00</span>
                        </div>
                        <div class="summary-row total-row">
                            <span>Total:</span>
                            <span id="cart-total">$0.00</span>
                        </div>
                        <button class="btn btn-primary full-width" onclick="goToChecko
                            Proceed to Checkout
                        </button>
                        <a href="products.html" class="btn btn-secondary full-width">
                            Continue Shopping
                        </a>
                    </div>
                </div>
            </div>

            <!-- Empty cart message -->
            <div id="empty-cart" class="empty-cart" style="display: none;">
                <h2>Your cart is empty</h2>
                <p>Looks like you haven't added anything yet.</p>
                <a href="products.html" class="btn btn-primary">Start Shopping</a>
            </div>
        </div>
    </main>

    <footer class="footer">
        <p>&copy; 2024 TechVibe. All rights reserved.</p>
    </footer>

    <script src="script.js"></script>
</body>
</html>
```

Lesson 15: Cart Page Styling

Add to `style.css`:

CSS

```css
/* Cart page layout */
.cart-layout {
    display: grid;
    grid-template-columns: 2fr 1fr;
    gap: 40px;
    margin-bottom: 60px;
}

/* Individual cart item */
.cart-item {
    display: flex;
    align-items: center;
    gap: 20px;
    padding: 20px;
    background: white;
    border-radius: 10px;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
    border: 1px solid #e2e8f0;
    margin-bottom: 15px;
}

.cart-item-image {
    width: 80px;
    height: 80px;
    object-fit: cover;
    border-radius: 5px;
}

.cart-item-details {
    flex: 1;
}

.cart-item-name {
    font-size: 18px;
    font-weight: bold;
    margin-bottom: 5px;
    color: #1e293b;
}

.cart-item-price {
    color: #64748b;
    font-size: 14px;
}

.cart-item-controls {
    display: flex;
    align-items: center;
    gap: 15px;
}

/* Quantity controls */
.quantity-controls {
    display: flex;
    align-items: center;
    gap: 10px;
    background-color: #f1f5f9;
    border-radius: 5px;
    padding: 5px;
}

.quantity-btn {
    width: 30px;
    height: 30px;
    border: none;
    background-color: white;
    border-radius: 3px;
    cursor: pointer;
    font-weight: bold;
    color: #64748b;
}

.quantity-btn:hover {
    background-color: #3b82f6;
    color: white;
}

.quantity-display {
    font-weight: bold;
    min-width: 20px;
    text-align: center;
}

.cart-item-total {
    font-weight: bold;
    font-size: 18px;
    color: #3b82f6;
    min-width: 80px;
    text-align: right;
}

.remove-btn {
    background-color: #ef4444;
    color: white;
    border: none;
    padding: 8px 12px;
    border-radius: 5px;
    cursor: pointer;
    font-size: 12px;
}

.remove-btn:hover {
    background-color: #dc2626;
}

/* Cart Summary */
.summary-card {
    background: white;
    padding: 25px;
    border-radius: 10px;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
    border: 1px solid #e2e8f0;
    position: sticky;
    top: 90px;
}

.summary-card h3 {
    font-size: 20px;
    margin-bottom: 20px;
    color: #1e293b;
}

.summary-row {
    display: flex;
    justify-content: space-between;
    margin-bottom: 12px;
    padding: 8px 0;
}

.total-row {
    border-top: 2px solid #e2e8f0;
    font-weight: bold;
    font-size: 18px;
    color: #1e293b;
    margin-top: 15px;
    padding-top: 15px;
}

.full-width {
    width: 100%;
    margin-bottom: 10px;
```

```
}

/* Empty cart */
.empty-cart {
    text-align: center;
    padding: 80px 20px;
}

.empty-cart h2 {
    font-size: 28px;
    color: #64748b;
    margin-bottom: 15px;
}

.empty-cart p {
    color: #94a3b8;
    margin-bottom: 30px;
    font-size: 16px;
}

/* Mobile responsive */
@media (max-width: 768px) {
    .cart-layout {
        grid-template-columns: 1fr;
    }

    .cart-item {
        flex-direction: column;
        align-items: flex-start;
        gap: 15px;
    }

    .cart-item-controls {
        width: 100%;
        justify-content: space-between;
    }
}
```

> 🎓 **What we learned:**
>
> - **position: sticky** makes the summary card stick to the top when scrolling
> - **flex: 1** makes an element grow to fill available space
> - **min-width** prevents elements from getting too small
> - **Media queries** (@media) apply different styles on mobile devices
> - **Grid-template-columns: 1fr** makes a single column layout on mobile

Lesson 16: Cart JavaScript Functionality

Now let's add the JavaScript to make our cart page work!

Add to `script.js`:

```
javascript
```

```
// Function to update item quantity in cart
function updateQuantity(productId, newQuantity) {
    console.log('Updating quantity for product', productId, 'to', newQuantity);

    // Find the item in our cart
    const item = cart.find(item => item.id === productId);
    if (item) {
        if (newQuantity <= 0) {
            // If quantity is 0 or less, remove the item
            removeFromCart(productId);
        } else {
            // Update the quantity
            item.quantity = newQuantity;
            updateCartCount();
            saveCart();
            displayCartItems();
            updateCartSummary();
        }
    }
}

// Function to remove item from cart
function removeFromCart(productId) {
    console.log('Removing product', productId, 'from cart');

    // Filter out the item we want to remove
    cart = cart.filter(item => item.id !== productId);
    updateCartCount();
    saveCart();
    displayCartItems();
    updateCartSummary();
    showNotification('Item removed from cart');
}

// Function to create HTML for cart item
function createCartItemHTML(item) {
    return `
        <div class="cart-item">
            <img src="${item.image}" alt="${item.name}" class="cart-item-image">
            <div class="cart-item-details">
                <h4 class="cart-item-name">${item.name}</h4>
                <p class="cart-item-price">${formatPrice(item.price)} each</p>
            </div>
            <div class="cart-item-controls">
                <div class="quantity-controls">
                    <button class="quantity-btn" onclick="updateQuantity(${item.id}, $
                    <span class="quantity-display">${item.quantity}</span>
                    <button class="quantity-btn" onclick="updateQuantity(${item.id}, $
                </div>
                <div class="cart-item-total">${formatPrice(item.price * item.quantity)
                <button class="remove-btn" onclick="removeFromCart(${item.id})">Remove
            </div>
        </div>
    `;
}

// Function to display cart items
function displayCartItems() {
    const cartItemsContainer = document.getElementById('cart-items');
    const emptyCartElement = document.getElementById('empty-cart');

    // Only run this code if we're on the cart page
    if (!cartItemsContainer) return;

    if (cart.length === 0) {
        // Show empty cart message
        cartItemsContainer.style.display = 'none';
        if (emptyCartElement) emptyCartElement.style.display = 'block';
        document.querySelector('.cart-summary').style.display = 'none';
    } else {
        // Show cart items
        cartItemsContainer.style.display = 'block';
        if (emptyCartElement) emptyCartElement.style.display = 'none';
        document.querySelector('.cart-summary').style.display = 'block';

        // Create HTML for all cart items
        cartItemsContainer.innerHTML = cart.map(createCartItemHTML).join('');
    }
}

// Function to calculate and update cart summary
function updateCartSummary() {
    // Calculate totals
    const subtotal = cart.reduce((sum, item) => sum + (item.price * item.quantity), 0)
    const shipping = subtotal > 0 ? 9.99 : 0;
    const tax = subtotal * 0.08; // 8% tax
    const total = subtotal + shipping + tax;

    // Update the display
    const subtotalElement = document.getElementById('cart-subtotal');
    const shippingElement = document.getElementById('cart-shipping');
    const taxElement = document.getElementById('cart-tax');
    const totalElement = document.getElementById('cart-total');
    if (subtotalElement) subtotalElement.textContent = formatPrice(subtotal);
    if (shippingElement) shippingElement.textContent = shipping > 0 ? formatPrice(ship
    if (taxElement) taxElement.textContent = formatPrice(tax);
    if (totalElement) totalElement.textContent = formatPrice(total);

    console.log('Cart summary - Subtotal:', formatPrice(subtotal), 'Total:', formatPri
}

// Function to go to checkout
function goToCheckout() {
    if (cart.length === 0) {
        alert('Your cart is empty!');
        return;
    }
    window.location.href = 'checkout.html';
}

// Update our page load function to handle cart page
document.addEventListener('DOMContentLoaded', function() {
    console.log('Page loaded, displaying products...');
    loadCart();
    displayProducts();
    setupFilters();

    // If we're on the cart page, display cart items
    if (document.getElementById('cart-items')) {
        displayCartItems();
        updateCartSummary();
    }
});
```

> 🎓 **What we learned:**
>
> - **filter()** creates a new array excluding items that don't match a condition
> - **reduce()** can calculate sums (adding up prices × quantities)
> - **window.location.href** navigates to a different page
> - **Conditional logic** (if statements) lets our code make decisions
> - **Template literals** make it easy to create complex HTML with JavaScript
> - **Error handling** with alerts provides user feedback

💳 Checkout Process

Lesson 17: Checkout Page HTML

Create `checkout.html`:

```
html
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Checkout - TechVibe</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <nav class="navbar">
        <div class="nav-container">
            <h2 class="logo">TechVibe</h2>
            <ul class="nav-menu">
                <li><a href="index.html">Home</a></li>
                <li><a href="products.html">Products</a></li>
                <li><a href="contact.html">Contact</a></li>
                <li><a href="cart.html" class="cart-link">Cart (<span id="cart-count">
            </ul>
        </div>
    </nav>

    <main class="main-content">
        <div class="container">
            <h1 class="page-title">Checkout</h1>

            <div class="checkout-layout">
                <!-- Shipping Form -->
                <div class="checkout-form">
                    <form id="checkout-form">
                        <div class="form-section">
                            <h3>Shipping Information</h3>
                            <div class="form-row">
                                <div class="form-group">
                                    <label for="firstName">First Name</label>
                                    <input type="text" id="firstName" name="firstName"
                                </div>
                                <div class="form-group">
                                    <label for="lastName">Last Name</label>
                                    <input type="text" id="lastName" name="lastName" r
                                </div>
                            </div>
                            <div class="form-group">
                                <label for="email">Email Address</label>
                                <input type="email" id="email" name="email" required>
                            </div>
                            <div class="form-group">
                                <label for="address">Street Address</label>
                                <input type="text" id="address" name="address" require
                            </div>
                            <div class="form-row">
                                <div class="form-group">
                                    <label for="city">City</label>
                                    <input type="text" id="city" name="city" required>
                                </div>
                                <div class="form-group">
                                    <label for="state">State</label>
                                    <select id="state" name="state" required>
                                        <option value="">Select State</option>
                                        <option value="CA">California</option>
                                        <option value="NY">New York</option>
                                        <option value="TX">Texas</option>
                                        <option value="FL">Florida</option>
                                    </select>
                                </div>
                                <div class="form-group">
                                    <label for="zipCode">ZIP Code</label>
                                    <input type="text" id="zipCode" name="zipCode" req
                                </div>
                            </div>
                        </div>

                        <div class="form-section">
                            <h3>Payment Information</h3>
                            <div class="form-group">
                                <label for="cardNumber">Card Number</label>
                                <input type="text" id="cardNumber" name="cardNumber" p
                            </div>
                            <div class="form-row">
                                <div class="form-group">
                                    <label for="expiryDate">Expiry Date</label>
                                    <input type="text" id="expiryDate" name="expiryDat
                                </div>
                                <div class="form-group">
                                    <label for="cvv">CVV</label>
                                    <input type="text" id="cvv" name="cvv" placeholder
                                </div>
                            </div>
                        </div>

                        <button type="submit" class="btn btn-primary full-width btn-la
                            Complete Order
                        </button>
                    </form>
```

```html
                </div>

                <!-- Order Summary -->
                <div class="order-summary">
                    <div class="summary-card">
                        <h3>Your Order</h3>
                        <div id="checkout-items">
                            <!-- Order items will be loaded here -->
                        </div>
                        <div class="summary-totals">
                            <div class="summary-row">
                                <span>Subtotal:</span>
                                <span id="checkout-subtotal">$0.00</span>
                            </div>
                            <div class="summary-row">
                                <span>Shipping:</span>
                                <span id="checkout-shipping">$9.99</span>
                            </div>
                            <div class="summary-row">
                                <span>Tax:</span>
                                <span id="checkout-tax">$0.00</span>
                            </div>
                            <div class="summary-row total-row">
                                <span>Total:</span>
                                <span id="checkout-total">$0.00</span>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </main>

    <footer class="footer">
        <p>&copy; 2024 TechVibe. All rights reserved.</p>
    </footer>

    <script src="script.js"></script>
</body>
</html>
```

Lesson 18: Checkout Form Styling

Add to `style.css`:

css

```css
/* Checkout page layout */
.checkout-layout {
    display: grid;
    grid-template-columns: 2fr 1fr;
    gap: 40px;
    margin-bottom: 60px;
}

/* Form styling */
.checkout-form {
    background: white;
    padding: 30px;
    border-radius: 10px;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
    border: 1px solid #e2e8f0;
}

.form-section {
    margin-bottom: 30px;
}

.form-section h3 {
    font-size: 20px;
    margin-bottom: 20px;
    color: #1e293b;
    border-bottom: 2px solid #e2e8f0;
    padding-bottom: 10px;
}

.form-row {
    display: grid;
    grid-template-columns: 1fr 1fr;
    gap: 15px;
}

.form-row.three-col {
    grid-template-columns: 1fr 1fr 1fr;
}

.form-group {
    margin-bottom: 15px;
}

.form-group label {
    display: block;
    font-weight: 500;
    color: #374151;
    margin-bottom: 5px;
}

.form-group input,
.form-group select {
    width: 100%;
    padding: 12px;
    border: 2px solid #d1d5db;
    border-radius: 5px;
    font-size: 16px;
    transition: border-color 0.3s ease;
}

.form-group input:focus,
.form-group select:focus {
    outline: none;
    border-color: #3b82f6;
}

.btn-large {
    padding: 15px 30px;
    font-size: 18px;
    font-weight: bold;
}

/* Checkout items display */
.checkout-items {
    margin-bottom: 20px;
}

.checkout-item {
    display: flex;
    align-items: center;
    gap: 12px;
    padding: 12px 0;
    border-bottom: 1px solid #e2e8f0;
}

.checkout-item:last-child {
    border-bottom: none;
}

.checkout-item-image {
    width: 50px;
    height: 50px;
    object-fit: cover;
    border-radius: 5px;
}

.checkout-item-details {
    flex: 1;
}

.checkout-item-name {
    font-weight: 500;
    font-size: 14px;
    color: #1e293b;
}

.checkout-item-quantity {
    font-size: 12px;
    color: #64748b;
}

.checkout-item-price {
    font-weight: bold;
    color: #3b82f6;
}

.summary-totals {
    border-top: 2px solid #e2e8f0;
    padding-top: 15px;
}

/* Success message */
.success-message {
    background-color: #10b981;
    color: white;
    padding: 40px;
    border-radius: 10px;
    text-align: center;
    margin-bottom: 30px;
}

.success-message h2 {
    font-size: 24px;
    margin-bottom: 15px;
}

/* Mobile responsive */
@media (max-width: 768px) {
    .checkout-layout {
        grid-template-columns: 1fr;
    }

    .form-row {
        grid-template-columns: 1fr;
    }

    .form-row.three-col {
        grid-template-columns: 1fr;
    }
}
```

Lesson 19: Checkout JavaScript

Add to `script.js`:

```
javascript
```

```javascript
// Function to display checkout items
function displayCheckoutItems() {
    const checkoutItemsContainer = document.getElementById('checkout-items');
    if (!checkoutItemsContainer || cart.length === 0) return;

    const itemsHTML = cart.map(item => `
        <div class="checkout-item">
            <img src="${item.image}" alt="${item.name}" class="checkout-item-image">
            <div class="checkout-item-details">
                <div class="checkout-item-name">${item.name}</div>
                <div class="checkout-item-quantity">Qty: ${item.quantity}</div>
            </div>
            <div class="checkout-item-price">${formatPrice(item.price * item.quantity)
        </div>
    `).join('');

    checkoutItemsContainer.innerHTML = itemsHTML;
}

// Function to update checkout summary
function updateCheckoutSummary() {
    const subtotal = cart.reduce((sum, item) => sum + (item.price * item.quantity), 0)
    const shipping = subtotal > 0 ? 9.99 : 0;
    const tax = subtotal * 0.08;
    const total = subtotal + shipping + tax;

    // Update checkout summary elements
    const subtotalElement = document.getElementById('checkout-subtotal');
    const shippingElement = document.getElementById('checkout-shipping');
    const taxElement = document.getElementById('checkout-tax');
    const totalElement = document.getElementById('checkout-total');

    if (subtotalElement) subtotalElement.textContent = formatPrice(subtotal);
    if (shippingElement) shippingElement.textContent = shipping > 0 ? formatPrice(ship
    if (taxElement) taxElement.textContent = formatPrice(tax);
    if (totalElement) totalElement.textContent = formatPrice(total);
}

// Simple form validation functions
function validateEmail(email) {
    // Check if email contains @ and .
    return email.includes('@') && email.includes('.');
}

function validateCardNumber(cardNumber) {
    // Remove spaces and check if it's 16 digits
    const cleanNumber = cardNumber.replace(/\s/g, '');
    return /^\d{16}$/.test(cleanNumber);
}

// Function to process the order (simulate)
function processOrder(formData) {
    // In a real website, this would send data to a server
    // For now, we'll just simulate it with a delay
    return new Promise((resolve) => {
        setTimeout(() => {
            // Clear the cart after successful order
            cart = [];
            updateCartCount();
            saveCart();
            resolve({
                success: true,
                orderNumber: 'TV-' + Date.now(),
                message: 'Your order has been placed successfully!'
            });
        }, 2000);
    });
}

// Function to show order success
function showOrderSuccess(orderInfo) {
    const container = document.querySelector('.container');
    container.innerHTML = `
        <div class="success-message">
            <h2>🎉 Order Placed Successfully!</h2>
            <p>Thank you for your purchase!</p>
            <p><strong>Order Number:</strong> ${orderInfo.orderNumber}</p>
            <p>You will receive a confirmation email shortly.</p>
            <a href="index.html" class="btn btn-primary">Continue Shopping</a>
        </div>
    `;
}

// Update our page load function to handle checkout
document.addEventListener('DOMContentLoaded', function() {
    console.log('Page loaded...');
    loadCart();
    displayProducts();
    setupFilters();

    // Cart page
    if (document.getElementById('cart-items')) {
        displayCartItems();
    updateCartSummary();
}

// Checkout page
if (document.getElementById('checkout-form')) {
    // Redirect if cart is empty
    if (cart.length === 0) {
        alert('Your cart is empty!');
        window.location.href = 'products.html';
        return;
    }

    displayCheckoutItems();
    updateCheckoutSummary();

    // Handle form submission
    const checkoutForm = document.getElementById('checkout-form');
    checkoutForm.addEventListener('submit', async function(e) {
        e.preventDefault(); // Prevent normal form submission

        // Get form data
        const formData = new FormData(checkoutForm);
        const data = Object.fromEntries(formData);

        // Simple validation
        let isValid = true;
        const errors = [];

        if (!validateEmail(data.email)) {
            errors.push('Please enter a valid email address');
            isValid = false;
        }

        if (!validateCardNumber(data.cardNumber)) {
            errors.push('Please enter a valid 16-digit card number');
            isValid = false;
        }

        if (!isValid) {
            alert('Please fix the following errors:\n' + errors.join('\n'));
            return;
        }

        // Show loading state
        const submitBtn = checkoutForm.querySelector('button[type="submit"]');
        const originalText = submitBtn.textContent;
        submitBtn.textContent = 'Processing...';
```

```
            submitBtn.disabled = true;

            try {
                const result = await processOrder(data);
                if (result.success) {
                    showOrderSuccess(result);
                }
            } catch (error) {
                alert('There was an error processing your order. Please try again.');
                submitBtn.textContent = originalText;
                submitBtn.disabled = false;
            }
        });
    }
});
```

> 🎓 **What we learned:**
> - **FormData** captures all form inputs at once
> - **Object.fromEntries()** converts form data to a regular JavaScript object
> - **preventDefault()** stops the default form submission behavior
> - **async/await** handles operations that take time (like processing orders)
> - **Promise** represents an operation that will complete in the future
> - **Regular expressions** (/^\d{16}$/) help validate text patterns
> - **try/catch** handles errors gracefully

📞 Contact Page

Lesson 20: Contact Page HTML & Styling

Create `contact.html`:

```
html
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Contact Us - TechVibe</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <nav class="navbar">
        <div class="nav-container">
            <h2 class="logo">TechVibe</h2>
            <ul class="nav-menu">
                <li><a href="index.html">Home</a></li>
                <li><a href="products.html">Products</a></li>
                <li><a href="contact.html">Contact</a></li>
                <li><a href="cart.html" class="cart-link">Cart (<span id="cart-count">
            </ul>
        </div>
    </nav>

    <main class="main-content">
        <div class="container">
            <h1 class="page-title">Contact Us</h1>

            <div class="contact-layout">
                <div class="contact-info">
                    <h2>Get in Touch</h2>
                    <p>We'd love to hear from you! Send us a message and we'll respond

                    <div class="contact-details">
                        <div class="contact-item">
                            <h4>📍 Address</h4>
                            <p>123 Tech Street<br>Silicon Valley, CA 94000</p>
                        </div>
                        <div class="contact-item">
                            <h4>📞 Phone</h4>
                            <p>(555) 123-4567</p>
                        </div>
                        <div class="contact-item">
                            <h4>✉️ Email</h4>
                            <p>support@techvibe.com</p>
                        </div>
                        <div class="contact-item">
                            <h4>🕒 Hours</h4>
                            <p>Monday - Friday: 9AM - 6PM<br>Saturday: 10AM - 4PM<br>S
                        </div>
                    </div>
                </div>

                <div class="contact-form">
                    <form id="contact-form">
                        <div class="form-group">
                            <label for="name">Full Name</label>
                            <input type="text" id="name" name="name" required>
                        </div>
                        <div class="form-group">
                            <label for="contactEmail">Email Address</label>
                            <input type="email" id="contactEmail" name="email" require
                        </div>
                        <div class="form-group">
                            <label for="subject">Subject</label>
                            <select id="subject" name="subject" required>
                                <option value="">Select a subject</option>
                                <option value="general">General Question</option>
                                <option value="support">Technical Support</option>
                                <option value="order">Order Status</option>
                                <option value="return">Returns</option>
                            </select>
                        </div>
                        <div class="form-group">
                            <label for="message">Message</label>
                            <textarea id="message" name="message" rows="6" required></
                        </div>
                        <button type="submit" class="btn btn-primary full-width">Send
                    </form>
                </div>
            </div>
        </div>
    </main>

    <footer class="footer">
        <p>&copy; 2024 TechVibe. All rights reserved.</p>
    </footer>

    <script src="script.js"></script>
</body>
</html>
```

Add contact page styles to `style.css`:

css

```css
/* Contact page layout */
.contact-layout {
    display: grid;
    grid-template-columns: 1fr 1fr;
    gap: 50px;
    margin-bottom: 60px;
}

.contact-info {
    padding: 30px;
}

.contact-info h2 {
    font-size: 24px;
    margin-bottom: 15px;
    color: #1e293b;
}

.contact-info p {
    color: #64748b;
    line-height: 1.6;
    margin-bottom: 30px;
    font-size: 16px;
}

.contact-details {
    display: flex;
    flex-direction: column;
    gap: 25px;
}

.contact-item h4 {
    font-size: 16px;
    margin-bottom: 8px;
    color: #3b82f6;
}

.contact-item p {
    color: #374151;
    margin: 0;
    font-size: 14px;
}

.contact-form {
    background: white;
    padding: 30px;
    border-radius: 10px;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
    border: 1px solid #e2e8f0;
}

.contact-form textarea {
    width: 100%;
    padding: 12px;
    border: 2px solid #d1d5db;
    border-radius: 5px;
    font-size: 16px;
    font-family: inherit;
    resize: vertical;
    transition: border-color 0.3s ease;
}

.contact-form textarea:focus {
    outline: none;
    border-color: #3b82f6;
}

/* Mobile responsive */
@media (max-width: 768px) {
    .contact-layout {
        grid-template-columns: 1fr;
        gap: 30px;
    }
}
```

Lesson 21: Contact Form JavaScript

Add to `script.js`:

```
javascript
```

```javascript
// Function to handle contact form submission
function handleContactForm() {
    const contactForm = document.getElementById('contact-form');

    if (contactForm) {
        contactForm.addEventListener('submit', function(e) {
            e.preventDefault(); // Stop the form from submitting normally

            // Get form data
            const formData = new FormData(contactForm);
            const data = Object.fromEntries(formData);

            // Simple validation
            let isValid = true;
            const errors = [];

            if (!data.name.trim()) {
                errors.push('Name is required');
                isValid = false;
            }

            if (!validateEmail(data.email)) {
                errors.push('Please enter a valid email address');
                isValid = false;
            }

            if (!data.subject) {
                errors.push('Please select a subject');
                isValid = false;
            }

            if (!data.message.trim()) {
                errors.push('Message is required');
                isValid = false;
            } else if (data.message.trim().length < 10) {
                errors.push('Message must be at least 10 characters long');
                isValid = false;
            }

            if (!isValid) {
                alert('Please fix the following errors:\n' + errors.join('\n'));
                return;
            }

            // Show loading state
            const submitBtn = contactForm.querySelector('button[type="submit"]');
            const originalText = submitBtn.textContent;
            submitBtn.textContent = 'Sending...';
            submitBtn.disabled = true;

            // Simulate sending the message
            setTimeout(() => {
                alert('Thank you for your message! We\'ll get back to you soon.');
                contactForm.reset(); // Clear the form
                submitBtn.textContent = originalText;
                submitBtn.disabled = false;
            }, 1500);
        });
    }
}

// Update our main page load function
document.addEventListener('DOMContentLoaded', function() {
    console.log('Page loaded...');
    loadCart();
    displayProducts();
    setupFilters();

    // Cart page
    if (document.getElementById('cart-items')) {
        displayCartItems();
        updateCartSummary();
    }

    // Checkout page
    if (document.getElementById('checkout-form')) {
        if (cart.length === 0) {
            alert('Your cart is empty!');
            window.location.href = 'products.html';
            return;
        }

        displayCheckoutItems();
        updateCheckoutSummary();

        const checkoutForm = document.getElementById('checkout-form');
        checkoutForm.addEventListener('submit', async function(e) {
            e.preventDefault();

            const formData = new FormData(checkoutForm);
            const data = Object.fromEntries(formData);

            let isValid = true;
            const errors = [];

            if (!validateEmail(data.email)) {
                errors.push('Please enter a valid email address');
                isValid = false;
            }

            if (!validateCardNumber(data.cardNumber)) {
                errors.push('Please enter a valid 16-digit card number');
                isValid = false;
            }

            if (!isValid) {
                alert('Please fix the following errors:\n' + errors.join('\n'));
                return;
            }

            const submitBtn = checkoutForm.querySelector('button[type="submit"]');
            const originalText = submitBtn.textContent;
            submitBtn.textContent = 'Processing...';
            submitBtn.disabled = true;

            try {
                const result = await processOrder(data);
                if (result.success) {
                    showOrderSuccess(result);
                }
            } catch (error) {
                alert('There was an error processing your order. Please try again.');
                submitBtn.textContent = originalText;
                submitBtn.disabled = false;
            }
        });
    }

    // Contact page
    handleContactForm();
});
```

> 🎓 **What we learned:**
>
> - **trim()** removes extra spaces from the beginning and end of text
> - **reset()** clears all form fields
> - **setTimeout()** creates delays (useful for simulating server responses)
> - Form validation helps ensure users provide good data

> - We can reuse validation functions across different forms

🎉 Final Touches & What You've Accomplished!

Lesson 22: Adding Mobile Responsiveness

Let's make sure our site works great on phones! **Add to `style.css`:**

css

```css
/* Mobile Navigation */
@media (max-width: 768px) {
    .nav-menu {
        flex-direction: column;
        gap: 15px;
    }

    .hero h1 {
        font-size: 32px;
    }

    .hero p {
        font-size: 16px;
    }

    .hero {
        padding: 120px 20px 60px;
    }

    .products-grid {
        grid-template-columns: 1fr;
        gap: 20px;
    }

    .filters {
        flex-direction: column;
        align-items: center;
    }

    .page-title {
        font-size: 28px;
    }
}

/* Smooth scrolling */
html {
    scroll-behavior: smooth;
}

/* Loading animation */
.loading {
    display: inline-block;
    width: 20px;
    height: 20px;
    border: 3px solid #f3f3f3;
    border-top: 3px solid #3b82f6;
    border-radius: 50%;
    animation: spin 1s linear infinite;
}

@keyframes spin {
    0% { transform: rotate(0deg); }
    100% { transform: rotate(360deg); }
}
```

> 🎓 **What we learned:**
> - **Media queries** apply different styles based on screen size
> - **@keyframes** creates custom animations
> - **scroll-behavior: smooth** makes page scrolling more elegant

---

🏆 Congratulations! You Built a Complete E-Commerce Website!

🎯 What You've Accomplished:

✅ **5 Complete Web Pages** - Homepage, Products, Cart, Checkout, and Contact
✅ **Interactive Shopping Cart** - Add items, change quantities, remove items
✅ **Product Filtering** - Sort products by category
✅ **Form Validation** - Professional checkout and contact forms
✅ **Local Storage** - Your cart remembers items between visits
✅ **Responsive Design** - Works perfectly on phones and computers
✅ **Professional Styling** - Clean, modern design that looks great

🧠 JavaScript Skills You've Mastered:

- **Variables** (`let`, `const`) - Storing information
- **Arrays** - Lists of products and cart items
- **Objects** - Storing related data together
- **Functions** - Organizing code into reusable pieces
- **Event Listeners** - Responding to clicks and form submissions
- **DOM Manipulation** - Updating webpage content with JavaScript
- **Local Storage** - Saving data in the browser
- **Form Validation** - Checking user input
- **Array Methods** - `map()`, `filter()`, `find()`, `reduce()`
- **Template Literals** - Creating HTML with JavaScript
- **Promises & Async/Await** - Handling time-consuming operations

🎨 CSS Techniques You've Learned:

- **Flexbox** - Arranging items in rows and columns
- **CSS Grid** - Creating responsive layouts
- **Hover Effects** - Interactive button and card animations
- **Media Queries** - Making your site work on all devices
- **Box Shadows** - Adding depth and professionalism
- **Transitions** - Smooth animations between states
- **Form Styling** - Making forms look professional

💡 Web Development Best Practices:

- **Separation of Concerns** - HTML for structure, CSS for style, JS for behavior
- **Code Organization** - Clean, readable, well-commented code
- **User Experience** - Clear feedback, loading states, error handling
- **Mobile-First Design** - Ensuring your site works everywhere
- **Progressive Enhancement** - Starting simple and adding complexity

🚀 What's Next?

You now have the foundation to build any kind of website! Here are some ideas to continue learning:

1. **Add More Features:**
   - Product search functionality
   - User reviews and ratings
   - Wish list feature
   - Product image galleries
2. **Learn New Technologies:**
   - **React** - A popular JavaScript framework
   - **Node.js** - JavaScript for servers
   - **Databases** - Store data permanently
   - **APIs** - Connect to external services
3. **Improve Your Skills:**
   - **Git** - Version control for your code
   - **Testing** - Making sure your code works correctly
   - **Performance** - Making websites load faster
   - **Accessibility** - Making sites usable for everyone

🎓 Key Takeaways:

- **Start Simple**: We began with basic HTML and gradually added complexity
- **Practice Makes Perfect**: Each lesson built upon the previous one
- **Real-World Application**: You built something actually useful!
- **Problem-Solving**: You learned to break down complex features into simple steps
- **Modern Development**: You used current best practices and techniques

You should be incredibly proud of what you've built! 🎉

Your TechVibe e-commerce site demonstrates real web development skills. You've created a fully functional online store with JavaScript cart functionality, professional styling, and a great user experience.

Keep coding, keep building, and most importantly - have fun creating amazing things on the web! 🚀

📄 Complete File Reference

File Structure:

```
TechVibe-Store/
├── index.html          (Homepage)
├── products.html       (Products catalog)
├── cart.html          (Shopping cart)
├── checkout.html      (Checkout form)
├── contact.html       (Contact form)
├── style.css          (All styling)
├── script.js          (All JavaScript)
└── images/            (Product images)
```

Quick Setup Guide:

1. Create the folder structure above
2. Copy the HTML code into each respective file
3. Copy all the CSS code into `style.css`
4. Copy all the JavaScript code into `script.js`
5. Open `index.html` in your browser
6. Start shopping!

Testing Your Site:

- ✅ Homepage loads with featured products

- ✅ Products page shows all items with working filters
- ✅ Add items to cart and see the count update
- ✅ Cart page shows items with quantity controls
- ✅ Checkout form validates input
- ✅ Contact form sends messages
- ✅ All pages work on mobile devices

Remember: Every expert was once a beginner. You've taken the first big step into the exciting world of web development!

This completes your comprehensive e-commerce website tutorial. You now have all the skills needed to build modern, interactive websites with HTML, CSS, and JavaScript!