Les tableaux et leurs méthodes

Objectif : manipuler des tableaux (accéder, modifier, ajouter, supprimer des éléments), parcourir un tableau et utiliser des méthodes modernes comme forEach(), map(), filter(), reduce()

Touchet Laurent / Afec 2025

Sommaire

C’est quoi un tableau en JavaScript ?

Un **tableau JS** permet de stocker plusieurs valeurs dans une seule variable.

Chaque valeur a une position, qu’on appelle un **index**, et qui **commence à 0**.

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

Valeur	Index
'pomme’	0
‘banane’	1
‘kiwi’	2

Accéder et modifier des éléments d'un tableau JS

Pour accéder à un élément spécifique d'un tableau, on indique son index :

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

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

console.log(fruits[2]); // "kiwi"
```

Pour modifier un élément du tableau, on indique également son index :

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

fruits[1] = "orange"; // ["pomme", "orange", "kiwi"]
```

Ajouter et supprimer des éléments au début d'un tableau JS

Ajouter un élément au début d'un tableau avec **unshift**

```
let fruits = [ "pomme", "banane", "kiwi" ];
fruits.unshift("raisin"); // [”raisin”, "pomme", "banane", "kiwi"]
```

Supprimer un élément au début d'un tableau avec **shift :**

```
let fruits = [ "pomme", "banane", "kiwi" ];
fruits.shift(); // ["banane", "kiwi"]
```

Astuce : push() et pop() = fin du tableau, unshift() et shift() = début du tableau.

Ajouter et supprimer des éléments à la fin d'un tableau JS

Ajouter un élément à la fin d'un tableau avec **push** :

```
let fruits = [ "pomme", "banane", "kiwi" ];
fruits.push(“fraise”); // ["pomme", "banane", "kiwi", “fraise”]
```

Supprimer un élément à la fin d'un tableau avec **pop :**

```
let fruits = [ "pomme", "banane", "kiwi" ];
fruits.pop(); // ["pomme", "banane"]
```

Astuce : push() et pop() = fin du tableau, unshift() et shift() = début du tableau.

Ajouter et supprimer des éléments à la fin d'un tableau JS

Retirer un élément peu importe sa position avec tableau**.indexOf()** si on connait sa valeur.

```
let fruits = ["pomme", "banane", "kiwi"];
fruits.splice(fruits.indexOf("banane"), 1);
console.log(fruits); // ["pomme", "kiwi"]
```

indexOf("banane") : cherche l'index de "banane" dans le tableau

splice(..., 1) : supprime 1 élément à cet index (1 indique combien d'éléments on veut supprimer à partir de l'index donné.) Attention, si "banane" n'est pas trouvé, indexOf retourne -1, ce qui ferait splice(-1, 1) : ça enlèverait le dernier élément par erreur. Sécuriser avec **include()** qui retournera true/false.

```
let fruits = ["pomme", "banane", "kiwi"];
if (fruits.includes("banane")) {
     fruits.splice(fruits.indexOf("banane"), 1);
}
console.log(fruits); // ["pomme", "kiwi"]
```

Parcourir un tableau JS

Avec une boucle for

```
let fruits = [ "pomme", "banane", "kiwi" ];
for (let i = 0; i < fruits.length; i++) {
 console.log("J'aime ce fruit : " + fruits[i]);
}
```

Avec forEach

```
let fruits = [ "pomme", "banane", "kiwi" ];
fruits.forEach(function(fruit) {
 console.log("J'aime ce fruit : " + fruit);
});
```

Méthodes utiles sur les tableaux JS : map()

map() : transformer chaque élément d'un tableau

```
let nombres = [1, 2, 3];
let doubles = nombres.map(function(n) {
 console.log(n * 2);
});
// Résultat : [2, 4, 6]
```

.map() crée un nouveau tableau en appliquant une fonction à chaque élément du tableau d'origine.
Elle retourne ce nouveau tableau, sans modifier l'original.

- Si tu n'écris pas de **return**, ou que tu oublies de retourner quelque chose dans la fonction, tu obtiens un tableau rempli de undefined.
- .map() **ne modifie pas** le tableau original.

Méthodes utiles sur les tableaux JS : filter()

filter() : pour ne garder que certains éléments d'un tableau selon une condition. Elle renvoie un nouveau tableau avec uniquement les éléments qui passent le test.

```
let nombres = [1, 2, 3];
let pairs = nombres.filter(function(n) {
     return n % 2 === 0;
});
// console.log(pairs)
```

Autre exemple :

```
const ages = [32, 33, 16, 40];
const result = ages.filter(checkAdult);

function checkAdult(age) {
  return age >= 18;
}
```

Méthodes utiles sur les tableaux JS : reduce()

reduce() : transforme un tableau en une seule valeur.

.reduce() sert à tout résumer en une seule valeur : une somme, un produit, une moyenne, une chaîne de caractères, un objet, etc.

```
let nombres = [1, 2, 3];
let total = nombres.reduce(function(acc, n) {
      return acc + n;
}, 0);
console.log(total); // Résultat : 6
// acc = accumulateur (la valeur en cours)
// 0 = valeur de départ
```

Autre exemple

```
let mots = ["Hello", "world"];
let phrase = mots.reduce((acc, mot) =>
acc + " " + mot);
console.log(phrase); // "Hello world"
```

Méthodes utiles sur les tableaux JS : récapitulatif

Méthode	À quoi ça sert
push()	Ajouter à la fin
pop()	Supprimer à la fin
unshift()	Ajouter au début
shift()	Supprimer au début
for	Parcourir (classique)
forEach()	Parcourir (plus lisible)
map()	Transformer chaque élément
filter()	Garder seulement certains éléments
reduce()	Fusionner tous les éléments en une seule valeur

Ressources

reduce() en JavaScript
https://www.devoreur2code.com/blog/javascript-reduce-method
https://www.youtube.com/watch?v=iDWtuWkuj8g&pp=0gcJCdgAo7VqN5tD
https://www.youtube.com/watch?v=snhaYhafUXE

map() en JavaScript
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map

filter() en JavaScript
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter

Tableaux et méthodes JS
https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Global_Objects/Array