La Plateforme

RunTrack JS : Bases

Why don't functions make good friends? They always return something.

Introduction du sujet

Ce sujet a pour but de vous faire reprendre les bases et de vous remettre en jambe sur JavaScript et sa syntaxe, en vous faisant travailler avec des fonctions.
Comme vous allez apprendre à faire des fonctions, **aucune fonction de base ou bibliothèque de JavaScript n'est autorisée à part console.log().**
Dans un dossier **runtrack-b2-js**, créez un dossier **jour-01**. Dans ce dossier, créez un dossier **job-XX** pour chacun des jobs **où XX correspond au numéro de chaque job**.
Pensez à faire des commits réguliers et à **respecter les consignes dans le sujet**.

Pour la bonne correction de vos projets, merci de mettre vos **déclarations de fonction au début** de chacun de vos fichiers. Pour chacun des jobs, vous **n'avez rien à afficher sur la page**. Votre code doit juste s'exécuter correctement. Vous pouvez bien évidemment utiliser console.log() pour le débugger, mais pensez à clean votre code avant de rendre votre projet.

Bon courage !

Job 01

Dans le dossier **job-01** faites un fichier **index.html** et un fichier **script.js**. Importez le script dans votre html et dans ce script, faites une fonction **myUpperCase()**. Cette fonction devra **passer une chaine de caractères en majuscule**.

Voici la signature de cette fonction, **veillez à la respecter** :

```
function myUpperCase(string) {
  /**
   * Your code here
   */
}
```

```
myUpperCase("Hello World") === "HELLO WORLD";
```

Le **paramètre devra être une chaîne de caractère** et la fonction devra **retourner une chaine de caractères**.

Job 02

Dans le dossier **job-02** faites un fichier **index.html** et un fichier **script.js**. Importez le script dans votre html et dans ce script, faites une fonction **myCountChar()**. Cette fonction devra **compter le nombre d'occurrences d'un caractère dans une chaîne de caractère donnée et retourner ce nombre**.

Voici la signature de cette fonction, **veillez à la respecter** :

```
function myCountChar(haystack, needle) {
  /**
   * Your code here
   */
}
```

```
myCountChar("Hello World", "o") === 2;
```

Cette fonction **prend en premier paramètre une chaine de caractères** et en **second paramètre le caractère à trouver.**

Job 03

Dans le dossier **job-03** faites un fichier **index.html** et un fichier **script.js**. Importez le script dans votre html et dans ce script, faites une fonction **myIsInString()**. Cette fonction devra **déterminer si une chaîne de caractère se trouve dans une autre en retournant un booléen**.

Voici la signature de cette fonction, **veillez à la respecter :**

```
function myIsInString(haystack, needle) {
  /**
   * Your code here
   */
}
```

```
myIsInString("Hello World", "llo") === true;

myIsInString("Hello World", "rele") === false;
```

Job 04

Dans le dossier **job-04** faites un fichier **index.html** et un fichier **script.js**. Importez le script dans votre html et dans ce script, faites une fonction **myArraySum()**. Cette fonction devra **faire la somme des éléments dans un tableau et retourner cette somme**.

Voici la signature de cette fonction, **veillez à la respecter** :

```
function myArraySum(array) {
  /**
   * Your code here
   */
}
```

```
myArraySum([4, 24, 52, 14, 32, 56]) === 182;
```

Job 05

Dans le dossier **job-05** faites un fichier **index.html** et un fichier **script.js**. Importez le script dans votre html et dans ce script, faites une fonction **myPrimeList()**. Cette fonction devra **retourner l'ensemble des nombres premiers jusqu'à un nombre donné**.

Voici la signature de cette fonction, **veillez à la respecter** :

```
function myPrimeList(limit) {
  /**
   * Your code here
   */
}
```

```
myPrimeList(18) === [2, 3, 5, 7, 11, 13, 17];
```

Job 06

Dans le dossier **job-06** faites un fichier **index.html** et un fichier **script.js**. Importez le script dans votre html et dans ce script, faites une fonction **mySquareArray()**. Cette fonction devra **retourner un tableau de nombre au carré par rapport au tableau passé en paramètre**.

Voici la signature de cette fonction, **veillez à la respecter** :

```
function mySquareArray(array) {
  /**
   * Your code here
   */
}
```

```
mySquareArray([3, 8, 4, 2, 5]) === [9, 64, 16, 4, 25];
```

Job 07

Dans le dossier **job-07** faites un fichier **index.html** et un fichier **script.js**. Importez le script dans votre html et dans ce script, faites une fonction **myNearZero()**. Cette fonction devra **retourner l'entier le plus proche de zéro selon un tableau envoyé en paramètre**.

Voici la signature de cette fonction, **veillez à la respecter** :

```
function myNearZero(array) {
  /**
   * Your code here
   */
}
```

```
myNearZero([3, 8, 4, 2, 5]) === 2;

myNearZero([-1, -4, 2, 5, 6, 9]) === -1;
```

Job 08

Dans le dossier **job-08** faites un fichier **index.html** et un fichier **script.js**. Importez le script dans votre html et dans ce script, faites une fonction **myArraySort()**. Cette fonction devra **trier un tableau dans l'ordre croissant ou décroissant en fonction d'un paramètre passé en entrée**.

Voici la signature de cette fonction, **veillez à la respecter** :

```
function myArraySort(array, sorting) {
  /**
   * Your code here
   */
}
```

```
myArraySort([3, 8, 4, 2, 5], "ASC") === [2, 3, 4, 5, 8];

myNearZero([-1, -4, 2, 5, 6, 9], "DESC") === [9, 6, 5, 2, -1, -4];
```

Compétences visées

- **Algorithmie :** 5 points
- **JavaScript** : 5 points

Rendu

Le projet est à rendre sur **https://github.com/prenom-nom/runtrack-b2-js**.

C'est le premier jour de RunTrack, donc normalement, vous n'avez que le dossier jour-01 à push. Faites attention à bien **respecter la signature de chacune de vos fonctions** et les **noms de fichiers et de dossiers à la lettre !**