

**Digital Empowerment Pakistan**

**TASK#02**

**Submitted by:**

**MUBASHRA MOBEEN MALIK**

**WEB DEVELPOMENT INTERNSHIP**

**TASK:**

Develop a dynamic blog website using Node.js, Express, and MongoDB, allowing users to create, read, update, and delete blog posts.

**CODE:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Dynamic Blog</title>

    <style>

        /\* Reset and general styles \*/

        body {

            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

            margin: 0;

            padding: 0;

            background-color: #f0f0f0;

        }

        /\* Header styles \*/

        header {

            background-color: #333;

            color: #fff;

            text-align: center;

            padding: 1rem 0;

        }

        header h1 {

            margin: 0;

            font-size: 2.5rem;

        }

        /\* Main content area \*/

        main {

            max-width: 800px;

            margin: 20px auto;

            padding: 20px;

            background-color: #fff;

            border-radius: 8px;

            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

        }

        /\* Blog post section \*/

        #blog-posts {

            margin-bottom: 20px;

        }

        .blog-post {

            border-bottom: 1px solid #ccc;

            padding-bottom: 10px;

            margin-bottom: 20px;

        }

        .blog-post h2 {

            font-size: 1.5rem;

            margin-bottom: 5px;

        }

        .blog-post p {

            line-height: 1.6;

        }

        .blog-post button {

            padding: 8px 16px;

            background-color: #333;

            color: #fff;

            border: none;

            cursor: pointer;

            font-size: 0.9rem;

            margin-right: 10px;

            transition: background-color 0.3s ease;

        }

        .blog-post button:hover {

            background-color: #555;

        }

        /\* New post form \*/

        #post-form {

            margin-top: 20px;

        }

        form {

            max-width: 600px;

            margin: auto;

        }

        label {

            font-weight: bold;

            display: block;

            margin-bottom: 5px;

        }

        input[type="text"],

        textarea {

            width: 100%;

            padding: 8px;

            margin-bottom: 10px;

            border: 1px solid #ccc;

            border-radius: 4px;

            box-sizing: border-box;

            font-size: 1rem;

        }

        button[type="submit"],

        button[type="button"] {

            padding: 10px 20px;

            background-color: #333;

            color: #fff;

            border: none;

            cursor: pointer;

            font-size: 1rem;

            transition: background-color 0.3s ease;

        }

        button[type="submit"]:hover,

        button[type="button"]:hover {

            background-color: #555;

        }

        /\* Edit post form \*/

        #edit-post-form {

            display: none;

        }

    </style>

</head>

<body>

    <header>

        <h1>Dynamic Blog</h1>

    </header>

    <main>

        <section id="blog-posts">

            <!-- Blog posts will be dynamically added here -->

        </section>

        <section id="post-form">

            <h2>New Post</h2>

            <form id="new-post-form">

                <label for="title">Title:</label>

                <input type="text" id="title" name="title" required><br>

                <label for="content">Content:</label><br>

                <textarea id="content" name="content" rows="4" required></textarea><br>

                <button type="submit">Create Post</button>

            </form>

        </section>

        <section id="edit-post-form">

            <h2>Edit Post</h2>

            <form id="edit-form">

                <input type="hidden" id="edit-id" name="id">

                <label for="edit-title">Title:</label>

                <input type="text" id="edit-title" name="title" required><br>

                <label for="edit-content">Content:</label><br>

                <textarea id="edit-content" name="content" rows="4" required></textarea><br>

                <button type="submit">Update Post</button>

                <button type="button" onclick="cancelEdit()">Cancel</button>

            </form>

        </section>

    </main>

    <script>

        // Sample initial posts (for demonstration)

        let posts = [

            { id: 1, title: "First Post", content: "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla euismod purus eu libero volutpat, non viverra eros posuere." },

            { id: 2, title: "Second Post", content: "Proin sed lacus interdum, ultricies nunc id, venenatis arcu. Vestibulum maximus fermentum justo, sit amet tristique mi venenatis a." }

        ];

        const blogPostsSection = document.getElementById('blog-posts');

        const newPostForm = document.getElementById('new-post-form');

        const editPostForm = document.getElementById('edit-post-form');

        const editForm = document.getElementById('edit-form');

        const editTitleInput = document.getElementById('edit-title');

        const editContentInput = document.getElementById('edit-content');

        let editingPostId = null;

        function displayPosts() {

            blogPostsSection.innerHTML = '';

            posts.forEach(post => {

                const postElement = document.createElement('article');

                postElement.classList.add('blog-post');

                postElement.innerHTML = `

                    <h2>${post.title}</h2>

                    <p>${post.content}</p>

                    <button onclick="editPost(${post.id})">Edit</button>

                    <button onclick="deletePost(${post.id})">Delete</button>

                `;

                blogPostsSection.appendChild(postElement);

            });

        }

        function deletePost(postId) {

            posts = posts.filter(post => post.id !== postId);

            displayPosts();

        }

        function editPost(postId) {

            const postToEdit = posts.find(post => post.id === postId);

            if (postToEdit) {

                editingPostId = postToEdit.id;

                editTitleInput.value = postToEdit.title;

                editContentInput.value = postToEdit.content;

                newPostForm.style.display = 'none';

                editPostForm.style.display = 'block';

            }

        }

        function cancelEdit() {

            editingPostId = null;

            editForm.reset();

            newPostForm.style.display = 'block';

            editPostForm.style.display = 'none';

        }

        newPostForm.addEventListener('submit', function(event) {

            event.preventDefault();

            const title = event.target.title.value;

            const content = event.target.content.value;

            const newPost = { id: Date.now(), title, content };

            posts.push(newPost);

            displayPosts();

            event.target.reset();

        });

        editForm.addEventListener('submit', function(event) {

            event.preventDefault();

            const updatedTitle = editForm.title.value;

            const updatedContent = editForm.content.value;

            posts = posts.map(post => {

                if (post.id === editingPostId) {

                    return { ...post, title: updatedTitle, content: updatedContent };

                }

                return post;

            });

            displayPosts();

            cancelEdit();

        });

        // Initial display of posts

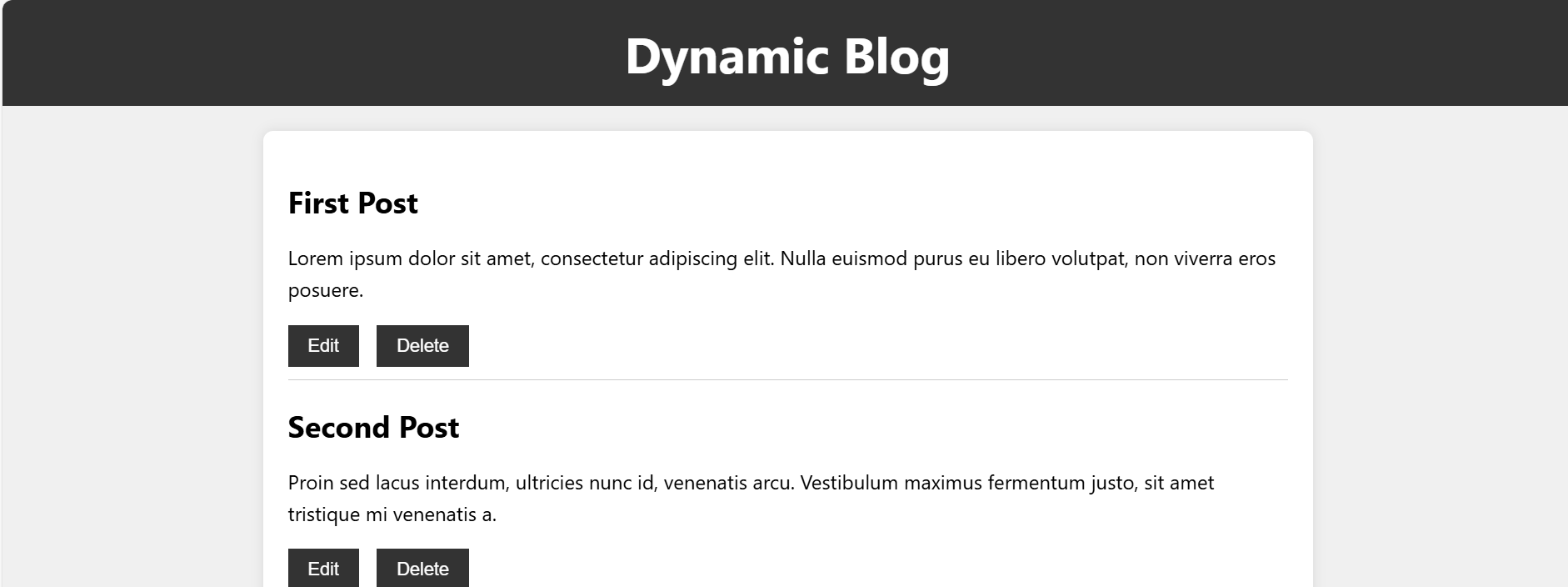
        displayPosts();

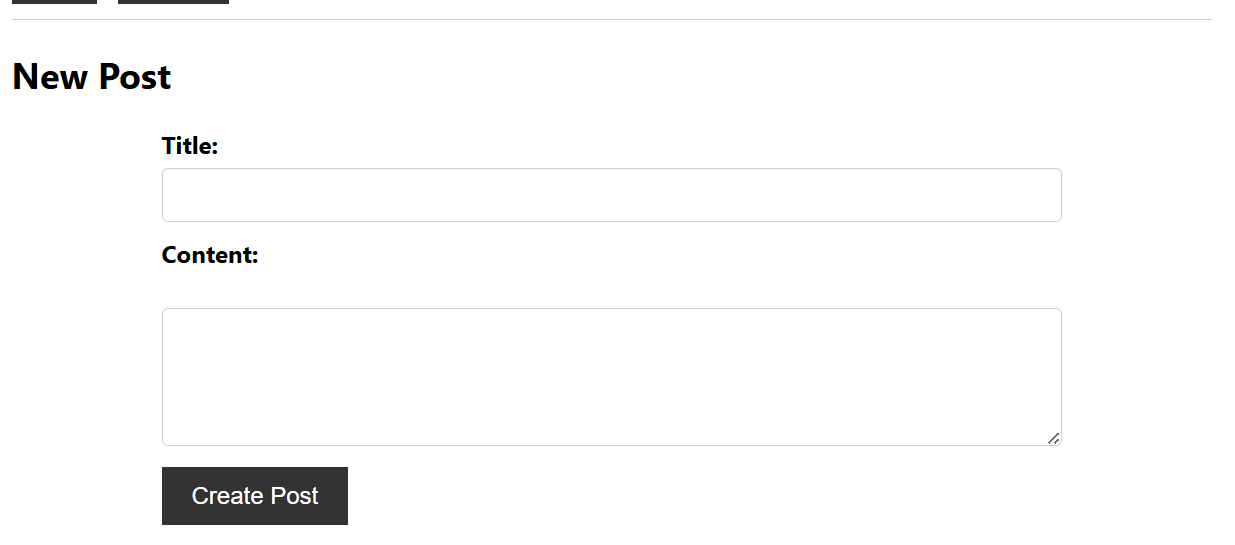
    </script>

</body>

</html>

**Output:**





**Explanation:**

HTML Structure: The structure remains the same as before, with <style> tags added in the <head> section to define internal CSS.

Internal CSS: All CSS styles are now included within <style> tags inside the <head> section. This approach keeps the styles scoped within the HTML file.

Added an edit form (#edit-post-form) that allows users to edit existing blog posts.

Updated the displayPosts() function to include "Edit" buttons for each post, which triggers the editPost(postId) function.

Added functionality for canceling the edit (cancelEdit() function) and updating the post (editForm.addEventListener('submit').

Create Post: Users can add new blog posts using the form in #new-post-form.

Read Posts: All existing posts are dynamically displayed under #blog-posts.

Update Post: Users can edit any post by clicking the "Edit" button, modifying the content in the edit form (#edit-post-form), and clicking "Update Post".

Delete Post: Each post has a "Delete" button that removes the post from the list.

**Conclusion:**

This setup provides a robust frontend solution for managing blog posts with CRUD operations. For a complete application, consider integrating backend services for data storage and user authentication.