PROJET WEB

Application de Gestion de Tâches

HTML / CSS / JavaScript

BTS SIO - 1ère Année

École Ensitech

Formateur : Abid Hamza

TABLE DES MATIÈRES

1. PRÉSENTATION DU PROJET

Ce projet consiste à créer une application web simple de gestion de tâches (To-Do List) en utilisant uniquement HTML, CSS et JavaScript. Cette application permettra aux utilisateurs de gérer leurs tâches quotidiennes.

Fonctionnalités principales :

- Ajouter de nouvelles tâches
- Visualiser la liste de toutes les tâches
- Marquer une tâche comme terminée
- Supprimer une tâche de la liste

Ce projet est conçu pour être progressif. Vous serez guidé étape par étape avec des rappels et des explications pour comprendre chaque concept.

2. OBJECTIFS PÉDAGOGIQUES

À l'issue de ce projet, vous serez capable de :

- Créer une structure HTML valide
- Utiliser les éléments de formulaire HTML
- Appliquer des styles CSS de base
- Manipuler le DOM avec JavaScript
- Gérer les événements utilisateur
- Stocker des données localement

3. PRÉREQUIS ET RAPPELS

RAPPEL : Qu'est-ce que HTML ?

- HTML (HyperText Markup Language) est le langage de balisage utilisé pour structurer le contenu d'une page web
- Les balises HTML délimitent les éléments : <balise>contenu</balise>
- Les attributs fournissent des informations supplémentaires : <balise attribut="valeur">

- Exemple : <h1 id="titre">Mon titre</h1>

RAPPEL : Qu'est-ce que CSS ?

- CSS (Cascading Style Sheets) permet de mettre en forme les éléments HTML
- Un sélecteur cible un élément : h1 { color: blue; }
- Les propriétés définissent l'apparence : color, background-color, padding, margin, etc.
- On peut cibler par balise, par id (#monId) ou par classe (.maClasse)

RAPPEL : Qu'est-ce que JavaScript ?

- JavaScript est un langage de programmation qui rend les pages web interactives
- Il permet de manipuler le DOM (Document Object Model)
- Le DOM représente la structure HTML comme un arbre d'objets
- On peut sélectionner des éléments avec getElementById(), querySelector(), etc.
- Les événements permettent de réagir aux actions de l'utilisateur (clic, saisie, etc.)

Prérequis techniques :

• Un éditeur de texte (Visual Studio Code recommandé)
• Un navigateur web (Chrome, Firefox, Edge)
• Notions de base en programmation (variables, fonctions)

4. STRUCTURE DU PROJET

Créez un dossier nommé "todo-app" sur votre ordinateur.

GUIDE : Organisation des fichiers

- Dans le dossier todo-app, créez trois fichiers :
- - index.html (structure de la page)
- - style.css (mise en forme)
- - script.js (fonctionnalités)

```
todo-app/
  ├── index.html
  ├── style.css
  └── script.js
```

5. PARTIE 1 : STRUCTURE HTML

Dans cette partie, vous allez créer la structure de base de votre page web. Nous allons utiliser les balises HTML essentielles.

RAPPEL : Structure de base d'un document HTML

- Un document HTML commence toujours par <!DOCTYPE html>
- L'élément <html> contient tout le document
- La section <head> contient les métadonnées (non visibles)
- La section <body> contient le contenu visible
- Les balises doivent être correctement fermées

GUIDE : Créer votre fichier index.html

- Ouvrez votre éditeur de texte
- Créez un nouveau fichier et enregistrez-le sous le nom 'index.html'
- Commencez par la structure de base HTML5

Question : Quelle est la première ligne à écrire dans un fichier HTML5 ?

Voici la structure minimale à créer :

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

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

RAPPEL : Explication des éléments

- <!DOCTYPE html> : déclare que c'est un document HTML5
- <meta charset="UTF-8"> : définit l'encodage des caractères (important pour les accents)
- <meta name="viewport"> : permet l'affichage responsive sur mobile
- <link rel="stylesheet"> : lie le fichier CSS externe
- <script src="script.js"> : lie le fichier JavaScript (à la fin du body)

Maintenant, ajoutez les éléments nécessaires pour votre application :

GUIDE : Éléments à ajouter

- Un formulaire avec un champ de saisie et un bouton
- Une liste pour afficher les tâches
- Une zone pour afficher des messages
- Une section pour les statistiques

Question : Quel élément HTML utilisez-vous pour créer un formulaire ?

Question : Quel attribut HTML permet d'identifier un élément de manière unique ?

Voici un exemple de structure complète (à compléter) :

```
<form id="taskForm">
    <div class="input-group">
        <input type="text" id="taskInput" placeholder="Entrez une
nouvelle tâche..." required>
        <button type="submit">Ajouter</button>
    </div>
</form>
<div id="message"></div>
<ul id="taskList"></ul>
<div class="stats">
    <p>Total : <span id="totalTasks">0</span></p>
    <p>Terminées : <span id="completedTasks">0</span></p>
</div>
```

RAPPEL : Points importants

- L'attribut id permet d'identifier un élément de manière unique
- L'attribut required empêche la soumission d'un formulaire vide
- L'attribut placeholder affiche un texte indicatif
- type="submit" sur le bouton déclenche la soumission du formulaire

Exercice : Créez votre fichier index.html avec tous les éléments nécessaires.

6. PARTIE 2 : MISE EN FORME CSS

Dans cette partie, vous allez styliser votre page web pour la rendre agréable à regarder et facile à utiliser.

RAPPEL : Rappels CSS

- Un style CSS se compose d'un sélecteur et de propriétés
- Syntaxe : sélecteur { propriété: valeur; }
- On peut cibler par balise (h1), par id (#monId) ou par classe (.maClasse)
- Les propriétés courantes : color, background-color, padding, margin, border, font-size

GUIDE : Créer votre fichier style.css

- Créez un nouveau fichier nommé 'style.css'
- Commencez par réinitialiser les styles par défaut
- Définissez les styles de base pour le body
- Stylez le conteneur principal

Question : Pourquoi réinitialiser les marges et paddings par défaut ?

Commencez par ces styles de base :

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
    padding: 20px;
}
.container {
    max-width: 600px;
    margin: 0 auto;
    background-color: white;
    padding: 30px;
    border-radius: 5px;
}
```

RAPPEL : Explication

- * est le sélecteur universel (tous les éléments)

- box-sizing: border-box inclut padding et border dans la largeur totale
- max-width limite la largeur maximale
- margin: 0 auto centre un élément horizontalement

Maintenant, stylez le formulaire et les éléments de saisie.

GUIDE : Styles à ajouter

- Stylez le groupe input + bouton (utilisez display: flex)
- Définissez l'apparence du champ de saisie
- Stylez le bouton avec une couleur de fond
- Ajoutez un effet au survol (:hover)

Question : Quelle propriété CSS permet de créer un conteneur flexible ?

Exemple de styles pour le formulaire :

```
.input-group {
    display: flex;
    gap: 10px;
    margin-bottom: 20px;
}
#taskInput {
    flex: 1;
    padding: 12px;
    border: 1px solid #ddd;
    border-radius: 4px;
}
button {
    padding: 12px 24px;
    background-color: #4CAF50;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}
button:hover {
    background-color: #45a049;
}
```

RAPPEL : Flexbox

- display: flex crée un conteneur flexible
- flex: 1 permet à un élément de prendre tout l'espace disponible
- gap ajoute un espacement entre les éléments flexibles
- :hover est une pseudo-classe pour l'effet au survol

Stylez maintenant la liste des tâches.

GUIDE : Styles pour la liste

- Supprimez les puces de la liste (list-style)
- Stylez chaque élément de la liste
- Ajoutez un style pour les tâches terminées (classe .completed)
- Créez des styles pour les boutons dans la liste

Exercice : Créez les styles pour afficher les tâches de manière claire.

7. PARTIE 3 : FONCTIONNALITÉS JAVASCRIPT

Dans cette partie, vous allez rendre votre application interactive en ajoutant les fonctionnalités avec JavaScript.

RAPPEL : Rappels JavaScript

- Une variable se déclare avec let ou const
- Une fonction se déclare avec function nomFonction() { }
- getElementById('id') permet de sélectionner un élément par son id
- addEventListener() permet d'écouter un événement
- localStorage permet de stocker des données dans le navigateur

GUIDE : Créer votre fichier script.js

- Créez un nouveau fichier nommé 'script.js'
- Commencez par déclarer un tableau pour stocker les tâches
- Créez les fonctions nécessaires une par une

Question : Comment déclarez-vous un tableau vide en JavaScript ?

Structure de base :

```
let tasks = [];
function loadTasks() {
    // À compléter
}
function saveTasks() {
    // À compléter
}
function displayTasks() {
    // À compléter
}
function addTask(taskText) {
    // À compléter
}
function completeTask(index) {
    // À compléter
}
function deleteTask(index) {
    // À compléter
}
```

RAPPEL : localStorage

- localStorage.setItem('cle', valeur) sauvegarde une donnée
- localStorage.getItem('cle') récupère une donnée
- JSON.stringify() convertit un objet en chaîne de caractères
- JSON.parse() convertit une chaîne en objet

Commençons par la fonction de sauvegarde.

GUIDE : Fonction saveTasks()

- Utilisez localStorage.setItem()
- Convertissez le tableau en JSON avec JSON.stringify()
- La clé peut être 'tasks'

Question : Comment convertissez-vous un tableau JavaScript en chaîne de caractères ?

Exemple de solution :

```
function saveTasks() {
    localStorage.setItem("tasks", JSON.stringify(tasks));
}
```

Maintenant, créez la fonction de chargement.

GUIDE : Fonction loadTasks()

- Récupérez les données avec localStorage.getItem()
- Vérifiez si des données existent
- Convertissez la chaîne en tableau avec JSON.parse()

Exercice : Écrivez la fonction loadTasks() qui charge les tâches sauvegardées.

Passons à l'ajout d'une tâche.

GUIDE : Fonction addTask()

- Vérifiez que le texte n'est pas vide (utilisez trim())
- Créez un objet tâche avec id, text et completed
- Ajoutez la tâche au tableau avec push()
- Sauvegardez et réaffichez

Question : Comment générez-vous un identifiant unique pour chaque tâche ?

Question : Quelle methode permet d'ajouter un element a la fin d'un tableau ?

Structure de la fonction (à compléter) :

```
function addTask(taskText) {
    if (taskText.trim() === "") {
        // Afficher un message d erreur
        return;
    }
    const task = {
        id: Date.now(),
        text: taskText.trim(),
        completed: false
    };
    tasks.push(task);
    saveTasks();
    displayTasks();
}
```

RAPPEL : Objets JavaScript

- Un objet se crée avec { propriete: valeur }
- On accède aux propriétés avec point : objet.propriete
- Date.now() retourne le nombre de millisecondes depuis 1970 (identifiant unique)
- trim() supprime les espaces au début et à la fin d'une chaîne

Maintenant, creons la fonction d affichage.

GUIDE : Fonction displayTasks()

- Sélectionnez l'élément ul avec getElementById()
- Videz son contenu avec innerHTML = ""
- Parcourez le tableau avec forEach()
- Pour chaque tâche, créez un élément li

- Ajoutez les boutons Terminer et Supprimer
- Ajoutez l'élément li à la liste

Question : Comment créez-vous un nouvel élément HTML en JavaScript ?

Question : Comment ajoutez-vous un élément enfant à un autre élément ?

Exemple de structure (à compléter) :

```
function displayTasks() {
    const taskList = document.getElementById("taskList");
    taskList.innerHTML = "";
    tasks.forEach(function(task, index) {
        const li = document.createElement("li");
        if (task.completed) {
            li.classList.add("completed");
        }
        const taskText = document.createElement("span");
        taskText.textContent = task.text;
        // Créer les boutons et les ajouter
        taskList.appendChild(li);
    });
}
```

RAPPEL : Manipulation du DOM

- createElement('balise') crée un nouvel élément
- textContent définit le texte d'un élément
- classList.add() ajoute une classe CSS
- appendChild() ajoute un élément enfant
- forEach() parcourt chaque élément d'un tableau

Créons les fonctions pour terminer et supprimer une tâche.

GUIDE : Fonction completeTask()

- Récupérez la tâche à l'index donné
- Inversez l'état completed (!)
- Sauvegardez et réaffichez

Exercice : Ecrivez la fonction completeTask(index) qui inverse l etat d une tache.

GUIDE : Fonction deleteTask()

- Demandez confirmation avec confirm()
- Supprimez la tâche avec splice(index, 1)
- Sauvegardez et réaffichez

Question : Quelle methode permet de supprimer un element d'un tableau a un index donne ?

Enfin, ajoutez les gestionnaires d evenements.

GUIDE : Gestionnaire d'événement pour le formulaire

- Sélectionnez le formulaire avec getElementById()
- Ajoutez un écouteur d'événement 'submit'
- Empêchez le comportement par défaut avec preventDefault()
- Récupérez la valeur du champ input
- Appelez addTask() avec cette valeur
- Videz le champ

Question : Pourquoi utilisez-vous preventDefault() sur l'evenement submit ?

Exemple :

```
document.getElementById("taskForm").addEventListener("submit",
function(e) {
    e.preventDefault();
    const taskInput = document.getElementById("taskInput");
    addTask(taskInput.value);
    taskInput.value = "";
});
```

Ajoutez l initialisation au chargement de la page.

GUIDE : Initialisation

- Utilisez addEventListener('DOMContentLoaded')

- Chargez les tâches avec loadTasks()
- Affichez les tâches avec displayTasks()

Question : Pourquoi utilisez-vous DOMContentLoaded plutot que de placer le script en haut ?

Exercice : Complétez votre fichier script.js avec toutes les fonctionnalités.

8. PARTIE 4 : TESTS ET AMÉLIORATIONS

Testez maintenant votre application pour vérifier que tout fonctionne correctement.

GUIDE : Tests à effectuer

- Ajoutez plusieurs tâches
- Marquez certaines tâches comme terminées
- Supprimez des tâches
- Fermez et rouvrez le navigateur (les tâches doivent être conservées)
- Testez avec des tâches très longues
- Testez avec des tâches vides

Si quelque chose ne fonctionne pas :

- • Ouvrez la console du navigateur (F12)
- • Vérifiez les erreurs affichées
- • Vérifiez que tous les fichiers sont bien liés
- • Vérifiez l'orthographe des id et des noms de fonctions

Améliorations possibles :

- • Ajouter une fonctionnalité de filtrage (toutes / actives / terminées)
- • Permettre l'édition d'une tâche existante
- • Ajouter une date à chaque tâche
- • Améliorer le design responsive pour mobile

9. EXERCICES COMPLÉMENTAIRES

EXERCICE : Exercice 1 : Ajouter une date

- Modifiez la structure d'une tâche pour inclure une date de création
- Affichez cette date à côté de chaque tâche
- Utilisez l'objet Date de JavaScript

EXERCICE : Exercice 2 : Éditer une tâche

- Ajoutez un bouton 'Modifier' à chaque tâche
- Quand on clique sur 'Modifier', transformez le texte en champ input
- Permettez la modification et la sauvegarde

EXERCICE : Exercice 3 : Filtrer les tâches

- Ajoutez trois boutons : Toutes, Actives, Terminées
- Créez une fonction filterTasks() qui filtre selon le critère
- Modifiez displayTasks() pour n'afficher que les tâches filtrées

EXERCICE : Exercice 4 : Validation améliorée

- Empêchez l'ajout de tâches en double
- Limitez la longueur du texte à 100 caractères
- Affichez un compteur de caractères restants

10. ANNEXES

10.1. Ressources utiles

Documentation officielle :

- • HTML : https://developer.mozilla.org/fr/docs/Web/HTML
- • CSS : https://developer.mozilla.org/fr/docs/Web/CSS
- • JavaScript : https://developer.mozilla.org/fr/docs/Web/JavaScript

10.2. Glossaire

- • DOM : représentation structurée du document HTML
- • Événement : action de l'utilisateur (clic, saisie, etc.)
- • localStorage : stockage local dans le navigateur
- • JSON : format de données texte
- • Responsive : adaptation à différentes tailles d'écran

10.3. Checklist de fin de projet

Vérifiez que votre projet contient :

- • Un fichier index.html valide et bien structuré
- • Un fichier style.css avec tous les styles nécessaires
- • Un fichier script.js avec toutes les fonctionnalités
- • Le code est commenté et lisible
- • L'application fonctionne correctement
- • Les tâches sont sauvegardées et rechargées correctement

CONCLUSION

Félicitations ! Vous avez créé votre première application web complète. Vous avez appris les bases du développement web front-end.

Continuez à pratiquer et à expérimenter. Bon développement !

Formateur : Abid Hamza

École Ensitech - BTS SIO 1ère Année