ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR
Membre de HONORIS UNITED UNIVERSITIES

TP: la Programmation Asynchrone en JavaScript

Nouhaila Moussammi

n.moussammi@emsi.ma

Objectifs

1. Apprendre à gérer des opérations asynchrones avec des fonctions de rappel (callbacks) et l'utilisation des promesses pour une gestion plus élégante des opérations asynchrones.

2. Apprendre à utiliser les structures try, catch, throw, et finally pour capturer et gérer les erreurs de manière efficace.

Exercice 1: Simulation d'un système de validation d'utilisateur

Développer une application web permettant de gérer des administrateurs. Cette application doit inclure un formulaire pour enregistrer de nouveaux administrateurs et un formulaire de connexion pour permettre aux administrateurs d'accéder à l'application avec leurs identifiants.

L'exercice est divisé en trois parties : **Partie 1 : Gestion des administrateurs avec tableau, fonction classique et un Callback**

1. **Création d'un tableau d'administrateurs**
   Créez un tableau nommé admins qui contiendra les administrateurs sous forme d'objets. Chaque admin- istrateur aura deux propriétés : nom (le nom d'utilisateur) et password (le mot de passe).

2. **Fonction pour ajouter un administrateur**
Créez une fonction classique (une fonction normale, non fléchée) qui permettra d'ajouter un administrateur au tableau. Cette fonction doit :

- * Vérifier que le nom d'utilisateur et le mot de passe sont valides (non vides).

- * Vérifier que l'administrateur n'existe pas déjà dans le tableau (en vérifiant le nom d'utilisateur).
- * Afficher un message de succès si l'administrateur est bien ajouté.
- * Après avoir affiché le message de succès, réinitialisez ce message après 3 secondes pour une

meilleure expérience utilisateur.

3. **Fonction validerUtilisateur**
Créez une fonction validerUtilisateur qui prend trois arguments :

* user : le nom d'utilisateur,
* pass : le mot de passe,
* callback : une fonction de rappel pour afficher le message de connexion réussie.

Cette fonction valide les informations de connexion de l'administrateur. Si l'administrateur est trouvé

dans le tableau avec les bons identifiants, appelez le callback pour afficher un message de succès.
4. **Connexion**

Créez une fonction connect qui sera appelée lors du clic sur le bouton de connexion. Cette fonction doit appeler validerUtilisateur pour vérifier les identifiants de l'utilisateur, et afficher le résultat du callback dans un élément .

Partie 2 : Intégration des Promesses

1. **Modification de la fonction validerUtilisateur pour utiliser des promesses**
Modifiez la fonction validerUtilisateur pour qu'elle retourne une promesse. Cette promesse doit se comporter ainsi :

* Si un administrateur est trouvé avec les bons identifiants, la promesse doit être résolue (resolve) avec un message de succès.

* Siaucunadministrateurn'esttrouvé,lapromessedoitêtrerejetée(reject)avecunmessaged'erreur.

2. **Utilisation des promesses dans la fonction connect**
Dans la fonction connect, lorsque l'utilisateur clique sur le bouton "Se connecter", appelez validerUtilisateur. Utilisez les méthodes .then() et .catch() pour gérer respectivement :

* Le cas où la promesse est résolue (connexion réussie).
* Le cas où la promesse est rejetée (identifiants incorrects).

Affichez le message de bienvenue ou le message d'erreur dans l'élément .

Partie 3 : Gestion des erreurs avec try...catch

1. **Intégration de try...catch avec les promesses**
Modifiez la fonction connect pour entourer l'appel à validerUtilisateur avec un bloc try...catch. Cela permet de capturer des erreurs imprévues pendant l'exécution de la fonction de validation.

* Le bloc try doit contenir l'appel à la fonction validerUtilisateur.
* Le bloc catch doit capturer toute erreur pouvant survenir durant l'exécution, comme une erreur de

syntaxe ou une exception inattendue, et afficher un message générique d'erreur.