# Introduction aux structures de données

## Les tableaux

Vous pouvez définir un tableau, qui n'est rien d'autre qu'un objet JS, simplement à l'aide de crochets :

```
let fruits =  ['Apple', 'Orange'];
```

## 01 Exercice reference array

1. Reprenez la variable fruits ci-dessus. Que vaut le console.log dans l'exemple suivant? Affichez le contenu des deux tableaux :

```
let fruits =  ['Apple', 'Orange'];

let newFruits = fruits;

newFruits.push('Banana')

console.log(newFruits === fruits)
```

2. Ecrivez un script pour créer un nouveau tableau newFruits qui n'a pas la même référence que le tableau fruits. Puis vérifiez que ce n'est plus la même référence :

```
console.log(newFruits === fruits);
```

## Fonction map

La méthode map permet de parcourir un tableau et d'exécuter une fonction pour chaqu'un de ses éléments. Elle retournera un nouveau tableau.

```
const sheeps = ['🐑', '🐑', '🐑'];

const newSheeps = sheeps.map( sheep => sheep + sheep );
 // ["🐑🐑", "🐑🐑", "🐑🐑"]
```

## 02 Exercice power 3

Soit numbers une liste de nombres entiers, élevez uniquement à la puissance 3 les nombres pairs.

```
const numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
```

*Indications : pour calculer une puissance utilisez l'opérateur suivant*

```
// opérateur puissance
2**3 // 8
```

- filter, il permet de filtrer des données dans un tableau en fonction d'un critère.

```
const numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

numbers.filter(number => number > 4);
// [5, 6, 7, 8, 9, 10]
```

- reduce, applique une fonction qui est un accumulateur et qui traite chaque valeur d'une liste de la gauche vers la droite afin de la réduire en une seule valeur. Vous pouvez passer en deuxième paramètre une valeur facultative.

```
const numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
// première paramètre fonction fléchée, deuxième paramètre val init de acc
const total = numbers.reduce((acc, curr) => curr + acc, 0);
console.log(total); // affiche 55

numbers.reduce((acc, curr) => curr + acc, 100);
// 155
```

# 03 Exercice populations

- 1. Parcourez le tableau populations et ajoutez un champ count qui compte le nombre d'occurence(s) de a et de l dans les noms. Utilisez un for of.

```
const populations = [
    { "id": 0, "name": "Alan" },
    { "id": 1, "name": "Albert" },
    { "id": 2, "name": "Jhon" },
    { "id": 3, "name": "Brice" },
    { "id": 4, "name": "Alexendra" },
    { "id": 5, "name": "Brad" },
    { "id": 6, "name": "Carl" },
    { "id": 7, "name": "Dallas" },
    { "id": 8, "name": "Dennis" },
    { "id": 9, "name": "Edgar" },
    { "id": 10, "name": "Erika" },
    { "id": 11, "name": "Isaac" },
    { "id": 12, "name": "Ian" }
];
```

- 2. Ordonnez maintenant le tableau par ordre croissant de nombre de a et l dans les noms.

# 04 Exercice max (challenge)

Soit le tableau d'entiers suivant :

```
const numbers = [1, 2, 3, 4, 50, 6, 7, 8, 9, 10];
```

Utilisez la méthode reduce pour calculer le max.

# 05 Exercice reduce sum impair (challenge)

Faites la somme des nombres impairs en utilisant la fonction reduce des valeurs suivantes :

```
const numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
```

# 06 Exercice fonction map sur un littéral

Utilisez la fonction map pour calculer le prix TTC des téléphones. Utilisez une fonction fléchée.

```
const phones = [
  { name: "iphone XX", priceHT: 900 },
  { name: "iphone X", priceHT: 700 },
  { name: "iphone B", priceHT: 200 },
];
```

# 07 Exercice square numbers

1. 1. Soit le point A suivant, calculez la distance de ce point à l'ensemble de chaques points de la liste positions. Vous donnerez les résultats dans un nouveau tableau distances.

```
const A = [8.3, 7.5];
const positions = [[1,1], [2, 2], [3, 4.5], [0, 9]];
const distances = [];
```

Ro ns comment on effectue le calcul de la distance avec une précision de deux chiffres après la virgule :

```
const X = [1,2];
const B = [4,1.5];

const d = Math.floor( Math.square( (X[0] - B[0] )**2 + (X[1] - B[1] )**2 ) * 100) / 100 ;
```

2. Trouvez le point le plus éloigné du point A.

# 08 Exercice string

1. Inversez la chaîne de caractères sentence ci-après.
2. Comptez le nombre de caractères de chaque mot.
3. Faites un script qui prend en argument une phrase et qui retourne dans un tableau le nombre de caractères de chaque mot. Vous ne compterez pas les espaces comme un caractère.

Indication : utilisez la méthode split pour transformer la chaîne de caractères en tableau.

```
const sentence = "Bonjour tout le monde, vous aimez JS ?";
```

# Structure de données Map

Un objet Map est une collection de paires clé/valeur qui peut utiliser n'importe quel type de valeur pour sa clé.

```
const jedi = new Map()
```

## Ajout de valeurs dans un Map

Vous utiliserez la méthode set de l'objet Map pour ajouter des valeurs.

```
jedi.set('firstName', 'Luke')
jedi.set('lastName', 'Skywalker')
jedi.set('job', 'Jedi Master')
```

Vous pouvez également ajouter des valeurs dans un map à l'aide d'un tableau de tableaux :

```
const jedi = new Map([
  ['firstName', 'Luke'],
  ['lastName', 'Skywalker'],
  ['job', 'Jedi Master'],
])
```

Exemple de quelques fonctions utiles :

```


// rechercher une clé
jedi.has('shark') // false

// accéder à une valeur à partir de sa clé
jedi.get('firstName')

// taille du Map
jedi.size

// supprimer un élément
jedi.delete('firstName');

// tout supprimer
jedi.clear()

// les keys et values
jedi.keys()
jedi.values()
// les deux
jedi.entries()
```

## Itération sur un Map

- à l'aide d'un for of

```
for (const [key, value] of jedi) {
  console.log(`${key}: ${value}`)
}
```

- à l'aide d'un foreEach

```
jedi.forEach(( v, k ) =>  console.log(v, k));
```

# Set

L'objet Set, qui se traduit par ensemble en français, permet de stocker des valeurs uniques de n'importe quel type : primitif ou objet.

Voici en résumé les différentes méthodes d'un Set.

```
const ensemble = new Set();

ensemble.add(1);
ensemble.add(5);
ensemble.add("100");

ensemble.has(1); // true
ensemble.has(3); // false
ensemble.size; // 3

ensemble.delete(5); // retire 5 du set
ensemble.has(5);    // false, 5 a été retiré de l'ensemble

ensemble.size; // 2, on a retiré une valeur de l'ensemble
console.log(ensemble); // Set [ 1, "du texte" ]
```

# 09 Exercice count letters

Comptez chacune des lettres dans "Mississipi". Affichez le résultat dans une structure de données lisible.

Généralisez et créez maintenant une fonction qui prend en paramètre une chaîne de caractères et qui retourne le nombre d'occurences de chacune de ses lettres.

# 10 Exercice count digit

Soit la chaîne de caractères suivante, récupérez tous les numériques de cette chaîne dans un tableau :

```
const phrase = '8790:bonjour le monde:8987:7777:Hello World:9007';
```

D'autres structures de données existent en JS comme les WeakSet, WeakMap par exemple. Nous vous invitons, pour approfondir vos connaissances, à les découvrir sur la documentation devlopper.mozilla.org.

Modifié le: vendredi 22 septembre 2023, 13:02