Javascript Array Methods

Le guide ultime du Ninja

Éditons 2020 par Nicolas Fazio

Javascript Array Methods

Le guide ultime du Ninja

Introduction

Présentation

Salut je m'appelle Nicolas Fazio, je suis l'auteur de cet ouvrage. Je suis Ingénieur en architecture software et je développe des applications pour les entreprises depuis plus de 15 ans.
Cela fait maintenant 4 ans que j'interviens en tant que formateur pour des formations Javascript & Typescript.

C'est notamment pour mes élèves que j'ai eu envie de créer cet outil afin de regrouper de manière simple et fonctionnelle, les principales méthodes existant pour les objets du type Tableau (Array) dans le langage Javascript.

Car connaitre et savoir utiliser les tableaux et les méthodes attachées est indispensable pour ne pas perdre de temps quand tu codes.

Contenu

Avec le temps, j'ai pu identifier 12 méthodes incontournables en Javascript et qu'il faut maitriser pour mieux coder et améliorer la lisibilité de ton code!

Dans cet ouvrage tu vas retrouver ces méthodes indispensables à connaitre et que l'on utilise couramment dans le langage de programmation Javascript mais aussi Typescript.

Je t'explique simplement avec des exemples concrets et réutilisables comment utiliser ces méthodes. Je te présent aussi la syntaxe qui est

disséquée en profondeur pour que tu aies bien toutes les informations en mains pour les utiliser dans tes projets ou dans ta vie de tous les jours au travail.

Tu as aussi pour chaque chapitre une petite démo online avec l'éditeur en ligne Stackblitz qui te permet de directement voir et tester le code que je te mets en exemple, pour mieux comprendre avec une situation concrète.

Et tu retrouveras aussi en bonus à la fin ce livre, la liste complète vers la documentation de référence de chaque méthode présentée (documentation en français).

J'espère que cet ouvrage te sera utile et t'aidera à améliorer la productivité et la lisibilité de ton code. Si tu veux plus de conseils et d'astuces vas vite consulter mon blog et abonnes-toi à la newsletter pour ne rater aucune info.

https://nicolasfazio.ch/blog

Table des matières

Array ForEach

C’est quoi la méthode Array ForEach( )

Array ForEach est une méthode de l’objet Array qui a été introduit depuis Javascript ES5 (ECMAScript 5). Elle est supportée par tous les navigateurs.

Array ForEach permet d’effectuer une itération des éléments contenus dans un tableau Javascript. Contrairement à de nombreuses autres méthodes d’itération, Array ForEach ne retourne pas de valeur en fin d’exécution et ne modifie pas les données contenues dans le tableau parcouru. Les valeurs retournées lors de l’itération sont toujours « *undefined* » sauf si tu ajout une méthode de traitement spécifique comme argument optionnel lors de l’appel de la méthode.

Array ForEach permet donc uniquement à accéder aux valeurs d’un tableau de manière itérative et il n’est pas possible de stopper l’itération du tableau, contrairement aux méthodes every( ), some( ) etc…

Définition et Syntaxe

Voici la définition syntaxique standard de cette méthode:

```
Array.forEach(callback, thisArg);
```

Et voici l'explication détaillé:

- **callback** - le premier argument est une fonction *callback* qui contient trois paramètres:
  - La valeur de l'élément du tableau en cours de traitement.
  - L'index *(optionnel)* de l'élément du tableau en cours de traitement
  - Le tableau *(optionnel)* sur lequel la méthode forEach est appliquée.
- **thisArg** *(optionnel)* - Le deuxième argument est la valeur à utiliser pour *this* lors de l'exécution de la fonction *callback.*

La valeur de retour de cette méthode est toujours *undefined.*

Implementation

Voici un exemple de code qui utilise forEach pour l'itération d'un tableau:

```
const items = ['a', 'b', 'c', 'd']
items.forEach((item, index) => {
  console.log(item, index);
});
```

Cet exemple affichera quelque chose comme cela dans la console:

```
a 0
b 1
c 2
d 3
```

Cela permet d'accéder aux valeurs du tableau et à l'index de l'élément parcouru.

Quand utiliser la méthode Array ForEach( )

La méthode ForEach( ) s'utilise quand tu veux exécuter une fonction donnée sur chaque élément d'un tableau.

Comment utiliser la méthode Array ForEach( )

Voici un exemple d'utilisation concrète que l'on retrouve souvent dans la construction d'élément du DOM. Ici tu vois comment afficher le contenu d'une liste d'élément dans une balise *<ul>.* Le code suivant va créer un <li> pour chaque élément de la liste et l'insérer à l'intérieur d'une balise *<ul>* avec l'identifiant *list* pour construire une liste HTML:

```
const list = document.querySelector('ul#list');
const items = [
  {id: 1, name: '🍌', quantity: 10},
  {id: 2, name: '🍓', quantity: 4},
```

```
  {id: 3, name: '🍋', quantity: 15},
  {id: 4, name: '🍏', quantity: 5}
];
items.forEach((item, index) => {
  list.innerHTML += `
    <li>${item.name}</li>
  `;
});
```

Lien vers la demo:
https://stackblitz.com/edit/ebook-js-array-ninja-foreach?file=index.js

Résumé

Voilà comment fonctionne la méthode Array ForEach( ) qui te permet d'effectuer une itération simple d'un tableau à la façon d'une boucle et d'appliquer une fonction sur chaque élément.

- Lors de l'itération, la méthode ForEach( ) ne retourne pas d'élément ce qui fait que la valeur de retour est toujours *undefined*.
- Il n'est pas possible de stopper l'itération de la boucle.
- Il n'est pas possible d'effectuer une intégration à l'envers.
- Tu peux accéder au tableau complet de données avec le troisième argument de la méthode *callback*.

- Tu peux utiliser les « *arrow function* » lors de l'utilisation mais rappel-toi que cela peut engendrer de conflit de scoop (*this*) si tu utilise l'argument optionnel *thisArg*.
- Lors de l'itération, les éléments avec des valeurs vides seront passés et donc ignorés.

Array Map

C’est quoi la méthode Array Map( )

Array Map est une méthode de l’objet Array qui a été introduit depuis Javascript ES5 (ECMAScript 5). Elle est supportée par tous les navigateurs

Array Map permet d’effectuer une itération, dans l'ordre du tableau, des éléments contenus dans un tableau Javascript et de retourner une nouvelle valeur pour chacun des éléments parcouru. Le résulta retourné par la méthode Map( ) est également un nouveau tableau ce qui permet de créer un nouveau tableau à partir d’un tableau existant.
Array Map peut par exemple être utilisé pour retourner une propriété spécifique des objets contenus dans un tableau.
La fonction *callback* sera exécutée pour tous les éléments contenus dans le tableau, y compris pour les valeurs *null* ou *undefined*.

Définition et Syntaxe

Voici la définition syntaxique standard de cette méthode:

```
const newArray = Array.map(callback, thisArg);
```

Et voici l'explication détaillé:

- **callback** - le premier argument est une fonction callback qui contient trois paramètres:
  - La valeur de l'élément du tableau en cours de traitement.
  - L'index (*optionnel)* de l'élément du tableau en cours de traitement
  - Le tableau *(optionnel)* sur lequel la méthode forEach est appliquée.
- **thisArg** *(optionnel)* - Le deuxième argument est la valeur à utiliser pour *this* lors de l'exécution de la fonction *callback.*

Implementation

Voici un exemple de code qui utilise map( ) pour l'itération d'un tableau:

```
const items = [1, 2, 3, 4];
const mapped =  items.map((item) => item * 2);
console.log(item);
console.log(mapped);
```

Cet exemple affichera quelque chose comme cela dans la console:

```
[1, 2, 3, 4];
[2, 4, 6, 8];
```

Cela permet d'accéder aux valeurs du tableau et d'y appliquer un traitement. Le tableau ainsi retourné par la méthode Array.Map( ) est un

nouveau tableau et il est donc possible de « chaîner » l'exécution de plusieurs méthodes de traitement. La méthode Array Map( ) ne modifie pas le tableau original.

Quand utiliser la méthode Array Map( )

La méthode Array Map( ) s'utilise quand tu veux crée un nouveau tableau avec les résultats de l'appel d'une fonction sur chaque élément d'un tableau.

Comment utiliser la méthode Array Map( )

Voici un exemple d'utilisation concrète que l'on retrouve souvent dans la construction d'élément du DOM. Ici tu vois comment afficher le contenu d'une liste d'élément dans une balise *<ul>.* Le code suivant va créer un <li> pour chaque élément de la liste et l'insérer à l'intérieur d'une balise *<ul>* avec l'id *list* pour construire une liste HTML. On utilise enfin la méthode *Array.join( )* pour convertir le tableau en chaine de caractère pour l'implémenter dans le DOM avec *innerHTML*:

```
const list = document.querySelector('ul#list');
const items = [
  {id: 1, name: '🍌', quantity: 10},
  {id: 2, name: '🍓', quantity: 4},
  {id: 3, name: '🍋', quantity: 15},
```

```
  {id: 4, name: '🍏', quantity: 5}
];
list.innerHTML = items.map((item) => `
  <li>${item.name}</li>
`).join('');
```

Lien vers la demo:
https://stackblitz.com/edit/ebook-js-array-ninja-map

Résumé

Tu a vu comment fonctionne la méthode Array Map( ) qui permet d'effectuer une itération simple d'un tableau à la façon d'une boucle.

- Lors de l'itération, la méthode Map( ) retourne chaque élément ce qui fait que la valeur de retour peut être modifié dans le but de créer un nouveau tableau.
- La méthode Map( ) retourne par défaut un nouveau tableau sans modifier le tableau d'origine.
- N'oublie pas de retourner une valeur lors de l'itération des éléments. Sinon tu aura un nouveau tableau rempli de *undefined*.
- Comme pour Array ForEach, il n'est pas possible de stopper l'itération de la boucle.
- Il n'est pas possible d'effectuer une intégration à l'envers.
- Tu peux accéder au tableau complet de données avec le troisième argument de la méthode *callback*.

- Tu peux utiliser les « *arrow function* » lors de l'utilisation mais rappel-toi que cela peut engendrer de conflit de scoop (*this*) si tu utilise l'argument optionnel *thisArg*.
- Lors de l'itération, les éléments avec des valeurs vides seront traités comme les autres éléments.

Array Find

C’est quoi la méthode Array Find( )

Array Find est une méthode de l’objet Array qui a été introduit depuis la version 1 de Javascript. Elle est supportée par tous les navigateurs

La méthode find( ) exécute une fonction une fois pour chaque élément présent dans le tableau jusqu'à ce qu'elle retourne une valeur vraie, et retourne immédiatement la valeur de l'élément. Si aucun élément n’est trouvé, la méthode Find( ) retourne *undefined*.

Définition et Syntaxe

Voici la définition syntaxique standard de cette méthode:

```
const element = Array.find(callback, thisArg);
```

Et voici l’explication détaillé:

- **callback** - le premier argument est une fonction callback qui contient trois paramètres:

    - La valeur de l'élément du tableau en cours de traitement.
    - L'index *(optionnel)* de l'élément du tableau en cours de traitement
    - Le tableau *(optionnel)* sur lequel la méthode forEach est appliquée.
- **thisArg** *(optionnel)* - Le deuxième argument est la valeur à utiliser pour *this* lors de l'exécution de la fonction *callback.*

Implementation

Voici un exemple de code qui utilise find( ):

```
const items = [1, 2, 3, 4];
const element = items.find(el => el === 2);
console.log(element);
// affichera: 2
```

Dans cet exemple on recherche l'élément du tableau dont la valeur est strictement égal à 2.

Quand utiliser la méthode Array Find( )

La méthode Array Find( ) s'utilise quand tu veux trouver un élément qui remplis une condition.

Comment fonctionne la méthodes Array Find( )

Voici un exemple concret qui permet d'identifier un élément dans un tableau pour l'isoler et effectuer un traitement dessus ultérieurement.

```
const items = [
  {id: 1, name: '🍌', quantity: 10},
  {id: 2, name: '🍓', quantity: 4},
  {id: 3, name: '🍋', quantity: 15},
  {id: 4, name: '🍏', quantity: 5}
];
const apple = items.filter(
  i => i.name === '🍏'
);
```

Le résultat contenu dans la constante *apple* est alors:

```
{id: 4, name: '🍏', quantity: 5}
```

Lien vers la démo:
https://stackblitz.com/edit/ebook-js-array-ninja-find

Résumé

Tu as vu comment fonctionne la méthode Array Find( ) qui retourne le premier élément qui remplis une condition déterminée par la fonction *callback*.

- La méthode Find( ) permet de retourner un élément spécifique à partir d'un tableau.
- La méthode Find( ) retourne un élément trouvé ou *undefined* sans modifier le tableau d'origine.
- N'oublie pas de retourner une valeur lors de l'itération des éléments.
- Lors du traitement, les éléments avec des valeurs vides seront ignoré.

Array Filter

C’est quoi la méthode Array Filter( )

Array Filter est une méthode de l’objet Array qui a été introduit depuis Javascript 1 (ECMAScript 1.6). Elle est supportée par tous les navigateurs

Array Filter crée et retourne un nouveau tableau contenant tous les éléments du tableau d'origine qui remplissent une condition déterminée par une condition que l’on renseigne dans la fonction *callback*. La fonction *callback* n'est utilisée que pour les éléments du tableau ayant une valeur assignée.
Les éléments du tableau qui ne passent pas le test effectué par la fonction *callback* sont ignorés, ils ne sont pas inclus dans le nouveau tableau.

Définition et Syntaxe

Voici la définition syntaxique standard de cette méthode:

```
const newArray = Array.filter(callback, thisArg);
```

Et voici l'explication détaillé:

- **callback** - le premier argument est une fonction *callback* qui contient trois paramètres:
  - La valeur de l'élément du tableau en cours de traitement.
  - L'index *(optionnel)* de l'élément du tableau en cours de traitement
  - Le tableau *(optionnel)* sur lequel la méthode *filter( )* est appliquée.
- **thisArg** *(optionnel)* - Le deuxième argument est la valeur à utiliser pour *this* lors de l'exécution de la fonction *callback.*

Implementation

Voici un exemple de code qui utilise filter( ):

```
const items = [true, false, false, true];
console.log(items);
console.log(items.filter((el) => el === true);
// affichera quelque chose comme:
// [true, false, false, true]
// [true, true]
```

Cela permet de créer un nouveau tableau contenant les éléments filtrés en fonction d'une condition spécifique. Ici on filtre tous les éléments d'un tableau qui sont égale à true. Tu remarque aussi que la valeur du tableau d'origine n'a pas été modifiée car Array Filter( ) crée un nouveau tableau à partir du tableau d'origine sans le modifier.

Quand utiliser la méthode Array Filter( )

La méthode Array Filter( ) s'utilise quand tu veux crée et retourner un nouveau tableau contenant tous les éléments d'un tableau d'origine qui remplissent une condition déterminée par la fonction *callback*.

Comment utiliser la méthode Array Filter( )

Voici un exemple d'utilisation concrète que l'on utilise pour filtrer les éléments d'un tableau en fonction d'une condition précise. Ici nous filtrons tous les éléments qui ont une quantité supérieure à 10:

```
const items = [
  {id: 1, name: '🍌', quantity: 10},
  {id: 2, name: '🍓', quantity: 4},
  {id: 3, name: '🍋', quantity: 15},
  {id: 4, name: '🍏', quantity: 5}
];

const fullStock = items.filter(i => i.quantity > 10);
```

Le résultat contenu dans la constante *fullStock* est alors:

```
[
  {id: 3, name: '🍋', quantity: 15}
];
```

Car c'est le seul élément dont la quantité est supérieur à 10.

Lien vers la démo:
https://stackblitz.com/edit/ebook-js-array-ninja-filter

Résumé

Tu as vu comment fonctionne la méthode Array Filter( ) qui retourne un nouveau tableau contenant tous les éléments du tableau d'origine qui remplissent une condition déterminée par la fonction callback.

- La méthode FIlter( ) permet de retourner un contenu spécifique à partir d'un tableau.
- La méthode Filter( ) retourne par défaut un nouveau tableau sans modifier le tableau d'origine.
- N'oublie pas de retourner une valeur lors de l'itération des éléments. Sinon tu aura un nouveau tableau remplis de *undefined*.
- La méthode Filter( ) est similaire à la méthode Array Find( ) mais est utilisé pour traiter plusieurs éléments. Elle retourne obligatoirement un tableau alors que Array Find( ) ne retourne qu'un seul élément, le premier qui remplis la condition.
- Tu peux accéder au tableau complet de données avec le troisième argument de la méthode *callback*.

- Tu peux utiliser les « *arrow function* » lors de l'utilisation mais rappel-toi que cela peut engendrer de conflit de scoop (*this*) si tu utilise l'argument optionnel *thisArg*.
- Lors du traitement, les éléments avec des valeurs vides seront ignoré.

Array IndexOf

C’est quoi la méthode Array IndexOf( )

Array IndexOf est une méthode de l’objet Array qui a été introduit depuis Javascript version (ECMAScript 1.6). Elle est supportée par tous les navigateurs

Array IndexOf compare un élément recherché aux éléments contenus dans un tableau en utilisant une égalité stricte. Ce qui permet savoir si cet élément est contenu dans le tableau et de récupérer l’index de position. Si aucun élément cordonnant n’est trouvé, la méthode retourne la valeur -1.

Définition et Syntaxe

Voici la définition syntaxique standard de cette méthode:

```
const newArray = Array.indexOf(dataToCheck,
startIndex);
```

Et voici l’explication détaillé:

- **dataToCheck** - L’élément ou la valeur qu'on cherche dans le tableau.
- **startIndex** *(optionnel)* - L'index à partir duquel commencer la recherche. La valeur par défaut est 0 (le tableau sera parcouru dans sa totalité). Si l'index est plus grand ou égal à la longueur du tableau, la méthode renverra -1. Si l'index est négatif, la recherche commencera d'autant d'éléments, à partir de la fin du tableau. À noter que même si l'index est négatif, la recherche s'effectue toujours du début jusqu'à la fin du tableau. Si l'index fourni est inférieur à 0, le tableau sera entièrement parcouru..

Implementation

Voici un exemple de code qui utilise indexOf( ):

```
const items = [1,2,3,4,5];
items.indexOf(3);
// affichera: 2
```

Cela permet de savoir si un élément recherché existe dans un tableau. Si élément recherché existe, la méthode retournera l’index de la position dans le tableau. Sinon elle retournera -1.

Quand utiliser la méthode Array IndexOf( )

La méthode Array IndexOf( ) s’utilise quand tu veux connaitre la position l’un élément recherché dans un tableau.

Comment fonctionne la méthode Array IndexOf( )

Voici un exemple d'utilisation concrète que l'on utilise pour connaitre la position d'un élément dans une liste. Si l'élément est trouvé, la méthode retournera la valeur de son index dans le tableau. Sinon elle retournera la valeur -1.

```
const items = ['🍌', '🍓', '🍋', '🍏'];

const indexFind = items.indexOf('🍓');

console.log(indexFind);

// affichera: 1
```

Résumé

Tu as vu comment fonctionne la méthode Array IndexOf( ) qui retourne la valeur de l'index d'un élément existant dans un tableau.

- La méthode IndexOf( ) permet de retourner l'index d'un élément.
- La méthode IndexOf( ) retourne -1 si aucun élément n'est trouvé.
- La méthode IndexOf( ) fonctionne pas avec les objets.
- Tu peux utiliser l'argument optionnel pour définir un index de départ lors du traitement de la méthode.
- La méthode IndexOf( ) ne modifie pas le tableau d'origine.

Array Reduce

C’est quoi la méthode Array Reduce( )

Array Reduce est une méthode de l’objet Array qui a été introduit depuis Javascript ES5 (ECMAScript 5). Elle est supportée par tous les navigateurs

Array Reduce applique une fonction callback qui est un « accumulateur » et qui traite chaque valeur d'une liste de gauche à droite, afin de les réduire à une seule valeur.
C’est une méthode qui fait un peut peur aux développeurs débutants et qui est souvent mal comprise. Pourtant cette méthode est extrêmement utile et pas difficile à utiliser.

Définition et Syntaxe

Voici la définition syntaxique standard de cette méthode:

```
const reduced = Array.reduce(callback, initValue);
```

Et voici l’explication détaillé:

- **callback** - le premier argument est une fonction callback exécuter sur chaque valeur de la liste (sauf le premier si aucune *initValue* n'est fournie), elle prend quatre arguments en entrée:
  - L'accumulateur qui est la valeur précédemment retournée par le dernier appel du *callback*, ou *initValue*, si elle est fournie, c'est la valeur « accumulée » au fur et à mesure des appels.
  - La valeur de l'élément du tableau en cours de traitement.
  - L'index *(optionnel)* de l'élément du tableau en cours de traitement
  - Le tableau *(optionnel)* sur lequel la méthode forEach est appliquée.
- **initVal** *(optionnel)* - Le deuxième argument est la valeur utilisée comme premier argument lors du premier appel de la fonction callback. Si aucune valeur initiale n'est fournie, le premier élément du tableau est utilisé (et la boucle de traitement ne le parcourra pas). Si on appelle reduce() sur un tableau vide sans fournir de valeur initiale, on aura une erreur.

Implementation

Voici un exemple de code qui utilise reduce( ):

```
const items = [1, 2, 3, 4];

const reduced = items.reduce((prev, next) => prev +
next, 0);

console.log(reduced);

// affichera quelque chose comme:

// 10
```

Cela permet d'obtenir facilement le somme total des valeurs contenu dans un tableau par exemple.

Quand utiliser la méthode Array Reduce( )

La méthode Array Reduce( ) s'utilise quand tu veux réduire à une seule valeur le contenu d'un tableau.

Comment fonctionne la méthode Array Reduce( )

Voici un exemple concret qui permet de calculer le stock total d'une liste d'élément qui contiennent la propriété *quantity:*

```
const items = [
  {id: 1, name: '🍌', quantity: 10},
  {id: 2, name: '🍓', quantity: 4},
  {id: 3, name: '🍋', quantity: 15},
  {id: 4, name: '🍏', quantity: 5}
];

const totalStock = items.reduce((prev, next) => prev
+ next.quantity, 0);
// la valeur de la constante totalStock est alors 34
console.log(totalStock);
```

```
// affichera 34
```

Lien vers la démo:
https://stackblitz.com/edit/ebook-js-array-ninja-reduce

Résumé

Tu as vu comment fonctionne la méthode Array Reduce( ) qui traite chaque valeur d'une liste afin de la réduire à une seule valeur

- Ne pas oublier de spécifier une valeur initial lors de l'utilisation de tableau d'objet.
- La méthode Reduce( ) ne modifie pas le tableau d'origine.
- N'oublie pas de retourner une valeur lors du traitement du callback.
- Tu peux accéder au tableau complet de données avec le troisième argument de la méthode *callback*.

Array Some

C’est quoi ma la méthode Array Some( )

Array Some est une méthode de l’objet Array qui a été introduit depuis Javascript ES5 (ECMAScript 5). Elle est supportée par tous les navigateurs.

La méthode Array Some( ) exécute une fonction callback une fois pour chaque élément présent dans le tableau jusqu'à ce qu'elle en trouve un élément pour lequel la fonction callback renvoie une valeur équivalente à *true*. Si un tel élément est trouvé, la méthode Array Some( ) renvoie immédiatement *true*. Dans le cas contraire, la méthode renvoie *false*.
La fonction callback n'est invoquée que pour les indices du tableau auxquels des valeurs sont assignées ; elle n'est pas invoquée pour les indices qui ont été supprimés ou auxquels aucune valeur n'a jamais été assignée.

Définition et Syntaxe

Voici la définition syntaxique standard de cette méthode:

```
const result = Array.some(callback, thisObj);
```

Et voici l’explication détaillé:

- **callback** - le premier argument est une fonction *callback* exécuter sur chaque valeur de la liste, elle prend trois arguments en entrée:
  - La valeur de l'élément du tableau en cours de traitement.
  - L’index *(optionnel)* de l'élément du tableau en cours de traitement
  - Le tableau *(optionnel)* sur lequel la méthode est appliquée.
- **thisObj** *(optionnel)* - Le deuxième argument correspond à la valeur à utiliser pour *this* lors de l'exécution de la fonction *callback.*

Implementation

Voici un exemple de code qui utilise reduce( ):

```
const items = [1, 2, 3, 4];

const plusGrandQue2 = items.some(x => x > 2);

console.log(plusGrandQue2);

// affichera quelque chose comme:

// true

const plusGrandQue5 = items.some(x => x > 5);
console.log(plusGrandQue5);

// affichera quelque chose comme:

// false
```

Cela permet de rapidement vérifier si un des éléments contenu dans un tableau remplis une condition spécifique.

Quand utiliser la méthode Array Some( )

La méthode Array Some( ) s'utilise quand tu veux tester si au moins un élément du tableau remplis une condition.

Comment fonctionne la méthode Array Some( )

Voici un exemple concret qui permet de savoir si un des élément présent dans le tableau remplis une condition. Ici on regard si un des éléments de la liste est « hors stock » :

```
const items = [
  {id: 1, name: '🍌', quantity: 10},
  {id: 2, name: '🍓', quantity: 4},
  {id: 3, name: '🍋', quantity: 15},
  {id: 4, name: '🍏', quantity: 5}
];

const haveOutOffStock = items.some(x => x.quantity <
= 0);
```

```
console.log(haveOutOffStock);
// affichera:  false
```

Si tu change la valeur « *quantity* » d'un des élément du tableau pour une valeur égale ou inférieur à 0, le résultat affiché sera alors *true*.

Lien vers la demo:
https://stackblitz.com/edit/ebook-js-array-ninja-some

Résumé

Tu as vu comment fonctionne la méthode Array Some( ) qui teste si au moins un élément du tableau passe le test implémenté par la fonction fournie. Elle renvoie un booléen indiquant le résultat du test.

- La méthode Some( ) ne modifie pas le tableau d'origine.
- N'oublie pas de retourner une valeur lors du traitement du callback pour éviter les bugs.
- Les valeurs vide sont ignorée.
- Tu peu accéder au tableau complet de données avec le troisième argument de la méthode *callback*.
- Tu peux modifier le scoop this avec le deuxième argument de la méthode.

Array Every

C’est quoi la méthode Array Every( )

Array Every est une méthode de l’objet Array qui a été introduit depuis Javascript ES5 (ECMAScript 5). Elle est supportée par tous les navigateurs

La méthode Array Every exécute la fonction *callback* sur chacun des éléments contenus dans le tableau jusqu'à ce qu'un élément pour lequel la fonction *callback* renvoie une valeur false. Si cela arrive, la méthode every renvoie directement false. Dans le cas contraire, la fonction callback renverra la valeur true si tous les éléments du tableau renvoient true à la condition. Les valeurs vides contenues dans le tableau serrant simplement ignorée.

Définition et Syntaxe

Voici la définition syntaxique standard de cette méthode:

```
const result = Array.every(callback, thisObj);
```

Et voici l’explication détaillé:

- **callback** - le premier argument est une fonction callback exécuter sur chaque valeur de la liste, elle prend trois arguments en entrée:
  - La valeur de l'élément du tableau en cours de traitement.
  - L'index *(optionnel)* de l'élément du tableau en cours de traitement
  - Le tableau *(optionnel)* sur lequel la méthode est appliquée.
- **thisObj** *(optionnel)* - Le deuxième argument correspond à la valeur à utiliser pour *this* lors de l'exécution de la fonction *callback.*

Implementation

Voici un exemple de code qui utilise reduce( ):

```
const items = [true, true, true];
const isAllTrue = items.every(x => x === true);
console.log(isAllTrue)
// affichera: true
```

Si tu modifie une valeur du tableau pour *false*, la valeur de retour de la fonction sera *false* et *isAllTrue* vaudra *false*.

Quand utiliser la méthode Array Every( )

La méthode Array Every( ) s'utilise quand tu veux tester si tous les éléments d'un tableau remplissent une condition.

Comment fonctionne la méthode Array Every( )

Voici un exemple concret qui permet de savoir si tous les éléments présents dans un tableau remplissent une condition. Ici on regard si un des éléments de la liste ont un stock supérieur à 10 :

```
const items = [
  {id: 1, name: '🍌', quantity: 10},
  {id: 2, name: '🍓', quantity: 4},
  {id: 3, name: '🍋', quantity: 15},
  {id: 4, name: '🍏', quantity: 0}
];

const isAllBiger10 = items.every(x => x.quantity > 10);
console.log(isAllBiger10);
// affiche false
```

Si tu modifie toutes les valeur *quantity* des éléments du tableau pour une valeur supérieur à 10, la valeur de retour (*isAllBiger10*) sera *true*.

Lien vers une demo:
https://stackblitz.com/edit/ebook-js-array-ninja-every

Résumé

Tu as vu comment fonctionne la méthode Array Every( ) qui permet de tester si tous les éléments d'un tableau vérifient une condition donnée par une fonction en argument. Cette méthode renvoie un booléen pour le résulta du test.

- La méthode Every( ) ne modifie pas le tableau d’origine.
- N’oublie pas de retourner une valeur lors du traitement du callback pour éviter les bugs.
- Les valeurs vide sont ignorée.
- Tu peux accéder au tableau complet de données avec le troisième argument de la méthode *callback*.
- Tu peux modifier le scoop this avec le deuxième argument de la méthode.

Array From

C’est quoi la méthode Array From( )

Array From est une méthode de l’objet Array qui a été introduit depuis Javascript ES6 (ECMAScript 2015). Elle est supportée par tous les navigateurs sauf Internet Explorer.

La méthode Array From permet de créer une instance de tableau (Array) à partir d'objets semblables à des tableaux, qui disposent d'une propriété *length* ou des éléments indexés ou d’objet itérables comme Map ou Set.

Définition et Syntaxe

Voici la définition syntaxique standard de cette méthode:

```
const result = Array.from(arrayLike, mapFn,
thisObj);
```

Et voici l’explication détaillé:

- **arrayLike** - Un objet semblable à un tableau ou bien un objet itérable dont on souhaite créer un tableau
- **mapFn** (optionnel) - Argument optionnel, une fonction à appliquer à chacun des éléments du tableau.
- **thisObj** *(optionnel)* - Le deuxième argument correspond à la valeur à utiliser pour *this* lors de l'exécution de la fonction *mapFn.*

Implementation

Voici un exemple de code qui utilise from( ):

```
const result = Array.from('Javascript');

console.log(result)

// affichera:

// ["J", "a", "v", "a", "s", "c", "r", "i", "p",
"t"]
```

Cet exemple va créer un tableau à partir d'une chaine de caractères.

Quand utiliser la méthode Array From( )

La méthode Array From( ) s'utilise quand tu veux créer une nouvelle instance de tableau à partir d'un objet itérable ou semblable à un tableau.

Comment fonctionne la méthode Array From( )

Voici un exemple concret qui permet récupérer tous éléments d'une pages web (afficher dans le DOM) qui ont la class « *item* » et de les modifier un par un en utilisant les méthode prototype Array:

```
const itemsDOM = document.querySelectorAll('.item');
const items = Array.from(itemsDOM);
const itemUpdated = items.map(i => {
  i.innerHTML += i.innerHTML
  return i;
});
```

Lien ver la démo:
https://stackblitz.com/edit/ebook-js-array-ninja-from

Si tu essaye d'utiliser la méthode Map( ) avec la constante itemsDOM, tu verra une erreur qui s'affiche qui t'explique que la fonction Map( ) n'est disponible que pour les objets de type Array. C'est donc un manière simple de convertir une liste d'élément HTML en objet Array natif Javascript.

Résumé

Tu as vu comment fonctionne la méthode Array From( ) qui permet de créer une nouvelle instance d'Array à partir d'un objet itérable ou semblable à un tableau.

- La méthode Array Form génère un tableau qui pointe sur les éléments de la liste d'itérabe d'origine.
- Cette méthode est pratique pour permettre l'utilisation des méthode *prototype.array* sur une liste itérabe qui n'est pas de type Array.

Array Sort

C’est quoi la méthode Array Sort( )

Array Sort est une méthode de l’objet Array qui a été introduit depuis Javascript 1st Edition. Elle est supportée par tous les navigateurs mais sont implémentation diffère pour les version de Javascript 1.2.

La méthode Array Sort permet de trier les éléments d'un tableau, dans ce même tableau, et renvoie le tableau. Par défaut, le tri s'effectue sur les éléments du tableau convertis en chaînes de caractères et triées selon les valeurs des unités de code UTF-16 des caractères.

Définition et Syntaxe

Voici la définition syntaxique standard de cette méthode:

```
const result = Array.sort(callback);
```

Et voici l’explication détaillé:

- **callback** *(optionnel)* - Ce paramètre optionnel permet de spécifier une fonction définissant l'ordre de tri. Si absente, le tableau est trié selon la valeur de point de code Unicode de chaque caractère, d'après la conversion en chaine de caractères de chaque élément. Cette fonction prendra deux arguments:
  - Le premier élément à comparer.
  - Le deuxième élément à comparer.

Implementation

Voici un exemple qui utilise array.sort( )

```
const items = [4, 2, 5, 1, 3];
items.sort();
console.log(items);
// affichera ceci: [1, 2, 3, 4, 5]
```

Dans cet exemple, on trie les valeur du tableau par ordre ascendant.

Quand utiliser la méthode Array Sort( )

La méthode Array Sort( ) s'utilise quand tu veux trier les éléments d'un tableau.

Comment fonctionne la méthode Array Sort( )

Voici un exemple concret qui permet trier tous éléments d'une liste de commission par ordre de quantité:

```
const items = [
  {id: 1, name: '🍌', quantity: 10},
  {id: 2, name: '🍓', quantity: 4},
  {id: 3, name: '🍋', quantity: 15},
  {id: 4, name: '🍏', quantity: 0}
];
items.sort((a, b) => a.quantity - b.quantity);
console.log(items);
// affichera ceci:
const items = [
  {id: 4, name: '🍏', quantity: 0},
  {id: 2, name: '🍓', quantity: 4},
  {id: 1, name: '🍌', quantity: 10},
  {id: 3, name: '🍋', quantity: 15},
];
```

Lien vers la démo:
https://stackblitz.com/edit/ebook-js-array-ninja-sort

Résumé

Tu as vu comment fonctionne la méthode Array Sort( ) qui permet de trier les éléments d'un tableau en utilisant une fonction de tris.

- La méthode Array Sort ne génère pas de nouveau tableau. Elle modifie l'ordre d'itération du tableau d'origine.
- Pour des chaines de caractères contenant des caractères spéciaux ou des accents (é, è, a, ä, etc.): il faut utiliser *String.localeCompare( )*. Cette fonction permet de comparer les chaines de caractères spéciaux.

Array Flat

C’est quoi la méthode Array Flat( )

Array Flat est une méthode de l’objet Array qui a été introduit depuis Javascript 2020 (ECMAScript 2019). Elle est supportée par tous les navigateurs, sauf Internet Explorer.

La méthode Array Flat permet « d’aplatir » un tableau qui contient d’autres tableaux afin de mettre au même niveau tous les valeurs.

Définition et Syntaxe

Voici la définition syntaxique standard de cette méthode:

```
const result = Array.flat(arg);
```

Et voici l’explication détaillé:

- **arg** *(optionnel)* - Ce paramètre optionnel permet de spécifier le niveau de profondeur en termes d'imbrication de tableau. Autrement dit,

jusqu'à quel niveau d'imbrication un tableau imbriqué doit il être aplati. La valeur par défaut est 1.

Implementation

Voici un exemple qui utilise array.flat( )

```
const items = [1, 2, [3, 4]];
const flatted = items.flat();
console.log(flatted);
// affichera ceci: [1, 2, 3, 4, 5]
```

Quand utiliser la méthode Array Flat( )

La méthode Array Flat( ) s'utilise quand tu veux créer un nouveau tableau à partit des valeurs d'un tableau qui contient des tableaux.

Comment fonctionne la méthodes Array Flat( )

Voici un exemple concret qui permet créer une nouveau tableau à partir des valeurs contenues dans les sous-tableaux du tableau d'origine.

```
const lists = [
  [
       {id: 1, name: '🍌', quantity: 10},
```

```
        {id: 2, name: '🍓', quantity: 4}
    ],
    [
        {id: 3, name: '🍋', quantity: 15},
        {id: 4, name: '🍏', quantity: 0}
    ]
];
const items = lists.flat();
console.log(items);
// affichera
[
  {id: 1, name: '🍌', quantity: 10},
  {id: 2, name: '🍓', quantity: 4},
  {id: 3, name: '🍋', quantity: 15},
  {id: 4, name: '🍏', quantity: 0}
];
```

Lien vers la démo:
https://stackblitz.com/edit/ebook-js-array-ninja-flat

Résumé

Tu as vu comment fonctionne la méthode Array Flat( ) qui permet d'aplatir un tableau qui contient des sous-tableaux.

- La méthode Array Flat ne modifie pas le tableau d'origine.
- La méthode Array Flat retour une nouvelle instance de tableau contenant les élément du tableau d'origine.
- L'argument de la méthode Flat permet de définir la profondeur, le niveau d'aplatissement voulu.

Array FlatMap

C’est quoi la méthode Array FlatMap( )

Array FlatMap est une méthode de l’objet Array qui a été introduit depuis Javascript 2020 (ECMAScript 2019). Elle est supportée par tous les navigateurs, sauf Internet Explorer.

La méthode flatMap( ) est identique à un appel de la méthode Array.prototype.map( ) suivi d'un appel de Array.prototype.flat( ) avec une profondeur de 1.

Définition et Syntaxe

Voici la définition syntaxique standard de cette méthode:

```
const result = Array.flatMap(callback, thisObj);
```

Et voici l’explication détaillé:

- **callback** - le premier argument est une fonction callback exécuter sur chaque valeur de la liste, elle prend trois arguments en entrée:

- La valeur de l'élément du tableau en cours de traitement.
- L'index *(optionnel)* de l'élément du tableau en cours de traitement
- Le tableau *(optionnel)* sur lequel la méthode est appliquée.

- **thisObj** *(optionnel)* - Le deuxième argument correspond à la valeur à utiliser pour *this* lors de l'exécution de la fonction *callback.*

Implementation

Voici un exemple qui utilise array.flatMap( )

```
const message = ["Hello", "", "Javascript Student"]
console.log(message.map(x => x.split(" ")));
// affichera ceci:
[["Hello"], [""], ["Javascript"], ["Student"]]
```

alors qu'avec flatMap()

```
console.log(message.flatMap(x => x.split(" ")));
// le résultat affichera ceci:
["Hello", "", "Javascript", "Student"]
```

Quand utiliser la méthode Array FlatMap( )

La méthode Array FlatMap( ) s'utilise quand tu veux d'appliquer une fonction à chaque élément du tableau puis aplatir le résultat en un tableau.

Comment fonctionne la méthodes Array FlatMap( )

Voici un exemple concret qui permet créer une nouveau tableau à partir des valeurs contenues dans les sous-tableaux des objets contenu dans tableau d'origine. Le but est de regrouper dans une seul liste la totalité des éléments acheter par la liste des utilisateurs:

```
const users = [
  {
    name: 'Fazio',
    card: [
      {id: 1, name: '🍌', quantity: 10},
      {id: 2, name: '🍓', quantity: 4}
    ]
  },
  {
    name: 'Jean',
    card: [
      {id: 3, name: '🍋', quantity: 15},
      {id: 4, name: '🍏', quantity: 0}
    ]
  }
```

```
];
const allItemsInCard = users.flatMap((u) => u.card);
console.log(allItemsInCard);
// affichera ceci:
[
   {id: 3, name: '🍋', quantity: 15},
   {id: 4, name: '🍏', quantity: 0},
   {id: 1, name: '🍌', quantity: 10},
   {id: 2, name: '🍓', quantity: 4}
];
```

Lien vers la démo:
https://stackblitz.com/edit/ebook-js-array-ninja-flatmap

Résumé

Tu as vu comment fonctionne la méthode Array FlatMap( ) qui permet d'aplatir un tableau qui contient des sous-tableaux.

- La méthode Array FlatMap ne modifie pas le tableau d'origine.
- La méthode Array FlatMap retour une nouvelle instance de tableau contenant les élément du tableau d'origine.
- L'argument de la méthode FlatMap ne permet pas d'aplatir plus que 1 niveau de profondeur.

Ressources

Démo online sur Stackblitz

- Foreach: https://stackblitz.com/edit/ebook-js-array-ninja-foreach
- Map: https://stackblitz.com/edit/ebook-js-array-ninja-map
- Find: https://stackblitz.com/edit/ebook-js-array-ninja-find
- Filter: https://stackblitz.com/edit/ebook-js-array-ninja-filter
- IndexOf: https://stackblitz.com/edit/ebook-js-array-ninja-indexof
- Reduce: https://stackblitz.com/edit/ebook-js-array-ninja-reduce
- Some: https://stackblitz.com/edit/ebook-js-array-ninja-some
- Every: https://stackblitz.com/edit/ebook-js-array-ninja-every
- From: https://stackblitz.com/edit/ebook-js-array-ninja-from
- Sort: https://stackblitz.com/edit/ebook-js-array-ninja-sort
- Flat: https://stackblitz.com/edit/ebook-js-array-ninja-flat
- FlatMap: https://stackblitz.com/edit/ebook-js-array-ninja-flatmap

Documentation MDN

- Foreach: https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/forEach
- Map: https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/map

- Find: https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/find
- Filter: https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/filter
- IndexOf: https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/indexOf
- Reduce: https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/reduce
- Some: https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/some
- Every: https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/every
- From: https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/from
- Sort: https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/sort
- Flat: https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/flat
- FlatMap: https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/flatMap

Notes perso

Ici tu peux écrire les remarques ou astuces que tu découvres et qui ne sont pas dans ce livre.