
# Web Developer – Scripts Clients

## 05 – Les tableaux


# Généralités

Si nous voulons stocker n valeurs puis ensuite les afficher, nous devons utiliser une fonction d'affichage pour chaque valeur.

Le procédé est fastidieux et surtout n'est pas adaptable à un nombre variable d'éléments.

Un type de donnée peut remplacer avantageusement une série de variables : il s'agit des tableaux (ou Array en JS)

# Déclaration

Déclaration d'un tableau vide :

```
let tab1 = new Array( );
// ou
let tab2 = [ ]; // plus rapide
```

# Déclaration

Déclaration d'un tableau avec valeurs à l'initialisation :

```
let tab1 = new Array('1', '2', '3');


let tab2 = [4, 5, 6];


// Mais également
let tabMixte = ['7', 8, 'neuf'];
```

# Accès

Pour accéder à un élément particulier d'un tableau, nous devons utiliser son index.

Sachant que l'index du premier élément vaut 0 (notation 0-based), l'index d'un élément est égal à sa position dans le tableau – 1

L'index est à placer entre crochets après le nom de la variable

```
let tab1 = ['1', '2', '3'];
console.log(tab1[0]); // élément d'index 0
console.log(tab1[2]); // élément d'index 2
```

# Accès

Si on veut accéder à un index en dehors des limites du tableau, la valeur retournée vaut undefined :

```
let tab1 = ['1', '2', '3'];
console.log(tab1[4]); // élément d'index 4
```

# Accès

Pour modifier un élément, on affecte simplement une valeur à l'index désiré :

```
let tab1 = ['1', '2', '3'];

tab1[1] = 'deux';

console.log(tab1);
```

# Accès

Pour ajouter un élément, on affecte simplement une valeur au nouvel index désiré :

```
let tab1 = ['1', '2', '3'];

tab1[3] = 4;

console.log(tab1);
```

# Accès

Il est possible d'ajouter des éléments au-delà de la fin de celui-ci (ce qui donnera un tableau discontinu) :

```
let tab1 = ['1', '2', '3'];

tab1[10] = 4;

console.log(tab1);
```

# Accès

Pour ajouter un (ou des) élément au début du tableau , on utilisera la **méthode** unshift **:**

```
let tab1 = ['1', '2', '3'];

tab1.unshift('0');

console.log(tab1);
```

# Accès

Pour retirer un élément au début du tableau , on utilisera la **méthode** shift **:**

```
let tab1 = ['1', '2', '3'];

tab1.shift();

console.log(tab1);
```

# Accès

Pour ajouter un (ou des) élément à la fin du tableau sans utiliser l'index, on utilisera la méthode push :

```
let tab1 = ['1', '2', '3'];

tab1.push(4);

console.log(tab1);
```

# Accès

Pour retirer un élément à la fin du tableau, on utilisera la méthode pop :

```
let tab1 = ['1', '2', '3'];

tab1.pop();

console.log(tab1);
```

# Accès

Pour retirer un élément en n'importe quel endroit du tableau, on utilisera la méthode splice à qui on passe l'index à partir duquel supprimer et le nombre d'éléments à supprimer :

```
let tab1 = [1, 2, 3, 4, 5];

tab1.splice(2, 1);

console.log(tab1); // Affiche : [1, 2, 4, 5]
```

# Accès

Pour ajouter un élément en n'importe quel endroit du tableau, on utilisera la méthode splice à qui on passe l'index à partir duquel insérer, le nombre d'éléments à remplacer par l'insertion (0 si aucun) et la valeur à insérer :

```
let tab1 = [1, 2, 3, 4, 5];

tab1.splice(2, 0, 99);

console.log(tab1); // Affiche : [1, 2, 99, 3, 4, 5]
```

# Index

Pour déterminer l'index d'un élément sur base de sa valeur, on utilisera la méthode indexOf de l'objet Array. Si la valeur n'est pas trouvée, la méthode retourne -1.

```
let tab1 = [1, 2, 3, 4, 5];

let index = tab1.indexOf(3);

console.log(index); // Affiche 2
```

# Parcourir

Pour parcourir un tableau, on peut utiliser un « for-in » (pour énumérer les clés, ici, les index) :

```
let tabMixte = ['7', 8, 'neuf'];


for (let key in tabMixte) {
    console.log(key + ' : ' + tabMixte[key]);
}
```

Rem : À utiliser sur des variables simples, car si une variable contient des méthodes, elles seront également énumérées.

# Parcourir

Pour parcourir un tableau, on peut utiliser un « for-of » (pour énumérer les valeurs) :

```
let tabMixte = ['7', 8, 'neuf'];


for (let value of tabMixte) {
    console.log(value);
}
```

Rem : À utiliser sur des variables simples, car si une variable contient des méthodes, elles seront également énumérées.

# Parcourir

On peut également utiliser une boucle basée sur la taille du tableau.

Pour cela, nous utiliserons la propriété length :

```
let tabMixte = ['7', 8, 'neuf'];


for (let i = 0; i < tabMixte.length; i++) {
    console.log(tabMixte[i]);
}
```

Rem : Attention aux résultats obtenus avec les tableaux discontinus

# Parcourir

Pour des raison de performances, il ne faut pas utiliser la propriété length **à chaque itération :**

```
let tabMixte = ['7', 8, 'neuf'];

for (let i = 0, end = tabMixte.length; i < end; i++) {
    console.log(tabMixte[i]);
}
```

# Parcourir

La dernière forme est l'utilisation de la méthode forEach de l'objet Array, à qui il faut passer une fonction anonyme qui prend trois paramètres : la valeur, l'index et le tableau complet.

```
let tabMixte = ['7', 8, 'neuf'];

tabMixte.forEach(function (item, index) {
    console.log(index + " : " + item);
});
```

# Tri

Pour trier un tableau par ordre croissant, on utilisera la méthode sort :

```
let tabMixte = ['8', 7, 'neuf'];


console.log(tabMixte) ;
tabMixte.sort();
console.log(tabMixte) ;
```

# Tri

Si on teste cette méthode sur le tableau suivant [11, 8, 6, 1, 12], on constate que le tri est effectué selon un ordre lexicographique et pas numérique :

```
let tabMixte = [11, 8, 6, 1, 12];


tabMixte.sort();
console.log(tabMixte) ;
// affiche [1, 11, 12, 6, 8]
```

# Tri

Pour modifier la fonction utilisée pour la comparaison, on passera une fonction anonyme de callback à la méthode sort :

```
let tabMixte = [11, 8, 6, 1, 12];


tabMixte.sort(function(a, b){
      return a-b;
});
console.log(tabMixte) ;
// affiche [1, 6, 8, 11, 12]
```

Rem : les notions de fonction anonyme et de callback seront vues en détails ultérieurement

# Tri

Pour trier un tableau par ordre décroissant, on utilisera la méthode sort **suivie de la méthode** reverse **:**

```
let tabMixte = ['8', 7, 'neuf'];


console.log(tabMixte) ;
tabMixte.sort();
tabMixte.reverse();
console.log(tabMixte) ;
```

# Copie

Pour copier un tableau, on pourrait simplement affecter une variable Array à une autre variable :

```
let tab1 = [1, 2, 3, 4, 5];
let tab2 = tab1;

console.log(tab1);
console.log(tab2);

tab1[2] = 10;

console.log(tab1); // Affiche [1, 2, 10, 4, 5]
console.log(tab2); // Affiche [1, 2, 10, 4, 5]
```

# Copie

Si vous testez ce code, vous constaterez que les deux variables sont deux références vers un même tableau.

Pour réaliser une copie « profonde », on doit faire appel à la méthode slice :

```
let tab1 = [1, 2, 3, 4, 5];
let tab2 = tab1.slice();
console.log(tab1);
console.log(tab2);


tab1[2] = 10;
console.log(tab1); // Affiche [1, 2, 10, 4, 5]
console.log(tab2); // Affiche [1, 2, 3, 4, 5]
```

# Deux dimensions

Pour déclarer un tableau à deux dimensions, on écrira :

```
let tab2D = [
    ['00', '01', '02'],
    ['10', '11', '12'],
    ['20', '21', '22']
];
```

# Deux dimensions

Pour accéder à un élément :

```
// élément de la 2ème ligne 4ème colonne
console.log(tab2D[1][3]);
```

# Deux dimensions

Et pour le parcourir :

```
for (let i in tab2D) {
    ligne = '';
    for (let j in tab2D[i]) {
        ligne += tab2D[i][j] + ' ';
    }
    console.log(ligne);
}
```

# Filter

Array.filter **permet de filtrer les éléments selon certains critères (conditions) définis dans la fonction :**

```
let tab_result = tab_source.filter((value, index)=>{
  return **critères à définir**;
});
```

**Avec :**

- tab_source **est le tableau qui contient les données à filtrer**
- **La fonction contient les critères**
- value **et** index **sont les valeurs et l'index des éléments du tableau**

# Filter

```
let result = tab.filter((value, index) => {
  // ne retourne que les éléments qui commencent par la lettre p
  return (value[0] === 'p');
});
```

# Filter

```
let result = tab.filter((value, index) => {
  // ne retourne que les éléments qui sont différents des valeurs
  // premier et troisième
  return (value != 'premier' && value != 'troisième');
});
```

# Filter

```
let result = tab.filter((value, index) => {
  // ne retourne que les éléments dont la longueur vaut 7
  return (value.length === 7);
});
```

# Map

Array.map **permet d'appliquer un traitement à tous les éléments d'un tableau :**

```
let tab_result = tab_source.map(function(value, index) {});
```

**Avec :**

- tab_source **est le tableau qui contient les données à traiter**
- value **et** index **sont les valeurs et l'index des éléments du tableau**
- function **contient le traitement**

# Map

```
let tab = ['pomme', 'fraise', 'cerise', 'poire', 'banane', 'ananas'];
let result = tab.map(function(value, index) {
  // va mettre chaque valeur entièrement en majuscules
  return value.toUpperCase();
});
```


# Exercices

Voir fichier exercices_tableaux.pdf sur moodle