TP 5

Méthodes d'itération

Le but de se TP est de réaliser quelques traitements sur des tableaux en se basant sur des méthodes d'itération.

Les méthodes d'itération sont documentées dans la page dédiée à Array :
https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array

Consigne générale pour ***ce*** TP : il est interdit d'écrire une boucle.

Compétences travaillées dans ce TP

- Syntaxe des méthodes d'itération
- Utilisation d'une documentation
- (Étude de complexité)

1 – Consignes générales

Mêmes consignes que les TP précédents concernant l'organisation de votre code : séparez les fonctions réutilisables, les tests et le programme principal dans des fichiers différents. Chaque fonction `xxx()` doit être testée par une fonction `xxxTest()`.

2 – Calcul de valeur absolue

Implémentez et testez une fonction :

abs`(numbers)`

renvoie un tableau contenant la valeur absolue de chaque nombre présent dans le tableau `numbers`.

Contrainte : Le tableau donné en argument doit être laissé intact.

Exemple d'utilisation :

```
let numbers = [-42, 3, 0, -1, 100];
let absNumbers = abs(numbers);
console.log(absNumbers);      // => [42, 3, 0, 1, 100]
console.log(numbers);         // => [-42, 3, 0, -1, 100]
```

3 – Filtrage sur la parité

Implémentez et testez une fonction :

evenOnly`(numbers)`

renvoie un tableau contenant uniquement les nombres pairs présents dans le tableau `numbers`.

Contrainte : Le tableau donné en argument doit être laissé intact.

Exemple d'utilisation :

```
console.log( evenOnly([0,1,2,3,4]) );  // => [0, 2, 4]
```

4 – Somme

Réimplémentez la fonction `sum()` du TP précédent en utilisant une méthode d'itération appropriée, et réexécutez les tests pour vérifier qu'elle se comporte comme attendu.

5 – Aplatissement 2D

Réimplémentez la fonction `flatten2D()` du TP précédent en utilisant une méthode d'itération appropriée, et réexécutez les tests pour vérifier qu'elle se comporte comme attendu.

6 – Suppression de doublons

Implémentez et testez une fonction :

removeDuplicates(values)

renvoie un tableau dans lequel tous les doublons de `values` tout ont été supprimés, tout en conservant l'ordre des éléments.

Exemple d'utilisation :

```
console.log( removeDuplicates([]) );                  // => []
console.log( removeDuplicates([1,2,3,4]) );           // => [1,2,3,4]
console.log( removeDuplicates([1,2,3,2,4,3,1,1]) );   // => [1,2,3,4]
```

Variante

Essayez d'implémenter une variante en utilisant une autre méthode d'itération.

Indice : on peut utiliser `filter()` ou `reduce()`.

[Facultatif] Aplatissement : généralisation

Réimplémentez la fonction `flatten()` du TP précédent en utilisant une méthode d'itération appropriée, et réexécutez les tests pour vérifier qu'elle se comporte comme attendu.

[Facultatif] Suppression de doublons sans contrainte d'ordre

Considérons à présent que `removeDuplicates(values)` ne doive plus respecter l'ordre initialement présent dans `values` : le tableau renvoyé ne doit plus contenir de doublons, mais l'ordre des éléments dans ce nouveau tableau est laissé libre.

Ce relâchement de contrainte permet d'envisager l'implémentation suivante :

```
function removeDuplicates(values){
    return values.sort().reduce( function(uniqueValues, value){
            if (value !== uniqueValues[0]){
                uniqueValues.unshift(value);
            }
            return uniqueValues;
        }, []);
}
```

Observez ce que donne cette fonction sur des exemples.

Étudiez la complexité de cette implémentation et comparez-la avec celle que vous avez réalisé dans l'exercice 6.