# JavaScript - Programmation objet et JSON

[Mise à jour le 11/8/2021]

- **Sources** et compléments sur **MDN Web Docs**
  - Le JavaScript orienté objet pour débutants
  - JSON
- **Lectures connexes**
  - Wikis WebPEM : "Préparer un projet de site Web avec l'IDE VSCode"

---

Les objets sont des ensembles de **paires clé-valeur**. La clé est une chaîne de caractères permettant d'accéder à la valeur. Il est ainsi possible d'enregistrer plusieurs informations pour un même objet. Les accolades et leur contenu sont appelés **objet littéral**.

## 1. Créer un objet

Pour créer un objet, il faut insérer des accolades (**{ }**). La **clé** se place avant les "**:**" (qui signifient = ) et la valeur après. La **valeur** est assignée à la clé. Les différentes paires de ***clés-valeur*** sont séparées par des "**,**".

*Exemples*

*.js

```
var objet = {};  // Objet vide

var voiture = {     // Objet littéral, car voiture est définie en une seule fois
    marque :"Peugeot",
    type : "3008",
    couleur: "rouge",
    nombre: 2,
    "En vente " : "oui"
};
```

Les clés étant obligatoirement des chaînes de caractères, il n'est pas nécessaire de les placer entre guillemets sauf si on souhaite qu'elles contiennent des **caractères spéciaux** comme l'espace.

### 2. Valeurs d'un objet

Pour accéder aux valeurs d'un objet, il suffit de placer la clé entre des guillemets et dans des crochets ou de la **lier** avec un **point**.

*Exemples*

```
var voiture = {
    marque :"Peugeot",
    type : "3008",
    couleur : "rouge",
    nombre : 2,
    "En vente " : true
};
// Première possibilité (même principe que les tableaux) ou deuxième possibilité (en utilisant un lien : le point)
voiture["marque"]; // résultat dans la console : "Peugeot"
voiture.marque; // résultat dans la console : "Peugeot"
```

Pour extraire la liste complète des clés d'un objet : appliquer la méthode ***Object.keys(***nomObjet***)***.

*Exemple*

*.js

```
Object.keys(voiture); // résultat dans la console : Array(5) ["marque", "type", "couleur", "nombre", "En vente "]
```

### 3. Ajout de valeurs

*Exemples*

*.js

```
var voiture = {};            // Première possibilité : même principe que les tableaux
voiture["marque "] = "Peugeot";
voiture["type "]= "3008";
voiture["couleur"]= "rouge";
                    // ou
var voiture = {};
voiture.marque = "Peugeot";       // Deuxième possibilité : (en utilisant un lien : le point)
voiture.type = "3008";
```

```
voiture.couleur = "rouge";
```

### 4. Combiner des tableaux et des objets

Il est possible d'utiliser un tableau ou un objet comme valeur dans un tableau ou un objet.

*Exemples*

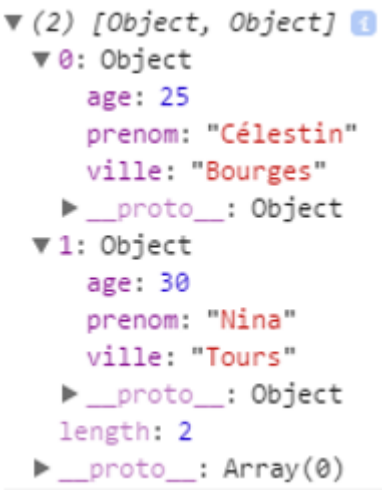
*.js

```
var voitures = [                          // Tableau d'objets
    { marque : "Peugeot", type : "3008", couleur : "rouge"},
    { marque : "Renault", type : "Captur", couleur : "bleu"},
    { marque : "Ford", type : "C-max", couleur : "gris"}
];
// Accès aux éléments du tableau
voitures[0]; // résultat dans la console : Object {marque: "Peugeot",
type: "3008", couleur: "rouge"}
voitures[1]["marque"]; // résultat dans la console : "Renault"
voitures[2].type; // résultat dans la console : "C-max"
```

### 5. Explorer des objets dans la console

Le navigateur permet de connaître le détail des objets qu'il affiche dans la console.

*Exemples*




*.js

```
var amis = [
     {prenom:"Célestin", age: 25, ville:"Bourges"},
     {prenom:"Nina", age: 30, ville:"Tours"}
];
```

```
amis; // résultat dans la console :
```

### 6. Parcourir un objet avec une boucle for in

Il n'est pas possible de parcourir un objet littéral avec une boucle **for**. Normal, puisqu'une boucle for est surtout capable d'incrémenter une variable numérique, ce qui ne nous est d'aucune utilité dans le cas d'un objet littéral puisque nous devons posséder un identifiant.
En revanche, la boucle **for in** se révèle très intéressante !

La boucle for in est l'équivalent de la boucle **foreach du PHP** : elle est très simple et **ne sert qu'à une seule chose : parcourir un objet**.
Le fonctionnement est quasiment le même que pour un tableau, excepté qu'ici il suffit de fournir une « **variable clé** » qui reçoit un identifiant (au lieu d'un index) et de spécifier l'objet à parcourir :

*Exemples*

*.js

```
var posX = {
    x1 : 12,
    x2 : 14,
    x3 : 16
};

for (var id in posX) { // On stocke l'identifiant dans « id » pour parcourir l'objet « posX »

    console.log(posX[id]);
}
// résultat dans la console : 12    14   16
```

### 7. Pour aller plus loin...

MDN web docs :