Travaux Dirigés

Mini-Projet : Liste de Tâches Interactive

HTML, CSS & JavaScript

Ibrahim ALAME

Formation Vue.js - 2025

Objectifs

Durée : 2 heures

Mettre en pratique les concepts vus en cours :

- Structure HTML sémantique
- Mise en forme CSS (box model, flexbox)
- Manipulation du DOM avec JavaScript
- Gestion des événements
- Stockage local (bonus)

Table des matières

1 Description du projet

Vous allez créer une application de gestion de tâches (Todo List) permettant À l'utilisateur de gérer efficacement ses tâches quotidiennes.

1.1 Fonctionnalités principales

1. Ajouter une nouvelle tâche
2. Marquer une tâche comme complétée
3. Supprimer une tâche
4. Filtrer les tâches (toutes / actives / complétées)
5. Afficher un compteur de tâches actives

1.2 Rendu attendu

Ma Liste de Tâches

[Nouvelle tâche...] [Ajouter]

☐ Faire les courses [Supprimer]
☒ ~~Réviser JavaScript~~ [Supprimer]
☐ Appeler le médecin [Supprimer]

---

2 tâches actives [Toutes] [Actives] [Complétées] [Effacer complétées]

FIGURE 1 – Aperçu de l'application Todo List

2 Spécifications fonctionnelles

2.1 Niveau 1 - Fonctionnalités obligatoires

2.1.1 Ajouter une tâche

— Champ input pour saisir le titre de la tâche
— Bouton "Ajouter" pour valider
— Validation également possible avec la touche `Enter`
— La tâche apparaît immédiatement dans la liste
— Le champ input est vidé après ajout
— Ne pas ajouter de tâche vide (validation)

2.1.2 Marquer comme complétée

— Case À cocher pour chaque tâche
— Le texte est barré quand la tâche est complétée
— Style visuel différent (opacité réduite)
— Possibilité de décocher (marquer comme non complétée)

2.1.3 Supprimer une tâche

— Bouton "Supprimer" pour chaque tâche
— Suppression immédiate de la liste
— Pas de confirmation demandée

2.1.4 Compteur de tâches

— Afficher le nombre de tâches actives (non complétées)
— Format : "X tâche(s) active(s)"
— Mise À jour automatique

2.2 Niveau 2 - Fonctionnalités avancées

2.2.1 Filtrer les tâches

— Trois boutons : Toutes / Actives / Complétées
— Affichage conditionnel des tâches selon le filtre
— Style différent pour le filtre actif
— Par défaut, afficher toutes les tâches

2.2.2 Effacer les tâches complétées

— Bouton "Effacer complétées"
— Supprime toutes les tâches marquées comme complétées
— Mise À jour immédiate de l'affichage

2.3 Niveau 3 - Bonus

Bonus

Pour aller plus loin :

— Sauvegarder les tâches dans le `localStorage`
— Restaurer les tâches au chargement de la page
— Ajouter des animations CSS (ajout/suppression)
— Implémenter un mode sombre/clair
— Permettre l'édition d'une tâche (double-clic)

3 Structure des fichiers

Créez une structure de projet avec trois fichiers :

```
projet-todo/
â”œâ”€â”€ index.html
â”œâ”€â”€ style.css
â””â”€â”€ script.js
```

4 Code de départ

4.1 Fichier HTML : index.html

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
     initial-scale=1.0">
  <title>Ma Todo List</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <div class="container">
    <header>
      <h1>Ma Liste de Taches</h1>
    </header>

    <main>
      <!-- Formulaire d'ajout -->
      <div class="add-task">
        <input type="text" id="task-input"
               placeholder="Nouvelle tache...">
        <button id="add-btn">Ajouter</button>
      </div>

      <!-- Liste des taches -->
      <ul id="task-list">
        <!-- Les taches seront ajoutees ici dynamiquement -->
      </ul>

      <!-- Filtres et statistiques -->
      <footer class="task-footer">
        <span id="task-count">0 taches actives</span>

        <div class="filters">
          <button class="filter-btn active"
                  data-filter="all">Toutes</button>
          <button class="filter-btn"
                  data-filter="active">Actives</button>
```

```
            <button class="filter-btn"
                    data-filter="completed">Completees</button>
        </div>

        <button id="clear-completed">Effacer completees</button>
      </footer>
    </main>
  </div>

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

4.2 Fichier CSS : style.css

```
/* Reset et variables */
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

:root {
  --primary-color: #4CAF50;
  --danger-color: #f44336;
  --bg-color: #f5f5f5;
  --text-color: #333;
  --border-color: #ddd;
}

body {
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  background-color: var(--bg-color);
  color: var(--text-color);
  line-height: 1.6;
}

.container {
  max-width: 600px;
  margin: 50px auto;
  background: white;
  border-radius: 10px;
  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
  overflow: hidden;
}

header {
  background: var(--primary-color);
  color: white;
  padding: 20px;
```

```
  text-align: center;
}

main {
  padding: 20px;
}

/* Formulaire d'ajout */
.add-task {
  display: flex;
  gap: 10px;
  margin-bottom: 20px;
}

#task-input {
  flex: 1;
  padding: 10px;
  border: 2px solid var(--border-color);
  border-radius: 5px;
  font-size: 16px;
}

#task-input:focus {
  outline: none;
  border-color: var(--primary-color);
}

button {
  padding: 10px 20px;
  background: var(--primary-color);
  color: white;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  font-size: 16px;
  transition: background 0.3s;
}

button:hover {
  opacity: 0.9;
}

/* Liste de taches */
#task-list {
  list-style: none;
  margin-bottom: 20px;
}

.task-item {
  display: flex;
  align-items: center;
```

```
  padding: 15px;
  border-bottom: 1px solid var(--border-color);
  transition: background 0.2s;
}

.task-item:hover {
  background: #fafafa;
}

.task-item.completed .task-text {
  text-decoration: line-through;
  opacity: 0.6;
}

.task-item input[type="checkbox"] {
  margin-right: 15px;
  width: 20px;
  height: 20px;
  cursor: pointer;
}

.task-text {
  flex: 1;
  font-size: 16px;
}

.task-delete {
  background: var(--danger-color);
  padding: 5px 10px;
  font-size: 14px;
}

/* Footer */
.task-footer {
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding-top: 15px;
  border-top: 2px solid var(--border-color);
}

.filters {
  display: flex;
  gap: 5px;
}

.filter-btn {
  background: transparent;
  color: var(--text-color);
  border: 1px solid var(--border-color);
  padding: 5px 15px;
```

```
  font-size: 14px;
}

.filter-btn.active {
  background: var(--primary-color);
  color: white;
  border-color: var(--primary-color);
}

#clear-completed {
  background: var(--danger-color);
  font-size: 14px;
  padding: 5px 15px;
}
```

4.3 Fichier JavaScript : script.js (Structure)

```
// Attendre que le DOM soit charge
document.addEventListener('DOMContentLoaded', () => {

  // === SELECTION DES ELEMENTS ===
  const taskInput = document.getElementById('task-input');
  const addBtn = document.getElementById('add-btn');
  const taskList = document.getElementById('task-list');
  const taskCount = document.getElementById('task-count');
  const filterBtns = document.querySelectorAll('.filter-btn');
  const clearCompletedBtn = document.getElementById('clear-
     completed');

  // === VARIABLES GLOBALES ===
  let tasks = [];
  let currentFilter = 'all';

  // === FONCTIONS ===

  // Generer un ID unique
  function generateId() {
    return Date.now().toString(36) +
           Math.random().toString(36).substr(2);
  }

  // TODO: Implementer la fonction addTask(text)

  // TODO: Implementer la fonction toggleTask(id)

  // TODO: Implementer la fonction deleteTask(id)

  // TODO: Implementer la fonction getFilteredTasks()

  // TODO: Implementer la fonction renderTasks()
```

```
33
34    // TODO: Implementer la fonction updateTaskCount()
35
36    // TODO: Implementer la fonction clearCompleted()
37
38    // === GESTIONNAIRES D'EVENEMENTS ===
39
40    // TODO: Ajouter les event listeners
41
42    // === INITIALISATION ===
43    renderTasks();
44    updateTaskCount();
45 });
```

5 Déroulement du TD

5.1 Phase 1 : Mise en place (15 min)

1. Créer la structure de fichiers
2. Copier le code HTML de base
3. Copier le code CSS complet
4. Tester l'affichage dans le navigateur
5. Ouvrir la console (F12) pour les tests JavaScript

Astuce

Utilisez l'extension "Live Server" de VS Code pour un rechargement automatique !

5.2 Phase 2 : Fonctionnalités de base (40 min)

5.2.1 Exercice 1 : Ajouter une tâche (15 min)

Implémentez la fonction `addTask(text)` qui :

1. Vérifie que le texte n'est pas vide
2. Crée un objet tâche avec : id, text, completed
3. Ajoute l'objet au tableau `tasks`
4. Appelle `renderTasks()` pour afficher
5. Vide le champ input

```
1 function addTask(text) {
2   if (text.trim() === '') return;
3
4   const task = {
5     id: generateId(),
6     text: text,
7     completed: false
8   };
```

```
  tasks.push(task);
  renderTasks();
  updateTaskCount();
  taskInput.value = '';
}
```

Ajoutez les gestionnaires d'événements :

```
// Bouton Ajouter
addBtn.addEventListener('click', () => {
  addTask(taskInput.value);
});

// Touche Enter
taskInput.addEventListener('keypress', (e) => {
  if (e.key === 'Enter') {
    addTask(taskInput.value);
  }
});
```

5.2.2 Exercice 2 : Afficher les tâches (10 min)

Implémentez la fonction `renderTasks()` :

```
function renderTasks() {
  const filteredTasks = getFilteredTasks();
  taskList.innerHTML = '';

  filteredTasks.forEach(task => {
    const li = document.createElement('li');
    li.className = `task-item ${task.completed ? 'completed' : ''
       }`;
    li.innerHTML = `
      <input type="checkbox" ${task.completed ? 'checked' : ''}>
      <span class="task-text">${task.text}</span>
      <button class="task-delete">Supprimer</button>
    `;

    // Evenement checkbox
    li.querySelector('input').addEventListener('change', () => {
      toggleTask(task.id);
    });

    // Evenement suppression
    li.querySelector('.task-delete').addEventListener('click', ()
       => {
      deleteTask(task.id);
    });

    taskList.appendChild(li);
  });
```

```
}
```

5.2.3 Exercice 3 : Marquer comme complétée (5 min)

```
function toggleTask(id) {
  const task = tasks.find(t => t.id === id);
  if (task) {
    task.completed = !task.completed;
    renderTasks();
    updateTaskCount();
  }
}
```

5.2.4 Exercice 4 : Supprimer une tâche (5 min)

```
function deleteTask(id) {
  tasks = tasks.filter(t => t.id !== id);
  renderTasks();
  updateTaskCount();
}
```

5.2.5 Exercice 5 : Compteur (5 min)

```
function updateTaskCount() {
  const activeCount = tasks.filter(t => !t.completed).length;
  const s = activeCount !== 1 ? 's' : '';
  taskCount.textContent =
    `${activeCount} tache${s} active${s}`;
}
```

Important

Testez chaque fonction dans la console avant de passer À la suivante !

5.3 Phase 3 : Fonctionnalités avancées (30 min)

5.3.1 Exercice 6 : Filtrer les tâches (20 min)

Implémentez `getFilteredTasks()` :

```
function getFilteredTasks() {
  switch (currentFilter) {
    case 'active':
      return tasks.filter(t => !t.completed);
    case 'completed':
      return tasks.filter(t => t.completed);
    default:
      return tasks;
```

```
  }
}
```

Ajoutez les gestionnaires pour les boutons de filtre :

```
filterBtns.forEach(btn => {
  btn.addEventListener('click', () => {
    // Retirer la classe active de tous les boutons
    filterBtns.forEach(b => b.classList.remove('active'));
    // Ajouter au bouton clique
    btn.classList.add('active');

    // Changer le filtre actuel
    currentFilter = btn.dataset.filter;
    renderTasks();
  });
});
```

5.3.2 Exercice 7 : Effacer les complétées (10 min)

```
function clearCompleted() {
  tasks = tasks.filter(t => !t.completed);
  renderTasks();
  updateTaskCount();
}

clearCompletedBtn.addEventListener('click', clearCompleted);
```

5.4 Phase 4 : Bonus (si temps) (10 min)

5.4.1 LocalStorage

Ajoutez la sauvegarde automatique :

```
// Sauvegarder apres chaque modification
function saveTasks() {
  localStorage.setItem('tasks', JSON.stringify(tasks));
}

// Charger au demarrage
function loadTasks() {
  const saved = localStorage.getItem('tasks');
  if (saved) {
    tasks = JSON.parse(saved);
  }
}

// Appelez loadTasks() au debut
// Appelez saveTasks() dans addTask, toggleTask, deleteTask
```

5.4.2 Animations CSS

Ajoutez dans `style.css` :

```css
.task-item {
  animation: slideIn 0.3s ease-out;
}

@keyframes slideIn {
  from {
    transform: translateX(-100%);
    opacity: 0;
  }
  to {
    transform: translateX(0);
    opacity: 1;
  }
}
```

6 Critères d'évaluation

Critère	Points
HTML bien structuré et valide	2
CSS propre et lisible	2
Ajouter une tâche (avec validation)	3
Marquer comme complétée	2
Supprimer une tâche	2
Compteur de tâches	1
Filtres fonctionnels	3
Code JavaScript propre et commenté	3
Bonus (LocalStorage, animations, etc.)	2
Total	**20**

TABLE 1 – Grille d'évaluation

7 Astuces et conseils

7.1 Astuces JavaScript

Astuce

Déboguer efficacement :

- Utilisez `console.log()` pour afficher les valeurs
- Inspectez le tableau `tasks` dans la console
- Utilisez le débogueur de Chrome (F12 > Sources)
- Vérifiez les messages d'erreur dans la console

7.2 Pièges courants

Important

Erreurs fréquentes À éviter :

1. Ne pas vider le champ input après ajout
2. Oublier de vérifier si le texte est vide
3. Ne pas mettre À jour le compteur
4. Oublier de rappeler `renderTasks()` après modification
5. Problèmes avec `this` dans les event listeners

7.3 Bonnes pratiques

— Utilisez `const` par défaut, `let` si nécessaire
— Noms de variables explicites (`task`, pas `t`)
— Commentez votre code
— Testez chaque fonction individuellement
— Utilisez les fonctions fléchées pour les callbacks
— À‰vitez la répétition de code (principe DRY)

8 Extensions possibles

Bonus

Pour aller encore plus loin :

1. **Catégories** : Ajouter des catégories aux tâches (Travail, Personnel, Urgent)
2. **Dates** : Ajouter une date d'échéance et trier par date
3. **Priorités** : Niveaux de priorité avec code couleur
4. **Drag & Drop** : Réorganiser les tâches par glisser-déposer
5. **Recherche** : Barre de recherche pour filtrer par texte
6. **Statistiques** : Graphique des tâches complétées
7. **Mode sombre** : Toggle pour changer le thème
8. **Export** : Télécharger la liste en format JSON ou TXT

9 Ressources

9.1 Documentation

— **MDN Web Docs** : https://developer.mozilla.org
— **JavaScript.info** : https://javascript.info
— **CSS-Tricks** : https://css-tricks.com

9.2 Outils

— Chrome DevTools (F12)
— VS Code avec Live Server
— JSHint pour vérifier le code

10 Conclusion

Objectifs

Compétences acquises :
À€ la fin de ce TD, vous devez Être capable de :

- ✓ Créer une structure HTML sémantique
- ✓ Appliquer des styles CSS modernes
- ✓ Manipuler le DOM avec JavaScript
- ✓ Gérer des événements utilisateur
- ✓ Créer et gérer un tableau d'objets
- ✓ Filtrer et afficher des données conditionnellement
- ✓ Utiliser le localStorage (bonus)

Prochaine étape : Nous allons recréer cette mÊme application avec Vue.js et constater la différence !

Bon travail !