ES6 : les collections avec clés (set et map)

Avec ES5 JavaScript n'a qu'un seul type de collection : les tableaux. Le souci c'est que les tableaux ont juste une indexation numérique, on ne peut pas utiliser des clés. Alors il est toujours possible d'utiliser les objets, qui ne sont pas faits pour ça à la base, pour contourner cette limitation.

ES6 introduits deux types de collections dont une avec des clés pour combler cette lacune.

Les ensembles (set)

Un ensemble (**set**) est une collection de valeurs distinctes. On peut parcourir ces valeurs dans l'ordre et savoir si une valeur particulière est présente.

Création d'un ensemble

Un ensemble est très facile à créer avec le constructeur **Set** et à garnir avec la méthode **add** :

```
let ensemble = new Set();
ensemble.add(4);
ensemble.add('texte');
console.log(ensemble.size);  // 2
```

*On voit que la méthode **size** permet de connaître le nombre d'éléments contenus dans l'ensemble.*

On peut ajouter des valeurs directement dans le constructeur. Le code ci-dessus peut s'écrire ainsi :

```
let ensemble = new Set([4, 'texte']);
console.log(ensemble.size);  // 2
```

J'ai dit en introduction que les valeurs d'un ensemble doivent être distinctes, le constructeur se charge de vérifier cela :

```
let ensemble = new Set([4, 4]);
console.log(ensemble.size);  // 1
```

Test de présence d'un élément

Il est souvent nécessaire de savoir si un élément particulier est présent dans un ensemble, on a la méthode **has** pour le faire :

```
let ensemble = new Set();
ensemble.add(4);
ensemble.add('texte');
console.log(ensemble.has(4));  // true
console.log(ensemble.has('truc'));  // false
```

*Avec un tableau on devrait utiliser **indexOf**.*

Supprimer un élément

Pour supprimer un élément d'un ensemble il faut utiliser la méthode **delete** en précisant la valeur de l'élément :

```
let ensemble = new Set();
ensemble.add(4);
console.log(ensemble.has(4));  // true
ensemble.delete(4);
console.log(ensemble.has(4));  // false
```

Avec un tableau il faudrait utiliser l'indice et non pas la valeur, d'autre part il faudrait couper le tableau.

Parcourir un ensemble

Pour parcourir un ensemble on peut utiliser la méthode **forEach** :

```
let set = new Set([1, 2, 3]);
set.forEach(function(value) {
  console.log(value);  // 1  2  3
});
```

Pour un ensemble la méthode **forEach** accepte 3 arguments, comme c'est déjà le cas pour les tableaux :

- la valeur de l'élément,

- la valeur de l’élément,
- l’objet Set parcouru.

Mais pourquoi on a deux fois la valeur ?

C’est pour être homogène avec la signature de la méthode déjà existante pour les tableaux avec ES5.

Conversion en tableau

On a vu ci-dessus qu’il est facile de convertir un tableau en ensemble, il suffit d’envoyer le tableau dans le constructeur.

Mais on peut aussi faire l’inverse, convertir un ensemble en tableau :

```
let set = new Set([1, 2, 3]);
let tableau = Array.from(set);
console.log(tableau);  // 1,2,3
```

Les maps

Le deuxième type de collection introduit par ES6 est l’objet **Map**. Contrairement aux ensembles on a là des clés et des valeurs qui peuvent être de n’importe quelle nature.

Création d’un map

Un **map** est très facile à créer avec le constructeur **Map**, à garnir avec la méthode **set** et pour récupérer un élément avec la méthode **get** :

```
let map = new Map();
map.set('prenom', 'Pierre');
map.set('nom', 'Durand');
console.log(map.get('prenom')); // Pierre
console.log(map.get('nom')); // Durand
```

On a ici enregistré deux couples clé/valeur.

On peut ajouter des valeurs directement dans le constructeur. Le

code ci-dessus peut s’écrire ainsi :

```
let map = new Map([['prenom', 'Pierre'], ['nom', 'Durand']]);
console.log(map.get('prenom')); // Pierre
console.log(map.get('nom')); // Durand
```

Test de présence d’un élément

Comme pour les ensembles on peut savoir si un élément particulier est présent dans un ensemble avec la méthode **has** :

```
let map = new Map();
map.set('prenom', 'Pierre');
map.set('nom', 'Durand');
console.log(map.has('nom')); // true
console.log(map.has('age')); // false
```

Supprimer un élément

Pour supprimer un élément il faut utiliser aussi la méthode **delete** en précisant cette fois la clé (ce qui supprime aussi la valeur associée) :

```
let map = new Map();
map.set('prenom', 'Pierre');
console.log(map.has('prenom')); // true
map.delete('prenom');
console.log(map.has('prenom')); // false
```

On dispose également de la méthode **clear** pour supprimer tous les éléments d’un coup :

```
let map = new Map();
map.set('prenom', 'Pierre');
map.set('nom', 'Durand');
console.log(map.size); // 2
map.clear();
console.log(map.size); // 0
```

*On voit au passage la méthode **size** qui renvoie le nombre d’éléments.*

Parcourir un map

Pour parcourir un **map** on peut utiliser la méthode **forEach** :

```
let map = new Map([['prenom', 'Pierre'], ['nom', 'Durand']]);
map.forEach(function(value, key) {
  console.log('clé : ' + key + ', valeur : ' + value);
});
```

Ce qui donne :

```
clé : prenom, valeur : Pierre
clé : nom, valeur : Durand
```

*On dispose d'un troisième paramètre pour connaître le **map** parcouru.*

En résumé

- ES6 nous offre deux nouveaux types de collections : les ensembles (**set**) et les **map**.
- Un ensemble (**set**) est une collection de valeurs distinctes itérable.
- Un **map** est une collection de clés/valeurs distinctes itérable.
- Pour ces deux collections on peut ajouter ou supprimer des éléments et savoir si un élément existe.