8

Module : JavaScript

Exercice n°5 - Arrays et Manipulating Arrays

# Durée de l’exercice : 30minArrays et Manipulating Arrays

La classe JavaScript Array est un objet global qui est utilisé dans la construction de tableaux ; qui sont des objets de haut niveau, semblables à des listes.

On stocke des valeurs liées dans une seule variable. Un tableau contient un ou plusieurs éléments.

Un tableau est composé d'un ou de plusieurs éléments. Un index est un nombre qui correspond à la position d'un élément dans un tableau. Le premier élément d'un tableau a l'indice 0, le deuxième élément a l'indice 1, et ainsi de suite. L'indexation est un modèle que les programmeurs utilisent souvent dans les tableaux. Ce modèle apparaîtra également plus tard dans des applications plus avancées.

**Objectif de l'exercice** : connaître et comprendre ce qu'est un tableau, comment créer un tableau et les méthodes JavaString Array.

**Énoncé** : Suivez les étapes ci-dessous pour réaliser votre exercice. Prédisez d'abord le résultat puis exécutez le code dans le terminal et comparez votre prédiction.

Créez un nouveau dossier nommé A\_Array. Créez un fichier nommé array\_one.js à l'intérieur.

Exécutez le code suivant pour comparer le résultat.

let fruits = ['Apple', 'Banana', 'Mango', 'Orange']

console.log(fruits[1])

console.log(fruits.length)

let first = fruits[0]

console.log(first)

// Pomme

let last = fruits[fruits.length - 1]

console.log(last)

// Orange

*Note* : Veuillez d'abord compléter les modules Conditions, Boucles et Fonctions avant de résoudre les problèmes restants dans Array.

Créez les fichiers ci-dessous dans le dossier A\_array et résolvez les problèmes.

# array\_sum.js

// Ecrivez une fonction `arraySum` qui accepte un tableau de nombres comme argument. La fonction doit retourner la somme totale de tous les éléments du tableau.

console.log(arraySum([6, 2, 3])) ; // 11

console.log(arraySum([7, 6, -5, 2])) ; // 10

console.log(arraySum([8])) ; // 8

console.log(arraySum([])) ; // 0

# Créez un fichier array\_odds.js et résolvez la fonction ci-dessous :

// Ecrivez une fonction `arrayOdds` qui accepte un tableau de nombres comme argument.

// La fonction doit retourner un nombre représentant le nombre d'éléments impairs dans le tableau.

console.log(arrayOdds([4, 7, 2, 5, 9])) ; // 3

console.log(arrayOdds([11, 31, 58, 99, 21, 60])) ; // 4

console.log(arrayOdds([100, 40, 4])) ; // 0

# Créez un fichier strings-to-num.js et résolvez la fonction ci-dessous :

// Ecrivez une fonction `stringsToNum` qui accepte un tableau de chaînes de caractères comme argument.

// La fonction doit renvoyer un nouveau tableau contenant les longueurs des éléments du tableau d'origine.

console.log(stringsToNum([ "orange", "game", "movie","beau"])) ; // [6, 4, 5, 9]

console.log(stringsToNum(["he", "off", "handmade"])) ; // [2, 3, 8]

# Créez un fichier divisors.js et résolvez la fonction ci-dessous :

// Ecrivez une fonction `divisors` qui accepte un nombre comme argument.

// La fonction doit retourner un tableau contenant tous les nombres positifs qui peuvent être divisés en arguments.

console.log(divisors(15)) ; // [1, 3, 5, 15].

console.log(divisors(7)) ; // [1, 7]

console.log(divisors(24)) ; // [1, 2, 3, 4, 6, 8, 12, 24]

Créez un nouveau dossier nommé B\_Array et enregistrez tous les fichiers pour les exercices ci-dessous.

# Créez un fichier make-acronym.js et résolvez la fonction ci-dessous :

// Ecrivez une fonction `makeAcronym` qui accepte une chaîne de phrases comme argument.

// La fonction doit retourner une chaîne de caractères contenant le premier caractère de chaque mot de chaque mot de la phrase en lettres capitales.

console.log(makeAcronym("los Angeles")) ; // LA

console.log(makeAcronym("same stuff different day")) ; // SSDD

console.log(makeAcronym("Laugh out loud")) ; // LOL

console.log(makeAcronym("don't over think stuff")) ; // DOTS

# Créez un fichier reverse-array.js et résolvez la fonction ci-dessous :

// Ecrivez une fonction `reverseArray` qui accepte un tableau comme argument.

// La fonction doit retourner un tableau contenant les éléments du tableau d'origine dans l'ordre inverse.

console.log(reverseArray(["Four", "Five", "Six", "Seven"])) ; // ['Seven', 'Six', 'Five', 'Four']

console.log(reverseArray([6, 1, 7])) ; // [7, 1, 6]

# Créez un fichier maximum.js et résolvez la fonction ci-dessous :

// Ecrivez une fonction `maximum` qui accepte un tableau de nombres comme argument.

// La fonction doit retourner le plus grand nombre du tableau. Si le tableau est vide, la fonction doit renvoyer null.

console.log(maximum([4, 6, 3, 9])) ; // 7

console.log(maximum([13, 15, 21, 11, 2])) ; // 19

console.log(maximum([])) ; // null

#Créer un fichier word-count.js et résoudre la fonction ci-dessous :

// Ecrivez une fonction `wordCount(sentence, targetWords)` qui accepte une chaîne de phrase et un tableau de mots cibles.

// La fonction doit retourner un compte du nombre de mots de la phrase qui se trouvent dans `targetWords`.

console.log(wordCount("ouvrez la fenêtre s'il vous plaît", ["s'il vous plaît", "ouvrir", "désolé"])) ; // 2

console.log(wordCount("conduire au cinéma", ["le", "conducteur"])) ; // 1

console.log(wordCount("puis-je avoir cette canette", ["can", "je"])) ; // 3

Créez un nouveau dossier nommé C\_Array et enregistrez tous les fichiers pour les exercices ci-dessous.

# Créez un fichier longest-word.js et résolvez la fonction ci-dessous :

// Ecrivez une fonction `longestWord` qui accepte une chaîne de phrase comme argument.

// La fonction doit retourner le mot le plus long de la phrase. S'il y a une égalité, elle renvoie le mot qui apparaît le plus tard.

console.log(longestWord("I am pretty hungry")) ; // 'hungry' (faim)

console.log(mot le plus long("nous devrions penser en dehors de la boîte")) ; // "à l'extérieur".

console.log(longestWord("down the rabbit hole")) ; // "rabbit" (lapin)

console.log(longestWord("elephant down")) ; // "elephant".

# Créez un fichier common-elements.js et résolvez la fonction ci-dessous :

// Écrivez une fonction `commonElements` qui accepte deux tableaux en tant que arguments.

// La fonction doit retourner un nouveau tableau contenant les éléments qui se trouvent dans les deux tableaux d'entrée. L'ordre des éléments dans le tableau de sortie n'a pas d'importance tant que la fonction renvoie les bons éléments.

let arr1 = ["a", "c", "d", "b"] ;

let arr2 = ["b", "a", "y"] ;

console.log(commonElements(arr1, arr2)) ; // ['a', 'b']

let arr3 = [4, 7] ;

let arr4 = [32, 7, 1, 4] ;

console.log(commonElements(arr3, arr4)) ; // [4, 7] ;

# Créez un fichier number-range.js et résolvez la fonction ci-dessous :

// Ecrivez une fonction `numberRange(min, max, step)` qui accepte comme arguments trois nombres comme, `min`, `max`, et `step`.

// La fonction doit retourner tous les nombres entre `min` et `max` avec un intervalle de `step`.

//`min` et `max` sont inclusifs.

console.log(numberRange(10, 40, 5)) ; // [10, 15, 20, 25, 30, 35, 40]

console.log(numberRange(14, 24, 3)) ; // [14, 17, 20, 23].

console.log(nombreRange(8, 35, 6)) ; // [8, 14, 20, 26, 32]