Travaux Pratiques - ÉNONCÉ

Application : Gestionnaire de Notes

HTML, CSS & JavaScript

Document pour les étudiants

Ibrahim ALAME

Formation Vue.js - 2025

Objectifs

Durée : 2 heures

Mettre en pratique les concepts avancés :

- Manipulation avancée du DOM
- Gestion des formulaires complexes
- LocalStorage et persistance des données
- Recherche et filtrage de données
- Édition dynamique de contenu

Table des matières

1 Description du projet

Vous allez créer une application de gestion de notes (Note Keeper) permettant à l'utilisateur de créer, modifier, rechercher et organiser ses notes.

1.1 Fonctionnalités principales

1. Créer une nouvelle note avec titre et contenu
2. Modifier une note existante
3. Supprimer une note
4. Rechercher dans les notes
5. Afficher la date de création
6. Sauvegarder automatiquement dans localStorage

1.2 Rendu attendu

Mes Notes

[Rechercher une note...]

Nouvelle Note
Titre : [ ]
Contenu :
[ ]
[ ]
[Ajouter la note]

Liste des Courses 23/11/2025 14 :30
Lait, Pain, Œufs, Fromage
[*Modifier*] [*Supprimer*]

Idées Projet Vue.js 22/11/2025 10 :15
Créer un dashboard avec graphiques...
[*Modifier*] [*Supprimer*]

FIGURE 1 – Aperçu de l'application Note Keeper

2 Spécifications fonctionnelles

2.1 Niveau 1 - Fonctionnalités de base (60 min)

2.1.1 Créer une note

— Formulaire avec champ titre et zone de texte (textarea)

— Validation : titre et contenu obligatoires
— Génération automatique de la date/heure de création
— Affichage immédiat de la nouvelle note
— Réinitialisation du formulaire après ajout

2.1.2 Afficher les notes

— Affichage en liste ou en grille
— Chaque note affiche : titre, extrait du contenu (100 premiers caractères), date
— Boutons Modifier et Supprimer pour chaque note
— Ordre : plus récentes en premier

2.1.3 Supprimer une note

— Bouton Supprimer sur chaque note
— Confirmation avant suppression
— Mise à jour immédiate de l'affichage

2.1.4 Sauvegarde automatique

— Utilisation du localStorage
— Sauvegarde après chaque modification
— Chargement automatique au démarrage

2.2 Niveau 2 - Fonctionnalités avancées (40 min)

2.2.1 Modifier une note

— Clic sur "Modifier" : remplir le formulaire avec les données existantes
— Le bouton "Ajouter" devient "Mettre à jour"
— Possibilité d'annuler la modification
— Conservation de la date de création originale

2.2.2 Rechercher dans les notes

— Barre de recherche en temps réel
— Recherche dans le titre ET le contenu
— Insensible à la casse
— Affichage dynamique des résultats

2.3 Niveau 3 - Bonus (20 min)

Bonus

Pour aller plus loin :

- Ajouter un système de catégories/tags
- Tri par date, titre ou catégorie
- Export des notes en JSON
- Mode d'affichage : liste ou grille
- Compteur de caractères en temps réel
- Coloration syntaxique pour le code

3 Structure des fichiers

Créez une structure de projet avec trois fichiers :

```
projet-notes/

  index.html
  style.css
  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>Gestionnaire de Notes</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <div class="container">
    <header>
      <h1>     Mes Notes</h1>
      <div class="search-box">
        <input type="text" id="search-input"
               placeholder="Rechercher une note...">
      </div>
    </header>

    <main>
      <!-- Formulaire de creation/edition -->
```

```
      <section class="note-form">
        <h2>Nouvelle Note</h2>
        <form id="note-form">
          <div class="form-group">
            <label for="note-title">Titre :</label>
            <input type="text" id="note-title"
                   placeholder="Titre de la note" required>
          </div>

          <div class="form-group">
            <label for="note-content">Contenu :</label>
            <textarea id="note-content" rows="5"
                      placeholder="Contenu de la note..." required>
            </textarea>
          </div>

          <div class="form-actions">
            <button type="submit" id="submit-btn">
              Ajouter la note
            </button>
            <button type="button" id="cancel-btn"
                    class="btn-secondary" style="display: none;">
              Annuler
            </button>
          </div>
        </form>
      </section>

      <!-- Liste des notes -->
      <section class="notes-list">
        <div id="notes-container">
          <!-- Les notes seront ajoutees ici -->
        </div>
        <div id="empty-state" style="display: none;">
          <p>Aucune note pour le moment. Creez votre premiere note
             !</p>
        </div>
      </section>
    </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: #6C5CE7;
  --secondary-color: #A29BFE;
  --danger-color: #FF7675;
  --success-color: #00B894;
  --bg-color: #F8F9FA;
  --text-color: #2D3436;
  --border-color: #DFE6E9;
  --shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
}

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: 900px;
  margin: 0 auto;
  padding: 20px;
}

/* Header */
header {
  background: linear-gradient(135deg, var(--primary-color),
                              var(--secondary-color));
  color: white;
  padding: 30px;
  border-radius: 10px;
  margin-bottom: 30px;
  box-shadow: var(--shadow);
}

header h1 {
  font-size: 2rem;
  margin-bottom: 20px;
  text-align: center;
}

.search-box {
  max-width: 500px;
  margin: 0 auto;
}

#search-input {
```

```
  width: 100%;
  padding: 12px 20px;
  border: none;
  border-radius: 25px;
  font-size: 16px;
  background: rgba(255, 255, 255, 0.9);
  transition: all 0.3s;
}

#search-input:focus {
  outline: none;
  background: white;
  box-shadow: 0 0 0 3px rgba(255, 255, 255, 0.3);
}

/* Formulaire */
.note-form {
  background: white;
  padding: 25px;
  border-radius: 10px;
  box-shadow: var(--shadow);
  margin-bottom: 30px;
}

.note-form h2 {
  color: var(--primary-color);
  margin-bottom: 20px;
}

.form-group {
  margin-bottom: 20px;
}

.form-group label {
  display: block;
  margin-bottom: 8px;
  font-weight: 600;
  color: var(--text-color);
}

.form-group input,
.form-group textarea {
  width: 100%;
  padding: 12px;
  border: 2px solid var(--border-color);
  border-radius: 8px;
  font-size: 16px;
  font-family: inherit;
  transition: border-color 0.3s;
}

```

```css
.form-group input:focus,
.form-group textarea:focus {
  outline: none;
  border-color: var(--primary-color);
}

.form-group textarea {
  resize: vertical;
  min-height: 120px;
}

.form-actions {
  display: flex;
  gap: 10px;
}

button {
  padding: 12px 25px;
  background: var(--primary-color);
  color: white;
  border: none;
  border-radius: 8px;
  cursor: pointer;
  font-size: 16px;
  font-weight: 600;
  transition: all 0.3s;
}

button:hover {
  transform: translateY(-2px);
  box-shadow: 0 4px 12px rgba(108, 92, 231, 0.3);
}

.btn-secondary {
  background: var(--border-color);
  color: var(--text-color);
}

.btn-secondary:hover {
  background: #B2BEC3;
}

/* Liste des notes */
#notes-container {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));
  gap: 20px;
}

.note-card {
  background: white;
```

```
  padding: 20px;
  border-radius: 10px;
  box-shadow: var(--shadow);
  transition: transform 0.3s, box-shadow 0.3s;
  border-left: 4px solid var(--primary-color);
}

.note-card:hover {
  transform: translateY(-5px);
  box-shadow: 0 5px 20px rgba(0, 0, 0, 0.15);
}

.note-header {
  display: flex;
  justify-content: space-between;
  align-items: start;
  margin-bottom: 15px;
}

.note-title {
  font-size: 1.2rem;
  font-weight: 700;
  color: var(--primary-color);
  margin-bottom: 5px;
}

.note-date {
  font-size: 0.85rem;
  color: #636E72;
}

.note-content {
  margin-bottom: 15px;
  color: var(--text-color);
  line-height: 1.6;
}

.note-actions {
  display: flex;
  gap: 10px;
}

.note-actions button {
  flex: 1;
  padding: 8px 15px;
  font-size: 14px;
}

.btn-edit {
  background: var(--success-color);
}
```

```css

.btn-delete {
  background: var(--danger-color);
}

/* État vide */
#empty-state {
  text-align: center;
  padding: 60px 20px;
  color: #B2BEC3;
  font-size: 1.2rem;
}

/* Animations */
@keyframes slideIn {
  from {
    opacity: 0;
    transform: translateY(20px);
  }
  to {
    opacity: 1;
    transform: translateY(0);
  }
}

.note-card {
  animation: slideIn 0.3s ease-out;
}
```

4.3 Fichier JavaScript : script.js (Structure)

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

  // === SÉLECTION DES ÉLÉMENTS ===
  const noteForm = document.getElementById('note-form');
  const noteTitle = document.getElementById('note-title');
  const noteContent = document.getElementById('note-content');
  const submitBtn = document.getElementById('submit-btn');
  const cancelBtn = document.getElementById('cancel-btn');
  const searchInput = document.getElementById('search-input');
  const notesContainer = document.getElementById('notes-container')
    ;
  const emptyState = document.getElementById('empty-state');

  // === VARIABLES GLOBALES ===
  let notes = [];
  let editingNoteId = null;

  // === FONCTIONS PRINCIPALES ===
```

```
19
20  // Generer un ID unique
21  function generateId() {
22    return Date.now().toString(36) +
23           Math.random().toString(36).substr(2);
24  }
25
26  // Obtenir la date formatee
27  function getFormattedDate() {
28    const now = new Date();
29    const day = String(now.getDate()).padStart(2, '0');
30    const month = String(now.getMonth() + 1).padStart(2, '0');
31    const year = now.getFullYear();
32    const hours = String(now.getHours()).padStart(2, '0');
33    const minutes = String(now.getMinutes()).padStart(2, '0');
34    return `${day}/${month}/${year} ${hours}:${minutes}`;
35  }
36
37  // TODO: Implementer loadNotes()
38  // Charger les notes depuis localStorage
39
40  // TODO: Implementer saveNotes()
41  // Sauvegarder les notes dans localStorage
42
43  // TODO: Implementer addNote(title, content)
44  // Ajouter une nouvelle note
45
46  // TODO: Implementer updateNote(id, title, content)
47  // Mettre a jour une note existante
48
49  // TODO: Implementer deleteNote(id)
50  // Supprimer une note
51
52  // TODO: Implementer startEditNote(id)
53  // Commencer l'edition d'une note
54
55  // TODO: Implementer cancelEdit()
56  // Annuler l'edition en cours
57
58  // TODO: Implementer searchNotes(query)
59  // Rechercher dans les notes
60
61  // TODO: Implementer renderNotes(notesToRender)
62  // Afficher les notes
63
64  // TODO: Implementer createNoteCard(note)
65  // Creer la carte HTML d'une note
66
67  // === GESTIONNAIRES D'EVENEMENTS ===
68
69  // TODO: Ajouter les event listeners
```

```
70
71    // === INITIALISATION ===
72    loadNotes();
73    renderNotes(notes);
74  });
```

5 Déroulement du TP

5.1 Phase 1 : Mise en place (10 min)

1. Créer la structure de fichiers
2. Copier le code HTML et CSS complets
3. Copier la structure JavaScript
4. Tester l'affichage dans le navigateur
5. Ouvrir la console développeur (F12)

5.2 Phase 2 : Fonctionnalités de base (50 min)

5.2.1 Exercice 1 : LocalStorage (10 min)

À faire : Implémentez les fonctions `loadNotes()` et `saveNotes()` selon les consignes fournies en cours.

5.2.2 Exercice 2 : Ajouter une note (15 min)

À faire : Implémentez la fonction selon les consignes du cours et les spécifications.

5.2.3 Exercice 3 : Afficher les notes (15 min)

À faire : Implémentez la fonction selon les consignes du cours et les spécifications.

5.2.4 Exercice 4 : Supprimer une note (10 min)

À faire : Implémentez la fonction selon les consignes du cours et les spécifications.

5.3 Phase 3 : Fonctionnalités avancées (40 min)

5.3.1 Exercice 5 : Modifier une note (20 min)

À faire : Implémentez la fonction selon les consignes du cours et les spécifications.

5.3.2 Exercice 6 : Recherche (20 min)

À faire : Implémentez la fonction selon les consignes du cours et les spécifications.

5.4 Phase 4 : Bonus (20 min)

5.4.1 Compteur de caractères

À faire : Implémentez cette fonctionnalité bonus en suivant les consignes.
En JavaScript :

```
const charCount = document.getElementById('char-count');

noteContent.addEventListener('input', () => {
  const count = noteContent.value.length;
  charCount.textContent = `${count} caractere${count > 1 ? 's' : ''}`;
});
```

5.4.2 Export JSON

Ajoutez un bouton d'export :

```
function exportNotes() {
  const dataStr = JSON.stringify(notes, null, 2);
  const dataBlob = new Blob([dataStr], { type: 'application/json' });
  const url = URL.createObjectURL(dataBlob);

  const link = document.createElement('a');
  link.href = url;
  link.download = 'mes-notes.json';
  link.click();

  URL.revokeObjectURL(url);
}
```

6 Critères d'évaluation

Critère	Points
HTML bien structuré et sémantique	2
CSS responsive et attractif	2
Ajouter une note (avec validation)	3
Afficher les notes correctement	2
Supprimer une note (avec confirmation)	2
Modifier une note	3
Recherche fonctionnelle	2
LocalStorage (sauvegarde/chargement)	2
Code propre et commenté	2
Total	**20**

TABLE 1 – Grille d'évaluation

7 Astuces et conseils

7.1 Astuces JavaScript

Astuce

Déboguer efficacement :

- Utilisez `console.log()` pour afficher l'état
- Vérifiez le contenu du localStorage dans DevTools
- Testez chaque fonction individuellement
- Utilisez le débogueur pour suivre l'exécution

7.2 Pièges courants

Important

Erreurs fréquentes à éviter :

1. Ne pas échapper les guillemets dans le HTML généré
2. Oublier de sauvegarder après modification
3. Ne pas gérer le cas où localStorage est vide
4. Oublier de rendre les fonctions globales pour onclick
5. Ne pas réinitialiser le formulaire après ajout
6. Problèmes avec `this` dans les event listeners

7.3 Bonnes pratiques

- Séparez la logique métier de l'affichage
- Utilisez des noms de fonctions descriptifs
- Commentez les parties complexes
- Validez toujours les entrées utilisateur
- Gérez les cas d'erreur
- Utilisez `const` par défaut, `let` si nécessaire
- Testez sur différents navigateurs

8 Extensions possibles

Bonus

Pour aller encore plus loin :

1. **Catégories** : Ajouter un système de tags/catégories avec filtrage
2. **Markdown** : Support du formatage Markdown dans les notes
3. **Favoris** : Marquer des notes comme favorites
4. **Tri** : Trier par date, titre ou contenu
5. **Mode sombre** : Toggle pour changer le thème
6. **Statistiques** : Afficher le nombre total de notes, de mots, etc.
7. **Partage** : Générer un lien pour partager une note
8. **Import** : Importer des notes depuis un fichier JSON
9. **Couleurs** : Attribuer une couleur à chaque note
10. **Pinboard** : Épingler des notes importantes en haut

9 Différences avec le TD

Ce TP se distingue du TD1 (Todo List) par :

— **Complexité accrue** : Gestion de contenu long (textarea vs input)
— **Édition en place** : Modification de notes existantes
— **Recherche avancée** : Filtrage en temps réel sur plusieurs champs
— **Affichage riche** : Cartes avec preview, dates formatées
— **UX améliorée** : Confirmation, annulation, scroll automatique

10 Ressources

10.1 Documentation

— **MDN Web Docs** : https://developer.mozilla.org
— **JavaScript.info** : https://javascript.info
— **LocalStorage API** : https://developer.mozilla.org/fr/docs/Web/API/Window/localStorage

10.2 Outils

— Chrome DevTools (F12)
— VS Code avec Live Server
— JSON Formatter pour vérifier le localStorage

11 Conclusion

Objectifs

Compétences acquises :
À la fin de ce TP, vous devez être capable de :

- ✓ Créer une application CRUD complète
- ✓ Gérer des données complexes (objets)
- ✓ Implémenter une recherche en temps réel
- ✓ Utiliser le localStorage efficacement
- ✓ Modifier dynamiquement le DOM
- ✓ Gérer les états de l'application (mode édition)
- ✓ Créer une UX fluide et intuitive

Prochaine étape : Nous allons recréer cette application avec Vue.js pour découvrir la puissance des frameworks modernes !

Bon travail !

N'oubliez pas de sauvegarder votre code et de tester toutes les fonctionnalités !