

Les tableaux

Table des matières

I. Contexte	3
II. Qu'est-ce qu'un tableau JavaScript ?	3
III. Exercice : Appliquez la notion	4
IV. Comment déclarer un tableau JavaScript ?	4
V. Exercice : Appliquez la notion	6
VI. Accès aux valeurs d'un tableau JavaScript	7
VII. Exercice : Appliquez la notion	12
VIII. Tri à bulles	12
IX. Exercice : Appliquez la notion	14
X. Autoévaluation	14
A. Exercice final	14
B. Exercice : Défi	16
Solutions des exercices	16

I. Contexte

Durée : 1 h

Environnement de travail : Repl.it

Pré-requis : Connaître les bases de JavaScript

Contexte

Maintenant que vous connaissez certaines bases de JavaScript, vous allez rencontrer des problématiques liées à l'utilisation de listes et de collections de valeurs (ex : *gérer une liste de produits, personnes...*). L'utilisation de tableaux va donc se révéler indispensable pour répondre à ces besoins de la manière la plus efficace.

II. Qu'est-ce qu'un tableau JavaScript ?

Objectifs

- Pourquoi utiliser un tableau ?
- Qu'est-ce qu'un tableau JavaScript ?
- Présentation rapide de l'objet Array.

Mise en situation

Lors de vos développements JavaScript, vous allez avoir besoin de gérer des listes de valeurs qu'il vous faudra : stocker, parcourir, ajouter, modifier ou supprimer. Ce cours va donc présenter les bases de l'utilisation des tableaux JavaScript, leur création ainsi que le parcours et l'accès aux valeurs.

Un tableau contient une liste de valeurs qui peuvent être des nombres, des textes, des objets ou encore d'autres tableaux. Dans ce dernier cas, on parle alors de tableau à plusieurs dimensions – mais nous y reviendrons.

Pourquoi utiliser un tableau ?

Vous devez gérer une liste de produits, catégories, personnes, etc. ? **Un tableau va vous permettre de gérer plusieurs éléments dans une seule variable.**

Définition

Qu'est-ce qu'un tableau en JavaScript ?

En JavaScript, il n'y a pas de type primitif pour les tableaux. C'est donc l'objet `Array`, présent nativement, qui est utilisé pour la manipulation de tableaux.

Il possède différentes propriétés et méthodes que nous présenterons plus en détails dans un prochain cours. Grâce à celui-ci, il est notamment possible :

- De connaître le nombre d'éléments d'un tableau (sa longueur),
- D'ajouter des éléments au début ou à la fin,
- De supprimer le premier/dernier élément,
- De trouver la position d'un élément (son index au sein du tableau), etc.

Exemple Tableaux JavaScript

```
1 // Number Array
2 let prices = [5, 10, 2, 994]
3
4 // Text Array
5 let animals = ['Lion', 'Chat', 'Chien', 'Cheval', 'Tigre', 'Ours']
```

Syntaxe À retenir

Un tableau permet de gérer une liste dans une seule variable.

En JavaScript, il n'y a pas de type primitif pour les tableaux, c'est l'**objet "Array"** qui est utilisé.

III. Exercice : Appliquez la notion

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail Repl.it¹.



Question

[solution n°1 p.17]

Dans cet exercice, nous allons copier/coller un exemple de tableau JavaScript et observer le résultat.

Avec Repl.it :

1. Accédez au site <https://repl.it/>.
2. Cliquez sur "+ new repl" (en haut à droite).
3. Sélectionnez "HTML, CSS, JS".
4. Cliquez sur "Create Repl".
5. Trois fichiers fictifs "index.html, script.js et style.css" ont été créés.
6. Cliquez sur "script.js" dans la colonne de gauche.
7. En dessous de l'onglet "script.js" de la fenêtre centrale, copiez/collez le code ci-dessous.
8. Cliquez sur le bouton "run", au-dessus, afin d'exécuter ce code.

Quel animal est affiché dans la console ?

```
1 let animals = ['Lion', 'Chat', 'Chien', 'Cheval', 'Tigre', 'Ours']
2
3 console.log(animals[0])
```

IV. Comment déclarer un tableau JavaScript ?

Objectifs

- Apprendre à créer un tableau JavaScript
- Apprendre à créer un tableau JavaScript à plusieurs dimensions

¹ <https://repl.it/>

² <https://repl.it/>

Mise en situation

Vous commencerez par un cas relativement simple qui consiste à créer un tableau à partir d'une liste d'éléments. Par la suite, vous enchaînez avec un cas plus complexe: créer une liste possédant des catégories, et chacune d'entre elles contenant des éléments (tableau à plusieurs dimensions).

Méthode Créer un tableau JavaScript

Nous allons voir qu'il existe plusieurs façons de déclarer un tableau JavaScript, mais une seule sera à utiliser, car elle est définie dans les normes de bonnes pratiques de codage.

La méthode littérale, définie dans les bonnes pratiques (entre crochets [. . .]) :

```
1 let animals = ['Lion', 'Chat', 'Chien', 'Cheval', 'Tigre', 'Ours']
```

Vous remarquerez que les différentes valeurs sont **séparées par des virgules suivies d'espaces. Ces espaces ne sont pas obligatoires mais ils sont recommandés afin d'améliorer la lisibilité.**

Ci-dessus, nous avons donc créé un tableau dans une variable "animals" qui contient 6 valeurs (6 noms d'animaux).

Afin de pouvoir compter, ajouter, supprimer et modifier, nous manipulerons directement la variable "animals".

Il est également possible d'initialiser un tableau vide comme ci-dessous (les valeurs sont alors ajoutées en JavaScript) :

```
1 let animals = []
```

Attention La « mauvaise » méthode pour déclarer un tableau, en utilisant le constructeur new Array()

Cette méthode ne doit pas être utilisée, car elle ne constitue pas une bonne pratique. Toutefois, il est possible de la rencontrer dans des exemples de code sur Internet ou dans du code écrit par d'autres personnes. C'est pourquoi nous vous la présentons ci-dessous :

```
1 let animals = new Array('Lion', 'Chat', 'Chien', 'Cheval', 'Tigre', 'Ours')
```

Méthode Créer un tableau JavaScript à plusieurs dimensions

Dans certains cas, vous aurez besoin d'un tableau à plusieurs dimensions, c'est-à-dire, un tableau ayant pour valeurs d'autres tableaux, et chacun d'eux possédant eux-mêmes leurs propres valeurs.

Reprenons notre exemple des animaux : imaginons que vous souhaitiez les catégoriser en animaux « sauvages » et « domestiques ».

On aura donc un tableau "animals" contenant deux valeurs, qui seront :

- Un tableau contenant les animaux sauvages (avec trois valeurs).
- Un tableau contenant les animaux domestiques (avec trois valeurs).

```
1 let animals = [  
2   ['Lion', 'Tigre', 'Ours'],  
3   ['Chat', 'Chien', 'Cheval']  
4 ]
```

Explications :

- Le crochet de la première ligne ouvre le tableau "animals".
- Sur la deuxième ligne, la première valeur du tableau "animals", est le tableau concernant les animaux sauvages (contenant lui-même 3 valeurs).
- À la suite, il y a une **virgule** qui permet de séparer cette première valeur de la seconde.
- Sur la troisième ligne, on a donc la deuxième valeur du tableau "animals", qui est le tableau des animaux domestiques (contenant lui-même 3 valeurs).

- Enfin, sur la dernière ligne se trouve le crochet fermant qui permet de clôturer le tableau "animals".

Syntaxe À retenir

La bonne pratique pour déclarer un tableau est d'utiliser les crochets :

- `let animals = ['Lion', 'Chat']`

Les valeurs d'un tableau sont séparées par des virgules suivies d'un espace.

Un tableau à plusieurs dimensions est un tableau ayant pour valeurs d'autres tableaux, comportant eux-mêmes leurs propres valeurs.

Complément

Il existe des subtilités lors de la création d'un tableau JavaScript, que nous allons voir ci-dessous :

```
1 let array = new Array(5)
```

- **L'exemple ci-dessus**, qui utilise la mauvaise syntaxe et qu'il ne faut donc **pas** reproduire, ne contiendra pas la valeur 5, mais créera un **tableau de longueur 5 avec 5 valeurs vides** : `[, , , ,]`.

```
1 let array = new Array('5')
```

- **L'exemple ci-dessus** va bien créer un tableau contenant uniquement la valeur '5', en effet, celle-ci étant entre apostrophes, elle constitue donc une chaîne de caractères.

```
1 let array = [5]
```

- **L'exemple ci-dessus** va bien créer un tableau comprenant uniquement la valeur 5.

V. Exercice : Appliquez la notion

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail Repl.it.



Question 1

[solution n°2 p.17]

Dans cet exercice, vous allez devoir créer un tableau JavaScript contenant cette liste de fruits : Fraise, Orange, Framboise et Citron.

Question 2

[solution n°3 p.17]

À présent, avec la liste de fruits "Fraise, Orange, Framboise, Citron", vous allez devoir créer un tableau à plusieurs dimensions afin d'avoir, dans une seule variable, une liste contenant les fruits rouges et une autre contenant les agrumes.

Indice :

Votre tableau de fruits aura deux valeurs : un tableau contenant les fruits rouges (Fraise, Framboise) et un autre contenant les agrumes (Orange, Citron).

1 <https://repl.it/>

VI. Accès aux valeurs d'un tableau JavaScript

Objectifs

- Apprendre à accéder à une valeur dans un tableau
- Savoir parcourir un tableau

Mise en situation

Maintenant que vous savez créer des tableaux JavaScript, nous allons voir comment parcourir ces tableaux afin de récupérer les valeurs qui s'y trouvent. Mais vous allez également voir qu'il est possible d'y accéder directement.

Méthode Accéder directement à une valeur d'un tableau JavaScript

Tout d'abord, il est important que vous sachiez comment sont stockées les valeurs dans un tableau JavaScript.

```
1 let animals = ['Lion', 'Chat', 'Chien', 'Cheval', 'Tigre', 'Ours']
```

Le tableau ci-dessus contient 6 valeurs : on dit que sa longueur (propriété `length`) est de 6.

Pour accéder à une valeur, on peut préciser son numéro d'indice (index) au sein du tableau.

IMPORTANT : il est indispensable que vous reteniez que la première valeur d'un tableau n'est pas à l'index 1, mais à l'index 0.

Ainsi, dans le tableau ci-dessus, la première valeur "Lion" est à l'index 0 du tableau, et la dernière, "Ours", se trouve à l'index 5.

Pour obtenir une valeur, on va utiliser la variable du tableau et préciser l'index de la valeur voulue entre crochets `[n]` :

```
1 // Lion
2 animals[0]
3
4 // Chat
5 animals[1]
6
7 // Chien
8 animals[2]
9
10 // Cheval
11 animals[3]
12
13 // Tigre
14 animals[4]
15
16 // Ours
17 animals[5]
18
19 // Display Lion in the console
20 console.log(animals[0])
```

Méthode Accéder directement à une valeur d'un tableau JavaScript à plusieurs dimensions

Reprenons notre tableau "animals" contenant deux valeurs, qui sont :

- Un tableau contenant les animaux sauvages (avec 3 valeurs).
- Un tableau contenant les animaux domestiques (avec 3 valeurs).

```
1 let animals = [
2   ['Lion', 'Tigre', 'Ours'],
3   ['Chat', 'Chien', 'Cheval']
4 ]
```

Comme vu précédemment, on utilise l'index de la valeur : sauf que, cette fois, la première valeur de notre tableau est un autre tableau (ex : ['Lion', 'Tigre', 'Ours']).

Exemple ci-dessous :

```
1 // ['Lion', 'Tigre', 'Ours']
2 animals[0]
3
4 // ['Chat', 'Chien', 'Cheval']
5 animals[1]
```

Cependant, on peut également accéder à une des valeurs des deux tableaux présents dans "animals".

Pour cela, après le premier index (tableau ciblé), on ajoutera l'index de la valeur voulue dans ce tableau.

Exemple ci-dessous :

```
1 // First JavaScript array in the animals array (animals[0])
2 // Lion
3 animals[0][0]
4
5 // Tigre
6 animals[0][1]
7
8 // Ours
9 animals[0][2]
10
11 // Second JavaScript array in the animals array (animals[1])
12 // Chat
13 animals[1][0]
14
15 // Chien
16 animals[1][1]
17
18 // Cheval
19 animals[1][2]
```

Méthode Parcourir un tableau JavaScript

En utilisant les tableaux, il arrive souvent que l'on doive les parcourir entièrement pour effectuer des traitements, des vérifications, etc. Nous allons donc voir comment parcourir un tableau JavaScript.

Propriété `length` :

Tout d'abord, pour parcourir entièrement un tableau, il faut connaître sa longueur : la propriété `length` de l'objet `Array` permet de récupérer cette information.

Exemple :

```
1 let animals = ['Lion', 'Chat', 'Chien', 'Cheval', 'Tigre', 'Ours']
2
3 // 6
4 animals.length
5
6 // Display array length (6) in the console
7 console.log(animals.length)
```


Boucle for :

Vous allez pouvoir parcourir le tableau en utilisant sa propriété `length` dans la boucle `for` afin d'exécuter le bloc d'instructions autant de fois que d'éléments présents au sein du tableau.

Exemple :

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

Explications :

Voici les détails de l'instruction `for` :

1. Pour "`i`" = 0, (`let i = 0`)
2. Tant que "`i`" reste inférieur à la longueur de notre tableau "`animals`", (`i < animals.length` (vaut 6))
3. on incrémente "`i`". (`i++`)

Incréments "`i`" signifie que l'on augmente sa valeur : "`i`" vaudra donc "`i + 1`" (`i++`) à la fin de chaque passage dans la boucle, soit 1 (=0+1), 2 (=1+1), 3, 4, 5 et enfin 6 (=5+1).

Au premier passage dans la boucle, "`i`" vaut 0, ce qui équivaut à exécuter l'instruction présente dans la boucle :

- `console.log(animals[0])`

Puis ainsi de suite tant que "`i`" est inférieur à 6 (`animals.length`) :

- `console.log(animals[1])`
- `console.log(animals[2])`
- `console.log(animals[3])`
- `console.log(animals[4])`
- `console.log(animals[5])`

Ensuite, "`i`" vaut 6 et n'est donc plus inférieur à `animals.length` (6), ce qui induit que la boucle se termine.

Les boucles `forEach`

Pour parcourir un tableau, vous pouvez également utiliser la méthode `forEach()` de l'objet `Array`, qui permet d'exécuter une fonction sur chaque élément qui le compose.

Vous pourrez vous renseigner plus en profondeur sur ce type de boucles, dans le chapitre concernant "les méthodes de l'Array".

Exemples :

```
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 its element in the console
9 animals.forEach(function(element, index){
10  console.log(index + ' : ' + element)
11 })
```

Explications :

Dans l'exemple ci-dessus, nous passons une fonction en paramètre de la méthode `forEach()`. Celle-ci s'exécute sur chacun des éléments du tableau.

Cette fonction peut prendre plusieurs paramètres, comme la valeur en cours de traitement (ici nommée `element`). Le premier exemple ci-dessus correspond donc à :

- `console.log(Lion)`
- `console.log(Chat)`
- ...
- `console.log(Ours)`

Mais il est également possible d'y passer en paramètre l'indice (`index`) de la valeur en cours de traitement. Le second exemple ci-dessus correspond donc à :

- `console.log(0 : Lion)`
- `console.log(1 : Chat)`
- ...
- `console.log(5 : Ours)`

Parcourir un tableau à plusieurs dimensions :

Pour parcourir un tableau à plusieurs dimensions, le principe est le même, sauf que vous allez également devoir parcourir les tableaux contenus à l'intérieur.

Vous allez donc faire une **boucle dans une boucle**.

Avec une boucle `for`, l'écriture est un peu fastidieuse, car il y a deux index à gérer : un pour les valeurs du tableau `animals` (`animalsIndex`) et un pour les valeurs de chacun des tableaux contenus à l'intérieur (`elementIndex`).

Exemple :

```
1 let animals = [
2   ['Lion', 'Tigre', 'Ours'],
3   ['Chat', 'Chien', 'Cheval']
4 ]
5
6 // Browse animals array
7 for (let animalsIndex = 0; animalsIndex < animals.length; animalsIndex++)
8 {
9   // Browse arrays in animals array
10  for (let elementIndex = 0; elementIndex < animals[animalsIndex].length; elementIndex++)
11  {
12    // Displays each array element in the console
13    console.log(animals[animalsIndex][elementIndex])
14  }
15 }
```

L'exemple ci-dessus équivaut donc à :

- `console.log(animals[0][0]) // Lion`
- `console.log(animals[0][1]) // Tigre`
- `console.log(animals[0][2]) // Ours`
- `console.log(animals[1][0]) // Chat`
- `console.log(animals[1][1]) // Chien`
- `console.log(animals[1][2]) // Cheval`

Avec `forEach` :

Ici également, nous allons faire une boucle dans une boucle :

```
1 let animals = [  
2   ['Lion', 'Tigre', 'Ours'],  
3   ['Chat', 'Chien', 'Cheval']  
4 ]  
5  
6 // For each value in the animals array (2 arrays)  
7 animals.forEach(function(animalsElement){  
8   // For each value in each array (the 2 arrays in animals array)  
9   animalsElement.forEach(function(element){  
10    console.log(element)  
11  })  
12 })
```

Résultat :

- `console.log(Lion)`
- `console.log(Tigre)`
- `console.log(Ours)`
- `console.log(Chat)`
- `console.log(Chien)`
- `console.log(Cheval)`

Rappel

Il existe d'autres paramètres pour `forEach()`, ainsi que d'autres notions pour les boucles, dans les cours dédiés.

Syntaxe **À retenir**

La première valeur d'un tableau est à l'**index 0**.

Pour accéder directement à une valeur du tableau, on précise (entre crochets) son **index** au sein de la variable utilisée pour le tableau :

- `animals[0]`
- ...
- `animals[5]`

La propriété **length** permet de connaître la longueur d'un tableau JavaScript :

- `animals.length` (égal à 6 dans cet exemple)

L'instruction **for** permet d'exécuter un bloc d'instructions en boucle sur un tableau.

La méthode **forEach** permet d'exécuter une fonction sur chaque élément qui compose le tableau.

Pour parcourir un tableau à plusieurs dimensions, on peut faire une boucle dans une boucle.

Avant d'utiliser une boucle comme **for** ou **forEach**, vérifiez si une fonction spécifique ne peut pas faire le traitement souhaité.

VII. Exercice : Appliquez la notion

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail Repl.it.



Question 1

[solution n°4 p.17]

À partir du tableau de fruits ci-dessous, écrivez le code JavaScript permettant d'afficher sa taille (longueur) dans la console.

```
1 let fruits = ['Fraise', 'Orange', 'Framboise', 'Citron']
```

Indice :

Méthode pour l'affichage dans la console : `console.log()`.

Question 2

[solution n°5 p.17]

À partir du tableau de fruits ci-dessous, écrivez le code JavaScript permettant d'afficher directement la valeur "Framboise" dans la console.

```
1 let fruits = ['Fraise', 'Orange', 'Framboise', 'Citron']
```

Indice :

Le premier élément d'un tableau est à l'indice (index) 0. Méthode pour l'affichage dans la console : `console.log()`.

Question 3

[solution n°6 p.17]

À partir du tableau de fruits ci-dessous, écrivez le code JavaScript permettant d'afficher chaque valeur dans la console.

```
1 let fruits = ['Fraise', 'Orange', 'Framboise', 'Citron']
```

Indice :

Utilisez une boucle `for` ou `forEach`. Méthode pour l'affichage dans la console : `console.log()`.

VIII. Tri à bulles

Objectifs

Comprendre le principe du tri à bulles

Découvrir le fonctionnement des tableaux

Voir comment parcourir un tableau

Remarque

Ce chapitre présente une notion plus avancée sur les tableaux. Donc n'hésitez pas à lire ce contenu plus tard si vous sentez que c'est pour l'instant difficile d'accès pour vous.

Mise en situation

En programmation, nous sommes souvent amenés à gérer des listes de valeurs, comme des listes de noms d'utilisateurs ou de nombres. Pour pouvoir manipuler ces listes, on utilise des **tableaux**, qui sont un moyen de stocker plusieurs valeurs dans une même variable. Chaque valeur se situe dans une case du tableau, mais le tableau en lui-même est stocké dans une variable.

1 <https://repl.it/>

Pour donner un exemple de manipulation de ce type de données, nous allons voir un exemple de tri de tableau : le tri à bulles.

Méthode Tri à bulles

Imaginons une liste de nombres que nous devons trier par ordre croissant. Or, comme pour les chaînes de caractères, nous ne pouvons traiter chaque valeur qu'une par une, du début à la fin, grâce à une boucle.

Une solution simple serait de parcourir le tableau du début à la fin en recherchant le nombre le plus grand de la liste, puis de le placer à la fin de la liste. Ensuite, on parcourt à nouveau le tableau du début jusqu'à l'avant-dernière valeur en recherchant le nombre le plus grand, puis on le place en avant-dernier, et ainsi de suite. Cette méthode est le **tri par sélection**. Celui-ci fonctionne bien, mais nécessiterait l'utilisation d'une variable supplémentaire pour stocker la valeur lors de la recherche du plus grand élément, donc utiliserait plus de mémoire. Nous allons donc réaliser une variante de cette technique : **le tri à bulles**.

Le principe est globalement le même, sauf qu'au moment de parcourir le tableau, nous allons comparer chaque valeur avec sa voisine pour savoir laquelle est la plus grande. Si la plus grande est placée avant, alors on échange les valeurs des deux cases : cela aura pour effet de "pousser" la plus grande valeur vers la fin du tableau. Nous pourrions ensuite répéter cette opération pour l'avant-dernière valeur et ainsi de suite jusqu'à avoir un tableau ordonné.

Pour réaliser cela, nous allons avoir besoin d'une boucle permettant de parcourir chaque case du tableau et de permuter les valeurs si nécessaire. Mais ce traitement ne permet que de placer une seule valeur. Par conséquent, cette boucle devra donc être contenue dans une boucle principale permettant de répéter cette opération pour chaque position (dernière, puis avant-dernière, etc.).

Voici le code en JavaScript :

```
1 let numbers = [3, 8, 5, 9, 1];
2 let numbersLength = numbers.length;
3 for (let main = 0; main < numbersLength - 1; main++) {
4   for (let cell = 0; cell < numbersLength - main - 1; cell++) {
5     if (numbers[cell] > numbers[cell + 1]) { // Si la valeur de la case actuelle est
        supérieure à la valeur de la case d'après, on permute les valeurs (en utilisant l'échange de
        variables)
6       numbers[cell] = numbers[cell] + numbers[cell + 1];
7       numbers[cell + 1] = numbers[cell] - numbers[cell + 1];
8       numbers[cell] = numbers[cell] - numbers[cell + 1];
9     }
10  }
11 }
12 console.log(numbers);
```

Deux boucles imbriquées sont ici créées. La première, où on définit un itérateur `main`, va nous servir à déterminer quelle valeur on va récupérer : au premier tour de la boucle, on va placer la dernière valeur, au second tour, l'avant-dernière et ainsi de suite.

La seconde boucle, avec l'itérateur `cell`, va s'occuper de réaliser les permutations (c'est cette boucle qui suit le cheminement du schéma ci-dessus). Elle va être appelée autant de fois qu'il y a d'éléments dans le tableau.

Exemple

Voici la trace de l'algorithme ci-dessus.

Pour simplifier la trace, les permutations n'apparaissent pas : chaque ligne de la trace (à partir des boucles) représente le résultat d'un tour de boucle (donc, si on a échangé des valeurs ou non).

Les lignes en jaune sont les lignes où une permutation a eu lieu.

Syntaxe À retenir

Pour gérer des listes de valeurs, il est possible d'utiliser des tableaux. Chaque valeur est contenue dans une case et possède un numéro de case permettant de récupérer la valeur. Les numéros de case commencent à 0.

Complément

Le tri à bulles (Wikipédia)¹

IX. Exercice : Appliquez la notion

Mise en contexte

Voyons un algorithme utilisant le parcours des tableaux : nous avons un tableau de prénoms d'utilisateurs et nous souhaitons trier ces prénoms dans l'ordre alphabétique en utilisant un algorithme de tri à bulles. On sait que JavaScript est capable de comparer deux chaînes de caractères avec les opérateurs > et < de la même manière qu'avec des nombres.

```
1 let persons = ['jules', 'laure', 'vincent', 'emma'];
2 let personsLength = persons.length;
3
4 for (let main = 0; main < personsLength - 1; main++) {
5   for (let cell = 0; cell < personsLength - main - 1; cell++) {
6     if (persons[cell] > persons[cell + 1]) {
7
8       let temp;
9       temp = persons[cell];
10      persons[cell] = persons[cell + 1];
11      persons[cell + 1] = temp;
12    }
13  }
14 }
15 console.log(persons);
```

Question

[solution n°7 p.18]

Écrivez la trace de l'exécution de ce programme.

X. Autoévaluation

A. Exercice final

Exercice 1

[solution n°8 p.18]

Exercice

En JavaScript, une seule variable peut permettre de gérer plusieurs valeurs.

- ☐ Vrai
- ☐ Faux

Exercice

En JavaScript, pour créer un tableau, on utilise un objet.

- ☐ Vrai
- ☐ Faux

¹ https://fr.wikipedia.org/wiki/Tri_%C3%A0_bulles

Exercice

Choisissez la bonne pratique pour créer un tableau JavaScript.

- ☐ let products = new Array('Smartphone', 'Tablette', 'PC')
- ☐ let products = ['Smartphone', 'Tablette', 'PC']
- ☐ let products = Array('Smartphone', 'Tablette', 'PC')
- ☐ let products = ('Smartphone', 'Tablette', 'PC')

Exercice

Le code ci-dessous va créer un tableau avec la valeur 3.

```
1 let array = new Array(3)
```

- ☐ Vrai
- ☐ Faux

Exercice

Les différents éléments d'un tableau sont séparés par :

- ☐ Une virgule
- ☐ Un point
- ☐ Un point-virgule

Exercice

Qu'est-ce qu'un tableau à plusieurs dimensions ?

- ☐ Un tableau ayant pour valeurs d'autres tableaux
- ☐ Un tableau ayant pour valeurs différentes mesures

Exercice

Pour connaître la taille/longueur d'un tableau, on utilise la propriété :

- ☐ size
- ☐ length

Exercice

Complétez le code suivant afin de retourner le premier élément du tableau language.

```
let language = ['JavaScript', 'HTML', 'CSS']
```

```
language[ ]
```

Exercice

Le premier élément d'un tableau se situe à l'indice (index) 1.

- ☐ Vrai
- ☐ Faux

Exercice

Quelle syntaxe est correcte pour accéder à la troisième valeur d'un tableau ?

- ☐ array(3)
- ☐ array[3]
- ☐ array[2]
- ☐ array(2)

Exercice

Pour parcourir un tableau en JavaScript, vous pouvez utiliser :

- ☐ for
- ☐ browse
- ☐ forEach
- ☐ navigate
- ☐ explore

B. Exercice : Défi

L'exercice final de ce cours porte sur la création et le parcours de tableaux JavaScript.

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail Repl.it.



Question

[solution n°9 p.20]

Dans cet exercice, vous allez devoir créer un tableau JavaScript des couleurs, avec les valeurs : Bleu, Rouge, Vert, Jaune, Blanc.

Vous devrez ensuite :

1. Faire afficher (dans la console par exemple) la longueur du tableau.
2. Faire afficher directement les valeurs : Bleu, Blanc, puis Rouge.
3. Faire afficher toutes les valeurs du tableau de deux manières différentes.
4. Faire un nouveau tableau à plusieurs dimensions pour avoir "Bleu, Blanc et Rouge" d'une part, et "Jaune et Vert" d'autre part.
5. À partir de ce nouveau tableau, afficher directement les valeurs : "Blanc" puis "Vert".
6. Utiliser la méthode la plus simple pour afficher toutes les valeurs de ce nouveau tableau.

Indice :

Méthode pour l'affichage dans la console : `console.log()`.

Solutions des exercices

1 <https://repl.it/>

p. 4 Solution n°1

L'animal affiché dans la console est : Lion.

On constate qu'avec une seule variable (`animals`), on peut gérer plusieurs valeurs et afficher celle que l'on souhaite.

p. 6 Solution n°2

```
1 let fruits = ['Fraise', 'Orange', 'Framboise', 'Citron']
```

Ici, nous avons créé une variable "`fruits`", qui est un tableau contenant les 4 valeurs demandées (séparées avec des virgules suivies d'un espace). Comme les valeurs sont des chaînes de caractères, on les met chacune entre apostrophes.

p. 6 Solution n°3

```
1 let fruits = [  
2   ['Fraise', 'Framboise'],  
3   ['Orange', 'Citron']  
4 ]
```

Ici, nous avons donc créé une variable "`fruits`", qui est un tableau contenant deux valeurs :

- Un tableau contenant les fruits rouges 'Fraise' et 'Framboise'.
- Un tableau contenant les agrumes 'Orange' et 'Citron'.

p. 12 Solution n°4

```
1 let fruits = ['Fraise', 'Orange', 'Framboise', 'Citron']  
2  
3 console.log(fruits.length)
```

p. 12 Solution n°5

```
1 console.log(fruits[2])
```

RAPPEL : le premier élément d'un tableau est à l'indice (index) 0.

p. 12 Solution n°6**Avec for**

```
1 let fruits = ['Fraise', 'Orange', 'Framboise', 'Citron']  
2  
3 for (var i = 0; i < fruits.length; i++) {  
4   console.log(fruits[i])  
5 }
```

Avec forEach

```
1 let fruits = ['Fraise', 'Orange', 'Framboise', 'Citron']
2
3 fruits.forEach(function(element){
4   console.log(element)
5 })
```

p. 14 Solution n°7

```
1 let persons = ['jules', 'laure', 'vincent', 'emma']
2
3 console.log(persons)
```


Rappel

Le premier élément d'un tableau est à l'indice (index) 0.

Exercice p. 14 Solution n°8


Exercice

En JavaScript, une seule variable peut permettre de gérer plusieurs valeurs.

- ☒ Vrai
- ☐ Faux
-  En effet, les tableaux gèrent une liste de valeurs dans une seule variable.


Exercice

En JavaScript, pour créer un tableau, on utilise un objet.

- ☒ Vrai
- ☐ Faux
-  Vrai, il s'agit de l'objet `Array`, car JavaScript n'a pas de type primitif pour les tableaux.

Exercice

Choisissez la bonne pratique pour créer un tableau JavaScript.

- ☐ `let products = new Array('Smartphone', 'Tablette', 'PC')`
- ☒ `let products = ['Smartphone', 'Tablette', 'PC']`
- ☐ `let products = Array('Smartphone', 'Tablette', 'PC')`
- ☐ `let products = ('Smartphone', 'Tablette', 'PC')`
-  La bonne pratique consiste à déclarer un tableau JavaScript à l'aide de crochets [...].


Exercice

Le code ci-dessous va créer un tableau avec la valeur 3.

```
1 let array = new Array(3)
```

☐ Vrai

☒ Faux

 Faux, cette syntaxe va créer un tableau de longueur 3 avec 3 valeurs vides : [, ,].


Exercice

Les différents éléments d'un tableau sont séparés par :

☒ Une virgule

☐ Un point

☐ Un point-virgule


 On sépare les valeurs par des virgules. On ajoute un espace après pour une meilleure lisibilité.

Exercice

Qu'est-ce qu'un tableau à plusieurs dimensions ?

☒ Un tableau ayant pour valeurs d'autres tableaux

☐ Un tableau ayant pour valeurs différentes mesures


 Un tableau à plusieurs dimensions a pour valeurs d'autres tableaux.

Exercice

Pour connaître la taille/longueur d'un tableau, on utilise la propriété :

☐ size

☒ length

 C'est la propriété `length` qui permet de connaître la taille/longueur d'un tableau.

Exercice

Complétez le code suivant afin de retourner le premier élément du tableau `language`.

```
let language = ['JavaScript', 'HTML', 'CSS']
```

```
language[0]
```

Exercice

Le premier élément d'un tableau se situe à l'indice (index) 1.

☐ Vrai

☒ Faux

Q Faux, le premier élément se situe à l'index 0.

Exercice

Quelle syntaxe est correcte pour accéder à la troisième valeur d'un tableau ?

- ☐ array(3)
- ☐ array[3]
- ☒ array[2]
- ☐ array(2)

Q Comme le premier élément se situe à l'index 0, alors le troisième est à l'index 2. Et pour accéder directement à un élément du tableau, on utilise les crochets.

Exercice

Pour parcourir un tableau en JavaScript, vous pouvez utiliser :

- ☒ for
- ☐ browse
- ☒ forEach
- ☐ navigate
- ☐ explore

Q Pour parcourir un tableau JavaScript, il est possible d'utiliser `for` et `forEach`.

p. 16 Solution n°9

```
1 // Colors Array
2 let colors = ['Bleu', 'Rouge', 'Vert', 'Jaune', 'Blanc']
3
4 // 1. Answer -----
5 // Array length (5)
6 console.log(colors.length)
7
8 // 2. Answer -----
9 // Bleu
10 console.log(colors[0])
11
12 // Blanc
13 console.log(colors[4])
14
15 // Rouge
16 console.log(colors[1])
17
18 // 3. Answer -----
19 // for - Display all values
20 for (let i = 0; i < colors.length; i++) {
21   console.log(colors[i])
22 }
23
```

```
24 // forEach - Display all values
25 colors.forEach(function(color){
26     console.log(color)
27 })
28
29 // 4. Answer -----
30 // JavaScript Multidimensional Array
31 let colorsMulti = [
32     ['Bleu', 'Blanc', 'Rouge'],
33     ['Vert', 'Jaune']
34 ]
35
36 // 5. Answer -----
37 // Blanc
38 console.log(colorsMulti[0][1])
39
40 // Vert
41 console.log(colorsMulti[1][0])
42
43 // 6. Answer -----
44 // JavaScript Multidimensional Array - forEach - Display all values of
45 colorsMulti.forEach(function(arrayColors){
46     arrayColors.forEach(function(color){
47         console.log(color)
48     })
49 })
50 // JavaScript Multidimensional Array - for - Display all values of
51 for (let i = 0; i < colorsMulti.length; i++) {
52     for (let j = 0; j < colorsMulti[i].length; j++) {
53         console.log(colorsMulti[i][j])
54     }
55 }
```