Chapitre 6: Objets

En Javascript, les objets sont des tableaux associatifs (voir chapitre 5).

L'orienté objet en javascript ne fonctionne pas sous la forme de classe/instance mais sur un modèle basé sur des prototypes.
On peut ajouter, supprimer, modifier les attributs et méthodes à la volée.

```
let personne = {};
```

> On peut utiliser comme attribut n'importe quel mot (même les mots réservés), même plusieurs mots ou encore des nombres.
> attention cependant à ne pas mettre de point.

Créer des objets

Il existe 2 façons de créer des objets en Javascript :

Débuter d'un objet vide puis de lui ajouter des éléments

```
let h = {};
h.nom = "bob";
```

Créer un objet qui possède déjà des propriétés (avec un littéral)

code Original :

```
let h = {
    "nom" : "bob",
    "age" : 42
    "toString" : function (){...}
};
```

Qui peut s'écrire sous la forme :

```
let h = {
    nom:"bob",
    age:42,
    toString(){...}
};
```

Cette forme est un ***sucre syntaxique***. Elle permets d'omettre les guillemets

> attention qu'il faut garder ceux-ci si l'on souhaite créer une clé composée de plusieurs mots

Un ***sucre syntaxique*** est une simplification du code pour la rendre plus facile à écrire et à comprendre par un Humain.

Propriétés

Ajouter

```
personne["attr"] = 12;
personne.attr = 12; // equivalent
```

Supprimer

```
delete monObj.attr;
```

Modification

Cfr Ajouter un attribut.

En effet, lorsque l'on tente de modifier un attribut inexistant on en crée en fait un nouveau.

> il est possible de modifier un objet déclaré const. Sa référence est la seule chose const

Important

Tests d'égalité entre 2 objets

un test d'égalité entre 2 objets correspond à un test d'égalité entre 2 références.

utiliser in

Pour savoir si un objet possède une propriété, on peut utiliser l'opérateur **in**

```
"prop" in monObj
```

For-in et For-of

Les boucles for-in et for-of fonctionnent aussi :

```
let msg= "";
for (let propin h)
    msg+= prop+ " -> " + h[prop] + "\n";
alert(msg);
```

Propriétés calculées

les noms de propriétés calculées

```
function infoArticle (nom, prixHtvaEU, veutDollars) {
    const tva = 0.21;
    const tauxEuDo = 1.12;
    const devise = veutDollars? "DO" : "EU";
    const prixHtva = prixHtvaEU * (veutDollars? tauxEuDo: 1);
    const prix = prixHtva * (1 + tva);
    return { nom, ["prix" + devise] : prix };
}
```

Comparaison entre valeurs primitives et Objets

	Valeur Primitive	Objets
Catégories	nb,bool,str	fct,tab,autres OBJETS
Mutabilité	immuables	mutables(à priori)
Comparaison	par valeur	par ref
passage(argument)	par valeur	par ref

Prototypes, l' "héritage" en Javascript

Un ***prototype*** est un parent commun à une série d'objets qui permets d'éviter de stocker la même information au sein de tous les objets.

Création

On peut créer un prototype de la même manière que n'importe quel autre objet.

Création d'un objet directement hérité (naissance)

```
let monEnfant = Object.create(monProto);
```

Mise en place du parent à postériori (adoption)

Il est important de noter que l'on ne peut avoir qu'un seul parent duquel on hérite

```
Object.setPrototypeOf(MonEnfant, MonProto);
```

Assign (Don)

Un objet peut se voir attribuer les propriétés d'un objet sans pour autant hériter de celui-ci. Contrairement à l'héritage prototypale, ici l'objet en question hérite de toutes les méthodes et les garde en lui. En cas de conflit, cette méthode écrase les propriétés existantes

Fonctionnement

__proto__ (ou [[proto]]) sont appelés Dunder-proto. Il s'agit d'une propriété dont la valeur est une référence vers le prototype de l'objet.

Quand on recherche une propriété et que celle ci n'est pas dans l'objet, on remonte l'arbre des prototypes pour retrouver la propriété.

Autres fonctions utiles

```
Object.getPrototypeOf(monObjet); // retourne le prototype de monObjet
monObjet1.isPrototypeOf(monObjet2); //monObjet1 est le prototype de monObjet2
```

Les Constructeurs

Un ***constructeur*** est une fonction qui permet de créer des objets à partir de ses arguments.

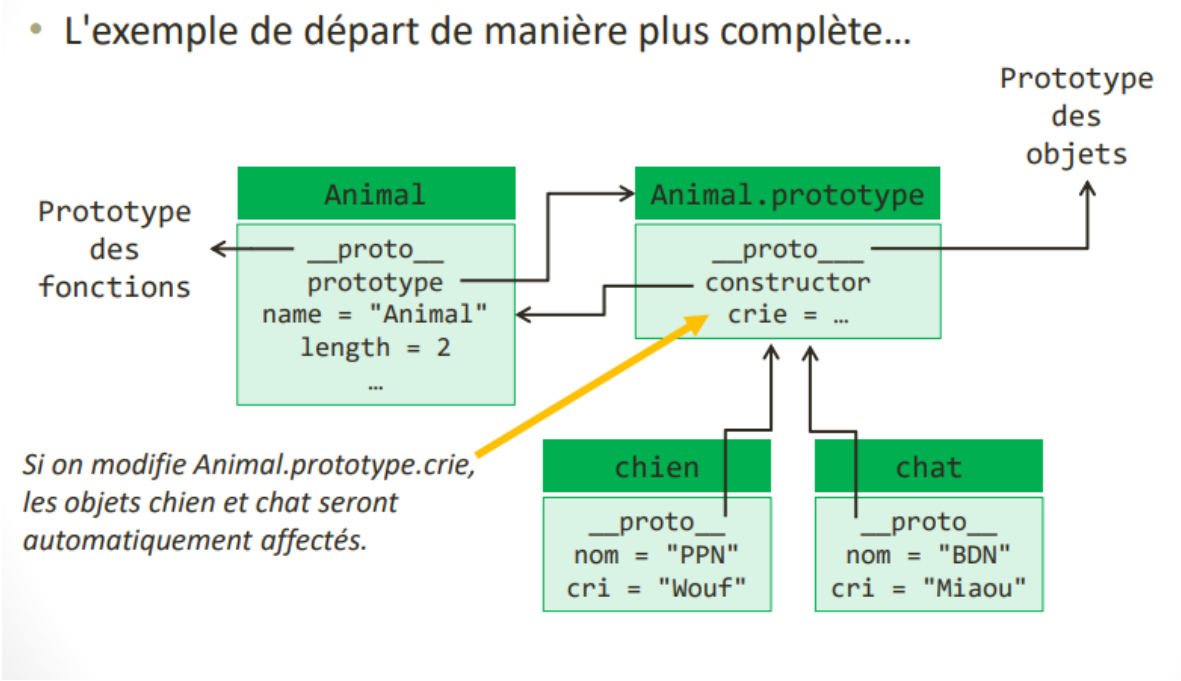
Par Convention, elle commencent avec une **Majuscule**.

```
function Animal(nom,cri){
    this.nom = nom;
    this.cri = cri;
}
// On instancie ensuite avec :
let chien = new Animal("Bouh","wouf");
```

Fonctionnement

Un Constructeur est une fonction standard qui attribue automatiquement un prototype aux objets crées.

ex : Animal.prototype



Ajouter des méthodes

Pour ajouter des méthodes il faut les ajouter dans Animal.prototype .

Exemple :

```
Animal.prototype.crie() = ... ;
```

Autres fonctions utiles

```
chat instanceOf Animal; //retourne true si chat à été crée à partir du constructeur Animal
```