Workshop Javascript

Prérequis avoir NodeJS et NPM d'installé

I. Les callbacks

I.1 Les callbacks Synchrone

L'objectif de cet exercice est de réaliser un callback synchrone. Un callback synchrone est un callback qui seras appelé directement sans passer par une possible 'phase d'attente'.

Par exemple, réaliser une fonction qui permet d'afficher l'heure et qui auras pour callback la suite de votre programme.

I.2 Les callbacks Asynchrone

Maintenant, vous allez réaliser un callback asynchrone. Un callback asynchrone est un callback qui permet de réaliser une tache entrainant un possible temps d'attente, et lorsque cette tâche est terminée, va appeler votre callback.

Ici, vous allez réaliser une fonction à callback qui va encapsuler un fs.readFile() Et qui lanceras votre callback qui affichera le résultat une fois terminé.

II. Les Promesses

II.1 Simple Promesse

L'objectif de cet exercice est de réaliser une promesse qui génèrera au bout de 2 secondes un chiffre aléatoire. (Avec l'aide de setTimeout() afin de simuler une attente)

Si le chiffre est impair, alors vous sortez une erreur, sinon vous validez la promesse.

II.2 Promesse dans fonction synchrone

Dans cet exercice, vous appeler la fonction précédente, mais dans une fonction synchrone (déclaré avec async).

Vous devrez gérer correctement les erreurs et les réussites.

II.3 Les promesses imbriqué

Ici, vous verrez qu'il est possible d'imbriquer des promesses afin de n'avoir qu'à gérer un seul catch pour la totalité de vos call redondant.

Dans le .then de votre promesse, vous pouvez return une promesse et récupérer le résultat dans un autre .then derrière le premier.

Essayez d'appliquer cette méthode sur plusieurs calls d'affilé de la fonction du II.1.

II.4 Liste de Promesse

Vous pouvez maintenant réaliser plusieurs calls à cette promesse simultanément, mais il faut que vous traitiez simultanément tous les résultats s'ils sont bon. Si un seul est faux, tous les autres sont considéré faux (voir Promise.all). N'hésitez pas à modifier la fonction du II.1 pour éviter d'avoir des échecs si vous voulez voir comment gérer les réussites.

III. Convertir l'un en l'autre

III.1 Convertir un callback en promesse

Nous allons ici utiliser la librairie bcrypt (npm install bcrypt dans votre dossier)

La librairie bcrypt a été développé de façon à fournir à la fois des callbacks et des promesses.

Ici, vous allez convertir le call bcrypt.hash() callback en promesse.

III.2 Convertir une promesse en callback

Ici, vous allez convertir le call bcrypt.hash() promesse en callback

Maintenant vous avez suffisamment d'information pour savoir comment bien gérer l'asynchrone du javascript

IV. Bonus

Créer votre propre API à l'aide d'Express Generator :

- Faite un GET sur votre api qui récupère les flux RSS reddit avec la librairie request
- Faite un GET/POST sur votre api qui lit/écrit dans une base de données avec mysql/knex/objections