Stocker les données dans des tableaux

Introduction

Cette note de cours va vous faire découvrir les tableaux (ou Array), utilisés dans de nombreuses applications multimédias interactives pour stocker des groupes de données. L'organisation et le stockage des données sont des concepts fondamentaux de la programmation. Dans le contexte d'un jeu, il y a plusieurs avantages à ce que les éléments d'un jeu se trouvent dans un tableau. Par exemple, il est aisé de les parcourir en boucle puis, de vérifier chaque élément afin de retrouver des correspondances, de détecter des collisions, ou autres.

Le rôle et la création des tableaux?

Une des façons d'organiser les données dans la vie réelle consiste à établir des listes (épicerie, cadeaux à acheter, travaux à faire, etc.). Faisons-en une ici:

***Liste d'épicerie* :**

1. Lait
2. Pain
3. Beurre

Écrivons maintenant cette liste en JavaScript, sous forme d'un tableau :

```
let listeEpicerie = ["Lait", "Pain", "Beurre"];
```

Les tableaux sont la façon dont JavaScript crée des listes. Une façon de créer un tableau en JavaScript consiste à utiliser un **littéral de tableau**, comme dans l'exemple ci-haut. Ainsi, on crée un tableau à l'aide d'une **paire de crochets [ ]**. Tout ce qui se trouve entre les crochets correspond au contenu du tableau. Les différents **éléments** stockés **sont séparés** par des **virgules** ",".

Ainsi, **au lieu de créer plusieurs variables uniques,** comme par exemple: *let element1*, let *element2*, let *element3*, etc… un **tableau** contient **plusieurs valeurs** regroupées sous un **même nom**.

Les tableaux peuvent **stocker tous les types de données** (y compris des chaînes, des nombres, des booléens, des objets d'affichage, tels que des images ou n'importe quel autre type d'objets représentant des éléments d'un jeu).

Nous pouvons créer un tableau contenant des éléments ayant tous le même type:

```
let lesVilles = [ "New-York", "Beijing", "Montréal", "Paris", "Nairobi"];
let lesFantomesDePacman = [ "Blinky", "Pinky", "Inky", "Clyde"];
let lesPositionsAuHasard = [ 100, 200, 300, 400];
```

Nous pouvons également créer un tableau contenant différents types de données, comme par exemple :

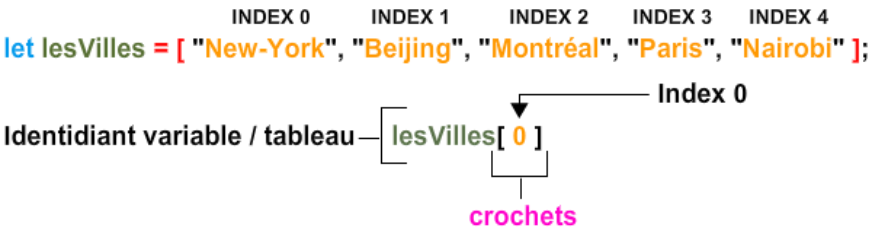
```
let lesElementsDuTableau = ["Bonjour", 7, { message: "Ça va bien!" }, true ];
```

Soulignons ici qu'en terme de **nomenclature**, puisqu'une **variable de type tableau** est destinée à **contenir plusieurs éléments,** une bonne pratique consiste à donner aux variables référençant des tableaux des **noms exprimant le pluriel**, comme par exemple: **ennemis**, **tabEnnemis**, ou encore: **lesEnnemis**.

Accéder aux éléments d'un tableau

Comme les listes, **les tableaux sont ordonnés**, ce qui signifie que **chaque élément** a une **position numérotée**, dénommée **index**. Nous pouvons accéder à des éléments individuels à l'aide de leur index, ce qui revient à référencer un élément dans une liste en fonction de la position de l'élément.

Les tableaux en JavaScript sont indexés à zéro, ce qui signifie que **les positions** commencent à compter **à partir de 0 ET NON PAS À PARTIR DE 1**. Par conséquent, le **premier élément** d'un tableau sera à la **position ou index 0**. Voyons comment nous pourrions accéder à un élément d'un tableau :



Dans l'exemple ci-dessus, **lesVilles** est une variable de type tableau (Array) qui comporte cinq éléments (index 0 à 4). Nous utilisons la notation entre crochets, **[ ]**, avec l'index après le nom du tableau pour accéder à l'élément: **lesVilles[0]**

lesVilles [0] accédera à l'élément à l'index 0 dans le tableau **lesVilles**. Vous pouvez imaginer que **lesVilles[0]** accèdent à l'espace en mémoire contenant la chaîne "*New-York*".

Il est possible de visualiser les éléments d'un tableau avec leur index dans la console, en utilisant la méthode **console.table()**. Par exemple, **console.table(lesVilles)**, affichera ceci dans la console:

(index)	Value
0	"New-York"
1	"Beijing"
2	"Montréal"
3	"Paris"
4	"Nairobi"

Mettre à jour des éléments d'un tableau

Dans l'exemple précédent, vous avez appris à **accéder aux éléments d'un tableau** à l'aide d'un **index**. Une fois que vous avez accès à un élément d'un tableau, vous pouvez aussi mettre à jour sa valeur. Par exemple:

```
let lesVilles = [ "New-York", "Beijing", "Montréal", "Paris", "Nairobi"];

lesVille[3] = "Londres";
console.log(lesVilles);
//Retourne:  [ "New-York", "Beijing", "Montréal", "Londres", "Nairobi"]
```

Ainsi, l'instruction : **lesVilles[3] = "Londres";** indique au programme de modifier l'élément à l'index 3 du tableau des villes pour qu'il devienne "*Londres*" au lieu de ce qui existait déjà ("*Paris*"). Une autre façon de le dire c'est : la chaîne "Londres" est affectée à la case 3 du tableau des villes.

La propriété .length

L'une des propriétés intégrées d'un tableau est sa **longueur**, ou sa taille, qui renvoie le **nombre d'éléments dans le tableau**. Nous accédons à la propriété **.length** avec la syntaxe à point. Par exemple, pour le tableau ***lesVilles***, vous pouvez connaître le nombre de villes indexées, comme suit :

```
let lesVilles = [ "New-York", "Beijing", "Montréal", "Paris", "Nairobi"];

console.log(lesVilles.length); //Retourne:  5
```

Accéder au dernier élément d'un tableau

Pour accéder au dernier élément d'un tableau, nous pouvons utiliser l'expression : **tableau.length - 1**, comme suit:

```
let lesVilles = [ "New-York", "Beijing", "Montréal", "Paris", "Nairobi"];

console.log(lesVilles[lesVilles.length - 1 ]); //Retourne "Nairobi"
```

Choisir un élément de tableau au hasard

Il est très fréquent dans un jeu d'afficher ou de choisir un élément au hasard. À l'aide de la propriété **.length** d'un tableau, et des méthodes de l'objet global **Math** de JavaScript, il est facile d'identifier un élément de tableau au hasard comme suit :

```
let lesVilles = [ "New-York", "Beijing", "Montréal", "Paris", "Nairobi"];

let indexVilleAuHasard = Math.floor(Math.random() * lesVilles.length);
```

```
/Si, par exemple, la valeur retournée pour indexVilleAuHasard est :  2, Alors:
console.log(lesVilles[indexVilleAuHasard ]);//Retourne: “ "Montréal"
```

Parcourir un tableau

Il existe plusieurs solutions pour **parcourir un tableau élément par élément**, notamment en utilisant des boucles de programmation (Voir la note de cours : *LesBouclesDeProgrammationAvecJavaScript.pdf*).

La **première** consiste à utiliser la **boucle for...of** qui est **spécialement créée** dans ce but. Cette boucle parcourt un objet de type Array (ou autre type *itérable*) et permet d'exécuter une ou plusieurs instructions pour la **valeur de chaque index**. Pour afficher la liste des villes présentes dans le tableau **lesVilles** avec la boucle **for...of,** le code sera le suivant:

```
let lesVilles = [ "New-York", "Beijing", "Montréal", "Paris", "Nairobi"];


for (let uneVille of lesVilles) {      //uneVille devient égale à chaque valeur du tableau
  console.log(uneVille);
}

//Retournera:
"New-York"
"Beijing"
"Montréal",
"Paris"
"Nairobi"
```

La **deuxième** consiste à utiliser la **boucle for**. L'exemple ci-dessous permet d'afficher la liste des villes présentes dans le tableau **lesVilles**:

```
let lesVilles = [ "New-York", "Beijing", "Montréal", "Paris", "Nairobi"];

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

```
//Retournera:
"New-York"
"Beijing"
"Montréal",
"Paris"
"Nairobi"
```

Quelques méthodes de tableau utiles

Il existe de nombreuses méthodes qui permettent la manipulation des éléments d'un tableau[1]. Il est par exemple possible d'ajouter de nouveaux éléments à un tableau, de retirer des éléments, de faire porter une recherche sur leur contenu, etc.

Apprenons quelques-unes des méthodes JavaScript intégrées qui facilitent le travail avec les tableaux. Ces méthodes sont spécifiquement appelées sur les tableaux pour simplifier les tâches courantes, telles que l'ajout et la suppression d'éléments.

Ajouter un élément à un tableau avec la méthode push()

La méthode **push ()** nous permet d'ajouter des éléments **à la fin d'un tableau**. Nous accédons à la méthode push() en utilisant la notation à point et en lui passant en paramètre l'élément à ajouter. Voici un exemple d'utilisation :

```
let lesVilles = [ "New-York", "Beijing", "Montréal", "Paris", "Nairobi"];

//On veut ajouter la ville de Toronto
lesVilles.push("Toronto");

console.log(lesVilles);
//Retourne:  [ "New-York", "Beijing", "Montréal", "Paris", "Nairobi", "Toronto"]
```

[1] Pour connaître l'ensemble des méthodes pour manipuler des tableaux, les curieux peuvent consulter le lien suivant: https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array

Identifier l'index d'un élément dans un tableau avec indexOf()

Dans un jeu, il arrive souvent que l'on veut identifier l'index d'un élément d'un tableau, le plus souvent pour supprimer l'élément du jeu à cet index.

Pour cela, nous devons utiliser la méthode **indexOf()** en lui passant en paramètre l'élément du tableau, dont on veut connaître l'index. Exemple:

```
let lesVilles = [ "New-York", "Beijing", "Montréal", "Paris", "Nairobi"];

//On veut connaître l'index de la ville de Montréal
let index = lesVilles.indexOf("Montréal")

console.log(index); //Retourne:  2
```

Supprimer un élément de tableau à un index donné

Pour supprimer un élément d'un tableau à un index spécifique, nous devons utiliser la méthode **splice()**.

La méthode **splice()** modifie le contenu d'un tableau en retirant des éléments à partir d'un index. Cette méthode gère **2 paramètres**: l'**index de départ** et le **nombre d'éléments** à supprimer à même le tableau à partir de cet index. On peut ainsi enlever plus d'un élément, mais il est plus fréquent d'enlever qu'un seul. Par exemple, dans le code qui suit, on veut enlever l'élément à l'index 2 ("*Montréal*"):

```
let lesVilles = [ "New-York", "Beijing", "Montréal", "Paris", "Nairobi"];

//On veut supprimer  l'élément à l'index 2
lesVilles.splice(2, 1)

console.log(lesVilles);
//Retourne:  [ "New-York", "Beijing", "Paris", "Nairobi"]
```