

# Le constructeur Array() : propriétés et méthodes

# Table des matières

<b>I. Contexte</b>	<b>3</b>
<b>II. Parcourir un tableau JavaScript</b>	<b>3</b>
<b>III. Exercice : Appliquez la notion</b>	<b>7</b>
<b>IV. Méthodes pour trouver, filtrer et trier</b>	<b>7</b>
<b>V. Exercice : Appliquez la notion</b>	<b>13</b>
<b>VI. Méthodes pour manipuler (copie, fusion...)</b>	<b>13</b>
<b>VII. Exercice : Appliquez la notion</b>	<b>17</b>
<b>VIII. Auto-évaluation</b>	<b>18</b>
A. Exercice final .....	18
B. Exercice : Défi .....	19
<b>Solutions des exercices</b>	<b>20</b>

## I. Contexte

**Durée :** 1 h

**Environnement de travail :** Repl.it

**Pré-requis :** Connaître les bases sur les tableaux JavaScript

### Contexte

Vous maîtrisez les bases de la création et de l'utilisation des tableaux JavaScript via l'objet `Array`. À présent, nous allons voir ensemble plus en détails les propriétés et méthodes dont il dispose. Cela vous permettra, lors de vos développements, de choisir la méthode qui répond le mieux à votre besoin.

## II. Parcourir un tableau JavaScript

### Objectifs

- Connaître les différentes possibilités pour parcourir un tableau JavaScript
- Utiliser la méthode répondant le mieux à son besoin

### Mise en situation

Vous savez comment parcourir un tableau avec `for` (à l'aide de la propriété `length`), puis avec `forEach`. Dans ce cours, vous allez apprendre qu'il existe des méthodes offrant d'autres possibilités pour travailler avec les tableaux.

### Rappel Les tableaux JavaScript

#### L'instruction `for` et la propriété `length`

L'instruction JavaScript `for` associée à la propriété `length` (indique le nombre d'éléments contenus dans le tableau) de l'objet `Array` permet de parcourir un tableau.

```
1 let animals = ['Lion', 'Chat', 'Chien', 'Cheval', 'Tigre', 'Ours']
2
3 // Display array length (6) in the console
4 console.log(animals.length)
5
6 // Displays the value of the array at index i in the console
7 for (let i = 0; i < animals.length; i++) {
8   console.log(animals[i])
9 }
```

#### La méthode `forEach()`

La méthode `forEach` quand à elle, permet d'exécuter une fonction sur chaque élément du tableau.

```
1 let animals = ['Lion', 'Chat', 'Chien', 'Cheval', 'Tigre', 'Ours']
2
3 // Displays each array element in the console
4 animals.forEach(function(element){
5   console.log(element)
6 })
7
8 // Displays each array index and his element in the console
9 animals.forEach(function(element, index){
10  console.log(index + ' : ' + element)
11 })
```

```
11 })
```

## Méthode Les autres méthodes permettant de parcourir un tableau JavaScript

### La méthode every ()

Cette-ci va s'appliquer sur l'ensemble des éléments d'un tableau afin de tester si la condition passée en paramètre (sous la forme d'une fonction) est vérifiée ou non.

Elle renvoie un booléen `true` (vrai) si **tous les éléments** du tableau remplissent la condition, sinon elle renvoie `false` (faux).

```
1 // Return true if the element is greater than zero, otherwise false.
2 function isGreaterThanZero(element) {
3     return element > 0
4 }
5
6 let numbers = [0, 9, 5, 4, 6, 8, 2]
7
8 // Display false in the console because ALL numbers are not strictly greater than 0 (0 is not
9 greater than 0)
10 console.log(numbers.every(isGreaterThanZero))
11
12 // Display true in the console because ALL elements are greater or equal to zero
13 console.log(numbers.every(function(element) {
14     return element >= 0
15 })))
```

### La méthode some ()

Va s'appliquer sur l'ensemble des éléments d'un tableau afin de tester si la condition passée en paramètre (sous la forme d'une fonction) est vérifiée ou non.

Elle renvoie un booléen `true` (vrai) si **au moins un des éléments** du tableau remplit la condition, sinon elle renvoie `false` (faux).

```
1 // Return true if the element is greater than zero, otherwise false.
2 function isGreaterThanZero(element) {
3     return element > 0
4 }
5
6 let numbers = [0, 9, 5, 4, 6, 8, 2]
7
8 // Display true in the console because at least one number is strictly greater than 0
9 console.log(numbers.some(isGreaterThanZero))
10
11 // Display false in the console because there is not at least one element strictly inferior to
12 zero
13 console.log(numbers.some(function(element) {
14     return element < 0
15 })))
```

### La méthode map ()

Cette méthode va appliquer une fonction sur l'ensemble des éléments d'un tableau, elle renvoie **un nouveau tableau** contenant uniquement les valeurs pour lesquelles la fonction en paramètre a renvoyé **true**.

Le tableau d'origine ne va donc pas être modifié.

```
1 // Returns the square of a given number
2 function squareNumber(number) {
3     return number * number
4 }
```

```

5
6 let numbers = [0, 9, 5, 4, 6, 8, 2]
7
8 // Create a new array containing squared numbers
9 let squaredNumbers = numbers.map(squareNumber)
10
11 // Create a new array containing numbers divided by 2, from the squaredNumbers array
12 let otherNumbers = squaredNumbers.map(function(element) {
13   return element / 2
14 })
15
16 // Display in the console of the numbers array
17 // [0, 9, 5, 4, 6, 8, 2]
18 console.log(numbers)
19
20 // Display in the console of the new array containing the squared numbers
21 // [0, 81, 25, 16, 36, 64, 4], (results of : 0*0, 9*9, ...)
22 console.log(squaredNumbers)
23
24 // Display in the console of the new array containing the numbers divided by 2
25 // [0, 40.5, 12.5, 8, 18, 32, 2], (results of : 0/2, 81/2, ...)
26 console.log(otherNumbers)

```

### La méthode `reduce()`

Elle va s'appliquer sur l'ensemble des éléments d'un tableau, à partir d'une fonction passée en paramètre elle va réduire la liste de valeurs à une valeur unique. Elle pourra servir par exemple pour additionner les valeurs d'un tableau.

Les valeurs sont parcourues de **gauche à droite**. Pour les parcourir de **droite à gauche** il faut utiliser la méthode `reduceRight()`.

La fonction passée en paramètre de la méthode `reduce()` prend 2 paramètres qui sont :

- `accumulator` : valeur retournée lors du dernier appel à la fonction, valeur accumulée au fur et à mesure que le tableau est parcouru.
- `currentValue` : valeur de l'élément en cours de traitement dans le tableau.

La méthode `reduce()` peut éventuellement prendre un deuxième paramètre qui est une valeur de départ qui sera utilisée avant la première valeur du tableau. Dans le cas d'une addition, on renseigne le plus souvent ce paramètre à **0**.

Cette méthode **retourne donc une seule valeur**.

```

1 // Returns the result of the addition of array accumulator and array current value.
2 // accumulator : value returned by sumReducer() of the elements already browsed in the array
3 // currentValue : value of the element being processed in the array
4 function sumReducer(accumulator, currentValue) {
5   return accumulator + currentValue
6 }
7
8 let numbers = [0, 9, 5, 4, 6, 8, 2]
9
10 // Display in the console the value returned by reduce : 34
11 // Step-by-step explanation (accumulator + currentValue) : 0 + 9 | 9 + 5 | 14 + 4 | 18 + 6 |
12 // 24 + 8 | 32 + 2
13 console.log(numbers.reduce(sumReducer))
14
15 // Example of a result with an initial value (6) passed in parameter : 40
16 // Step-by-step : 6 + 0 | 6 + 9 | 15 + 5 | 20 + 4 | 24 + 6 | 30 + 8 | 38 + 2
17 console.log(numbers.reduce(sumReducer, 6))

```

```

17
18 // Display -34 in the console (0 - 9 | -9 - 5 | -14 - 4 | ...)
19 console.log(numbers.reduce(function(accumulator, currentValue) {
20     return accumulator - currentValue
21 })))
22
23 // Display 66 in the console (initial value : 100), (100 - 0 | 100 - 9 | 91 - 5 | ...)
24 console.log(numbers.reduce(function(accumulator, currentValue) {
25     return accumulator - currentValue
26 }, 100))

```

## Syntaxe À retenir

**Choisissez la méthode de parcours qui correspond à votre cas avant de vous lancer dans le développement.**

- `length` : nombre d'éléments du tableau.
- `for` : instruction pour créer une boucle.
- `forEach()` : exécute une fonction sur chaque élément du tableau.
- `every()` : parcourt le tableau et renvoie `true` si tous les éléments remplissent une condition donnée, dans une fonction passée en paramètre, sinon `false`.
- `some()` : parcourt le tableau et renvoie `true` si au moins un des éléments remplit une condition donnée, dans une fonction passée en paramètre, sinon `false`.
- `map()` : parcourt le tableau et crée un nouveau tableau contenant les valeurs retournées par la fonction passée en paramètre.
- `reduce()` : parcourt le tableau et à partir d'une fonction passée en paramètre elle va réduire la liste de valeurs à une valeur unique.

## Complément

`length`<sup>1</sup>

`for`<sup>2</sup>

`forEach()`<sup>3</sup>

`every()`<sup>4</sup>

`some()`<sup>5</sup>

`map()`<sup>6</sup>

`reduce()`<sup>7</sup>

<sup>1</sup> [https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets\\_globaux/Array/length](https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/length)

<sup>2</sup> <https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Instructions/for>

<sup>3</sup> [https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets\\_globaux/Array/forEach](https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/forEach)

<sup>4</sup> [https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets\\_globaux/Array/every](https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/every)

<sup>5</sup> [https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets\\_globaux/Array/some](https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/some)

<sup>6</sup> [https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets\\_globaux/Array/map](https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/map)

<sup>7</sup> [https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets\\_globaux/Array/reduce](https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/reduce)

### III. Exercice : Appliquez la notion

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail :



#### Question

Dans cet exercice, imaginons que nous avons une liste de produits (par exemple des fruits), que pour chaque produit nous en avons une certaine quantité en stock, et que nous sommes en charge de la gestion des stocks.

À partir des quantités de produits en stock (15, 184, 29, 0, 35), vous allez devoir utiliser la méthode adéquate afin de vérifier si tous les produits sont toujours en stock et d'afficher l'information dans la console.

#### Indice :

Créer un tableau de ces valeurs.

Trouver la méthode permettant de savoir si elles sont toutes supérieures à 0.

Afficher le résultat dans la console avec `console.log()`.

### IV. Méthodes pour trouver, filtrer et trier

#### Objectifs

- Connaître les différentes méthodes pour trouver une valeur, filtrer ou trier un tableau JavaScript
- Utiliser la méthode répondant le mieux à son besoin

#### Mise en situation

Nous avons vu dans la première partie les différentes méthodes pour parcourir un tableau. Cependant, pour filtrer, trier ou trouver une valeur, il n'est pas forcément utile ni performant de le parcourir entièrement.

En effet, il existe des méthodes permettant d'effectuer ces actions : ce sont celles-ci que nous allons étudier à présent.

#### Méthode

##### La méthode `find()`

Retourne la **première valeur trouvée** répondant à une condition donnée, passée en paramètre sous la forme d'une fonction.

Si aucune valeur ne respecte la condition, alors la méthode renvoie : `undefined`.

Cette méthode ne parcourt donc pas forcément le tableau en entier, ce qui peut représenter un gain de performance !

```
1 function isGreaterThanTen(element) {  
2   return element > 10  
3 }  
4  
5 let numbers = [0, 9, 51, 100, 6, 18, 2]  
6  
7 // Display 51 in the console because it's the first value found to be greater than 10  
8 console.log(numbers.find(isGreaterThanTen))
```

<sup>1</sup> <https://repl.it/>

### La méthode `findIndex()`

Elle est similaire à `find()` sauf qu'au lieu de retourner la première valeur trouvée, elle va retourner **l'indice (index)** de la première valeur trouvée répondant à une condition donnée, passée en paramètre sous la forme d'une fonction.

Si aucune valeur ne respecte la condition, alors la méthode renvoie : `-1`.

Elle ne parcourt donc pas forcément le tableau en entier.

**`findIndex`** est une méthode très utilisée : cependant, le fait qu'elle renvoie `-1` pourrait être source de bugs dans le code d'un développeur débutant. Attention à bien tester le retour de vos fonctions et à ne pas utiliser le résultat tel quel !

```
1 function isGreaterThanTen(element) {
2   return element > 10
3 }
4
5 let numbers = [0, 9, 51, 100, 6, 18, 2]
6
7 // Display 2 in the console because it is the index of 51 which is the first value found to be
  greater than 10.
8 console.log(numbers.findIndex(isGreaterThanTen))
```

### La méthode `indexOf()`

Renvoie **l'index du premier élément trouvé** correspondant à l'élément recherché, passé en paramètre.

Si l'élément n'est pas trouvé, alors la méthode retournera : `-1`.

La différence avec `findIndex()` est donc qu'ici on ne passe pas de fonction, mais directement l'élément recherché.

Cette méthode accepte éventuellement un deuxième paramètre permettant de préciser l'index de départ, afin que la recherche ne commence pas au début du tableau.

```
1 let animals = ['Lion', 'Chat', 'Chien', 'Lion', 'Ours']
2
3 // Display 0 in the console because the first 'Lion' value found is at index 0, even if there
  is a second one at index 3.
4 console.log(animals.indexOf('Lion'))
5
6 // Display 3 in the console because the first 'Lion' value found when starting to browse from
  the index 2 is at index 3.
7 console.log(animals.indexOf('Lion', 2))
```

### La méthode `lastIndexOf()`

Celle-ci est similaire à `indexOf()` sauf qu'elle renvoie **l'index du dernier élément trouvé** correspondant à l'élément recherché, passé en paramètre.

Si l'élément n'est pas trouvé, alors la méthode retournera : `-1`.

**IMPORTANT :** Cette méthode parcourt le tableau **à l'envers**, c'est-à-dire en commençant par la valeur située au dernier index. Cependant, les index gardent leur valeur d'origine (le premier élément à gauche du tableau reste à l'index 0).

La méthode **`lastIndexOf()`** accepte éventuellement un deuxième paramètre permettant de préciser l'index de départ, afin que la recherche ne commence pas à la fin du tableau.

```
1 let animals = ['Lion', 'Chat', 'Chien', 'Lion', 'Ours']
2
3 // Display 3 in the console because the last 'Lion' value found is at index 3.
4 console.log(animals.lastIndexOf('Lion'))
5
6 // Display 0 in the console because the last 'Lion' value found when starting to browse from
  the index 1 is at index 0.
```



```

7 // As the search is from right to left and starts at index 1 ('Chat') then the last 'Lion'
  value found is at index 0.
8 console.log(animals.lastIndexOf('Lion', 1))

```

### La méthode `sort()`

Elle permet de trier un tableau. Elle renvoie ce même tableau trié.

**ATTENTION :** Cette méthode trie des chaînes de caractères, par conséquent si vous lui passez des nombres, ceux-ci seront convertis en chaînes de caractères et triés en tant que telles et non en tant que nombres. Vous n'aurez pas un tableau de nombres triés par ordre croissant ou décroissant.

Trier un tableau de nombres de 1 à 10 de cette manière donnera le résultat suivant : 1, 10, 2, 3, 4, 5, 6, 7, 8, 9. Ce n'est pas tout à fait idéal !

**sort()** peut éventuellement prendre en paramètre une fonction de comparaison des éléments permettant de déterminer l'ordre de tri :

- Utiliser cette fonction permet de pouvoir comparer des nombres (*voir l'exemple de code ci-dessous où nous effectuons une simple soustraction*).
- Cette fonction de comparaison prend 2 arguments en paramètres : le premier élément (a) et le deuxième élément (b) à comparer.
- Si le retour de cette fonction est inférieur à 0, alors "a" est trié avec un index inférieur à "b" ; si le retour est supérieur à 0, alors "b" est trié avec un index inférieur à "a" ; si le retour est égal à 0, alors "a" et "b" sont inchangés l'un par rapport à l'autre, mais triés par rapport aux autres éléments.

```

1 let animals = ['Lion', 'Chat', 'Chien', 'Cheval', 'Tigre', 'Ours']
2
3 let numbers = [0, 900, 1000, 11, 40, 6, 8, 2, 61]
4
5 // Display in the console : ['Chat', 'Cheval', 'Chien', 'Lion', 'Ours', 'Tigre']
6 console.log(animals.sort())
7
8 // Display in the console : [0, 1000, 11, 2, 40, 6, 61, 8, 900]
9 console.log(numbers.sort())
10
11 // Function for comparing the length of strings in order to sort them in ascending order
12 function compareTextLength(a, b) {
13   if (a.length > b.length) {
14     return 1
15   } else if (a.length < b.length) {
16     return -1
17   }
18
19   // The case where a.length = b.length
20   return 0
21 }
22
23 // Display in the console : ['Chat', 'Lion', 'Ours', 'Chien', 'Tigre', 'Cheval']
24 // 'Chat' has 4 characters, 'Cheval' has 6
25 console.log(animals.sort(compareTextLength))
26
27 // Function for comparing numbers in order to sort them in ascending order
28 function compareNumbers(a, b) {
29   return a - b
30 }
31
32 // Display in the console : [0, 2, 6, 8, 11, 40, 61, 900, 1000]
33 console.log(numbers.sort(compareNumbers))

```

### La méthode `filter()`

Cette méthode va vous retourner un nouveau tableau créé avec les valeurs du tableau d'origine, qui répondent à une condition passée en paramètre sous la forme d'une fonction.

```

1 function isTextLengthGreaterThan4(element) {
2   return element.length > 4
3 }
4
5 let animals = ['Lion', 'Chat', 'Chien', 'Cheval', 'Tigre', 'Ours']
6
7 let animalsFiltered = animals.filter(isTextLengthGreaterThan4)
8
9 let animalsFilteredBis = animalsFiltered.filter(function(element){
10  return element.length > 5
11 })
12
13 // Display in the console : ['Lion', 'Chat', 'Chien', 'Cheval', 'Tigre', 'Ours']
14 console.log(animals)
15
16 // Display in the console : ['Chien', 'Cheval', 'Tigre']
17 // Elements from animals with more than 4 characters
18 console.log(animalsFiltered)
19
20 // Display in the console : ['Cheval']
21 // Elements from animalsFiltered with more than 5 characters
22 console.log(animalsFilteredBis)

```

### La méthode `slice()`

Renvoie un tableau qui est une partie du tableau original sans modifier celui-ci.

La fonction a 2 paramètres facultatifs :

- Le premier est l'index à partir duquel il faut commencer l'extraction dans le tableau. Il peut être négatif : dans ce cas, il indique les X derniers éléments à extraire (*par exemple, -3 va extraire les 3 derniers éléments*).
- Le deuxième est l'index auquel il faut stopper l'extraction dans le tableau. **ATTENTION** : la valeur contenue à cet index ne sera pas incluse dans le tableau renvoyé, il faut donc penser à indiquer l'index de l'élément situé après le dernier élément que l'on veut dans notre extraction.
- Si le deuxième paramètre est absent, alors l'extraction va de l'index indiqué en premier paramètre jusqu'à la fin du tableau.
- Le deuxième paramètre peut également être négatif, dans ce cas il indique de ne pas intégrer les X derniers éléments dans l'extraction (*par exemple -2 va arrêter l'extraction avant les 2 derniers éléments*).

```

1 let animals = ['Lion', 'Chat', 'Chien', 'Cheval', 'Tigre', 'Ours']
2
3 // Display in the console : ['Lion', 'Chat', 'Chien', 'Cheval', 'Tigre', 'Ours']
4 console.log(animals.slice())
5
6 // Display in the console : ['Tigre', 'Ours']
7 console.log(animals.slice(4))
8
9 // Display in the console : ['Cheval', 'Tigre', 'Ours']
10 // The first parameter is negative so we extract the last 3 elements.
11 console.log(animals.slice(-3))
12
13 // Display in the console : ['Chat', 'Chien']

```

```

14 // 'Cheval' which is at index 3 is not part of the extraction as indicated in slice()
    description.
15 console.log(animals.slice(1, 3))
16
17 // Display in the console : ['Chat', 'Chien', 'Cheval']
18 // The second parameter is negative, so the last 2 values are not part of the extraction.
19 console.log(animals.slice(1, -2))
20
21 // Display in the console : ['Chien', 'Cheval']
22 // The 2 parameters are negative, so we take the last 4 values (-4) without the last 2 (-2).
23 console.log(animals.slice(-4, -2))
24
25 // Display in the console : ['Lion', 'Chat', 'Chien', 'Cheval', 'Tigre', 'Ours']
26 // The original array has not been changed.
27 console.log(animals)

```

### La méthode `pop()`

Supprime le **dernier élément** d'un tableau et le retourne.

**ATTENTION :** la longueur du tableau est donc modifiée.

```

1 let animals = ['Lion', 'Chat', 'Chien', 'Cheval', 'Tigre', 'Ours']
2
3 // Display in the console : 6
4 console.log(animals.length)
5
6 // Display in the console : Ours
7 console.log(animals.pop())
8
9 // Display in the console : 5
10 // Because the last value has been deleted
11 console.log(animals.length)

```

### La méthode `shift()`

Supprime le **premier élément** d'un tableau et le retourne.

**ATTENTION :** la longueur du tableau est donc modifiée.

```

1 let animals = ['Lion', 'Chat', 'Chien', 'Cheval', 'Tigre', 'Ours']
2
3 // Display in the console : 6
4 console.log(animals.length)
5
6 // Display in the console : Lion
7 console.log(animals.shift())
8
9 // Display in the console : 5
10 // Because the first value has been deleted
11 console.log(animals.length)

```

### La méthode `includes()`

Renvoie "true" si la valeur recherchée est présente dans le tableau, sinon "false".

Le premier paramètre est la valeur que l'on souhaite trouver.

Le deuxième, facultatif, est l'index à partir duquel il faut commencer à chercher.

```

1 let animals = ['Lion', 'Chat', 'Chien', 'Cheval', 'Tigre', 'Ours']
2
3 // Display true
4 console.log(animals.includes('Ours'))

```

```
5
6 // Display false because 'Lion' is at index 0 while we are searching from index 1
7 console.log(animals.includes('Lion', 1))
```

### Syntaxe À retenir

- `find()` : retourne la **première valeur trouvée** répondant à une condition donnée.
- `findIndex()` : retourne **l'indice (index)** de la première valeur trouvée répondant à une condition donnée.
- `indexOf()` : renvoie **l'index du premier élément trouvé** correspondant à l'élément recherché.
- `lastIndexOf()` : renvoie **l'index du dernier élément trouvé** correspondant à l'élément recherché.
- `sort()` : permet de trier un tableau. Elle renvoie ce même tableau trié.
- `filter()` : retourne un nouveau tableau créé avec les valeurs du tableau d'origine qui répondent à une condition.
- `slice()` : renvoie un tableau qui est une partie du tableau original sans modifier celui-ci.
- `pop()` : supprime le **dernier élément** d'un tableau et le retourne. **Modifie la longueur du tableau.**
- `shift()` : supprime le **premier élément** d'un tableau et le retourne. **Modifie la longueur du tableau.**
- `includes()` : renvoie "true" si la valeur recherchée est présente dans le tableau, sinon "false".

### Complément

```
find()1
findIndex()2
indexOf()3
lastIndexOf()4
sort()5
filter()6
slice()7
pop()8
shift()9
includes()10
```

<sup>1</sup> [https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets\\_globaux/Array/find](https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/find)

<sup>2</sup> [https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets\\_globaux/Array/findIndex](https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/findIndex)

<sup>3</sup> [https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets\\_globaux/Array/indexOf](https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/indexOf)

<sup>4</sup> [https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets\\_globaux/Array/lastIndexOf](https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/lastIndexOf)

<sup>5</sup> [https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets\\_globaux/Array/sort](https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/sort)

<sup>6</sup> [https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets\\_globaux/Array/filter](https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/filter)

<sup>7</sup> [https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets\\_globaux/Array/slice](https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/slice)

<sup>8</sup> [https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets\\_globaux/Array/pop](https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/pop)

<sup>9</sup> [https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets\\_globaux/Array/shift](https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/shift)

<sup>10</sup> [https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets\\_globaux/Array/includes](https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/includes)

## V. Exercice : Appliquez la notion

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail :



### Question

À partir du tableau des navigateurs web ci-dessous, vous allez devoir utiliser les méthodes pour :

1. Supprimer définitivement "Internet Explorer" et l'afficher dans la console.
2. Supprimer définitivement "Edge" et l'afficher dans la console.
3. À partir du tableau restant, ne conserver que "Firefox" et "Chrome" sans modifier le tableau obtenu après les précédentes suppressions et afficher ces deux tableaux dans la console.

```
1 let webBrowsers = ['Internet Explorer', 'Vivaldi', 'Firefox', 'Chrome', 'Opera', 'Edge']
```

### Indice :

Affichez le résultat dans la console avec `console.log()`.

## VI. Méthodes pour manipuler (copie, fusion...)

### Objectifs

- Copier des éléments de tableaux
- Fusionner des tableaux
- Manipuler des tableaux

### Mise en situation

Dans la précédente partie, vous avez appris à filtrer, trier et trouver une valeur dans un tableau JavaScript. Dans cette dernière partie, consacrée à l'objet `Array()`, nous allons voir ensemble les méthodes dédiées à la copie, fusion et manipulation des tableaux.

#### Méthode

##### La méthode `reverse()`

Permet d'inverser l'ordre des éléments en plaçant le dernier en premier, l'avant-dernier en deuxième, etc.

```
1 let numbers = [101, 0, 25, 9, 51]
2
3 numbers.reverse()
4
5 // Display [51, 9, 25, 0, 101]
6 console.log(numbers)
```

##### La méthode `push()`

Va vous permettre d'insérer un ou plusieurs élément(s) **à la fin d'un tableau**. Elle renvoie la nouvelle taille de celui-ci.

```
1 let colors = ['Bleu']
2
```

<sup>1</sup> <https://repl.it/>

```

3 // Display 3 in the console : the new array length (2 added values)
4 console.log(colors.push('Blanc', 'Rouge'))
5
6 // Display colors array with the 2 added values : ['Bleu', 'Blanc', 'Rouge']
7 console.log(colors)

```

### La méthode `unshift()`

Est semblable à `push()` sauf que les éléments sont ajoutés **au début du tableau**. Elle renvoie la nouvelle taille de celui-ci.

```

1 let colors = ['Bleu']
2
3 // Display 3 in the console : the new array length (2 added values)
4 console.log(colors.unshift('Blanc', 'Rouge'))
5
6 // Display colors array with the 2 added values : ['Blanc', 'Rouge', 'Bleu']
7 console.log(colors)

```

### La méthode `splice()`

Permet d'ajouter, modifier ou supprimer des éléments du tableau.

Elle peut prendre jusqu'à 3 paramètres :

- Le premier est l'index à partir duquel il faut commencer à modifier le tableau. Il peut être négatif : dans ce cas, il indique que la modification débutera à partir du X dernier élément (*par exemple, -3 va indiquer de commencer à partir du troisième élément en partant de la fin*).
- Le deuxième indique le nombre d'éléments à remplacer.
- Si le deuxième paramètre est absent, alors la suppression ira de l'index indiqué en premier paramètre jusqu'à la fin du tableau. S'il vaut 0, aucun élément ne sera supprimé (pour ajouter un élément sans en supprimer).
- Le troisième paramètre indique les éléments à insérer dans le tableau à partir de l'index indiqué en premier paramètre.
- Si ce paramètre est absent, alors `splice()` supprimera les éléments ciblés par les deux premiers paramètres.

```

1 let animals = ['Lion', 'Chat', 'Chien', 'Cheval', 'Tigre', 'Ours']
2
3 // Display ['Chien']
4 // Because the deletion starts at index 2 and concerns only 1 item.
5 console.log(animals.splice(2, 1))
6
7 // Display ['Lion', 'Chat', 'Cheval', 'Tigre', 'Ours']
8 // Without 'Chien' which has been deleted
9 console.log(animals)
10
11 // Display ['Tigre', 'Ours']
12 // Because the deletion starts at index 3 of the modified array and concerns 2 items.
13 console.log(animals.splice(3, 2, 'Abeille', 'Aigle', 'Lynx'))
14
15 // Display ['Lion', 'Chat', 'Cheval', 'Abeille', 'Aigle', 'Lynx']
16 // Without 'Tigre' and 'Ours' which has been deleted and replaced by 'Abeille', 'Aigle' and 'Lynx'.
17 console.log(animals)
18
19 // Display ['Abeille', 'Aigle']
20 // Because the deletion starts at the third element from the end (-3) and concerns 2 elements.
21 console.log(animals.splice(-3, 2, 'Faucon'))
22

```

```

23 // Display ['Lion', 'Chat', 'Cheval', 'Faucon', 'Lynx']
24 console.log(animals)
25
26 // Display ['Chat', 'Cheval', 'Faucon', 'Lynx']
27 // Because the deletion goes from index 1 to the end because no end value has been passed as a
    parameter.
28 console.log(animals.splice(1))
29
30 // Display ['Lion']
31 console.log(animals)
32
33 // Display []
34 // Empty array because there is no deletion (the second parameter is 0)
35 console.log(animals.splice(0, 0, 'Requin'))
36
37 // Display ['Requin', 'Lion']
38 console.log(animals)

```

### La méthode concat ()

Va concaténer les tableaux : vous l'utiliserez donc si vous avez besoin de fusionner un ou plusieurs tableau(x) en un nouveau tableau.

En effet, cette méthode n'apporte pas de modification aux tableaux existants, mais en crée un nouveau.

Elle prend en paramètres le ou les tableau(x) à concaténer au tableau.

Vous pouvez aussi concaténer directement des valeurs, comme vous pouvez le voir dans les exemples ci-dessous.

```

1 let animals = ['Chat', 'Lynx']
2 let otherAnimals = ['Lion', 'Tigre']
3 let otherAnimals2 = ['Requin', 'Aigle']
4
5 let allAnimals = animals.concat(otherAnimals, otherAnimals2)
6
7 // Display : ['Chat', 'Lynx', 'Lion', 'Tigre', 'Requin', 'Aigle']
8 // Concatenation of animals, otherAnimals and otherAnimals2
9 console.log(allAnimals)
10
11 // Concat values
12 let otherAnimals3 = otherAnimals.concat('Chien', 'Ours')
13
14 // Display : ['Lion', 'Tigre', 'Chien', 'Ours']
15 // Concatenation of otherAnimals and 'Chien', 'Ours'
16 console.log(otherAnimals3)
17
18 // Creation of a multi-dimensional array by concatenation
19 let allAnimals2 = [].concat(
20   [animals],
21   [otherAnimals],
22   [otherAnimals2]
23 )
24
25 // Display : [['Chat', 'Lynx'], ['Lion', 'Tigre'], ['Requin', 'Aigle']]
26 console.log(allAnimals2)

```

### La méthode join ()

Cette méthode donne la possibilité de créer une chaîne de caractères en concaténant tous les éléments d'un tableau.

Par défaut, le séparateur est la virgule, mais vous pouvez en préciser un autre en le passant en paramètre.

```

1 let animals = ['Lion', 'Chat', 'Chien', 'Cheval', 'Tigre', 'Ours']
2
3 // Display : Lion,Chat,Chien,Cheval,Tigre,Ours
4 console.log(animals.join())
5
6 // Display : Lion / Chat / Chien / Cheval / Tigre / Ours
7 console.log(animals.join(' / '))

```

### La méthode flat()

Va créer un nouveau tableau contenant tous les éléments du tableau et des tableaux situés à l'intérieur de celui-ci.

Par exemple, vous pouvez utiliser cette méthode si vous voulez créer un tableau avec toutes les valeurs d'un tableau à plusieurs dimensions.

Cette fonction accepte un paramètre facultatif qui permet de spécifier la profondeur de tableaux imbriqués à "aplatir" (1, par défaut).

```

1 let animals = [
2   'Zèbre',
3   ['Lion', 'Tigre'],
4   'Vache',
5   ['Chat', 'Chien'],
6   'Aigle'
7 ]
8
9 let animalsFlat = animals.flat()
10
11 // Display : ['Zèbre', 'Lion', 'Tigre', 'Vache', 'Chat', 'Chien', 'Aigle']
12 console.log(animalsFlat)
13
14 let animals2 = [
15   'Chat',
16   [
17     'Lynx',
18     ['Aigle', 'Faucon'],
19     ['Ours', 'Zèbre']
20   ],
21   'Chien',
22   [
23     ['Lion', 'Tigre']
24   ],
25   'Cheval'
26 ]
27
28 let animals2Flat1 = animals2.flat()
29
30 // Display :
31 // As the default depth is 1, only the arrays at the first depth level have been flattened.
32 /*
33 [
34   'Chat',
35   'Lynx',
36   ['Aigle', 'Faucon'],
37   ['Ours', 'Zèbre'],
38   'Chien',
39   ['Lion','Tigre'],
40   'Cheval'
41 ]

```



```

42 */
43 console.log(animals2Flat1)
44
45 let animals2Flat2 = animals2.flat(2)
46
47 // Display : ['Chat', 'Lynx', 'Aigle', 'Faucon', 'Ours', 'Zèbre', 'Chien', 'Lion', 'Tigre',
48 // As the defined depth is 2, all arrays have been flattened.
49 console.log(animals2Flat2)

```

### Syntaxe À retenir

- `reverse()` : inverse l'ordre des éléments en plaçant le dernier en premier et ainsi de suite.
- `push()` : insère un ou plusieurs élément(s) **à la fin d'un tableau**.
- `unshift()` : insère un ou plusieurs élément(s) **au début d'un tableau**.
- `splice()` : ajoute, modifie ou supprime des éléments du tableau.
- `concat()` : concatène les tableaux.
- `join()` : crée une chaîne de caractères en concaténant tous les éléments d'un tableau.
- `flat()` : crée un nouveau tableau contenant tous les éléments du tableau et des tableaux situés à l'intérieur de celui-ci (pour un tableau à plusieurs dimensions).

### Complément

`reverse()`<sup>1</sup>  
`push()`<sup>2</sup>  
`unshift()`<sup>3</sup>  
`splice()`<sup>4</sup>  
`concat()`<sup>5</sup>  
`join()`<sup>6</sup>  
`flat()`<sup>7</sup>

## VII. Exercice : Appliquez la notion

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail :



<sup>1</sup> [https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets\\_globaux/Array/reverse](https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/reverse)

<sup>2</sup> [https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets\\_globaux/Array/push](https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/push)

<sup>3</sup> [https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets\\_globaux/Array/unshift](https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/unshift)

<sup>4</sup> [https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets\\_globaux/Array/splice](https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/splice)

<sup>5</sup> [https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets\\_globaux/Array/concat](https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/concat)

<sup>6</sup> [https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets\\_globaux/Array/join](https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/join)

<sup>7</sup> [https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets\\_globaux/Array/flat](https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/flat)

<sup>8</sup> <https://repl.it/>

## Question

À partir du tableau des navigateurs web ci-dessous, vous allez devoir utiliser les méthodes pour :

1. Ajouter les éléments "Opera" et "Safari" au début du tableau en affichant la nouvelle taille du tableau dans la console.
2. Ajouter l'élément "Vivaldi" à la fin du tableau en affichant la nouvelle taille du tableau dans la console.
3. Remplacer Safari, Internet Explorer, et Edge par Firefox et Chrome en affichant les éléments supprimés et le tableau modifié.

```
1 let webBrowsers = ['Internet Explorer', 'Edge']
```

### Indice :

Pour vous aider, vous pouvez afficher un `console.log(webBrowsers)` aux différentes étapes afin de voir vos modifications.

## VIII. Auto-évaluation

### A. Exercice final

#### Exercice

Exercice

Quelle est la méthode de l'objet `Array()` renvoyant "true", si tous les éléments répondent à une condition donnée ?

- ☐ `all()`
- ☐ `every()`

Exercice

Quelle est la méthode de l'objet `Array()` renvoyant "true", si au moins un élément répond à une condition donnée ?

- ☐ `some()`
- ☐ `atLeastOne()`

Exercice

Quelle est la méthode de l'objet `Array()` qui va réduire la liste de valeurs à une valeur unique ?

- ☐ `reduce()`
- ☐ `theOne()`

Exercice

`find()` : retourne l'indice (index) de la première valeur trouvée répondant à une condition donnée.

- ☐ Vrai
- ☐ Faux

Exercice

Quel est le résultat affiché dans la console ?

```
1 let animals = ['Lion', 'Chat', 'Chien', 'Chat', 'Ours']
2
3 console.log(animals.lastIndexOf('Chat'))
```

- ☐ 0
- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4

#### Exercice

Si on utilise la méthode `sort()` sans paramètre sur un tableau de nombres, ceux-ci vont être triés par ordre croissant.

- ☐ Vrai
- ☐ Faux

#### Exercice

Que fait la méthode `pop()` ?

- ☐ Elle supprime le dernier élément d'un tableau et le retourne. La longueur du tableau est modifiée.
- ☐ Elle crée un nouveau tableau en supprimant la première valeur. Le tableau original n'est pas modifié.

#### Exercice

`push()` va vous permettre d'insérer un ou plusieurs élément(s) à la fin d'un tableau.

- ☐ Vrai
- ☐ Faux

#### Exercice

Avec la méthode `splice()`, vous pouvez... (cochez les bonnes réponses)

- ☐ Remplacer des éléments du tableau par de nouveaux éléments.
- ☐ Ajouter des éléments sans en supprimer ou sans en remplacer d'autres.
- ☐ Ajouter un élément n'importe où dans le tableau

#### Exercice

`flat()` crée une chaîne de caractères en concaténant tous les éléments d'un tableau.

- ☐ Vrai
- ☐ Faux

### B. Exercice : Défi

L'exercice final de ce cours porte sur les méthodes de l'objet JavaScript `Array()`.

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail :



<sup>1</sup> <https://repl.it/>

### Question

À partir du tableau des navigateurs web ci-dessous, vous allez devoir :

1. Utiliser la méthode `isTextLengthGreaterThan5` ci-dessous afin de savoir, **en affichant `true` ou `false` dans la console**, si tous les éléments remplissent la condition de cette fonction.
2. Trier le tableau puis afficher le résultat dans la console.
3. Utiliser la méthode `isTextLengthGreaterThan5` ci-dessous afin de **créer un nouveau tableau filtré** par cette fonction et afficher ce nouveau tableau dans la console.
4. Renvoyer la **liste filtrée** des navigateurs sous la forme d'une chaîne de caractères dont tous les éléments sont séparés par ' - '.

```
1 function isTextLengthGreaterThan5(element) {
2   return element.length > 5
3 }
4
5 let webBrowsers = ['Opera', 'Safari', 'Internet Explorer', 'Vivaldi', 'Firefox', 'Chrome',
  'Edge']
```

### Indice :

Les méthodes à utiliser sont: `every`, `sort`, `filter`, `join`.

Pour vous aider, vous pouvez afficher un `console.log(webBrowsers)` aux différentes étapes afin de voir vos modifications.

## Solutions des exercices

**Exercice p. Solution n°1**

```

1 let productStocks = [15, 184, 29, 0, 35]
2
3 console.log(productStocks.every(function(element) {
4   return element > 0
5 })))

```

`every()` : parcourt le tableau et renvoie `true` si **TOUS** les éléments remplissent une condition donnée, dans une fonction passée en paramètre, sinon `false`.

Il est également possible de créer une fonction à part pour le test :

```

1 function isGreaterThanZero(element) {
2   return element > 0
3 }
4
5 let productStocks = [15, 184, 29, 0, 35]
6
7 console.log(productStocks.every(isGreaterThanZero))

```

**Exercice p. Solution n°2**

```

1 let webBrowsers = ['Internet Explorer', 'Vivaldi', 'Firefox', 'Chrome', 'Opera', 'Edge']
2
3 // Delete 'Internet Explorer' and display it in the console.
4 console.log(webBrowsers.shift())
5
6 // Delete 'Edge' and display it in the console.
7 console.log(webBrowsers.pop())
8
9 // Display ['Firefox', 'Chrome'] in the console without modifying the array containing the
  remaining values.
10 console.log(webBrowsers.slice(1, 3))
11
12 // Display the array containing the remaining values.
13 // ['Vivaldi', 'Firefox', 'Chrome', 'Opera']
14 console.log(webBrowsers)

```

**Exercice p. Solution n°3**

```

1 let webBrowsers = ['Internet Explorer', 'Edge']
2
3 // 1) - Adds the elements at the beginning and display the new length : 4
4 console.log(webBrowsers.unshift('Opera', 'Safari'))
5
6 // 2) - Adds the elements at the end and display the new length : 5
7 console.log(webBrowsers.push('Vivaldi'))
8
9 // 3)
10 // a) Display : ['Safari', 'Internet Explorer', 'Edge']
11 // Because the selection starts at index 1 and the deletion is for 3 items
12 console.log(webBrowsers.splice(1, 3, 'Firefox', 'Chrome'))
13
14 // b) Display : ['Opera', 'Firefox', 'Chrome', 'Vivaldi']
15 // Safari, 'Internet Explorer', 'Edge' have been replaced by 'Firefox', 'Chrome'.

```

```
16 console.log(webBrowsers)
```

### Exercice p. 18 Solution n°4

Exercice

Quelle est la méthode de l'objet `Array()` renvoyant "true", si tous les éléments répondent à une condition donnée ?

☐ `all()`

☒ `every()`



C'est la méthode `every()` qui renvoie "true" si tous les éléments correspondent à une condition donnée.

Exercice

Quelle est la méthode de l'objet `Array()` renvoyant "true", si au moins un élément répond à une condition donnée ?

☒ `some()`

☐ `atLeastOne()`



C'est la méthode `some()` qui renvoie "true" si au moins un élément correspond à une condition donnée.

Exercice

Quelle est la méthode de l'objet `Array()` qui va réduire la liste de valeurs à une valeur unique ?

☒ `reduce()`

☐ `theOne()`



C'est la méthode `reduce()` qui réduit la liste de valeurs à une valeur unique.

Exercice

`find()` : retourne l'indice (index) de la première valeur trouvée répondant à une condition donnée.

☐ Vrai

☒ Faux



C'est faux, `find()` retourne la première valeur trouvée répondant à une condition donnée.

C'est `findIndex()` qui retourne l'indice (index) de la première valeur trouvée répondant à une condition donnée.

Exercice

Quel est le résultat affiché dans la console ?

```
1 let animals = ['Lion', 'Chat', 'Chien', 'Chat', 'Ours']
2
3 console.log(animals.lastIndexOf('Chat'))
```


☐ 0

☐ 1

☐ 2

☒ 3

☐ 4


 `lastIndexOf()` renvoie l'index du dernier élément trouvé correspondant à l'élément recherché, passé en paramètre. Le dernier élément 'Chat' est à l'index 4.

#### Exercice

Si on utilise la méthode `sort()` sans paramètre sur un tableau de nombres, ceux-ci vont être triés par ordre croissant.

☐ Vrai

☒ Faux


 Faux, cette méthode trie des chaînes de caractères : par conséquent, si vous lui passez des nombres, ceux-ci seront convertis en chaînes de caractères et triés en tant que telles et non en tant que nombres. Vous n'aurez pas un tableau de nombres triés par ordre croissant ou décroissant.

#### Exercice

Que fait la méthode `pop()` ?

☒ Elle supprime le dernier élément d'un tableau et le retourne. La longueur du tableau est modifiée.

☐ Elle crée un nouveau tableau en supprimant la première valeur. Le tableau original n'est pas modifié.

 La méthode `pop()` supprime le dernier élément d'un tableau et le retourne. La longueur du tableau est modifiée.

#### Exercice

`push()` va vous permettre d'insérer un ou plusieurs élément(s) à la fin d'un tableau.

☒ Vrai

☐ Faux

 C'est vrai.

#### Exercice

Avec la méthode `splice()`, vous pouvez... (cochez les bonnes réponses)

☒ Remplacer des éléments du tableau par de nouveaux éléments.

☒ Ajouter des éléments sans en supprimer ou sans en remplacer d'autres.

☒ Ajouter un élément n'importe où dans le tableau

 Toutes ces affirmations sont justes.

#### Exercice

`flat()` crée une chaîne de caractères en concaténant tous les éléments d'un tableau.

☐ Vrai

☒ Faux

 Faux, c'est `join()` crée une chaîne de caractères en concaténant tous les éléments d'un tableau.

`flat()` crée un nouveau tableau contenant tous les éléments du tableau et des tableaux situés à l'intérieur de celui-ci (pour un tableau à plusieurs dimensions).

**Exercice p. Solution n°5**

```

1 function isTextLengthGreaterThan5(element) {
2   return element.length > 5
3 }
4
5 let webBrowsers = ['Opera', 'Safari', 'Internet Explorer', 'Vivaldi', 'Firefox', 'Chrome',
6   'Edge']
7 // 1)
8 // Display : false
9 // Not all elements have at least 5 characters (Edge has 4 characters)
10 console.log(webBrowsers.every(isTextLengthGreaterThan5))
11
12 // 2)
13 webBrowsers.sort()
14
15 // Display : ['Chrome', 'Edge', 'Firefox', 'Internet Explorer', 'Opera', 'Safari', 'Vivaldi']
16 console.log(webBrowsers)
17
18 // 3)
19 let webBrowsersFiltered = webBrowsers.filter(isTextLengthGreaterThan5)
20
21 // Display : ['Chrome', 'Firefox', 'Internet Explorer', 'Safari', 'Vivaldi']
22 console.log(webBrowsersFiltered)
23
24 // 4)
25 // Display : Chrome - Firefox - Internet Explorer - Safari - Vivaldi
26 console.log(webBrowsersFiltered.join(' - '))

```