



FORMATION À JQUERY

Thomas Morin

Novembre 2011



La bibliothèque JavaScript libre jQuery

Utilise JavaScript de façon facultative et non-intrusive

Dans le header, on appelle la librairie jQuery, des fonctions personnalisées, et un " lanceur ".

Encourage la séparation de la forme et du contenu

Les styles (fichiers .css)

Les scripts (fichiers .js)

Le contenu html (fichier .html avec un contenu sémantique)

Est complétée par des plugins

jqGrid, Autocomplete, UI jQuery...

Interaction entre HTML et JavaScript

Parcours et modification de la DOM (Document Object Model)

Manipulations des styles CSS (Cascading Style Sheets)

Gestion des événements et animations

Echange de données avec Ajax

Tout cela repose sur une fonction unique : `jQuery()`

Les objets de jQuery

Le document, la fenêtre

```
$document = jQuery(document)
```

```
$fenetre = jQuery(window)
```

Un nœud ou un groupe de nœuds de la DOM

```
$objet = jQuery(parametres)
```

Alias " \$ " et encapsulation pour éviter les conflits

```
(function ($) {
```

```
    $objet = $(parametres)
```

```
    // ⇔ $objet = jQuery(parametres)
```

```
})(jQuery);
```

Les fonctions de jQuery

Méthodes de jQuery : des fonctions comme les autres

```
// chaine = jQuery.trim(chaine) ;  
chaine = $.trim(chaine) ;
```

Méthodes des objets jQuery

```
// $objet = jQuery(sélecteur) ;  
$objet = $(sélecteur) ;  
$objet.methode(paramètres);
```

Des fonctions anonymes en paramètre de fonctions

Exemple : commencer au chargement de la page

```
// Encapsulation
(function ($) {
    // Méthode .ready( ) pour le document
    $(document).ready(function(){
        // Fonction $.trim de jQuery
        var text = $.trim(" Mon text ");
        // Fonction alert native de javascript
        alert( text ) ;
    });
})(jQuery);
```

JSON : La notation d'objet en JavaScript

Une écriture synthétique

```
// Déclaration
```

```
objet = {  
    chaine : "du texte",  
    nombre : 4,  
    vecteur : ["a","b","c"],  
    fonction : function(a) { alert(a) ; }  
}
```

```
// Appel
```

```
chaine = objet.chaine ;  
objet.fonction(" appel de la fonction ") ;
```

Le même chemin qu'en CSS

Syntaxe :

```
$capture = $(" sélecteur ")
```

Élément, identifiant et classe de la CSS2

```
$h1 = $(" h1 ") ;
```

```
$top = $(" #top ") ;
```

Attributs de la CSS3

```
$lienSommaire = $(" a[href='./sommaire.htm'] ") ;
```

Opérateurs

```
$suivant = $("h1+*, h2+*") ;
```

Pseudo-éléments

```
$selected = $(" option:selected ") ;
```


Modifier les styles avec la méthode `.css()`

Paires de clés et de valeurs

```
$node.css(" color ", " red " );  
$node.css(" text-decoration ", " italic " );
```

Utilisation du format Json

```
$node.css({  
    color : " red ",  
    textDecoratoin : " italic "  
})
```

Information sur le style

```
color = $node.css(" color ");
```

TRAVAUX PRATIQUES 1

Attributs avec la méthode `.attr()`

Obtenir les attributs d'un nœud

```
href = $lien.attr("href " );  
value = $input.val()  
// ⇔ $input.attr(" value ")
```

Modifier les attributs

```
$a.attr(" href ", " ../sommaire.htm ")
```

Format Json

```
$a.attr({  
    title:" Sommaire ",  
    href:" ../sommaire.htm "  
})
```

Modifier la classe d'un élément

Ajouter ou retirer la classe d'un élément

```
$node.addClass(" nouvelle_classe ");  
$node.removeClass(" ancienne_classe ");  
$node.toggleClass(" actif ").toggleClass(" inactif ");
```

Information sur la classe

```
if($objet.hasClass(" ma_classe ")) {  
    // ... Instructions ...  
}
```

Manipuler les classes ou injecter du style ?

Plusieurs nœuds peuvent composer l'objet

Sélecteur

```
$enfants = $(" p > * " ) ;
```

Dans la DOM

```
<p>
```

```
  <strong>Attention </strong>, <em>Trois</em>  
  éléments sont <strong>sélectionnés</strong> !
```

```
</p>
```

Récupérer le nombre de nœuds

```
var combien = $enfants.length
```

Parcourir les composants d'un objet jQuery

Boucle sur les nœuds : `.each()`

```
$enfants.each(function(i){  
    var $fils = $(this) ; // this est le nœud javascript  
    $fils.css(" color " , " red " ) ;  
})
```

Occurrence d'un nœud : `.eq()`

```
$fils2 = $enfants.eq(1) ; // Les index commencent à 0
```

Retrouver le nœud javascript : `.get()`

```
tag2 = $enfants.get(1).tagName // Balise du 2nd fils
```

Composants texte et html

Insérer du contenu html

```
$a.html(" <strong>Sommaire</strong> ")
```

Insérer du contenu texte

```
$a.text(" Revenir au sommaire ")
```

Récupérer les contenus texte et html

```
html = $a.html() ;
```

```
texte = $a.text() ;
```

TRAVAUX PRATIQUES 2

Représenter la DOM comme un arbre

```
< Ancêtre >  
  < Parent >  
    <Frère>...</Frère >  
    < Nœud >  
      < Enfant >  
        < Descendant />  
      </ Enfant >  
    </ Nœud >  
    <Frère>...</Frère >  
  </ Parent >  
</ Ancêtre >
```

Des méthodes pour parcourir

Syntaxe

```
$enfants = $parent.children(" * ") ;
```

Descendre

```
find(), children()
```

Remonter

```
parents(), parent(), closest()
```

En latéral

```
next(), prev(), siblings()
```

Manipuler

```
filter(), not()
```

Des méthode pour modifier

Syntaxe

`$parent.append($nouvelEnfant) ;`

Descendre

`append(), appendTo(), prepend(), prependTo(), html()`

Remonter

`wrap(), unwrap(), wrapInner()`

En latéral

`after(), insertAfter(), before(), insertBefore()`

Manipuler

`remove(), empty(), detach(), replaceWith()`

Créer un élément avec jQuery() ou clone()

Créer un élément avec une chaîne de caractère

```
$a = $("<a href='../sommaire.htm'> Sommaire </a>");
```

Création avec le JavaScript natif

```
$a = $(document.createElement(" a "));
```

```
$a.text(" Sommaire ").attr(" href ", " ../sommaire.htm ");
```

Dupliquer un élément

```
$b = $a.clone();
```

Trier un tableau selon l'identifiant des lignes

```
$tbody = $("#tableau tbody") ;  
    // Capture des lignes du tableau  
    .children("tr")  
    // Tri selon l'identifiant  
    .sort(function(ligneA,ligneB) {  
        var valeurA = $(ligneA).attr(" id ") ;  
        var valeurB = $(ligneB) .attr(" id ") ;  
        return valeurA > valeurB ? 1 : -1 ;  
    })  
    // Les lignes sont ajoutée à $tbody dans l'ordre de tri  
    .appendTo($tbody) ;
```

TRAVAUX PRATIQUES 3

Fonction déclenchée par l'évènement

Exemple avec la méthode " change "

```
$("# select ").change(  
    // Fonction lancée quand la liste de sélection change :  
    function() {  
        var $select = $(this) ;  
        var $option = $select.find("option:selected") ;  
        var value = $option.eq(0).val() ;  
        alert(value ) ;  
    }  
)
```

Quelques évènements classiques

Chargement de la DOM

`$(document).ready()`

Modification de la taille de la fenêtre

`$(window).resize()`

Cliquer, survoler avec la souris

`click()`, `hover()`, `mousemove()`, `toggle()`

Formulaires

`submit()`, `change()`, `focus()`, `keyup()`

Plusieurs fonctions comme paramètres

Survoler : événements " entrer " et " sortir "

```
$( ".pseudo-lien" ).hover(  
    // La souris arrive sur l'élément  
    function(){  
        $(this).addClass(" pseudo-hover " );  
    },  
    // La souris quitte l'élément  
    function(){  
        $(this).removeClass(" pseudo-hover " );  
    }  
)
```

Passer l'évènement en paramètre

Annuler l'action par défaut

```
$a.click(function(event){  
    event.preventDefault ; // annule l'action par défaut  
})
```

Position du curseur

```
$body.mousemove(function(event) {  
    var curseurX = event.pageX ;  
    var curseurY = event.pageY ;  
    $input.val(curseurX + ", " + curseurY ) ;  
})
```

Valeur de retour

L'action par défaut est annulée lorsque la fonction renvoie " faux "

```
$("#form").submit(function(){  
    // Pop-up (javascript natif) :  
    var bool = confirm("Soumettre le formulaire ?") ;  
    // Si bool vaut "false", le formulaire n'est pas soumis  
    return bool ;  
});
```

TRAVAUX PRATIQUES 4

Syntaxe des méthodes pour les animations

Exemple avec la méthode fadeOut

```
$("#h1").fadeOut(  
    // Paramètre (durée en milliseconde)  
    2000,  
    // Fonction déclenchée  
    // quand la transition est terminée :  
    function(){  
        alert("Le titre a disparu !") ;  
    }  
);
```

Quelques effets classiques

Jouer sur la transparence

`fadeIn(durée)`, `fadeOut(durée)`, `fadeTo(durée,opacité)`,

Jouer sur la hauteur

`slideUp(durée)`, `slideDown(durée)`,

Montrer, cacher

`Hide(durée)`, `Show(durée)`

Toujours la possibilité d'ajouter une fonction à exécuter à la fin de l'animation (" callback ").

Effet personnalisé avec `.animate()`

Transition d'un objet vers le style que vous indiquez :

```
$objet.animate(  
    // Style au format Json :  
    { opacity : 0.5, width:"500px" },  
    // Durée de l'animation :  
    2000,  
    function() {  
        // ...instructions quand c'est fini...  
    }  
)
```

Temporiser les effets

Arrêter l'animation en cours :

```
.stop()
```

Désactiver toutes les animations :

```
$.fx.off = true
```

Temporiser avec `.delay()`

```
$objet.hide(2000).delay(1000).show(2000) ;
```


Exemple

```
$objet.css('background','#c5d6e7')
// Première animation
.animate( { left: '+=250' }, 1000 )
// Pause d'une seconde
.delay(1000)
// Seconde animation
.animate( { left: '-=250' }, 1000 , function() {
    // Quand la seconde animation est finie
    $(this).css('background','#e68984')
})
```

TRAVAUX PRATIQUES 5

Echanger de données avec le serveur

Interroger le serveur sans recharger la page

Ajax pour " Asynchronous JavaScript and XML "

Déclencher quand on veut

la page est chargée

un bouton est cliqué

une option est sélectionnée ...

Principale utilisation (pour cette formation) :

charger un document

traiter le document chargé

modifier la DOM

Méthode GET pour charger un document

```
$.get(  
    // URL :  
    " auteurs.xml.asp ",  
    // Paramètres :  
    {identifiant: ident },  
    // Fonction exécutée quand le document est chargé :  
    function(data) {  
        // data est le contenu textuel de la page  
        $(data).each(function(){...}) ;  
    }  
);
```

Méthode POST pour charger un document

```
$.post(  
    // URL :  
    " authentication.asp ",  
    // Paramètres :  
    {login: log, password: pwd },  
    // Fonction exécutée quand le document est chargé :  
    function(data) {  
        if(data!== " ok ")  
            alert(" login ou mot de passe incorrect ")  
    }  
);
```

Méthode LOAD pour charger un document

Utilisable en local

Injecte le document chargé dans un conteneur

```
$conteneur.load(  
    "fichier.html",  
    function() {  
        var data = $objet.html() ;  
        // ...instructions...  
    }  
);
```

TRAVAUX PRATIQUES 6

Extensions de jQuery

Fonctions et propriétés de jQuery

```
$.mesPropriétés = {texte:"a",nombre:22}  
$.maFonction = function(paramètres) {  
    // ...Instructions...  
}
```

Exemples : \$.fx.off, \$.trim()

Méthodes des objets jQuery

```
$.fn.maMéthode() = function(paramètres) {  
    // ...Instructions...  
}
```

Exemple : \$.fn.prepend()

Créer les extensions avec `$.extend()`

Fonctions et propriétés de jQuery :

```
$.extend({  
    maFonction : function(parametres) {...},  
    mesPropriétés : {...}  
});
```

Méthodes des objets jQuery :

```
$.fn.extend({  
    maMéthode1 : function(parametres) {...},  
    maMéthode2 : ...  
});
```

Bien construire une méthode pour pouvoir chaîner les instructions

Déclaration de la fonction

```
$.fn.maMéthode() = function() {  
    // ...initialisation...  
    return this.each(function(i){  
        var $this = $(this) ;  
        //...Instructions...  
    })  
}
```

Utilisation chaînée

```
$objet.find(" h2 ").maMéthode().autreMethode() ;  
// non chaînés : .attr(" href "), .css(" color ") => string
```

Paramètres par défaut `$.extend(conf,options)`

Les paramètres passés à la fonction écrasent les paramètres par défaut

```
$.fn.maMéthode() = function(options) {  
    configuration = $.extend(  
        {  
            vitesse : 2000 ,  
            callback : function(a) { ... }  
        },  
        options)  
    // ... Instructions...  
}
```

Concevoir sa fonction comme un plugin

Une fonction principale :

```
$.fn.monPlugin = function(conf) {...}
```

Des paramètres par défaut :

```
$.monPlugin.default = {...}
```

Des fonctions utilisées par le plugin :

```
$.monPlugin.outil = function() {...}
```

Des feuilles de style associées

```
<link rel="stylesheet" type="text/css" href="mon.plugin.css"/>
```

TRAVAUX PRATIQUES 7

Utiliser des plugins

Télécharger et installer les fichiers sources

Appeler les scripts et les styles dans le header

Utiliser les plugins comme des méthodes natives de jQuery

`$("# a ").monPlugin(mesParamètres)`

Enjeux

Bien choisir les paramètres

Parfois, adapter la CSS

Exemples de plugins (1)

jCarousel :

Faire défiler des images

<http://sorgalla.com/projects/jcarousel/>

Lightbox :

Afficher des images

<http://leandrovieira.com/projects/jquery/lightbox/>

Jwysiwyg :

Editeur de contenu html

<http://akzhan.github.com/jwysiwyg/help/examples/>

Exemples de plugins (2)

Autocomplete :

suggestions de saisies au cours de la frappe

<http://docs.jquery.com/Plugins/autocomplete>

jqGrid :

boîtes à outils pour les tableaux

<http://www.trirand.net/demophp.aspx>

Jquery UI :

boîtes à outils et support pour d'autres plugins :

glisser/déposer, autocomplétion, formulaires avancés...

<http://jqueryui.com/>

TRAVAUX PRATIQUES 8