

3. Extension de classe

L'extension de classe, permet de rajouter des propriétés, fonctions etc... à une classe déjà définie, n'importe où dans le code.

Considérons une classe personne :

```
// Classe Personne
var Personne = function(prenom, nom)
{
    this.prenom = prenom;
    this.nom = nom;

    this.toString = function()
    {
        return 'je suis ' + this.prenom + ' ' + this.nom;
    }
}
```

Je trouve que dans cette classe il manque la propriété age. Pour la rajouter je peux modifier le code de ma classe mais aussi rajouter ma propriété/méthode à la suite de la classe. Ceci est possible en se servant du prototype de la classe (je reviendrais plus tard sur la notion importante de prototype).

Le code devient donc :

```
// Classe Personne
var Personne = function(prenom, nom)
{
    this.prenom = prenom;
    this.nom = nom;

    this.toString = function()
    {
        return 'je suis ' + this.prenom + ' ' + this.nom;
    }
}

// Ajout d'une propriété age ayant pour valeur 0 par défaut.
Personne.prototype.age = 0;

// Instanciation d'un nouvel objet
var _Cyril = new Personne('Cyril', 'Durand');
_Cyril.age = 19;
alert(_Cyril.age);
```

Ce que j'ai fait avec une propriété, on pourrait tout aussi bien le faire pour des fonctions, le raisonnement serait le même.

Attention : vous ne pouvez pas surcharger une fonction à l'extérieur de la classe. En effet, imaginons que je veuille changer la méthode toString de ma classe, on aurait pu imaginer faire quelque chose comme ca :

```
// Classe Personne
var Personne = function(prenom, nom)
{
    this.prenom = prenom;
    this.nom = nom;

    this.toString = function()
    {
        return 'je suis ' + this.prenom + ' ' + this.nom;
    }
}

Personne.prototype.age = 0;
Personne.prototype.toString = function()
{
    return 'je suis ' + this.prenom + ' ' + this.nom + ' et j\'ai ' +
this.age + ' ans';
}

// Instanciation d'un nouvel objet
var _Cyril = new Personne('Cyril', 'Durand');
_Cyril.age = 19;
```

Cela ne fonctionnera pas ! En fait, le problème se situe au niveau de l'instanciation, quand vous faites votre `new Personne`, une nouvelle instance de l'objet `Personne` va être créée. Pour l'instant, l'ordinateur n'est jamais passé par le `new`, il n'est donc jamais passé par l'intérieur de la classe :

```
    this.prenom = prenom;
    this.nom = nom;

    this.toString = function()
    {
        return 'je suis ' + this.prenom + ' ' + this.nom;
    }
}
```

Par contre, il est déjà passé par les lignes :

```
Personne.prototype.age = 0;
Personne.prototype.toString = function()
{
    return 'je suis ' + this.prenom + ' ' + this.nom + ' et j\'ai ' +
this.age + ' ans';
}
```

La classe `Personne`, possède donc déjà, une propriété `âge` et une fonction `toString`, une fois l'initialisation, il va créer les propriétés `prenom` et `age`, mais aussi redéfinir la méthode `toString`.

Pour éviter ce problème, certaines personnes mettent le minimum de chose dans le constructeur de la classe, et mettent toutes les fonctions publiques via prototype. C'est une solution pour créer des classes, mais il existe beaucoup de façons de faire, à vous de trouver celle qui vous convient.

On peut étendre les classes sur tout les types d'objets, y compris sur les objets de bases fournis par JavaScript. C'est d'ailleurs ce qu'a fait Aurélien sur l'objet string dans sa librairie [String.net](#).