

TP : Les Fonctions Avancées et la Programmation Asynchrone en JavaScript

Nouhaila Moussammi

n.moussammi@emsi.ma

Objectifs

1. Comprendre la différence entre les fonctions classiques, les fonctions fléchées et les fonctions anonymes.
2. 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.
3. Apprendre à utiliser les structures try, catch, throw, et finally pour capturer et gérer les erreurs de manière efficace.

Exercice 1: Trier une Liste de Mots

Créez un petit programme qui permet à l'utilisateur de saisir une série de mots, d'ajouter ces mots à un tableau, et de trier ce tableau en fonction de la longueur des mots (la fonction doit être fléchée).

Liste des Mots Triée par Longueur

Liste des Mots Triée par Longueur

* Oui
* Chat
* Panier
* Longueur
* Exercices
* Placeholder
* Hello World
* Fatima-Zahra

```

<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title> Trier une Liste de Mots</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      text-align: center;
    }
    #display {
      font-size: 3em;
      margin: 20px;
    }
    button {
      padding: 10px 20px;
      margin: 5px;
      font-size: 1em;
    }
    input {
      margin: 5px;
      padding: 10px 20px;
    }
  </style>
</head>
<body>

  <h1>Liste des Mots Triée par Longueur</h1>
  <input type="text" id="mot" placeholder="Enter une mot"> <br><br>
  <button onclick="addWord()">Ajouter</button>
  <button onclick="displayWords()">Afficher</button>

  <button onclick="sortWord()">Trier par Longueur</button>

  <h3 class="display" id="wordList"></h3>

  <script>
    // Creation du tableau
    const words = [];

    function addWord(){
      let word = document.getElementById("mot").value;
      if (word === "") {
        alert("Veuillez entrer un mot valide.");
        return;
      }
      words.push(word);
      document.getElementById("mot").value = '';
    }

    // Fonction pour afficher la liste des mots
    function displayWords() {
      let wordList = '';
      words.forEach(function(word) {
        wordList += `* ${word} <br>`;
      });
      document.getElementById("wordList").innerHTML = wordList;
    }
  </script>

```

```

// Fonction fléchée pour trier les mot par longueur
const sortWord = () => {
  // Trier par la longueur du nom (du plus court au plus long)
  words.sort((a, b) => a.length - b.length);

  // Afficher la liste triée
  displayWords();
};

</script>

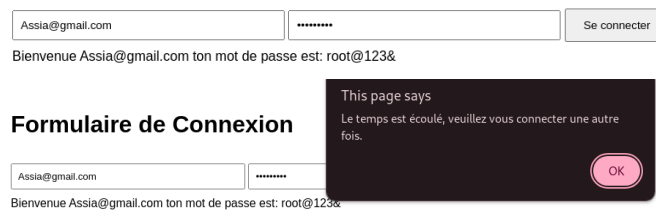
</body>
</html>

```

Exercice 2 : Formulaire de Connexion avec Temporisation

Créez une page web interactive qui simule un formulaire de connexion. L'utilisateur devra entrer son nom d'utilisateur et son mot de passe, puis un message affichant ces informations sera montré (La fonction d'affichage qui prend deux arguments (le nom d'utilisateur et le mot de passe) doit être écrite sous forme de fonction fléchée). Après 10 secondes, un message d'alerte s'affichera et les champs de saisie devront être réinitialisés.

Formulaire de Connexion



```

<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Formulaire de Connexion avec Temporisation</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      padding: 20px;
    }
    input {
      margin-bottom: 10px;
      padding: 8px;
      width: 100%;
      max-width: 300px;
    }
    button {
      padding: 10px 20px;
      margin-top: 10px;
    }
  </style>
</head>
<body>

```

```
<h1>Formulaire de Connexion</h1>
<input type="text" id="username" placeholder="Nom d'utilisateur" required>
<input type="password" id="password" placeholder="Mot de passe" required>
<button type="submit" onclick="connect()">Se connecter</button><br>

<div id="welcome"></div>

<script>
  const displayWelcome = (user,password) => {
    return document.getElementById("welcome").innerHTML = `Bienvenue ${user}
      ton mot de passe est: ${password}`;
  }

  function connect(){
    let user = document.getElementById("username").value;
    let pass = document.getElementById("password").value;

    if (user === "" || pass === "") {
      alert("Veuillez entrer un Nom d'utilisateur et Mot de passe valide.");
      ;
      return;
    }

    displayWelcome(user,pass);
    clearInput();
  }

  function clearInput(){
    setTimeout(function(){
      document.getElementById("username").value = "";
      document.getElementById("password").value = "";

      document.getElementById("welcome").innerHTML = "";
      alert("Le temps est écoulé, veuillez vous connecter une autre fois.");
      ;
      return;
    },10000);
  }

</script>

</body>
</html>
```

Exercice 3: 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 administrateur 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 `<div id="welcome"></div>`.

Ajouter administrateur

Le nouvel administrateur Assia a bien été ajouté !

Formulaire de Connexion

Ajouter administrateur

Formulaire de Connexion

Ajouter administrateur

Formulaire de Connexion

Connexion réussie, bienvenue Assia!

Ajouter administrateur

Formulaire de Connexion

Nom d'utilisateur ou mot de passe incorrect.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Formulaire de Connexion avec Temporisation</title>
</style>
  body {
    font-family: Arial, sans-serif;
    padding: 20px;
  }
  input {
    margin-bottom: 10px;
    padding: 8px;
  }
```

```

        width: 100%;
        max-width: 300px;
    }
    button {
        padding: 10px 20px;
        margin-top: 10px;
    }
    #message {
        color: green;
    }
}
</style>
</head>
<body>
    <div>
        <h1>Ajouter administrateur</h1>
        <input type="text" id="newUsername" placeholder="Nom d'utilisateur" required>
        <input type="password" id="newPassword" placeholder="Mot de passe" required>
        <button type="button" onclick="addAdmin()">Ajouter</button><br><br>

        <p id="message"></p>
    </div>

    <h1>Formulaire de Connexion</h1>
    <input type="text" id="username" placeholder="Nom d'utilisateur" required>
    <input type="password" id="password" placeholder="Mot de passe" required>
    <button type="button" onclick="connect()">Se connecter</button><br><br>

    <div id="welcome"></div>

    <script>

        var admins = [];

        function addAdmin(){
            let user = document.getElementById("newUsername").value;
            let pass = document.getElementById("newPassword").value;

            if (user === "" || pass === "") {
                alert("Veuillez entrer un Nom d'utilisateur et Mot de passe valide.");
                return;
            }

            // Vérification si l'administrateur existe déjà
            const exists = admins.some(admin => admin.nom === user);
            if (exists) {
                alert("Cet administrateur existe déjà !");
                return;
            }

            // Ajouter l'administrateur à la liste
            admins.push({nom: user, password: pass});
            document.getElementById("newUsername").value = '';
            document.getElementById("newPassword").value = '';

            // Affichage du message de succès
            document.getElementById("message").textContent = `Le nouvel
                administrateur ${user} a bien été ajouté !`;

            // Réinitialiser le message après 3 secondes
            setTimeout(() => {

```

```

        document.getElementById("message").textContent = '';
    }, 3000);
}

function connect(){
    let user = document.getElementById("username").value;
    let pass = document.getElementById("password").value;

    if (user === "" || pass === "") {
        alert("Veuillez entrer un Nom d'utilisateur et Mot de passe valide.");
        ;
        return;
    }

    // Appel de la fonction de validation avec callback
    validerUtilisateur(user, pass, (result) => {
        document.getElementById("welcome").textContent = result; // Affichage
        du résultat
    });
}

function validerUtilisateur(user, pass, callback){
    let found = false; // Indicateur pour savoir si un utilisateur est
    trouvé

    // Vérification des utilisateurs
    admins.forEach((admin) => {
        if (admin.nom === user && admin.password === pass) {
            callback(`Connexion réussie, bienvenue ${user}!`);
            // Marquer l'utilisateur comme trouvé
            found = true;
            // Sortir de la fonction forEach dès qu'une correspondance est
            trouvée
            return;
        }
    });

    // Si l'utilisateur n'a pas été trouvé, afficher un message d'erreur
    if (!found) {
        callback("Nom d'utilisateur ou mot de passe incorrect.");
    }
}
</script>

</body>
</html>

```

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.
- * Si aucun administrateur n'est trouvé, la promesse doit être rejetée (reject) avec un message d'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 `<div id="welcome"></div>`.

```
<script>

    var admins = [];

    function addAdmin(){
        let user = document.getElementById("newUsername").value;
        let pass = document.getElementById("newPassword").value;

        if (user === "" || pass === "") {
            alert("Veuillez entrer un Nom d'utilisateur et Mot de passe valide.");
            ;
            return;
        }

        // Vérification si l'administrateur existe déjà
        const exists = admins.some(admin => admin.nom === user);
        if (exists) {
            alert("Cet administrateur existe déjà !");
            return;
        }

        // Ajouter l'administrateur à la liste
        admins.push({nom: user, password: pass});
        document.getElementById("newUsername").value = '';
        document.getElementById("newPassword").value = '';

        // Affichage du message de succès
        document.getElementById("message").textContent = `Le nouvel
            administrateur ${user} a bien été ajouté !`;

        // Réinitialiser le message après 3 secondes
        setTimeout(() => {
            document.getElementById("message").textContent = '';
        }, 3000);
    }

    function connect(){
        let user = document.getElementById("username").value;
        let pass = document.getElementById("password").value;

        if (user === "" || pass === "") {
            alert("Veuillez entrer un Nom d'utilisateur et Mot de passe valide.");
            ;
            return;
        }

        // Appel de la fonction de validation avec promesse
        validerUtilisateur(user, pass)
            .then(result => {
                document.getElementById("welcome").textContent = result; //
                Affichage du résultat
            })
            .catch(error => {
                document.getElementById("welcome").textContent = error; //
                Affichage du message d'erreur
            });
    }

```



```

    }

    function validerUtilisateur(user, pass){
        return new Promise((resolve, reject) => {
            let found = false; // Indicateur pour savoir si un utilisateur est
                               trouvé

            // Vérification des utilisateurs
            admins.forEach((admin) => {
                if (admin.nom === user && admin.password === pass) {
                    resolve(`Connexion réussie, bienvenue ${user}!`);
                    found = true;
                    // Sortir de la fonction forEach dès qu'une correspondance
                    est trouvée
                    return;
                }
            });

            // Si l'utilisateur n'a pas été trouvé, rejeter la promesse
            if (!found) {
                reject("Nom d'utilisateur ou mot de passe incorrect.");
            }
        });
    }

</script>
</body>
</html>

```

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.

```

<script>

    var admins = [];

    function addAdmin(){
        let user = document.getElementById("newUsername").value;
        let pass = document.getElementById("newPassword").value;

        if (user === "" || pass === "") {
            alert("Veuillez entrer un Nom d'utilisateur et Mot de passe valide.");
            ;
            return;
        }

        // Vérification si l'administrateur existe déjà
        const exists = admins.some(admin => admin.nom === user);
        if (exists) {
            alert("Cet administrateur existe déjà !");
            return;
        }
    }

```

```
// Ajouter l'administrateur à la liste
admins.push({nom: user, password: pass});
document.getElementById("newUsername").value = '';
document.getElementById("newPassword").value = '';

// Affichage du message de succès
document.getElementById("message").textContent = `Le nouvel
    administrateur ${user} a bien été ajouté !`;

// Réinitialiser le message après 3 secondes
setTimeout(() => {
    document.getElementById("message").textContent = '';
}, 3000);
}

async function connect(){
    let user = document.getElementById("username").value;
    let pass = document.getElementById("password").value;

    if (user === "" || pass === "") {
        alert("Veuillez entrer un Nom d'utilisateur et Mot de passe valide.");
        ;
        return;
    }

    try {
        // Appel de la fonction de validation avec promesse
        const result = await validerUtilisateur(user, pass);
        document.getElementById("welcome").textContent = result; // Affichage
            du résultat
    } catch (error) {
        // Si une erreur survient, elle est capturée ici
        document.getElementById("welcome").textContent = error; // Affichage
            du message d'erreur
    }
}

function validerUtilisateur(user, pass){
    return new Promise((resolve, reject) => {
        let found = false; // Indicateur pour savoir si un utilisateur est
            trouvé

        // Vérification des utilisateurs
        admins.forEach((admin) => {
            if (admin.nom === user && admin.password === pass) {
                resolve(`Connexion réussie, bienvenue ${user}!`);
                found = true;
                // Sortir de la fonction forEach dès qu'une correspondance
                    est trouvée
                return;
            }
        });

        // Si l'utilisateur n'a pas été trouvé, rejeter la promesse
        if (!found) {
            reject("Nom d'utilisateur ou mot de passe incorrect.");
        }
    });
}
```

```
</script>  
</body>  
</html>
```