Introduction à la bibliothèque JavaScript jQuery

par Dave Lizotte (PcKULT.NET) Didier Mouronval (Articles)

Date de publication : 04/11/2008

Dernière mise à jour : 22/12/2010

Nous allons, dans cet article, effectuer une introduction à la bibliothèque JavaScript : jQuery. Il va sans dire que pour comprendre tant soit peu l'article qui suit, il vous faut un minimum de connaissances JavaScript et du modèle objet des documents (DOM). Nous verrons donc la base en vous donnant les détails nécessaires à votre apprentissage afin que vous soyiez en mesure de progresser par vous-même facilement.

Pour démarrer, il vous faudra la bibliothèque jQuery. La version la plus récente se trouve ici : **http://jquery.com/src/**

Hello world !

Comme presque tout ce que nous faisons avec jQuery, ce dernier lit ou manipule le modèle objet du document (DOM). Il faut donc être sûr que nous n'ajoutons des événements ou autres que lorsque le DOM (le document) est complètement chargé. Pour cela, nous enregistrons un événement « document chargé » ou « $(document).ready(function(){}) ».

```
$(document).ready(function()
{
    // Traitement à effectuer lorsque le DOM est prêt
});
```

Mettre un message d'alerte dans cette fonction n'a pas beaucoup de sens, puisqu'un message n'a pas besoin que le DOM soit chargé. Alors poussons l'exemple un peu plus loin et essayons d'afficher un message lorsqu'un lien est cliqué.

```
$(document).ready(function()
{
    $("a").click(   function()
                    {
                        alert("Hello world!");
                    }
                );
});
```

Cela devrait afficher le message si vous cliquez sur un lien de votre page.

Un coup d'oeil sur notre script :

- **$("a")** est un sélecteur jQuery, ici, il sélectionne tous les éléments de type lien *<a>*.
- **$** est un alias pour la classe jQuery : *$()* fabrique un nouvel objet jQuery.
- La fonction **click()** appelée dans la suite est une méthode de l'objet jQuery. Elle associe un événement à tous les éléments sélectionnés et exécute la fonction fournie quand l'événement se déclenche.

Le code suivant génère le même message :

```
<a href="#" onclick="alert('Hello world');">Lien</a>
```

La différence est évidente :

- Nous n'avons pas besoin d'écrire un événement *onclick* pour chaque élément ;
- Nous obtenons une séparation claire entre la structure (HTML) et le comportement (JavaScript) de la même manière que votre feuille de style CSS est dissociée de votre code HTML.

Présentation des sélecteurs

Les sélecteurs

La première chose à apprendre de jQuery afin d'être en mesure de séparer vos codes JavaScript et (X)HTML sont les sélecteurs. Les sélecteurs permettent de sélectionner un ou plusieurs éléments du DOM, ces derniers pouvant être manipulés comme tout autre objet. Si vous êtes familier avec CSS, vous allez saisir rapidement les sélecteurs. Ceux-ci sont en réalité la même chose et ils utilisent presque la même syntaxe. jQuery permet la sélection des éléments grâce à une fonction spéciale nommée « **$** ».

Sélecteurs

```
$(document) // Extension de l'objet document à la classe jQuery.

$('*') // Sélectionne tous les éléments.

$('#monDiv') // Sélectionne l'élément ayant l'ID "monDiv".

$('p.first') // Sélectionne les éléments <p> ayant la classe "first".

$('p[title]') //Sélectionne les éléments <p> ayant un attribut "title".

$('p[title="Bonjour"]') // Sélectionne les éléments <p> dont l'attribut title est "Bonjour".

$('p[title!="Bonjour"]') // Sélectionne les éléments <p> dont l'attribut title n'est pas "Bonjour".

$('p[title^="H"]') // Sélectionne les éléments dont l'attribut title commence par "H".

$('p[title$="H"]') // Sélectionne les éléments dont l'attribut title fini par "H".

$('p[title*="H"]') // Sélectionne les éléments dont l'attribut title contient "H".

$('ul, ol, dl') // Sélectionne les éléments <ul>, <ol> et <dl>

$('div .desc') // Sélectionne les éléments ayant la classe "desc" descendants (au sens CSS) d'éléments <div>.

$('div > .enfant') // Sélectionne les éléments ayant la classe "enfant" enfants d'éléments <div>.

$('label + input') // Sélectionne les éléments <input /
> dont l'élément précédent (dans le DOM) est <label>.

$('#debut ~ div') // Sélectionne les éléments <div> frères se situant après l'élément dont l'id est "debut".
```

La sélection par filtre

jQuery offre une très large possibilité de sélection d'éléments en fonction de filtres sur des collections d'éléments. Le fonctionnement de la sélection par filtre est simple : on sélectionne d'abord un ensemble d'éléments (par défaut, tous) puis on affine cette sélection à partir de certains critères.

Pour bien comprendre les notions de filtre, il est important d'avoir des connaissances minimum sur la structure du DOM (Document Object Model). Pour en savoir plus ***FAQ sur le DOM****.*

Filtres

```
// La notation [a|b] signifie que l'on applique le filtre a ou le filtre b.

$('div:first') // Sélectionne le premier élément <div>.

$('div:last') // Sélectionne le dernier élément <div>.

$('div:not(.ok)') // Sélectionne les <div> n'ayant pas la classe "ok".

$('div:[even|odd]') // Sélectionne les éléments <div> de rang [pair|impair] (le premier rang est 0).

$('div:[eq|lt|gt](n)')// Sélectionne le ou les éléments <div> de rang [égal|inférieur|supérieur] à n.

$(':header') // Sélectionne les éléments <hn>.

$(':animated') // Sélectionne les éléments actuellement animés.

$("div:contains('dvp')") // Sélectionne les éléments <div> contenant le texte "dvp" (sensible à la casse !)

$('div:empty') // Sélectionne les éléments <div> vides.

$('div:has(p)') // Sélectionne les éléments <div> ayant un descendant <p>.
```

Filtres

```
$('div:parent') // Sélectionne les éléments <div> ayant des enfants (y compris les noeuds texte).

$('div:nth-child([n|even|odd|equation])') // Les enfants de <div> [de rang n|pairs|impairs|
résultat de].

$('div:[first-child|last-child]') // Les éléments [premier|dernier] enfants d'un élément <div>.

$('div:only-child') // Les éléments qui sont les seuls enfants d'un élément <div>.
```

Les éléments de formulaire

Pour terminer, jQuery possède quelques filtres spécifiques à la sélection d'éléments de formulaires en fonction de leur nature ou de leur état.

Eléments de formulaires

```
$(':input') // Tous les éléments <input />, <textarea>, <select> et <button>.

$(':[text|password|radio|checkbox|submit|image|reset|button|file|hidden]') // Les <input /
> du type choisi.

$(':[enabled|disabled|checked|selected]') // Les <input /> possédant l'attibut indiqué.
```

Utilisation des sélecteurs et des événements

JQuery nous propose 2 approches afin de sélectionner des éléments.
La première utilise une combinaison de sélecteurs CSS et XPath passés comme chaîne de caractères au constructeur jQuery (comme par exemple *$("div > ul a")*).
La seconde méthode, quant à elle, se sert de différentes fonctions de l'objet jQuery. Les deux approches peuvent être combinées.

Pour essayer certains de ces sélecteurs, sélectionnons et modifions, par exemple, une liste ordonnée *<ol>* insérée dans une page.

Tout d'abord, sélectionnons la liste elle-même. Cette liste a comme id « listeOrdonnee ». En JavaScript classique, vous pourriez la sélectionner avec *document.getElementById("listeOrdonnee")*. Avec jQuery, cela devient :

```
$(document).ready(  function()
                    {
                        $("#listeOrdonnee").addClass("rouge");
                    }
);
```

Admettons que notre feuille de style propose une classe « rouge » qui ajoute simplement un fond rouge. Donc, si vous ne rechargez pas la page dans votre navigateur, vous devriez voir que la première liste ordonnée obtient un fond rouge. Travaillons maintenant avec les enfants de cette liste.

```
$(document).ready(  function()
                    {
                        $("#listeOrdonnee > li").addClass("blue");
                    }
);
```

Ce dernier sélectionne tous les enfants *<li>* de l'élément avec l'id « listeOrdonnee » et leur ajoute la classe « bleu ». Quelque chose d'un peu plus complexe maintenant : nous voulons ajouter et enlever la classe quand l'utilisateur survole l'élément *<li>*, mais seulement pour le dernier élément de la liste.

```
$(document).ready(  function()
                    {
                        $("#listeOrdonnee li:last").hover(  function()
                                                            {
                                                                $(this).addClass("Rouge");
                                                            },
                                                            function()
                                                            {
                                                                $(this).removeClass("Rouge");
                                                            }
                                                        );
                    }
                );
```

De nombreux autres sélecteurs à la syntaxe similaire à CSS ou XPath existent. Des exemples supplémentaires et une liste de toutes les expressions disponibles se trouvent **ici**, sur le site web de jQuery.
Pour chaque événement existant, tels que *onclick*, *onchange*, *onsubmit*, *etc.*, il existe un équivalent jQuery. D'autres événements comme *ready* et *hover* sont fournis par commodité pour certaines tâches afin de vous rendre la vie plus facile. Vous pouvez trouver une liste complète des événements sur **Visual jQuery** dans la section Events du site web. Grâce à ces sélecteurs et ces événements, vous pouvez déjà faire plein de choses, mais il y a mieux soyez en sûrs.

```
$(document).ready(  function()
                    {
                        $("#listeOrdonnee").find("li").each(    function(i)
                                                                {

 $(this).html( $(this).html() + " WOW! " + i );

                                                                }
                                                            );
                    }
);
```

- **find()** vous permet des recherches supplémentaires parmi les descendants de l'élément déjà sélectionné. Ainsi, *$("#listeOrdonnee).find("li")* est tout à fait similaire à *$("#listeOrdonnee li")*.
- **each()** parcours chacun des éléments et permet des traitements supplémentaires. La plupart des méthodes, telles que *addClass()*, utilisent *each()* elles-mêmes. Dans cet exemple, *html()* est utilisé pour récupérer le texte de chaque élément *<li>*, lui ajouter du texte et le réaffecter comme texte de ces éléments.

Une autre tâche que vous avez fréquemment à effectuer est d'appeler des méthodes sur des éléments DOM non couverts par jQuery. Imaginez un formulaire que vous voudriez réinitialiser lorsqu'il a été traité correctement par soumission AJAX.

```
$(document).ready(  function()
                    {
                        // Réinitialiser un formulaire unique
                        $("#reset").click(  function()
                                            {
                                                $("#form")[0].reset();
                                            }
                                        );
                    }
);
```

Le code précédent sélectionne l'élément ayant l'id « form » et appelle la méthode *reset()* du premier élément sélectionné. Si vous aviez plusieurs formulaires, vous pourriez aussi faire ceci :

```
$(document).ready(  function()
                    {
                        // Réinitialiser plusieurs formulaires
                        $("#reset").click(  function()
                                            {
                                                $("form").each( function()
                                                                {
                                                                    this.reset();
                                                                }
                                                              );
                                            }
                                          );
                    }
);
```

Cela sélectionne tous les formulaires au sein de votre document, les parcours et appelle la méthode *reset()* pour chacun d'entre eux.

Un autre besoin que vous pouvez rencontrer est de ne pas sélectionner certains éléments. jQuery fournit *filter()* et *not()* pour cela. Alors que *filter()* réduit la sélection aux éléments qui correspondent à l'expression du filtre, *not()* fait le contraire et enlève tous les éléments correspondant à l'expression. Imaginez une liste non ordonnée où vous voudriez sélectionner tous les éléments *<li>* qui n'ont pas d'enfant *<ul>*.

```
$(document).ready(  function()
                    {
                        $("li").not("[ul]").css("border", "1px solid black");
                    }
);
```

Cela sélectionne tous les éléments *<li>* puis retire de cette sélection tous ceux qui ont un *<ul>* comme enfant. Ainsi, tous les éléments *<li>* récupèrent une bordure, hormis ceux qui ont un enfant *<ul>*. La syntaxe de *[expression]* est issue de XPath et peut être utilisée pour filtrer par éléments enfants et par attributs.

D'une autre part, il y a des situations où l'on a besoin de sélectionner les éléments précédents ou suivants, dénommés « siblings » (frères et soeurs : enfants du même parent). Imaginez une page de FAQ, où toutes les réponses sont cachées au départ et affichées lorsque la question est cliquée.

```
$(document).ready(  function()
                    {
                        $('#faq').find('dd').hide().end().find('dt').click( function()
                                                                            {
                                                                                var answer =

 $(this).next();

                                                                                if

 (answer.is(':visible'))

                                                                                {
                                                                                    answer.slideUp();
                                                                                }
                                                                                else
                                                                                {
                                                                                    answer.slideDown();
                                                                                }
                                                                            }
                                                                          );
                    }
);
```

Nous utilisons des enchaînements pour réduire la taille du code et obtenir de meilleures performances, puisque *#faq* ne sera sélectionné qu'une seule fois. En utilisant *end()*, le premier *find()* est annulé, ainsi nous pouvons nous mettre à chercher avec le prochain *find()* pour notre élément *#faq* plutôt que les enfants <*dd*>. À l'intérieur de l'événement « click », nous utilisons *$(this).next()* pour trouver le « sibling » suivant à partir du <*dt*> courant. Cela nous permet de sélectionner rapidement la réponse suivant la question cliquée.

En conclusion

La force de jQuery réside donc manifestement dans la richesse de sa fonction **$**, d'autant plus qu'en dehors de la richesse de ces sélecteurs, vous pouvez bien sûr les combiner entre eux pour atteindre n'importe quel élément du DOM très facilement.

Tous les sélecteurs CSS vont fonctionner avec jQuery et ce dernier permet encore plus de possibilités que CSS.

Remerciements

Tous mes remerciements à **romaintaz** pour sa relecture.