JDeDev

Formation Javascript avancé

En partenariat avec

livecampus

Julien Demettre EIRL ▸ Expert Windev Freelance
+33 (0)6 49 24 41 62 ▸ julien@jdedev.fr ▸ www.jdedev.fr
Siège social ▸ 125 Rue de Viller, 54300 LUNEVILLE - FRANCE
SIRET ▸ 832 481 220 00024 | APE ▸ 5829C

JDeDev

Sommaire

Ternaire

L'opérateur ternaire ? : est un opérateur qui vous permet en une ligne de remplacer un bloc if/else et de stocker le résultat dans une variable

```
const affich = true

let test
if (affich){
    test="affich est à vrai"
}
else {
    test="affiche est à faux"
}
```

```
const affich = true

const test = (affich) ? "affich est à vrai":"affiche est à faux"
```

Le spread operateur

Le spread opérateur … est un opérateur qui vous permet d'obtenir une copie de surface d'un objet ou un tableau.

Les tableaux et les objets en JS sont géré par référence et non par valeur, ainsi lorsque vous mettez un tableau A = tableau B si vous modifiez le tableau A, le tableau B est modifié également.

```
const A = ['Pomme', 'Poire', 'Banane']
const B = A
B[2] = 'Fraise'
console.log(A, B);
```

Les tableaux A et B sont modifié car ils sont 2 variables différentes faisant référence à la même zone mémoire

```
const A = ['Pomme', 'Poire', 'Banane']
const B = [...A]
B[2] = 'Fraise'
console.log(A, B);
```

Ici B est une nouvelle variable faisant référence à une zone mémoire contenant une copie de surface de la zone mémoire pointé par A. Ainsi B est modifié mais pas A

La décomposition

La décomposition vous permet de récupérer une portion de tableau ou d'objet

```
const A = ['Pomme', 'Poire', 'Banane']
const [B, C] = A
console.log(A, B, C);
```

Ici B est le premier indice du tableau A et C le deuxième indice.

Cela marche également pour les objets

```
const user = {
    nom: "Demettre",
    prenom: "Julien",
    age: "40"
}
const { nom, age } = user
console.log(user, nom, age);
```

nom et age ont pour valeur les attributs nom et age décomposé de la variable user

Fonctions fléchées

Les fonctions fléchées sont un peu comme les fonctions anonymes mais elles n'ont pas leur propre contexte (this) elles utilisent celui de leur parent

```
const addition = (a,b) =>{
    return a+b
}
```

A noter si vous n'avez qu'une seule ligne pas besoin d'accolades ni de return

```
const addition = (a,b) => a+b
```

Égalité forte

L'opérateur === test à la fois la valeur et le type, == ne teste que les valeurs

```
console.log("3" === 3, "3" == 3);
```

De nombreuses fonctions sur les tableaux sont intéressantes en JS

Tableau LIFO

Pour ajouter ou retirer un élément dans un tableau LIFO (Last In First Out), comme une pile d'assiettes, on utilisera les méthodes push pour ajouter et pop pour retirer

```
const fruits = ['Pomme', 'Poire', 'Banane']
fruits.push('Fraise')
console.log(fruits);
console.log(fruits.pop())
```

On ajoute les éléments à la fin de la liste et on retire également les éléments de la fin

Tableau FIFO

Pour ajouter ou retirer un élément dans un tableau FIFO (First In First Out), comme une fille d'attente, on utilisera les méthodes push pour ajouter et shift pour retirer

```
const fruits = ['Pomme', 'Poire', 'Banane']
fruits.push('Fraise')
console.log(fruits);
console.log(fruits.shift())
```

Tableau Bufferisé

Pour ajouter ou retirer un élément dans un tableau buffurisé, comme les wagons d'un train, on utilisera les méthodes unshift pour ajouter et pop pour retirer

```
const fruits = ['Pomme', 'Poire', 'Banane']
fruits.unshift('Fraise')
console.log(fruits);
console.log(fruits.pop())
```

Suppressions d'éléments

Il existe deux manières de supprimer des éléments dans un tableau : l'opérateur delete et la méthode splice

```
const fruits = ['Pomme', 'Poire', 'Banane']
delete fruits[1]
console.log(fruits);
```

L'opérateur delete conserve la case du tableau mais la vide

```
const fruits = ['Pomme', 'Poire', 'Banane']
fruits.splice(1, 1)
console.log(fruits);
```

La méthode splice supprime non seulement la valeur de la case du tableau mais également la case mémoire en elle-même

Tests sur les tableaux

Il existe 3 méthodes pour faire des tests sur les valeurs d'un tableau : includes, some et every

```
const fruits = ['Pomme', 'Poire', 'Banane']
console.log(fruits.includes('Pomme'));
```

Includes permet de savoir de manière simple si une valeur est présente dans un tableau

```
const fruits = ['Pomme', 'Poire', 'Banane']
console.log(fruits.some((fruit) => fruit[0] === 'P'));
```

Some prends en paramètre une fonction de test, ici nous vérifions si au moins un élément du tableau commence par le caractère P

```
const fruits = ['Pomme', 'Poire', 'Banane']
console.log(fruits.every((fruit) => fruit[0] === 'P'));
```

Every prends en paramètre une fonction de test, ici nous vérifions si tous les éléments du tableau commencent par le caractère P

Tri des tableaux

Il existe deux méthodes pour trier les tableaux : reverse et sort

```
const fruits = ['Pomme', 'Poire', 'Banane']
fruits.reverse()
console.log(fruits);
```

La méthode reverse ne prends pas de paramètre et inverse complètement l'ordre du tableau, les premiers seront les derniers….

```
const fruits = ['Pomme', 'Poire', 'Banane']
fruits.sort((a,b)=>a.localeCompare(b))
console.log(fruits);
```

La méthode sort prends en paramètre une fonction de trie qui doit retourner un entier positif (si a doit aller après b) ou négatif si a doit aller avant b

Recherche dans les tableaux

Il existe deux méthodes principales de recherche dans un tableau : find et findIndex

```
const fruits = ['Pomme', 'Poire', 'Banane']
console.log(fruits.find((fruit) => fruit[0] === "P"))
```

Find prends en paramètre une fonction de recherche et renvoie la première valeur vraie présente dans le tableau

```
const fruits = ['Pomme', 'Poire', 'Banane']
console.log(fruits.findIndex((fruit) => fruit[0] === "P"))
```

FindIndex prends en paramètre une fonction de recherche et permet de récupérer l'index de la case du tableau du premier élément dont la condition de recherche est vraie

Filtre des tableaux

La méthode filtrer permet de récupérer une portion du tableau

```
const fruits = ['Pomme', 'Poire', 'Banane']
console.log(fruits.filter((fruit) => fruit[0] === "P"))
```

Parcours des tableaux

Il existe 3 méthodes de parcours des tableaux : forEach, map et reduce

```
const fruits = [1,2,3,4]
fruits.forEach((fruit)=> console.log(fruit))
```

ForEach parcours tout le tableau et permet d'effectuer des traitements standards

```
const fruits = [1, 2, 3, 4]
const bigFruits = fruits.map((fruit) => fruit * 2)
console.log(bigFruits);
```

Map permet de parcourir un tableau, d'effectuer des traitements sur chaque case et renvoie un tableau contenant le résultat du traitement sur chaque case

```
const fruits = [1, 2, 3, 4]
const bigFruits = fruits.reduce((accumulateur, valeur) => accumulateur + valeur)
console.log(bigFruits);
```

Reduce prends en paramètre une fonction de deux paramètres : un accumulateur qui commence à 0 et la valeur de la case du tableau, il permet de faire des opérations sur un tableau et renvoie la valeur finale de l'accumulateur

Récupérations d'un tableau de clés

```
const user = {
    nom: "Demettre",
    prenom: "Julien",
    age: 40
}
console.log(Object.keys(user));
```

Grâce à la méthode Object.keys, vous pouvez récupérer les clés de n'importe quel objet sous forme d'un tableau

Récupération d'un tableau de valeurs

```
const user = {
    nom: "Demettre",
    prenom: "Julien",
    age: 40
}
console.log(Object.values(user));
```

Grâce à la méthode Object.values vous pouvez récupérer les valeurs de n'importe quel objet sous forme d'un tableau

Objet comme tableau associatif

```
const user = {}
user["nom"] = "Demettre"
user["prenom"] = "Julien"
user.age = 40
console.log(user);
```

Les valeurs d'un objet peuvent être récupérés ou assigné soit à la manière d'un objet soit à la manière d'un tableau associatif. Ce fonctionnement est super utile pour supprimer des doublons d'un tableau, ou pour optimiser le parcours des tableaux en utilisant des objets au lieu de tableaux pour accéder directement à la clé sans avoir à faire de parcours du tableau (je l'utilise très souvent sur codingame)

ENTRAINEZ VOUS

A partir du code fournis (tableau d'utilisateur et affichage des cards

Faites une fonction de trie qui permet d'afficher les cards utilisateur par ordre alphabétique du nom de famille

Faites une fonction de filtre qui n'affiche que les utilisateurs dont le code postal commence par 75