JavaScript - Les objets

Qu’est-ce qu’un objet en JS?

Un type de données, comme `Boolean`, `Number`, `String`.
Sauf que contrairement aux autres types qui ne contiennent qu'une seule valeur (un nombre, une chaine de caractère etc...), **un objet est la collection de plusieurs valeurs**.

Exemple: En JS, **les tableaux sont considérés comme des objets**, car ils ont le potentiel d'être un stockage pour plusieurs valeurs

Déclarer un objet

La façon littérale

```
let me = { // Object me
    firstname: "Jean-Baptiste", // value "Jean-Baptiste" in key firstname
    lastname: "Lavisse",  // value "Lavisse" in key lastname
    age: 29 , // value 29 in key age

    sayHi() {
        alert("Hello");
    }
};

console.log(me.firstname); // Jean-Baptiste
console.log(me.lastname); //Lavisse
console.log(me.age); // 29
```

Un peu de vocabulaire

On peut rajouter des données à notre objet grâce à un **système de clé/valeur**, que l'on nomme `propriété`.

```
firstname: "Jean-Baptiste"
```

On peut rajouter des fonctions à notre objet. On les nomme `méthode`.

```
sayHi: function() {
    alert("Hello");
}
```

```
// Add a property after object initialisation
me.isTeacher=true;
console.log(me.isTeacher); //true

//Delete a property
delete me.age;

/* Multiword property */
me.likes kebab = true; // Syntax error
me["likes kebab"]=true; // Square bracket notation

// Test if "key" in object
console.log("firstname" in me); // true, me.firstname exists

// loop on each object's key
for(key in me)
{
    console.log(key); // firstname, lastname, isTeacher, etc...
    consoe.log(me[key]); //Jean-Baptiste, Lavisse, true, etc...
}
```

Les méthodes d'objet et `this`

Dans une méthode, on peut accéder aux propriétés et autres méthodes d'un objet grâce au mot-clé `this`.
`this` est une référence à l'objet courant.

```
let me = {
  name: "Jean-Baptiste",

  sayHi() {
    alert(this.name);
  }
};

user.sayHi(); // Jean-Baptiste
```

La méthode constructeur et `new`

Souvent, on veut créer des objets similaires, comme une voiture ou un stylo.
On peut le faire grâce aux **fonctions constructeurs**

Différences d'une fonction constructeur:

- Doit être nommée avec la première lettre en majuscule
- Est exécutée grâce au mot-clé `new`

```
function Car(model) {
    this.model = model;
    this.nbWheels = 4;
}


let madCar = new Car("Audi");
console.log(madCar); // {model:"Audi",nbWheels:4}
console.log(madCar.nbWheels); // 4

let maxCar = new Car("Mazda");
console.log(maxCar); // {model:"Mazda",nbWheels:4}
```

Processus détaillé

```
function Car(model) {
    this.model = model;
    this.nbWheels = 4;
}

//IS THE SAME AS

function Car(model) {
  // this = {};  (implicitly)

  // add properties to this
  this.model = model;
  this.nbWheels = 4;

  // return this;  (implicitly)
}
```

Les méthodes des primitives

Les données primitives de type `number`, `string`, `boolean`, etc..peuvent disposer aussi de méthodes.

```
"Coucou".toUpperCase(); //COUCOU
```

Le processus:
Une valeur primitive qui fait appel à une méthode crée un objet temporaire du type de la valeur, appelle la méthode retourne le résultat et supprime l'objet temporaire

Les méthodes `number`

```
let nb=12.353;

// .toString(base) -> chaine de caractère dans le système numérique de la base
// Base par défaut: 10
nb.toString(); // "12.356"

// .toFixed(n) -> nombre à n chiffres après la virgule
nb.toFixed(2); // 12.35

// Number(el) : transforme `el` en nombre
Number(true); // 1

//  parseInt() et parseFloat() -> Retourne un entier
// ou un nombre réel depuis une chaine de caractère
parseInt("3.14"); // 3
parseInt("3.14"); // 3.14
```

Les méthodes `string`

- `toLowerCase()` : tout en minuscule
- `toUpperCase()` : tout en majuscule
- `charAt(n)` : Quel caractère à la position `n`
- `indexOf("n")` : Première occurrence du caractère `n`
- `lastIndexOf("n")` : Dernière occurrence du caractère
- `replace("ef","il)` : Remplace dans la chaine `ef` par `il`
- `trim()` : Enlève l'espace inutile
- `split(',')` : sépare une `string` en tableau à chaque `,`

NB: `mystr[1]` affiche le 2ème caractère de la chaîne!

Les méthodes et propriétés `array`

- `length` : Donne le nombre d'éléments d'un tableau
- `push()` : ajoute un élément au tableau à la fin et retourne la nouvelle taille du tableau
- `pop()` : Supprime un élément du tableau à la fin et retourne la valeur supprimée en `string`
- `unshift():` ajoute un élément en début de tableau et retourne la nouvelle taille du tableau
- `shift()` : enlève le premier élément du tableau et retourne la valeur supprimée

- `splice(1, 2,"chlo","é")` : supprimer 2 éléments du tableau à partir de la position 1 et met à la suite de la position 1 `chlo` et `é`
- `slice(1,3)` : crée une copie du tableau de l'index 1 à 3
- `sort()` : trie un tableau mais en comparant chaque caractère (25 > 100 car 2 > 1)
- `reverse()` : retourne le tableau inversé
- `join("|")` : retourne une chaine de caractères comprenant tous les éléments du tableau avec `|` en séparation
- `concat(tab2)` : concatène un tableau avec un autre
- `forEach(f)` : Exécute une fonction `f` pour chaque élément du tableau

- `indexOf/lastIndexOf(item, pos)` – Cherche le premier/dernier élément trouvé qui correspond à `item` à partir de `pos`. Retourne l'index ou `-1` si non trouvé
- `includes(value)` – Retourne true si la valeur existe dans le tableau, false sinon