

Mini-Projet : Gestion de Vente en JavaScript

Titre : Développement d'une Application de Gestion de Vente

Contexte :

Dans le cadre de l'apprentissage des technologies web (HTML, CSS, et JavaScript), vous allez développer une application simple de gestion de vente. Cette application permettra à un utilisateur de gérer un panier d'achat en ajoutant et supprimant des produits, tout en calculant automatiquement le total de la commande. Ce projet vous permettra de mettre en pratique vos compétences en programmation web et de comprendre les bases de la gestion d'état dans une application.

Objectif Général :

Développer une application web dynamique et interactive pour gérer des ventes en utilisant principalement JavaScript.

Objectifs Spécifiques :

1. Créer une interface utilisateur simple et ergonomique avec HTML et CSS.
 2. Manipuler le DOM (Document Object Model) avec JavaScript pour ajouter des produits dynamiquement.
 3. Gérer un panier d'achat (ajout, suppression, calcul du total) en temps réel.
 4. Appliquer les bonnes pratiques de programmation et structuration de code.
-

Cahier des Charges :

Fonctionnalités à implémenter :

1. **Affichage des produits disponibles :**
 - Une liste de produits sera affichée dynamiquement.

- Chaque produit doit inclure un nom, un prix, et un bouton "Ajouter au panier".
- 2. **Gestion du panier :**
 - L'utilisateur pourra ajouter un produit au panier en cliquant sur un bouton.
 - Le panier doit afficher le nom, le prix de chaque produit, et un bouton "Supprimer" pour retirer un produit.
 - Le total de la commande doit être calculé et mis à jour automatiquement.
- 3. **Interface utilisateur :**
 - Une présentation claire et attrayante avec des styles CSS.
 - Une interface intuitive pour que l'utilisateur puisse interagir facilement avec l'application.

Contraintes techniques :

- Utilisez **HTML**, **CSS**, et **JavaScript** uniquement. Aucun framework ou bibliothèque externe n'est autorisé.
- Le code JavaScript doit être bien organisé et commenté.
- L'application doit être fonctionnelle dans un navigateur moderne (Google Chrome, Firefox, etc.).

Indications Techniques :

Voici un exemple simple pour démontrer comment manipuler des tableaux en JavaScript, notamment en ajoutant et en supprimant des éléments :

```
// Création d'un tableau initial
let fruits = ["Pomme", "Banane", "Orange"];
console.log("Tableau initial :", fruits);

// **Ajouter un élément** au tableau
// 1. Utiliser `push()` pour ajouter un élément à la fin
fruits.push("Mangue");
console.log("Après ajout avec push :", fruits);

// 2. Utiliser `unshift()` pour ajouter un élément au début
fruits.unshift("Fraise");
console.log("Après ajout avec unshift :", fruits);

// **Supprimer un élément** du tableau
// 1. Utiliser `pop()` pour supprimer le dernier élément
```

```
// Ajouter un élément à l'index 1
fruits.splice(1, 0, "Ananas"); // (index de départ, 0 suppression, éléments à ajouter)
console.log("Après ajout avec splice :", fruits);
```

```

let fruitRetire = fruits.pop();
console.log("Après suppression avec pop :", fruits);
console.log("Fruit retiré :", fruitRetire);

// 2. Utiliser `shift()` pour supprimer le premier élément
fruitRetire = fruits.shift();
console.log("Après suppression avec shift :", fruits);
console.log("Fruit retiré :", fruitRetire);

// 3. Utiliser `splice()` pour supprimer un élément à un index spécifique
// Supposons que nous voulons supprimer le deuxième élément (index 1)
fruitRetire = fruits.splice(1, 1); // (index de départ, nombre d'éléments à supprimer)
console.log("Après suppression avec splice :", fruits);
console.log("Fruit retiré avec splice :", fruitRetire);

// **Ajouter un élément avec `splice()`**
// Ajouter un élément à l'index 1
fruits.splice(1, 0, "Ananas"); // (index de départ, 0 suppression, élément à ajouter)
console.log("Après ajout avec splice :", fruits);

```

Vous pouvez utiliser un tableau à deux dimensions pour stocker les produits :

```

listProduits = [
    {"pomme", 12.5},
    {"gloria", 45},
    {"ordinateurs", 75},
    ...
]

```

On a le nom du produit et son prix. Vous pouvez stocker dans *listProduits* au temps de produits que vous voulez.

Explications :

- ❑ **push()** : Ajoute un élément à la fin du tableau.
- ❑ **unshift()** : Ajoute un élément au début du tableau.
- ❑ **pop()** : Supprime et retourne le dernier élément.
- ❑ **shift()** : Supprime et retourne le premier élément.

- **splice()** : Permet d'ajouter, de supprimer ou de remplacer des éléments à une position spécifique.

Livrables :

1. Les fichiers sources du projet :
 - index.html: le fichier principal contenant la structure de la page.
 - styles.css: le fichier de styles pour personnaliser l'apparence.
 - script.js: le fichier JavaScript pour gérer la logique de l'application.
 2. Une documentation (fichier texte ou PDF) expliquant :
 - Les fonctionnalités implémentées.
 - La structure du code.
 - Les difficultés rencontrées et les solutions trouvées.
-

Évaluation :

- **Qualité de l'interface utilisateur (20%)** : Respect des règles de design et accessibilité.
- **Fonctionnalité et interactivité (50%)** : L'application doit être pleinement fonctionnelle et répondre aux exigences.
- **Organisation et clarté du code (20%)** : Code propre, bien structuré et documenté.
- **Documentation (10%)** : Explications claires des choix techniques et des fonctionnalités.

Conseils pour réussir :

1. Testez régulièrement votre application au fur et à mesure du développement.
2. Respectez une organisation claire du code en séparant structure (HTML), style (CSS), et logique (JavaScript).
3. Pensez à anticiper les erreurs (exemple : éviter d'ajouter des produits inexistant au panier).