Constructeurs d'Objets en JS :

- Utilisation d'un constructeur d'objet en utilisant le mot clé new :
  Dans le chapitre précédent nous avons créer un objet en utilisant la syntaxe littérale d'objet (initialiseur)

```
function ObjectName(name){
        this.name = name ;
}
// syntaxe du constructeur d'objet
Var MyObject = new ObjectName("My Object ") ;
```

- Exemple de compréhension :

```
-
- function Person(name, age, color) {
-     this.name = name;
-     this.age = age;
-     this.colorFav = color;
- }
-
- // on cré deux objet type Peron, --> instanciation de deux objets Person
- var Michel = new Person('Michel', 42, 'Bleu');
- var Jean = new Person('Jean', 39);
-
- // les propriétes des deux Objet sont accéssible soir avec le poit ou
  crochets.
- console.log("Using Constructor of Michel Object : ", Michel);
- console.log("Length of object Michel : ",
  Object.getOwnPropertyNames(Michel).length);
- console.log("Type Of Object : ", typeof Michel);
-
- console.log("Using Constructor of Jean Object : ", Jean);
- console.log("length of Jean Object : ",
  Object.getOwnPropertyNames(Jean).length);
- console.log("Type of Jean Object : ", typeof Jean);
```

```
Using Constructor of Michel Object :  ▼ Person {name: 'Michel', age: 42, colorFav: 'Bleu'} i
                                        age: 42
                                        colorFav: "Bleu"
                                        name: "Michel"
                                      ▶ [[Prototype]]: Object
Length of object Michel :  3
Type Of Object :  object
Using Constructor of Jean Object :  ▼ Person {name: 'Jean', age: 39, colorFav: undefined} i
                                      age: 39
                                      colorFav: undefined
                                      name: "Jean"
                                    ▶ [[Prototype]]: Object
length of Jean Object :  3
Type of Jean Object :  object
```

Les constructeurs d'objets en JS :

Les constructeurs et les fonctions sont étroitement liés, un constructeur d'objet n'est rien d'autre qu'un chemin vers une meilleure création d'objets.
En termes de code un constructeur est assez similaire à une fonction qui renvoie un objet

Création d'un objet en JS en utilisant un constructeur d'objet avec la méthode

```
function chien(nom, race, poids) {
   this.nom = nom;
   this.race = race;
   this.poids = poids;
   this.bark = function () {
      if (this.poids > 25) {
         document.write(this.name + "dit Woof " + '<br/>');
      } else {
         document.write(this.name + "dit Yip ! " + '<br/>');
      }
   }
};

var fido = new chien("Fido", "Mixte", 36);
var fluffy = new chien("Fluffy", "caniche", 33);
var spot = new chien("Spot", "Chihuahua", 10);
var chiens = [fido, fluffy, spot];
for (var i = 0; i < chiens.length; i++) {
   chiens[i].bark();
}
```

Le constructeur a la base est une fonction, on peut l'appeler sans utiliser le mot clé new, mais cela peut provoquer un code bogué.
Supposant qu'on appelle fido sans le mot clé new.

```
Var fido = chien (''Fido'', ''Mixte '' , 36) ;
Fido.bark() ;
```
← cela produit une erreur il est impossible de lire la propriété ''bark() ''

Explication :
Le new crée un nouvel objet avant de l'assigner a this puis appeler la fonction constructeur
Si le new n'est pas utilisé aucun objet n est créé. → Cela signifie que toutes les références a this dans le constructeur ne feront référence à un nouvel objet, mais plutôt a l'objet global – l'objet global est l'objet de niveau supérieur –

NB :
A l'utilisation de plusieurs paramètres dans le constructeur, plus nous en ajoutons la lecture devient difficile parce que nous devons assurer que tous les arguments sont dans le bon ordre. Donc il existe une technique courante pour faciliter le passage de tous les arguments qui agisse sur le constructeur

```
function Car(params) {
   this.Marque = params.Marque;
   this.Modele = params.Modele;
   this.annee = params.annee;
```

```
    this.passagers = params.passagers;
    this.convertible = params.convertible;
    this.kilometrage = params.kilometrage;
    this.started = false;
    this.stop = function () {
        this.started = false;
    }
    this.start = function () {
        this.started = true;
    };
    this.drive = function () {
        if (this.started) {
            console.log("Zoom and ZZooom")
        }
        else {
            console.log("You need to start the engine")
        }
    };

};


var cadiParams = {
    Marque: "GM",
    Modele: "Cadillac",
    annee: 1955,
    passagers: 5,
    convertible: false,
    kilometrage: 12892
};

var cadi = new Car(cadiParams);
cadi.start();
cadi.drive();
cadi.stop();
```