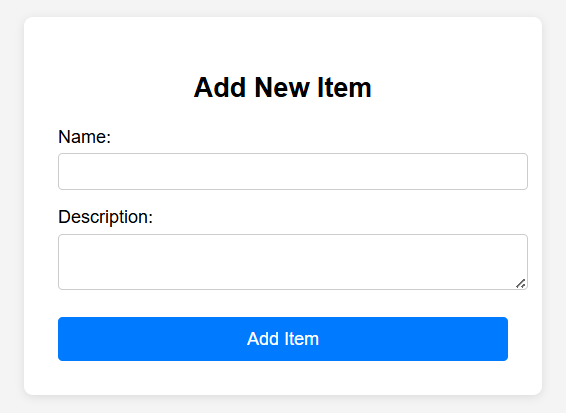
**Q1: Can you create a simple HTML interface to add new items with fields like name and description?**



<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Add New Item</title>

<style>

body { font-family: Arial, sans-serif; background: #f4f4f4; margin: 0; padding: 0; }

.container { max-width: 400px; margin: 50px auto; background: #fff; padding: 30px; border-radius: 8px; box-shadow: 0 2px 8px rgba(0,0,0,0.1); }

h2 { text-align: center; }

label { display: block; margin-top: 15px; }

input, textarea { width: 100%; padding: 8px; margin-top: 5px; border: 1px solid #ccc; border-radius: 4px; }

button { margin-top: 20px; width: 100%; padding: 10px; background: #007bff; color: #fff; border: none; border-radius: 4px; font-size: 16px; cursor: pointer; }

button:hover { background: #0056b3; }

</style>

</head>

<body>

<div class="container">

<h2>Add New Item</h2>

<form id="addItemForm">

<label for="name">Name:</label>

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

<label for="description">Description:</label>

<textarea id="description" name="description" rows="4" required></textarea>

<button type="submit">Add Item</button>

</form>

<div id="result" style="margin-top:20px; text-align:center;"></div>

</div>

<script>

document.getElementById('addItemForm').addEventListener('submit', function(e) {

e.preventDefault();

const name = document.getElementById('name').value;

const description = document.getElementById('description').value;

document.getElementById('result').textContent = `Item added: ${name} - ${description}`;

this.reset();

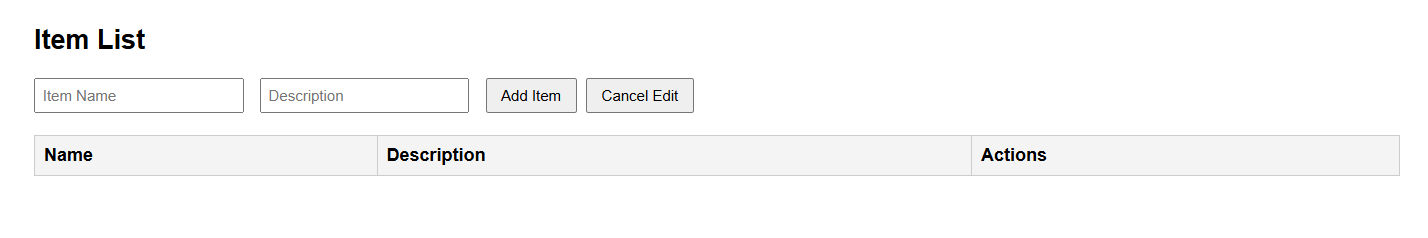
});

</script>

</body>

</html>

**Q2: Build an HTML table with a form to add new items and buttons to edit or delete each one using JavaScript.**

****

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Item Table with Add/Edit/Delete</title>

<style>

body { font-family: Arial, sans-serif; margin: 40px; }

table { border-collapse: collapse; width: 100%; margin-top: 20px; }

th, td { border: 1px solid #ccc; padding: 8px; text-align: left; }

th { background: #f4f4f4; }

form { margin-bottom: 20px; }

input[type="text"] { padding: 6px; margin-right: 10px; }

button { padding: 6px 12px; margin-right: 4px; }

</style>

</head>

<body>

<h2>Item List</h2>

<form id="itemForm">

<input type="text" id="itemName" placeholder="Item Name" required />

<input type="text" id="itemDesc" placeholder="Description" required />

<button type="submit" id="submitBtn">Add Item</button>

<button type="button" id="cancelEditBtn" style="display:none;">Cancel Edit</button>

</form>

<table id="itemTable">

<thead>

<tr>

<th>Name</th>

<th>Description</th>

<th>Actions</th>

</tr>

</thead>

<tbody>

<!-- Items will be inserted here -->

</tbody>

</table>

<script>

const itemForm = document.getElementById('itemForm');

const itemName = document.getElementById('itemName');

const itemDesc = document.getElementById('itemDesc');

const itemTable = document.getElementById('itemTable').querySelector('tbody');

const submitBtn = document.getElementById('submitBtn');

const cancelEditBtn = document.getElementById('cancelEditBtn');

let items = [];

let editIndex = null;

function renderTable() {

itemTable.innerHTML = '';

items.forEach((item, idx) => {

const row = document.createElement('tr');

row.innerHTML = `

<td>${item.name}</td>

<td>${item.desc}</td>

<td>

<button onclick="editItem(${idx})">Edit</button>

<button onclick="deleteItem(${idx})">Delete</button>

</td>

`;

itemTable.appendChild(row);

});

}

window.editItem = function(idx) {

editIndex = idx;

itemName.value = items[idx].name;

itemDesc.value = items[idx].desc;

submitBtn.textContent = 'Update Item';

cancelEditBtn.style.display = 'inline';

};

window.deleteItem = function(idx) {

if (confirm('Delete this item?')) {

items.splice(idx, 1);

renderTable();

if (editIndex === idx) {

resetForm();

}

}

};

function resetForm() {

editIndex = null;

itemForm.reset();

submitBtn.textContent = 'Add Item';

cancelEditBtn.style.display = 'none';

}

itemForm.onsubmit = function(e) {

e.preventDefault();

const name = itemName.value.trim();

const desc = itemDesc.value.trim();

if (!name || !desc) return;

if (editIndex !== null) {

items[editIndex] = { name, desc };

} else {

items.push({ name, desc });

}

renderTable();

resetForm();

};

cancelEditBtn.onclick = resetForm;

// Initial render

renderTable();

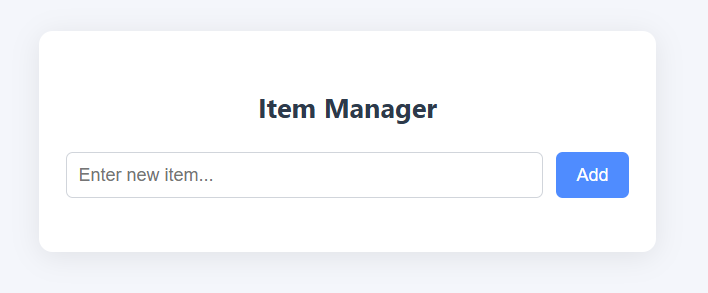
</script>

</body>

</html>

**Q3: Create a fully styled HTML and JavaScript interface for managing a list of items, including add, edit, and delete actions, with editable fields and confirmation prompts.**

**Información adicional: In only one file**



<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Item Manager</title>

<style>

body {

font-family: 'Segoe UI', Arial, sans-serif;

background: #f4f6fb;

margin: 0;

padding: 0;

min-height: 100vh;

}

.container {

max-width: 500px;

margin: 40px auto;

background: #fff;

border-radius: 12px;

box-shadow: 0 4px 24px rgba(0,0,0,0.08);

padding: 32px 24px 24px 24px;

}

h2 {

text-align: center;

color: #2d3a4b;

margin-bottom: 24px;

}

.item-form {

display: flex;

gap: 12px;

margin-bottom: 24px;

}

.item-form input[type="text"] {

flex: 1;

padding: 10px;

border: 1px solid #d1d5db;

border-radius: 6px;

font-size: 1rem;

outline: none;

transition: border 0.2s;

}

.item-form input[type="text"]:focus {

border: 1.5px solid #4f8cff;

}

.item-form button {

background: #4f8cff;

color: #fff;

border: none;

border-radius: 6px;

padding: 10px 18px;

font-size: 1rem;

cursor: pointer;

transition: background 0.2s;

}

.item-form button:hover {

background: #2563eb;

}

ul.item-list {

list-style: none;

padding: 0;

margin: 0;

}

.item {

display: flex;

align-items: center;

justify-content: space-between;

background: #f7f9fc;

border-radius: 6px;

padding: 10px 14px;

margin-bottom: 10px;

transition: box-shadow 0.2s;

}

.item .item-text {

flex: 1;

font-size: 1rem;

color: #2d3a4b;

word-break: break-word;

}

.item input[type="text"] {

flex: 1;

font-size: 1rem;

padding: 6px 8px;

border: 1px solid #d1d5db;

border-radius: 4px;

margin-right: 8px;

}

.item .actions {

display: flex;

gap: 6px;

}

.item .actions button {

background: none;

border: none;

color: #4f8cff;

font-size: 1rem;

padding: 4px 8px;

border-radius: 4px;

cursor: pointer;

transition: background 0.2s, color 0.2s;

}

.item .actions button:hover {

background: #e0e7ff;

color: #2563eb;

}

.item .actions .delete-btn {

color: #ef4444;

}

.item .actions .delete-btn:hover {

background: #fee2e2;

color: #b91c1c;

}

@media (max-width: 600px) {

.container {

padding: 16px 4px 12px 4px;

}

.item-form button {

padding: 10px 8px;

}

}

</style>

</head>

<body>

<div class="container">

<h2>Item Manager</h2>

<form class="item-form" id="itemForm">

<input type="text" id="itemInput" placeholder="Enter new item..." required />

<button type="submit">Add</button>

</form>

<ul class="item-list" id="itemList"></ul>

</div>

<script>

const itemForm = document.getElementById('itemForm');

const itemInput = document.getElementById('itemInput');

const itemList = document.getElementById('itemList');

let items = [];

let editIndex = null;

function renderItems() {

itemList.innerHTML = '';

items.forEach((item, idx) => {

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

li.className = 'item';

if (editIndex === idx) {

// Edit mode

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

input.type = 'text';

input.value = item;

input.autofocus = true;

input.addEventListener('keydown', (e) => {

if (e.key === 'Enter') saveEdit(idx, input.value);

if (e.key === 'Escape') cancelEdit();

});

const actions = document.createElement('div');

actions.className = 'actions';

const saveBtn = document.createElement('button');

saveBtn.textContent = 'Save';

saveBtn.onclick = () => saveEdit(idx, input.value);

const cancelBtn = document.createElement('button');

cancelBtn.textContent = 'Cancel';

cancelBtn.onclick = cancelEdit;

actions.appendChild(saveBtn);

actions.appendChild(cancelBtn);

li.appendChild(input);

li.appendChild(actions);

} else {

// View mode

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

span.className = 'item-text';

span.textContent = item;

const actions = document.createElement('div');

actions.className = 'actions';

const editBtn = document.createElement('button');

editBtn.textContent = 'Edit';

editBtn.onclick = () => startEdit(idx);

const deleteBtn = document.createElement('button');

deleteBtn.textContent = 'Delete';

deleteBtn.className = 'delete-btn';

deleteBtn.onclick = () => confirmDelete(idx);

actions.appendChild(editBtn);

actions.appendChild(deleteBtn);

li.appendChild(span);

li.appendChild(actions);

}

itemList.appendChild(li);

});

}

function addItem(text) {

items.push(text);

renderItems();

}

function startEdit(idx) {

editIndex = idx;

renderItems();

// Focus the input after rendering

setTimeout(() => {

const input = itemList.querySelector('input[type="text"]');

if (input) input.focus();

}, 0);

}

function saveEdit(idx, newText) {

if (newText.trim() === '') {

alert('Item cannot be empty.');

return;

}

items[idx] = newText;

editIndex = null;

renderItems();

}

function cancelEdit() {

editIndex = null;

renderItems();

}

function confirmDelete(idx) {

if (confirm('Are you sure you want to delete this item?')) {

items.splice(idx, 1);

if (editIndex === idx) editIndex = null;

renderItems();

}

}

itemForm.addEventListener('submit', function(e) {

e.preventDefault();

const value = itemInput.value.trim();

if (value) {

addItem(value);

itemInput.value = '';

itemInput.focus();

}

});

// Initial render

renderItems();

</script>

</body>

</html>