TRAINING REACT !

EXERCICE - Todo List Interactive

Le but de l'exercice est de se familiariser avec la gestion d'état en React (useState) et la manipulation de listes dynamiques ainsi que des formulaires. Pour cela, on va créer une application de gestion de tâches (Todo List) avec les fonctionnalités d'ajout, de marquage comme complétée et de suppression.

L'application réalisée devra permettre :

- D'ajouter une nouvelle tâche
- De marquer une tâche comme complétée/non complétée
- De supprimer une tâche
- D'afficher le nombre de tâches restantes

Question 1

- Créez un nouveau projet React nommé `todo-app` .
- Vous êtes libre d'utiliser le framework CSS de votre choix (TailwindCSS, Bootstrap, Material-UI) ou de créer vos propres styles CSS/CSS Modules.
- Dans le fichier `App.js` , créez une structure HTML de base avec un conteneur pour votre application.

Question 2

Ajoutez un sous-dossier `components` au dossier `src` . Dans `components` , créez un composant nommé `TodoWrapper` qui pour le moment va :

- Afficher un titre avec le texte "Ma Todo List"
- Renvoyer une structure de base pour accueillir les futurs composants

Effectuez le rendu de ce composant dans `App` . Appliquez le style CSS de votre choix pour rendre l'interface agréable.

Question 3

Dans le dossier `components` , créez un composant nommé `TodoForm` qui va gérer l'ajout de nouvelles tâches. Ce composant doit :

- Contenir un formulaire avec :
  - Un input de type texte pour saisir la tâche
  - Un bouton pour soumettre le formulaire
- Recevoir une prop `onAddTodo` qui sera appelée lors de la soumission du formulaire
- Utiliser `useState` pour gérer la valeur de l'input
- Empêcher l'ajout de tâches vides (validation)
- Vider l'input après l'ajout d'une tâche

Intégrez ce composant dans `TodoWrapper` et vérifiez que la soumission du formulaire affiche un `console.log` de la tâche saisie.

Question 4

Créez un composant nommé `TodoItem` qui représente une tâche individuelle. Ce composant doit :

- Recevoir les props suivantes :
  - `todo` : un objet contenant les informations de la tâche (vous définirez la structure de cet objet)
  - `onToggle` : fonction appelée pour changer l'état de la tâche
  - `onDelete` : fonction appelée pour supprimer la tâche
- Afficher :
  - Le texte de la tâche
  - Un élément interactif (checkbox, bouton, icône , texte barré, ...) pour marquer la tâche comme complétée
  - Un élément interactif pour supprimer la tâche
- Appliquer un style visuel différent aux tâches complétées (par exemple : texte barré, couleur différente, opacité réduite...)

Testez ce composant en créant une instance manuelle dans `TodoWrapper` avec des données fictives.

Question 5

Modifiez le composant `TodoWrapper` pour gérer l'état de la liste de tâches avec `useState` . Le composant doit maintenant :

- Initialiser un état `todos` avec un tableau vide
- Créer une fonction `addTodo` qui :
  - Reçoit le texte de la nouvelle tâche
  - Crée un nouvel objet représentant la tâche (pensez aux propriétés nécessaires : id, texte, état de complétion...)
  - Ajoute cette tâche au tableau `todos`
  - Utilise un système de génération d'identifiants uniques(UUID) pour chaque tâche (par exemple, la bibliothèque `nanoid` ou `uuid`)

Exemple avec nanoid :

> *installation :* `npm install nanoid`

```
import { nanoid } from 'nanoid';

const newTodo = {
    id: nanoid(),
    text: "Ma nouvelle tâche",
    completed: false
};
```

- Passer la fonction `addTodo` au composant `TodoForm` via la prop `onAddTodo`
- Afficher la liste des tâches en utilisant la méthode `.map()` pour rendre un composant `TodoItem` pour chaque tâche du tableau
- N'oubliez pas d'ajouter une `key` unique à chaque `TodoItem` rendu

Vérifiez que vous pouvez ajouter plusieurs tâches et qu'elles s'affichent correctement dans la liste.

Question 6

Implémentez la fonctionnalité de marquage comme complétée. Dans `TodoWrapper` :

- Créez une fonction `toggleTodo` qui :
  - Reçoit l'identifiant de la tâche à modifier
  - Trouve la tâche correspondante dans le tableau
  - Inverse son état de complétion
  - Met à jour l'état `todos` avec la modification
- Passez cette fonction au composant `TodoItem` via la prop `onToggle`
- Dans `TodoItem` , appelez cette fonction avec l'identifiant de la tâche lors de l'interaction avec l'élément de marquage

Vérifiez que l'interaction avec une tâche change bien son apparence visuelle pour indiquer qu'elle est complétée ou non.

Question 7

Implémentez la fonctionnalité de suppression. Dans `TodoWrapper` :

- Créez une fonction `deleteTodo` qui :
  - Reçoit l'identifiant de la tâche à supprimer
  - Filtre le tableau `todos` pour retirer la tâche correspondante
  - Met à jour l'état avec le nouveau tableau
- Passez cette fonction au composant `TodoItem` via la prop `onDelete`
- Dans `TodoItem` , appelez cette fonction avec l'identifiant de la tâche lors du clic sur l'élément de suppression

Vérifiez que cliquer sur l'élément de suppression retire bien la tâche de la liste.

Question 8

Créez un composant `TodoStats` qui affiche des statistiques sur les tâches. Ce composant doit :

- Recevoir en prop le tableau de tâches
- Calculer et afficher :
  - Le nombre total de tâches
  - Le nombre de tâches complétées
  - Le nombre de tâches restantes (non complétées)

Intégrez ce composant dans `TodoWrapper` à l'emplacement de votre choix.