Exercice : To-Do List Interactive

Objectif de l'exercice

- Créer une application web simple qui permet de gérer une liste de tâches (to-do list).
- Travailler sur la manipulation du DOM et la gestion des événements en JavaScript.
- Organiser le code en plusieurs fonctions pour structurer l'application.

Description de l'exercice

Les étudiants devront concevoir une interface qui permet de :

- **Ajouter une tâche**
  Saisir une nouvelle tâche via un champ de texte et l'ajouter à une liste visible.
- **Marquer une tâche comme terminée**
  Cliquer sur une tâche pour la signaler comme complétée, par exemple en modifiant son apparence (texte barré, changement de couleur, etc.).
- **Supprimer une tâche**
  Disposer d'un bouton associé à chaque tâche pour pouvoir la retirer de la liste.

Compétences travaillées

- **Manipulation du DOM :**
  - Sélection d'éléments HTML.
  - Création et suppression dynamique d'éléments.
  - Ajout d'éléments enfants dans le DOM.
- **Gestion d'événements :**
  - Utilisation de `addEventListener` pour gérer les clics.
  - Gestion des événements de formulaire et de saisie (par exemple, pour ajouter une tâche via le bouton ou la touche "Entrée").
- **Fonctions et organisation du code :**
  - Définir des fonctions pour ajouter, supprimer et modifier des éléments.
  - Structurer le code pour une meilleure lisibilité et maintenabilité.
- **Validation et interactivité :**
  - Vérifier que l'entrée utilisateur n'est pas vide avant d'ajouter une tâche.
  - Donner un retour visuel immédiat à l'utilisateur en fonction de ses actions (ajout, suppression, marquage comme terminé).

Consignes de réalisation

1. **Interface utilisateur :**

    - Créer un champ de saisie pour entrer le texte d'une nouvelle tâche.
    - Prévoir un bouton pour ajouter la tâche saisie à la liste.
    - Afficher la liste des tâches en dessous du champ de saisie.
2. **Ajout d'une tâche :**
    - Lorsqu'une tâche est ajoutée, l'élément correspondant doit apparaître dans la liste.
    - Si le champ de saisie est vide, prévenir l'utilisateur avec un message d'alerte.
3. **Gestion des tâches :**
    - Permettre à l'utilisateur de cliquer sur une tâche pour la marquer comme terminée (modifier son style pour indiquer son état).
    - Chaque tâche doit comporter un bouton ou un mécanisme pour être supprimée de la liste.
4. **Retour utilisateur :**
    - Assurer une mise à jour visuelle claire pour chaque action (ajout, suppression, modification de l'état de la tâche).
    - Eventuellement, ajouter des styles CSS pour différencier les tâches terminées des tâches en cours.

Points clés à retenir

- **Structuration du code :**
  Organiser le script en fonctions dédiées pour chaque action (ajout, suppression, marquage).
- **Utilisation des événements :**
  Associer correctement les événements aux éléments (clic sur bouton, touche "Entrée" pour le champ de saisie).
- **Manipulation du DOM :**
  Utiliser les méthodes de création, insertion et suppression d’éléments pour mettre à jour l’interface utilisateur en temps réel.
- **Validation de l'entrée utilisateur :**
  S’assurer que l’utilisateur ne peut pas ajouter une tâche vide et gérer les erreurs de saisie de manière appropriée.

Cet exercice offre une base solide pour comprendre et pratiquer les interactions de base en JavaScript avec le DOM et constitue une bonne préparation pour des projets plus complexes.