Apprendre JavaScript - Les tableaux

Table des matières

Introduction

Les tableaux sont des objets qui permettent de stocker plusieurs valeurs dans une seule variable. Ces valeurs peuvent être de n'importe quel type, et elles sont indexées par des entiers. Les tableaux sont des objets de type `Array`, et ils possèdent de nombreuses méthodes pour les manipuler.

https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Global_Objects/Array

Création d'un tableau

Il existe plusieurs façons de créer un tableau en JavaScript. La plus simple consiste à utiliser la syntaxe littérale, qui consiste à entourer les valeurs du tableau par des crochets `[]` :

```
let fruits = ["pomme", "banane", "orange"];
```

Accès aux éléments d'un tableau

Les éléments d'un tableau sont indexés par des entiers, à partir de zéro. On peut accéder à un élément d'un tableau en utilisant la notation `[]` :

```
let fruits = ["pomme", "banane", "orange"];

console.log(fruits[0]); // Affiche 'pomme'

console.log(fruits[1]); // Affiche 'banane'

console.log(fruits[2]); // Affiche 'orange'
```

Modifier un tableau

Modification des éléments d'un tableau

On peut modifier les éléments d'un tableau en utilisant la notation `[]` :

```
let fruits = ["pomme", "banane", "orange"];

fruits[1] = "kiwi";

console.log(fruits); // Affiche ['pomme', 'kiwi', 'orange']
```

Ajout d'éléments à la fin un tableau

On peut ajouter des éléments à un tableau en utilisant la méthode `push` :

```
let fruits = ["pomme", "banane", "orange"];

fruits.push("kiwi");

console.log(fruits); // Affiche ['pomme', 'banane', 'orange', 'kiwi']
```

Suppression d'éléments à la fin un tableau

On peut supprimer le dernier élément d'un tableau en utilisant la méthode `pop`. Lorsqu'on supprime, il est possible de récupérer la valeur supprimée et la stocker dans une variable.

```
let fruits = ["pomme", "banane", "orange"];

let dernierFruit = fruits.pop();

console.log(fruits); // Affiche ['pomme', 'banane']
console.log(dernierFruit); // Affiche 'orange'
```

Ajout d'éléments au début d'un tableau

On peut ajouter des éléments au début d'un tableau en utilisant la méthode `unshift`. C'est l'équivalent de `push` mais pour le début du tableau.

Suppression d'éléments au début d'un tableau

On peut supprimer le premier élément d'un tableau en utilisant la méthode `shift`. C'est l'équivalent de `pop` mais pour le début du tableau.

Meilleures pratiques

Ajouter des éléments à la fin d'un tableau est plus efficace que d'ajouter des éléments au début, car cela ne nécessite pas de décaler tous les éléments existants. De même, supprimer un élément à la fin d'un tableau est plus efficace que de supprimer un élément au début. Cela demande moins de ressources à l'ordinateur.

Parcourir un tableau

Méthode avec une boucle `for`

On peut parcourir un tableau en utilisant une boucle `for` :

```
let fruits = ["pomme", "banane", "orange"];

for (let i = 0; i < fruits.length; i++) {
    console.log(fruits[i]);
}
```

Méthode avec la méthode `forEach`

On peut parcourir un tableau en utilisant la méthode `forEach`. Cette méthode prend une fonction en argument, qui sera exécutée pour chaque élément du tableau.

Cette méthode est plus élégante que la boucle `for`, et elle est souvent préférée par les développeurs JavaScript. Elle contient 3 paramètres : l'élément, l'index et le tableau en soi.

```
let fruits = ["pomme", "banane", "orange"];

fruits.forEach(function (fruit, index, tableau) {
    console.log(fruit, index);
});
```

Méthode avec la méthode `map`

On peut parcourir un tableau en utilisant la méthode `map`. Cette méthode renvoie un nouveau tableau qui contient les éléments transformés par une fonction. Celà permet de transformer les éléments du tableau sans modifier le tableau original.

```
let fruits = ["pomme", "banane", "orange"];

let fruitsEnMajuscules = fruits.map(function (fruit) {
    return fruit.toUpperCase();
});

console.log(fruitsEnMajuscules); // Affiche ['POMME', 'BANANE', 'ORANGE']
```

Recherche d'éléments dans un tableau

Méthode avec une boucle `for`

On peut rechercher un élément dans un tableau en utilisant une boucle `for` :

```
let fruits = ["pomme", "banane", "orange"];

let fruitRecherche = "banane";

for (let i = 0; i < fruits.length; i++) {
    if (fruits[i] === fruitRecherche) {
        console.log("Le fruit a été trouvé à l'index " + i);
        break; // On arrête la boucle dès qu'on a trouvé le fruit. Plus besoin de
continuer.
    }
}
```

Méthode avec la méthode `indexOf`

On peut rechercher un élément dans un tableau en utilisant la méthode `indexOf`. Cette méthode renvoie l'index de l'élément recherché, ou `-1` s'il n'est pas trouvé.

```
let fruits = ["pomme", "banane", "orange"];

let fruitRecherche = "banane";

let index = fruits.indexOf(fruitRecherche);

if (index !== -1) {
    console.log("Le fruit a été trouvé à l'index " + index);
} else {
    console.log("Le fruit n'a pas été trouvé");
}
```

Méthode avec la méthode `includes`

On peut rechercher un élément dans un tableau en utilisant la méthode `includes`. Cette méthode renvoie `true` si l'élément est trouvé, ou `false` sinon.

```
let fruits = ["pomme", "banane", "orange"];

let fruitRecherche = "banane";

if (fruits.includes(fruitRecherche)) {
    console.log("Le fruit a été trouvé");
} else {
    console.log("Le fruit n'a pas été trouvé");
}
```

Filtrer un tableau

Méthode avec un boucle `for`

On peut filtrer un tableau en utilisant une boucle `for` :

```
let fruits = ["pomme", "banane", "orange", "banane", "kiwi"];

let fruitsSansBanane = [];

for (let i = 0; i < fruits.length; i++) {
    if (fruits[i] !== "banane") {
        fruitsSansBanane.push(fruits[i]);
```

```
    }
}
```

Méthode avec la méthode `filter`

On peut filtrer un tableau en utilisant la méthode `filter`. Cette méthode renvoie un nouveau tableau qui contient les éléments qui passent un test donné par une fonction.

Cette méthode est plus élégante que la boucle `for`, et elle est souvent préférée par les développeurs JavaScript.

```
let fruits = ["pomme", "banane", "orange", "banane", "kiwi"];

let fruitsSansBanane = fruits.filter(function (fruit) {
    return fruit !== "banane";
});
```

Modifier l'ordre des éléments dans un tableau

Trier un tableau

On peut trier un tableau en utilisant la méthode `sort`. Cette méthode trie les éléments du tableau en place, et renvoie le tableau trié.

```
let fruits = ["pomme", "banane", "orange", "kiwi"];

fruits.sort();

console.log(fruits); // Affiche ['banane', 'kiwi', 'orange', 'pomme']
```

Préciser l'ordre de tri

Par défaut, la méthode `sort` trie les éléments du tableau en les convertissant en chaînes de caractères, puis en les comparant selon leur code Unicode. Cela peut donner des résultats inattendus pour les nombres.

Pour trier des nombres, on peut passer une fonction de comparaison en argument à la méthode `sort`. Cette fonction prend deux arguments, et renvoie un nombre négatif si le premier argument est inférieur au deuxième, un nombre positif si le premier argument est supérieur au deuxième, et zéro si les deux arguments sont égaux.

```
let nombres = [10, 5, 20, 1, 15];

nombres.sort(function (a, b) {
    return a - b;
```

```
});

console.log(nombres); // Affiche [1, 5, 10, 15, 20]
```

Inverser l'ordre des éléments d'un tableau

On peut inverser l'ordre des éléments d'un tableau en utilisant la méthode `reverse`. Cette méthode inverse les éléments du tableau en place, et renvoie le tableau inversé.

```
let fruits = ["pomme", "banane", "orange", "kiwi"];

fruits.reverse();

console.log(fruits); // Affiche ['kiwi', 'orange', 'banane', 'pomme']
```

Suppression d'éléments à un index donné

On peut supprimer un élément à un index donné en utilisant la méthode `splice`. Cette méthode prend deux arguments : l'index de l'élément à supprimer, et le nombre d'éléments à supprimer à partir de cet index.

Il est possible de récupérer les éléments supprimés et de les stocker dans une variable.

Il est aussi possible d'ajouter des éléments à la place des éléments supprimés en les passant en argument à la fin de la méthode `splice`.

```
let fruits = ["pomme", "banane", "orange"];

let fruitSupprime = fruits.splice(1, 1, "kiwi", "ananas");
```

Fusion et copie de tableaux

Copie d'un tableau

Il est important de savoir que les tableaux sont des objets en JavaScript. Lorsqu'on affecte un tableau à une autre variable, on ne copie pas le tableau, mais on crée une référence vers le même tableau. Cela signifie que si on modifie le tableau à travers une des variables, les modifications seront visibles à travers l'autre variable.

Pour copier un tableau, on peut utiliser la méthode `slice`. Ainsi, on crée une copie du tableau, et non une référence vers le même tableau. On peut aussi utiliser la méthode `concat` pour concaténer un tableau vide avec le tableau à copier.

```

```

```
let fruits = ["pomme", "banane", "orange"];

let copieFruits = fruits.slice();
```

Joindre deux tableaux

On peut joindre deux tableaux en utilisant la méthode `concat`. Cette méthode renvoie un nouveau tableau qui contient les éléments des deux tableaux. Cela brise la référence entre les deux tableaux et crée un nouveau tableau.

```
let fruits1 = ["pomme", "banane", "orange"];

let fruits2 = ["kiwi", "ananas"];

let tousLesFruits = fruits1.concat(fruits2);
```