Asynchrone JS

Promesses

Rappel : une promesse est un objet (de type Promise) qui permet de traiter des fonctions asynchrones. Le paramètre d'une promesse est une fonction qui admet deux paramètres (fonctions de callback) : resolve et reject qui respectivement sont appelées si le code asynchrone est un succès ou un échec.

Une fois définie la promesse est consommée, on utilise alors deux méthodes :

- then pour récupérer le succès de la Promesse.
- catch pour récupérer l'échec de la Promesse.

01 Exercice Comsommation d'une promesse

- Ci-dessous on a défini une promesse myPromise, ré-écrivez la en utilisant uniquement des fonctions fléchées.

Notez que même si une Promesse permet de traiter du code asynchrone, vous pouvez y mettre du code synchrone comme dans l'exemple ci-dessous :

```
function myPromise(data, state) {

    return (new Promise( function(successCallback, failureCallback){
                if(state) {
                    successCallback(data);
                }else{
                    failureCallback('error');
                }
            })
    );
}

function error(){

    return "Error";
}

// Consommation
myPromise([1,2,3], true)
    .then(
        function (resultat){
            console.log(resultat);
            return resultat;
        }
    )
    .catch( error );
```

02 Exercice pile ou face

Définissez une Promesse (fonction fléchée) qui affiche pile ou face (resolve), utilisez la fonction Math.random() de JS pour un lancer de pile/face. Ils se feront avec un décalage de 500ms (setTimeout) fonction asynchrone.

Faites 3 lancers et définssez les paris suivants :

- 3 piles vous gagnez 1 bitcoin
- 1 pile exactement vous gagnez 0.001 bitcoin
- le reste des combinaisons vous perdez 0.5 bitcoin

Indications : vous pouvez utiliser Promise.all pour gérer les lancers.

03 Exercice fibonacci async (**)

Ecrivez un script qui retourne toutes les 500ms les nombres successifs de la suite de Fibonacci.

```
1 1 2 3 5 8 13 21 34 ...
```

Essayez maintenant d'encapsuler votre code dans une Promesse, que constatez-vous ?

04 Exercice algorithmique (**)

Soit la matrice de valeurs munériques dans le fichier data/matrix.json. Certaine(s) valeur(s) sur certaine(s) ligne(s) de cette matrice sont manquantes (notées None). Récupérez ces données et complétez les données manquantes en les remplaçant par la moyenne des valeurs de la ligne.

Modifié le: vendredi 22 septembre 2023, 13:08