

6. Événements

Nous pouvons également simuler des événements en JavaScript. Mais voyons plutôt un exemple.

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

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

    // Déclaration de l'événement
    this.Event = function(){}

    // fonction permettant de déclencher l'événement
    this.callEvent = function()
    {
        this.Event();
    }
}
```

Dans ma classe personne, j'ai déclaré un Eventent "Event" et la fonction callEvent permet d'appeler cet événement, dans la vrai vie c'est bien sur plus compliqué que ça.

Ensuite on instancie notre objet :

```
var obj = new Personne('Cyril', 'DURAND');
```

Puis l'on s'abonne à l'événement

```
obj.Event = function()
{
    alert('l\'événement à été déclenché');
}
```

Certains l'ont peut être remarqué, mais on ne s'abonne pas vraiment à l'événement, en fait on surcharge la méthode 'Event'. Là est toute l'astuce des événements en JavaScript ☺

Ensuite on appelle la méthode permettant de déclencher l'événement :

```
obj.callEvent();
```

Ce qui donne :

```
// instaciation de l'objet
var obj = new Personne('Cyril', 'DURAND');

// Abonnement à l'événement
obj.Event = function() { alert('l\'événement à été déclenché'); }
```

```
obj.callEvent();
```

Puisque un événement n'est qu'une surcharge de méthode, on pourrait penser à optimiser la déclaration de celui-ci.

Par exemple, en pensant à détecter si la méthode (l'événement) existe ou non.

```
this.callEvent = function()
{
    if (this.Event)
        this.Event();
}
```

De ce fait on est plus obligé de déclarer l'événement. On peut encore faire mieux, comme par exemple créer une fonction qui vérifierais la présence de la méthode, et la déclencherai.

```
this.fireEvent = function(name)
{
    if (this[name] && typeof(this[name]) == 'function')
    {
        this[name]();
    }
}
```

Cette fonction utilise `this[name]` en fait `this` fonctionne comme un super tableau, on peut donc appeler les méthodes / propriétés en passant leur noms entre crochet ou même en donnant l'index de la méthode si on le connaît.

On peut maintenant être encore plus fainéant et faire :

```
this.callEvent = function()
{
    this.fireEvent('Event');
}
```

Puisque l'événement 'Event' est réduit à un string, on peut même écrire ceci :

```
this.callEvent = function(name)
{
    this.fireEvent(name);
}
```

Et dans l'instance de l'objet on peut écrire des choses de ce genre :

```
// instantiation de l'objet
var obj = new Personne('Cyril', 'DURAND');

obj.Event3 = function(){alert('evenement 3 appelé');}
obj.Event8 = function(){alert('evenement 8 appelé');}

for (var i = 0; i < 10; i++)
    obj.callEvent('Event' + i);
```

Vous en voyez peut être pas l'utilité tout de suite, mais je vous assure que ce genre d'astuce peut très vite vous être indispensable 😊

Notre fonction de déclenchement d'événement est presque finit, mais il y a un problème : on ne peut pas passer d'arguments à nos événements.

Voici comment je procède :

```
// fonction permettant de déclencher les evenements
this.fireEvent = function(name)
{
    // On vérifie que l'evenement existe
    if (this[name] && typeof(this[name] == 'function'))
    {
        // On recupere les différents paramètre
        var params = new Array();
        for (var i = 1 ; i < arguments.length; i++)
            params.push(arguments[i]);

        // On appelle la fonction avec ces différents paramètres
        this[name].apply(this, params);
    }
}
```

Ceci à l'air compliqué mais c'est en fait tout simple ☺ l'objet arguments permet de récupérer les différents paramètres de la fonction, ca ressemble à un Array mais ce n'en est pas un ! Voir ici pour plus de détails : [explication msdn de l'objet arguments](#)

Après avoir récupéré un tableau des différents arguments, on appelle la fonction apply de la méthode/événement. J'ai au début eu un peu de mal à comprendre les fonctions apply et call qui est sa petite sœur, ce qu'il faut savoir ce que ce sont des fonctions applicable sur les fonctions, qui permettent d'appeler les fonctions en spécifiant qui les appelle ainsi que les paramètres.

En gros:

```
this[name].apply(this, params);
```

Va appeler la fonction `this[name]` avec les bons paramètres. Je reviendrais plus tard sur cette fonction très importante, en attendant vous pouvez consulter ce lien : [document mozilla sur la fonction apply](#)

On peut désormais appeler notre evenement de la sorte :

```
this.callEvent = function(name)
{
    this.fireEvent(name, 'param1', 'param2');
}
```

Et s'abonner à celui-ci comme cela :

```
obj.Event2 = function(p1, p2){alert(p1 + ' - ' + p2);}
```

On verra plus tard l'héritage, ce qui nous permettra de mettre la fonction `fireEvent` dans une classe de base.

Voici le code au final :

```
// Classe Personne
var Personne = function(prenom, nom)
```

```
{

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

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

    // fonction permettant d'appeller la fonction/Evenement
    this.fireEvent = function(name)
    {
        // On vérifie que l'evenement existe
        if (this[name] && typeof(this[name] == 'function'))
        {
            // On recupere les différents paramètre
            var params = new Array();
            for (var i = 1 ; i < arguments.length; i++)
                params.push(arguments[i]);

            // On appelle la fonction avec ces différents paramètres
            this[name].apply(this, params);
        }
    }

    // fonction permettant de déclencher l'evenement
    this.callEvent = function(name)
    {
        this.fireEvent(name, this.nom, this.prenom);
    }
}

// instaciation de l'objet
var obj = new Personne('Cyril', 'DURAND');

// Abonnements aux evenements
obj.Event8 = function(){alert('evenement 8 appelé');}
obj.Event2 = function(p1, p2){alert(p1 + ' - ' + p2);}
obj.Event3 = function(){alert('evenement 3 appelé');}

// appels de la fonction déclenchant les events
for (var i = 0; i < 10; i++)
    obj.callEvent('Event' + i);
```