5: Tableaux

1. Tableaux, introduction

Un tableau (appelé `array` en anglais) est un type avancé de valeurs. C'est une liste ordonnée de valeurs.

Exemple:

```
var fruits = [ 'Mangue', 'Raisin', 'Figue' ];
```

Caractéristiques

Comme tous les autres types de valeur, un tableau peut être stocké dans une variable.

En revanche, en JavaScript, les tableaux sont considérés comme une forme particulière du type `object`. Autrement dit, `typeof []` retourne `"object"`.

Comme toute variable en JavaScript, les valeurs stockées dans un tableau peuvent être de n'importe quel type (y compris un tableau), et n'ont pas besoin d'être tous du même type:

```
var fourreTout = [ null, true, 'bonjour', 1.2, fruits, undefined ];
```

Tout tableau possède une propriété `length` qui vaut le nombre de valeurs qu'il contient:

```
fourreTout.length; // => retourne 6, car il y a 6 valeurs dans notre tableau
```

Accéder aux éléments d'un tableau

Chaque élément d'un tableau est numéroté par ce qu'on appelle un "indice" (ou "index" en anglais).

A noter que cette numérotation commence par 0 (zéro), et non par 1.

Du coup, l'indice du premier élément est `0`, et l'indice du dernier élément est `length - 1` (`length` étant le nombre d'éléments du tableau).

On peut accéder à un élément de tableau en précisant son indice entre crochets:

```
var fruits = [ 'Mangue', 'Raisin', 'Figue' ];
fruits[0]; // vaut 'Mangue'
fruits[1]; // vaut 'Raisin'
fruits[2]; // vaut 'Figue'
fruits[3]; // undefined
```

Modification d'un élément

Pour modifier la valeur d'un élément, il suffit de l'adresser par indice (comme vu juste avant) puis de lui affecter une valeur, comme on le ferait pour une variable:

```
var fruits = [ 'Mangue', 'Raisin', 'Figue' ];
fruits[1] = 'Kiwi';
fruits; // => vaut [ 'Mangue', 'Kiwi', 'Figue' ]
```

Ajout d'élément

La méthode `push()` permet d'ajouter un élément à la fin du tableau. La valeur de cet élément est à passer en paramètre (entre parenthèses) de la méthode, quand on l'appelle:

```
var fruits = [ 'Mangue', 'Raisin', 'Figue' ];
fruits.push('Banane'); // appel de la méthode push() sur le tableau fruits, avec `Banane` en paramètre
fruits; // => [ 'Mangue', 'Raisin', 'Figue', 'Banane' ]
```

À chaque fois qu'on ajoute un élément à un tableau en appelant la méthode `push()`, sa longueur `length` est incrémentée (c.a.d. sa valeur augmente de `1`):

```
fruits.length; // => 4, désormais
```

Retrait d'élément

La méthode `pop()` retourne la dernière valeur du tableau puis la retire de ce tableau:

```
var fruits = [ 'Mangue', 'Raisin', 'Figue' ];
var f = fruits.pop();
f;      // => `Figue`
fruits; // => [ 'Mangue', 'Raisin' ];
```

À chaque fois qu'on retire un élément d'un tableau en appelant la méthode `pop()`, sa longueur `length` est décrémentée (c.a.d. sa valeur décroit de `1`):

```
fruits.length; // => 2, désormais
```

Pratique: Calendrier

```
var jours = [ 'lun', 'OO7', 'mer', 'jeu', 'ven', 'sam', 'BUG' ];
```

Quelles instructions JavaScript faut-il exécuter pour effectuer les opérations suivantes ?

1. Retirer le dernière valeur du tableau `jours`
2. Afficher les valeurs du tableau dans la console
3. Ajouter la valeur `'dim'` à la fin du tableau
4. Remplacer la valeur `'OO7'` par `'mar'`
5. Afficher le nombre de valeurs du tableau dans la console
6. Afficher la troisième valeur du tableau dans la console

Quelle est la valeur finale du tableau, après avoir effectué toutes ces opérations ?

2. Tableaux, fonctions avancées

Recherche d'élément par valeur

La méthode `indexOf()` retourne l'indice du premier élément d'un tableau, pour une valeur donnée. La valeur recherchée est à passer en paramètre (entre parenthèses) de la méthode, quand on l'appelle:

```
var fruits = [ 'Mangue', 'Raisin', 'Figue', 'Raisin' ];
fruits.indexOf('Raisin'); // => 1
```

Si aucun élément du tableau ne contient la valeur cherchée, l'appel à `indexOf()` retourne `-1`:

```
fruits.indexOf('Pomme'); // => -1, car il n'y a pas de valeur 'Pomme' dans
fruits
```

Concaténation de tableaux

Comme pour les chaînes de caractères, il est possible de concaténer des tableaux.

La concaténation de deux tableaux consiste à créer un nouveau tableau contenant les éléments de ces deux tableaux.

La méthode `concat()` retourne un nouveau tableau contenant les éléments du tableau sur lequel elle est appelée, et d'un autre tableau passé en paramètre:

```
var fruits1 = [ 'Mangue', 'Raisin' ];
var fruits2 = [ 'Figue', 'Kiwi' ];
fruits1.concat(fruits2); // => [ 'Mangue', 'Raisin', 'Figue', 'Kiwi' ]
```

Il est bien entendu possible d'utiliser des tableaux littéraux, au lieu de variables:

```
[ 'Mangue', 'Raisin' ].concat([ 'Figue', 'Kiwi' ]); // => [ 'Mangue', 'Raisin',
'Figue', 'Kiwi' ]
```

Partitionnement de tableaux

La méthode `slice()` retourne un nouveau tableau contenant un extrait du tableau sur lequel elle est appelée.

A chaque appel de cette méthode, il faut fournir deux paramètres: l'indice de l'élément où commence cet extrait, et l'indice de l'élément où se termine l'extrait (non compris):

```
var fruits = [ 'Mangue', 'Raisin', 'Figue', 'Kiwi' ];
fruits.slice(1, 3); // => [ 'Raisin', 'Figue' ]
```

Altération de tableau

La méthode `splice()` (à ne pas confondre avec `slice()`) permet à la fois de supprimer et d'insérer des éléments dans un tableau, en fournissant leur indice(s).

Syntaxe d'appel de la méthode: `tableau.splice(i, c, v1, v2, ...)`, avec

- paramètre `i`: indice à partir duquel on va effectuer la modification
- paramètre `c`: nombre d'éléments à supprimer depuis l'indice `i`
- paramètre(s) suivant(s): valeur(s) d'élément(s) à insérer à partir de l'indice `i`

Voici un exemple utilisant les paramètres `i` et `c`: (suppression seulement)

```
var fruits = [ 'Mangue', 'Raisin', 'Figue', 'Kiwi' ];
fruits.splice(1, 2); // depuis l'indice 1, supprimer 2 valeurs
fruits; // => [ 'Mangue', 'Kiwi' ]
```

Voici un exemple utilisant tous les paramètres: (insertion seulement)

```
var fruits = [ 'Mangue', 'Raisin', 'Figue', 'Kiwi' ];
fruits.splice(1, 0, 'Pomme'); // depuis l'indice 1, supprimer 0 valeurs, puis y
insérer 'Pomme'
fruits; // => [ 'Mangue', 'Pomme', 'Raisin', 'Figue', 'Kiwi' ]
```

> Contrairement aux méthodes `concat()` et `slice()`, `splice()` ne retourne pas un nouveau tableau, mais modifie le tableau sur lequel elle est appelée.

Pratique: Épicerie

```
var fruits = [ 'Mangue', 'Raisin', 'Figue', 'Kiwi' ];
```

Écrire un programme qui:

1. Affiche la liste de `fruits` disponibles;
2. Demande au client quel fruit il désire acheter:

- s'il est présent dans le tableau `fruits`: le retirer du tableau, et afficher `'ok!'`,
- sinon, afficher `'indisponible...'`.

3. Affiche à nouveau la liste de fruits disponibles.