Calendario e Lista delle cose da fare (To-Do List).

Questo tipo di progetto è ideale per comprendere e applicare concetti fondamentali come HTML5, CSS3, JavaScript, tipi di dato, operatori, condizioni, cicli, funzioni e ambito (scope).

Specifiche Funzionali del Progetto

Il progetto prevede:

1. Una semplice interfaccia per visualizzare la data corrente.
2. Una lista di elementi "da fare" che l'utente può aggiungere o rimuovere.
3. Un contatore che mostra il numero di attività completate e da completare.
4. Possibilità di contrassegnare un'attività come "completata" con un semplice click.
5. Aggiornamento dinamico dell'interfaccia, utilizzando JavaScript per interagire con HTML e CSS.

Struttura del Progetto

- **HTML**: per creare la struttura base dell'interfaccia.
- **CSS**: per rendere l'interfaccia esteticamente gradevole.
- **JavaScript**: per aggiungere funzionalità interattive.

Passo 1: Struttura HTML

Iniziamo con la struttura della pagina.

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Lista delle cose da fare</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>

    <!-- Sezione Calendario -->
    <div class="calendar">
        <h1>Data corrente: <span id="current-date"></span></h1>
    </div>

    <!-- Sezione To-Do List -->
    <div class="todo-container">
        <h2>Lista delle Cose da Fare</h2>
        <div class="input-container">
            <input type="text" id="task-input" placeholder="Aggiungi
un'attività...">
            <button id="add-task-btn">Aggiungi</button>
        </div>
        <ul id="task-list"></ul>
```

```
        <p id="task-counter">Attività da completare: <span id="remaining-
tasks">0</span></p>
    </div>

    <script src="script.js"></script>
</body>
</html>
```

Passo 2: Stili CSS

Di seguito, il CSS per rendere l'interfaccia visivamente gradevole.

```
/* styles.css */

body {
    font-family: Arial, sans-serif;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
    background-color: #f4f4f9;
}

.calendar {
    margin-bottom: 20px;
}

h1, h2 {
    color: #333;
}

.todo-container {
    background-color: #fff;
    padding: 20px;
    border-radius: 5px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    width: 300px;
}

.input-container {
    display: flex;
}

input[type="text"] {
    width: 100%;
    padding: 8px;
    border: 1px solid #ccc;
    border-radius: 3px;
    outline: none;
}

button {
    padding: 8px 12px;
    background-color: #007bff;
    color: white;
    border: none;
    border-radius: 3px;
    cursor: pointer;
    margin-left: 5px;
```

```
}

button:hover {
    background-color: #0056b3;
}

#task-list {
    list-style-type: none;
    padding: 0;
}

.task-item {
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 10px;
    border-bottom: 1px solid #eee;
}

.task-item.completed {
    text-decoration: line-through;
    color: #888;
}

#task-counter {
    margin-top: 10px;
    color: #555;
}
```

Passo 3: Codice JavaScript

Infine, aggiungiamo il codice JavaScript per gestire le funzionalità della To-Do List.

```
// script.js

// Mostra la data corrente nell'elemento con id "current-date"
function displayCurrentDate() {
    const currentDate = new Date();
    document.getElementById("current-date").textContent =
currentDate.toLocaleDateString();
}

// Aggiornamento della data
displayCurrentDate();

// Seleziona elementi dal DOM per manipolarli
const taskInput = document.getElementById("task-input");
const addTaskBtn = document.getElementById("add-task-btn");
const taskList = document.getElementById("task-list");
const remainingTasks = document.getElementById("remaining-tasks");

// Inizializza il conteggio delle attività
let taskCount = 0;

// Aggiunge una nuova attività alla lista
function addTask() {
    const taskText = taskInput.value.trim();
    if (taskText === "") return; // Controllo di validità: la stringa non deve
essere vuota
```

```
    const taskItem = document.createElement("li");
    taskItem.classList.add("task-item");
    taskItem.textContent = taskText;

    // Aggiungi evento click per contrassegnare come completato
    taskItem.addEventListener("click", () => toggleTaskComplete(taskItem));

    // Crea pulsante di rimozione e gestisci l'evento
    const removeBtn = document.createElement("button");
    removeBtn.textContent = "Rimuovi";
    removeBtn.addEventListener("click", () => removeTask(taskItem));

    // Aggiungi pulsante alla voce di attività
    taskItem.appendChild(removeBtn);

    // Aggiungi l'elemento alla lista
    taskList.appendChild(taskItem);

    // Aggiorna il conteggio e resetta il campo input
    taskInput.value = "";
    taskCount++;
    updateRemainingTasks();
}

// Rimuove un'attività dalla lista
function removeTask(taskItem) {
    taskList.removeChild(taskItem);
    taskCount--;
    updateRemainingTasks();
}

// Contrassegna un'attività come completata o non completata
function toggleTaskComplete(taskItem) {
    taskItem.classList.toggle("completed");
}

// Aggiorna il contatore delle attività rimanenti
function updateRemainingTasks() {
    const incompleteTasks = document.querySelectorAll(".task-
item:not(.completed)").length;
    remainingTasks.textContent = incompleteTasks;
}

// Aggiunge l'attività al click sul pulsante
addTaskBtn.addEventListener("click", addTask);

// Permette di aggiungere l'attività premendo Invio
taskInput.addEventListener("keypress", (event) => {
    if (event.key === "Enter") {
        addTask();
    }
});
```

Spiegazione del Codice JavaScript

1. **`displayCurrentDate()`**: questa funzione utilizza l'oggetto `Date` di JavaScript per ottenere la data corrente e la visualizza nell'elemento con ID `current-date`.
2. **Elementi DOM**: variabili come `taskInput`, `addTaskBtn`, `taskList`, `remainingTasks` servono per accedere a elementi HTML specifici e gestirli con JavaScript.

3. **`addTask()`**:
    - Verifica che l'input non sia vuoto.
    - Crea un nuovo elemento `<li>` con il testo dell'attività e aggiunge un pulsante per la rimozione.
    - Aggancia il pulsante di rimozione e una funzione che alterna lo stato completato dell'attività.
    - Aggiorna il contatore delle attività.
4. **`removeTask(taskItem)`**: rimuove l'elemento dalla lista e decrementa il contatore.
5. **`toggleTaskComplete(taskItem)`**: utilizza la classe `completed` per visualizzare l'attività come completata o da completare.
6. **`updateRemainingTasks()`**: aggiorna il contatore basandosi sugli elementi non completati.
7. **Event Listener**: consente di aggiungere attività sia cliccando il pulsante "Aggiungi" sia premendo "Enter".

Conclusione

Questo progetto incorpora molti dei concetti fondamentali di HTML, CSS e JavaScript:

- **Struttura HTML**: crea una base organizzata per l'interfaccia.
- **CSS**: migliora la presentazione e l'usabilità.
- **JavaScript**: offre funzionalità interattive, gestisce dati e condizioni e aggiorna il DOM.