**Exercise 1: Changing the Text of an Element**

**Task:**  
Create an HTML page with a paragraph. Using JavaScript, change the text of the paragraph to "Hello, World!" when a button is clicked.

**HTML:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Change Text</title>

</head>

<body>

<p id="text">Original Text</p>

<button id="changeTextBtn">Change Text</button>

<script src="dom-exercise1.js"></script>

</body>

</html>

**JavaScript (dom-exercise1.js):**

document.getElementById('changeTextBtn').addEventListener('click', function() {

document.getElementById('text').textContent = 'Hello, World!';

});

**Exercise 2: Changing the Style of an Element**

**Task:**  
Create a button that changes the background color of a div when clicked.

**HTML:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Change Color</title>

<style>

#box {

width: 200px;

height: 200px;

background-color: lightblue;

}

</style>

</head>

<body>

<div id="box"></div>

<button id="changeColorBtn">Change Color</button>

<script src="dom-exercise2.js"></script>

</body>

</html>

**JavaScript (dom-exercise2.js):**

document.getElementById('changeColorBtn').addEventListener('click', function() {

document.getElementById('box').style.backgroundColor = 'lightgreen';

});

**Exercise 3: Adding New Elements**

**Task:**  
Create a form where users can input a task. When the user submits the form, the task should be added as a list item to an unordered list.

**HTML:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Todo List</title>

</head>

<body>

<form id="taskForm">

<input type="text" id="taskInput" placeholder="Enter a task" required>

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

</form>

<ul id="taskList"></ul>

<script src="dom-exercise3.js"></script>

</body>

</html>

**JavaScript (dom-exercise3.js):**

document.getElementById('taskForm').addEventListener('submit', function(event) {

event.preventDefault(); // Prevent form submission

const taskText = document.getElementById('taskInput').value;

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

newTask.textContent = taskText;

document.getElementById('taskList').appendChild(newTask);

document.getElementById('taskForm').reset(); // Clear the input field

});

**Exercise 4: Removing Elements from the DOM**

**Task:**  
Add a "Delete" button next to each task in the task list from Exercise 3. When clicked, the corresponding task should be removed from the DOM.

**Updated HTML:**

<!-- Use the same HTML structure as Exercise 3 -->

**JavaScript (dom-exercise4.js):**

document.getElementById('taskForm').addEventListener('submit', function(event) {

event.preventDefault();

const taskText = document.getElementById('taskInput').value;

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

newTask.textContent = taskText;

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

deleteBtn.textContent = 'Delete';

deleteBtn.addEventListener('click', function() {

newTask.remove(); // Remove the task when delete button is clicked

});

newTask.appendChild(deleteBtn);

document.getElementById('taskList').appendChild(newTask);

document.getElementById('taskForm').reset();

});

**Exercise 5: Displaying a Dynamic List**

**Task:**  
Create a list of items using an array. When the page loads, dynamically populate the list with the array elements.

**HTML:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Dynamic List</title>

</head>

<body>

<ul id="itemList"></ul>

<script src="dom-exercise5.js"></script>

</body>

</html>

**JavaScript (dom-exercise5.js):**

const items = ['Apple', 'Banana', 'Orange', 'Grapes', 'Mango'];

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

items.forEach(item => {

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

listItem.textContent = item;

itemList.appendChild(listItem);

});

**Exercise 6: Todo List with Edit and Delete Functionality**

**Task:**  
Expand the todo list application to include editing tasks. Each task should have "Edit" and "Delete" buttons. Clicking "Edit" should allow the user to update the task, and clicking "Delete" should remove it.

**HTML:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Advanced Todo List</title>

</head>

<body>

<form id="taskForm">

<input type="text" id="taskInput" placeholder="Enter a task" required>

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

</form>

<ul id="taskList"></ul>

<script src="dom-exercise6.js"></script>

</body>

</html>

**JavaScript (dom-exercise6.js):**

document.getElementById('taskForm').addEventListener('submit', function(event) {

event.preventDefault();

const taskText = document.getElementById('taskInput').value;

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

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

taskContent.textContent = taskText;

// Create Edit Button

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

editBtn.textContent = 'Edit';

editBtn.addEventListener('click', function() {

const newTaskText = prompt('Edit your task', taskContent.textContent);

if (newTaskText) {

taskContent.textContent = newTaskText;

}

});

// Create Delete Button

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

deleteBtn.textContent = 'Delete';

deleteBtn.addEventListener('click', function() {

newTask.remove();

});

newTask.appendChild(taskContent);

newTask.appendChild(editBtn);

newTask.appendChild(deleteBtn);

document.getElementById('taskList').appendChild(newTask);

document.getElementById('taskForm').reset();

});

**Exercise 7: Shopping Cart System**

**Task:**  
Create a simple shopping cart system. Display a list of products with "Add to Cart" buttons. When a product is added, it should be displayed in a shopping cart with its name, price, and quantity. The user should be able to remove products and update the quantity.

**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</title>

</head>

<body>

<h1>Products</h1>

<div id="products"></div>

<h2>Shopping Cart</h2>

<ul id="cart"></ul>

<script src="dom-exercise7.js"></script>

</body>

</html>

**JavaScript (dom-exercise7.js):**

const products = [

{ id: 1, name: 'Product 1', price: 100 },

{ id: 2, name: 'Product 2', price: 200 },

{ id: 3, name: 'Product 3', price: 300 }

];

const cart = [];

const productContainer = document.getElementById('products');

const cartContainer = document.getElementById('cart');

// Render Products

products.forEach(product => {

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

productDiv.textContent = `${product.name} - $${product.price}`;

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

addButton.textContent = 'Add to Cart';

addButton.addEventListener('click', function() {

addToCart(product);

});

productDiv.appendChild(addButton);

productContainer.appendChild(productDiv);

});

// Function to Add Product to Cart

function addToCart(product) {

const cartItem = cart.find(item => item.id === product.id);

if (cartItem) {

cartItem.quantity++;

} else {

cart.push({ ...product, quantity: 1 });

}

renderCart();

}

// Function to Render Cart

function renderCart() {

cartContainer.innerHTML = ''; // Clear existing cart

cart.forEach(item => {

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

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

// Remove Button

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

removeButton.textContent = 'Remove';

removeButton.addEventListener('click', function() {

removeFromCart(item.id);

});

// Update Quantity Button

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

updateQuantity.textContent = 'Update Quantity';

updateQuantity.addEventListener('click', function() {

const newQuantity = prompt('Enter new quantity:', item.quantity);

if (newQuantity) {

item.quantity = parseInt(newQuantity, 10);

renderCart();

}

});

cartItem.appendChild(removeButton);

cartItem.appendChild(updateQuantity);

cartContainer.appendChild(cartItem);

});

}

// Function to Remove Product from Cart

function removeFromCart(productId) {

const index = cart.findIndex(item => item.id === productId);

if (index !== -1) {

cart.splice(index, 1);

}

renderCart();

}

**Exercise 8: Dynamic Form Generation**

**Task:**  
Create a dynamic form generation system where users can specify the number of form fields they need (e.g., text inputs). Upon submission, display the form with the specified number of fields.

**HTML:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Dynamic Form</title>

</head>

<body>

<h1>Create Form</h1>

<label for="numFields">Number of Fields:</label>

<input type="number" id="numFields" min="1" max="10" required>

<button id="generateFormBtn">Generate Form</button>

<form id="dynamicForm"></form>

<script src="dom-exercise8.js"></script>

</body>

</html>

**JavaScript (dom-exercise8.js):**

document.getElementById('generateFormBtn').addEventListener('click', function() {

const numFields = parseInt(document.getElementById('numFields').value, 10);

const form = document.getElementById('dynamicForm');

form.innerHTML = ''; // Clear the existing form

for (let i = 1; i <= numFields; i++) {

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

label.textContent = `Field ${i}:`;

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

input.type = 'text';

input.name = `field${i}`;

input.placeholder = `Enter value for field ${i}`;

form.appendChild(label);

form.appendChild(input);

form.appendChild(document.createElement('br')); // Line break for better layout

}

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

submitBtn.textContent = 'Submit';

submitBtn.type = 'submit';

form.appendChild(submitBtn);

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

event.preventDefault(); // Prevent page reload on form submission

const formData = new FormData(form);

for (let [name, value] of formData.entries()) {

console.log(`${name}: ${value}`);

}

});

});

**Exercise 9: Interactive Image Gallery with Modal View**

**Task:**  
Create an interactive image gallery where users can click on a thumbnail to view a larger version in a modal (popup) window. The modal should also have "Next" and "Previous" buttons to navigate between images.

**HTML:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Image Gallery</title>

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

</head>

<body>

<h1>Image Gallery</h1>

<div class="gallery">

<img src="image1.jpg" class="thumbnail" alt="Image 1">

<img src="image2.jpg" class="thumbnail" alt="Image 2">

<img src="image3.jpg" class="thumbnail" alt="Image 3">

<img src="image4.jpg" class="thumbnail" alt="Image 4">

</div>

<div id="modal" class="modal">

<span id="closeBtn" class="close">&times;</span>

<img class="modal-content" id="modalImg">

<div id="caption"></div>

<button id="prevBtn" class="nav-btn">Previous</button>

<button id="nextBtn" class="nav-btn">Next</button>

</div>

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

</body>

</html>

**CSS (styles.css):**

body {

font-family: Arial, sans-serif;

}

.gallery {

display: flex;

flex-wrap: wrap;

gap: 10px;

}

.thumbnail {

width: 200px;

height: 150px;

object-fit: cover;

cursor: pointer;

transition: transform 0.2s;

}

.thumbnail:hover {

transform: scale(1.1);

}

.modal {

display: none;

position: fixed;

z-index: 1;

padding-top: 50px;

left: 0;

top: 0;

width: 100%;

height: 100%;

background-color: rgba(0,0,0,0.9);

}

.modal-content {

margin: auto;

display: block;

max-width: 80%;

}

#caption {

margin: auto;

display: block;

text-align: center;

color: #ccc;

padding: 10px;

font-size: 18px;

}

.close {

position: absolute;

top: 15px;

right: 35px;

color: white;

font-size: 40px;

font-weight: bold;

cursor: pointer;

}

.close:hover {

color: #bbb;

}

.nav-btn {

position: absolute;

top: 50%;

color: white;

font-size: 24px;

background-color: rgba(0, 0, 0, 0.5);

border: none;

padding: 10px;

cursor: pointer;

}

#prevBtn {

left: 10px;

}

#nextBtn {

right: 10px;

}

**JavaScript (gallery.js):**

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

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

const captionText = document.getElementById('caption');

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

const thumbnails = document.querySelectorAll('.thumbnail');

let currentIndex = 0;

thumbnails.forEach((thumbnail, index) => {

thumbnail.addEventListener('click', function() {

modal.style.display = 'block';

modalImg.src = this.src;

captionText.textContent = this.alt;

currentIndex = index;

});

});

closeBtn.addEventListener('click', function() {

modal.style.display = 'none';

});

document.getElementById('prevBtn').addEventListener('click', function() {

currentIndex = (currentIndex === 0) ? thumbnails.length - 1 : currentIndex - 1;

showImage();

});

document.getElementById('nextBtn').addEventListener('click', function() {

currentIndex = (currentIndex === thumbnails.length - 1) ? 0 : currentIndex + 1;

showImage();

});

function showImage() {

modalImg.src = thumbnails[currentIndex].src;

captionText.textContent = thumbnails[currentIndex].alt;

}

**Exercise 10: Drag and Drop Task Manager**

**Task:**  
Create a task manager where users can drag tasks between "To Do", "In Progress", and "Completed" sections. The task items should be draggable, and the sections should be droppable.

**HTML:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Task Manager</title>

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

</head>

<body>

<h1>Task Manager</h1>

<div class="columns">

<div class="column" id="todo">

<h2>To Do</h2>

<div class="task" draggable="true">Task 1</div>

<div class="task" draggable="true">Task 2</div>

</div>

<div class="column" id="inProgress">

<h2>In Progress</h2>

</div>

<div class="column" id="completed">

<h2>Completed</h2>

</div>

</div>

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

</body>

</html>

**CSS (taskmanager.css):**

body {

font-family: Arial, sans-serif;

}

.columns {

display: flex;

justify-content: space-around;

margin: 20px;

}

.column {

width: 30%;

padding: 10px;

border: 1px solid #ccc;

min-height: 300px;

background-color: #f9f9f9;

}

h2 {

text-align: center;

margin-bottom: 10px;

}

.task {

padding: 10px;

margin: 5px 0;

background-color: lightblue;

cursor: grab;

border-radius: 5px;

transition: background-color 0.2s;

}

.task:hover {

background-color: lightgreen;

}

.task.dragging {

opacity: 0.5;

}

.column.dropzone {

background-color: lightyellow;

}

**JavaScript (taskmanager.js):**

const tasks = document.querySelectorAll('.task');

const columns = document.querySelectorAll('.column');

tasks.forEach(task => {

task.addEventListener('dragstart', () => {

task.classList.add('dragging');

});

task.addEventListener('dragend', () => {

task.classList.remove('dragging');

});

});

columns.forEach(column => {

column.addEventListener('dragover', event => {

event.preventDefault();

column.classList.add('dropzone');

});

column.addEventListener('dragleave', () => {

column.classList.remove('dropzone');

});

column.addEventListener('drop', event => {

const draggingTask = document.querySelector('.dragging');

column.appendChild(draggingTask);

column.classList.remove('dropzone');

});

});

**Exercise 11: Reservation System with Table and Form Validation**

**Task:**  
Build a table reservation system for a restaurant. Users can select a date, time, number of guests, and a table from available tables. The form should include validation, and after submitting, display the reservation details.

**HTML:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Reservation System</title>

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

</head>

<body>

<h1>Table Reservation</h1>

<form id="reservationForm">

<label for="date">Date:</label>

<input type="date" id="date" required><br>

<label for="time">Time:</label>

<input type="time" id="time" required><br>

<label for="guests">Number of Guests:</label>

<input type="number" id="guests" min="1" max="10" required><br>

<label for="table">Table:</label>

<select id="table" required>

<option value="">Select a table</option>

<option value="1">Table 1</option>

<option value="2">Table 2</option>

<option value="3">Table 3</option>

</select><br>

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

</form>

<div id="reservationDetails"></div>

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

</body>

</html>

**CSS (reservation.css):**

body {

font-family: Arial, sans-serif;

}

form {

width: 300px;

margin: 20px auto;

padding: 20px;

border: 1px solid #ccc;

background-color: #f9f9f9;

}

label {

display: block;

margin: 10px 0 5px;

}

input, select, button {

width: 100%;

padding: 8px;

margin-bottom: 10px;

}

button {

background-color: #4CAF50;

color: white;

border: none;

cursor: pointer;

font-size: 16px;

}

button:hover {

background-color: #45a049;

}

#reservationDetails {

width: 300px;

margin: 20px auto;

padding: 20px;

border: 1px solid #ccc;

background-color: #f1f1f1;

display: none;

}

**JavaScript (reservation.js):**

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

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

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

event.preventDefault();

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

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

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

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

if (!date || !time || !guests || !table) {

alert('Please fill out all fields.');

return;

}

displayReservationDetails(date, time, guests, table);

});

function displayReservationDetails(date, time, guests, table) {

reservationDetails.style.display = 'block';

reservationDetails.innerHTML = `

<h2>Reservation Details</h2>

<p><strong>Date:</strong> ${date}</p>

<p><strong>Time:</strong> ${time}</p>

<p><strong>Guests:</strong> ${guests}</p>

<p><strong>Table:</strong> ${table}</p>

`;

}

**Key Concepts Covered:**

1. **DOM Events** – Listening for click, dragstart, dragover, and submit events.
2. **Modal Popups** – Creating and manipulating modals with buttons for navigation.
3. **Drag-and-Drop API** – Making elements draggable and implementing drop zones.
4. **Form Validation** – Checking user input and conditionally displaying errors.
5. **Dynamic Content** – Inserting HTML dynamically to show details or update UI.