Création d'un tableau

Les tableaux ne sont pas des valeurs primitives. Cependant, nous ne sommes pas obligés d'utiliser le constructeur Array() avec le mot clef new pour créer un tableau en JavaScript.
Cette syntaxe utilise les crochets [...] comme ceci :

```
let prenoms = ['Pierre', 'Paul', 'Jacques'] ;
let ages = [25, 15, 41] ;
```

On a créé deux tableaux différents : le premier tableau est stocké dans une variable let prenoms. Par simplification, on parlera du « tableau prenoms ». Il contient des chaines de caractères (type de valeur String).
Le deuxième tableau ages contient des chiffres.

Accéder à une valeur dans un tableau

Lorsqu'on crée un tableau, un indice est automatiquement associé à chaque valeur du tableau. Chaque indice dans un tableau est toujours unique et permet d'identifier et d'accéder à la valeur qui lui est associée. Pour chaque tableau, l'indice 0 est automatiquement associé à la première valeur, l'indice 1 à la deuxième et etc.

Pour accéder à une valeur en particulier dans un tableau, il suffit de préciser le nom du tableau puis l'indice associé à la valeur à laquelle on souhaite accéder entre crochets.
Dans le cas où un tableau stocke un autre tableau, il faudra utiliser deux paires de crochets : la première paire va mentionner l'indice relatif à la valeur à laquelle on souhaite accéder dans notre tableau de base.

Exemple

```
let prenoms = ['Pierre', 'Paul', 'Jacques'];
let ages = [25, 15, 41] ;
console.log(prenoms[0] + ' a ' + ages[0] + ' ans.');
```

La console va renvoyer :

Pierre a 25 ans.

Utiliser une boucle for ... of pour parcourir toutes les valeurs d'un tableau

Pour parcourir un tableau élément par élément, on va pouvoir utiliser une boucle spécialement créée dans ce but qui est la boucle for...of .
Exemple :

```
let prenoms = ['Pierre', 'Paul', 'Jacques'];
let ages = [25, 15, 41] ;
for (valeur of prenoms)
  {
```

```
    console.log(valeur);
}
```

La console va renvoyer :

Pierre
Paul
Jacques

On définit une variable let valeur (on peut lui donner le nom que l'on souhaite) qui va stocker les différentes valeurs de notre tableau une à une. La boucle for…of va en effet exécuter son code en boucle jusqu'à ce qu'elle arrive à la fin du tableau.

Les propriétés et les méthodes du constructeur Array()

Le constructeurArray() ne possède que deux propriétés : la propriété length qui retourne le nombre d'éléments d'un tableau et la propriété prototype qui est une propriété que possèdent tous les constructeurs en JavaScript.

Array () possède également une trentaine de méthodes et certaines d'entre elles vont être très puissantes et vont pouvoir nous être très utiles. Nous allons ici étudier celles qu'il faut connaitre.

Les méthodes push() et pop()

La méthode push() va permettre d'ajouter des éléments en fin de tableau et va retourner la nouvelle taille du tableau. Nous passons les éléments à ajouter en argument.

La méthode pop() va elle nous permettre de supprimer le dernier élément d'un tableau et va retourner l'élément supprimé.

```
let prenoms = ['Pierre', 'Paul', 'Jacques'];
let ages = [25, 15, 41] ;
let taille = prenoms.push('Bob', 'Etienne');
let del = ages.pop();
for (valeur of prenoms)
  {
    console.log(valeur);
  }
console.log(del);
```

La console va renvoyer :

Pierre
Paul
Jacques
Bob
Etienne
41

Les méthodes unshift() et shift()

La méthode unshift() permet d'ajouter des éléments en début de tableau et retourne la nouvelle taille du tableau. Il faut passer les éléments à ajouter en argument.
La méthode shift() va elle permettre de supprimer le premier élément d'un tableau et va retourner l'élément supprimé.
Ces deux méthodes sont donc les équivalentes des méthodes push() et pop() à la différence que les éléments vont être ajoutés ou supprimés en début de tableau et non pas en fin.

```
let prenoms = ['Pierre', 'Paul', 'Jacques'];
let ages = [25, 15, 41] ;
let taille = prenoms.unshift('Bob', 'Etienne');
let del = ages.shift();
for (valeur of prenoms)
  {
     console.log(valeur);
  }
console.log(del);
```

La console va renvoyer :

Bob
Etienne
Pierre
Paul
Jacques
25

La méthode splice()

Pour ajouter, supprimer ou remplacer des éléments dans un tableau, on peut utiliser splice(). L'avantage de cette méthode est qu'elle permet d'ajouter, de supprimer ou de remplacer des éléments n'importe où dans un tableau.

Elle peut prendre trois arguments : une position de départ à partir d'où commencer le changement, le nombre d'éléments à remplacer et finalement les éléments à ajouter au tableau.
En précisant la position de départ 0, les changements seront effectués à partir du début du tableau. En précisant la position 1, ils se feront à partir du deuxième élément, etc.
En précisant une position négative, les changements seront faits en comptant à partir de la fin : -1 pour commencer en partant du dernier élément, -2 pour commencer en partant de l'avant dernier élément, etc.
Si on précise 0 en nombre d'éléments à remplacer, alors aucun élément ne sera supprimé du tableau de base. Dans ce cas, il sera nécessaire de préciser des éléments à rajouter.

Enfin, si on ne précise pas d'éléments à rajouter au tableau, le nombre d'éléments à remplacer tel quel précisé en deuxième argument seront supprimés du tableau à partir de la position indiquée en premier argument.
Cette méthode va également retourner un tableau contenant les éléments supprimés.

Exemple :

```
let prenoms = ['Pierre', 'Paul', 'Jacques'];
let ages = [25, 15, 41] ;
prenoms.splice(1, 0, 'Bob', 'Etienne');
let del = ages.splice(1, 1, 35, 55);
for (valeur of prenoms)
   {
      console.log(valeur);
   }

for (val of ages)
   {
      console.log(val);
   }
```

La console va renvoyer :

Pierre
Bob
Etienne
Paul
Jacques
25
35
55
41

La méthode join()

La méthode join() retourne une chaine de caractères créée en concaténant les différentes valeurs d'un tableau. Le séparateur utilisé par défaut sera la virgule mais on peut également passer le séparateur de notre choix en argument de join().

```
let prenoms = ['Pierre', 'Paul', 'Jacques'];
let ages = [25, 15, 41] ;

console.log(prenoms.join(' - '));
```

```
console.log(ages.join(' / '));
```

La console va renvoyer :

Pierre – Paul – Jacques
25 / 15 / 41

La méthode slice()

La méthode slice() renvoie un tableau créé en découpant un tableau de départ.
Cette méthode va prendre en premier argument facultatif la position de départ où doit commencer la découpe du tableau de départ. Si la position passée est un nombre négatif, alors le début de la découpe sera calculé à partir de la fin du tableau de départ. Si aucune position de départ n'est passée, la découpe commencera depuis le début du tableau de départ.
On peut également lui passer en second argument facultatif la position où doit s'arrêter la découpe du tableau de départ. Si la position passée est un nombre négatif, alors la fin de la découpe sera calculée à partir de la fin du tableau de départ. Si aucune position de fin n'est passée, alors on récupèrera le tableau de départ jusqu'à la fin pour créer notre nouveau tableau.
Exemple :

```
let prenoms = ['Pierre', 'Paul', 'Jacques', 'Etienne', 'Bob'];
let ages = [25, 15, 41, 55, 35] ;

let decoupe = prenoms.slice(2,4);
let coupe = ages.slice(2);

console.log(decoupe.join());
console.log(coupe.join());
```

La console va renvoyer :

Jacques,Etienne
41,55,35

La méthode concat()

La méthode concat() permet de fusionner différents tableaux entre eux pour en créer un nouveau qu'elle va renvoyer.
Cette méthode va prendre en arguments les tableaux que l'on souhaite concaténer à un premier de départ qu'on va pouvoir choisir arbitrairement.
Il est possible de fusionner autant de tableaux que l'on veut entre eux. Les tableaux de départ ne sont pas modifiés.

Exemple :

```
let prenoms = ['Pierre', 'Paul', 'Jacques'];
let ages = [25, 15, 41] ;

let tableau = prenoms.concat(ages);

console.log(tableau.join(' - '));
```

La console va renvoyer :
Pierre – Paul – Jacques – 25 – 15 - 41

La méthode includes()

La méthode includes permet de déterminer si un tableau contient une valeur qu'on va passer en argument. Si c'est le cas, includes() renvoie true. Dans le cas contraire, cette méthode renvoie false. Cette méthode est sensible à la casse (une majuscule est considérée comme une entité différente d'une minuscule).

Exemple :

```
let prenoms = ['Pierre', 'Paul', 'Jacques'];
let ages = [25, 15, 41, 55, 35] ;

let tableau = prenoms.concat(ages);
if (prenoms.includes('Paul'))
  {
    console.log('Paul est dans le tableau');
  }
```

La console va renvoyer :
Paul est dans le tableau