Javascript

les tableaux

Définition

- Les tableaux javascript sont des objets semblables à des listes qui possèdent plusieurs méthodes incorporées pour exécuter des opérations de parcours et de modification.
- Ni la taille d'un tableau ni les types de ses éléments ne sont fixés.

Création

Constructeur

```
let tab = new Array(); //Création d'un tableau vide;
let tab2 = new Array("toto","tata","titi"); //éléments du tableau
let tab3 = new Array(12); //longueur du tableau
```

Notation littérale

A privilégier

```
let tab = []; //Création d'un tableau vide;
let tab2 = ["toto","tata","titi"]; //éléments du tableau
```

Accès aux éléments

Les tableaux javascript sont des objets possédant des noms de propriété numériques et une propriété `length`.

La notation avec point ne permettant pas un nom commençant par un chiffre, on utilise la notation avec crochets.

```
let tab = ["tata","toto","titi"];
tab.0; //erreur de syntaxe
tab[0]; //accès à la propriété 0 => "tata"
tab[1]; //accès à la propriété 1 => "toto"

tab.length; //3
```

Accès aux éléments

La méthode at permet d'obtenir un élément du tableau en partant de la fin si l'argument est négatif.

```
let tab = ["tata","toto","titi"];

tab.at(0); //accès à la propriété 0 => "tata"
tab.at(-1); //accès à la dernière propriété => "titi"

tab[tab.length - 1]; // équivalent sans la méthode at
```

Enumération

for...of

```
let tab = ["tata","toto","titi"];

for (let item of tab) {
  console.log(item);
}
```

forEach

```
let tab = ["tata","toto","titi"];

tab.forEach((item,i) => console.log(`item n°${i} : ${item}`));
```

Enumération

~~for...in~~

Ne jamais utiliser la boucle `for...in` pour énumérer un tableau

```
let tab = ["tata","toto","titi"];

for (let n in tab) console.log(tab[n]);

//Pourquoi puisque ça marche ?!
```

Enumération

~~for...in~~

```
let tab = ["tata","toto","titi"];

tab.nom = "Mes prénoms";

for (let n in tab) console.log(tab[n]);

//ça ne fonctionne plus
```

Accesseurs

Ces méthodes ne modifient pas le tableau.

- `concat` : renvoie un tableau constitué de ce tableau concaténé avec un ou d'autres tableaux et/ou valeurs.
- `join` : concatène les éléments en une chaîne de caractères.
- `slice` : extrait une portion du tableau pour retourner un nouveau tableau constitué de ces éléments.
- `indexOf` : retourne le plus petit index d'un élément égal à la valeur passée en paramètre, ou -1 si aucun n'a été trouvé.
- `lastIndexOf` : retourne le plus grand index d'un élément égal à la valeur passée en paramètre.
- `includes` : détermine si le tableau contient ou non un certain élément.

Mutateurs

Ces méthodes modifient le tableau.

- `pop` : supprime le dernier élément et retourne cet élément.
- `push` : ajoute un ou des éléments à la fin du tableau.
- `reverse` : renverse l'ordre des éléments.
- `shift` : supprime le 1er élément et retourne cet élément.
- `sort` : trie en place les éléments.
- `splice` : supprime et/ou ajoute des éléments.
- `unshift` : ajoute un ou des éléments au début du tableau.

Equivalents sans mutation

- `toReversed` : copie le tableau en renversant l'ordre des éléments.
- `toSorted` : copie le tableau en triant les éléments.
- `toSpliced` : copie le tableau en supprimant et/ou en ajoutant des éléments.
- `with` : copie le tableau en remplaçant un élément.

Itération

- `forEach` : appelle une fonction sur chacun des éléments.
- `every` : renvoie true si chaque élément satisfait la fonction.
- `some` : renvoie true si au moins un élément satisfait la fonction.
- `filter` : crée un tableau contenant tous les éléments satisfaisant la fonction.
- `map` : crée un tableau contenant les images de chaque élément par la fonction.
- `reduce` : applique une fonction sur un accumulateur et sur chaque élément de façon à obtenir une unique valeur.

➡ Référence

Reconnaître un tableau

Array.isArray

```
let tab = ["tata","toto","titi"];

let tab2 = { "0" : "tata", "1" : "toto", "2" : "titi", length : 3 };

Array.isArray(tab); //true;

Array.isArray(tab2); //false;
```

Syntaxe de décomposition

Affecter par décomposition

```
let coord = [0.25, 45.56];

let [lon, lat] = coord;

lon; //0.25
lat; //45.56
```

Equivalent à :

```
let coord = [0.25, 45.56];

let lon = coord[0];
let lat = coord[1];
```

Syntaxe de décomposition

Affecter par décomposition

```
let chaine = "jacques;durand;23;12;1967";

let tab = chaine.split(";");

let [prenom, nom, ...naissance] = tab;

prenom; //jacques
nom; //durand
naissance; //["23", "12", "1967"]
```

Syntaxe de décomposition

Omettre certaines valeurs

```
let chaine = "jacques;durand;23;12;1967";

let tab = chaine.split(";");

let [,, ...naissance] = tab;

naissance; //["23", "12", "1967"]
```

Syntaxe de décomposition

Appels de fonction

```
function translateCoord(lon, lat) {
    return [lon + 10, lat + 5];
}

let coord = [0.25, 45.56];

let newCoord = translateCoord(...coord);
```

Syntaxe de décomposition

Concaténation de tableaux

```
let articulations = ["épaules", "genoux"];

let corps = ["têtes", ...articulations, "bras", "pieds"];

corps; // ["tête", "épaules", "genoux", "bras", "pieds"]
```

Syntaxe de décomposition

Copie de tableau

Rappel

```
let coord = [0.25, 45.56];
let newCoord = coord;
newCoord[0] = 0.5;
coord[0]; // 0.5
```

Solution

```
let coord = [0.25, 45.56];
let newCoord = [...coord];
newCoord[0] = 0.5;
coord[0]; // 0.25
```