Exercice d'évaluation : Gestion d'un compte bancaire

Objectif de l'évaluation

Dans cet exercice, vous allez créer une interface de gestion de comptes bancaires en JavaScript. Vous utiliserez les concepts suivants :

- La création de classes et d'objets pour modéliser un compte bancaire.
- La manipulation du DOM pour afficher dynamiquement des formulaires et des messages.
- La gestion des événements pour réagir aux actions de l'utilisateur (clics sur boutons).
- La gestion d'un tableau pour stocker et rechercher des comptes.
- La validation des entrées utilisateur et l'affichage de retours (messages d'erreur ou de succès).

Description de l'exercice

Vous devez implémenter les fonctionnalités suivantes :

1. **Création d'un compte bancaire :**
    - Le compte doit être représenté par une instance d'une classe `CompteBancaire` (avec les propriétés : propriétaire, numéro de compte et solde).
    - Lors de la création d'un compte, vérifiez que les identifiants (propriétaire et numéro de compte) ne sont pas déjà utilisés dans le tableau des comptes.
    - Si les identifiants sont uniques, ajoutez le compte au tableau et affichez un message de confirmation.
2. **Effectuer un dépôt :**
    - Permettez à l'utilisateur de saisir un montant à déposer sur un compte existant.
    - Vérifiez que le montant saisi est bien un nombre.
    - Recherchez le compte correspondant aux identifiants fournis.
    - Si le compte est trouvé, ajoutez le montant au solde du compte et affichez un message confirmant la transaction ainsi que le nouveau solde.
    - Si le compte n'est pas trouvé, affichez un message d'erreur.
3. **Navigation et affichage :**
    - Créez des formulaires dynamiques pour la création de compte et pour le dépôt.
    - Mettez en place des fonctions pour nettoyer l'interface (supprimer les formulaires ou messages précédents) afin d'éviter l'encombrement de la page.
    - Utilisez des messages de réponse pour informer l'utilisateur de l'état de ses actions (succès ou erreur).

Consignes techniques

- **Manipulation du DOM :**
  - Créez et insérez dynamiquement des éléments (formulaires, messages de réponse) dans la page.
  - Supprimez les anciens éléments affichés avant d'en ajouter de nouveaux.
- **Gestion des événements :**
  - Associez des écouteurs d'événements aux boutons pour déclencher la création des formulaires et la validation des actions (création de compte, dépôt).
- **Validation des entrées :**
  - Vérifiez que les champs de saisie ne sont pas vides.
  - Vérifiez que le montant saisi pour le dépôt est un nombre valide.
  - Assurez-vous que les identifiants du compte (propriétaire et numéro de compte) sont uniques lors de la création.
- **Recherche dans le tableau de comptes :**
  - Pour rechercher un compte existant dans le tableau, vous pouvez utiliser la méthode `find` sur le tableau `comptes`.
  - **Aide sur l'utilisation de `find` :**
    - La méthode `find` permet de rechercher dans un tableau le premier élément qui satisfait une condition donnée. Par exemple, pour trouver un compte avec un numéro de compte spécifique, vous pourriez écrire :

      ```
      const compte = users.find(user => user.name === name);
      ```

- **Affichage des messages :**
  - Créez des fonctions pour générer et afficher des messages de réponse (succès ou erreur) en fonction des actions réalisées par l'utilisateur.
  - Les messages doivent clairement indiquer le résultat de l'opération (compte créé, dépôt effectué, erreur de saisie, etc.).

Consignes de réalisation

1. **Création du compte bancaire :**
   - Implémentez une fonction qui vérifie que les champs de saisie (nom et numéro de compte) sont correctement remplis.
   - Vérifiez que les identifiants ne sont pas déjà utilisés dans le tableau des comptes. Si c'est le cas, affichez un message d'erreur.
   - Sinon, créez une instance de `CompteBancaire` avec un solde initial de 0, ajoutez-la au tableau, et affichez un message de confirmation.
2. **Dépôt sur un compte existant :**
   - Implémentez une fonction qui vérifie que le montant saisi est un nombre.
   - Recherchez dans le tableau le compte correspondant aux identifiants fournis.

    - Si le compte existe, mettez à jour son solde en ajoutant le montant et affichez un message confirmant la transaction et le nouveau solde.
    - Si le compte n'est pas trouvé, affichez un message d'erreur indiquant que les identifiants sont incorrects.
3. **Navigation et nettoyage de l'interface :**
    - Mettez en place des fonctions pour créer et supprimer dynamiquement les formulaires et les messages affichés.
    - Ajoutez des boutons pour déclencher l'affichage des formulaires de création de compte et de dépôt, ainsi qu'un bouton pour fermer et nettoyer l'interface.

Bonnes pratiques

- Organisez votre code en fonctions claires et bien nommées pour chaque fonctionnalité (création de compte, dépôt, affichage de messages, etc.).
- Testez minutieusement chaque fonctionnalité pour vous assurer que l'interface réagit correctement aux actions de l'utilisateur.
- Commentez votre code afin de faciliter la compréhension et la maintenance.
- Utilisez des messages d'erreur et de succès explicites pour améliorer l'expérience utilisateur.

Bonne chance dans la réalisation de cet exercice !