Création d'objet JS à l'aide de la méthode Object.create() ;

➔ La Méthode Object.create() crée un nouvel Objet JS dont les propriétés peuvent être initialisées à l'aide de la notation (.) ou crochets ([]).
On peut créer des new Object qui permet de spécifier l'objet prototype (possibilité d'héritage des propriétés et du comportement d'autre Objets.

➔ Cette Méthode est très utile, elle permet de choisir l'objet prototype pour l'objet que nous souhaitons créer sans avoir besoin de définir une fonction constructeur.

NB : En JS, tous les objets peuvent hériter des propriétés et du comportement d'autre Objets (objet hérité s'appelle Prototype accédé par objectName.__proto__)

```
Var myObj = Object.create({}) ;
myObj.propName = valeur ;  // notation par dot (point)
myObj[''propName''] = valeur ;  // notation par crochets
```

Exemple :

```
Var Voiture = {
    Model : '' Renault''
    Color : ''Noir ''
}
Var HybridVoiture = Object.create(Voiture) ;
Console.log(hybridVoiture.Model) ;
```

On a utilisé l'objet Voiture comme prototype pour créer un autre objet (HybridVoiture). L'objet HybridVoiture aura toutes les propriétés de l'objet Voiture.

Exemple de compréhension :

```
// creation d'objet en JS en utilisant la méthode Object.create()
var voiture = {
   model: 'Peugeot',
   color: 'Noire',
   type: 'SUV',
   displayType: function () {
      return this.type;
   }
};
var ElectricVoiture = Object.create(voiture);
ElectricVoiture.displayType();

var BerlineVoiture = Object.create(voiture);
BerlineVoiture.type = 'Peugeot 208';
BerlineVoiture.displayType();

console.log("Using methode Object.create() **  voiture **  --> ", voiture);
console.log("length voiture Object --> ", Object.getOwnPropertyNames(voiture).length);
console.log("Type de l'objet voiture  --> ", typeof voiture);

console.log(" ----------------------------------------------------");
```

```
console.log("using methode Object.create() ** electricVoiture ** sans __proto__ ---> ",
ElectricVoiture);
console.log("using methode Object.create() ** electricVoiture ** avec __proto__ --> ",
ElectricVoiture.__proto__);
console.log("length de l'objet ElectricVoiture --> ",
Object.getOwnPropertyNames(ElectricVoiture.__proto__).length);
console.log("type de l'objet ElectricVoiture --> ", typeof ElectricVoiture);

console.log(" ----------------------------------------------------");
console.log("using methode Object.create() ** BerlineVoiture ** sans __proto__  --> ",
BerlineVoiture);
console.log("using methode Object.create() ** BerlineVoiture ** avec __proto__  -->",
BerlineVoiture.__proto__);
console.log("length de l'objet BerlineVoiture --> ",
Object.getOwnPropertyNames(BerlineVoiture.__proto__).length);
console.log("type de l'objet BerlineVoiture --> ", typeof BerlineVoiture);

console.log(" ----------------------------------------------------");
console.log("Acces Aux données :")
console.log(" ----------------------------------------------------");

console.log("type value of ElectricVoiture" + ElectricVoiture.displayType());
console.log("ElectricVoiture.model: " + ElectricVoiture.model + " ---  ElectricVoiture.color: " +
ElectricVoiture.color + " --- ElectricVoiture.type:  " + ElectricVoiture.type);
console.log("ElectricVoiture['model']:  " + ElectricVoiture['model'] + " --- ElectricVoiture['color']:  " +
ElectricVoiture['color'] + " --- ElectricVoiture['type']: " + ElectricVoiture['type']);

console.log("erlineVoiture.model: " + BerlineVoiture.model + " --- BerlineVoiture.color: " +
BerlineVoiture.color + " ---BerlineVoiture.type:  " + BerlineVoiture.type);
console.log(" BerlineVoiture['model']:  " + BerlineVoiture['model'] + " --- BerlineVoiture['color']: " +
BerlineVoiture['color'] + " ---BerlineVoiture['type']: " + BerlineVoiture['type']);
```

```
Using methode Object.create() **  voiture **  -->  ▶ {model: 'Peugeot', color: 'Noire', type: 'SUV', displayType: ƒ}
length voiture Object -->  4
Type de l'objet voiture  -->  object
 ----------------------------------------------------
using methode Object.create() ** electricVoiture ** sans __proto__ --->  ▶ {}
using methode Object.create() ** electricVoiture ** avec __proto__ -->
▶ {model: 'Peugeot', color: 'Noire', type: 'SUV', displayType: ƒ}
length de l'objet ElectricVoiture -->  4
type de l'objet ElectricVoiture -->  object
 ----------------------------------------------------
using methode Object.create() ** BerlineVoiture ** sans __proto__  -->  ▶ {type: 'Peugeot 208'}
using methode Object.create() ** BerlineVoiture ** avec __proto__  -->
▶ {model: 'Peugeot', color: 'Noire', type: 'SUV', displayType: ƒ}
length de l'objet BerlineVoiture -->  4
type de l'objet BerlineVoiture -->  object
 ----------------------------------------------------
Acces Aux données :
 ----------------------------------------------------
type value of ElectricVoitureSUV
ElectricVoiture.model: Peugeot ---  ElectricVoiture.color: Noire --- ElectricVoiture.type:  SUV
ElectricVoiture['model']:  Peugeot --- ElectricVoiture['color']:  Noire --- ElectricVoiture['type']: SUV
erlineVoiture.model: Peugeot --- BerlineVoiture.color: Noire ---BerlineVoiture.type:  Peugeot 208
 BerlineVoiture['model']:  Peugeot --- BerlineVoiture['color']: Noire ---BerlineVoiture['type']: Peugeot 208
```

Dans cet exemple on a créé un objet ElectricVoiture en utilisant l'objet voiture comme prototype, donc l'objet ElectricVoiture aura toutes les propriétés de l'objet voiture.

Une possibilité de créer l'objet a partir de Object.create(), Lorsqu'on appelle directement l'objet ElectricVoiture pour voir son contenu on reçoit un objet vide car toutes les propriétés hérités sont contenues dans __proto__ c'est-à-dire on accède par ElectricVoiture.__proto__

De la même manière on a créé l'objet BerlineVoiture en utilisant l'objet Voiture comme prototype et la méthode Object.create() pour créer l'héritage entre les deux objets.

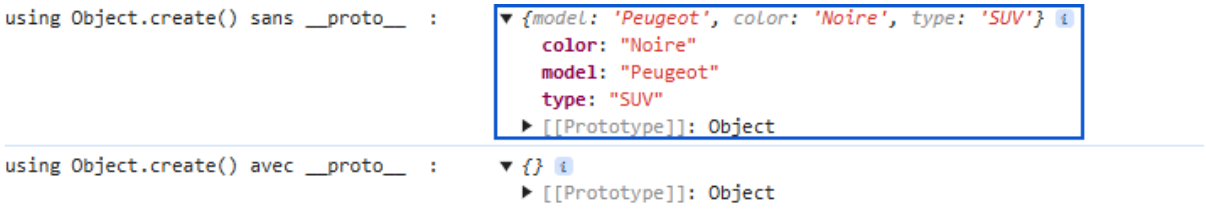
Comme on a modifié le type de l'objet BerlineVoiture à l'appel direct de l'objet BerlineVoiture on reçoit la propriété {type : 'Peugeot 208'} et pour lire le reste des propriétés de BerlineVoiture on appelle __proto__ de l'objet (BerlineVoiture.__proto__)

Création d'un Objet en JS en utilisant la méthode Object.create() avec un prototype nul

Dans ce cas on utilisera un prototype vide pour créer un objet avec la méthode Object.create()
Ce qui implique que l'objet n'héritera aucune fonction / propriété d'un autre objet

```
// creation d'objet en JS en utilisant la méthode Object.create()
var voiture = Object.create({});
voiture.model = 'Peugeot';
voiture['color'] = "Noire";
voiture.type = 'SUV';

console.log("using Object.create() sans __proto__  :  ", voiture);
console.log("using Object.create() avec __proto__  :  ", voiture.__proto__);
```



la différence entre un tableau et l'objet en JS.

➔ Array is a subclass or sub-prototype of object ➔ Arrays are denoted by square brackets [] ➔ In JS arrays use numbered index ➔ An object of array is an object of Object	➔ Object is the superclass of JS ➔ Object is denoted with curly bracket {} ➔ In JS Objects use named index ➔ An object of Object is not an object of Array is an Object