<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>YCV Shopping Site</title>

    <link rel="stylesheet" href="styles.css">

</head>

<body>

    <header>

        <nav>

            <marquee behavior="scroll" direction="left">WELCOME TO YELLOW COMMUNICATION VENTURE  MULTIPURPOSE ENTERPRISE</marquee>

            <ul>

                <li><a href="#">Home</a></li>

                <li><a href="#">About</a></li>

                <li><a href="#">Contact</a></li>

                <li><input type="text" id="search" placeholder="Search..."></li>

                <li><button id="search-button">Search</button></li>

            </ul>

        </nav>

    </header>

    <!-- Image Slideshow -->

    <div class="slideshow">

        <img src="shopping1.jpg" class="slideshow-image active" alt="Shopping 1">

        <img src="shopping2.WEBP" class="slideshow-image" alt="Shopping 2">

        <img src="shopping3.jpg" class="slideshow-image" alt="Shopping 3">

    </div>

    <main>

        <section class="products" id="products-section">

            <div class="product" data-name="Android Device">

                <img src="phone.jpg" alt="Product 1">

                <p class="price">₦250,000</p>

                <button class="add-to-cart" data-name="Android Device" data-price="250000">Add to Cart</button>

            </div>

            <div class="product" data-name="YCV1">

                <img src="YCV1.jpg" alt="Product 2">

                <p class="price">₦200,000</p>

                <button class="add-to-cart" data-name="YCV1" data-price="200000">Add to Cart</button>

            </div>

            <div class="product" data-name="YCV2">

                <img src="YCV2.jpg" alt="Product 3">

                <p class="price">₦150,000</p>

                <button class="add-to-cart" data-name="YCV2" data-price="150000">Add to Cart</button>

            </div>

            <div class="product" data-name="YCV3">

                <img src="YCV3.jpg" alt="Product 4">

                <p class="price">₦18,800,000</p>

                <button class="add-to-cart" data-name="YCV3" data-price="18800000">Add to Cart</button>

            </div>

            <div class="product" data-name="YCV4">

                <img src="YCV4.jpg" alt="Product 5">

                <p class="price">₦5,000</p>

                <button class="add-to-cart" data-name="YCV4" data-price="5000">Add to Cart</button>

            </div>

            <div class="product" data-name="YCV5">

                <img src="YCV5.jpg" alt="Product 6">

                <p class="price">₦280,000</p>

                <button class="add-to-cart" data-name="YCV5" data-price="280000">Add to Cart</button>

            </div>

            <div class="product" data-name="YCV6">

                <img src="YCV6.webp" alt="Product 7">

                <p class="price">₦350,000</p>

                <button class="add-to-cart" data-name="YCV6" data-price="350000">Add to Cart</button>

            </div>

            <div class="product" data-name="YCV7">

                <img src="YCV7.jpg" alt="Product 8">

                <p class="price">₦450,000</p>

                <button class="add-to-cart" data-name="YCV7" data-price="450000">Add to Cart</button>

            </div>

            <div class="product" data-name="YCV8">

                <img src="YCV8.jpg" alt="Product 9">

                <p class="price">₦120,000</p>

                <button class="add-to-cart" data-name="YCV8" data-price="120000">Add to Cart</button>

            </div>

            <div class="product" data-name="YCV9">

                <img src="YCV9.jpg" alt="Product 10">

                <p class="price">₦56,000</p>

                <button class="add-to-cart" data-name="YCV9" data-price="56000">Add to Cart</button>

            </div>

            <div class="product" data-name="YCV10">

                <img src="YCV10.webp" alt="Product 11">

                <p class="price">₦40,000</p>

                <button class="add-to-cart" data-name="YCV10" data-price="40000">Add to Cart</button>

            </div>

            <div class="product" data-name="YCV11">

                <img src="YCV11.jfif" alt="Product 12">

                <p class="price">₦130,000</p>

                <button class="add-to-cart" data-name="YCV11" data-price="130000">Add to Cart</button>

            </div>

            <div class="product" data-name="YCV12">

                <img src="YCV12.webp" alt="Product 13">

                <p class="price">₦300,000</p>

                <button class="add-to-cart" data-name="YCV12" data-price="300000">Add to Cart</button>

            </div>

            <div class="product" data-name="YCV13">

                <img src="YCV13.png" alt="Product 14">

                <p class="price">₦235,000</p>

                <button class="add-to-cart" data-name="YCV13" data-price="235000">Add to Cart</button>

            </div>

            <div class="product" data-name="YCV14">

                <img src="YCV14.jfif" alt="Product 15">

                <p class="price">₦900,000</p>

                <button class="add-to-cart" data-name="YCV14" data-price="900000">Add to Cart</button>

            </div>

            <div class="product" data-name="YCV15">

                <img src="YCV15.jpg" alt="Product 16">

                <p class="price">₦11,000</p>

                <button class="add-to-cart" data-name="YCV15" data-price="11000">Add to Cart</button>

            </div>

            <div class="product" data-name="YCV16">

                <img src="YCV16.webp" alt="Product 17">

                <p class="price">₦420,000</p>

                <button class="add-to-cart" data-name="YCV16" data-price="420000">Add to Cart</button>

            </div>

            <div class="product" data-name="YCV17">

                <img src="YCV17.jpg" alt="Product 18">

                <p class="price">₦5,500,000</p>

                <button class="add-to-cart" data-name="YCV17" data-price="5500000">Add to Cart</button>

            </div>

            <div class="product" data-name="YCV18">

                <img src="YCV18.jpg" alt="Product 19">

                <p class="price">₦8,000</p>

                <button class="add-to-cart" data-name="YCV18" data-price="8000">Add to Cart</button>

            </div>

            <div class="product" data-name="YCV19">

                <img src="YCV19.jpg" alt="Product 20">

                <p class="price">₦134,000</p>

                <button class="add-to-cart" data-name="YCV19" data-price="134000">Add to Cart</button>

            </div>

        </section>

        <aside id="cart-section" style="display: none;">

            <h2>MY ITEMS</h2>

            <ul id="cart-items"></ul>

            <button id="clear-cart">Clear Cart</button>

            <p>Total: ₦<span id="total-price">0</span></p>

            <button id="place-order">Place Order</button>

        </aside>

        <footer>

            <div class="footer-container">

                <div class="footer-left">

                    <p>Address: Along Keffi-Abuja Express way Sabon Gari</p>

                </div>

                <div class="footer-middle">

                    <p>Tel: +2348062470009, +2347046555306</p>

                </div>

                <div class="footer-right">

                    <p>Email: <a href="mailto:yellowycv55@gmail.com">yellowycv55@gmail.com</a>, <a href="mailto:yellowycv551@yahoo.com">yellowycv551@yahoo.com</a></p>

                </div>

            </div>

            <p class="footer-bottom">All rights reserved @ 2024</p>

        </footer>

    </main>

    <script src="script.js"></script>

</body>

</html>

THE JAVASCRIPT TO MAKE THE HTML CODE FUNCTION ABOVE

document.addEventListener("DOMContentLoaded", () => {

    // Slideshow logic

    let slideshowImages = document.querySelectorAll(".slideshow-image");

    let currentImageIndex = 0;

    function showNextImage() {

        slideshowImages[currentImageIndex].classList.remove("active");

        currentImageIndex = (currentImageIndex + 1) % slideshowImages.length;

        slideshowImages[currentImageIndex].classList.add("active");

    }

    setInterval(showNextImage, 5000); // Change image every 5 seconds

    // Cart logic

    let cartItems = [];

    let cartList = document.getElementById("cart-items");

    let totalPriceElement = document.getElementById("total-price");

    let cartSection = document.getElementById("cart-section");

    document.querySelectorAll(".add-to-cart").forEach(button => {

        button.addEventListener("click", (e) => {

            let name = e.target.getAttribute("data-name");

            let price = parseInt(e.target.getAttribute("data-price"));

            let existingItem = cartItems.find(item => item.name === name);

            if (existingItem) {

                existingItem.quantity++;

            } else {

                cartItems.push({ name, price, quantity: 1 });

            }

            updateCart();

        });

    });

    document.getElementById("clear-cart").addEventListener("click", () => {

        cartItems.length = 0;

        updateCart();

    });

    function updateCart() {

        cartList.innerHTML = "";

        let totalPrice = 0;

        cartItems.forEach(item => {

            let li = document.createElement("li");

            li.textContent = `${item.name} - ₦${item.price} x ${item.quantity}`;

            let removeButton = document.createElement("button");

            removeButton.textContent = "Remove";

            removeButton.addEventListener("click", () => {

                let itemIndex = cartItems.indexOf(item);

                cartItems.splice(itemIndex, 1);

                updateCart();

            });

            li.appendChild(removeButton);

            cartList.appendChild(li);

            totalPrice += item.price \* item.quantity;

        });

        totalPriceElement.textContent = totalPrice;

        // Show or hide the cart section

        if (cartItems.length > 0) {

            cartSection.style.display = "block";

        } else {

            cartSection.style.display = "none";

        }

    }

    document.getElementById("place-order").addEventListener("click", () => {

        // Add order placing logic here

        alert("Order placed successfully!");

    });

    // Search functionality

    let searchInput = document.getElementById("search");

    let productsSection = document.getElementById("products-section");

    let products = [

        { name: "Android Device", price: 250000, img: "phone.jpg" },

        { name: "YCV1", price: 200000, img: "YCV1.jpg" },

        { name: "YCV2", price: 150000, img: "YCV2.jpg" },

        { name: "YCV3", price: 18800000, img: "YCV3.jpg" },

        { name: "YCV4", price: 5000, img: "YCV4.jpg" },

        { name: "YCV5", price: 280000, img: "YCV5.jpg" },

        { name: "YCV6", price: 350000, img: "YCV6.webp" },

        { name: "YCV7", price: 450000, img: "YCV7.jpg" },

        { name: "YCV8", price: 120000, img: "YCV8.jpg" },

        { name: "YCV9", price: 56000, img: "YCV9.jpg" },

        { name: "YCV10", price: 40000, img: "YCV10.webp" },

        { name: "YCV11", price: 130000, img: "YCV11.jfif" },

        { name: "YCV12", price: 300000, img: "YCV12.webp" },

        { name: "YCV13", price: 235000, img: "YCV13.png" },

        { name: "YCV14", price: 900000, img: "YCV14.jfif" },

        { name: "YCV15", price: 11000, img: "YCV15.jpg" },

        { name: "YCV16", price: 420000, img: "YCV16.webp" },

        { name: "YCV17", price: 5500000, img: "YCV17.jpg" },

        { name: "YCV18", price: 8000, img: "YCV18.jpg" },

        { name: "YCV19", price: 134000, img: "YCV19.jpg" }

    ];

    searchInput.addEventListener("input", () => {

        let query = searchInput.value.toLowerCase();

        let filteredProducts = products.filter(product => product.name.toLowerCase().includes(query));

        productsSection.innerHTML = filteredProducts.map(product => `

            <div class="product">

                <img src="${product.img}" alt="${product.name}">

                <p class="price">₦${product.price.toLocaleString()}</p>

                <button class="add-to-cart" data-name="${product.name}" data-price="${product.price}">Add to Cart</button>

            </div>

        `).join("");

        // Add event listeners for new Add to Cart buttons

        document.querySelectorAll(".add-to-cart").forEach(button => {

            button.addEventListener("click", (e) => {

                let name = e.target.getAttribute("data-name");

                let price = parseInt(e.target.getAttribute("data-price"));

                let existingItem = cartItems.find(item => item.name === name);

                if (existingItem) {

                    existingItem.quantity++;

                } else {

                    cartItems.push({ name, price, quantity: 1 });

                }

                updateCart();

            });

        });

    });

});

THE FUNCTION OF EACH SECTION OF THE JAVASCRIPT ABOVE

document.addEventListener("DOMContentLoaded", () => {

let slideshowImages = document.querySelectorAll(".slideshow-image");

let currentImageIndex = 0;

function showNextImage() {

slideshowImages[currentImageIndex].classList.remove("active");

currentImageIndex = (currentImageIndex + 1) % slideshowImages.length;

slideshowImages[currentImageIndex].classList.add("active");

}

setInterval(showNextImage, 5000); // Change image every 5 seconds

 **document.addEventListener("DOMContentLoaded", () => { ... });**: This ensures the code inside runs only after the HTML content has fully loaded.

 **let slideshowImages = document.querySelectorAll(".slideshow-image");**: This selects all elements with the class slideshow-image.

 **let currentImageIndex = 0;**: This keeps track of the current image index.

 **function showNextImage() { ... }**: This function changes the image by removing the active class from the current image and adding it to the next one.

 **setInterval(showNextImage, 5000);**: This calls the showNextImage function every 5 seconds to change the image.

**Cart Logic**

This section handles adding items to the cart, updating the cart, and clearing the cart

let cartItems = [];

let cartList = document.getElementById("cart-items");

let totalPriceElement = document.getElementById("total-price");

let cartSection = document.getElementById("cart-section");

document.querySelectorAll(".add-to-cart").forEach(button => {

button.addEventListener("click", (e) => {

let name = e.target.getAttribute("data-name");

let price = parseInt(e.target.getAttribute("data-price"));

let existingItem = cartItems.find(item => item.name === name);

if (existingItem) {

existingItem.quantity++;

} else {

cartItems.push({ name, price, quantity: 1 });

}

updateCart();

});

});

document.getElementById("clear-cart").addEventListener("click", () => {

cartItems.length = 0;

updateCart();

});

function updateCart() {

cartList.innerHTML = "";

let totalPrice = 0;

cartItems.forEach(item => {

let li = document.createElement("li");

li.textContent = `${item.name} - ₦${item.price} x ${item.quantity}`;

let removeButton = document.createElement("button");

removeButton.textContent = "Remove";

removeButton.addEventListener("click", () => {

let itemIndex = cartItems.indexOf(item);

cartItems.splice(itemIndex, 1);

updateCart();

});

li.appendChild(removeButton);

cartList.appendChild(li);

totalPrice += item.price \* item.quantity;

});

totalPriceElement.textContent = totalPrice;

if (cartItems.length > 0) {

cartSection.style.display = "block";

} else {

cartSection.style.display = "none";

}

}

document.getElementById("place-order").addEventListener("click", () => {

alert("Order placed successfully!");

});

* **let cartItems = [];**: This is an array to store items added to the cart.
* **let cartList = document.getElementById("cart-items");**: This selects the HTML element where cart items will be displayed.
* **let totalPriceElement = document.getElementById("total-price");**: This selects the element where the total price will be displayed.
* **let cartSection = document.getElementById("cart-section");**: This selects the cart section to show or hide it based on cart content.
* **document.querySelectorAll(".add-to-cart").forEach(button => { ... });**: This adds a click event listener to each "Add to Cart" button to add the item to the cart.
* **document.getElementById("clear-cart").addEventListener("click", () => { ... });**: This adds an event listener to the "Clear Cart" button to empty the cart.
* **function updateCart() { ... }**: This function updates the cart display and total price whenever items are added or removed.
* **document.getElementById("place-order").addEventListener("click", () => { ... });**: This adds an event listener to the "Place Order" button to show a success message.

**Search Functionality**

This part enables product search functionality, filtering the displayed products based on the search input.

let searchInput = document.getElementById("search");

let productsSection = document.getElementById("products-section");

let products = [

{ name: "Android Device", price: 250000, img: "phone.jpg" },

// ... (other products)

{ name: "YCV19", price: 134000, img: "YCV19.jpg" }

];

searchInput.addEventListener("input", () => {

let query = searchInput.value.toLowerCase();

let filteredProducts = products.filter(product => product.name.toLowerCase().includes(query));

productsSection.innerHTML = filteredProducts.map(product => `

<div class="product">

<img src="${product.img}" alt="${product.name}">

<p class="price">₦${product.price.toLocaleString()}</p>

<button class="add-to-cart" data-name="${product.name}" data-price="${product.price}">Add to Cart</button>

</div>

`).join("");

document.querySelectorAll(".add-to-cart").forEach(button => {

button.addEventListener("click", (e) => {

let name = e.target.getAttribute("data-name");

let price = parseInt(e.target.getAttribute("data-price"));

let existingItem = cartItems.find(item => item.name === name);

if (existingItem) {

existingItem.quantity++;

} else {

cartItems.push({ name, price, quantity: 1 });

}

updateCart();

});

});

});

});

* **let searchInput = document.getElementById("search");**: This selects the search input field.
* **let productsSection = document.getElementById("products-section");**: This selects the section where products are displayed.
* **let products = [ ... ];**: This is an array of product objects, each with a name, price, and image.
* **searchInput.addEventListener("input", () => { ... });**: This adds an event listener to the search input field to filter products as the user types.
* **let filteredProducts = products.filter(product => product.name.toLowerCase().includes(query));**: This filters the products based on the search query.
* **productsSection.innerHTML = filteredProducts.map(product => { ... }).join("");**: This updates the displayed products based on the filtered results.
* **document.querySelectorAll(".add-to-cart").forEach(button => { ... });**: This re-adds the "Add to Cart" button event listeners for the newly displayed products after filtering.

**Summary**

* **Slideshow Logic**: Automatically changes images every 5 seconds.
* **Cart Logic**: Adds items to the cart, updates the cart display, and handles clearing the cart and placing an order.
* **Search Functionality**: Filters products based on user input and updates the displayed products accordingly.

Feel free to ask if you have any questions or need further clarification on any part!

WHAT IS SET INTERVAL AND EVENT LINSTENER

### setInterval

**setInterval** is a function in JavaScript that allows you to run a piece of code repeatedly at a specified interval (in milliseconds).

#### How setInterval Works:

* **Syntax**: setInterval(function, milliseconds);
* **Parameters**:
  + function: The code you want to execute repeatedly.
  + milliseconds: The time delay between each execution in milliseconds (1000 milliseconds = 1 second).

#### Example:

function sayHello() {

console.log("Hello!");

}

setInterval(sayHello, 2000); // Runs the sayHello function every 2 seconds

In this example:

* The sayHello function prints "Hello!" to the console.
* setInterval(sayHello, 2000); calls the sayHello function every 2 seconds (2000 milliseconds).

### addEventListener

**addEventListener** is a method in JavaScript used to attach an event handler to an element. An event handler is a function that runs when a specific event occurs, like a button click or a key press.

#### How addEventListener Works:

* **Syntax**: element.addEventListener(event, function);
* **Parameters**:
  + event: The type of event you want to listen for (e.g., "click", "input").
  + function: The code you want to run when the event occurs.

#### Example:

let button = document.getElementById("myButton");

button.addEventListener("click", function() {

alert("Button was clicked!");

});

In this example:

* let button = document.getElementById("myButton"); selects the HTML element with the ID "myButton".
* button.addEventListener("click", function() { ... }); attaches a "click" event listener to the button. This means when the button is clicked, the code inside the function (which shows an alert saying "Button was clicked!") runs.

### Breaking Down the Provided Code with setInterval and addEventListener

#### setInterval in the Code:

setInterval(showNextImage, 5000); // Change image every 5 seconds

* **What it does**: Calls the showNextImage function every 5 seconds.
* **Purpose**: To automatically change the slideshow image at regular intervals.

#### addEventListener in the Code:

document.querySelectorAll(".add-to-cart").forEach(button => {

button.addEventListener("click", (e) => {

let name = e.target.getAttribute("data-name");

let price = parseInt(e.target.getAttribute("data-price"));

let existingItem = cartItems.find(item => item.name === name);

if (existingItem) {

existingItem.quantity++;

} else {

cartItems.push({ name, price, quantity: 1 });

}

updateCart();

});

});

 **What it does**: Attaches a "click" event listener to each element with the class add-to-cart.

 **Purpose**: To add an item to the cart when an "Add to Cart" button is clicked. The function inside the event listener retrieves the item’s name and price, checks if the item is already in the cart, updates the quantity if it is, or adds it as a new item if it isn’t, and then updates the cart display.

ASYNCHRONOUS FUNCTION

Asynchronous functions in JavaScript allow you to write code that performs tasks like fetching data from a server without blocking the execution of other code. This makes your applications more responsive and efficient, especially when dealing with tasks that take time, such as network requests or reading files.

### Async/Await

**async** and **await** are keywords in JavaScript that make working with asynchronous code more readable and easier to write. They are built on top of promises and allow you to write asynchronous code in a synchronous style.

#### Basic Syntax

* **async**: Used to declare an asynchronous function.
* **await**: Used to pause the execution of an async function until a promise is resolved.

#### Example

Here's a simple example of using async and await with a fetch request:

async function fetchData() {

try {

let response = await fetch('https://jsonplaceholder.typicode.com/users');

let data = await response.json();

console.log(data);

} catch (error) {

console.error('Error fetching data:', error);

}

}

fetchData();

In this example:

* async function fetchData() declares an asynchronous function named fetchData.
* await fetch('https://jsonplaceholder.typicode.com/users') pauses the execution of the function until the fetch request is complete.
* await response.json() pauses the execution until the response is converted to JSON.
* console.log(data) logs the fetched data.
* The try and catch blocks handle any errors that might occur during the fetch operation.

### Why Use Async/Await?

* **Readability**: Async/await makes asynchronous code look more like synchronous code, making it easier to read and understand.
* **Error Handling**: You can use try and catch blocks to handle errors in a more straightforward way compared to .then() and .catch() with promises.

### Converting Promise-Based Code to Async/Await

Here’s a comparison of promise-based code and its async/await equivalent:

#### Promise-Based Code

fetch('https://jsonplaceholder.typicode.com/users')

.then(response => response.json())

.then(data => {

console.log(data);

})

.catch(error => {

console.error('Error fetching data:', error);

});

Async/Await Code

async function fetchData() {

try {

let response = await fetch('https://jsonplaceholder.typicode.com/users');

let data = await response.json();

console.log(data);

} catch (error) {

console.error('Error fetching data:', error);

}

}

fetchData();

### Advanced Example: Fetching and Displaying Data

Let's expand the previous example to fetch user data and display it in a list on the webpage.

HTML:

<ul id="user-list"></ul>

document.addEventListener("DOMContentLoaded", async () => {

try {

let response = await fetch('https://jsonplaceholder.typicode.com/users');

let users = await response.json();

const userList = document.getElementById('user-list');

users.forEach(user => {

const li = document.createElement('li');

li.textContent = user.name;

userList.appendChild(li);

});

} catch (error) {

console.error('Error fetching user data:', error);

}

});

In this example:

* The DOMContentLoaded event ensures the script runs after the DOM is fully loaded.
* The async function fetches user data and processes it to update the DOM.

FETCH API

The Fetch API in JavaScript is a modern interface for making HTTP requests to servers, such as fetching data from an API or submitting data to a server. It is a simpler and more powerful alternative to the older XMLHttpRequest.

### Basic Concepts of Fetch API

 **Making a Request**: You use the fetch function to make a request to a URL.

 **Handling Responses**: Fetch returns a promise that resolves to a Response object, which contains the data from the server.

 **Promises**: Fetch uses promises, which allow you to handle asynchronous operations more cleanly.

### Basic Syntax

The basic syntax of the Fetch API is:

fetch(url, options)

.then(response => {

// handle the response

})

.catch(error => {

// handle the error

});

* **url**: The URL you want to fetch data from.
* **options** (optional): An object that contains settings for the request (like method, headers, body, etc.).

**Example 1: Simple GET Request**

Let's say you want to fetch data from a public API that returns user information.

fetch('https://jsonplaceholder.typicode.com/users')

.then(response => {

// Convert the response to JSON

return response.json();

})

.then(data => {

// Handle the data

console.log(data);

})

.catch(error => {

// Handle any errors

console.error('Error fetching data:', error);

});

In this example:

* fetch('https://jsonplaceholder.typicode.com/users') makes a GET request to the given URL.
* .then(response => response.json()) converts the response to JSON.
* .then(data => console.log(data)) logs the JSON data to the console.
* .catch(error => console.error('Error fetching data:', error)) handles any errors that occur during the fetch.

**Example 2: POST Request**

If you want to send data to the server, you can use the POST method with fetch.

fetch('https://jsonplaceholder.typicode.com/posts', {

method: 'POST',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify({

title: 'foo',

body: 'bar',

userId: 1

})

})

.then(response => response.json())

.then(data => {

console.log('Success:', data);

})

.catch(error => {

console.error('Error:', error);

});

In this example:

* method: 'POST' specifies that the request is a POST request.
* headers: { 'Content-Type': 'application/json' } sets the request headers, indicating that the body of the request is JSON.
* body: JSON.stringify({ ... }) converts the data to a JSON string before sending it.
* The rest of the code handles the response and any errors in the same way as the GET request example.

**Summary**

* **Fetch API**: A modern way to make HTTP requests in JavaScript.
* **Promises**: Fetch returns promises, making it easier to handle asynchronous operations.
* **GET Request**: Fetch data from a server.
* **POST Request**: Send data to a server.

**Advanced Example: Fetching Data and Updating the DOM**

Here's an example that fetches user data and displays it in a list on the web page.

<ul id="user-list"></ul>

document.addEventListener("DOMContentLoaded", () => {

fetch('https://jsonplaceholder.typicode.com/users')

.then(response => response.json())

.then(users => {

const userList = document.getElementById('user-list');

users.forEach(user => {

const li = document.createElement('li');

li.textContent = user.name;

userList.appendChild(li);

});

})

.catch(error => {

console.error('Error fetching user data:', error);

});

});

In this example:

* The fetch request gets user data from the API.
* The response is converted to JSON.
* The user data is used to create list items (li) and append them to the ul element with the ID user-list.

This covers the basics of using the Fetch API to make requests and handle responses. Let me know if you have any questions or need further explanation!