**TUGAS 4 PRAK DESAIN WEB A**

MEMBUAT APLIKASI MANAJEMEN DAFTAR TUGAS (To-Do List) INTERAKSI

**

Adjie Firmansyah (4523210004)

 Prak Desain Web A

1 - Oktober - 2024

Dosen Pembimbing :

Adi Wahyu Pribadi, S.Si., M.Kom.

**FAKULTAS TEKNIK**

**UNIVERSITAS PANCASILA**

**JAKARTA**

**2024**

Pendahuluan

Praktikum ini bertujuan untuk meningkatkan kemampuan mahasiswa dalam membangun aplikasi berbasis web menggunakan JavaScript. Mahasiswa diharapkan dapat memahami dan menerapkan konsep-konsep dasar JavaScript, seperti manipulasi DOM (Document Object Model), penggunaan class dan object, serta penanganan event untuk menciptakan aplikasi interaktif.

Aplikasi yang akan dibuat dalam praktikum ini adalah Aplikasi Manajemen Daftar Tugas (To-Do List). Aplikasi ini memungkinkan pengguna untuk menambahkan tugas baru, menandai tugas yang telah diselesaikan, dan menghapus tugas dari daftar. Selain itu, ada fitur validasi input untuk memastikan pengguna tidak dapat menambahkan tugas yang kosong, serta opsi untuk memfilter tugas berdasarkan statusnya (selesai atau belum selesai).

Teori dasar yang diterapkan dalam praktikum ini mencakup:  
Manipulasi DOM dengan JavaScript: Teknik untuk mengambil, memanipulasi, dan memperbarui elemen HTML secara dinamis.

Class dan Object: Penerapan konsep pemrograman berorientasi objek (OOP) dalam JavaScript untuk membangun struktur kode yang lebih terorganisir dan modular.

Penanganan Event: Mekanisme untuk menangani interaksi pengguna dengan antarmuka aplikasi, seperti mengklik tombol atau mengubah status checkbox, yang membuat aplikasi menjadi lebih interaktif.

Langkah Pengerjaan  
Berikut adalah langkah-langkah dalam mengerjakan praktikum pembuatan Aplikasi To-Do List interaktif menggunakan HTML, CSS, dan JavaScript:

1. **Menyiapkan Struktur HTML**

Langkah pertama adalah membuat file HTML yang berfungsi sebagai kerangka utama aplikasi. Pada tahap ini, saya membuat elemen-elemen dasar seperti input teks untuk menambahkan tugas, tombol untuk menambah tugas, dan daftar tugas yang akan ditampilkan.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

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

<title>To-Do List App</title>

</head>

<body>

<h1>To-Do List</h1>

<input type="text" id="task-input" placeholder="Enter a new task">

<button id="add-task-btn">Add Task</button>

<h3>Filter Tasks</h3>

<button onclick="filterTasks('all')">All</button>

<button onclick="filterTasks('active')">Active</button>

<button onclick="filterTasks('completed')">Completed</button>

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

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

</body>

</html>

#### **2. Membuat Desain dengan CSS**

Setelah struktur HTML selesai, langkah berikutnya adalah menambahkan gaya menggunakan CSS untuk membuat aplikasi lebih menarik dan mudah digunakan.

body {

font-family: Arial, sans-serif;

margin: 0 auto;

padding: 20px;

max-width: 600px;

background-image: url('https://static.vecteezy.com/system/resources/thumbnails/007/677/104/small\_2x/white-background-with-blue-geometric-and-white-line-pattern-free-vector.jpg'); /\* Replace with your image path \*/

background-size: cover;

}

h1 {

text-align: center;

color: blue;

}

h3 {

margin-left: 9px;

}

#task-input {

width: 80%;

padding: 10px;

font-size: 16px;

margin-left: 9px;

}

#add-task-btn {

padding: 10px;

font-size: 16px;

margin-top: 5px;

}

ul {

list-style-type: none;

padding: 0;

}

li {

display: flex;

justify-content: space-between;

padding: 10px;

border-bottom: 1px solid #ccc;

}

li.completed {

text-decoration: line-through;

}

button {

margin-left: 10px;

padding: 5px;

color: blue;

}

#### **3. Implementasi Fitur Menambahkan Tugas**

Setelah desain selesai, saya mulai mengimplementasikan fungsi JavaScript untuk menambahkan tugas ke daftar ketika pengguna menekan tombol "Add Task". Pada tahap ini, saya juga menambahkan validasi agar pengguna tidak dapat menambahkan tugas kosong.

**Potongan kode JavaScript untuk menambahkan tugas:**

function addTask() {

const taskInput = document.getElementById('task-input');

const taskName = taskInput.value.trim();

if (taskName === "") {

alert("Task cannot be empty!");

return;

}

const task = new Task(taskName);

tasks.push(task);

displayTasks();

taskInput.value = ''; // Kosongkan input setelah menambahkan

}

document.getElementById('add-task-btn').addEventListener('click', addTask);

**Class Task dan Fungsinya:**

**class Task {**

**constructor(name) {**

**this.name = name;**

**this.completed = false;**

**}**

**toggleCompleted() {**

**this.completed = !this.completed;**

**}**

**}**

* Class Task adalah representasi dari tugas yang akan dikelola dalam aplikasi. Setiap instance dari Task memiliki dua properti:
  + name: Nama atau deskripsi tugas.
  + isCompleted: Status apakah tugas sudah selesai (default: false).
* Fungsi class ini untuk menyimpan informasi tentang tugas secara struktural.

**Fungsi addTask:**

**function addTask() {**

**const taskInput = document.getElementById('task-input');**

**const taskName = taskInput.value.trim();**

**if (taskName === "") {**

**alert("Task cannot be empty!");**

**return;**

**}**

**const task = new Task(taskName);**

**tasks.push(task);**

**displayTasks();**

**taskInput.value = ''; // Kosongkan input setelah menambahkan**

**}**

* Fungsi ini digunakan untuk menambahkan tugas baru ke dalam array tasks yang menyimpan semua daftar tugas. Fungsi ini membuat objek Task baru dan menambahkannya ke array, kemudian menampilkan kembali seluruh daftar tugas yang ada.
* Validasi dilakukan untuk memastikan input tidak kosong sebelum tugas ditambahkan.

**Fungsi displayTasks:**

**function displayTasks() {**

**const taskList = document.getElementById('task-list');**

**taskList.innerHTML = '';**

**tasks.forEach((task, index) => {**

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

**taskItem.className = task.completed ? 'completed' : '';**

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

**taskText.textContent = task.name;**

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

**taskCheckbox.type = 'checkbox';**

**taskCheckbox.checked = task.completed;**

**taskCheckbox.addEventListener('click', () => toggleTask(index));**

**const deleteButton = document.createElement('button');**

**deleteButton.textContent = 'Delete';**

**deleteButton.addEventListener('click', () => deleteTask(index));**

**taskItem.appendChild(taskCheckbox);**

**taskItem.appendChild(taskText);**

**taskItem.appendChild(deleteButton);**

**taskList.appendChild(taskItem);**

**});**

**}**

* Fungsi ini bertanggung jawab untuk memperbarui tampilan daftar tugas di DOM. Setiap tugas akan ditampilkan dalam bentuk elemen <li> dengan checkbox untuk menandai status selesai dan tombol untuk menghapus tugas.
* Manipulasi DOM digunakan di sini untuk membuat elemen-elemen HTML yang baru sesuai dengan isi array tasks.

**Fungsi toggleTask:**

**function toggleTask(index) {**

**tasks[index].toggleCompleted();**

**displayTasks();**

**}**

* Fungsi ini digunakan untuk mengubah status selesai atau belum selesai pada sebuah tugas. Status isCompleted diubah berdasarkan interaksi pengguna dengan checkbox.

**Fungsi deleteTask:**

**function deleteTask(index) {**

**tasks.splice(index, 1);**

**displayTasks();**

**}**

* Fungsi ini menghapus tugas dari array tasks dan kemudian memperbarui tampilan DOM setelah tugas dihapus.

**Fungsi filterTasks:**

**function filterTasks(filter) {**

**let filteredTasks = tasks;**

**if (filter === 'active') {**

**filteredTasks = tasks.filter(task => !task.completed);**

**} else if (filter === 'completed') {**

**filteredTasks = tasks.filter(task => task.completed);**

**}**

**const taskList = document.getElementById('task-list');**

**taskList.innerHTML = '';**

**filteredTasks.forEach((task, index) => {**

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

**taskItem.className = task.completed ? 'completed' : '';**

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

**taskText.textContent = task.name;**

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

**taskCheckbox.type = 'checkbox';**

**taskCheckbox.checked = task.completed;**

**taskCheckbox.addEventListener('click', () => toggleTask(index));**

**const deleteButton = document.createElement('button');**

**deleteButton.textContent = 'Delete';**

**deleteButton.addEventListener('click', () => deleteTask(index));**

**taskItem.appendChild(taskCheckbox);**

**taskItem.appendChild(taskText);**

**taskItem.appendChild(deleteButton);**

**taskList.appendChild(taskItem);**

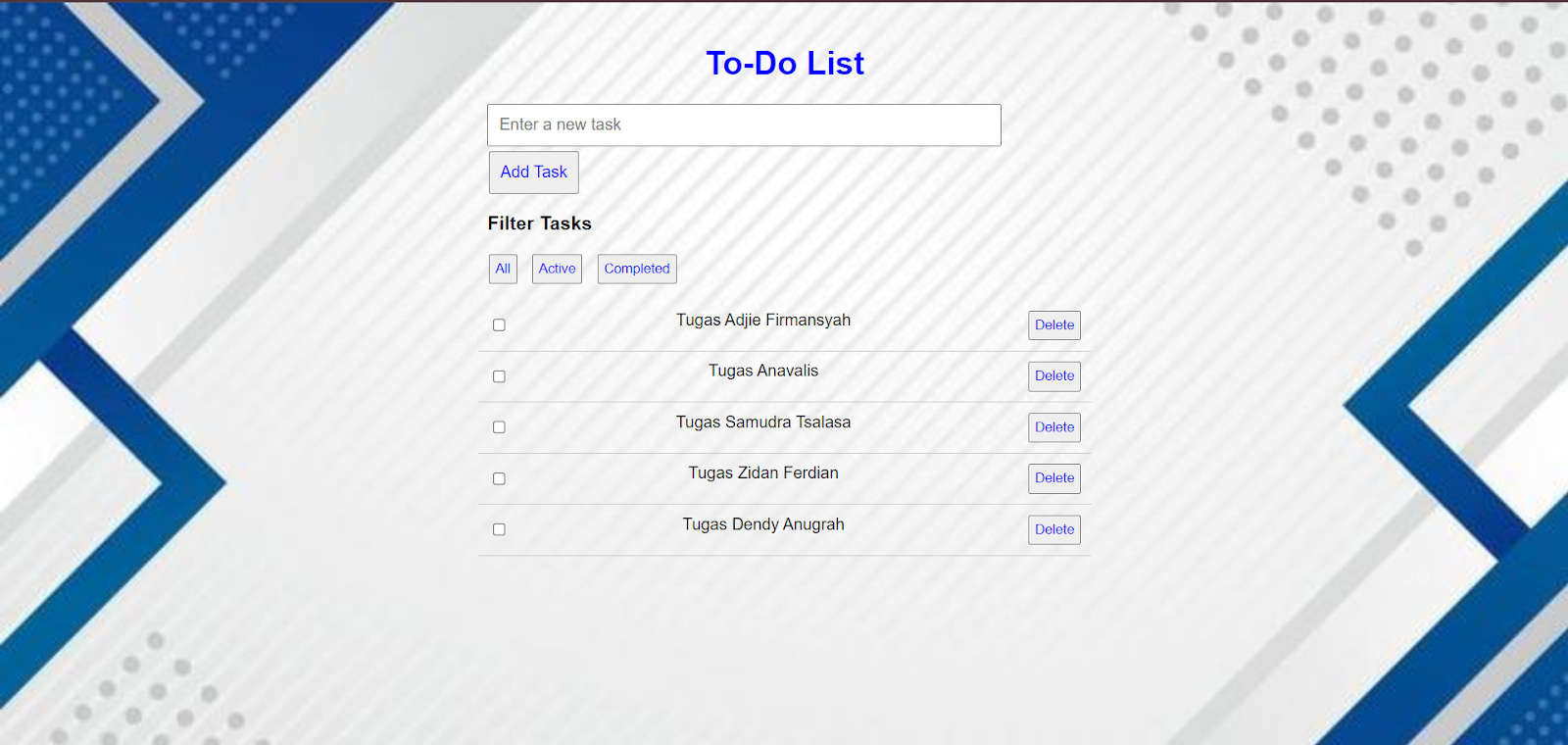
**});**

**}**

* Fungsi ini memungkinkan pengguna untuk memfilter tugas yang ditampilkan berdasarkan statusnya (semua tugas, tugas selesai, atau tugas belum selesai).

### Manipulasi DOM dan Event Handling

* **Manipulasi DOM:**
  + Digunakan di fungsi displayTasks untuk membuat elemen-elemen HTML dinamis seperti <li>, checkbox, dan tombol "Delete". Hal ini memungkinkan pembaruan real-time pada tampilan daftar tugas berdasarkan interaksi pengguna.
* **Event Handling:**
  + Pada tombol **“Add Task”**, event handling dilakukan menggunakan addEventListener('click', addTask) yang memicu fungsi addTask ketika tombol diklik.
  + Pada checkbox, event handling menangani perubahan status tugas (checked/unchecked). Fungsi toggleTask dipanggil saat checkbox diubah, menggunakan data index dari tugas.
  + Pada tombol **“Delete”**, event handling digunakan untuk menghapus tugas dari daftar, dengan memanfaatkan atribut data-index untuk mengetahui tugas yang harus dihapus.
* Hasil Uji Coba



* Kesimpulan

Melalui praktikum ini, saya telah memperoleh pemahaman yang lebih mendalam tentang HTML, JavaScript, dan CSS. Implementasi langsung dalam pembuatan aplikasi To-Do List memberikan gambaran praktis tentang bagaimana teknologi ini dapat diterapkan di dunia kerja nantinya. Semoga pengetahuan yang didapat dari praktikum ini terus bermanfaat di masa depan, terutama dalam pengembangan aplikasi berbasis web.